2012-04-26

Improving Search In Genetic Algorithms Through Instinct-Based Mating Strategies

Thiago S. Quirino
University of Miami, thiago.quirino@noaa.gov

UNIVERSITY OF MIAMI

IMPROVING SEARCH IN GENETIC ALGORITHMS THROUGH INSTINCT-BASED MATING STRATEGIES

By

Thiago S. Quirino

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2012

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

IMPROVING SEARCH IN GENETIC ALGORITHMS THROUGH INSTINCT-BASED MATING STRATEGIES

Thiago S. Quirino

Approved:

Miroslav Kubat, Ph.D. Associate Professor, Electrical and Computer Engineering	Terri A. Scandura, Ph.D. Dean of the Graduate School
Kamal Premaratne, Ph.D. Professor of Electrical and Computer Engineering	Michael Scordilis, Ph.D. Associate Professor of Electrical and Computer Engineering
Nigel M. John, Ph.D. Lecturer of Electrical and Computer Engineering	S. G. Gopalakrishnan, Ph.D. Scientist, National Oceanic and Atmospheric Administration, Miami, Florida

QUIRINO, THIAGO S. (Ph.D., Electrical and Computer Engineering)
Improving Search In Genetic Algorithms Through (May 2012)
Instinct-Based Mating Strategies

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Miroslav Kubat, Ph.D.
No. of pages in text. (256)

The Genetic Algorithm (GA) is a blueprint for writing computer programs capable of solving search and optimization problems. The GA blueprint describes an iterative search process that seeks to improve the quality of an initial random set of solutions (known as a "population of specimens") with respect to some user-defined optimization criteria. All of the components of this iterative search process mimic Darwinian biological evolutionary processes such as mating, recombination, mutation, and survival of the fittest. Thus, the GA is an evolutionary approach to search and optimization.

The GA has been applied to thousands of research and industrial applications across numerous domains of science. Due to its success and popularity, researchers have attempted to improve various aspects of the GA search process over the years. However, the impact of the *mating strategy*, which determines how existing solutions to a problem are paired during the genetic search process to generate new and better solutions, has so far been neglected in the rich and vast GA literature.

The long-standing *conventional mating strategy*, which has been used for decades in implementations of the GA, is based on the random selection of mating partners. This paradigm has known issues. First, due to its stochastic nature, the conventional

mating strategy does not guarantee that the solutions paired to generate new solutions have orthogonal (or independent) information content. This leads the GA to under-utilize the information available for the genetic search process. Second, the conventional mating strategy promotes highly-fit solutions to mate more often than others. These highly-fit solutions eventually *overrun* the entire set of solutions during the genetic search process, leading to issues of *premature convergence* to suboptimal solutions.

The goal of this work is to show that the GA search process can be expedited, with the quality of its solutions improved, by replacing the conventional mating strategy with more sophisticated ones that are inspired from the Darwinian principle of "opposites-attract." The mating strategies proposed in this work improve over the conventional mating strategy, in which specimens (or solutions) have no *mating choice*, by "endowing" specimens in the GA population with *mating-instincts* that promote a more diverse pairing of solutions. The end result is an improved utilization of the information available for the genetic search process to discover new and better solutions and, consequently, a more efficient *sampling* of the solution search space of complex optimization problems.

A total of five novel "instinct-based" mating strategies are proposed in this work. Four of these mating strategies are designed as single-population GA mating strategies, and one as a multi-population GA mating strategy. The five proposed mating strategies are collectively referred to by the acronym *IM-GA*, which stands for Instinct-Based Mating in Genetic Algorithms. To measure the effectiveness of the proposed mating strategies, they were tested on two well-known, complex optimization

problems from the domain of supervised classification: 1) the 1-NN Tuning problem ("Testbed Problem 1"), and 2) the Optimal Decision Forests problem ("Testbed problem 2"). The 1-NN Tuning problem involves the search for optimal subsets of examples and attributes that simultaneously maximize the accuracy and minimizes the classification costs of the 1-NN classifier. The Optimal Decision Forests problem involves the search for highly-accurate and compact decision tree classifiers that can be grouped into ensembles to simultaneously maximize their combined voting classification accuracy while minimizing the overall classification costs.

Testbed Problem 1 was solved by an improved implementation of a well-known GA, the *RK-GA* (hereinafter "baseline *RK-GA*$_0$"). Testbed Problem 2 was solved by a novel GA that was implemented for the purposes of this research, the baseline *TM-GA*$_0$. Rigorous experiments were performed to evaluate the performances of these two GAs both *with* and *without* the proposed *IM-GA* "instinct-based" mating strategies. The various data sets used in the experiments were taken from the well-known UCI Machine Learning Repository.

The experimental results indicated a statistically significant increase in the search speed of the GA when the *IM-GA* mating strategies were applied to the two chosen testbed problems. Furthermore, this increased search speed did not come at the cost of the quality of the discovered solutions and required only negligible additional computational overhead. **Therefore, this work shows that more sophisticated mating strategies can indeed improve the genetic search process over the conventional mating strategy**.

To the author's best knowledge, no other work has so far been developed that combines both the single-population and multi-population versions of the GA with the concept of "instinct-based" mating. Furthermore, the fact that the rich GA literature has devoted less attention to mating strategies than what is deserved gives this research work a "pioneering flavor."

To my wonderful parents, Paulo and Gloria, who dedicated their lives to my education. To my beautiful wife, Rose, who has supported me unconditionally along the way. And to all my colleagues who have taken on this arduous, yet enlightening and undoubtedly rewarding, journey of knowledge alongside me.

Acknowledgements

I am grateful to Dr. Miroslav Kubat, my professor, advisor and mentor, for the incredible support, direction, and patience he has provided me throughout my studies. I learned about the fields of Machine Learning, Evolutionary Computation, and Data Mining while under Dr. Kubat's tutelage. His insightful suggestions have made this work possible. I have learned many things from Dr. Kubat through his experience as a seasoned scientist and professor and look forward to working with Dr. Kubat for many years to come.

I also thank my committee members, Dr. Kamal Premaratne, Dr. Michael Scordilis, Dr. John Nigel, and Dr. Gopalakrishnan Sundararaman, for providing useful insights that enabled me to improve my work and presentation. Having my work critiqued by such knowledgeable individuals, all of which have been my professors and advisors in various courses and projects throughout my decade-long studies at the University of Miami, has been an incredible experience. I am honored and humbled to partake in this experience.

I am grateful to the University of Miami and, in particular, to the faculty of the Electrical and Computer Engineering Department for all their efforts in providing a challenging and stimulating learning environment and encouraging me all along the

way. I thank all my professors for sharing with me, in such an inspiring fashion, their invaluable scientific knowledge and research experience.

I also thank Dr. Gopalakrishnan Sundararaman ("Gopal"), and all my colleagues at NOAA, including Dr. Frank Marks (NOAA's Hurricane Research Division Director), Dr. Xuejin Zhang, Joe D. Griffin, and William P. Barry, for their continued support and encouragement. I have learned so much from each of you in the past few years, and I am forever grateful that you have shared your experience with me. I look forward to working alongside you toward achieving NOAA's goals for many years to come.

Finally, I thank my wonderful family for their unconditional support. I thank my beautiful wife Rose for her expert editing skills and moral support. Rose's dedication, brilliance and inner-strength inspire me daily. I thank my parents, Paulo and Gloria, for always believing in me and for making my education a priority in their lives for the past three decades. And above all, I thank God for blessing me with a wonderful family, perfect health, an incredible group of friends both at school and work, and for the opportunity to live in a country where I can freely pursue my dreams. I love you all so very much.

Thiago S. Quirino

University of Miami

May 2012

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The Genetic Algorithm (GA) is a popular approach to solving search and optimization problems. The GA is a blueprint for writing programs that are capable of finding good solutions to complex search and optimization problems under constrained computational requirements (Holland 1975). The GA blueprint describes an iterative optimization process. This process operates on a (initially) random set of solutions to a problem (because it is assumed that optimal solutions are unknown) and improves these solutions over a great number of iterations guided by some *user-defined* optimization criteria.

A program written to implement the GA blueprint is known as an "implementation of the GA".[1] Such programs will have an "evolutionary flavor" to them. That is because the way solutions to optimization problems are represented in memory, as well as how the operations applied iteratively to existing solutions to generate better solutions are implemented, all mimic Darwinian biological evolutionary process such

[1] An implementation of the GA is a program written to implement the iterative search process described by the GA blueprint. This implementation should have: 1) a representation of solutions to an optimization problem known as *specimens*, 2) a process to randomly initialize the specimens, and 3) the implementation of an iterative process that improves the initially random solutions through functions, or operators, that mimic the Darwinian evolutionary processes of mating, recombination, mutation, and survival of the fitness. See Section 3.2 for a detailed overview of the GA.

as mating (the paring of existing solutions to "mate"), recombination (the exchange of genetic information between paired solutions to generate new and improved ones), mutation (random distortions applied to the genetic information of newly generated solutions), and survival of the fittest (solutions promising more favorable optimization of a problem "survive" while all others are "killed-off").[2]

Due to its immense generality, the GA has been successfully applied to numerous search and optimization problems across various domains of science. Over the years, researchers have "pushed the envelope" in applying the GA to numerous challenging, real-world optimization problems such as the automated computer design of industrial equipment, electronic circuits, trading systems, water distribution systems, turbines, wings, medicines, job scheduling systems, routing systems, and control systems, among numerous others problems (Karr and Freeman 1998; Haupt and Haupt 2004; Popescu, Popescu, and Mastorakis 2009), and in attempting to improve many aspects of the GA iterative search process.

The mating strategy (or "mating") is a key component of the genetic search process. Mating determines how existing solutions to a problem are paired with the goal of exchanging information to generate *better* solutions (e.g. mimicking Darwinian natural adaptation). Unfortunately, the impact of the mating strategy on the performance of genetic search has so far been neglected in the GA literature (Quirino and Kubat 2010). While numerous previous research works have sought to optimize genetic search by improving the other various components of the GA iterative process

[2]The implementations of the GA processes of mating, recombination, mutation, and survival as functions are referred to as *operators* (i.e. the recombination operator and the mutation operator).

(e.g. initialization, recombination, and survival) (Rozsypal and Kubat 2003; Ishibuchi and Nakashima 2000; Ho, Liu, and Liu 2002), research attempts to improve mating in the GA is still an area of scarce publications (Quirino and Kubat 2010).

There are two main reasons for the lack of research interest in optimizing the mating strategy in genetic search. The primary reason is the existence of a long-standing, *conventional mating strategy* which can be very easily used in practically all implementations of the GA. This conventional mating strategy, which has been successfully applied for decades in numerous applications of the GA, does not require any domain-specific knowledge (i.e. it is *problem-independent*) and simply relies on probabilistic distributions, built based on the *user-defined* fitness-function,[3] to randomly pair existing solutions to generate new solutions (or recombination).

The second reason for the lack of research interest on the optimization of mating in genetic search is that improving the mating process requires the adoption of *problem-dependent* parameters. Performing more "informed" pairing of solutions in the GA requires problem-dependent optimization criteria to be considered during the mating process. The optimization criteria must somehow dictate how existing solutions should be optimally paired in a manner that maximizes their potential to generate better solutions that optimize the specific objectives of the problem under investigation. Optimizing mating in genetic search is a problem-dependent task. In-

[3]In Genetic Algorithms, the fitness-function is a *user-defined* function that measures the quality of the solutions generated through genetic search for a specific search or optimization problem. The genetic search seeks to find solutions that optimize the fitness-function. Hence, the definition of the fitness-function is a key issue in implementations of the GA. The fitness-function guides the genetic search by pointing out which solutions are more favorable than others. The terms of the fitness-functions reflect the different optimization objectives of a search problem, e.g. when using a GA to optimize the accuracy of a classifier in a supervised learning problem, the fitness-function may be defined as the accuracy of the GA-generated classifiers.

tuitively, this complicates both the design and implementation of the GA mating process. As a result, many researchers who are primarily interested on using the GA as a "black-box" optimization tool in some arbitrary problems, will tend to adopt the more conventional and generalized implementations of the GA over more custom ones.

The tendency of the conventional mating strategy in genetic search is to promote the pairing of solutions that promise the optimization of the fitness-function. That is, the pairing of solutions to optimize the *user-defined* objectives of a given problem. At first glance, this natural tendency of the conventional mating strategy seems very adequate. However, conventional mating is known to lead the GA to prematurely converge to suboptimal solutions (Shamir, Saad, and Marom 1993). **The main issue with the conventional mating strategy is that it favors the pairing of more "fit" solutions for mating without regard to the relative diversity of the information content of the paired solutions.** A clear example of this "design flaw" is that under the conventional mating strategy a solution is even allowed to "mate-with-itself" due to the random pairing process. When this process is repeated over many generations, it leads the GA population to be eventually *overrun* by highly-fit solutions. The population ends up dominated by what are essentially "duplicates" of the same highly-fit solution. At this stage, the genetic search ends due to an undesired premature convergence to a generally suboptimal solution.

Understanding how the degenerative loss of information occurs in the GA, and leads the population to be *overrun* by highly-fit solutions, requires examining the *sources* and *sinks* of information in the GA. From the very first iteration, the GA

search process starts loosing small amounts of the information available in its population of solutions through the combined process of recombination and survival. When a new solution is generated through recombination of an arbitrary pair of solutions and then used to subsequently replace another existing solution in the population through *survival of the fittest*, the information that was available in the *replaced* solution is simply lost (or "information sink"). If the replaced solution was very different from the newly generated solution, then a lot of information has been lost. And even in the newly generated solution replaced one of its parent solutions, than at least some information has been lost because a child solution generally contains only partial information from each of its parents, which is the nature of recombination. In summary, the GA starts *converging* to a solution (optimal or not) from the very beginning of its run. While different operators, such as the mutation operator, are introduced to retard the effects of information loss in the GA, their impact is somewhat limited. For example, using high mutation rates to inject new information into the population (or "information source") can lead the genetic search to essentially mimic random search, since newly generated solutions would be very different from their parents (i.e. no Darwinian adaptation would take place). On the other hand, low mutation rates cannot efficiently inject new information into the population (Rozsypal and Kubat 2003; Shamir, Saad, and Marom 1993).

What the above discussion alludes to is that improving genetic search requires the adoption of mechanisms capable of slowing-down the loss of information that naturally occurs in the genetic search process. This is achieved by optimizing the use of the information available in the existing solutions to a problem to generate new and

better solutions in each iteration of the GA. Slowing down the information loss helps prevent premature convergence to sub-optimal solutions by retaining more diversity of information in the *solution pool*. This consequently improves the ability of the GA to more effectively sample the solution space of a problem and find better solutions. Dealing with information loss rate in genetic search is a critical consideration to improving the GA.

Mating is a key component of the genetic search process that can be optimized to reduce information loss in the GA. That is because, from among the various genetic operators, the mating strategy is the one which determines how information is used to generate new solutions. The mating process can be optimized in the GA by abandoning the conventional mating strategy, which exacerbates the information loss by leading highly-fit solutions to overrun the GA population, and adopting better mating strategies that promote more diverse pairing of solutions. Moreover, promoting diverse pairing of solutions entails that solutions paired for recombination should "complement" each other according to some problem-dependent criteria. That is because paired solutions that "complement" each other according to problem-dependent criteria must posses orthogonal (or independent) information contents relative to each other. For example, a pair of decision tree classifiers that tend to misclassify different testing examples, where the classification error is a problem-dependent criteria in the domain of supervised learning, must have been trained with different input parameters (i.e. different subsets of examples and attributes). That is, the behavior of different solutions with respect to a problem-dependent criteria reflects their information content relative to each other.

In addition, promoting diverse pairing of solutions in the sense described above would also allow the GA to more effectively sample the solution spaces of optimization problems in search for new solutions that improve over existing ones. This may even lead the GA to converge *faster* to optimal solutions. Because of the stochastic (random) nature of conventional mating, and also because of its lack of consideration for problem-dependent optimization criteria, conventional mating cannot guarantee that solutions are optimally paired in a way that they somehow complement each other along some problem-dependent optimization criteria. In other words, conventional mating does not make optimal use of the information available in the GA population of solutions in order to optimize genetic search.

The goal of this work is to show that more sophisticated mating strategies can indeed optimize genetic search and lead the GA to *faster convergence* without impacting the quality of its generated solutions. To this end, a total of five (5) novel "instinct-based" mating strategies are proposed in this work to improve over the long-standing, conventional mating strategy where specimens have no mating choice (described above). The five proposed mating strategies "endow" specimens in the GA population with "mating instincts" that guide them in their selection of mating partners that "complement" their own abilities to optimize some problem-dependent optimization criteria. As a result, the proposed mating strategies promote more diverse pairing of specimens having orthogonal (or independent) information content. **This process optimizes the genetic search by improving the utilization of the information available in the GA population for the generation of better solutions, slowing down the information loss inherent to the genetic search**

process, and allowing the GA to more effectively and diversely sample the search spaces of optimization problems.

The concept of "instinct-based" mating proposed in this work was inspired from the natural Darwinian principle of "opposites-attract" that is so common in nature. Darwin pointed out in his *Descent of Man (1859)* that some animals give preference to mating partners whose abilities "complement" their own. He mentioned as an example that in those bird species where nest-building is a male responsibility, females sometimes tend to instinctually test the nest-building ability of potential mates. Furthermore, more recent research has indicated that human females tend to prefer partners with a dissimilar genetic makeup (ScienceDaily 2009). Surprisingly, these concepts have so far been neglected in applications of Genetic Algorithms. The fact that the rich GA literature has devoted to mating strategies less attention than what is deserved gives this work a "pioneering flavor."

The five "instinct-based" mating strategies proposed in this work are collectively referred to by the acronym *IM-GA*, which stands for Instinct-Based Mating in Genetic Algorithms. Four of these strategies are designed as single-population mating strategies[4] and one as multi-population mating strategy.[5] To evaluate the impact of these proposed mating strategies on the performance of the GA, two well-known testbed problems were chosen from the domain of supervised classification. The chosen

[4]The single-population GA uses one single population of specimens, or a single pool of potential solutions, to solve optimization problems.

[5]The multi-population GA uses multiple populations of specimens, or multiple pools of potential solutions, to solve an optimization problem. Generally, different populations evolve independently with periodic interbreeding through exchange of specimens among the population. The design of a multi-population GA is more complex than that of a single-population GA because of the added complexities of choosing from which populations *parent* specimens should come from and into which populations *children* specimens should be inserted to (i.e. migration).

testbed problems were: 1) the 1-NN Tuning Problem (hereinafter "Testbed Problem 1") and 2) the Optimal Decision Forests Problem (hereinafter "Testbed Problem 2"). Both testbed problems are complex optimization problems categorized as NP-hard (Quirino and Kubat 2010) and NP-complete (Chandra and Yao ; Okun, Valentini, and Re 2011) problems, respectively (see Section 2.2). This means that no algorithms capable of solving these problems in polynomial time[6] have yet been developed. Both Testbed Problems 1 and 2 are described in detail in Chapter 3.

Since mating is only a component of the GA search process, two GAs were implemented to facilitate the application of the proposed *IM-GA* mating strategies to the optimization of Testbed Problems 1 and 2, respectively. To apply the *IM-GA* mating strategies to the optimization of Testbed Problem 1, a well-known GA called "*RK-GA*" was first re-implemented from the literature to find optimal 1-NN classifiers (Rozsypal and Kubat 2003). Moreover, the fitness-function of *RK-GA* was improved and the resulting GA, referred to as the baseline $RK\text{-}GA_0$, was then used as the baseline for all experiments with Testbed Problem 1. Similarly, to apply the *IM-GA* mating strategies to the optimization of Testbed Problem 2, a novel GA called the baseline $TM\text{-}GA_0$ was designed and implemented specifically for this research to discovery optimal ensembles of decision tree classifiers (or optimal decision forests).

Note that both the baselines $RK\text{-}GA_0$ and the $TM\text{-}GA_0$ were designed to work with the conventional mating strategy. Therefore, it was imperative to first evaluate their performances *without* "instinct-based" mating to ensure that they were not

[6]Search and optimization problems that can be solved in the polynomial order of computational time-complexity are considered *cheap* problems.

strawman, or weak methods that would be easily improved by the application of the *IM-GA* "instinct-based" mating strategies. To this end, the performance of both *RK-GA*$_0$ and *TM-GA*$_0$ using the conventional mating strategy was evaluated first through various experiments using 24 benchmark data sets from the UCI Machine Learning Repository (Newman and Merz 1998). These experiments compared the performance of the baselines *RK-GA*$_0$ and *TM-GA*$_0$ against those of state-of-the-art techniques designed to solve Testbed Problems 1 and 2, respectively. The experiments yielded a good set of baseline results for the performance of the baselines *RK-GA*$_0$ and *TM-GA*$_0$ using the conventional mating strategy. Then, the five proposed *IM-GA* mating strategies were introduced into the baselines *RK-GA*$_0$ and *TM-GA*$_0$, replacing the conventional mating strategy with "instinct-based" ones, and the experiments were rerun. Finally, the two sets of experimental results attained, (1) with conventional mating, and (2) with "instinct-based" mating, were compared to evaluate the impact of the *IM-GA* mating strategies on the performances of *RK-GA*$_0$ and *TM-GA*$_0$.

The experimental results presented in Chapter 5 confirmed that the proposed *IM-GA* "instinct-based" mating strategies can *accelerate* the GA without impacting the quality of the generated solutions. Furthermore, the results also indicated that the faster convergence attained required only negligible additional GA computational time. **Thus, the conclusion is that genetic search can indeed be optimized by the use of more sophisticated "instinct-based" mating strategies in the GA mating process.**

1.1 Motivation and Research Objectives

The major motivation of this work is the fact that the impact of the mating strategy on the performance of genetic search has been neglected in the GA literature and in real-world applications of the GA. Over the years, researchers have proposed many improvements to the various components of the GA search process, components other than the mating strategy (Rozsypal and Kubat 2003; Ishibuchi and Nakashima 2000; Ho, Liu, and Liu 2002).

For decades, a single paradigm for mating strategy has dominated implementations of the GA across various fields of science: random selection of mating partners according to probabilistic distributions based on the fitness-function. This paradigm has known issues (Shamir, Saad, and Marom 1993). First, it does not guarantee that GA-generated solutions are paired based on their abilities to "complement" each other in maximizing some problem-dependent optimization criteria. This is due to the purely random selection of the mating partners promoted by the conventional mating strategy. As a result of this "design flaw", the conventional mating strategy can severely under-utilize the diversity of information available in the GA population for the generation of new and better solutions in each iteration of the genetic search. Conventional mating inadvertently guides the genetic search into allowing highly-fit specimens to *overrun* the GA population. This issue is known as *premature convergence* to suboptimal solutions. Mitigating this issue, by proposing newer and more sophisticated mating strategies, is a major motivation this work.

Another major motivation of this work is to encourage other researchers to extend the proposed ideas of "instinct-based" mating to speed up genetic search in their own applications. The GA is one of the most competitive and frequently adopted heuristic approaches to search and optimization, having been employed in countless real-world industrial problems having large and complex search spaces (Safe, Carballido, Ponzoni, and Brignole 2004; Man, Tang, and Kwong 1996; Stender 1993). In some industrial applications of the GA, a single *fitness evaluation* can require anywhere from minutes to days to complete (Albert 2002). Those are prohibitive costs! Applications of the GA having prohibitively costly fitness evaluations can greatly benefit from an improved mating process that requires less fitness evaluations to discover comparably, or perhaps even better, solutions. This is a major motivation of this work, to show that the GA mating process can be improved by more sophisticated mating strategies that promote more diverse and efficient sampling of the solution search space of optimization problems.

The primary set of research objectives of this work is concerned with speeding up genetic search through improved mating strategies. The secondary set of objectives is concerned with the development and/or improvement of the *instruments* needed to both achieve and evaluate the impact of the primary set of objectives. In total, the research objectives of this work are five fold:

- Research Objective 1: To design and implement five (5) novel "instinct-based" mating strategies that improve over the conventional mating strategy, lead the

GA to faster convergence without impacting the quality of generated solutions, and require minimal additional computational overhead.

- Research Objective 2: To show that the proposed "instinct-based" mating strategies can be applied to optimize genetic search in two complex testbed problems from the domain of supervised classification.

- Research Objective 3: To propose principles that simplify to recognition of useful "mating-instincts" in different optimization problems in order to facilitate the extension of the proposed ideas to other real-world applications of the GA.

- Research Objective 4: To design and implement a new GA capable of discovering both accurate and compact ensembles of decision trees (decision forests). The goal is to improve the *interpretability* of decision forests, which has been lost in applications of the classical ensemble learning methods (e.g. Bagging, AdaBoost, and Random Forest). The classical methods require large ensembles of large decision trees to work properly from a statistical point of view (reducing classifier variance through averaging), which harms the interpretability of the decision forests.

- Research Objective 5: To improve the original *RK-GA*, which was designed to discover optimal 1-NN classifiers (i.e. accurate and compact), by modifying its fitness-function with automatically-set weight parameters based on data set relevant features. The goal is to eliminate the need for manual weight parameter tuning in the fitness-function of *RK-GA*.

1.2 Research Challenges

Various challenges were involved in completing each of the research objectives described in Section 1.1. The research challenges varied from the design of new mating strategies capable of optimizing genetic search over the long-standing, conventional mating strategy, to the implementation of GAs capable of solving challenging optimization problems, such as the 1-NN Tuning and Optimal Decision Forests problems.

One of the main challenges involved in this work was the definition of what constitutes useful "mating-instincts" for optimization of the GA mating process. Thorough research of the GA literature on the principles of mating in the GA, as well as a thorough analysis of the genetic search process itself, revealed that the definition of useful "mating-instincts" in the GA was problem-dependent: specimens in the GA population should tend to mate with those that "complement" them on their own abilities to optimize the objectives of the problem under investigation. It took time to realize that "mating-instincts" defined using the above criteria should theoretically promote the pairing of specimens having orthogonal (or independent) information, which is reflected on their "complementary" behavior toward the optimization objectives in question. For example, take a pair of specimens that represent two supervised learning classifiers. If these classifiers tend to misclassify different examples in a testing set, then they must have been built with somewhat orthogonal input parameters. This is a deterministic cause-effect relationship reflected in the orthogonal classification behavior of the classifiers. Hence, "mating-instincts" that are defined according to the above criteria have the potential to theoretically promote more diverse pairing

of solutions in the GA and consequently more diverse sampling of the search space of optimization problems during genetic search.

The second research involved the actual selection of testbed problems. The choice of testbed problem was critical because it defined the "mating-instincts." The chosen testbed problems should be well-known optimization problems that have been thoroughly investigated, and thus well-understood, by the research community. They should preferably also have real-world applicability, in order to inspire other researchers dealing with complex, computational intensive optimization problems to extend the proposed ideas and benefit in their own research. Furthermore, the chosen testbed problems should have optimization objectives that translated into an intuitive definition of useful "mating-instincts", in order to facilitate the implementation and evaluation of the proposed "instinct-based" mating ideas. The chosen testbed problems were the 1-NN Tuning problem detailed in Section 3.3 and the Optimal Decision Forests problem detailed in Section 3.4.

Another challenge involved the implementation of new GAs to facilitate the evaluation of the proposed "instinct-based" mating strategies on the performance of genetic search applied to the two chosen testbed problems. Recall that mating is only *one* component of the genetic search process, thus, GAs had to be implemented to solve the two chosen testbed problems. Moreover, these GAs had to be designed to work well (i.e. discover good solutions) with the conventional mating strategy for two reasons: 1) in order to generate a good set of baseline experimental results using the convention mating strategy to be compared to those results attained with "instinct-based" mating, and 2) to ensure that any improvement brought forth by

"instinct-based" mating on the performance of the GA was a *real* improvement, and not due to mere chance. To this end, the well-known *RK-GA* (Rozsypal and Kubat 2003) was re-implemented from the GA literature, and further improved upon, for use as the baseline GA (hereinafter "$RK\text{-}GA_0$") for all experiments with "instinct-based" mating in Testbed Problem 1.

However, applying "instinct-based" mating to Testbed Problem 2 required the design of a new GA, hereinafter $TM\text{-}GA_0$. The main reason for this choice, instead of simply picking an existing GA from the literature, was to extend the framework of the original *RK-GA* which is applicable to *single classifier* optimization, to be also applicable to the optimization of *ensembles of classifiers*. The main benefit of building a new GA by extending the original *RK-GA*'s framework stems from *RK-GA*'s pioneering use of a variable-length, value-encoded specimen chromosome encoding scheme. This chromosome-encoding scheme was found to significantly reduce the GA's computational costs when applied to large data sets. Thus, intuitively, given that Testbed Problem 2 involves the simultaneous optimization of multiple decision tree classifiers in an ensemble, the specimen chromosome-encoding scheme pioneered by *RK-GA* was a prime design choice for use in a new GA capable of handling large data sets efficiently.

There were further research challenges associated with the design of $TM\text{-}GA_0$. For example, in *RK-GA* a specimen represented a single 1-NN classifier. In $TM\text{-}GA_0$, however, a specimen represents multiple decision tree classifiers (induced with the C4.5 Decision Trees program) in an ensemble of variable size. Thus, the design of $TM\text{-}GA_0$'s recombination process posed two new challenges: 1) how to best pair

the actual decision trees composing paired decision forests for recombination, and 2) how to pair the newly generated decision trees into new decision forests. The design choice for *TM-GA*$_0$'s recombination process that tackled these challenge is detailed in Section 4.5.1.

Another major challenge in the design of *TM-GA*$_0$ stemmed from the fact that the C4.5 Decision Trees program tends to induce decision trees that over-fit their training sets (Hall and Smith 1998; Ho 1998a). In this case, when the C4.5 Decision Trees classifier is optimized by a powerful optimization tool such as the GA, the training set over-fitting potential can literally *shoot-through-the-roof*! The main reason for the explosive over-fitting behavior is that the classification accuracy of the GA-generated decision forests is generally measured on the training set itself during genetic search as a term in the fitness-function. Hence, the GA can be easily biased toward favoring the highly over-fit decision trees and corresponding ensembles that are discovered during genetic search. In the design of *TM-GA*$_0$, this issue was dealt with by introducing a novel ensemble diversity measure into *TM-GA*$_0$'s fitness-function that prevented specimens (or decision forests) from being penalized for the classification errors of its individual decision tree classifiers as long as those errors did not impact the overall ensemble accuracy. Unfortunately, the existing ensemble diversity measures described in the literature were not designed for use in the process of building individual classifiers, but rather only on the process of grouping pre-built classifiers into ensembles (Kuncheva 2003). The novel ensemble diversity measure was dubbed the "triple-fault" measure because it measures the "complementary" classification behavior of every possible *sub-ensemble* of "3" decision trees from among all

the decision trees making up a decision forest. The design of the novel "triplet-fault" diversity measure is detailed in Section 4.5.1.

Similarly, the design of a fitness-function for *TM-GA*$_0$ also posed a very challenging task because the fitness-function needed to include terms that reflected both (1) *ensemble-level* optimization objectives as well as (2) *decision trees-level* optimization objectives. The detailed explanation on how this challenge was tackled in *TM-GA*$_0$ is presented in Section 4.5.1.

Yet another challenge encountered in the design of *TM-GA*$_0$ was the choice of the stopping criteria, which determines when better solutions can no longer be discovered by the genetic search. Choosing a stopping criteria is a common challenge in the design of GAs (X. Yu 2010; Safe, Carballido, Ponzoni, and Brignole 2004). For example, *TM-GA*$_0$ cannot determine when the *optimal* decision forest has been discovered because it does not know when a forest is optimal enough. This issue is further complicated by the fact that the genetic search can sometimes become *trapped* in a solution corresponding to some local optimal in the fitness-function. While the genetic search might remain *trapped* there indefinitely, it can sometimes escape to better solutions after an arbitrary number of iterations. To tackle these challenge, *TM-GA*$_0$ was allowed to run for an ample amount of iterations before it determined that better solutions could not be discovered. The choice of stopping criteria for *TM-GA*$_0$ is detailed in Section 4.5.1.

Finally, another challenge involved in the design of *TM-GA*$_0$ as well as in the evaluation of the impact of "instinct-based" mating under Testbed Problem 2, was the **significant computational power required** by *TM-GA*$_0$. As discussed in

Section 4.5.2, the computational time complexity of *TM-GA*$_0$ is dominated by the decision tree induction process (i.e. the time required by the C4.5 Decision Trees program to induce decision trees). On average, during each experiment with data sets from the UCI Machine Learning Repository, *TM-GA*$_0$ generated approximately 300 decision trees and discovered 60 new decision forests per iteration of the GA. Furthermore, *TM-GA*$_0$ runs lasted from some few hundred iterations for smaller UCI data sets to as many as $10,000+$ iterations for large and complex UCI data sets (i.e. those having large number of examples, attributes, and classes). While *TM-GA*$_0$ was implemented as a program that made use of parallelization to build multiple decision trees simultaneously during the recombination process, the actual *off-the-shelf* implementation of the C4.5 Decision Trees program that was used to induce decision trees was not internally parallelized to more efficiently induce decision trees. Parallelization of the C4.5 Decision Trees program is a complex task because numeric attributes have to be re-evaluated at different levels of the decision tree induction process, which hinders parallelization. As a consequence, the speed of the tree induction process slowed down significantly when large data sets were used. Moreover, the process of building multiple decision trees in parallel also requires significant amount of memory to store the induced decision tree models. This requirement is inherent from the computational storage cost of the decision tree induction process, a cost which increases proportionally to the size of the training data set, as detailed in Section 4.5.2.

In conclusion, various challenges were encountered during the execution of this research. These challenges attest to the *thought-provoking* nature of the novel work presented here.

1.3 Summary of Contributions

The work presented here is likely to produce new and useful insights for Machine Learning practitioners engaged in applications of Genetic Algorithms. There is the potential to benefit for researchers across various domains of science who rely on genetic search for the optimization of problems in their own research fields.

1.3.1 Faster Genetic Search with Minimal Computational Overhead

One major contribution of this work to the field of Machine Learning are the five (5) proposed *IM-GA* "instinct-based" mating strategies, which replace the long-standing, conventional mating strategy, which based on random selection of mating partners. The *IM-GA* mating strategies are more sophisticated strategies designed to mitigate the issue that is premature convergence of the GA to suboptimal solutions generated by the conventional mating strategy. The major advantages of the *IM-GA* mating strategies over the conventional mating strategy are:

1. The *IM-GA* mating strategies promote statistically significant increased convergence speed of the GA when compared to the conventional mating strategy when applied to two complex optimization problems from the domain of supervised classification;

2. The *IM-GA* mating strategies do not impact the quality of the GA-generated solutions when compared to the conventional mating strategy. Instead, the quality of the generated solutions are often improved, and;

3. The *IM-GA* mating strategies require only minimal additional computational overhead when used to optimize the GA mating process.

In real-world terms, this means that applications of the GA that replace the conventional mating strategy in favor of the *IM-GA* mating strategies in their genetic search process, should discover *just as good* or perhaps even better solutions, but in significantly shorter periods of computational time. This benefit is confirmed in this research from the application of the *IM-GA* mating strategies in the complex, real-world problems of optimizing the 1-NN classifier (Testbed Problem 1) and building optimal ensembles of decision trees (Testbed Problem 2). **The results presented in Chapter 5 confirmed that the *IM-GA* mating strategies indeed *accelerated* the search speeds of both the baselines *RK-GA*$_0$ and *TM-GA*$_0$ when compared to the conventional mating strategy. Moreover, the increased search speed did not come at the cost of the quality of the GA-generated solutions nor additional computational overhead.**

1.3.2 Improved Genetic Search In Various Optimization Problems

Another major contribution of this work to the field of Machine Learning is also reflected as another major advantages of the proposed *IM-GA* mating strategies: their applicability to numerous optimization problems in the domain of supervised classification. This feature is due to *IM-GA*'s adoption of measures that are *problem domain* dependent (as opposed to simply *problem* dependent) to implement "mating-

instincts" in the GA mating process. The *IM-GA* mating strategies treat the classifier being optimized in a problem as a "black-box." Moreover, the "mating-instincts" are implemented from measures relevant to the input parameters used to build the classifiers (i.e. the example and attribute sets) and to the output (or response) of the classifiers (i.e. the classification error on different testing examples). Hence, no custom *classifier dependent* measures are required to improve genetic search through the use of the *IM-GA* mating strategies. This feature was reflected in this work by the simple application of the five proposed *IM-GA* mating strategies to the two chosen testbed optimization problems from the domain of supervised classification. Because of their adaptability, the applications of the proposed *IM-GA* mating strategies to other problems in the domain of supervised classification are boundless.

1.3.3 A Novel GA-Based Approach To Building Decision Forests

Another contribution of this work to the field of Machine Learning is the novel GA, the *TM-GA*$_0$, which was designed and implemented in this work to facilitate the application of the proposed *IM-GA* mating strategies in Testbed problem 2; the Optimal Decision Forests problem. As discussed in Chapter 5, the decision forests generated by *TM-GA*$_0$ were as accurate or better, as well as significantly more compact, than those generated by the state-of-the-art, non-GA, ensemble learning approaches of Bagging, AdaBoost, and Random Forest when applied to UCI data sets. The experimental results revealed that the decision forests generated by *TM-GA*$_0$ have three main advantages over those generated by the non-GA approaches:

1. Enhanced interpretability: *TM-GA*$_0$'s decision forests are both accurate and compact (i.e. optimal), making results much more easy to interpret when compared to the large forests generated by the state-of-the-art, non GA approaches;

2. Low Memory Requirements: Due to their compactness, they consume significantly less memory than those generated by classical approaches. They can be efficiently converted to a set of decision rules, stored in memory-constrained micro-chips, and used in real-world applications, and;

3. Fast Classification Response: Also due to their compactness, their fast response time (i.e. average number of tests required to classify an example) makes them useful in real-world applications where real-time response is critical, and otherwise large forests could not be practically used.

The optimal decision forests (i.e accurate and compact) generated by *TM-GA*$_0$ are valuable tools for use in many real-world, real-time response demanding, memory constrained applications.

1.3.4 Elimination of Manual Weight Tuning in the Original *RK-GA*'s Fitness-Function

Another major contribution of this work is the modification made to the original *RK-GA*'s fitness-function (see **Equation 4.6** in Section 4.4.1) to eliminate the effort of manual tuning of weight parameter. In the original *RK-GA*, the three weight parameters c_1, c_2, and c_3 were set to 1 to provide the same level of significance to all the terms. This was done due to limited knowledge about the sensitivity of the GA to the different terms of the fitness-function. However, this is not always desirable because

the different terms of *RK-GA*'s fitness-function correspond to different optimization objectives having diverse search space sizes. To eliminate this objection, the fitness-function of the original *RK-GA* was modified in a manner that the weight parameters are set automatically using measures that are data set relevant (see Section 4.4.2). The proposed approach eliminated the effort involved in manually tuning the weight parameters. Furthermore, the new automatically-set weight parameter values were found to actually improve the performance of *RK-GA*. The new *RK-GA*, with the modifications to the weight parameters, was dubbed the baseline $RK\text{-}GA_0$ and used in all experiments performed in this work with Testbed Problem 1, the 1-NN Tuning problem.

1.4 Organization of the Dissertation

The dissertation is organized as follows. Chapter 2 introduces the problem statement followed by a brief overview of the two complex optimization problems from the domain of supervised classification that were chosen as testbeds for the evaluation of proposed ideas. Moreover, the performance criteria applied to the two chosen testbed problems are also reviewed. Chapter 3 surveys existing literature on the application of both GA-based and non-GA-based, state-of-the-art approaches to solving the two chosen testbed problems. Chapter 4 describes the five (5) proposed *IM-GA* mating strategies as well as the GAs that were designed to facilitate the application of *IM-GA* in the two chosen testbed problems, namely, the baselines $RK\text{-}GA_0$ and $TM\text{-}GA_0$. Chapter 5 discusses the extensive experiments that were performed to evaluate the

proposed ideas and presents the experimental results. The conclusion and ideas for future research on "instinct-based" mating is presented in Chapter 6.

CHAPTER 2

Improving the Genetic Algorithm

This chapter describes the research problem addressed by this work, the two testbed problems chosen to evaluate the five (5) proposed *IM-GA* mating strategies, and the performance criteria used.

2.1 Problem Statement

Whether the genetic search can be expedited, with the quality of the solutions improved, and with minimal additional computational overhead required, by more sophisticated mating strategies.

While the long-standing, conventional mating strategy is based on random selection, the mating strategies proposed in this work are based on the Darwinian principle of "opposites-attract" commonly found in nature. In the conventional mating strategy, specimens have no mating choice. In the mating strategies proposed in this work, specimens are "endowed" with "mating instincts" that guide them in their selection of mating partners that "complement" them according to some problem dependent optimization criteria.

The importance of this work lies on the fact that the impact of the mating strategy on the performance of the GA has been neglected in the rich GA literature. This work shows that more can done to optimize genetic search through improved mating strategies that require negligible additional computation overheard.

2.2 Two Testbed Problems

A *testbed problem* is used to evaluate an idea. In this case, identifying testbeds is important because the choice of the testbed determines the design of "mating-instincts" for the proposed *IM-GA* mating strategies. Recall from the discussion in the Introduction that the "mating instincts" proposed in this work are defined according to the unique optimization criteria of a particular problem.

As was discussed in the Introduction, the domain of supervised classification offers numerous optimization problems under which useful "mating-instincts" can be very intuitively defined (see Section 4.1.1 for a more detailed elaboration). In this work, two testbed problems were adopted from the domain of supervised classification to evaluate the performance of the proposed *IM-GA* mating strategies. The chosen testbed problems are:

1. Testbed Problem 1: The 1-NN Tuning problem, and;

2. Testbed Problem 2: The Optimal Decision Forests problem.

Both Testbed Problems 1 and 2 are complex optimization problems (*NP-hard* and *NP-complete*, respectively), for which no algorithms have yet been developed to directly or "quickly" (i.e. in polynomial order of computational time complexity) find

optimal solutions (Quirino and Kubat 2010; Chandra and Yao ; Okun, Valentini, and Re 2011). Finding "good" solutions to these problems is currently a task for heuristic approaches such as the GA. Hence, improving the application of the GA in these problems, as well as designing new and better GAs to solve them (as is accomplished in this work through the design of the novel *TM-GA*$_0$ for Testbed Problem 2) are both useful contributions to the field of Machine Learning. Testbed Problems 1 and 2 are described in the next section.

2.2.1 The 1-NN Tuning Problem (Testbed Problem 1)

The first testbed optimization problem chosen from the domain of supervised classification is the 1-NN Tuning problem, which is more thoroughly elaborated in Section 3.3. The 1-NN Tuning problem consists of the search for *optimal subsets* of examples and attributes from a data set that optimize the classification accuracy of the 1-NN classifier, while minimizing its classification costs (Rozsypal and Kubat 2003).

The 1-NN Tuning problem is an NP-hard problem that has been thoroughly investigated by the research community. Numerous heuristic approaches have been developed over the years to discover good solutions to the 1-NN Tuning Problem. These developed approaches consisted of both GA-based and non-GA-based approaches (Hart 1968; Wilson 1972; E. Cantu-Paz 2004; Ishibuchi and Nakashima 2000; Rozsypal and Kubat 2003; Kuncheva and Jain 1999; Quirino and Kubat 2010). The GA-based approaches seemed to have an advantage over the non-GA-based approaches because they were capable of *simultaneously* optimizing both the example and attribute sets

of a data set. In contrast, non-GA-based approaches attempted to optimize the example and attribute sets *sequentially* (Quirino and Kubat 2010). An overview of various GA-based and non-GA-based approaches to the 1-NN Tuning problem is given in Sections 3.3.1 and 3.3.2.

In order to apply the proposed *IM-GA* "instinct-based" mating strategies to the optimization of Testbed Problem 1, the well-known GA called the "*RK-GA*" (Rozsypal and Kubat 2003) was re-implemented from the literature and further improved. *RK-GA* pioneered in the use of a value-encoded, variable-length specimen chromosome representation that significantly reduced the computational costs associated with the optimization of large data sets. Moreover, *RK-GA* compared favorably to other well-known approaches to the 1-NN Tuning problem. As a result, *RK-GA* is a good baseline GA onto which to implement the proposed *IM-GA* "instinct-based" mating strategies and evaluate their impact on genetic search applied to Testbed Problem 1. The new *RK-GA*, with the improvements developed in this work, is hereinafter referred to as the baseline $RK\text{-}GA_0$.

2.2.2 Optimal Decision Forest Problem (Testbed Problem 2)

The second testbed optimization problem chosen from the domain of supervised classification is the Optimal Decision Forests problem, which is more thoroughly elaborated in Section 3.4. The Optimal Decision Forest problem consists of the search for optimal decision trees (i.e. both highly-accurate and compact decision trees) that when combined into an ensemble (or *decision forest*) are capable of "complementing" each others' classification errors in order to maximize the ensemble classification accuracy

while also minimizing the overall ensemble classification costs (i.e. minimizing the number of tests required to classify unknown examples) (Chandra and Yao).

Finding optimal decision forests requires the simultaneous optimization of multiple objectives: 1) optimizing the example and attribute sets used to induce decision trees (i.e. maximize classification accuracy and minimize the size of the decision trees), and 2) optimizing the grouping of the decision trees into ensembles to attain higher predictive power than any individual decision tree in the ensemble. These optimization tasks make up an NP-complete problem (Hyafil and Rivest 1976; Murthy and Salzberg 1995; Chandra and Yao ; Chikalov 2011).

Building optimal ensembles of classifiers is a popular research topic in Machine Learning because classifier ensembles have been found both empirically and theoretically to outperform single classifiers. According to Thomas Dietterich, a pioneer in ensemble learning research, classifier ensemble learning is a major research direction in Machine Learning (Dietterich 1997). Over the years, both GA-based and non-GA-based approaches have been developed to search for optimal decision forests (Oza and Tumer 2008; Rokach 2010). An overview of representative approaches is given in Sections 3.4.3 and 3.4.4.

In order to apply the proposed *IM-GA* "instinct-based" mating strategies to the optimization of Testbed Problem 2, a novel GA called the "baseline *TM-GA*$_0$" was implemented specifically for the needs of this research. *TM-GA*$_0$ extends the original *RK-GA*'s single classifier optimization framework, which relies on the conventional mating strategy, into a framework for the optimization of *variable sized* ensembles of classifiers. The results of rigorous experiments comparing *TM-GA*$_0$'s performance

to those of state-of-the-art, non-GA-based ensemble learning approaches of Bagging, AdaBoost, and Random Forest are given in Chapter 5. The results show that *TM-GA*$_0$ compared favorably to these state-of-the-art approaches under various performance criteria. As a result, *TM-GA*$_0$ is a good baseline GA onto which to implement the proposed *IM-GA* "instinct-based" mating strategies and evaluate their impact on genetic search applied to Testbed Problem 2.

2.3 Performance Criteria

This section describes the different criteria that were used to evaluate the performance of the GA *with* and *without* the application of the *IM-GA* "instinct-based" mating strategies. *The performance of the GA is measured both by its search speed (defined in the subsequent section) as well as by the quality of its final generated solutions: the 1-NN classifier in Testbed Problem 1 and the decision forest (ensemble of C4.5 Decision Trees) classifier in Testbed Problem 2.* The chosen performance criteria have been commonly used throughout the GA literature (Quirino and Kubat 2010; Lim and Shih 2000; Alkhalid, Chikalov, and Moshkov 2011; Murthy and Salzberg 1995).

Most of the performance criteria used to evaluate the GA under Testbed Problems 1 and 2 are common. These common criteria include:

- Common Criterium 1: The classification accuracy of the GA-generated classifiers, and;
- Common Criterium 2: The "attribute set reduction" ability of the GA (i.e. the ability to remove of noisy/irrelevant attributes from a data set).

Other criteria were also adopted to measures the complexity of the GA-generated classifiers in Testbed Problems 1 and 2. Measuring classifier model complexity is correlated to measuring classification costs, which is a classifier dependent task. That is because different classifiers have different internal models (i.e. a decision tree classifier is represented by test nodes or decision rules while a 1-NN classifier is represented by the raw examples in its training data set).

To measure the complexity of the GA-generated 1-NN classifiers in Testbed problem 1, the "example set reduction" ability of the GA (i.e. removal of noisy/redundant examples from a data set) was used. In addition, the Common Criterium 2 listed above, the "attribute set reduction" ability of the GA, was also used as a measure of the complexity of the 1-NN classifier. Both of these measures are relevant to the complexity of the 1-NN classifier because the classification costs of the 1-NN classifier increase proportionally with respect to the number of examples and attributes in the training data set (Quirino and Kubat 2010). The complexity of a decision forests was measured by *summing-up* the measures from the individual decision trees making up the decision forest.

The performances of the baselines *RK-GA*$_0$ and *TM-GA*$_0$ (using the conventional mating strategy) according to the criteria above were compared to those of existing state-of-the-art approaches for the generation of 1-NN classifier and decision forests, respectively. This was done to ensure that both *RK-GA*$_0$ and *TM-GA*$_0$ are not *weak* GAs that could be easily improved by the application of the five proposed *IM-GA* "instinct-based" mating strategies.

To evaluate the impact of the five *IM-GA* "instinct-based" mating strategies on the performances of *RK-GA*$_0$ (in Testbed Problem 1) and *TM-GA*$_0$ (in Testbed Problem 2), experiments with *RK-GA*$_0$ and *TM-GA*$_0$ were first performed using the conventional mating strategy and the results were collected. Then, the *IM-GA* mating strategies were introduced into both *RK-GA*$_0$ and *TM-GA*$_0$ and the experiments were rerun. The two sets of experimental results were then compared using the chosen performance criteria described in the following sections.

Finally, note that all experiments were performed using various benchmark data sets from acquired from the UCI Machine Learning Repository. For all techniques used in this work, both GA-based and non-GA-based, the experiments with each UCI data set were run as 5-fold cross-validation, repeated 10 times for different seeds of the random number generator. This corresponds to a total of 50 experiments per UCI data set per technique used in this work. The statistical significance of the differences in performances among the various techniques was assessed by the *paired t-test* with 5% confidence level.

2.3.1 Genetic Search Speed

The genetic search speed (or *convergence speed*) is the most important performance criteria used to evaluate the impact of the proposed *IM-GA* "instinct-based" mating strategies on the performances both baselines *RK-GA*$_0$ and *TM-GA*$_0$. The genetic search speed is used to determine whether or not the *IM-GA* "instinct-based" mating indeed lead the GA to *faster convergence*: the ability to discover good solution in a minimum number of generations (or iterations). In the experiments with each UCI

data set, *convergence speed* was defined as the *time* required for the *average specimen accuracy*[7][8] of each GA to reach a certain *convergence target*.

Understanding that the number of generations is inappropriate for measuring *time* (e.g. larger populations need fewer generations), and that CPU-time is highly dependent both on the available computing power as well as the programmer's skills, *time* was measured instead by the number of fitness-function evaluations. *Time* ranged from *t=0*, at the start of the GA run, to *t=total fitness evaluations at the convergence target*. This measure of *time* was used in a previous work, and it avoids biases (Quirino and Kubat 2010).

Additionally, the *convergence target* was set differently for *RK-GA*$_0$ and *TM-GA*$_0$, since each deal with a unique classifier having different susceptibility to over-fitting. For *RK-GA*$_0$, the *convergence target* was set as the value corresponding to the accuracy of the 1-NN classifier (averaged values obtained from multiple cross-validation runs and presented in Table 5.6). However, for *TM-GA*$_0$, the *convergence target* was set as the value corresponding to the *95$_{th}$-percentile* (or 95%) of the accuracy of the C4.5 Decision Trees program (averaged values obtained from multiple cross-validation runs and presented in Table 5.17). The *convergence targets* of *RK-GA*$_0$ and *TM-GA*$_0$ differ because *TM-GA*$_0$ seeks to optimize the C4.5 Decision Trees program, which tends to induce decision trees that very easily over-fit their training sets. In contrast, the 1-NN classifier optimized by *RK-GA*$_0$ is less prone to over-fitting. Hence, the *convergence target* for *TM-GA*$_0$ was chosen to prevent it from *undershooting* the val-

[7]Contrast the top-fitness specimen's accuracy, which is more random.

[8]The average specimen accuracy is measured as the average classification accuracy of the classifiers represented by the GA population

ues given in Table 5.17 since it avoids over-fitting the decision trees induced by the C4.5 Decision Trees program during genetic search.

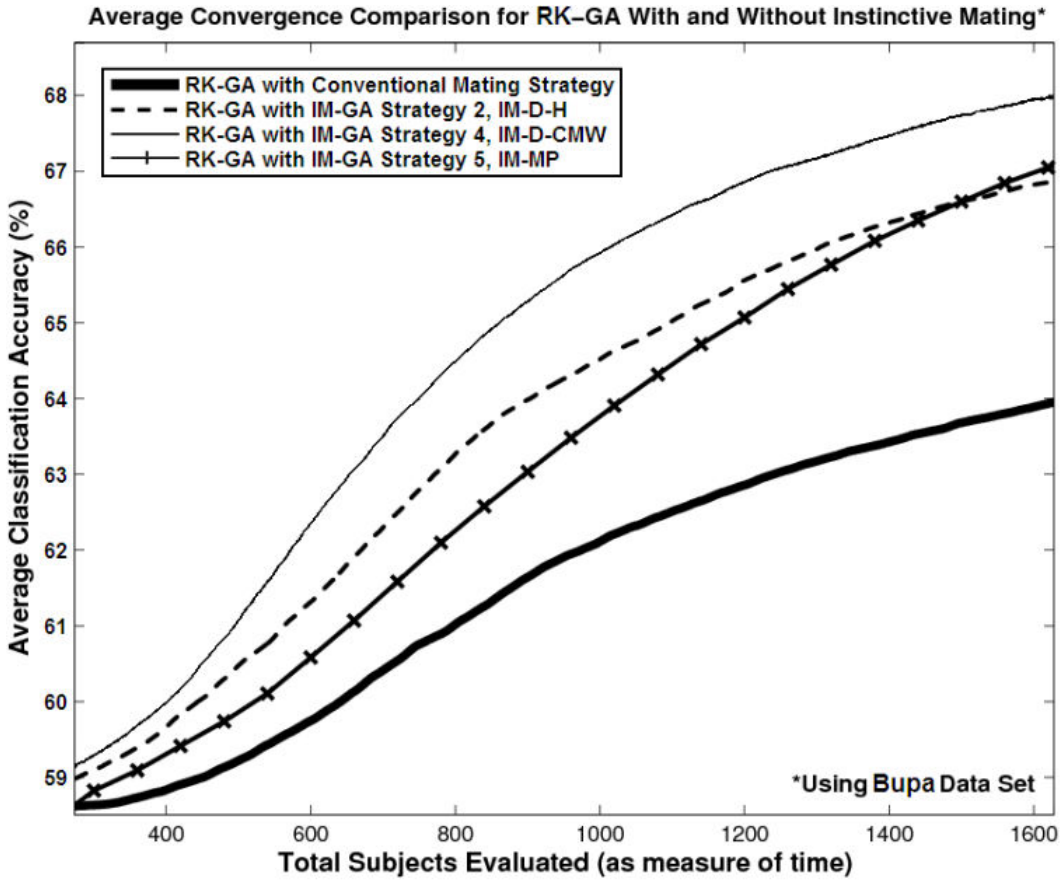

Figure 2.1: Convergence speed of the baseline $RK\text{-}GA_0$ with and without the *IM-GA* strategies in the UCI `bupa` data set.

To demonstrate how the GA convergence typically accelerated with the use of the *IM-GA* mating strategies, **Figure 2.1** shows how the *average specimen accuracy* improved over *time* (measured by the total number of fitness-function evaluations) for the baseline $RK\text{-}GA_0$[9] and three selected *IM-GA* mating strategies on the UCI data set "`bupa`" (all five proposed *IM-GA* mating strategies are discussed in Chapter 4,

[9]The baseline $RK\text{-}GA_0$ is an improved version of the original *RK-GA* that was implemented in this work. Just as the original *RK-GA*, the baseline $RK\text{-}GA_0$ also relies on the conventional mating strategy to solve Testbed Problem 1.

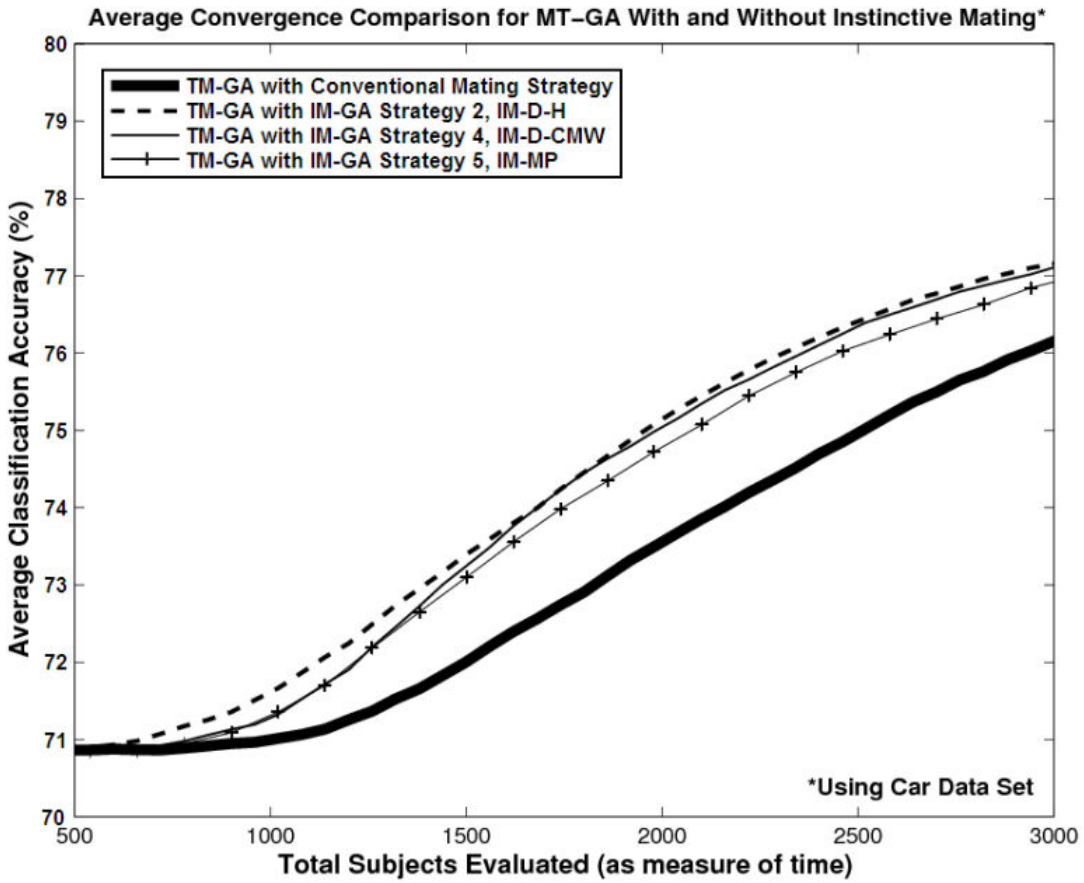

Figure 2.2: Convergence speed of the baseline *TM-GA*$_0$ with and without the *IM-GA* strategies in the UCI `car` data set.

and these results are simple illustrations). The plot presents results averaged over repeated cross-validation runs. Notice that in the *IM-GA* strategies, the average accuracy of the induced 1-NN classifiers grew faster than in the baseline *RK-GA*$_0$.

Similarly, **Figure 2.2** shows how the baseline *TM-GA*$_0$'s convergence typically accelerated with the use *IM-GA* mating strategies (the same three strategies illustrated in **Figure 2.1**) on the UCI data set "`car`." The plot presents results averaged over repeated cross-validation runs. Notice that in the *IM-GA* strategies, the average accuracy of the induced decision forests (whose decision trees were induced by the C4.5 Decision Trees program) grew faster than in the baseline *TM-GA*$_0$.

2.3.2 Classification Accuracy

In the Machine Learning literature, the classification accuracy is the main criteria used to evaluate the performance of supervised learning classifiers. The classification accuracy of a classifier is formally measured as the percentage of examples in a separate testing data set[10] that are correctly "labeled" by a classifier. Additionally, the classification accuracy is the primary optimization objective in the definition of Testbed Problems 1 and 2. Hence, both baselines *RK-GA*$_0$ and *TM-GA*$_0$ contain classification accuracy terms in their respective fitness-functions. During the genetic search, the classification accuracies of the classifiers discovered by both *RK-GA*$_0$ and *TM-GA*$_0$ are measured through evaluation on the training data set examples.

2.3.3 Data Set Reduction

This section describes a performance criteria used to measure how well the GA is able to recognize and discard noisy examples (i.e. having wrong class labels), redundant examples (i.e. promoting increased training and classification costs), and noisy/irrelevant attributes from a data set in order to build better classifiers. The ability of the GA to recognize and discard noisy/redundant examples in data set is referred to as the "example set reduction" ability. The ability of the GA to recognize and discard the noisy/irrelevant attributes in a data set is referred to as "attribute set reduction" ability.

In Machine Learning, data quality is known to impact the quality of the classifiers built. Removal of harmful examples and attributes from the training data set prior

[10] A testing data set usually refers to a separate set of data that was not used to train a classifier.

to building a classifier is an absolute requirement for the simultaneous optimization of classification accuracy and classifier model complexity (i.e. size or memory footprint) (Zhu, Wu, and Chen 2003; Dietterich 1995). Hence, this work evaluates the ability of the GA to optimize both the example and attribute sets used to build a classifier by discarding noisy/redundant examples and noisy/irrelevant attributes.

Attribute Set Reduction In Testbed Problems 1 and 2

To evaluate the "attribute set reduction" ability, two steps were taken following the examples from (Rozsypal and Kubat 2003; Quirino and Kubat 2010). First, irrelevant (or artificial) attributes were introduced into the UCI data sets used for experimentation. This was done because UCI data sets are known to have been designed by experts and consequently have mostly relevant attributes. Those attributes originally present in the UCI data sets are referred to as "original attributes", and those irrelevant attributes that were manually introduced are referred to as "artificial attributes". Second, two measures were used to capture the "attribute set reduction" ability of the GA. The two measure are as follows:

- Measure 1: The percentage of "original" attributes retained by each of the GA-generated classifiers, and;
- Measure 2: The percentage of "artificial" (irrelevant) attributes retained by each the GA-generated classifiers.

Notice that both Measures 1 and 2 capture the ability of the GA to discover classifiers with compact models (i.e. using less attributes). However, more importantly,

Measure 2 clearly captures the ability of the GA to discard the irrelevant attributes in a data set.

Example Set Reduction in Testbed Problem 1 Only

The size of the example set used to train a classifier is a measure of classifier complexity that is more relevant when applied to the 1-NN classifier (investigated in Testbed Problem 1) than to the decision tree classifier (investigated in Testbed Problem 2). That is because the number of examples in a training data set directly impacts the classification costs of the 1-NN classifier. Indeed, the ability of the GA to remove noisy/redundant examples from a data set is an important optimization objective for both Testbed Problems 1 and 2. However, in terms of measuring classifier model complexity, the complexity of a decision tree classifier is better captured by criteria such as the total number of nodes, the total number of leaves, and the average number tests required to classify an example. Hence, the "example set reduction" ability of the GA is used as a performance criteria for Testbed Problem 1 only. Criteria that are more relevant for measuring the complexity of decision tree model are presented in the next section.

The "example set reduction" ability of the GA was evaluated in Testbed Problem 1 by measuring the percentage of examples retained by each of the GA-generated 1-NN classifiers.

2.3.4 Classification Costs Reduction In Testbed Problem 2

This section describes the performance criteria chosen to evaluate the complexity (or size) of the GA-generated decision forests in Testbed Problem 2. The criteria

discussed here have been commonly used in the literature as measures of classification costs for decision trees and, thus, can be analogously extended to decision forests (Murthy and Salzberg 1995; Lim and Shih 2000; Alkhalid, Chikalov, and Moshkov 2011).

In addition to the common "attribute set reduction" criterium described in the previous section, which is a suitable measure of decision forest complexity, four additional measures were adopted to further capture the different aspects of the model complexity of decision forests. These four measures are cumulative measures computed by *summing-up* the measures for the individual decision trees in a decision forest. For example, given a decision forest, the following four measures were computed:

- Measure 1: The number of trees making up the decision forest (or ensemble size);
- Measure 2: The total sum of the number of nodes in all decision trees making up the decision forest;
- Measure 3: The total sum of the number of leaves (or decision rules) in all decision trees making up the decision forest, and;
- Measure 4: The total sum of the *average number of tests required to classify an example* by each decision tree making up the decision forest. Each decision tree in a decision forest may require a different *average number of tests* to classify an unknown example. These *average number of tests* were added up for all trees.

Notice how Measures 1, 2, and 3 combined reflect the *computational storage cost* of a decision forest (i.e. the amount of *memory* needed to store the decision tree models). In addition, Measures 1, 3, and 4 combined reflect the *computational time cost* associated with the classification of an unknown example by a decision forest.

CHAPTER 3

Literature Review

This chapter reviews principles from the field of Artificial Intelligence (AI) that are relevant to this work. In particular, this work builds upon ideas from two main sub-fields of AI: (1) *Genetic Algorithms* (GA) from the AI sub-field of Evolutionary Computation and (2) *Supervised classification* from the AI sub-field of Machine Learning. Both of these sub-fields have have been thoroughly investigated in the rich and vast Machine Learning literature. Research works related to the GA have sought to both improve genetic search as well as apply it to numerous search and optimization problems across various domains of sciences. Similarly, works related to the domain of supervised classification have sought to create more accurate, compact, interpretable, and generalized techniques capable of extracting relevant information patterns from imperfect sets of labeled examples. Some researchers have also investigated the conjoint applications of these two sub-fields by making use of the natural connection between *learning* and *search*. For example, the problem of inducing the most accurate and compact version of a given supervised classifier can be theoretically solved by *searching* through countless possible configurations of such classifier

for the optimal one. Thus, *learning* and *search* can be viewed as *complementary* AI research topics.

The review begins with an introduction to supervised classification, which is considered one of the most important tasks in the field of Machine Learning. Moreover, a key issue in supervised classification is discussed: the impact of data quality on the performance of supervised classifiers. In this work, this key issue was addressed through the design and implementation of GAs capable of discovering optimal subsets of example and attributes from data sets with the goal of inducing more accurate and compact classifiers. Next, an introduction to the GA is given that provides an overview of various principles constituents of genetic search. Finally, a survey is presented on previous works related to the application of the GA to the two chosen testbed problems from the domain of supervised classification: 1) the 1-NN Tuning problem, and 2) the Optimal Decision Forests problem.

3.1 Machine Learning - Learning Relevant Patterns From Imperfect Data

Machine learning is a central research topic in the field of Artificial Intelligence (AI), a branch of computer science concerned with creating systems that mimic human intelligence. Under AI, Machine Learning is the scientific discipline concerned with the development of techniques that allow computers to use example data (i.e. observations) to solve problems (Alpaydin 2004). This concept of solving problems based on *past experiences* (i.e. observations) is founded on the universal principle of *inductive inference*: the ability to infer general rules about the nature of statistical processes

from imperfect observed data in order to make predictions on future data (Angluin and Smith 1983).

One of the main reasons why the study of Machine Learning is important to the engineering sciences is because various engineering problems can only be adequately defined through examples[11]. In numerous engineering problems, the response of a system (i.e. its output value) to predetermined input values is known. However, the internal *function* of the system itself, which describes the concise relationship between the input and the output values, is unknown. Moreover, in many problems, the amount of data available to define the input and output values is very large, making it very difficult for a human to generalize a concise relationship between the input and response of a system (Nilsson 1998). This issue is further aggravated by the presence of noise in the observed data, which can mislead the interpretation of intrinsic relationships between input and output values by a human. For example, the UCI `sonar` data set described in Table 5.1 is a classical example of a complex supervised classification problem that can only be adequately described by examples (i.e. observations). The goal of this problem is to determine whether a detected cylindrical object is either a "rock" or a "metal" based on the energy in the frequency spectrum of reflected sonar signals (Gorman and Sejnowski 1988). The examples in

[11] An example represents some observation about the input and response of a system. An example is described by a pair of (1) a set of real, boolean ("0" or "1"), or categorical-valued (i.e. discrete and finite values) attributes representing the input to a system and (2) either a label that categorizes the example into a group (i.e. category or class) or a real value representing the observed response of a system. Moreover, when examples carry labels, the process of inferring a function that maps the attribute values to the labels is known as a *classification problem*. In contrast, when the examples carry a real value instead of a label to represent the response of a system, the process of inferring a function that maps the attribute values to the real value is known as a *regression problem*. Moreover, in some Machine Learning applications, examples can belong to multiple groups and, thus, have multiple class labels.

the `sonar` data set describe a complex relationship between the properties of the sonar signal emitter (i.e. the angle, aperture, and power of the emitted sonar signals), the material properties of the cylindrical objects, and the energy spectrum of the reflected sonar signals. The goal of Machine Learning is to develop algorithms that allow computers to automatically decipher such complex relationships and represent them as *functions* that accurately *map* sets of input values to their corresponding observed output values.

In Machine Learning problems where the response of the system under investigation is a categorical value (i.e. taking on discrete and finite values)[12], *supervised classification* is the Machine Learning task of deciphering the complex relationships between the attribute values of a set of examples (representing the input values to a system) and the observed labels of the examples (representing the system's response to a given input) with the goal of making intelligent predictions on the categories of future unlabeled examples. Supervised classification, and its inherent issues, is described in detail in the following section.

3.1.1 Supervised Classification - Learning How to Discriminate From Examples

Classification is, in essence, the task of predicting the group membership (i.e. class) of objects. It is a process which occurs naturally and continually in our everyday lives. For example, classification takes place when recognizing familiar faces (e.g. John versus Mary) or different objects on a desk (e.g. pencil versus pen), when discriminating between the road and a pedestrian while driving, when learning new routes

[12]Contrast systems whose response are real values.

to or from work in order to avoid known heavy traffic patterns, or when predicting a friend's reaction to a re-occurring circumstance. In general, classification involves the processes of *learning* from examples (or past experiences), and consequently also *predicting* the outcome of different situations.

When applied to computer related applications, the *objects* are simply represented by data examples. Supervised classification predicts the class (or label) of unknown examples (i.e. those without labels) by comparing their observed properties (i.e. attribute values) to those of known examples (i.e. those with labels). In essence, supervised classification is the Machine Learning task of inferring a *function* that accurately maps the attribute values of known examples to their corresponding labels with the goal of making intelligent predictions on the class of future unknown examples.

Algorithms designed to infer a *function* for a given classification problem are called supervised classifiers (hereinafter "classifier"). The process of building a classifier is referred to as "training a classifier", an alias that reflects the fact that the learning processes of many existing classifiers require multiple passes through the training data set in order to adequately "train" the classifier to recognize meaningful information patterns in a training data set. The performance of classifiers is generally evaluated on the basis of their classification accuracy on unknown examples (also referred to as its *generalization ability*[13]) as well as their classification costs[14], both of which vary greatly among different existing classifiers.

[13]The generalization ability of a classifier is the ability to correctly assign class labels to unknown examples which were not used to train the classifier's model.

[14]The classification costs of a classifier are related to the computational time and storage costs associated with assigning a label to an unknown example.

The performance of classifiers is affected by many issues. For example, when classifiers are trained on large data sets, their training processes can require very high computational costs of time and storage (Fuller, Groom, and Jones 1994). Additionally, large training sets generally lead to large classifier models, which in turn leads to high classification costs. Another key issue in supervised classification is the impact of data quality on the performance of classifiers. Real-world data sets have noise, which misleads classifiers into creating models that are more complex than required to represent the underlying relationship between the training examples and their class labels (Dietterich 1995). This issue is known as classifier *over-fitting* and it is discussed in more detail in the next section.

The Machine Learning literature reveals that numerous classifiers have been developed over the past decades. Some well-known and widely employed classifiers are the C4.5 Decision Trees program developed by Quinlan (1993), Neural networks (Haykin 1999), Support Vector Machines (Cristianini and Taylor 2000), and the k-NN classifier (Quirino and Kubat 2010), all for which software implementations are freely available through software packages such as the *Weka* Data Mining Software Package (Hall, Frank, Holmes, Pfahringer, Reutemann, and Witten 2009)[15]. A more recent strategy is the adoption of *ensembles of classifiers*, which are classification systems that combine the output of multiple (heterogeneous or homogeneous) classifiers to achieve higher predictive power. The concept of classifier ensembles is described in

[15] *Weka* is an open-source, Java-based API for the development of Machine Learning and Data Mining tools

more detail in Section 3.4, where the Optimal Decision Forests problem (Testbed Problem 2) is discussed.

The demand for improved supervised classifiers is always high. Numerous real-world applications such as face recognition, video surveillance, natural language processing, and online search engines, have all led to an increasing demand for more accurate and compact (i.e. having lower classification costs and faster response time) classifiers. This work addresses this real-world demand through the design and implementation of GAs that are capable of optimizing both the accuracy and size of popular classifiers described in the Machine Learning literature.

3.1.2 The Impact of Data Noise in Supervised Classification

In supervised classification, data set quality has a major impact on the quality of the classifiers that are built. Unfortunately, the quality of real-world data sets is frequently damaged by noise from various sources. This makes the removal of noise from data sets a practical Machine Learning task. The main motivation for removing noise from data sets prior to inducing classifier models is to achieve higher classification accuracy. Zhu et al. (2003) claims that classification accuracy cannot be optimized unless the data set is "cleansed" of noise prior to building a classifier's model. This claim has been supported by various works in the GA literature which have shown that the simultaneous removal of harmful examples and attributes from data sets can indeed lead to the optimization of both the accuracy and size (i.e. classification costs) of existing classifiers (Rozsypal and Kubat 2003; Quirino and Kubat 2010; Ishibuchi

and Nakashima 2000; Kuncheva and Jain 1999; Soryani and Rafat 2006; Hart 1968; Gates 1972; Tomek 1976).

Data noise emerges from sources such as machine error (i.e. sensor errors during automatic data collection), as well as from human data entry errors (i.e. entering the wrong class label for an example). Moreover, data noise affects both the attribute values describing the examples of a data sets (hereinafter *attribute noise*), as well as the labels that categorize the examples into different classes (hereinafter *example noise*). Classifiers trained with noisy data sets tend to create overly complex models that capture not only the relevant underlying patterns in their training data sets, but also the irrelevant ones. This issue is known as over-fitting (Dietterich 1995). Over-fitting renders classifiers unable to correctly recognize the category labels of unknown examples (i.e. those examples without labels and which have unique combinations of attribute values that were not present in the original training data set). In the Machine Learning literature, data noise is referred to as one of the major causes of over-fitting in the task supervised classification (Tan, Steinbach, and Kumar 2005).

Certain classes of classifiers are more prone to over-fitting due to data noise than others. In Machine Learning, the likelihood of a classifier to build a model that over-fits a training data set is measured in terms of the classifier's *bias* and *variance* statistics. The sum of the bias and variance make up the *mean squared error* (MSE) of a classifier, or its *expected error*, computed over all possible training data sets sampled from an unknown distribution that a given classifier attempts to estimate (Oza and Tumer 2008; Rokach 2010). The noise resulting from sampling data examples from an unknown distribution (i.e. corresponding to sensor errors in the real-world) also

contributes to a classifier's expected error. However, data noise is generally ignored in computations of classifier MSE because estimating sampled data noise can be difficult, unless sufficiently repeated data samples are available to estimate the variance in the labels of repeated samples. The bias measures the difference between (1) the true function that generated the possible training data sets and (2) the "average" function estimated by a classifier over all possible training data sets. In other words, the bias describes the average error of a classifier and it indicates the ability of a classifier to adequately fit its training data set. The bias can reveal systematic errors in a classifier. The variance describes how much the function estimated by a classifier varies from its "average" function over all possible training data sets (i.e. the spread of the estimated functions). That is, the variance is an indicator of how much the structure of a classifier's model varies from one training data set to another. A good classifier should have both low bias and low variance in order to achieve good performance. That is, the classifier should on average be correct (i.e. low bias) while the structure of its model should remain stable from one training set to another (i.e. low variance). In practice, however, there is a trade-off between bias and variance in classifiers. For example, a classifier that adequately fits a training data set (i.e. low bias) also requires flexibility to fit the training data set (i.e. high variance).

Figure 3.1 illustrates the decomposition of the MSE of a classifier into bias, variance, and noise. For simplicity of illustration, this example uses a true function that outputs numerical values instead of class labels (a similar analysis applies for class labels). The top-left plot of **Figure 3.1** shows the true function (solid line) along with 30 estimated fits. Each of the 30 estimated fits were produced by sampling sets

of 10 examples (with simulated sampling noise) from the true function and fitting the sampled example set to a 5_{th}-degree polynomial (hereinafter "polynomial estimator"). Thus, each of the 30 sampled sets correspond to a different training set having 10 examples. The simulated sampling noise is illustrated in the bottom-right plot of **Figure 3.1**; notice how the vertical coordinate value of sampled examples (e.g. the plus-signs) vary above and below the true function (in the real-world, this variability is caused by sensor errors). The top-right plot of **Figure 3.1** gives the true function and the "average" function estimated from the 30 estimated fits shown on the top-left plot. The regions where the true function and the "average" estimated function differ significantly correspond to regions of high bias (i.e. systematic prediction errors) by the polynomial estimator; overall, the polynomial estimator exhibit a low bias. The bottom-left plot of **Figure 3.1** gives the the 30 estimated fits and their corresponding "average" function. The spread of the 30 estimated fits around the "average" function estimates the variance of the polynomial classifier. Notice how the 30 estimated fits vary significantly around their average, which indicates that the polynomial estimator has a high variance. **Figure 3.1** gives a clear illustration of the bias/variance trade-off that is common to classifiers; in this example, the polynomial estimator exhibits low-bias/high-variance behavior.

Classifiers exhibiting high variance are generally more prone to over-fitting the noise present in data sets than those with low variance. For example, the C4.5 Decision Trees (Quinlan 1993) program (hereinafter "C4.5"), which is used extensively in this work as a decision tree builder (see Section 2.2 for a detailed description of C4.5) is known to be a low-bias/high-variance classifier. C4.5 easily over-fits its

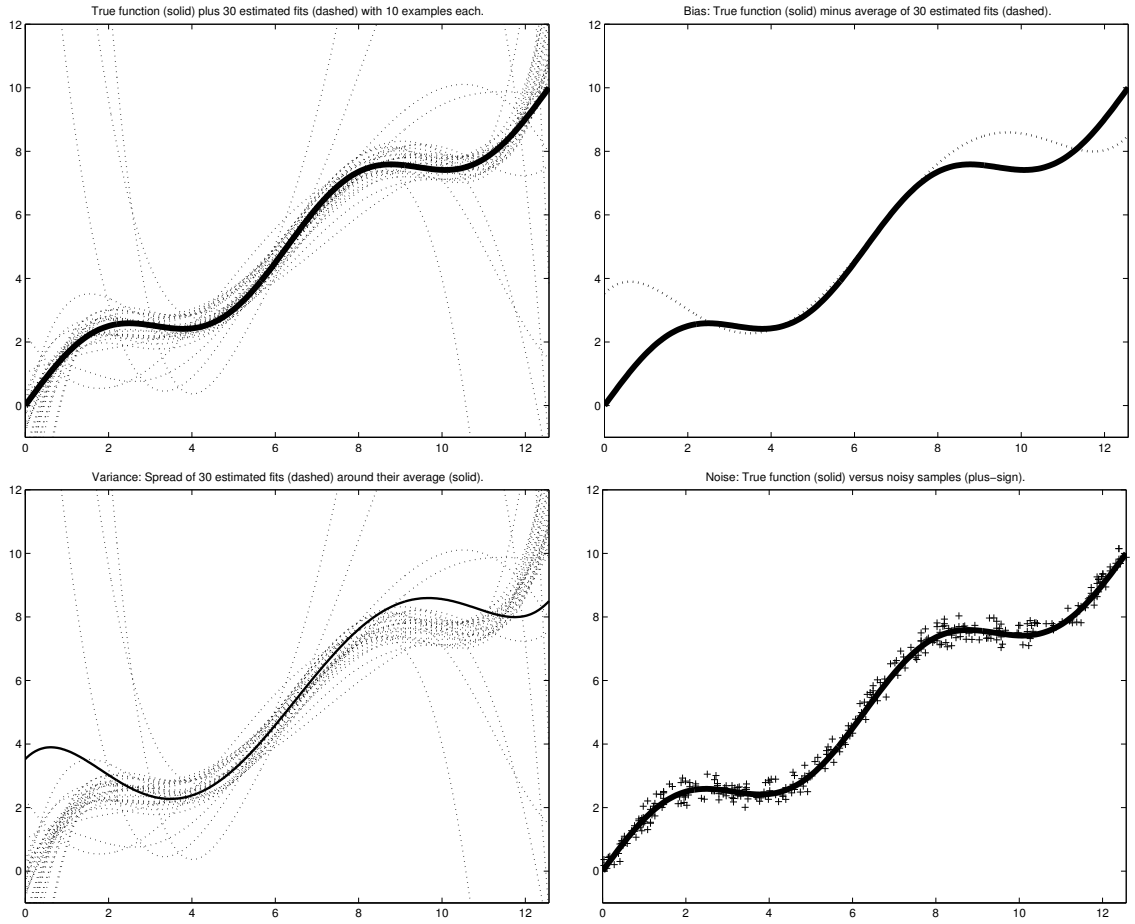

Figure 3.1: Decomposing the *Mean Squared Error* (MSE) of a classifier into *bias*, *variance*, and data sampling *noise*. In this example, the estimator is a 5_{th}-degree polynomial fitting 30 randomly sampled training sets, each with 10 examples.

induced decision tree models to noise found in data sets (Hall and Smith 1998; Ho 1998a). In C4.5, over-fitting due to data noise is directly manifested in the structure of the induced decision trees as noisy/irrelevant test patterns that are built into the model simply to fit the noise in the training data set.

The issue of classifier over-fitting is further aggravated by the presence of irrelevant attributes in data sets. Irrelevant attributes are those that contribute to increased data set dimensionality without the addition of meaningful patterns (or information) to a data set. As a result from the presence of irrelevant attributes in data sets, Ma-

chine Learning problems become more difficult and more computationally expensive to solve. For example, as the results in Section 5.2 will demonstrate, irrelevant attributes are often retained in the decision tree models induced by C4.5, consequently harming the classification accuracy of the induced decision trees. Similarly, in the case of the 1-NN classifier, irrelevant attributes skew the Euclidean distance computations, harming classification accuracy (Rozsypal and Kubat 2003; Quirino and Kubat 2010). In essence, the presence of irrelevant attributes "confuse" supervised classifiers (Wu and Zhang 2004).

In addition, too many irrelevant attributes can greatly increase classification costs. This issue is more readily apparent in applications of "lazy learners"[16] such as the 1-NN classifier investigated in Testbed Problem 1. For example, the classification costs of the 1-NN classifier increase linearly with respect to increases in the number of attributes in the training data set.

Among the reasons for the presence of irrelevant attributes in data sets are:

1. Complexity of data collection: The complexity and/or monetary costs associated with the collection of more relevant attributes (i.e. the cost of collecting deep ocean seismic soundings in order to better forecast underwater earthquakes events or the inherent dangers of steering a ship into the eye of a hurricane in order to collect precious inner-core soundings that can lead to better forecasts of hurricane intensity and track);

[16]Lazy learners are leaning methods which wait until a query is made to the classification system before making any generalizations about the underlying relationship between training examples and their labels. That is, no function is induced until a query is done to the system.

2. Data collection errors: Sensor errors occurring during automatic data collection can yield meaningless data.

3. Quality control: Failure by an expert to control the quality of the attributes during the design stage of a data collection experiment, i.e. considering "shoe size" as a relevant feature when making generalizations about an individual's car driving risk, and;

4. Unavailability of domain knowledge: The low availability of published domain knowledge on a new research topic hinders the design of more relevant attributes for data collection experiments, while at the same time promoting the adoption of more "experimental" ones.

In summary, the presence of noise is common in real-world data sets and its removal is a practical Machine Learning task. The removal of noise from data prior to inducing a classifier's model is an important step toward optimizing the performance of supervised classifiers: maximizing their classification accuracy while minimizing classification costs. In this work, data noise removal is the approach taken to build optimal classifiers for Testbed Problems 1 and 2. To achieve this, two GAs were designed and implemented (the baseline $RK\text{-}GA_0$ for Testbed Problem 1 and the baseline $TM\text{-}GA_0$ for Testbed Problem 2) to recognize and discard the harmful examples and attributes hidden in data sets with the goal of building optimal classifiers.

3.2 An Overview of Genetic Algorithms (GA)

The Genetic Algorithm (GA) is a popular approach to search and optimization. Over the past decades, the GA has been applied to hundreds of real-world problems across various domains of science (Karr and Freeman 1998; Haupt and Haupt 2004; Popescu, Popescu, and Mastorakis 2009). The GA is part of a group of optimization techniques known as Evolutionary Algorithms (EA). EAs exhibit three main traits. First, EAs operate on *populations* (i.e. groups) of solutions that are generally randomly initialized; the use of multiple solutions allows optimization problems to be solved in a parallel fashion. Second, the solutions are improved iteratively though the sequential application of mechanism that mimic Darwinian biological evolutionary process such as mating, recombination, mutation, and survival of the fittest. This process is referred to as *natural adaptation*; better solutions evolve from existing solutions. Third, EAs are "fitness-driven". Under this paradigm, each solution in the population represents a biological specimen whose *genetic code* (e.g. chromosomes) encodes a potential solution to an optimization problem. The quality of the genetic code of a specimen determines its ability to survive the environment. Moreover, the environment is determined by a *fitness-function* (X. Yu 2010) that captures each specimen's ability to optimize the objectives of a problems. Thus, EAs approach the task of optimization by iteratively *adapting* the specimens in the population toward becoming more "fit" to their environment. This approach is analogous to that of searching through the space of possible solutions to an optimization problem by generating new solutions from existing solutions (as opposed to random search) with the goal

of discovering good solutions capable of optimizing the objectives described by the fitness-function.

From among the various existing EAs, the GA is the most popular approach. Its popularity is due to its generality when compared to other EAs. The GA is a blueprint, or recipe, for writing computer programs capable of solving numerous search and optimization problems. The initial blueprint of the GA was developed by Holland (1975), and although it has been greatly extended, the basics still remain the same. To start, a "population" of potential solutions (i.e. specimens) is maintained, which is a typical trait of EAs. Each specimen carries one or more "chromosomes" that encode a potential solution to an optimization problem. Specimens are evaluated to determine their fitness to the environment. This process generally involves decoding the solutions encoded in the chromosomes of specimens and measuring their quality according to the *user-defined* fitness-function. In each succeeding iteration (also referred to as a "generation"), the GA retains the most fit specimens from among those of the previous generation and the generated "offspring." Offsprings are generated using crossover and mutation operators. Crossover combines the chromosomes of two specimens (also referred to as a "parent-pair") to create two new children specimens, echoing reproduction in the natural world. Mutation randomly modifies the chromosomes of children specimens to introduce new information into the population, allowing the GA to search in diverse regions of the space of possible solutions to an optimization problem. Though the initial population is often composed of randomly generated solutions, and thus performs poorly, performance improves greatly over subsequent generations.

The fitness-function and the chromosomal representation of the GA are the two mechanisms that allows users to *map* the custom objectives of different optimization problems into the realm of natural evolutionary adaptation. The fitness-function is *user-defined* and it is responsible for providing the GA with feedback regarding the quality of its discovered solutions. It is through the fitness-function that the user guides the genetic search. The chromosomal representation allows the user to *map* real-world solutions to optimization problems into a symbolic representation that is suitable for genetic search. Chromosomes have traditionally been represented as binary ("0" or "1" valued) strings. This representation is very general and offers 2 main advantages. First, it facilitates the representation of real-valued solutions to optimization problems (e.g. the radius of a circle, the bandwidth of a network route, the number of tasks in a pipeline, and the accuracy of a classifier)[17]. Second, the individual bits of a binary string can also be used to indicate the absence (i.e. "0") or presence (i.e. "1") of objects, which facilitates the representation of solutions as subsets of objects picked from larger sets of objects. For example, this representation was used in the pioneering work of Kuncheva and Jain (1999), where binary strings were used to represent subsets of examples and attributes from a data set that optimized the accuracy of the 1-NN classifier.

The above discussion reveals one of the main advantages of the GA as an optimization tool: the GA is "problem-agnostic". That is, the GA makes no assumption about the shape of the fitness-function being optimized. This feature makes the

[17]When binary strings are used to represent real-valued solutions to an optimization problem, the precision of the solutions depend on the length of the binary string. However, the GA computational costs also increase proportionally to the length of the strings.

GA a suitable tool for solving global optimization problems represented by fitness-functions having numerous local minima/maxima and even discontinuities. Another major advantage of the GA is that it can be used to solve multi-objective optimization problems (Quirino and Kubat 2010) requiring the simultaneous optimization of multiple *conflicting* goals. When the GA is applied to such problems, the fitness-function can be designed to capture the ability of the GA-generated solutions to optimize all objectives of a problem simultaneously. One disadvantage of the GA is that its convergence tends to be slow when applied to the optimization of "well-behaved" fitness-functions. One reason is that the GA does not make use of properties of the fitness-function, such as gradients, to guide the genetic search. Another reason is that the stochasticity (i.e. randomness) introduced by the recombination and mutation operators, which enable the GA to search through complex fitness-function landscapes, also tend to slow down the GA convergence on "well-behaved" functions.

One of the main challenges involved in the design of new GAs is the need to prevent the genetic search from converging prematurely to suboptimal solutions. This is achieved by maintaining the diversity of information in the GA population. The issue of premature convergence is inherent from the fact that the genetic search process continuously looses information from its initial population of solutions with each iteration (Shamir, Saad, and Marom 1993). This loss takes place as newly generated and more "fit" solutions replace older and less "fit" solutions in the population; the information that was available in the replaced solutions is simply lost. Eventually, the GA runs out of information to generate new and better solutions and the genetic search ends. While mechanisms such as mutation attempt to retard the effects of

information loss by periodically introducing new information into the population, the effect is only limited. For example, if high mutation rates were used in the GA, the natural *adaptation* process would not occur because the generated children specimens would not resemble their parent specimens. In that case, the genetic search would be reduced to simple random search. Thus, preventing premature convergence in the GA requires that all components of the genetic search process (e.g. mating, recombination, mutation, and even survival) work together to maintain the diversity of information in the population in order to promote more efficient and diverse sampling of the search space of possible solutions to optimization problems.

3.2.1 The GA as a Tool For Multi-Objective Optimization

The GA is an excellent tool for solving multi-objective optimization problems; those problems having two or more mutually contradicting goals that must be simultaneously optimized. The majority of real-world problems are multi-objective in nature. For example, the two testbed problems (detailed in Sections 3.3 and 3.4) chosen in this work to evaluate the proposed *IM-GA* mating strategies are multi-objective optimization problems. Solving these testbed problems require the simultaneous optimization of the accuracy and size of supervised classifiers, which are mutually conflicting goals: training a classifier with too little data may lead to poor classification accuracy with low classification costs, however, training a classifier with too much data may improve accuracy while heavily degrading classification costs.

Researchers have developed two main alternative techniques that allow the GA to be applied to the optimization of multi-objective problems (Coello 1999; Konak,

Coit, and Smith 2006): (1) tailoring of the fitness-functions as a weighted sum of optimization objectives, that combines all criteria into a single formula, and (2) the use of a different "gender" in the GA for each of the conflicting goals. In particular, the weighted sum technique has been adopted by various researchers who applied their own versions of the GA to the 1-NN Tuning Problem, including Kuncheva and Jain (1999, Ishibuchi and Nakashima (2000, Rozsypal and Kubat (2003, Quirino and Kubat (2010). One drawback of these two techniques is that the relevance of the optimization objectives is determined on a somewhat subjective basis. For example, weight coefficients in the fitness-function are generally tuned manually to reflect the relevance of different optimization objectives.

A more general approach lies on the principle of Pareto-dominance (Chen, Chen, and Ho 2005), developed in the field of Multi-objective Evolutionary Algorithms (MOEA) (Coello 1999)[18]. In the presence of two or more optimization objectives, solution X is deemed to be a "Pareto-improvement" over solution Y if X is better than Y according to at least one optimization objective without being worse than Y along any other objective (this is also referred to as "Y is then Pareto-dominated by X"). Moreover, the set of all Pareto non-dominated solutions to a problem is called the Pareto-optimal set. Identifying the Pareto-optimal of a multi-objective optimization problem is the main goal of MOEAs, which have the following features (Konak, Coit, and Smith 2006):

1. MOEAs address optimization problems by identifying competing objective functions;

[18]Coello (1999) offers a detailed survey on the development of MOEAs.

2. MOEAs don't require prioritization or scaling of objectives, which eliminates the need for weights;

3. MOEAs yield *multiple* feasible solutions to a problem.

.

One of the first researchers to experiment with the Pareto-dominance framework in Genetic Algorithms was Schaffer (1985), who employed it in his system VEGA. VEGA splits the population at each generation into k sub-populations, each with its own fitness-function. Parents in each sub-population are selected according to one of these k different fitness-functions. The sub-populations are then merged, and recombination is used to create a new generation. The technique sometimes led to premature convergence, but experiments still showed it to outperform random search. VEGA has been successfully applied to numerous applications (Coello 1999).

Pareto-based fitness-functions are also mentioned in Goldberg's famous book (Goldberg 1989). The algorithm he describes searches for specimens that are Pareto non-dominated with respect to the rest of the population. These specimens are assigned the highest "rank" and are exempted from further competition. The process is repeated on the rest of the population until all specimens have thus been ranked. In the experiments reported by Liepins et al. (1990), this approach outperformed VEGA in a variety of set covering problems. Ritzel et al. (1994) employed Pareto-non-dominating ranking, selection, and niching schemes in experiments related to cost and reliability optimization.

These early approaches inspired numerous other variations. Among these, attention deserve the Non-dominated Sorting Genetic Algorithm (NSGA) by Srinivas and Deb (1994), and the Niched Pareto Genetic Algorithm (NPGA) by Coello (1999). Both are characterized by their combined use of Pareto ranking and fitness sharing as "niching" mechanisms (to promote diversity in the Pareto fronts). For example, NSGA starts by finding the first "front" of Pareto non-dominated specimens from the population and assigning to them the same high "dummy" fitness value. A normalized sharing function is then used to compute for each specimen its distances from all other specimens in the front. The sum of these distances then defines this specimen's "niche count" that measures how much the spatial region surrounding the specimen is crowded. The specimen's fitness is computed by dividing its dummy fitness value by its niche count. After the removal of the specimens in the first "front" from the population, NSGA collects the subsequent Pareto non-dominated front and assigns a dummy fitness value that is lower than the minimum shared fitness of the previous front. The process is repeated until the entire population has been classified. The stochastic selection that follows is based on the final shared fitness: the first non-dominated front is granted a greater portion of the recombination process. Non-dominated sorting and "fitness sharing" allow NSGA efficient search in non-dominated regions, and the sharing mechanism also allows NSGA to attain diverse Pareto-optimal distributions. NSGA outperformed VEGA and other approaches in numerous applications (Coello 1999).

More recent MOEAs also adopt the elitistic survival strategy (Bentley 1999), a technique that speeds up MOEAs and improves their search performance. Represen-

tative examples include the Strength Pareto Evolutionary Algorithm (Zitzler, Laumanns, and Thiele 1999) (SPEA), the Pareto Archived Evolution Strategy (Knowles and Corne 2000) (PAES), and NSGA-II (Deb, Pratap, Agarwal, and Meyarivan 2002). In particular, NSGA-II improved over NSGA by adopting a better sorting algorithm to reduce the computational costs of population ranking and replacing the user-defined sharing parameter with a "crowding density" measure that eliminated manual parameter tuning. NSGA-II also adopted survival-selection elitism to speed up convergence.

Among the numerous sub-classes of MOEAs, the Artificial Immune Systems (AIS) received attention. It takes inspiration from immunology, where specialized B-cells in the immune system can adapt to new types of antigens through such biological processes as cloning and hypermutation (Hart and Timmis 2008). The first attempt to use AIS in multi-objective optimization was the MISA approach by Coello and Cortes (2005) that splits the population into two types: 1) antigens (Pareto non-dominated solutions), and 2) antibodies (Pareto-dominated solutions). The fitness value of each antibody is obtained from its similarity to a randomly selected antigen. A percentage of the most fit antibodies are cloned and mutated at a rate inversely proportional to each clone's similarity to a randomly selected antigen. MISA's emulation of immune system adaptations was found to perform better than NSGA-II and PAES. A more recent adaptation is NNIA (Gong, Jiao, Du, and Bo 2008). Its unique feature is its adaptation of NSGA-II's crowding distance measure into a fitness measure. The intention is to give higher selection and recombination probability to solutions in less crowded regions of the search space.

3.2.2 The Use of Multiple Populations in Genetic Algorithms

The original motivation behind multi-population GA (also known as Parallel or Distributed GA) was the desire to harness multiple processors in solving large optimization problems (Cantu-paz 1997b).[19] Early work thus often ignored such issues as the problem of premature convergence and promotion of diversity (niching) inherent from the use of multiple populations in the GA, and focused instead on programmatic ways to exploit massive parallelism. The fact that multi-population approaches may help improve optimization properties of the GA was discovered later.

Three main types of parallel GAs have been studied. The simplest is the master-slave GA that uses a single population but distributes the fitness-function calculations among multiple processors. Another approach, the fine-grained parallel GA, imposes mating restrictions on vast populations: the population is first spatially structured and recombination is restricted to small neighborhoods that minimize communication among processing units; some neighborhoods overlap, allowing interactions. The third, and most important, approach is the "multi-deme" or "multi-population" GA that further reduced the communication costs of parallel GAs by evolving multiple populations with periodic "interbreeding" (migration of individuals among populations). This is the philosophy behind one of our own techniques. In what follows, we will use the terms "multi-deme" and "multi-population" interchangeably.

The one practical advantage of the multi-deme architecture is that it is basically a simple extension of the single-population case—execution of a single-population GA processes in parallel computers, networked so as to periodically exchange a few

[19] Cantu-paz (1997b) has an excellent survey on the origins of "multi-population GA".

individuals. Extending a single-population GA to a multi-population GA is thus quite simple. Optimizing its performance is more difficult because of the added decision factors such as population sizes, migration rates, and migration topologies (where should migrants be allowed to go?).

One of the first studies of parameter tuning in multi-population GA was conducted by Grosso (1985). His experimental setup comprised of 5 populations exchanging individuals at a fixed rate over a "dynamic" topology where all populations were allowed to communicate. Experiments showed that improvements in average fitness were faster in smaller parallel populations than in a single large population. Isolating small populations led to poorer final solutions, and the convergence speed was greatly affected by migration rate—with low migration rates, the migrant individuals were not well absorbed into the destination populations.

Various studies found that the performance of multi-population GA is greatly affected by topology—the restrictions on how the populations are connected for the purpose of exchanging individuals. A more recent study on "static migration" topologies (Cantu-paz 1997a) investigated such topologies as uni-directional and bi-directional rings, the 4×4 toroidal mesh, 4-D hypercubes, and fully connected topologies, and the configuration of the migratory connections remained unchanged throughout the experiment. The main observation was that densely connected topologies converged faster than sparse configurations.

Popular is also the "dynamic topology," where the migratory connections are not fixed. Each population is evaluated as a whole according to some criteria, and the migrant is sent to where it has the greatest potential of "making a difference" (Grosso

1985). Some studies showed that this accelerated convergence as compared to fixed migration topologies (Munetomo, Takai, and Sato 1993; Lin, Punch, and Goodman 1994). In particular, Lin et al. (1994) utilized the genotypic distance between two populations as the migratory criteria, while Munetomo et al. (1993) employed to this end a measure of the diversity within a population.

The parameter-tuning problem extends to the multi-population algorithms. Some authors therefore sought to develop algorithms that automatically tune their performance. They proposed an approach that inputs the desired number of demes and the migration topology, and outputs estimates of the required population size and the number of epochs needed to achieve certain level of quality in the final solution (Cantu-paz 1999). The algorithm yielded accurate prediction for a handful of different migration topologies.

The multi-population GA has been successfully applied to numerous multi-objective applications, including synthesis of integrated circuits, generalized multi-modal function optimization, and even the feature selection problem. A representative approach is the BMPGA algorithm (J. Yao 2005), designed to survey multi-modal function environments. Here, fitness and gradient are derived from the function being optimized. The idea behind this bi-objective evaluation is that the gradient term is a better criterion for distinguishing between global and local maxima points than the fitness-function. The population sizes can vary with each generation. An algorithm was proposed to determine in each generation the correct cluster of each specimen. The approach has its own way to measure similarity between a specimen and a population. This measure then determines how specimens are to migrate to different

clusters. Experiments on various complex multi-modal functions favored BMPGA's performance over other genetic approaches.

Finally, the MGAFS algorithm from H. Zhu (2006), which employs multi-population GA principles for feature selection in 1-NN classifiers, randomly creates two sub-populations, one biased toward chromosomes with prevailing zeroes, and the other biased toward those with prevailing ones. The fitness-function is defined as the specimen's accuracy, mating is rank based. Migration relies on the best-worst policy, where the best specimen in a randomly chosen subpopulation replaces the worst specimen in the other.

Gendered Genetic Algorithms

Another idea, though less common, seen in implementations of the multi-population GA is to divide the specimens into multiple populations and allow them to mate according to "gender" (Rukovansky 2009).

Goh et al. (2003) were able to show that sexual selection in the recombination process can indeed improve the GA performance. In their implementation, they distinguished *exploration* and *exploitation*. The program divides the population into males and females. All females are allowed to reproduce (exploration), choosing males by a mechanism that gives preference to males with higher fitness value (exploitation). Experiments on diverse optimization problems showed that the scheme performed as well or better than the (fitness-based) roulette wheel, tournament, and rank-based stochastic selection schemes.

Also the approach by Velazco and Bullinaria (2003b, Velazco and Bullinaria (2003a) splits the population into males and females. Males are evaluated by the fitness-function, and females are evaluated by a combination of age, fertility, *and* fitness-function. Each gender has its own mutation rate.

Zhu et al. (2006) used age and gender to address premature convergence. They gave higher exploratory ability to males through higher mutation rates, while giving females higher local searching ability through lower mutations rates; mutation rates of individuals grew with age. Male specimens were only allowed to mate upon reaching a certain age, the idea being to make the resulting offsprings more stable.

Another version of the male-female duality has been proposed by Raguwanshi and Kakde (2006). In their system FAS3, females are treated as "niches" of sub-species in the population. Male specimens of a given species compete to mate with representatives from female niches. Specimens that do not perform well in a number of generations are merged with their nearby specimens. FAS3's performance compared favorably with asexual as well as gendered methods in a variety of experiments with uni/multi-modal functions.

Finally, Lis and Eiben (1996) proposed a new way to exploit the Pareto-optimum search in a multi-sexual GA paradigm they dubbed MSGA. In a domain with multiple goals, MSGA represented each objective function by its own gender and by its own sub-population. Each mating partner was selected from a different sub-population in proportion to the values of their respective fitness-functions. To these parents, uniform scanning crossover was applied, and each child was assigned the gender of

the parent who donated more genes. At each generation, the Pareto non-dominated solutions of the current and all previous generations were collected.

The subsequent sections report on previous research works that have specifically applied the GA to the optimization of Testbed Problems 1 and 2.

3.3 What is the 1-NN Tuning Problem?

This section discusses Testbed Problem 1, the 1-NN Tuning problem, a well-known optimization problem from the field of nearest-neighbor classifiers (k-NN). The 1-NN Tuning Problem involves the simultaneous selection of optimal subsets of examples and attributes (i.e. features or predictors) from a data set with the goal of maximizing the accuracy while minimizing the classification costs of the 1-NN classifier (Rozsypal and Kubat 2003).

A k-NN classifier keeps a store of "training examples," each described by a vector of attribute values and assigned a class label. When presented with a testing example, $\mathbf{x}$, the k-NN classifier assigns to it the class prevailing among the k training examples that have the shortest geometric distance from $\mathbf{x}$ (Cover and Hart 1967; Fix and J. L. Hodges 1989). The 1-NN classifier is, thus, a special case k-NN where $k = 1$. The training set is known to have a strong effect on the classifier's behavior. Not only that classification costs are high if there are too many training examples described by too many attributes; but noise in the class labels and/or attribute values can mislead the classifier; and if many of the attributes are unrelated to the output class ("irrelevant attributes"), the geometric distances are skewed, degrading the classification performance.

These problems can be mitigated. For example, computational costs are reduced by indexing mechanisms (Freidman, Bentley, and Finkel 1977; Nene and Nayar 1997; Sproull 1991), and the other problems can be addressed by the removal of noisy/redundant examples (referred to as the process of *data editing*) and by the removal of noisy/irrelevant attributes (referred to as the process of *feature selection*). Figure 3.2 illustrates the point. On the left is the original set of examples $x_1, \ldots, x_6$, described by attributes $y_1, \ldots, y_6$; on the right is the "reduced" training set obtained by the removal of three attributes and four examples. The combined application of *data editing* and *feature selection* is called k-NN tuning.

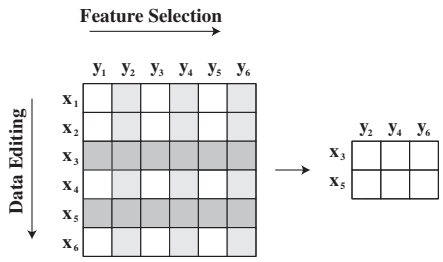

Figure 3.2: Simultaneous feature selection and data editing (i.e. k-NN tuning).

k-NN tuning is in essence a multi-objective optimization problem that seeks to maximize classification accuracy while minimizing the number of examples and the number of attributes (i.e. classification costs). Let N and m, respectively, denote the number of examples and attributes in a data set. Horowitz et al. (1997) showed that the task can be cast as a binary optimization problems with $N + m$ boolean decision variables and an NP-hard search space of 2^{N+m} elements. Exponentially growing costs being deemed prohibitive, scientists have searched for suboptimal, though acceptable, solutions. Many powerful techniques have been proposed (Ho, Liu, and Liu

2002; Llora and Guiu 2003; Kuncheva and Bezdek 1998; Raymer, Punch, Goodman, Kuhn, and Jain 2000; Hart 1968). This research work builds on the promising results reported by several authors who have experimented with various versions of the GA. A survey of these studies is presented next.

3.3.1 Classical Approaches to 1-NN Tuning

The Machine Learning literature has reported numerous techniques for both feature and example selection, the oldest dating back to the 1960's.[20] The simplicity of these classical non-GA-based approaches makes them popular even today, and they often serve as benchmarks in comparisons (Rozsypal and Kubat 2003; Kuncheva ; Quirino and Kubat 2010). As for example selection, the Wilson's Edited Nearest-Neighbor (E-NN) (Wilson 1972) and Hart's Condensed Nearest-Neighbor (C-NN) (Hart 1968) are the most famous. These techniques have inspired dozens of other more up-to-date approaches (Angiulli 2005; Cano, Herrera, and Lozano 2003; Kuncheva and Jain 1999).

As for attribute selection, a representative approach is Sequential Forward Selection (SFS) (E. Cantu-Paz 2004), a greedy-search approach that starts with an empty set of attributes and, in successive iterations, adds attributes that appear to be best at improving the classifier's accuracy. The opposite approach, Sequential *B*ackward Selection (SBS), starts with a complete set of attributes, and gradually removes those that appear to be irrelevant. Another famous approach is the C4.5 Decision Trees

[20]The work in (Cano, Herrera, and Lozano 2003) presents a good overview of various heuristic and genetic approaches to example selection.

program developed by Quinlan (1993), that is known be very good at distinguishing between relevant and irrelevant attributes in data sets.

3.3.2 1-NN Tuning Solved By Genetic Algorithms

Classical methods usually separated example selection from attribute selection. It was only in the first attempt to employ the GA, that Kuncheva and Jain (1999) combined the two tasks. In their GA, each specimen was described by binary chromosomes, where each of the first N_E bits represented the presence ("1") or absence ("0") of the corresponding example (total N_E examples), and each of the last N_A bits represented the presence ("1") or absence ("0") of the corresponding attribute (total N_A attributes). Every chromosome thus defined a 1-NN classifier that used the examples and attributes labeled with "1." The fitness-function rested on the classification performance against training set size.

The success of this approach inspired further research. Thus Ho et al. (2002) improved the performance of this early solution by the use of the *Intelligent Crossover* operator that employs orthogonal arrays and factor analysis to measure and quantify the contribution of each individual gene to a specimen's resulting fitness. This is then used to select for crossover those genes that are likely to contribute more than others. The approach was shown to outperform its immediate predecessor as well as some non-genetic techniques.

Ishibuchi and Nakashima (2000) proposed further improvement in their *HT-GA*. They experimented with various parameters of the GA, including varying mutation and fitness-functions. The most notable aspects are the use of different mutation

rates for the "0" or "1" bits in a chromosome. In their experiments, the resulting 1-NN classifiers outperformed 10 other classification techniques.

Another improvement was proposed by Rozsypal and Kubat (2003) in their *RK-GA*, which used a pair of variable-length chromosomes; one for examples, the other for attributes. This encoding scheme made the chromosomes less costly to handle, especially with large data sets. Experiments showed that the technique led to smaller sets of examples and attributes without impairing classification performance. In this work, an improved version of *RK-GA*, the baseline *RK-GA*$_0$, was implemented and used as the baseline GA for all experiments with Testbed Problem 1.

While the previous approaches relied on the weighted sum approach to combine the multiple objectives of the 1-NN Tuning problem in their fitness-functions, Chen et al. (2005) employed the Pareto-improvement approach (detailed in Section 3.2.1) in their GA, the IMOEA. IMOEA curtailed the subjectivity of the weights in the weighted sum approach (Coello 1999).

As for practical applications, the approach proposed by Cheatham and Rizki (2006) used GA-tuned k-NN classifiers for text selection in web search. The fitness-function combined classification accuracy with the size of the example set. Experiments showed improvement over classical approaches as well over GA-based random search. The authors also asked whether simultaneous optimization of example and attribute sets outperformed their sequential optimization. Interestingly, in their particular domain, simultaneous search did not seem better. Apparently, certain domains (in this case, the large number of attributes making classes almost linearly separable), may call for different techniques.

Some authors sought to demonstrate that GA can bring about improvement over classical non-genetic approaches even when focusing only on one of the two aspects: example selection or attribute selection. Sometimes, this makes sense. The attributes could have been selected by knowledgeable experts, but the examples could each have a different reliability. Alternatively, examples may be sparse relative to the number of attributes of which many can be irrelevant.

Focusing on example selection, Gil-Pita and Yao (2007) used crossover operator and mutation based on *clusters* of examples (rather than on individual examples), the idea being that an example's performance is better captured when it is associated with other examples surrounding it in the instance space. The clusters are created in every generation by the k-means algorithm. Crossover mixes clusters of examples, and mutation is performed only in the example (within the given cluster) that most improves the average fitness. In experiments on UCI data sets, the approach outperformed classical heuristic approaches, the price being the added overhead of k-means clustering.

Focusing on attribute selection, Soryani and Rafat (2006) used a single-population GA to reduce the dimensionality of large attribute sets in the field of optical character recognition in Farsi, a problem characterized by a large number of attributes. As the fitness-function, they used the k-NN classifier's accuracy in matching individual character patterns. The GA-tuned attribute sets yielded better recognition rates at lower computational costs than earlier OCR techniques.

Similarly, the MGAFS (H. Zhu 2006) introduced a multi-population GA that improved the k-NN accuracy through attribute selection. Two populations are ran-

domly initialized, each with a different set of attributes. Population 1 initially selects more attributes as relevant than population 2. In the course of the GA run, relevant attributes replace less relevant ones, depending on the fitness-function which measures the k-NN accuracy. In migration, the best specimen of one population replaces the worst specimen of the other. In experiments, MGAFS outperformed the single-population GA.

k-NN Tuning as a Binary Optimization Problem

Other approaches formulated k-NN tuning as a binary optimization problem. For problems of this kind, a few GA techniques have been proposed. The most relevant are *CHC* (Eshelman 1991) and *PBIL* (Cano, Herrera, and Lozano 2003) that have been employed for attribute selection (S. Chen 1999) and example selection (Cano, Herrera, and Lozano 2003).

In *CHC*, the fitness-function is used only in survival decisions; mating partners are chosen at random. The program measures the Hamming distance between subsequent pairs, and only those with distance above $L/4$ (where L is the chromosome length) are allowed to mate. Recombination is carried out by a mechanism that randomly exchanges exactly half of the bits in which the two mating chromosomes differ. The program also has "incest prevention" mechanism that reduces the danger of premature convergence. *CHC* does not apply mutation to the generated children. Instead, when the population converges to a local optimum (which coincides with the mating threshold reaching 0), the program restarts the population by retaining a copy of a "template specimen" and generating the remaining specimens by mutating this

template. When applied to the 1-NN tuning problem, *CHC* outperformed other GA-based and non-GA-based techniques (S. Chen 1999; Cano, Herrera, and Lozano 2003).

PBIL (Cano, Herrera, and Lozano 2003) does not generate an actual population of chromosomes. Instead, it initializes a probability vector whose length is equal to the number of decision variables, and initializes its values to 0.5 (every bit has the same chance of being 0 or 1). At each generation, *PBIL* creates a population by a probability distribution determined by this vector. The vector is then updated by being "pushed toward" the best solution and "pushed away" from the worst solution. Adding small random values then mutates the probability vector. As a result, *PBIL* keeps explicit statistics about the search space. These are then used to decide where to sample next. In the experiments reported by Cano et al. (2003), *PBIL* compared favorably with other approaches on example selection.

3.4 What is the Optimal Decision Forests Problem?

This section discusses Testbed Problem 2, the Optimal Decision Forests problem, a well-known and complex optimization problem from the field of ensemble learning. The Optimal Decision Forests problem inherits the optimization objectives of its parent problem, the Optimal Classifier Ensemble problem (Chandra and Yao). The Optimal Decision Forest problem consists of the simultaneous optimization of (1) the example and attribute sets used to induce decision trees (i.e. maximize classification accuracy while minimizing classification costs) and (2) the grouping of the induced

decision trees into diverse ensembles, where the classification behavior of the underlying classifiers disagree as much as possible (Hyafil and Rivest 1976; Murthy and Salzberg 1995; Chikalov 2011).

The process of building decision forests requires the use of a decision tree inducer. The C4.5 Decision Trees program (hereinafter "C4.5") developed by Quinlan (1993) is one of the most popular tools used for this purpose (Skurichina and Duin 2002). C4.5 induces decision tree models by recruiting some of the most relevant attributes in a training data set. At each tree node, starting at the root node, C.45 recursively splits the training data set on the most informative attribute. The choice of the attribute is based on the *information gain ratio* criterium. Numerical attributes are handled by discretizing them at each tree node using a simple binary-split. A *user-defined* threshold for the minimum number of examples per node controls the induction process, which ends when the training data set cannot be split any further. C4.5 is known to induce decision trees having low bias and high variance. This feature makes it an ideal decision tree builder for use with ensemble generation techniques. However, the training data set quality is known to have a strong effect on the quality of the decision trees induced by C4.5. Classification costs are high if there are too many training examples, and noise in the class labels and/or attribute values, as well as irrelevant attributes, can mislead the classifier (Hall and Smith 1998; Ho 1998a). These issues can be mitigated by the removal of harmful examples and noisy/irrelevant attributes from the training data set prior to inducing a decision tree.

The Optimal Decision Forest problem is in essence a multi-objective optimization problem; the goal of maximizing the classification accuracy of the individual deci-

sion trees *conflict* with the goal of minimizing their overall classification agreement (the latter implies high classification errors). Let N and m, respectively, denote the number of examples and attributes in a data set. Also, let S denote the maximum allowable size of the ensemble of decision trees in a particular problem. The task can be cast as a binary optimization problems with $s \cdot (N + m)$ boolean decision variables and an NP-complete search space of $2^{s \cdot (N+m)}$ elements (Lu, Wu, Zhu, and Bongard 2010; Chandra and Yao). Compared to 1-NN Tuning problem (Testbed Problem 1), whose search space is in the order of 2^{N+m}, this problem is exponentially more complex. Notice how the search costs grow exponentially, making brute-force search for optimal decision forests simply prohibitive. As a result, scientist have searched for suboptimal, though acceptable, solutions. Many powerful techniques have been proposed over the years to generate accurate decision forests (Breiman 1996; Freund and Schapire 1996; Breiman and Schapire 2001; Ho 1998b; Rokach 2008; Hu, Yu, and Wang 2005). However, more can be done to reduce their classification costs. This work builds on the promising results reported by several authors who have experimented with various versions of both GA-based and non-GA-based techniques for decision forests generation. A survey of these techniques is presented next.

3.4.1 The Role of Classifier Bias/Variance Trade-off In Ensemble Generation

The concept of classifier bias/variance trade-off that was discussed in Section 3.1.2 forms the basis for the theory of *classifier ensemble generation* (Oza and Tumer 2008; Rokach 2010). Recall from the discussion in Section 3.1.2 that a good classifier should

exhibit both low bias and low variance in order to achieve good performance. That is, a classifier should on average be correct (i.e. low bias) while the structure of its model should remain stable from one training set to another (i.e. low variance). This is an idealized requirement, because in practice, classifiers do not simultaneously exhibit both low bias and low variance. Instead, there is always a trade-off between bias and variance in classifiers. For example, classifiers having low bias (i.e. having low average classification error) also exhibit high variance because they require flexibility to fit their models to different training data sets. In contrast, classifiers having high bias (i.e. having high average classification errors) also exhibit low variance because their models do not change significantly to fit different training sets, leading to systematic classification errors.

The primary objective of ensemble generation techniques is to reduce the variance of classifiers by "averaging" the predictions of many classifiers. This prediction averaging process is usually achieved by some form of "voting scheme", where all classifiers in an ensemble predict a class label and the class attaining the majority of the votes is considered the "winner". Voting is a form of averaging, and averaging reduces variance. This principle was clearly illustrated in the example given in **Figure 3.1** of Section 3.1.2. Compare the top-left and top-right plots of **Figure 3.1**. The top-left plot gives the true function being estimated (solid line) along with 30 estimates (dashed lines) produced by fitting of a 5_{th}-degree polynomial (hereinafter "polynomial estimator") to 30 data sets randomly sampled from the true function (the simulated sampling noise is depicted in the bottom-right plot). Similarly, the top-right plot also shows the true function (solid line), but this time overlayed with

the "average" function estimated by averaging the 30 estimates of the polynomial estimator (dashed line). Notice in the top-left plot how the 30 individual estimates of the polynomial estimator (dashed lines) vary significantly around from the true function (i.e. high variance). However, the top-right plot reveals that the "average" of the 30 individual estimates is much more "well-behaved" and also a much closer estimate of the true function.

The illustration in **Figure 3.1** depicts exactly what ensemble generation techniques seek to achieve. An ensemble classification system resulting from the "averaging" (or voting) of multiple *slightly different* classifiers has a lower variance than that of of the individual classifiers in the ensemble. In the top-left plot of **Figure 3.1**, the differences in prediction behavior among the 30 estimates of the polynomial estimator (i.e. its variance) are due to their slightly different training data sets result from a random data sub-sampling process. The differences in predictive behavior among the various estimates of the true function is an important property of ensemble systems. For example, the 30 different estimates must be able to "complement" each others' predictive biases in order for their "average" function to improve over any single estimate. This key property is known as "ensemble diversity". Furthermore, ensemble classification systems inherit the bias of their underlying classifiers, as illustrated in the top-right plot of **Figure 3.1**. Thus, if an ensemble is generated from a classifier having low bias and high variance (i.e. such as the decision trees induced by the C4.5 Decision Trees program), the resulting ensemble classification systems will exhibit both low bias, the inherited property, as well as low variance, which is attained through the voting (or "averaging") process. This explains why

C4.5 is one of the classifiers most frequently used for ensemble generation (Skurichina and Duin 2002). In summary, ensemble classification systems exhibit the ideal properties of a good classifier; both low bias and low variance. The model search space of ensemble classification systems encompasses that of single classifiers and beyond! Ensemble classification systems are capable of inducing *hypotheses* about the underlying properties of unknown statistical process that single classifiers cannot possibly achieve (Zhou 2009; Oza and Tumer 2008).

Bagging (Breiman 1996), AdaBoost (Freund and Schapire 1996), Random Subspace (Ho 1998b), and Random Forests (Breiman and Schapire 2001) are classical ensemble generation techniques built upon the principle of variance reduction described above. These techniques are both powerful and simple to implement, which make them popular tools for use in numerous applications and also as benchmarks in comparisons. These classical techniques, and others, are discussed in the subsequent sections.

3.4.2 The Role of Classifier Diversity in Ensemble Generation

The key to the success of ensemble generation techniques is the ability to build both *accurate* and *diverse* classifiers (Parvin, Alizadeh, and Bidgoli 2009; Lu, Wu, Zhu, and Bongard 2010). Diversity is the ability of the classifiers making up an ensemble to "complement" each other's errors in order to achieve higher predictive power (Kuncheva 2003). When applied to classification problems, this means that the classifiers making up an ensemble should have *orthogonal* (or independent) errors

with respect to each other. As discussed in Section 3.4, accuracy and diversity are conflicting optimization objectives that ensemble generation techniques must optimize simultaneously to create good ensembles.

Introducing diversity into ensembles is not a straight-forward process, the main reason being that the concept of ensemble diversity itself is not well-defined in the Machine Learning literature (Kuncheva and Whitaker 2003). Over the years, researchers have developed a few different approaches to introduce diversity into ensembles of classifiers. The earliest approaches to ensemble generation introduced diversity into their ensembles by injecting randomness into the choice of input parameters used to build the underlying classifiers (Gunter and Bunke 2004). For example, in Bagging (Breiman 1996) and AdaBoost (Freund and Schapire 1996), different classifiers are generated by randomly resampling (with replacement) subsets of examples from the original training set. In contrast, in the Random Subspace (Ho 1998b) and Random Forests (Breiman and Schapire 2001) techniques, the choice of the attributes used to build the classifiers are randomized instead. These technique proved to be very efficient in creating accurate ensembles over single classifiers.

Another common approach to introducing diversity in classifier ensembles is called "ensemble pruning". Ensemble pruning is a process based on an *overproduced-and-select* paradigm in which a pool of classifiers is first built using an existing ensemble generation method (i.e. Bagging or AdaBoost) and then a subset of the most accurate and diverse built classifiers are selected to create and ensemble (Kuncheva 2003; Rokach 2010). Ensemble pruning is an optimization problem, and it can be seen as a

simplified, or one-sided, version of the Optimal Classifier Ensemble problem described in Section 3.4.

In ensemble pruning methods, the classifier selection process is generally guided by a *user-defined* ensemble diversity measure. Over the years, researchers have proposed various diversity measures to quantify ensemble diversity. Existing measures are generally sub-divided into two groups: 1) pairwise measures, and 2) ensemble-based measures (Kuncheva and Whitaker 2003). Pairwise measures rely on the computation of some distance metric between the pairs of classifiers making up an ensemble. The results are then averaged over the total number of possible classifier pairs. The *plain-disagreement* and *double-fault* measures are representative examples of pairwise diversity measures. Given a pair of classifiers, the *plain-disagreement* (also known as the "average pairwise Hamming distance") measures the probability of two classifiers making different predictions on the same example. The *double-fault* measures the probability of both classifiers being incorrect. The *disagreement* measure is maximized, while the *double-fault* measure is minimized, during the ensemble pruning process. Other popular pairwise measures include the Q-statistics and the correlation coefficient (Tsymbal, Pechenizkiy, and Cunningham" 2005). As for ensemble-based diversity measures, which use metrics based on the distribution of votes to different classes on a per instance basis, the most popular measures are the *kappa* and the entropy measures (Kuncheva and Whitaker 2003; Kuncheva 2003).

3.4.3 Classical Approaches to Optimal Decision Forests

Approaches to ensemble generation date back to the 1960's, with numerous approaches having been developed over the years. Bagging (Breiman 1996), AdaBoost (Freund and Schapire 1996), and the Random Subspace (Ho 1998b) methods are amongst the most representative classical techniques. A major advantage of these techniques is that they are "classifier agnostic" (i.e. they work with all classifiers), although they were primarily built for use with low-bias/high-variance classifiers such as decision trees (Skurichina and Duin 2002). A major drawback of classical ensemble generation techniques is that the gains attained in classification accuracy come at the cost of lower interpretability of the results (Breiman 1996). This occurs because the classical techniques require large ensemble sizes to work adequately, making results very hard to interpret.

A common trait among classical techniques is in the way they introduce diversity into their ensembles. Classical techniques build classifiers by randomly resampling from their training pool of input parameters (Gunter and Bunke 2004). For example, some classical techniques build classifiers using randomly resampled examples from a training data set. Other techniques retain all examples but randomly resample the attributes from the training set. In either case, implementing randomized sampling for either examples or attributes is an easy task, which has contributed to the popularity of classical ensemble generation techniques.

Some of the classical techniques preferred to introduce diversity into their ensembles by injecting randomness into the example selection process. For example,

given a training data set with N_{ET} examples and a *user-defined* ensemble size k, Bagging (Breiman 1996) builds an ensemble of k classifiers by randomly resampling (with replacement) k new bootstrap training sets (each having N_{ET} examples) from the original training set. On average, each of the k bootstrap training sets will contain approximately $0.63 \cdot N_{ET}$ unique examples, although they each contain a total of N_{ET} examples. Bagging then subsequently builds a different classifier using each of the k bootstrap training sets. A simple majority voting scheme is used to combine the predictions of the k generated classifiers. Bagging is known to work well with classifiers having high variance.

AdaBoost (Freund and Schapire 1996) also promotes ensemble diversity through the injection of randomness into the example selection process. However, there are two major differences between AdaBoost and Bagging. First, AdaBoost builds each of its k classifiers sequentially using a weighted random resampling (with replacement) scheme. In the first iteration, every training example is assigned an equal selection probability of $\frac{1}{N_{ET}}$. In subsequent iterations, examples that were misclassified by the classifier generated in the previous iteration have their weights increased, while all other examples have their weights decreased. The process is repeated k times to generate k classifiers. The second difference is that AdaBoost uses a weighted voting scheme, in which the class that attains the maximum weighted vote over the k classifiers is considered the "winner". The classification weight of a generated classifier is proportional to its accuracy on the weighted training data set presented to it. This is a contrast to Bagging's simple majority voting scheme. AdaBoost outperformed Bagging in experiments with UCI data sets and various different classifiers.

Other classical techniques preferred to generate diversity in ensembles by injecting randomness into the feature selection process. The Random Subspace (Ho 1998b) technique is a representative approach that is also referred to as "attribute bagging". The concept is simple. To generate an ensemble of k classifiers, the user defines the number N_a of attributes to be used to build each classifier. At each of the k iterations, Random Subspace randomly selects a subset of N_a attributes from among the original ones to build a classifier. In contrast to bagging and AdaBoost, all examples are used to build the classifiers in each iteration. The predictions of the k generated classifiers are combined through majority voting, or in the particular case of decision trees, by averaging the conditional probability of each class at the leaves. Decision forests generated by Random Subspace outperformed those generated by both Bagging and AdaBoost.

In addition, various classical techniques were designed specifically for the task of inducing decision forests. Random Forests (Breiman and Schapire 2001) is a representative approach that builds upon ideas from both Bagging and Random Subspace. To generate an ensemble of k classifiers from N_{ET} training examples, the user defines how many N_a attributes will determine the decision at each node of the generated decision trees. At each of the k iterations, Random Forests creates a bootstrap training set with N_{ET} examples by resampling (with replacement) from the original training set. An unpruned decision tree is then induced from the bootstrap training set. At each node of the induced decision tree, N_a attributes are randomly chosen and the best split is determined from among the chosen attributes. A simple majority voting

scheme is used to combine the predictions of the k generated decision trees. Random Forests outperformed AdaBoost in experiments with UCI data sets.

A more recent technique is Rotation Forest (Rodriguez, Kuncheva, and Alonso 2006), which relies on Principal Component Analysis (PCA) to generate diverse feature spaces from a training set in order to train diverse decision trees. Similar to Bagging, Rotation Forests trains each classifier independently by resampling (with replacement) k new bootstrap training sets from the original training set. For each of the k classifiers, Rotation Forest randomly splits the feature space into s subsets. PCA is the applied independently to each of the s subsets. The resulting principal components are regrouped to form a new feature space, the corresponding bootstrap training set is projected into the new feature space, and a decision tree is built. The advantage of this approach is that different subsets of the original feature space will lead to different projected feature spaces, thus promising more diverse ensembles. Rotation Forest outperformed Bagging, AdaBoost, and Random Forests in experiments with UCI data sets.

3.4.4 Optimal Decision Forests Solved By Genetic Algorithms

Various research works reveal that GA-generated decision forests work better than existing classical techniques for ensemble generation (Rokach 2008; Hu, Yu, and Wang 2005; Podgorelec and Kokol 2001). While classical techniques for ensemble generation generally rely on single or limited number of *hypotheses* to generate classifiers, the GA is able to search through vast search spaces of classifier configurations to discover optimal ones. This is a major advantage of GA-based techniques over classical ones.

However, a survey of the Machine Learning literature reveals that very little attention has been given to the idea of inducing *optimal* decision forests; being simultaneously accurate and compact. Instead, the trend among existing GA-based research works has been to either (1) optimize individual decision trees and then group the resulting trees into an ensemble, or (2) induce a set of decision trees using an existing technique and then apply the GA to to select an optimal subset that maximizes ensemble accuracy. In addition, existing works focusing on the optimization of decision trees usually do so through GA-based feature selection, ignoring the presence of noisy/redundant examples in data sets. This hinders the optimization of the decision tree classifiers, which are sensitive to this type of data noise, as discussed in Section 3.1.2. The trends found in current research works indicate that much more remains to be done to optimize the performance of decision forest classifiers.

The ensemble generation technique proposed by Rokach (2008) provides an example of GA-based feature selection used in the generation of accurate and diverse decision trees. The GA is used to discover multiple disjoint "reducts" (i.e. compact and informative subsets of training attributes) that are used to train the decision trees using all examples in the training set. Specimens in the GA are represented by strings of integers whose lengths are equal to the number of attributes in the training set. Each element in a string corresponded to a training attribute and the value in the each element corresponds to the index of a "reduct" that the corresponding attribute is to be placed into. The fitness-function rests on the accuracy of the decision tree classifiers induced by the GA-generated reducts. The ensemble size is automatically set by the final number of GA-generated disjoint reducts. The technique outperformed

AdaBoost, C4.5, Naive Bayes, and other classical ensemble generation techniques on various experiments with UCI data sets.

Similarly, the technique proposed by Podgorelec and Kokol (2001) also uses the GA to build optimal decision trees, but the algorithm randomly creates tree structures of a user-specified depth and lets the GA optimize the choice of attributes used at each test node. The fitness-function weighted classifier accuracy against the importance of the chose decision nodes. The technique then combines the GA-generated trees into a decision forest. The experimental results revealed that the GA-generated decision forest outperformed the single decision tree.

The technique proposed by Hu et al. (2005) is an example of GA-based classifier subset selection. This technique uses an external reducer to extract a user-specified number of "reducts" from a data set. The reducts are used to train decision trees and the GA is used to select a subset of the trees that maximizes ensemble classification accuracy. In particular, the technique adopted binary strings to represent specimens. Each bit indicated the presence or absence of a classifier in the GA-generated ensemble. The fitness of each solution rested on the classification accuracy of the induced decision forest. The technique was found to outperform other classical reduct-based ensemble generation techniques.

CHAPTER 4

Speeding Up the Genetic Search

This chapter presents the five (5) novel "instinct-based" mating strategies designed to optimize genetic search. These strategies are designed to speed up the GA without impacting the quality of the generated solutions, while requiring minimal additional computational overhead. The proposed mating strategies are sophisticated mating strategies based on the Darwinian evolutionary principle of "opposites attract" that is commonly observed in nature. A total of five (5) different "instinct-based" mating strategies are presented; four of them as single-population[21] GA mating strategies and one as a multi-population[22] GA mating strategy.

Given the nature of this work, an intuitive questions arises: why seek to improve upon the conventional mating strategy if it already works? The conventional mating strategy, which has been used for decades in implementations of the GA, is known to contribute to premature convergence in genetic search. First, the conventional mating strategy is based on *random* selection of partners using probability distributions

[21]The single-population GA uses one single population of specimens, or a single pool of potential solutions, to solve an optimization problem.

[22]The multi-population GA uses multiple populations of specimens, or multiple pools of potential solutions, to solve an optimization problem. Generally, different populations evolve independently with periodic interbreeding through exchange of specimens among the population.

implied by the fitness-function,[23], which means that individual specimens in the population have no choice on their selection of mating partners. Second, the conventional mating strategy tends to allow the most "fit" specimens in the population to "mate" more often. This eventually leads the entire population to be *overrun* by some highly-fit specimens. At this stage the the genetic search ends, unable to further improve the quality of the solutions. ***The "instinct-based" mating strategies proposed in this work improve over the conventional mating strategy by "endowing" specimens with "mating instincts" that promote a more diverse pairing of mating partners and helps mitigate the issues of premature convergence in genetic search***. The proposed "instinct-based" mating strategies are collectively referred to by the acronym *IM-GA*, which stands for Instinct-Based Mating in Genetic Algorithms.

The organization of this chapter is a follows. First, Section 4.1 introduces the theory behind *IM-GA* — the inherent issues with the conventional mating strategy are discussed, followed by a description of how *IM-GA* was designed to effectively deal with those issues. Next, a discussion follows on the choice of testbed problems. This is a key topic because the definition of what constitutes "useful" mating instincts in *IM-GA* is problem-dependent. Following the discussion of the choice of testbed problems is the definition of required measures used to implement the actual "mating instincts" in the *IM-GA* mating strategies. Next, Section 4.2 describes the *IM-GA* Strategies 1 through 4, the single-population GA mating strategies. Then, *IM-GA* multi-population GA mating Strategy 5 is described in detail in its own Section 4.3.

[23]See Section 3.2 for a detailed overview of the GA

Following that, Section 4.4 discusses Testbed Problem 1, the 1-NN Tuning problem, and details how the *IM-GA* Strategies 1 through 5 are implemented in the GA mating process to optimize genetic search in this problem. Finally, Section 4.5 describes Testbed Problem 2, the Optimal Decision Forests problem, and details how the *IM-GA* Strategies 1 through 5 are implemented in the GA mating process to optimize genetic search in this second problem. The goal is to demonstrate that the new *IM-GA* mating strategies improve genetic search in different problems.

4.1 Instinct-Based Mating in Genetic Algorithms (*IM-GA*)

The five *IM-GA* "instinct-based" mating strategies presented in this chapter were designed is to promote faster *adaptation* (or improvement) of solutions in the GA population in every iteration of the genetic search process, without impacting the long-term quality of the solutions.

As was discussed in the Introduction, improving genetic search requires mitigating the issue of information loss rate that is inherent in the genetic search process. Optimizing the mating process is a key step toward mitigating this issue. That is because mating is the process that determines how the available information in the GA population is used to generate new solutions. Unfortunately, the tendency of the *conventional mating strategy* is to exacerbate the issue of information loss rate by promoting highly-fit specimens to eventually *overrun* the GA population after multiple iterations of the genetic search process. In the conventional mating strategy, specimens have no choice on the selection of their mates. They are simply paired

randomly. This is not always true in nature. In nature, the principle of "opposites attract" is common to many species, where specimens will tend to mate with those that "complement" them on their own abilities to survive and generate better offsprings (ScienceDaily 2009; Shamir, Saad, and Marom 1993).

The proposed *IM-GA* "instinct-based" mating strategies improve over the conventional mating strategy by implementing the natural principle of "opposites-attract" in the GA mating process. The *IM-GA* mating strategies "endow" specimens with "mating-instincts" that guide them in their selection of partners that "complement" them on their own abilities to optimize some problem dependent optimization criteria. This promotes the pairing of specimens having more diverse information content, consequently improving the utilization of the information available in the GA population for recombination, and leading the GA to more efficiently and diversely sample the search space of possible solutions to optimization problems.

Since the definition of "mating-instincts" is problem-dependent, this section reviews the choice of testbed problems adopted in this work to evaluate the proposed ideas. After defining the "mating-instincts" according to the choice of testbed problems, the subsequent sections will then show how these "mating-instincts" were used to implement the five proposed *IM-GA* mating strategies.

4.1.1 Defining Useful Instincts

As was discussed in Section 1.2, one of the main challenges involved in applying "instinct-based" mating in Genetic Algorithms is that of defining what constitutes "useful" mating instincts in different problem domains. In contrast, this is not an

issue in applications of the conventional mating strategy, where specimens do not a have a choice in the selection of their mating partners. Defining "useful" mating instincts is a key issue in *IM-GA* because of two main reasons: 1) the definition will vary greatly from one problem domain to another, and 2) because the performance of the GA is sensitive to the choice of mating instincts.

In particular, the domain of supervised classification offers numerous optimization problems under which useful "mating instincts" can be very intuitively defined. Many of the optimization problems in the domain of supervised classification involve optimizing the performance of classifiers (i.e. improving classification accuracy, building more compact models to reduce training and/or classification costs, automatically identifying the most relevant attributes and examples in data sets to build better classifiers, etc). In applications of the GA to such problems, the general paradigm is to let the specimens in the GA population represent different classifiers, where each classifier constitutes a possible "optimal" solution to the problem under investigation (Rozsypal and Kubat 2003; Quirino and Kubat 2010; Ishibuchi and Nakashima 2000; Kuncheva and Jain 1999; Soryani and Rafat 2006). Furthermore, each of the classifiers are trained with a different set of input parameters, i.e. different training sets and/or program argument choices. These input parameters are "encoded" in the one or more chromosomes of the corresponding specimen (see Section 3.2 for an overview of the GA). That is, each specimen in the GA population encodes the parameters needed to build a unique classifier in its chromosomes. Under this commonly adopted paradigm, it is simple to define as a useful "mating instinct" the ability of a classifier (or specimen in the GA population) to "complement" another

classifier's errors. In other words, in applications of the GA to problems in the domain of supervised classification, there is a natural connection between the use of "mating instincts" in the GA mating process and the optimization of classifiers. ***In this case, a "useful" mating instinct can be defined as the tendency of each specimen (or classifier) to mate with those specimens that "err" on different examples.***

What the above discussion establishes is that problems in the domain of supervised classification are suitable testbeds under which "instinct-based" mating strategies can be developed and their impact on the performance of the GA can be evaluated. Hence, due to this natural connection, this research borrowed two well-known problems from the domain of supervised classification to test the ability of the proposed *IM-GA* mating strategies to optimize the GA. These testbed problems are: 1) the 1-NN Tuning problem (hereinafter "Testbed Problem 1"), and 2) the Optimal Decision Forests problem (hereinafter "Testbed Problem 2").

Now, having discussed both the problem domain dependence of the choice of "mating instincts" and also the choice of testbed problems used to evaluate the *IM-GA* mating strategies, let us focus on the clarification of two important consequent questions:

1. What is the role played by these "mating instincts" in the selection of mating partners in the GA?

2. How does *IM-GA* actually implement the "mating instincts" in the GA mating process?

To answer these questions, first recall from the overview presented in Section 3.2 that the GA mating process consists of selecting pairs of specimens (hereinafter "parents") which are then recombined[24] to generate new specimens (hereinafter "children"). In the conventional mating strategy, both parents (hereinafter "first parent" and "second parent" in each parent pair) in each parent pair are selected randomly based on fitness (i.e. no partner selection choice is allowed). In contrast, in *IM-GA*, only the "first parent" of each parent pair is selected based on fitness (either randomly or as the top fit specimens in the population). Then, the "second parent" in each parent pair is selected by "endowing" the chosen "first parent" specimens in each pair with "mating instincts" that guide them in their choice of a mate.

These "mating instincts" are essentially *pairwise measures* computed between the chosen "first parent" in each parent pair and the remaining specimens in the GA population. For example, let the number of specimens in the GA population (the "population size") be N_P for an arbitrary optimization problem. For each chosen "first parent" in each parent pair, a total of $N_P - 1$ pairwise values of some chosen measure are computed between the "first parent" and all remaining specimens in the GA population. The resulting $N_P - 1$ measures are transformed into a probability distribution which is then used to randomly select the "second parent" in each pair.

In order for this type of mating partner selection scheme to effectively emulate the natural principle of "opposites attract" in the GA mating process, pairwise measures chosen to define "mating instincts" in *IM-GA* must be carefully formulated to

[24]Recombination in the GA is the process of generating new solutions to a problem from existing solutions. This consists of exchanging chromosomal information between pairs of specimens (parents) to generate new specimens (children).

adequately capture, numerically, the ability of each specimen in the GA population to "complement" every other specimen. In addition, recall that the choice of "mating instincts" is problem domain dependent. Therefore, in this work, any chosen pairwise measure must be somehow derived from criteria related to the domain of supervised classification (recall that Testbed Problems 1 and 2 are optimization problems from the domain of supervised classification). The measures should consider one or more of the following:

1. Criterium 1: The structural properties of the input data used to build a classifier (i.e. choice of examples and or attributes used to build a model);

2. Criterium 2: The classification behavior of a classifier (i.e. the properties of the classification error on a specific data set), and;

3. Criterium 3: The structural properties of the custom model built by a classifiers.

Here, there are numerous choices available for such custom, problem domain dependent, pairwise "mating instinct" measures. In this work, a total of four generalized pairwise measures were adopted. These chosen pairwise measures were derived from both Criterium 1 and 2 described above. Measures based on Criterium 3 were deliberately avoided in this work because this criterium entails the derivation of measures that are "classifier-dependent" as opposed to more general measures that are"problem-domain-dependent" (i.e. useful in numerous optimization problems across the domain of supervised classification). The four chosen pairwise measures are as follows (their detailed definitions are presented in the subsequent sections):

1. Mating Instinct 1: The Hamming distance[25] measure between the error vectors of a pair of classifiers;

2. Mating Instinct 2: The novel "*Correct-My-Wrongs*" distance measure, also derived from the error vectors of a pair of classifiers;

3. Mating Instinct 3: The Hamming distance between the sets of attributes retained by a pair of classifiers (i.e. the size of the *symmetric difference* between the sets of attributes retained by the pair of classifiers), and;

4. Mating Instinct 4: The Hamming distance between the sets of examples retained by a pair of classifiers (i.e. the size of the *symmetric difference* between the sets of examples retained by the pair of classifiers).

Notice how this particular choice of measures is not tied to any single optimization problem. Instead, they are sufficiently general to be used in numerous optimization problems across the domain of supervised classification. The analogy associated with this particular choice of pairwise measures is that of observing the input (i.e. training set properties) and response (i.e. classification error) of a classifier and using that information to make more "informed" decisions about the optimal selection of mating partners in the GA. Furthermore, notice that this choice of pairwise measures avoids any dependence on the diverse/custom structural properties of different classifiers (e.g. the 1-NN classifier, decision trees, and the SVM classifiers are all described by their own custom model structures). Instead, the chosen measures allow the GA

[25]The Hamming distance between a pair of binary ("0" or "1" valued) strings is the number of corresponding positions in both strings that differ.

to virtually treat any classifier being optimized as a "black box" (i.e. only observations about the properties of a classifier's input and response variables are required). These feature simplify the implementation of the proposed *IM-GA* mating strategies into the GA mating process because no knowledge about a classifier's internal model structure is required. ***The proposed IM-GA mating strategies are thus capable of adapting to numerous optimization problems across the domain of supervised classification.***

Regarding the use of the four chosen pairwise measures in the five proposed *IM-GA* mating strategies, each of the four single-population *IM-GA* strategies adopts one of the pairwise measures as their principal "mating instinct." In contrast, the fifth proposed strategy, the multi-population *IM-GA* strategy, adopts three measures and employs each of them as the principal "mating instinct" of a sub-population of specimens. **This is a key difference between the single versus multi-population implementation of the *IM-GA* "instinct-based" mating strategies**. In the single-population case, specimens are "endowed" with the same "mating instinct," representing the optimization of a single objective. In contrast, that is not a restriction in the multi-population case, where the optimization of multiple objectives through the mating process can be simultaneously pursued by "endowing" specimens in each sub-population with a different "mating instinct."

The following sections detail how the four chosen "mating instinct" pairwise measures are actually computed from pairs of specimens in the GA population. The discussion starts with the definition of the classification "error vector" of a classifier, which is a key instrument in the derivation of Mating Instinct 1 and 2 listed

above. The detailed derivation of the four "mating instinct" pairwise measures are then presented.

4.1.2 The Classification Error Vector

Used in Measure 1 (Hamming distance) and Measure 2 (*Correct-My-Wrongs*), the classification error vector (hereinafter "error vector"), describes the ability of a classifier to correctly assign category labels to every example in a validation set. During the GA run, the training set itself is used as the validation set needed to compute the error vectors of the GA-generated classifiers. Each specimen in the GA population represents a different classifier, or a different potential solution. That is because each specimen contains one or more chromosomes that encode a unique set of input parameters used to build a unique classifier. Because of this *one-to-one* relationship between a specimen and its corresponding classifier, the term "classifier" and "specimen" will be used interchangeably hereinafter. Now, let us assume that a training set has N_{ET} examples. For each specimen in the GA population, a binary ("0" or "1" valued) error vector having length N_{ET} is created. Each element in a particular specimen's error vector corresponds to an example in the training set. When a training set example is misclassified by a given specimen (i.e. by the classifier built from its chromosomes), a value of "1" is assigned to that example's corresponding location in the specimen's error vector. Conversely, if the given example is correctly classified, a value of "0" is assigned. **Figure 4.1** illustrates the error vector of an arbitrary specimen. Notice from this illustration that the first, third, and last elements in the error vector are set to "1." This indicates that the corresponding examples were

misclassified by the specimen (i.e. misclassified by the unique classifier built from the data in its chromosomes):

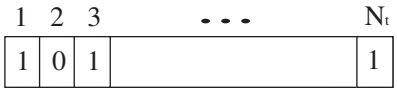

Figure 4.1: Sample binary error vector ("0" or "1" valued) of a classifier.

The computational costs of time and storage associated with the computation of classifier error vectors are negligible in both Testbed Problems 1 and 2. First, computation of the error vectors do not require any additional training set evaluations beyond those that are already commonly done for fitness evaluation purposes. This is because the classification accuracy is an optimization objective in the two testbed problems, and thus, the GA-generated classifiers are already evaluated on the training data set for the purpose of fitness evaluation. Second, computing the error vectors require only *computationally cheap* boolean operations to compare the class labels *assigned* to the training set examples by a GA-generated classifier versus the training set examples' *true* class labels. As discussed in Sections 4.4.3 and 4.4.5 for Testbed Problem 1, and Sections 4.5.2 and 4.5.4 for Testbed Problem 2, the error vector computational costs are negligible when compared to the costs of inducing and evaluating the GA-generated 1-NN and decision trees classifiers.

4.1.3 Proposed Mating Instinct 1 - The Hamming Distance Between Error Vectors

The first "mating instinct" (Mating Instinct 1) is designed using the Hamming distance between error vector pairs. **Figure 4.2** illustrates the Hamming distance computation between a pair of arbitrary error vectors **A** and **B**:

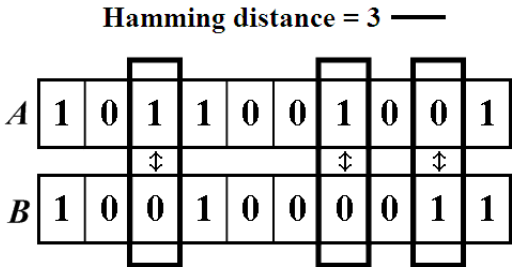

Figure 4.2: Sample Hamming distance calculation for a pair of error vectors **A** and **B**.

Then, once the error vectors are computed for all specimens in the GA population, a symmetric distance matrix is created using the computed Hamming distance values between every pair of error vectors. Let the number of specimens in the GA population for an arbitrary optimization problem be N_P. Then, the generated $N_P \times N_P$ distance matrix is referred to as the *Complementary Distance Matrix*. In this matrix, the field in the i-th row and j-th column gives the Hamming distance between the error vectors of the i-th and j-th specimens, respectively. This indicates how well the two specimens "complement" each other in terms of their classification errors, a criterium that naturally fulfills the principle of "opposites attract" that inspired this research. The matrix is symmetrical, with zeroed out main diagonal.[26] **Figure 4.3** gives an example of a distance matrix for an imaginary population of $N_P = 6$ specimens and

[26]This is because the Hamming distance computation is symmetric.

is populated with arbitrary numbers. The illustration shows the general form of a distance matrix: symmetric and with zeroed out diagonal:

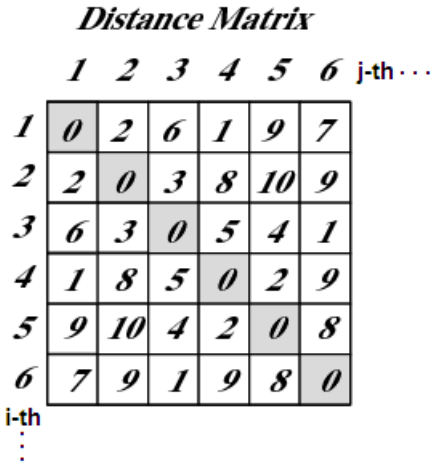

Figure 4.3: Sample Complementary Distance Matrix for six specimens.

4.1.4 Proposed Mating Instinct 2 - The "*Correct-My-Wrongs*" Distance Between Error Vectors

The second "mating instinct" (Mating Instinct 2) was designed using a novel distance metric between error vectors pairs. This novel measure is called the "*Correct-My-Wrongs*" (CMW) distance and was developed specifically for the purposes of this research. **Figure 4.4** illustrates the CMW measure computation for a given a pair of error vectors, A and B. Note how error vector B "corrects" two of error vector A's five errors (represented as the solid rectangles), while error vector A "corrects" only one of error vector B's four errors (represented as the dashed rectangle).

Again, let the GA population size be N_P specimens in an arbitrary optimization problem. Similar to Mating Instinct 1, the CMW distance values between every N_P pairs of error vectors (corresponding to N_P possible pairs of specimens) can also be

represented as an $N_P \times N_P$ matrix. This matrix is referred to as the *Supplementary Distance Matrix*. In this matrix, the field in the i-th row and j-th column contains the number of entries for which the i-th specimen's error vector is "1" and the j-th specimen's error vector entries are "0." The distance matrix is asymmetric, with the i-th row measuring the potential of each of the N_P specimen in the population to correct the i-th specimen's classification errors. As in Mating Instinct 1, this criterium also naturally fulfills the principle of "opposites attract" that inspired this research because it tends to pair a specimen with others that can "supplement" its classification errors.

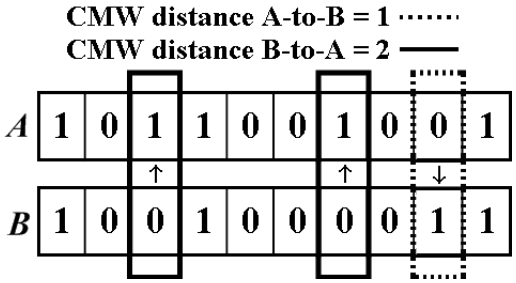

Figure 4.4: Sample *CMW* distance calculation for a pair of error vectors **A** and **B**.

4.1.5 Proposed Mating Instinct 3 - The Hamming Distance Between Attribute Sets

Mating Instincts 3 and 4 were developed for conjoint use in the proposed *IM-GA* multi-population mating strategy detailed in Section 4.3. In contrast to Mating Instincts 1 and 2, which rely on the classification error of specimens (or classifiers) in the GA population, Mating Instincts 3 and 4 use pairwise metrics based on properties of the training sets chosen by the GA to build classifiers. Notice the dichotomy between using observations on the input (i.e. training set) and output/response (i.e. classi-

fication error) of a classifier treated as a "black box" during genetic search. While Mating Instincts 1 and 2 were based on a classifier's "response", Mating Instincts 3 and 4 are based on a classifier's "input".

Mating Instinct 3, in particular, focuses on the differences between the attributes sets used to build a pair of classifiers. This "instinct" tends to promote the pairing of specimens whose corresponding classifiers are built with different sets of attributes (i.e. orthogonal information content). This property naturally fulfills the principle of "opposites attract" by promoting such pairing. To achieve this, a measure was defined to capture the dissimilarity between the attribute sets of a pair of classifiers. This measure is dubbed the *Hamming Distance Between Attribute Sets* and is computed as follows: Let the total number of attributes available in a data set be N_{AT}; the first attribute has index "1" and the last attribute has index "N_{AT}" (the subscripts "A" and "T" stand for "Attributes" and "Total", respectively). Also, assume an arbitrary specimen X in the GA population encodes a classifier built with N_{A_X} attribute whose unique indices are given by a set $A = \{a_1 \ldots a_{N_{A_X}}\}$, where each a_i can assume a unique value in A between "1" and "N_{AT}." Also, assume another arbitrary specimen Y in the GA population encodes a classifier built with N_{A_Y} attribute whose unique indices are given by a set $B = \{b_1 \ldots b_{N_{A_Y}}\}$, where each b_i can assume a unique value in B between "1" and "N_{AT}." The Hamming distance H_A between the attribute sets A and B, corresponding to specimens X and Y respectively, is defined by **Equation 4.1** below:

$$H_A(X, Y) = 2 \cdot |A \cup B| - |A| - |B| \tag{4.1}$$

where $|A \cup B|$ is the size of the union set of A and B, $|A|$ is the size of set A and $|B|$ is the size of set B. The measure $H_A(X, Y)$ is defined as the *symmetric difference* between the attributes sets A and B. The measure is symmetric and gives the number of attribute indices that differ in sets A and B. That is, it gives the count of attribute indices found uniquely in set A and not in B plus the count of attribute indices found uniquely in B and not in A. The maximum value of $H_A(X, Y)$ is reached when attribute sets A and B are disjoint[27]. For example, let sets A and B be disjoint sets, then **Equation 4.1** above yields **Equation 4.2**:

$$H_A(X, Y)_{MAX} = 2 \cdot (|A| + |B|) - (|A| + |B|) = |A| + |B| \tag{4.2}$$

.

For a GA population of N_P specimens, a symmetric $N_P \times N_P$ distance matrix is created using the H_A distance measure defined above. This matrix is referred to as the "*Attribute Dissimilarity Matrix*." The i-th row and j-th column in this matrix gives the *Hamming Distance Between Attribute Sets* of the i-th and j-th specimens, respectively. This indicates the difference between the attribute sets of two different specimens. Like in Mating Instinct 1 and 2, the matrix is symmetrical, with zeroed a out main diagonal.

Mating Instinct 3 helps optimize the GA mating process by improving the utilization of the attribute set information contained in specimens, which leads to the generation of more diverse solutions. This property is not guaranteed in the con-

[27] A pair of sets A and B are disjoint sets if none of the elements in set A are found in set B, and vice-versa.

ventional mating strategy, where fitness alone dictates the pairing of mating partners without regard to how well these partners "complement" each other in some criterium relevant to the problem under investigation.

4.1.6 Proposed Mating Instinct 4 - The Hamming Distance Between Example Sets

Mating Instinct 4 is computed in a similar fashion to Mating Instinct 3. The only difference is that the example sets (instead of the attribute sets) encoded in the specimens are used to compute the pairwise Hamming distance given by **Equation 4.1**. This distance will be referred to as the *Hamming Distance Between Example Sets*. This "instinct" tends to promote the pairing of specimens whose corresponding classifiers are built with different sets of examples. This property naturally fulfills the principle of "opposites attract" by promoting such diverse pairing of specimens. Similarly, the maximum value of the pairwise Hamming distance is reached when the example sets of two specimens are disjoint sets.

For a GA population of N_P specimens, a symmetric $N_P \times N_P$ distance matrix is created using the *Hamming Distance Between Example Sets* from all pairs of specimens in the GA population. This matrix is referred to as the *Example Dissimilarity Matrix*. The i-th row and j-th column in this matrix gives the Hamming distance between the example sets of the i-th and j-th specimens, respectively. This indicates the difference between the example sets of two different specimens. Like in Mating Instinct 1, 2, and 3, the matrix is symmetrical, with a zeroed out main diagonal.

Mating Instinct 4 helps optimize the GA mating process by improving the utilization of the example set information contained in specimens, which leads to the generation of more diverse solutions. When Mating Instincts 3 and 4 are used conjointly with either Mating Instincts 1 or 2 in the multi-population *IM-GA* strategy, the GA mating process can perform more "informed" decisions on the optimal pairing of specimens based on information regarding both the input and response of the classifier being optimized in a given problem. This is an improvement over the conventional mating strategy because fitness-based selection alone does not guarantee the optimal pairing of specimens.

The following two sections detail how the proposed Mating Instincts 1 through 4 are used in the design of five new mating strategies, both the single-population and multi-population *IM-GA* "instinct-based" mating strategies. As was discussed earlier in this section, the proposed *IM-GA* mating strategies selects pairs of specimens for mating by combining both fitness-based selection and "instinct-based" selection of specimens. This is done by choosing the first parents in each parent pair using fitness-based selection and the second parent in each pair by "endowing" the chosen first parents with a "mating instinct" that guides their partner selection. The fitness-based selection component of mating is important because it promotes the further adaptation of good solutions discovered by the GA (assuming the fitness-function designed for a problem is indeed a good indicator of solution quality). However, the GA mating process can be further optimized by pairing these "well-fit" solutions with other solutions that are orthogonal (complementary) according some other optimization criterium, which is the role played by the "instinct-based" mating strategy.

Table 4.1 defines each of the five *IM-GA* "instinct-based" mating strategies as a combination of (1) a fitness-based selection scheme and (2) one or more "mating instincts." Column 1 of Table 4.1 gives the *IM-GA* strategy name, column 2 specifies the corresponding choice of fitness-based scheme for the selection of the first parents (either random or deterministic), and column three specifies the corresponding "mating instincts" (defined earlier in this section) for the selection of the second parents. In column 2 of Table 4.1, the term "Random" stands for the random selection of the first parents while the term "Deterministic" stands for the selection of the first parents as the top-fitness specimens in the GA population. This setup is clearly reflected in the design of the five (5) *IM-GA* mating strategies discussed in the following two sections.

Table 4.1: The framework of the *IM-GA* mating strategies.

***IM-GA* Strategy Name**	**Fitness-based Selection Scheme (First Parents)**	**Mating Instinct From Section 4.1.1 (Second Parents)**	**GA Population Type**
Strategy 1 (*IM-R-H*)	Random	Mating Instinct 1	Single
Strategy 2 (*IM-D-H*)	Deterministic	Mating Instinct 1	Single
Strategy 3 (*IM-R-CMW*)	Random	Mating Instinct 2	Single
Strategy 4 (*IM-D-CMW*)	Deterministic	Mating Instinct 2	Single
Strategy 5 (*IM-MP*)	Deterministic	Mating Instincts 2, 3, 4	Multiple

4.2 Single-Population *IM-GA* Mating Strategies

This section discusses the four (4) single-population *IM-GA* "instinct-based" mating strategies. These four mating strategies use Mating Instincts 1 and 2 defined in the

previous Section 4.1.1 (see Table 4.1). These four mating strategies are designed for implementations of the GA that adopt a single population of specimens (or a single pool of potential solutions) to solve optimization problems in the domain of supervised classification. This setup is commonly referred to as the "single-population GA" and it is among the most frequently adopted implementations of the GA. The use of a single-population in the GA also means that the *IM-GA* mating strategies presented here "endow" the specimens in the population with one single "mating instinct."

4.2.1 Proposed Single-Population *IM-GA* Strategy 1 (*IM-R-H*)

The first proposed *IM-GA* strategy is referred to by the acronym *IM-R-H*, where "*R*" stands for the "R*andom*" (stochastic) selection of the first parent based on fitness and "*H*" stands for the use of the Hamming distance between pairs of error vectors to randomly select the second parent in each parent pair. Let the GA population size be N_P in an arbitrary problem in the domain of supervised classification. A total of $(N_P/2)$ pairs of specimens must be selected for mating in order to generate N_P new children specimens. In this strategy, the first $(N_P/2)$ parents are selected probabilistically according to their relative fitness values. Then, the partner for the i-th individual is chosen according to the probability distribution determined by the values of the i-th row of the *Complementary Distance Matrix*. This means that specimens with higher Hamming distance (defined in Section 4.1.3) thus get a higher chance of being selected, which favors the mating of specimens that err on different training examples.

Note that a specimen never mates with itself because the Hamming distance between two identical error vectors is "0", unless all specimens in the GA population have identical error vectors.

4.2.2 Proposed Single-Population *IM-GA* Strategy 2 (*IM-D-H*)

The second proposed *IM-GA* strategy is referred to by the acronym *IM-D-H*, where "*D*" stands for the "D*eterministic*" selection of the first parent and "*H*" stands for the use of the Hamming distance between pairs of error vectors to randomly select the second parent in each parent pair. The only difference between this strategy and *IM-GA* Strategy 1 (*IM-R-H*) is that in this strategy the first $(N_P/2)$ parent specimens for each parent pair are selected deterministically as those specimens in the GA population having the highest fitness.

4.2.3 Proposed Single-Population *IM-GA* Strategy 3 (*IM-R-CMW*)

The third proposed *IM-GA* strategy is referred to by the acronym *IM-R-CMW*, where "*R*" stands for the "R*andom*" (stochastic) selection of the first parent based on fitness and "*CMW*" stands for the use of the "*Correct-My-Wrongs*" distance between pairs of error vectors to randomly select the second parent in each parent pair. Let N_P be the GA population size. In this strategy, the first $(N_P/2)$ parents are selected probabilistically according to their relative fitness values. Then, the partner for the i-th individual is chosen according to the probability distribution determined by the values of the i-th row of the *Supplementary Distance Matrix*. This mating strat-

egy promotes the pairing of specimens that promise the correction of misclassified examples.

Note, again, that a specimen never mates with itself because the "*Correct-My-Wrongs*" distance between two identical error vectors is "0", unless all specimens in the GA population have identical error vectors.

4.2.4 Proposed Single-Population *IM-GA* Strategy 4 (*IM-D-CMW*)

The fourth proposed *IM-GA* strategy is referred to by the acronym *IM-D-CMW*, where "*D*" stands for the "*D*eterministic" selection of the first parent and "*CMW*" stands for the use of the "*Correct-My-Wrongs*" distance between pairs of error vectors to randomly select the second parent in each parent pair. The only difference between this strategy and *IM-GA* Strategy 3 (*IM-R-CMW*) is that in this strategy the first $(N_P/2)$ parent specimens for each parent pair are selected deterministically as those specimens in the GA population having the highest fitness value.

The single-population *IM-GA* Strategies 1 through 4 offer two major advantages over the conventional mating strategy. The first advantage is its incest prevention mechanism that prevents a specimen from mating with itself. The second advantage is that the proposed strategies grant higher probabilities to pairs of parents having higher distance measure values (see Sections 4.1.3-4.1.6 for a discussion of "mating instincts"), but without making it a clear choice. This means that even specimens considered "weaklings", in terms of complementary/supplementary classification performance potential, still have a chance to mate. This is important because a proba-

bility exists that good solutions may come out of such pairings, hence, they cannot be ignored. This concept is reflected in nature as well; the evolutionary process is such that even "weaklings" carry information with the potential for promoting positive impacts in the future (Shamir, Saad, and Marom 1993).

These two advantages combined aid in diminishing the effects of premature convergence toward sub-optimal solutions in the genetic search, while promoting a more "informed" specimen pairing behavior in the GA.

4.3 Multi-Population *IM-GA* Mating Strategy

This section presents the fifth proposed *IM-GA* "instinct-based" mating strategy, which is dubbed *IM-MP*, where "*MP*" stands for the use of "*M*ultiple *P*opulations." The main difference between this multi-population strategy and the single-population strategies is that this strategy uses three "mating instincts" versus only one (see Table 4.1). This strategy is designed for implementations of the GA that adopts multiple populations of specimens (or multiple pools of potential solutions) to solve optimization problem in the domain of supervised classification. This setup is commonly referred to as the "multi-population GA" and, just like the single-population GA, it is frequently adopted in implementations of the GA (see Section 3.2.2 for an overview of the multi-population GA). While the strategies from the previous section each introduced single "mating instincts" that guide the mating process, the mating strategy discussed here uses *multiple* "mating instincts" to optimize the GA mating process. It achieves this by "endowing" the specimens in each sub-population of the GA with a different "mating instinct."

Three "mating instincts" were chosen for implementation of this mating strategy, namely, Mating Instincts 2, 3, and 4 described in Section 4.1.1. As a result, *IM-GA* Strategy 5 described immediately below, relies on a total of three sub-populations to accommodate the use of the three "mating instincts"; specimens in population 1 mate according to Mating Instinct 2 (the "*Correct-My-Wrongs*" distance between pairs of error vectors), specimens in population 2 mate according to Mating Instinct 3 (the *Hamming Distance Between Pairs of Attribute Sets*), and specimens in population 3 mate according to Mating Instinct 4 (the *Hamming Distance Between Pairs of Example Sets*).

As will be discussed later, this choice of "mating instincts" fits naturally with the optimization objectives of the two chosen testbed problems from the domain of supervised classification. Recall from Chapter 3 that the chosen testbed problems deal with the optimization of the 1-NN and C4.5 Decision Trees classifiers. To optimize these classifiers, both noisy/irrelevant attributes and noisy/redundant examples in the training set have to be discarded to improve classification accuracy and minimize classification costs. In an analogous fashion, the three "mating instincts" chosen to implement this mating strategy (*IM-MP*) also seek to optimize the classification accuracy of the classifiers (Mating Instinct 2) as well as the quality of the attribute and example sets (Mating Instincts 3 and 4) of the training set, respectively.

The design of a multi-population "instinct-based" mating strategy is more complex than the single-population counterpart. This is due to the added complexities of having to choose both from which populations the parent specimens should be picked from as well as the populations that the generated children should be inserted

into (migration). Regarding the mating partner selection process, this strategy also pairs specimens by choosing the first parents in each parent pair using fitness-based selection and the second parent in each pair by "endowing" the chosen first parents with a "mating instinct" that guide their partner selection. The main difference between this strategy and those presented previously is that the parent specimens can be selected from any of 3 populations. Moreover, the "mating instinct" adopted by each population seeks to optimize the GA mating process along a different criterium.

Note that various evolutionary principles inspired the design of this multi-population mating strategy, *IM-MP*. These principles are clearly reflected in the stages of the GA mating process described in the subsequent sections. The evolutionary scenario depicted in this strategy is that of a small microcosm exhibiting both natural behaviors of *competition* and *cooperation* among sub-populations. A specimen in a particular population focuses on the search for another individual with whom a potential offspring could have a better chance of surpassing it in optimizing a specific objective. However, competition among populations arise due to the limited availability of genetic information (or diversity) in each population. Without this precious resource, none are able to fulfill their purpose of thrusting their offsprings to higher evolutionary stages (*adaptation*). Specimens must then unite as nations to compete against other populations for this same resource; populations compete with each other to place their specimens in the mating pool. At the same time, a cooperative behavior also emerges from a natural consensus that all populations must be able to improve simultaneously. After all, the well-being of the microcosm as a whole depend on the well-being of all.

To this end, unrestricted mating and migration of specimens (and children) among the individual populations is adopted to ensure a balanced distribution of information diversity among all populations. Specimens which have been successful in passing along their best traits to future generations, are also willing to relinquish their children to other populations. As a child is absorbed into the "archetype" of its adoptive population, it is led to seek that population's optimization objectives that is represented as a "mating instinct".

This sacrifice is one that all must concede to in order to reap the benefits of two important aspects: one is the ability to tap into the genetic diversity of other populations through unrestricted mating, and the second is contributing to the enhancement of the mating pool through more diverse individuals in every generation. Through this cooperative behavior, the stagnation of any individual population is better prevented.

To the author's best knowledge, no other research has been developed that combines the multi-population GA with a form of "instinct-based" mating. That is, this is a novel idea.

4.3.1 Proposed Multi-Population *IM-GA* Strategy 5 (*IM-MP*)

This strategy is dubbed *IM-MP*, where "*MP*" stands for the use of "*M*ultiple *P*opulations." Let the total number of specimens in an application the GA to an arbitrary problem in the domain of supervised classification be N_P. Also, let the N_P specimens be divided equally into three independent populations, each having a total of $N_P/3$

specimens. The three populations are referred to as "population 1", "population 2", and "population 3." Then, the mating process consists of selecting a total of $(N_P/2)$ pairs of specimens (parents) from among the three populations. These $N_P/2$ pairs of chosen specimens are then recombined into N_P new children specimens. For simplicity, assume $N_P = 30$ for all of the examples that follow. The mating partner selection process consists of the following four steps:

1. Step 1: Choose a population from which each parent will be selected, that is, choose each parent's population ID (either 1, 2, or 3);

2. Step 2: Select the actual parent specimens according to the selected population IDs;

3. Step 3: To obtain children, apply recombination (i.e. two-point crossover) and mutation, and ;

4. Step 4: Decide how to assign each generated child to a population.

These four steps are detailed as follows:

Step 1: This first step consists of probabilistically selecting 15 pairs $(N_P/2)$ of population ID's for the parent specimens using a wheel-of-fortune scheme given by **Equation 4.3** below. This scheme uses the average fitness of the specimens in each population, where the choice of the fitness-function is problem-dependent, to randomly select the population ID's for the parent specimens. This mechanism causes the populations to compete among themselves in order to enter the mating pool of 15 $(N_P/2)$ mating pairs. Valid chosen population ID values are within the integer range

$\{1, 2, 3\}$ and correspond to populations 1, 2, and 3, respectively. This population ID selection is accomplished in two steps. First, 30 ID values (N_P) are consecutively selected randomly using a wheel-of-fortune scheme based on the probability values assigned to the different populations by **Equation 4.3**. Next, the randomly chosen ID values are paired consecutively to generate 15 pairs of population ID values. Population ID pair repetition is allowed. That is, pairs of population ID values such as $\{[1\ 1], [2\ 2], [3\ 3]\}$ are valid and simply entail that specimens from the same population will be chosen for crossover (since the population IDs match in each pair). On the other hand, population ID pairs such as $\{[1\ 2], [1\ 3], [2\ 1], [2\ 3], [3\ 1], [3\ 2]\}$ entail that specimens from different populations will be chosen for mating (since the population IDs differ in each pair). Notice that **Equation 4.3** favors those populations having higher average fitness by granting them higher selection probability values:

$$Prob(P_i)_{Parent} = \frac{MeanFitness(P_i)}{\sum_{j \epsilon J} MeanFitness(G_j)} \tag{4.3}$$

where

- P_i and P_j, for $i = 1, \ldots 3$ and $j = 1, \ldots 3$, denote the $\underline{\mathrm{i}}_{th}$ and $\underline{\mathrm{j}}_{th}$ populations, respectively;
- $Prob(P_i)_{Parent}$, for $i = 1, \ldots 3$, denotes the probability of a parent from the $\underline{\mathrm{i}}_{th}$ population being selected for mating;
- $MeanFitness(P_i)$, for $i = 1, \ldots 3$, is the mean fitness value of all specimens in the $\underline{\mathrm{i}}_{th}$ population, and;

- $MeanFitness(P_j)$, for $j = 1, \ldots 3$, is the mean fitness value of all specimens in the $\underline{j}_{th}$ population.

The 15 ($N_P/2$) selected pairs of population ID's can be represented by a 15×2-dimensional matrix **G** that holds the 15 pairs of population ID's as row vectors $\mathbf{G}_k$=[$\mathtt{p}_{k1}, \mathtt{p}_{k2}$], for k=1,2,...,15 and $\mathtt{p}$=1,2,3. **Figure 4.5** below illustrates the process of computing the **G** matrix from a set of arbitrary probability values computed by **Equation 4.3**. Notice from the example that population 1 has higher selection probability than populations 2 and 3 (43%). This is reflected in population 1's greater share of the total area of the wheel-of-fortune, followed by population 3 and 2, respectively:

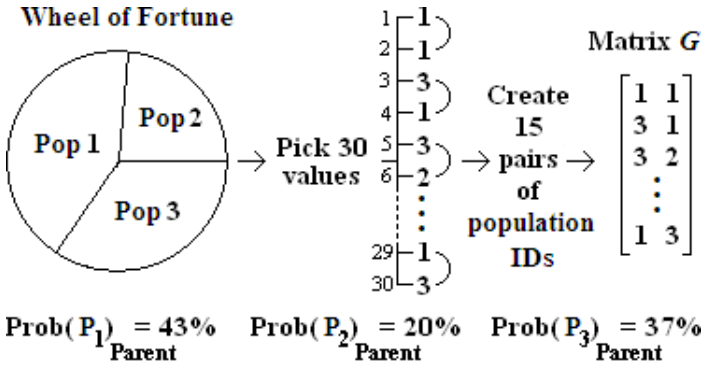

Figure 4.5: Sample selection of population ID's for parent specimens in *IM-GA* Strategy 5.

Step 2: Once matrix **G** has been computed, the second step involves selecting actual specimens from the chosen populations for mating. At this stage, certain considerations come into play. First, recall that this multi-population mating strategy must optimize search in three distinct populations with size $N_P/3$, in contrast to the single-population *IM-GA* Strategies 1 through 4 which assume a single population of size N_P. While constraining the total number of specimens in all populations to N_P specimens ensures that the experimental results of all strategies are comparable, the

reduced size of the individual populations in this strategy (10 specimens, or $N_P/3$), makes the optimization problem slightly harder. To ensure the efficient use of the information diversity available in the populations of reduced sizes, this strategy adopts the deterministic selection of the first parents in each mating pair (those corresponding to the left-hand side of matrix **G**) as the top-fitness specimens in each population. This improves the odds of probabilistically matching better parents from different populations and also improves the utilization of information diversity in the smaller populations by avoiding duplicate selection of specimens (contrast random selection, which allows duplicate selection of specimens).

However, the second parents of each mating pair (those corresponding to the right-hand side of matrix **G**) are then selected according to a "mating instinct." Consider that each mating pair combination consists of both a left-hand side and a right-hand side parent. This means that column 1 and 2 of matrix **G** represent the population ID's of the left-hand side and right-hand side parents, respectively. Let the constants $Col1_i$ and $Col2_i$, for $i = 1, \ldots 3$, be the total count of times the i_{th} population ID appears in column 1 and 2 of matrix **G**, respectively. First, the left-hand side parent specimens are chosen by picking the $Col1_i$ top-fitness specimens from each population as the left-hand side representative parents in the 15 ($N_P/2$) parent-pair combinations. This process is performed such that the first occurrence of the i_{th} population ID in column 1 of matrix **G** corresponds to the first top-fitness specimen of the i_{th} population, the second occurrence of the i_{th} population ID in column 1 of matrix **G** corresponds to the second top specimen of the i_{th} population, and so on, for all populations 1, 2, and 3. **Figure 4.6** illustrates more clearly how the population

ID values in column 1 of matrix **G** are used to select the top-fitness specimens of each population as the left-hand side parents in the 15 ($N_P/2$) mating pair combinations. Notice that each parent specimen is represented by **both** a population ID and a specimen ID. The former ID consecutively matches the population ID's in column 1 of matrix **G** while the latter consecutively matches the top-fitness specimens of each population. Also, notice that the right-hand side parents are left blank for now, to illustrate that the right-hand side parents are selected only in the subsequent step using an "instinct-based" selection process:

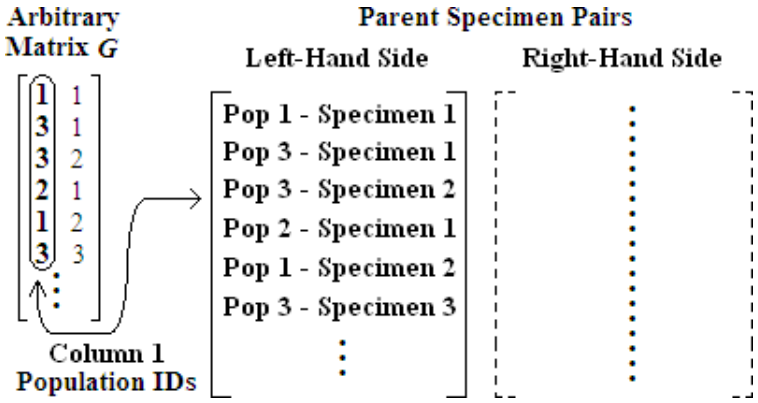

Figure 4.6: Translating population ID's in column 1 of matrix **G** to actual left-hand side parent specimens in *IM-GA* Strategy 5.

When choosing specimens for the right-hand side of matrix **G**, the population ID of the left-hand side parents play an important role. The right-hand side parents are, in essence, matched to the left-hand side parents using "mating instincts." The choice of "mating instinct" is determined based on a left-hand side population ID; each population seeks to optimize a different mating objective. For each parent-pair represented by a row of matrix **G**, the selection of the right-hand side parents works as follow:

1. If the population ID of the left-hand side parent is "1", then the "*Correct-My-Wrongs*" (CMW) distance is computed between the error vector of the chosen left-hand side parent from population 1 and the error vectors of all $N_P/3$ specimens from the population identified by the corresponding right-hand side population ID;

2. If instead the population ID is "2", then the Hamming distance is computed between the training data attribute set encoded by the chosen left-hand side parent specimen and the training data attribute sets encoded by all $N_P/3$ specimens from the population identified by the corresponding right-hand side population ID, and;

3. Finally, if the population ID is "3", then the Hamming distance is computed between the training data example set encoded by the chosen left-hand side parent specimen and the training data example sets encoded by all $N_P/3$ specimens from the population identified by the corresponding right-hand side population ID.

For any of the criteria above, the $N_P/3$ computed distance values are transformed into probability shares in a wheel-of-fortune based selection scheme. If the i-th value is picked, then the i-th specimen from the corresponding right-hand side population will mate with the left-hand side parent. In summary, specimens from the different populations that were chosen as the left-hand side parents in the mating pool will mate according to the principal "mating instinct" of their corresponding populations.

Furthermore, referring back to the example given in **Figure 4.6**, this also means that for the fifth row of matrix **G**, the CMW distance measure is computed between the error vectors of specimen 2 from population 1 (left-hand side parent) and all specimens of population 2. A specimen from population 2 is then randomly picked as the right-hand side parent based on the resulting probabilistic distribution. For the second row of matrix **G**, the Hamming distances is computed between the example set encoded by specimen 1 from population 3 and the example sets of all specimens in population 1. Again, the values are used to randomly pick an actual parent from among the specimens in population 1.

The overall tendency of this proposed mating strategy is to guide specimens from population 1 to pair-up with other specimens having a high potential for reinforcing, or further improving, their classification performance (e.g. correcting their errors). Moreover, the tendency is also to guide specimens in population 2 to pair-up with other specimens encoding more diverse set of attributes. Finally, the tendency is also to guide specimens in population 3 to pair-up with other specimens having more diverse set of examples. This tendency to promote more diverse specimen pairing is an improvement over the conventional mating strategy (because it attempts to make explicit use of the information in the specimens to optimize the GA mating process.

Step 3: In the third step, crossover[28] is then applied to the 15 pairs ($N_P/2$) of selected parent specimen from different combinations to generate a total of 30 children (N_P). Note that crossover step is mentioned here only as a placeholder for any desired

[28]Crossover is the exchange of genetic information encoded in the chromosomes of specimens to generate new specimens. Typically, the crossover operation is performed on a pair of specimens ("parents") to generate a new pair of specimens ("children"), although some implementations of the GA allow crossover between more than two parents.

implementation. The crossover step does not impact the mating strategy. Crossover is mentioned here for completeness since the mating strategy of the multi-population GA dictates how specimens are paired for mating *before* the crossover step and also to which populations the generated children are migrated *after* the crossover step. Hence, the crossover step has to be mentioned here only for the completeness of the description of the mating strategy, and not because the crossover step has an impact on mating.

Step 4: Finally, in the fourth step, the 30 (N_P) generated children have their population ID's assigned probabilistically, and independent of their original parents' ID's, using the wheel-of-fortune scheme given by **Equation 4.4**. The process of choosing into which population to send a new specimen is known as "migration" in the GA:

$$Prob(P_i)_{Child} = \frac{1/MeanFitness(P_i)}{\sum_{j \epsilon J} 1/MeanFitness(p_j)} \tag{4.4}$$

where

- P_i and P_j, for $i = 1, \ldots 3$ and $j = 1, \ldots 3$, denote the $\underline{\mathrm{i}}_{th}$ and $\underline{\mathrm{j}}_{th}$ populations, respectively;
- $Prob(P_i)_{Child}$, for $i = 1, \ldots 3$, is the probability that a generated child will be migrated to the $\underline{\mathrm{i}}_{th}$ population;
- $MeanFitness(P_i)$, for $i = 1, \ldots 3$, is the mean fitness value of all specimen in the $\underline{\mathrm{i}}_{th}$ population, and;

- $MeanFitness(P_j)$, for $j = 1, \ldots 3$, is the mean fitness value of all specimen in the j_{th} population.

Figure 4.7 illustrates how a set of arbitrary probability values given by **Equation 4.4** are mapped into the population ID's chosen for the children specimens (migration). The probability values are inversely related to average fitness of each population. Notice from the illustration that population 2, which in the previous example given in **Figure 4.5** was depicted as having the lowest average fitness value, is now depicted as having the largest probability share of the wheel-of-fortune, followed by population 3 and 1, respectively.

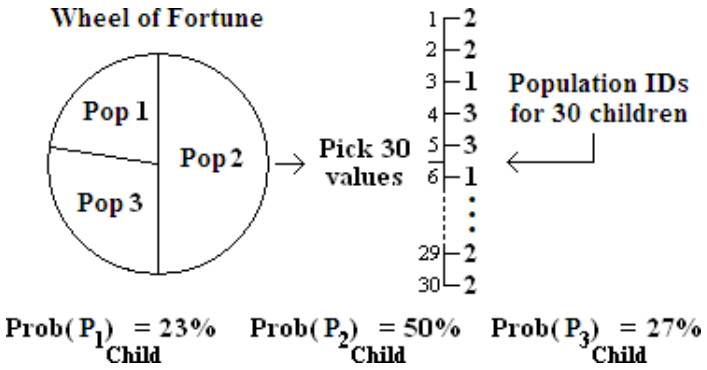

Figure 4.7: Sample selection of the children's population ID's in *IM-GA* Strategy 5.

Each child specimen is thus migrated, probabilistically, to one of the three populations that are chosen as described above. A similar approach was advocated by Grosso (1985), the motivation being that the migrant specimens should be sent preferably to populations where they are likely "to make a difference."

Since each child can be "seeded" into any population, genetic diversity benefits. Populations with higher average fitness tend to mate more frequently, populations with lower average fitness are more likely to receive the children. The goal is an

equilibrium between the probabilities of selecting a parent from, and assigning a child to, any population.

Figure 4.8 illustrates by showing the convergence behavior of the three populations of *IM-GA* Strategy 5 when used in the baseline *RK-GA*$_0$ and applied to the `cmc` data set from the UCI Machine Learning Repository (Newman and Merz 1998). The horizontal axis gives the cumulative number of specimens for which the fitness-function has been evaluated (this way of measuring time is independent of the programmer's skills and the computer's performance). The vertical axis plots the fitness of the best specimen in each population. Notice how the average fitness of the populations periodically surpass each other, reflecting the selection/migration processes. In the long run, all populations tend to converge to similar fitness.

The migration thus described also ensures that the mating pool of each population is enriched with specimens optimized along the other criteria (classification error, example set diversity, and attribute set diversity). This leads to simultaneous optimization of multiple objectives in the GA mating process as the populations share their best traits.

Finally, children are placed in their target populations in each new generation of the GA. At this point, the survival operator is applied to retain the top $N_P/3$ specimens in each population, according to some criteria, and discard the remaining specimens. The choices of the survival operator do not affect the proper operation of *IM-GA* Strategy 5, which was designed to work properly even with single specimen populations.

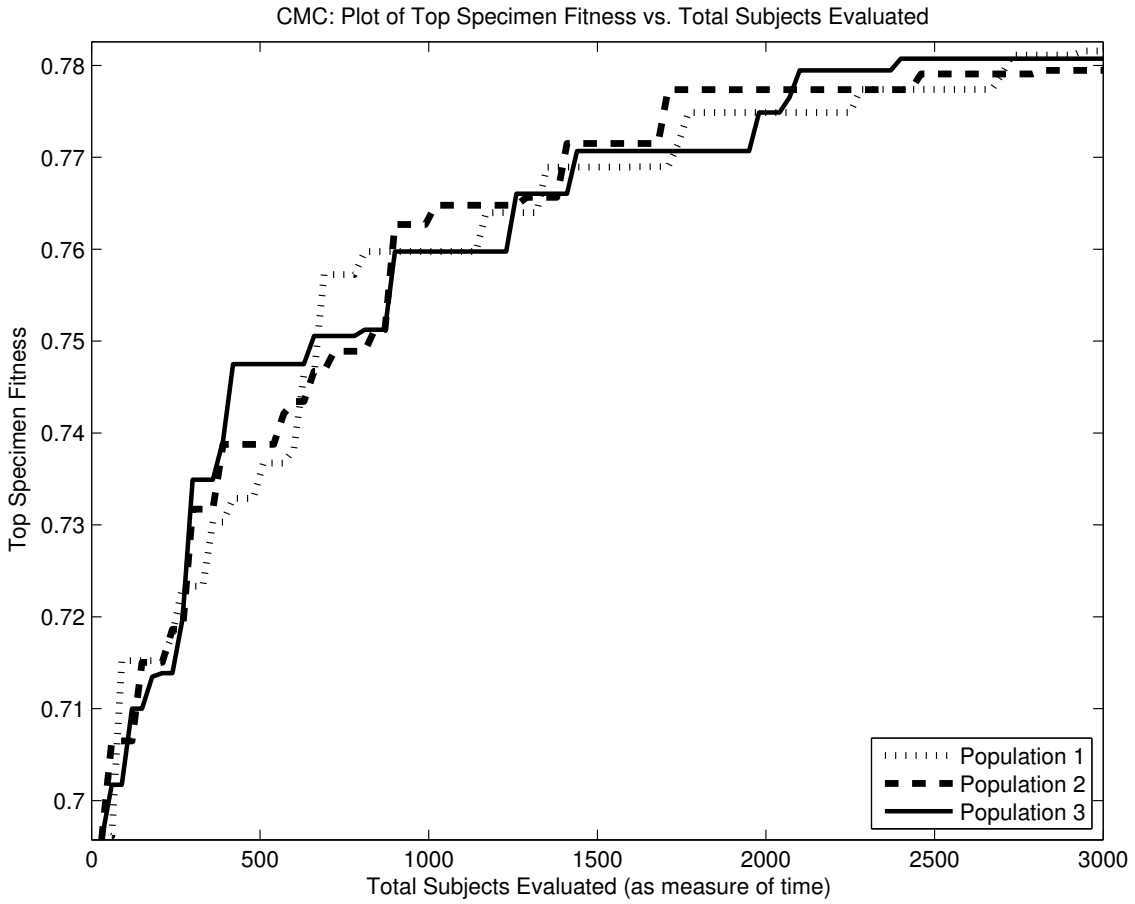

Figure 4.8: Sample convergence of the *IM-GA* multi-population GA Strategy 5 (*IM-MP*) when used in the baseline *RK-GA*$_0$ and applied to the `cmc` data set.

The next sections describe the application of *IM-GA* Strategies 1 through 5 to optimize genetic search in the two chosen testbed problems of 1-NN Tuning and Optmal Decision Forest.

4.4 Testbed Problem 1 - The 1-NN Tuning Problem

This section discusses how the the five proposed *IM-GA* "instinct-based" mating strategies are used to optimize genetic search in the testbed problem of 1-NN Tuning. Briefly, the 1-NN Tuning problem consists of searching for optimal subsets of examples

and attributes in a data set with the goal of improving the accuracy of the 1-NN classifier (as removal of noisy examples and irrelevant/noisy attributes from a data set improves the 1-NN accuracy). This section shows how the *IM-GA* strategies can replace the conventional fitness-based mating strategy in a real implementation of the GA that is designed to search for optimal 1-NN classifiers.

Recall from Section 3.2 that "mating" is only a part of the GA iterative process. Therefore, the *IM-GA* mating strategies cannot be used alone to solve search and optimization problems because they are not a complete GA. Hence, the first step in the process of evaluating the impact of *IM-GA* on the performance of the GA when applied to a certain optimization problems, is to design a good GA capable of finding good solutions to that optimization problem. Moreover, this GA should adopt and work well with the conventional mating strategy. This way, a good set of baseline experimental results based on the performance of the conventional mating strategy can be attained. These "baseline results" can then be compared to those results obtained after applying the five (5) new *IM-GA* strategies. This will indicate real improvement.

In this work, the well-known GA, the *RK-GA* (Rozsypal and Kubat 2003), was re-implemented and improved to be used as the baseline GA to generate optimal 1-NN classifiers for Testbed Problem 1. This improved version of *RK-GA* is hereinafter referred to as the baseline *RK-GA*$_0$. *RK-GA*$_0$ fits all the prerequisites discussed above. First, the original *RK-GA* has been shown to compare favorably with other GA-based and non-GA-based techniques on various experiments with UCI data sets. It is, therefore, a good GA that finds goods solutions to the 1-NN Tuning problem.

Because the original *RK-GA* adopted the conventional fitness-based mating strategy and performed well (Rozsypal and Kubat 2003), any improvements to the baseline *RK-GA*$_0$ by *IM-GA* are real improvements.

The next few sections show how the five proposed *IM-GA* mating strategies were incorporated into the baseline *RK-GA*$_0$'s framework. The experimental results presented in Chapter 5 confirmed that *IM-GA* can indeed accelerate *RK-GA*$_0$ and that the faster convergence did not come at the cost of lower quality of the discovered solutions. In addition, the next few sections also demonstrate theoretically that the *IM-GA* mating strategies require negligible computational overhead when introduced in *RK-GA*$_0$. In summary, *IM-GA* sped up *RK-GA*$_0$ without harming its performance under any criteria.

4.4.1 Solving 1-NN Tuning with the Original *RK-GA*

This section discusses a GA built upon in this work to solve the 1-NN Tuning problem (Testbed Problem 1). This GA is called *RK-GA* and it is named after Rozsypal and Kubat who introduced this GA in 2003 (Rozsypal and Kubat 2003), and who were inspired by successful earlier works in the application of the GA to the 1-NN Tuning problem (Kuncheva and Jain 1999; Kuncheva and Bezdek 1998). In their work, Rozsypal and Kubat introduced an alternative method to the original approach developed by Kuncheva and Jain (1999) which used a novel variable-length, value-encoded specimen chromosome scheme. In the *RK-GA* method, each specimen in the population is represented by two chromosomes: one attribute chromosome and one example chromosome. Combined, the two chromosomes encode a training set that

is used to build potentially optimal 1-NN classifiers at the end of the genetic search. Their proposed chromosomal structure is illustrated in **Figure 4.9**.

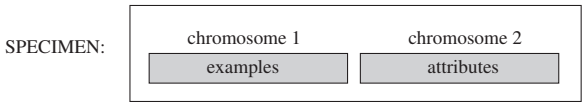

Figure 4.9: Original *RK-GA* specimen with one example chromosome and one attribute chromosome.

Value encoding is employed such that each "allele" (or field) of a given chromosome carries an integer value that points either to a training example or attribute. For instance, the specimen **{2, 9, 15},{1, 3}** represents the three training examples **({2, 9, 15})** whose values are defined using only the first and third attributes (all other attributes and examples in the training set are ignored). When this specimen is decoded to build a 1-NN classifier, the training set examples are selected via the example chromosome, while the training set attributes are selected via the attribute chromosome. The resulting 1-NN classifier uses the distance metric defined by **Equation 4.5** between any two example vectors to classify unknown examples:

$$D(x, y) = \sqrt{\sum_{i=1}^{n} d(x_i, y_i)} \tag{4.5}$$

where $d(x_i, y_i)$ is the contribution of the i-th attribute. For numerical attributes, $d(x_i, y_i) = (x_i - y_i)^2$, while for boolean and discrete attributes the value is computed as $d(x_i, y_i) = 0$, for $x_i = y_i$, and $d(x_i, y_i) = 1$, for $x_i \neq y_i$.

Specimen Fitness Evaluation

Fitness evaluation is the process of measuring the quality of a GA-generated solution (or specimen). Rozsypal and Kubat introduced in their *RK-GA* a fitness-function

that proved to be an effective specimen discriminating measure, as supported by the good results reported in (Rozsypal and Kubat 2003). The fitness-function of *RK-GA* is defined by **Equation 4.6**, where E_R is the number of training examples misclassified by a given specimen (or its corresponding 1-NN classifier), N_E is the number of retained examples (or the cardinality of the retained example set), N_A is the number of retained attributes (or cardinality of the retained attribute set), and c_1, c_2, and c_3 are user defined weight parameters. The fitness-function assumes value $f_{RK\text{-}GA}= 0$, for $N_E = 0$ or $N_A=0$, since no meaningful training set can have 0 examples or attributes. **Equation 4.6** utilizes the weighted sum approach described in (Coello 1999) to combine the multiple optimization objectives of (1) maximizing the classification accuracy, (2) reducing the cardinality of the attribute set, and (3) reducing the cardinality of the example set, into a single fitness-function:

$$f_{RK\text{-}GA} = \frac{1}{c_1 \cdot E_R + c_2 \cdot N_E + c_3 \cdot N_A} \tag{4.6}$$

Recombination

At each iteration of *RK-GA*, a new population is created by choosing parents from which children are generated through the process of recombination. Parents are selected probabilistically using the wheel-of-fortune based scheme given by **Equation 4.7**:

$$Prob(S') = \frac{f(S')}{\sum f(S)} \tag{4.7}$$

where $f(S)$ is the fitness of an arbitrary specimen S. This parent selection scheme is the conventional mating strategy that *IM-GA* seeks to replace. Once a pair of parents have been chosen, a two-point crossover scheme (see below) is employed to create a pair of children.

Two-Point Crossover

To apply the two-point crossover on a given parent-pair, let the length of one of the parent's first chromosome be denoted by N_1 and the length of the second chromosome be denoted by N_2. A pair of two integers for each chromosome is then randomly selected over the closed interval $[1, N_1]$ and $[1, N_2]$, respectively, using the uniform distribution. Finally, the two pairs of integers are used to define substrings in the respective chromosomes, which are then exchanged with the respective chromosome substrings that were defined for the second parent in the same fashion. **Figure 4.10** illustrates this two-point crossover scheme for a single chromosome on an arbitrary pair of parent specimens A and B. This crossover scheme is applied in the same fashion to both the example and attribute chromosomes:

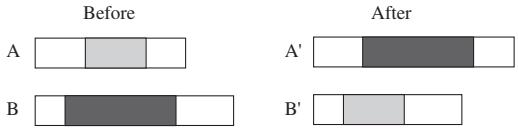

Figure 4.10: Two-Point Crossover of a single chromosome

Mutation

Mutation is applied to each of the generated children in each generation of the GA. The mutation operator randomly selects a specified percentage of the "alleles" (or fields) in both the example and attribute chromosomes of a given child specimen

and adds to them a randomly generated integer. The final mutated values are then modulo[29] the total number of examples or attributes. This confides the final mutated values to a valid range that is data set relevant. *RK-GA* used a mutation rate of 5% for all of the experiments.[30] The sensitivity experiments showed no detrimental effects due to small variations of this value.

Population and Survival

In *RK-GA* the GA population size was fixed to a total of 30 specimens. For consistency, this population size is also adopted in all experiments with *RK-GA* presented in this work. The population is initialized (the chromosomes filled) with uniformly distributed random integers over the range of the number of examples and attributes, respectively. In each new generation, 30 children are generated, then merged with the original/surviving population. Finally, all specimens, new and old, are then sorted by rank of fitness. The principle of survivor selection elitism (Bentley 1999) is then used to "*kill-off*" the worst half specimens.

Termination Criterion

In *RK-GA*, a fixed number of 100 generations was adopted for the experiments with benchmark data sets acquired from the UCI Machine Learning Repository (Newman and Merz 1998). Again for consistency with the previous work, all experiments with the baseline $RK\text{-}GA_0$ presented in this work also run for 100 generations. An increased number of generational runs was found to be overly laborious, in addition to not

[29]The modulo operation gives the remainder of a division operation.
[30]A mutation rate of 5%-10% is a typical range

necessary to demonstrate the relative improvement of the proposed *IM-GA* "instinct-based" mating strategies over conventional fitness-based mating strategy.

4.4.2 An Improvement to the Original *RK-GA* (Baseline *RK-GA*$_0$)

This section describes an improvement introduced by this work to original *RK-GA*'s fitness-function presented in (Rozsypal and Kubat 2003). One of the main advantages of the proposed improvement is that it allows the weight parameters of **Equation 4.6** to be set in a data set relevant fashion that completely eliminates the need for manual tuning. The new *RK-GA* adopting the new fitness-function is referred to as the baseline *RK-GA*$_0$. All experiments relevant to Testbed Problem 1 (1-NN Tuning) that were performed in this work used *RK-GA*$_0$ as a baseline GA. Hence, the experimental results of *RK-GA*$_0$ serve as the baseline for evaluating the impact of the proposed *IM-GA* strategies on the optimization of genetic search.

In spite of the original *RK-GA*'s promising results as it exists unmodified, a controversy arises regarding the arbitrary values of the *user-defined* weight parameters c_1, c_2, and c_3 employed in the original experiments. Due to limited knowledge about the sensitivity of the GA to the different terms of the fitness-function, all three parameters were assigned the value 1 in the original *RK-GA*. This provided the same level of significance to all terms (optimization objectives) of **Equation 4.6**. However, this is not an intuitive choice because different terms can assume different ranges of values and, therefore, have different search space sizes (e.g. a data set can have a significantly larger number of examples than attributes). Consequently, the GA needs

to spend significantly more effort searching through the larger search spaces of certain terms for optimal solutions. The differences in the search space sizes of the different terms of **Equation 4.6** should be reflected in the values of the weights assigned to the different terms. These differences in weight values would consequently instruct the GA on the degree of optimization priority of each term. Here, the original *RK-GA*'s fitness-function is improved through a new approach for setting of weight values that take into account the aforementioned considerations. The new weight values are set automatically in a data set relevant way. Finally, another advantage of this automatic setting of weights is that manual effort spent in coefficient tuning is no longer required.

It would be unwise, however, to simply discard the adequacy of **Equation 4.6**, especially in view of the excellent performance depicted in the experimental results presented by Rozsypal and Kubat (2003). Hence, the new fitness-function was built by dissecting the inner-workings of **Equation 4.6** to reach a better understanding of its contribution to the original *RK-GA*. This process resulted in a new (and unitless) fitness-function developed directly from the baseline **Equation 4.6**.

The new fitness-function is derived as follows. Let N_{ET} and N_{AT} be the total number of examples and attributes in a given training data set, respectively. Also, let N_C be the number of training examples correctly classified by a given specimen (or its corresponding 1-NN classifier) and N_{ED} and N_{AD} be the total number of training examples and attributes discarded by a given specimen, respectively. Using these variables, **Equation 4.6** can be sequentially re-written into a different form through **Equations 4.8**, **4.9**, and **4.10**, respectively:

$$f_{RK\text{-}GA} = \frac{1}{c_1 \cdot (N_{ET} - N_C) + c_2 \cdot (N_{ET} - N_{ED}) + c_3 \cdot (N_{AT} - N_{AD})} \tag{4.8}$$

$$f_{RK\text{-}GA} = \frac{1}{[(c_1 + c_2) \cdot N_{ET} + c_3 \cdot N_{AT}] - [c_1 \cdot N_C + c_2 \cdot N_{ED} + c_3 \cdot N_{AD}]} \tag{4.9}$$

$$f_{RK\text{-}GA} = ([(c_1{+}c_2){\cdot}N_{ET}{+}c_3{\cdot}N_{AT}]{-}[c_1{\cdot}(N_{ET}{-}E_R){+}c_2{\cdot}(N_{ET}{-}N_E){+}c_3{\cdot}(N_{AT}{-}N_A)])^{-1} \tag{4.10}$$

Notice from the denominator of **Equation 4.10** that the first term in brackets simply sums up to a positive constant value. Therefore, maximizing **Equation 4.10**, or in other words, maximizing the fitness value of a given specimen, consists of maximizing the second term in brackets that holds the independent variables E_R, N_E, and N_A. These findings allude to the following three sensible conclusions regarding the objectives that the original *RK-GA*'s original fitness-function sought to optimize:

- In terms of classification accuracy: maximize the difference between the total number of examples in a given data set and the total number of misclassified examples by a given specimen, i.e., $(N_{ET} - E_R)$;
- In terms of the size of the example set: maximize the difference between the total number of examples in a given data set and the total number of examples retained by a given specimen, i.e., $(N_{ET} - N_E)$, and;

- In terms of the size of the attribute set: maximize the difference between the total number of attributes in a given data set and the total number of attributes retained by a given specimen, i.e., $(N_{AT} - N_A)$.

These three conclusions allow for the derivation of a new fitness-function given by **Equation 4.11** that also seeks to optimize the three aforementioned objectives. Notice that this new fitness-function is also maximized with decreasing classification error and size of the example and attribute sets. This fitness-function is used in the improved version of *RK-GA*, the baseline *RK-GA*$_0$:

$$f_{RK_{New}} = c_1 \cdot \left(\frac{N_{ET} - E_R}{N_{ET}} \right) + c_2 \cdot \left(\frac{N_{ET} - N_E}{N_{ET} - 1} \right) + c_3 \cdot \left(\frac{N_{AT} - N_A}{N_{AT} - 1} \right) \qquad (4.11)$$

Notice also that each of the terms of **Equation 4.11** are normalized by constants placed in their respective denominators. These constants are the maximum values of each term. For instance, the maximum value of **Equation 4.11** is attained when an arbitrary specimen encodes a 1-NN classifier that is 100% accurate ($E_R = 0$) while being described by a single training example and attribute ($N_E = N_A = 1$); this is as compact and accurate as any 1-NN classifier can possibly become. Therefore, the highest possible values in the numerators of the different terms of **Equation 4.11** are N_{ET}, $N_{ET} - 1$, and $N_{AT} - 1$ when when $E_R = 0$, $N_E = 1$, and $N_A = 1$, respectively. This normalization step ensures that all terms of the fitness-function are expressed as unitless, fractional percentage values.

Finally, the genetic search emphasis on each of the terms (optimization objectives) of **Equation 4.11** is controlled by the weight parameters c_1, c_2, and c_3. It

was found empirically that the domain size[31] of each of the terms of **Equation 4.11** is a good estimators of the relative share of genetic search emphasis that each term should receive. This intuitive solution is based on the assumption that the simultaneous optimization of all objectives of **Equation 4.11** requires proportionally larger appropriations of the GA search emphasis to those terms having larger search spaces. For example, assume the baseline $RK\text{-}GA_0$ is applied to a data set containing a significantly larger number of examples than attributes. To build optimal 1-NN classifiers, $RK\text{-}GA_0$ will naturally have to spend more computational time attempting to determine which examples are noisy or not versus which attributes are noisy/irrelevant. The weight selection scheme proposed here acknowledges that the total number of possible values that each of the terms of **Equation 4.11** can assume (all terms in **Equation 4.11** are integer valued and have finite domains) also explicitly reflect their intrinsic optimization complexity relative to one another.

The properties of the data set can be used to estimate the number of possible values that each term can have. Notice that the size of the domains of the first and second terms of **Equation 4.11** are both equal to $N_{ET}+1$, since $E_R \epsilon \{0, 1, 2, \ldots, N_{ET}\}$ and $N_E \epsilon \{0, 1, 2, \ldots, N_{ET}\}$. Also, the size of the domain of the third term is found to be equal to $N_{AT} + 1$, since $N_A \epsilon \{0, 1, 2, \ldots, N_{AT}\}$. This choice of weight parameters is data set adaptive and, consequently, completely eliminates the need for manual effort in the tuning of weight values for any data set. As a final requirement, the weight parameters c_1, c_2, and c_3 should also be normalized in order to conveniently conserve

[31] The domain size of an integer valued variable is equivalent to the total number of possible values that the variables may assume.

the unitless and normalized nature of the terms of **Equation 4.11**. To this end, the weights c_1, c_2, and c_3 are respectively defined by **Equations 4.12**, **4.13**, and **4.14**:

$$c_1 = \frac{N_{ET} + 1}{2 \cdot N_{ET} + N_{AT} + 3} \tag{4.12}$$

$$c_2 = \frac{N_{ET} + 1}{2 \cdot N_{ET} + N_{AT} + 3} \tag{4.13}$$

$$c_3 = \frac{N_{AT} + 1}{2 \cdot N_{ET} + N_{AT} + 3} \tag{4.14}$$

The experimental results attained by the application of the baseline *RK-GA*$_0$ to Testbed Problem 1 were attained using this new fitness-function. The results were either comparable or better than those previously reported in (Rozsypal and Kubat 2003), where the old fitness-function was employed in the original *RK-GA*. The new fitness-function lead *RK-GA*$_0$ to generate better solutions in terms of classification accuracy and also example and attribute reduction. **Thus, it is safe to conclude that the new fitness-function performs better than the old one.**

The next sections define different permutations of the baseline *RK-GA*$_0$ combined with an *IM-GA* mating strategy that were investigated in this work. The name of each permutation carries the keyword "*RK-GA*" and a number that corresponds to the *IM-GA* mating strategy adopted in that combination (e.g. *RK-GA*$_1$ denotes the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 1). This is done to ease the recognition of the chosen combinations of mating strategies in the experimental section.

4.4.3 The Computational Costs of the Baseline *RK-GA*$_0$

All GAs have computational costs of both time and storage (the "costs"). Costs are dominated by various factors, such as by the GA architecture or by problem-dependent factors. An example of the architectural dependence of costs is that of GAs based on Pareto-dominance (Coello 1999). In such GAs, most of the computational time is easily spent performing the required Pareto-dominance sorting procedure needed to determine non-dominated individuals in each generation. Problem dependency of costs is often reflected in the complexity of the computations required for fitness evaluations. The baseline *RK-GA*$_0$, as well as the novel GA designed for Testbed Problem 2 (the baseline *TM-GA*$_0$), are good example of GAs whose costs are dominated by the fitness-function evaluation. In *RK-GA*$_0$, and even in the original *RK-GA*, computing the specimens' training errors is the most costly step because it involves multiple iterations of 1-NN search (Cheatham and Rizki 2006). All of the other components (mating, recombination, mutation, survival) add only negligible cost compared to fitness evaluation. The importance of examining the computational costs of *RK-GA*$_0$ is to ensure that the benefits of using the proposed *IM-GA* mating strategies are not outweighed by additional computational costs.

In *RK-GA*$_0$, the computational costs are dominated by the *nearest-neighbor* search process required for fitness evaluation. For a closer look at how computing training errors impact cost, let us consider the cost of finding the nearest neighbor of a single training example X in an arbitrary specimen in the GA population. Let N_E and N_A be the size of the specimen's example and attribute chromosomes, and let Y denote

any other single example in the arbitrary specimen. Let T_M, T_S, and T_A be the CPU time required to perform multiplication, subtraction, and addition operations, respectively. Assuming the *worst-case scenario* of entirely numerical attributes (in the implementation of the 1-NN classifier used in this work, the distance between nominal attribute values is computed as a lightweight boolean operation), **Equation 4.5** (the "Euclidean distance") can be simplified as follows:

$$D'(X,Y) = \sum_{i=1}^{N_A} (X_i - Y_i) \cdot (X_i - Y_i) \tag{4.15}$$

Equation 4.15 gives a distance between the training example X and any one example Y in the arbitrary specimen. The computational time of the Euclidean distance computation is estimated by **Equation 4.16**:

$$Time_{CPU}(D'(X,Y)) = N_A \cdot T_M + 2 \cdot N_A \cdot T_S + (N_A - 1) \cdot T_A \tag{4.16}$$

In the worst-case of large attribute and example chromosomes, where $N_E \rightarrow N_{ET}$ and $N_A \rightarrow N_{AT}$, the computational time required to find the nearest neighbor of X, defined here as or NN(X), for an arbitrary specimen is given by **Equation 4.17**:

$$Time_{CPU}(NN(X)) = N_{ET} \cdot N_{AT}(T_M + 2 \cdot T_S + T_A) \tag{4.17}$$

Then, the computational time required to find the nearest neighbors of all training examples in the arbitrary specimen is given by **Equation 4.18**:

$$Time_{CPU\ Total} = {N_{ET}}^2 \cdot N_{AT}(T_M + 2 \cdot T_S + T_A) \tag{4.18}$$

Equation 4.18 shows that the computational time required to compute the training error of the 1-NN classifier is proportional to the training set size. In contrast, the computational time of all other steps in the GA iterative process is a function of the population size (N_P) alone, which is usually a small constant and independent of data set properties. Thus, the worst case computational time cost of *RK-GA*$_0$ is: $O({N_{ET}}^2 \cdot N_{AT})$.

Similarly, the storage cost is dependent on the training set size only, or: $O(N_{ET} \cdot N_{AT})$.

The 1-NN search process is the dominating factor in the scalability of *RK-GA*$_0$ when applied to large data sets. We will show that the time and storage costs the proposed *IM-GA* mating strategies is negligible compared to that of 1-NN search.

4.4.4 Improving the Baseline *RK-GA*$_0$ with *IM-GA*: *RK-GA*$_1$, *RK-GA*$_2$, *RK-GA*$_3$, *RK-GA*$_4$, and *RK-GA*$_5$

For the purposes of evaluating the impact of the proposed *IM-GA* mating strategies in the optimization of *RK-GA*$_0$, the following five combinations of *RK-GA*$_0$ and one of the *IM-GA* mating strategies are defined:

- *RK-GA*$_1$: Let this be a new GA resulting from the adoption by *RK-GA*$_0$ of *IM-GA* single-population Strategy 1 in its mating process;
- *RK-GA*$_2$: Let this be a new GA resulting from the adoption by *RK-GA*$_0$ of *IM-GA* single-population Strategy 2 in its mating process;
- *RK-GA*$_3$: Let this be a new GA resulting from the adoption by *RK-GA*$_0$ of *IM-GA* single-population Strategy 3 in its mating process;

- *RK-GA*$_4$: Let this be a new GA resulting from the adoption by *RK-GA*$_0$ of *IM-GA* single-population Strategy 4 in its mating process, and;

- *RK-GA*$_5$: Let this be a new GA resulting from the adoption by *RK-GA*$_0$ of *IM-GA multi-population* Strategy 5 in its mating process.

These five different configurations of *RK-GA*$_0$ will be referred to in the experimental section. It is important to note that for *RK-GA*$_5$, the original single population of *RK-GA*$_0$ with 30 specimens was split into three *equally sized* populations, each having 10 specimens. This modification was done to ensure that all *IM-GA* mating strategies run with the same total number of specimens in the GA population.

4.4.5 The Computational Costs of the Baseline *RK-GA*$_0$ with *IM-GA*

It is imperative that the benefits of the *IM-GA* "instinct-based" mating strategies to *RK-GA*$_0$ are not outweighed by additional computational costs. In other words, the computation of the distance metrics representing *IM-GA*'s "mating instincts" should require only negligible additional computational costs when compared to *RK-GA*$_0$.

In theory, the costs of computing the *Complementary*, *Supplementary*, *Example Dissimilarity*, or *Attribute Dissimilarity* distance matrices are negligible when compared to the cost of 1-NN search. Recall that 1-NN search is needed to compute the specimens' training errors (see Section 4.4.3). The difference in costs is due to the following three reasons:

1. Error vectors are simply computed during classification by comparing predicted and actual class labels. These are simple boolean operations with negligible cost.

On the other hand, 1-NN search requires far more costly Euclidean distance computations;

2. Distance matrices are computed by comparing error vectors and attributes/examples sets, which are also simple boolean operations, and;

3. A distance matrix computation is a function of the population size N_P (small and constant), while 1-NN search is a function of the training set size (large and variable).

The computational time cost of *RK-GA*$_0$ remains unchanged with the adoption of any *IM-GA* mating strategy because of the high costs of 1-NN search discussed in Section 4.4.3. *In other words, IM-GA Strategies 1 through 5 can be used in applications of RK-GA*$_0$ *to very large data sets while requiring minimal additional computational overhead.*

As for the storage cost, it also remains unchanged for the same reasons. **Equation 4.19** shows that the additional cost of storing any of the $N_P \times N_P$ distance matrices representing *IM-GA*'s "mating instincts", plus N_P error vectors of length N_{ET} each, is negligible compared to that required by 1-NN search (recall N_P is a small constant for all data sets):

$$O(N_{ET} \cdot N_{AT}) + O({N_P}^2) + O(N_P \cdot N_{ET}) = O(N_{ET} \cdot N_{AT}) \tag{4.19}$$

The conclusion is that the computational costs of 1-NN search outweighs any costs introduced by the use of *IM-GA* to optimize the baseline *RK-GA*$_0$.

4.5 Testbed Problem 2 - The Optimal Decision Forests Problem

This section discusses how the the proposed *IM-GA* "instinct-based" mating strategies are used to optimize genetic search in a second testbed problem of Optimal Decision Forests, Testbed Problem 2. As was done for Testbed Problem 1, the discussion starts with the design of a novel GA capable of discovering optimal ensembles of decision trees (or decision forests). In addition, as in *RK-GA*$_0$ for Testbed Problem 1, this novel GA initially relies on the the conventional mating strategy to pair specimens in the GA population for recombination. The conventional mating strategy is then replaced by the *IM-GA* "instinct-based" mating strategies to further optimize the genetic search for optimal decision forests.

A key factor in the design of a GA for the discovery of optimal decision forests is the choice of the algorithm used to build the actual decision trees. While a few choices exist in the Machine Learning literature, this work adopted the well-known C4.5 Decision Trees program developed in Quinlan (1993)'s famous work as the decision tree builder. The C4.5 Decision Trees program was used by all ensemble learning methods, genetic and non-genetic, recruited for empirical experiments with Testbed Problem 2. The C4.5 Decision Trees program is a powerful decision trees builder. It is also one of the most frequently adopted learning methods applied to supervised classification problems. C4.5 Decision Trees uses the principle of "information gain ratio" to decide upon the optimal splits (or tests) on attributes needed to build a

decision tree model. C4.5 Decision Trees will be referred to as the "tree builder" for the remaining of this work.

Recall from the discussions in Section 4.4 that the *IM-GA* mating strategies cannot be used directly to solve search and optimization problems. This is because they are not a complete GA, but only a step of the GA iterative search process (mating). Hence, the first step in the process of evaluating *IM-GA*'s impact on genetic search applied to Testbed Problem 2, is to design a good GA capable of finding good "ensembles of decision trees" (or decision forests). The same design requirement fulfilled by the baseline *RK-GA*$_0$ in Testbed Problem 1 applies here; this novel GA should initially adopt, and work reasonably well, with the conventional mating strategy. This way, a good set of baseline experimental results on the performance of the conventional mating strategy can be attained. These results can then be compared to those attained by the application of the *IM-GA* "instinct-based" mating strategies in the mating process of the novel GA.

In this work, the novel GA designed to search for optimal decision forests is called *TM-GA*$_0$. The acronym "*TM*" is derived from the name of this novel GA's designers, namely, Thiago Quirino and Dr.Miroslav Kubat. *TM-GA*$_0$ adopts many GA design aspects from the framework of the original *RK-GA* (Rozsypal and Kubat 2003). For example, *TM-GA*$_0$ expands on the variable-length, value-encoded representation of chromosomes in specimens that was pioneered in the original *RK-GA* to handle large data sets. Recall that *RK-GA* represented each specimen as a pair of chromosomes (example and attribute chromosomes) which combined encoded a single 1-NN classifier. *TM-GA*$_0$ expands on this scheme for "genetic encoding of classifiers" by adopting

the use of multiple pairs of chromosomes per specimen. In *TM-GA*$_0$, specimens represent decision forests and each chromosome pair in a specimen represents a different C4.5 Decision Trees classifier in the ensemble. Furthermore, *TM-GA*$_0$ also naturally adopts the recombination and mutation operators developed for the original *RK-GA* to handle the custom variable-length, value-encoded chromosomes.

Experiments with various data sets from the UCI Machine Learning Repository discussed in Chapter 5 showed that *TM-GA*$_0$ is a good GA capable of discovering highly accurate and compact decision forests. *TM-GA*$_0$ performed favorably when compared to the state-of-the-art, classical approaches to ensemble learning such as Bagging (Breiman 1996), AdaBoost (Freund and Schapire 1996), and Random Forests (Breiman and Schapire 2001). *TM-GA*$_0$ proved to be a good baseline GA for investigating the impact of *IM-GA* "instinct-based" mating strategies on the performance of genetic search applied to the Optimal Decision Forests problem.

The next few sections show how the five proposed *IM-GA* mating strategies were incorporated into *TM-GA*$_0$'s framework. The experiments presented in Chapter 5 confirmed that *IM-GA* can indeed accelerate *TM-GA*$_0$ without impacting the quality of the discovered decision forests. As was done for the baseline *RK-GA*$_0$ in the discussions of Testbed Problem 1, it is also shown theoretically that the *IM-GA* mating strategies require negligible computational overhead when introduced into *TM-GA*$_0$. The end result is that *IM-GA* sped up *TM-GA*$_0$ in Testbed Problem 2 without harming its performance under any criteria.

Finally, *TM-GA*$_0$ will also be referred to as the baseline *TM-GA*$_0$ since it relies on the conventional mating strategy, just as the baseline *RK-GA*$_0$. *TM-GA*$_0$ is the

baseline GA used to investigate the impact of the *IM-GA* mating strategies on the optimization of Testbed Problem 2.

4.5.1 A Novel GA for Building Optimal Decision Forests, the Baseline *TM-GA*$_0$

The goal of the baseline *TM-GA*$_0$ is to discover optimal decision forests while relying on the conventional mating strategy. However, optimal decision forests are ensembles of optimal decision trees. In turn, an optimal decision tree is simply a concept that describes a highly accurate classifier having a model in the form of a decision tree structure that is also as compact as possible in order to minimize classification costs (i.e. average number of tests required to classify unknown examples) (Chikalov 2011; Murthy and Salzberg 1995). Building optimal decision trees has for decades been known to be an NP-Complete problem (Hyafil and Rivest 1976). That is, it is a costly problem to solve for which no algorithm is currently known to solve it in polynomial time (or "cheap" computational time). In supervised learning, classical approaches to decision tree induction do not build optimal decision trees (i.e. both accurate and compact decision tree models). The reason is that real-world data sets have noise. Consequently, classical tree builders end up learning both the useful *and* meaningless patterns available in their training data sets. The meaningless patterns are reflected in the actual structure of decision trees as custom noisy "tests" that make the decision trees unnecessarily larger (more extra nodes). The end result is large decision trees having high classification costs and lower overall generalization ability.

One way to create optimal decision trees is through optimization (by an external optimization tool) of the input parameters (data set and/or program arguments) that are fed into a classical decision tree builder (i.e. the C4.5 Decision Trees program or CART by Breiman et al. (1984)). This approach is very general and practical because it makes use of existing algorithms for building decision trees instead of requiring new ones to be custom-built into the chosen optimization tool. Classical decision tree builders are limited in their ability to determine the optimal choice of input parameters, such as the optimal subsets of examples and attributes from a training data set (i.e. data noise reduction), that optimizes the decision tree building process. Therefore, the optimization of the choice of input parameters must be undertaken by a good external optimization tool. In this work, the external optimization tool is the GA proposed in this section: the baseline *TM-GA*$_0$.

What this discussion establishes is that *TM-GA*$_0$ must simultaneously deal with two major *optimization tasks* in order to build optimal decision forests; one task at the decision trees level and the other at the ensemble (or decision forest) level. The simultaneous optimization of both objectives is an NP-complete problem (Chandra and Yao):

- *TM-GA*$_0$ must be able to optimize the individual decision trees making up an ensemble (or decision forest) by building both accurate, less over-fit, and compact decision tree models (optimal decision trees), and;

- *TM-GA*$_0$ must be able to optimally group decision trees into ensembles in a manner that the classification behaviors of the grouped decision trees "comple-

ment" each other to achieve higher combined predictive power (optimal decision forests).

TM-GA$_0$ tackles optimization task 1 by simultaneously searching for the optimal subsets of examples and attributes from a data set that optimizes the individual decision tree building process. To tackle optimization task 2, *TM-GA*$_0$ takes 2 steps. First, it allows specimens to vary in size (i.e. contain arbitrary number of example and attribute chromosome-pairs representing different decision trees) in order to represent decision forests of arbitrary size. Second, *TM-GA*$_0$ uses a novel "ensemble diversity" measure called the "triplet-fault" measure in its novel fitness-function. The novel "triplet-fault" diversity measure allows *TM-GA*$_0$ to evaluate the quality of its generated ensembles in terms of how well the grouped decision trees "complement" each others' classification errors. In this fashion, *TM-GA*$_0$ seeks to find good solutions to the Optimal Decision Forests problem.

Notice the natural connection between optimization objective 2 and the natural principle of "opposites-attract" that inspired this research. The intuitive "mating instinct" that can be derived from this connection above is that of the tendency of specimens (or decision forests) in the GA population to mate with those that complement their classification behavior. This "mating instinct" promotes the utilization of the genetic diversity (or available information) in the GA population, increasing the chances for pairing of specimens whose children can potentially become more accurate decision forests. It will be shown in the next sections how the *IM-GA* "instinct-based" mating strategies achieve that.

TM-GA$_0$ expands on many design aspects from the original *RK-GA*. For example, *TM-GA*$_0$ adopts the same variable length, real-encoded chromosome representation described in Section4.4.1. This is an important design choice because this particular chromosome representation pioneered by Rozsypal and Kubat (2003) greatly decreased the computational costs associated with representing large data sets as specimens in the GA. In the original *RK-GA*, each specimen had two chromosomes. The chromosome-pair represented the attribute and example sets used to build a unique 1-NN classifier. In contrast, *TM-GA*$_0$ uses multiple pairs of chromosomes per specimen. In *TM-GA*$_0$, each specimen represents a different decision forest. Moreover, each of the multiple chromosome-pairs in a specimen represents the example and attribute sets used to build a unique C4.5 Decision Trees classifier. The number of chromosome pairs in a specimen, as well as the size of each chromosome, will vary during the genetic search. This allows *TM-GA*$_0$ to perform a simultaneous search for both optimal decision trees which combined become optimal decision forests (having optimal number of ensemble members). The proposed multi-chromosomal structure of *TM-GA*$_0$ specimens is illustrated in **Figure 4.11**:

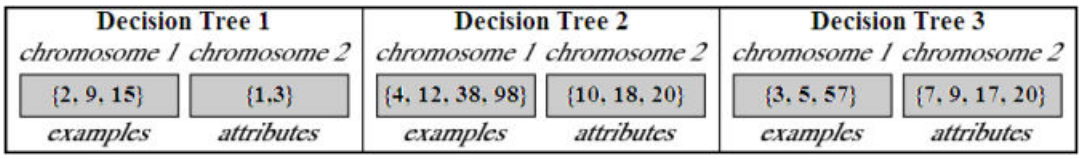

Figure 4.11: Sample baseline *TM-GA*$_0$ specimen with three example/attribute chromosome-pairs encoding an ensemble of three C4.5 Decision Trees classifiers.

Figure 4.11 depicts a specimen with three chromosome pairs, which is equivalent to a decision forest having three decision trees classifiers as members in an ensemble. Each of the three chromosome pairs represent a subset (example and attributes) of

the original training data set that will used to build a unique C4.4 Decision Trees classifier. For instance, the first chromosome pair (from left to right) in **Figure 4.11** "encodes" a decision tree built using training examples **{2,9,15}** whose values are defined using attributes **{1,3}** only (all other attributes and examples in the training set are ignored). Similarly, the second chromosome pair "encodes" a decision tree built using training examples **{4,12,38,98}** whose values are defined using attributes **{10,18,20}** only. Also, the third chromosome pair "encodes" a decision tree built using training examples **{3,5,57}** whose values are defined using attributes **{7,9,17,20}** only. When this specimen is decoded into a decision forest, three different C4.5 Decision Trees classifiers are built using the example/attribute sets defined in the three chromosome pairs. Finally, when the resulting decision forest is used to classify an example, which occurs either during the genetic search stage (to evaluate the training set error of the decision forest and build its error vector) or simply during the final evaluation stage (where the decision forest is evaluated on a separate testing set for estimation of its generalization error), each of the decision trees in the ensemble will classify that example to produce a class label. All of the class labels are then combined using a simple majority voting scheme (as is done by the Bagging classifier), where the "winning"class label is the one appearing most frequently. When majority voting leads to a tie between two or more class labels, the "winning" class label is then randomly chosen.

The Novel "Triplet-Fault" Measure of Ensemble Diversity

Prior to mathematically defining *TM-GA*$_0$'s fitness-function in the subsequent section, let's discuss a critical concept in the evaluation of the quality of decision forests during genetic search. This key concept is known as the "ensemble diversity" (or simply "diversity"), which was detailed in Section 3.4.3. The diversity indicates how well the individuals members of an ensemble complement each others' classification behavior. When applied to the Optimal Decision Forests problem, the diversity indicates whether the decision trees making up a decision forest tend to "err" on different examples. In *TM-GA*$_0$, optimization of the diversity among the decision trees making up a decision forest is one of the main optimization objectives. In *TM-GA*$_0$, the diversity is measured as a term in the fitness-function. The diversity term guides *TM-GA*$_0$ in the discovery of optimal groups of decision trees that tend to "err" on different examples. That is, it promotes the optimal grouping of decision trees whose combined predictions have higher predictive power than that of any individual decision tree in the ensemble. The end result is the discovery of optimal decision forests.

In this research, rigorous experiments were performed to investigate the impact of various diversity measures found in the literature (Kuncheva 2003) on the performance of *TM-GA*$_0$. The conclusion attained from these experiments was that an optimal "ensemble diversity" measure for the GA is one that does not penalize an ensemble for the individual imperfections (misclassifications) of its members as long as the overall ensemble accuracy is not impacted by these imperfections. Such a diversity measure provides the GA with a very "subtle hint", or an *implicit* sugges-

tion, that it is acceptable for the individual members of an ensemble to have some degree of misclassifications, but only as long as when combined, they will *complement* each other in a way that their majority vote correctly classifies any example. This "subtle hint" helps prevent the GA from over-fitting the individual classifiers because it *implicitly* (versus explicitly) promotes the existence of classification errors in the individual classifiers of an ensemble. Furthermore, it also allows the GA to consider diverse grouping of classifiers whose "complementary" classification behavior also reflect their diverse model structures.

Unfortunately, none of the existing "ensemble diversity" measures found in the literature seem to provide the GA with this "subtle hint." This issue stems from the fact that the existing diversity measures were not designed for use in the process of building individual classifiers, but rather on the process of grouping pre-built classifiers into ensembles (Kuncheva 2003). The ensemble generation techniques employing existing diversity measures are based on an *overproduced-and-select* paradigm, where a pool of classifiers is first generated using some existing technique (such as Bagging or AdaBoost) and a subset of the built classifiers are then grouped in a fashion that optimizes some diversity measure. However, when the existing diversity measures are applied to the process of simultaneously building classifiers and grouping them into ensembles, such as is accomplished by *TM-GA*$_0$, they fail to reflect that the imperfections of the individual members of an ensemble will not adversely impact the quality of an ensemble's predictions in every scenario. That is, the individual members' imperfections should count as penalties in the diversity measure only when their combined imperfections actually lead the ensemble to produce misclassifications.

Otherwise, from the ensemble point of view, the individual members' imperfections are harmless.

Let's discuss, for example, the impact of some of the diversity measures presented in the literature on the genetic search of *TM-GA*$_0$. The results discussed here were determined after rigorous experimentation with these measures on various UCI data sets. Note that diversity measures assign a diversity value to an ensemble. Therefore, when applied in *TM-GA*$_0$, diversity measures assign a diversity value to each specimen on the GA population. The two diversity measures considered here are pairwise measures. Pairwise diversity measures compute some metric for between every possible pair of members in an ensemble and then averages the results over the total number of possible pairs. Therefore, when pairwise measures are applied to *TM-GA*$_0$, a metric is computed between every possible pair of decision trees making up a decision forest, and then the results are averaged over the total number of decision tree pairs in the forest. The resulting diversity value is used as term in *TM-GA*$_0$'s fitness-function. The two measures considered here are:

1. The average pairwise Hamming distance between the error vectors of every pair of classifiers making an ensemble, which counts the number of "01" and "10" value-pairs in every possible pair of error vectors (and ignores the value-pairs "00", "00" and "11"), averaged over the total number of possible classifier pairs. This measure is to be "maximized" by the GA, and;

2. The double-fault measure between the error vectors of every pair of classifiers making an ensemble, which counts the number of "11" value-pairs in each pair

of error vectors (and ignores the value-pair combinations "00", "01", and "10"), averaged over the total number of possible classifier pairs. This measure is to be "minimized" by the GA.

Let's start with *diversity measure 1* above, the Hamming distance. Seeking to maximize this measure in *TM-GA*$_0$'s fitness-function for all specimens lead to the discovery of decision forests having decision trees with large classification errors. This is an intuitive result explained by the fact that the Hamming distance between a pair of error vectors from two different decision trees does not take into account how the misclassifications of the individual trees actually impact the ensemble predictions. The Hamming distance diversity measure is maximized when the error vectors of the decision trees making up the ensemble have orthogonal errors (disjoint errors on different examples). This is a major roadblock to the optimization of the accuracy of the decision trees by the GA because this measure *explicitly* requires large errors in the trees to exist in order to maximize the Hamming distance. In summary, the Hamming distance diversity measure completely ignores the impact that large errors in the decision trees have on the overall accuracy of the resulting decision forest.

In contrast, seeking to minimize the *diversity measure 2* in *TM-GA*$_0$'s fitness-function, the "double-fault" measure, led to the discovery of highly over-fit decision trees, and consequently, also decision forests with very poor generalization ability. Again, this is a very intuitive result that follows from two observations. First, the double-fault measure does not take into account that an example may be correctly classified by an ensemble even when a few individuals pairs of decision trees in a forest

misclassifies the example. Moreover, because the double-fault measure is so highly correlated to the classification error (it considers only the "11" error pattern between pairs of error vectors), it can lead *TM-GA*$_0$ to favor ensembles having decision trees that highly over-fit the training set. it was observed that the double-fault measure biased *TM-GA*$_0$ toward building ensembles with small number of trees. There were two reasons for this behavior. First, *TM-GA*$_0$ attempted to reduce (minimize) the number of "11" error patterns among pairs of error vectors by building highly accurate decision trees. However, classification accuracy is measured on the training set during genetic search, therefore, the generated decision trees are highly over-fit. Second, *TM-GA*$_0$ will tend to optimize the double-fault measure by also very quickly *dropping* the worst decision trees from the decision forests to improve the average error of the decision trees. The conclusion is that the double-fault measure alone does not promote the improvement of the decision trees in *TM-GA*$_0$. The double-fault measure frequently led the *TM-GA*$_0$ to converge the population to decision forests having small number of decision trees by quickly *dropping* the number of decision trees in the decision forests and over-fitting the resulting decision trees to reduce their training set classification error. Consequently, the generated decision forests exhibited very poor generalization error.

The discussion above illustrate the main issues encountered from the application of ensemble diversity measures presented in the literature in the genetic search for optimal classifier ensembles. To mitigate these issues, this work proposes a novel ensemble diversity measure called the "triplet-fault" measure that is to be minimized by the GA. The triplet-fault diversity measure is a triplet-wise measure, in contrast

to the pairwise measures commonly proposed in the literature. The triplet-fault measure considers the classification behavior of every possible combination of 3 classifiers in an ensemble (or sub-ensembles of 3 classifiers). Furthermore, the triplet-fault measure only penalizes the imperfections of individual ensemble members when those imperfections actually impact the overall accuracy of the individual sub-ensembles. In other words, harmless classification error from part of the individual members of a sub-ensemble are ignored as long as these errors do not impact the predictive power of the sub-ensembles. Consider, for example, an ensemble with M classifiers as members. The total number "**c**" of unique combination of 3 classifiers is the binomial coefficient (Kreyszig 1999) given by **Equation 4.20**:

$$\mathbf{c} = \frac{M!}{(M-3)! \cdot 3!} \tag{4.20}$$

For example, a decision forest having 5 decision trees have 10 unique combinations of 3 decision trees (or sub-ensembles). For each of these 10 (or **c**) sub-ensembles, the triplet-fault measure counts the total number of "110", "101", "011", and "111" value-triplets found in the corresponding error vectors decision trees making up the sub-ensemble. The total count is then averaged over the total number of sub-ensembles (which is 10 in this case). This is equivalent to counting the average number of examples misclassified by at least 2 or more decision trees in each sub-ensemble that can be created from the original 5 decision trees making up the parent decision forest. Notice how error vector value-triplets "000", "001", "010", and"100" are simply ignored in the count. These specific value-triplets do not count against classification

performance because they represent combination of 3 decision trees (or sub-ensemble) that correctly classified an example even though 1 out the 3 classifiers misclassified that example. That is, as long as the classifiers in a sub-ensemble can complement each others' errors to attain higher predictive power, the individual errors can be deemed harmless.

The above are crucial considerations for the success of *TM-GA*$_0$ in optimizing Testbed Problem 2. This is because the triplet-fault measures promotes the discovery by *TM-GA*$_0$ of decision forests made up of decision trees having diverse model structures. The diversity in the decision trees model structures is reflected by the acceptance of *TM-GA*$_0$ of certain classification error patterns among decision trees as harmless (e.g. "000", "001", "010", and"100"). This is achieved by using the triplet-fault measure as a term in fitness-function and attempting to minimize it. Notice that to minimize the triplet-fault measure, *TM-GA*$_0$ does not have to discover highly over-fitted classifiers (as required by the double-fault measure) or classifiers having large errors (as required by the Hamming distance). Instead, *TM-GA*$_0$ *simply has to discover classifiers that are accurate enough to complement each others' classification errors when combined into sub-ensembles of three classifiers.* Consequently, *TM-GA*$_0$ is able to consider certain grouping of decision trees into ensembles (or decision forests) that would otherwise be completely ignored had other ensemble diversity measures been adopted, such as those listed in the literature.

In summary, the novel "triplet-fault" diversity measure proposed in this work provides the GA with a "subtle hint" that none of the other diversity measures in the literature have so far provided: the errors of individual members of an ensemble are

not always harmful to the overall ensemble predictive power and those imperfections should be considered only when they impact the overall ensemble accuracy. This approach helps prevent the over-fitting of the individual classifiers during the genetic search and promotes more diverse combinations of classifiers into ensembles.

Specimen Fitness Evaluation

As in the original *RK-GA*, *TM-GA*$_0$ also utilizes the weighted sum approach described in (Coello 1999) to transform the complex, multi-objective optimization problem composed of (1) maximizing the classification accuracy of the decision forests (specimens in the GA population) and (2) discovering decision trees that are both highly accurate and compact, into a single-objective optimization problem.

Let's start the discussion by assuming that *TM-GA*$_0$ is attempting to find an optimal decision forest for an optimization problem based on a training data set having a total of N_{ET} training examples described by N_{AT} total attributes. Now, let's also consider an arbitrary specimen (or decision forest) in *TM-GA*$_0$'s population having a total of k decision trees as members in the ensemble. These k decision trees have many important properties that are relevant to their accuracy and model complexity. For example, let these k decision trees have a combined total of N_n decision nodes (where "n" stands for "$n$odes"). Also let the k decision trees require a combined total of N_t tests (where "t" stands for "$t$ests") to classify all N_{ET} examples in the training data set during the fitness evaluation process. Furthermore, let the combined total number of training examples misclassified by the k individual decision trees be E_c (where "c" stands for "$c$lassifiers"). Similarly, the ensemble as a whole also

has some important properties that are relevant to its accuracy and model complexity. For example, let the total number of examples misclassified by the decision forest (ensemble) be E_e (where "e" stands for "$e$nsemble"). Moreover, let the total number of unique attributes used to build the ensemble from among the N_{AT} original training attributes be N_a (where "a" stands for "$a$ttributes retained"). N_a is measured as the set size of the union of the k attribute chromosomes in the specimen. Finally, let D be the "Diversity" value of the specimen assigned by the novel "triplet-fault" measure that was introduced in the previous section. Considering all these properties, the fitness-function of *TM-GA*$_0$, $f_{TM\text{-}GA_0}$, is defined by **Equation 4.21**.

$$f_{TM\text{-}GA_0} = \frac{1}{[c_1 \cdot E_e + c_2 \cdot N_a + c_3 \cdot D] + [c_4 \cdot \frac{E_c}{k} + c_5 \cdot \frac{N_n}{k} + c_6 \cdot \frac{N_t}{k \cdot N_{ET}}]} \tag{4.21}$$

To define the conditions under which $f_{TM\text{-}GA_0} = 0$, let's first define another variable N_e the total number of unique example used to build the ensemble from among the N_{ET} original training examples (where "e" stands for "$e$xamples retained"). N_e is measured as the set size of the union of the k example chromosomes in the specimen. Note that minimizing N_e is not an optimization objective in *TM-GA*$_0$ for Testbed problem 2, hence, it is not present in **Equation 4.21**. However, the value of N_e is used as a test to determine when $f_{TM\text{-}GA_0} = 0$. The value of **Equation 4.21** is zero ($f_{TM\text{-}GA_0} = 0$) when $N_e = 0$, since no meaningful classifier ensemble can be built from "0" examples.

Now, let's understand what the terms in **Equation 4.21** mean. The first three terms in the denominator of **Equation 4.21** (purposely placed inside the first pair of

brackets) can be categorized as "ensemble-related" terms. This is because these terms are measures of ensemble classification error, complexity, and diversity, respectively. These terms guide the optimization by the GA of the overall ensemble properties (accuracy and model complexity). The remaining three terms inside the second pair of brackets can be categorized as "decision trees-related" terms because they guide the optimization by the GA of the individual decision tree classifiers making up the ensemble. For example, the first term inside the second pair of brackets corresponds to the average number of examples misclassified by all k decision trees (the average error of the trees). The second and third terms measure the complexity of the discovered decision trees. The second term corresponds to the average number of nodes in the k decision trees and the third term corresponds to the average number of tests required by the k trees combined (or by the ensemble) to classify an arbitrary training example.

Again, notice that the fitness-function of *TM-GA*$_0$ does not include any terms related to the size of the examples set used to build the decision trees or ensemble (N_e). That is because it is left up to the genetic search to find the optimal subset of examples that optimize the decision trees. In contrast, in Testbed Problem 1, the minimization of the example set was one of the the baseline *RK-GA*$_0$'s main optimization objective. That is because optimizing the 1-NN classifier actually requires the minimization of the examples set in order to reduce classification costs and eliminate noisy examples that impact classifier accuracy.

As for the sensitivity of *TM-GA*$_0$' performance to each of the six terms in **Equation 4.21**, rigorous experimentation has revealed that *TM-GA*$_0$ is mostly sensitive to the diversity term "D", which is a specimen's diversity value assigned by the triplet-

fault diversity measure. Removal of the diversity term "D" from **Equation 4.21** leads *TM-GA*$_0$ to discover highly over-fit decision forests with poor generalization abilities. This is an intuitive result because the diversity term "D" competes against other two terms: 1) the ensemble error and 2) the average error of the k decision trees. The combination of the three terms (diversity, ensemble error, and average decision trees error) promotes a balanced search for simultaneously accurate and diverse ensembles by the GA. When the diversity term "D" is removed from **Equation 4.21**, no other terms can compete against the combined terms of ensemble error and average error of the k decision trees. Notice that both terms are correlated to the classification error. Consequently, the *genetic pressure* exerted by these combined terms on *TM-GA*$_0$ leads to the discovery of decision trees that are highly over-fit to the training set, since the generalization error of a decision forest is estimated from the training set error during the genetic search. Therefore, in conclusion, the optimal performance of **Equation 4.21** is attained when all three terms (diversity, ensemble error, and average decision trees error) *compete* in the fitness-function.

As for the weights $c_1 \ldots c_6$, their values are all set to "1." This choice of weight values is dependent on the initial conditions of the example and attribute chromosomes of every specimen (the initial amount of data available to build the decision trees). To understand this dependency, let's first discuss the choice of initial conditions for the specimens' chromosomes in *TM-GA*$_0$Through rigorous experimentation, this research found that the genetic search for optimal decision forests works best when the decision trees are built from bottom up. That is, when the GA search starts with small and relatively inaccurate decision trees, which is achieved by filling

the example chromosomes with a small number of examples. This initial conditions provides 2 main advantages: 1)the initially small decision trees have not yet over-fit the training data set, and 2)more independent examples are left in the training data set to evaluate the discovered decision forests, which makes the training set error (used in fitness evaluations) a good indicator of the true generalization error of the discovered decision forests. Through the use of this initial conditions, the GA is able to simultaneously improve the decision trees and forests over many generations by carefully considering, right from the start of a run, the optimal subsets of examples and attributes that should be used to induce the most accurate and compact decision trees and forests.

On the other hand, had the specimens been initialized instead with large number of examples in their chromosomes, the initial decision trees would have been much larger. Consequently, they would also have relatively low initial training set errors and would have over-fit the training set from the start. This is a "bad" initial condition because it makes the training set classification error a poor indicator (i.e. a very optimistic indicator) of the true generalization error of the decision trees and forests. Under such initial conditions, the GA would not be able to build optimal decision trees because the highly over-fitted initial decision trees would not be easily *broken down* into smaller ones without impacting their initially high (and misleading) training set classification accuracies. This would result from the presence of meaningless, noisy test patterns in the decision trees that fit the noise in the training data set. This issues was confirmed in this research through rigorous experimentations performed with the

goal of determining the most optimal initial conditions for the chromosome-pairs in the specimens of *TM-GA*$_0$'s population.

Now, having discussed the optimal choice of initial conditions for *TM-GA*$_0$, let's return to the issue concerning the value of the weights $c_1 \ldots c_6$. The values of these weights are all set to "1" because the initial values of the terms of the fitness-function (given by **Equation 4.21**) already provide "natural weights" through their initial extreme opposite values. Recall from the discussion above that *TM-GA*$_0$ initializes the specimens' chromosomes with small amounts of examples, sufficient only to trigger the genetic search. What this also implies is that initially, the GA population is composed of specimens (or decision forests) having: 1)high ensemble error rates, 2)high average decision trees error rate, and 3)poor diversity (due to many classification errors). All these properties correspond to terms in **Equation 4.21** having initially high values. On the other hand, this also implies that the remaining terms in **Equation 4.21** have very low values. For example, small decision trees have small number of nodes (possibly being composed of single leaves) and require small average number of tests to classify any example. These initial conditions of the terms in **Equation 4.21** were found experimentally to be the most optimal because it initially sets different terms in the fitness-function at extreme opposite values. As the GA is run, the genetic search then proceeds to slowly *balance out* the terms in **Equation 4.21**, carefully considering the optimal subsets of examples and attributes needed to build the most accurate and compact decision trees. As result, no other weights are needed for any of the terms in **Equation 4.21**.

In summary, the fitness-function presented in this section is a good recipe for building optimal decision forests. All of the terms of the fitness-function given by **Equation 4.21** reflect a set of optimization objectives that have been regarded in the literature as *fundamental* for the discovery of optimal decision forests. These optimization objectives provide a balance between the relative importance of classification accuracy and model structural complexity (or size) both at the ensemble level and also at the level of individual decision trees.

Recombination

Recombination is the process of selecting parent specimens to mate and generate new children through crossover (exchange of genetic information between parent chromosomes). In *TM-GA*$_0$, recombination takes place at the decision trees level through the following five steps:

1. Parent-pairs (decision forests) are chosen randomly;
2. For each parent-pair, their decision trees are merged into a single pool;
3. The decision trees in the pool are randomly paired;
4. The paired decision trees are recombined to generate new decision trees (children), and;
5. The children trees are randomly grouped into two new decision forests.

The first step is mating (selecting pairs of parents to mate)). Assume *TM-GA*$_0$ has a population of N_P specimens (or decision forests). At each iteration of *TM-GA*$_0$,

$N_P/2$ pairs of parents are selected randomly based on fitness. The parents are selected probabilistically using the wheel-of-fortune based scheme given by **Equation 4.7**, just as was done in the original *RK-GA*. This specimen pairing scheme is the conventional mating strategy that *IM-GA* seeks to replace.

Then, steps 2, 3, and 4 are applied in sequence to each parent-pair chosen randomly in Step 1. For each chosen parent-pair, the second step involves creating a pool of decision trees containing all the decision trees from the chosen parent decision forests. The third step involves randomly shuffling the decision trees in the pool and pairing them up sequentially. Note that if the number of decision trees in the pool is odd, one of the decision trees will end up without a mating partner. In such cases, the decision tree without a mating partner is paired randomly with another tree in the pool. A decision tree in the pool never mates with itself, unless: 1)the parent decision forests correspond to the same specimen due to the random selection, or 2) the parent decision forests happen to have exact duplicates of the same decision trees. The first scenario will occur more often than the second scenario due to the use of the conventional mating strategy. Notice also that in this scheme, decision trees coming from the same parent specimen can mate with each other as well as with decision trees coming from other parents. This design choice makes the best use of the diversity (information) available in the pool by using all decision trees in the recombination process.

The fourth step involves generating children decision trees by recombining the parents decision trees that were paired randomly in the previous step. Recall that in *TM-GA*$_0$ each decision tree is represented as a chromosome-pair (attribute and

example chromosomes), just as in the original *RK-GA* each chromosome-pair represented a unique 1-NN classifier. Therefore, the same two-point cross-over operation employed in the original *RK-GA* to generate new 1-NN classifiers is also applicable in *TM-GA*$_0$ to generate new decision trees. The two-point crossover scheme referred here was illustrated in **Figure 4.10** for a single chromosome.

Finally, step 5 involves randomly grouping the generated children decision trees into two new children decision forests. Hence, the size of the children decision forests must be determined. Let N' be the the size of the original pool of parent decision trees, that is, the sum of the number of decision trees in the parent decision forests. A Gaussian distribution is then created with mean value $N/2$ (the average size of the parent decision forests) and ranging from 0 to N. A number n is then randomly picked from the Gaussian distribution. The first child decision forest will have the first n children decision trees and the second child decision forest will have the remaining $N - n$ child decision trees. The goal of this scheme is to promote moderate changes in the size of the specimens (decision forests) in the GA population, since most of the time the Gaussian distribution will lead to balanced sized forests that deviate little from the mean of the distribution. This allows the genetic search to slowly search for optimal decision forests of varying sizes. Other probability distributions, such as the uniform distribution ranging from 0 to N were investigated. However, random numbers picked from the uniform distribution frequently diverge from the mean ($N/2$), producing decision forests of unbalanced sizes when compared to the size of the parent decision forests. The use of the Gaussian distribution lead to the best experimental result.

Mutation

In each generation of *TM-GA*$_0$, mutation is applied to the decision trees making up each of the generated children decision forests. For each child decision forest, the mutation operator is applied independently to its decision trees. For each decision tree, the mutation operator randomly selects a specified percentage of the "alleles" (or fields) in both the example and attribute chromosomes of each decision tree and adds to them a randomly generated integer. The final mutated values are then modulo the total number of examples or attributes. This confides the final mutated values to a valid range that is data set relevant. The mutation rate chosen for *TM-GA*$_0$ is the same rate used in the original *RK-GA*: 5%. This mutation rate worked well for all experiments with UCI data sets.

Population and Survival

In *TM-GA*$_0$, the GA population size was fixed to a total of 60 specimens. Compared to the original *RK-GA*, this population size is twice as large. The reason is that Testbed Problem 2 is more complex than Testbed Problem 1. This is evident from the fact that a specimen in *TM-GA*$_0$ represents a decision forest with arbitrary number of decision trees. In contrast, a specimen in the original *RK-GA* represents a single 1-NN classifier. The search space of Testbed problem 2 is thus much larger than that of Testbed Problem 1 because combinations of decision tree classifiers must be considered. Consequently, *TM-GA*$_0$ requires a larger number of specimens than *RK-GA* to efficiently perform the genetic search for optimal decision forests. Notice that the increase in population size is, however, negligible when compared to the

differences in search space size. This fact illustrates one of the main advantages of genetic search: its ability to search through vast spaces of solutions with minimal computational requirements.

The population was initialized with 60 specimens at the beginning of the GA search process. Each of the 60 specimens were initialized with 7 random decision trees. Experiments showed no benefit from initializing the specimens with larger number of decision trees. At each generation, the size of the decision forests in the population varied due to the recombination process discussed above. The acceptable range of size for a decision forest was between 3 and 7 trees. The lower boundary for this range was chosen because of three reasons. First, decision forests having a single decision tree were never found to outperform larger decision forests, which is an intuitive result. Second, in decision forests having two decision trees, the decision trees were never able to complement each other's classification errors. These decision forests consistently lead to high number of class label ambiguities during the voting process. This means that the "winning" class label was frequently selected randomly. The third reason for requiring at least 3 decision trees per decision forest was that the triplet-fault diversity measure requires at least three decision trees.

The chromosome-pairs of each of the 7 decision trees in the 60 specimens making up- the population were initialized (their attribute and example chromosomes filled) with 1% of randomly selected training data examples and all attributes. The random values were picked from the uniform distribution over the range of the number of examples and attributes, respectively. This is the same initialization choice used in

the original *RK-GA*. The initial small amount of data in the chromosomes allows the GA to build optimal decision trees from the ground up.

Finally, in each new generation, 60 children specimens (decision forests) are generated, then merged with the original/surviving population. Finally, all decision forests, new and old, are then sorted by rank of fitness. The principle of survivor selection elitism (Bentley 1999) is then used to "*kill-off*" the worst half specimens, just as was done in the original *RK-GA*.

Termination Criterion

In each experiment, *TM-GA*$_0$ was run until no further improvement to the population fitness could be achieved. This is a different termination criteria than the one used in the original *RK-GA*, in which the GA was run for a max of 100 generations. Generally, *TM-GA*$_0$ run for anywhere between a few hundred generations (on smaller UCI data sets) to a much as 10,000+ generations (for larger UCI data sets) in order to discover optimal decision forests. This difference in the termination criteria further supports the fact that Testbed Problem 2 is more complex than Testbed Problem 1; *TM-GA*$_0$ spent much more effort in search of optimal decision forests (i.e. optimization of classifier ensembles) than the baseline *RK-GA*$_0$ spent in search of optimal 1-NN classifiers (i.e. optimization of a single classifier).

TM-GA$_0$ runs on UCI data sets were stopped when the fitness value of the top-fitness specimen of the population became stagnant (e.g. did not improve further) for at least 1000 generations. This fitness stagnation threshold performed well on all

experiments with the UCI data sets. This stopping criteria ensured that *TM-GA*$_0$ had ample search time to find stable solutions to Testbed Problem 2.

At the end of each *TM-GA*$_0$ run, the specimen (decision forest) having the lowest classification error was selected as the "winner." Most of the time, this specimen was also the top-fitness specimen in the population. Finally, the "winner" decision forest was then evaluated on the testing set in order to obtain an estimate of the generalization error of the optimal solution discovered by *TM-GA*$_0$ for Testbed Problem 2.

4.5.2 The Computational Costs of the Baseline *TM-GA*$_0$

The computational costs of both time and storage of the baseline *TM-GA*$_0$ are dominated by the costs associated with the *fitness evaluation* process: decision tree induction accompanied by its evaluation on the training data set. This is a similar scenario to that of the original *RK-GA* (as well as the baseline *RK-GA*$_0$), except that *RK-GA*'s computational costs were dominated simply by the *nearest-neighbor* search required to evaluate the accuracy of the generated 1-NN classifiers (the 1-NN classifier requires no induction). In *TM-GA*$_0$, during the fitness evaluation process, where the quality of the newly generated children decision forests is evaluated according to **Equation 4.21**, all of the decision trees making up the children decision forests have to be induced by the C4.5 Decision Trees program. Once they have been induced, they must also be evaluated on the training data set so that the ensemble error of the children decision forests can be computed. The resulting ensemble error values are then used in **Equation 4.21**. All of the other components of the GA

search process (i.e. mating, recombination, mutation, survival) add only negligible cost compared to the cost of inducing multiple decision trees per iteration of *TM-GA*$_0$ using the C4.5 Decision Trees program. The importance of examining the computational costs of *TM-GA*$_0$ is to ensure that the benefits of using the proposed *IM-GA* mating strategies are not outweighed by additional computational costs.

To derive the *computational time cost* of the decision tree induction process, let's consider the *worst-case scenario* of a decision tree that is to be built from the chromosome-pair of a specimen in *TM-GA*$_0$ that contains all N_{ET} examples and N_{AT} attributes from a large training data set. Also, assume that the training data set is composed entirely of N_{AT} numerical attributes, which are more costly for induction than nominal attributes because of required sorting procedures. Moreover, some assumptions about the size of the decision tree have to be made. Assume the worst-case scenario of a decision tree having N_{ET} leaves. The *standard depth* of a decision tree having N_{ET} leaves is in the order of $log(N_{ET})$ (I. Witten 2011). Now, in the worst-case scenario, at each of the possible depths of the induced decision tree, all N_{AT} attributes must be considered along with all N_{ET} examples. Since the depth of the tree is assumed to be in the order of $logN_{ET}$, the computational time cost of inducing the decision tree is given by **Equation 4.22**, where $DT_{Induction}$ stands for "*D*ecision *T*ree *I*nduction":

$$TimeCost(DT_{Induction}) = O(N_{ET} \cdot N_{AT} \cdot log(N_{ET})) \tag{4.22}$$

Now, let's compute the cost of evaluating the induced decision tree on the training data set. Given N_{ET} training examples and a decision tree depth in the order of $log(N_{ET})$, the computational time cost of evaluating the induced decision tree is given by **Equation 4.23**, where $DT_{Evaluation}$ stands for "*D*ecision *T*ree *E*valuation":

$$TimeCost(DT_{Evaluation}) = O(N_{ET} \cdot log(N_{ET})) \tag{4.23}$$

Assuming both large number of examples and attributes in the training data set, the total computational time cost associated with the induction and subsequent evaluation of the decision tree on the training data set is given by **Equation 4.24**:

$$TimeCost_{Total} = O(N_{ET} \cdot N_{AT} \cdot log(N_{ET})) + O(N_{ET} \cdot log(N_{ET})) = O(N_{ET} \cdot N_{AT} \cdot log(N_{ET})) \tag{4.24}$$

Similarly, let's consider the *computational storage cost* associated with storing the test nodes and leaves of the induced decision tree. For numerical attributes, let the *branching factor*[32] of each test node be "2". Thus, given that the depth of the induced decision tree is in the order $log(N_{ET})$, the computational cost of storing the nodes in the decision tree is given by **Equation 4.25**:

$$StorageCost_{Nodes} = O(2^{log(N_{ET})} - 1) = O(N_{ET}) \tag{4.25}$$

[32]The *branching factor* is the maximum number of values that an attribute can take on when used as a test node in a decision tree. For example, in the C4.5 Decision Trees program, test nodes corresponding to numeric attributes can take on only 2 possible values, thus the branching factor is "2".

Finally, the total computational cost of storing the test nodes plus the N_{ET} leaves of the induced decision tree is given by **Equation 4.26**:

$$StorageCost_{Total} = O(N_{ET}) + O(N_{ET}) = O(N_{ET}) \tag{4.26}$$

In summary, the computational time cost of *TM-GA*$_0$ is dominated by the decision tree induction process required for fitness evaluation of the generated children decision forests, as illustrated in **Equation 4.24**. Also, the computational storage of *TM-GA*$_0$ increases proportionally to the number of examples in the training data set, as illustrated by **Equation 4.26**.

4.5.3 Improving the Baseline *TM-GA*$_0$ with *IM-GA*: *TM-GA*$_2$, *TM-GA*$_4$, and *TM-GA*$_5$

In Testbed Problem 2, the impact of *IM-GA* Strategies 2, 4, and 5 on the performance of *TM-GA*$_0$ were investigated only. As will be discussed in Chapter 5, the results for Testbed Problem 1 revealed that all *IM-GA* strategies indeed succeeded in leading *RK-GA*$_0$ to faster convergence and that such was attained without impacting the quality of the generated solutions. However, from among the four single-population *IM-GA* mating strategies, *IM-GA* Strategies 2 and 4 performed better than their counterparts Strategies 1 and 3, respectively, when applied in Testbed Problem 1. This means that the fitness-based deterministic selection of the first parents in each mating pair performed better than the random selection. Moreover, since *IM-GA* Strategy 5 is the multi-population "flavor" of *IM-GA*, it was also imperative to also

retain this strategy as a representative *IM-GA* mating strategy for the experiments with *TM-GA*$_0$ in Testbed Problem 2.

For the purposes of evaluating the impact of the chosen *IM-GA* mating strategies in the optimization of *TM-GA*$_0$, the following 3 GAs are defined:

- *TM-GA*$_2$: Let this be a new GA resulting from the adoption by *TM-GA*$_0$ of *IM-GA* single-population Strategy $\underline{2}$ in its mating process;
- *TM-GA*$_4$: Let this be a new GA resulting from the adoption by *TM-GA*$_0$ of *IM-GA* single-population Strategy $\underline{4}$ in its mating process, and;
- *TM-GA*$_5$: Let this be a new GA resulting from the adoption by *TM-GA*$_0$ of *IM-GA* multi-population Strategy $\underline{5}$ in its mating process.

These five different configurations of the baseline *TM-GA*$_0$ will be referred to in the experimental section. It is important to note that for *TM-GA*$_5$, the original single population of *TM-GA*$_0$ with 60 specimens was split into three *equally sized* populations, each having 20 specimens. This modification was done to ensure that all *IM-GA* mating strategies run with the same total number of specimens in the GA population.

4.5.4 The Computational Costs of the Baseline *TM-GA*$_0$ with *IM-GA*

It is imperative that the benefits of the *IM-GA* "instinct-based" mating strategies to the baseline *TM-GA*$_0$ are not outweighed by additional computational costs. In other words, the computation of the distance metrics representing *IM-GA*'s "mating

instincts" should require only negligible additional computational costs when compared to *TM-GA*$_0$.

Recall from the discussion in Section 4.4.5 that the *IM-GA* mating strategies require negligible additional computational time for simple boolean operations. Hence, just as in the case of *RK-GA*$_0$, the computational time cost of the *TM-GA*$_0$ remains unchanged with the adoption of any *IM-GA* mating strategy because of the high costs of decision tree induction required by the fitness evaluation process, as was discussed in Section 4.5.2. *In other words, IM-GA Strategies 1 through 5 can be used in applications of TM-GA*$_0$ *to very large data sets while requiring minimal additional computational overhead.*

As for *TM-GA*$_0$'s computational storage cost, it also remains unchanged with the use of the *IM-GA* mating strategies. Let N_P be the number of specimens in *TM-GA*$_0$'s population, where N_P is a small constant for all data sets. Also, let N_{ET} and N_{AT} be the total number of examples and attributes in the training data set, respectively. **Equation 4.27** shows that the additional cost of storing any of the $N_P \times N_P$ distance matrices representing *IM-GA*'s "mating instincts", plus N_P error vectors of length N_{ET} each, do not change the overall computational storage costs of *TM-GA*$_0$ ($N_{ET} \gg N_P$ for all data sets):

$$O(N_{ET}) + O(N_{ET}) = O(N_{ET}) \tag{4.27}$$

The conclusion is that the computational costs of decision tree induction outweighs any costs introduced by the use of the *IM-GA* mating strategies to optimize *TM-GA*$_0$.

CHAPTER 5

Empirical Evaluation of the *IM-GA* Mating Strategies

This chapter is devoted to presenting the results of the rigorous experiments that were performed to evaluate the impact of the proposed *IM-GA* "instinct-based" mating strategies on genetic search applied to Testbed Problems 1 and 2.

Section 5.1 presents the experimental results for Testbed Problem 1. The results compare the performance of (1) the baseline *RK-GA*$_0$ with the conventional mating strategy versus (2) the baseline *RK-GA*$_0$ with the *IM-GA* mating strategies (see Section 4.4.4). In addition, the same section also compares the performance of *RK-GA*$_0$ against that of well-established, GA-based and non-GA-based techniques for solving Testbed problem 1.

Section 5.2 presents the experimental results for Testbed Problem 2. The results compare the performance of (1) the baseline *TM-GA*$_0$ with the conventional mating strategy versus (2) the baseline *TM-GA*$_0$ with the *IM-GA* mating strategies. Moreover, the same section also compares the performance of *TM-GA*$_0$ against that of the well-established ensemble learning techniques of Bagging, AdaBoost, and Random Forests for solving Testbed problem 2.

5.1 Performance of IM-GA in Testbed Problem 1

The experimental results presented in this section support the following three claims about the impact of the proposed *IM-GA* mating strategies in the baseline *RK-GA*$_0$ when applied to Testbed Problem 1:

1. Faster Convergence: The *IM-GA* "instinct-based" mating strategies *RK-GA*$_0$ to faster convergence in Testbed Problem 1.

2. Improved Quality of Solutions: The *IM-GA* "instinct-based" mating strategies occasionally improve the accuracy of the 1-NN classifier induced by *RK-GA*$_0$ and *do not impact* its quality along any of the following two criteria: a)percentage of examples retained, and b)ability to remove noisy/irrelvant attributes.

3. *RK-GA*$_0$ *outperforms* other GA-based and non-GA-based techniques to solving Testbed 1 Problem along all the following criteria: a)classification accuracy, b)percentage of examples retained, and c)ability to remove noisy/irrelvant attributes. This establishes that the baseline *RK-GA*$_0$ is not a *weak* GA and that the improvements achieved by the application of the *IM-GA* mating strategies in *RK-GA*$_0$ are real improvements.

Section 5.1.1 presents the UCI benchmark data sets used in all experiments with Testbed Problem 1. Section 5.1.2 list all the techniques, GA-based and non-GA-based, used in the experiments and discusses the parameters used to tune all the techniques for their best performances. Then, results for the *convergence speed*, which is the main performance criteria used to evaluate the impact of the *IM-GA* mating

strategies in Testbed problem 1, are presented in Section 5.1.3. The experimental results reported in Section 5.1.4 show that the improvements in *convergence speed* achieved by the *IM-GA* mating strategies were not achieved at the cost of other performance criteria. Finally, the same section also shows that *RK-GA*$_0$ outperforms other well-established, GA-based and non-GA-based techniques for solving Testbed Problem 1 along the various performance criteria discussed in Section 2.3.

In those result tables comparing the performance of *RK-GA*$_0$ *with* and *without* "instinct-based" mating, the bullet symbol ("•") indicates that the value achieved by an *IM-GA* strategy is statistically better than that achieved by the conventional mating strategy. Conversely, the checkbox symbol ("☒") identifies those cases where *IM-GA* performed statistically worse than the conventional mating strategy. Similarly, in those result tables comparing the performance of *RK-GA*$_0$ against other GA-based and non-GA-based benchmark techniques, the bullet symbol ("•") indicates that the performance of a benchmark technique is statistically better than that of *RK-GA*$_0$, while the checkbox symbol ("☒") indicates that a benchmark technique performed statistically worse than *RK-GA*$_0$.

Finally, for convenience, see Section 5.1.5 for all the tables of results.

5.1.1 Experimental Data

A total of 24 benchmark data sets from the UCI Machine Learning Repository (Newman and Merz 1998) were used in the experiments with Testbed problem 1. The training data examples were described by vectors of both numeric and nominal attributes. Following the example of Rozsypal and Kubat (2003, Quirino and Kubat

(2010), the UCI data sets were modified to make them better suited for the experiments. First, all examples having unknown attribute values were removed from each data set. Then, all numerical attributes were normalized to mean value 0 and standard deviation 1. Next, artificially created attributes were added to each data set. Given a data set with N_{AT} "original" attributes, a total of $2 \cdot N_{AT}$ attributes were added as random values picked from the *standard uniform distribution*. This modification was done, as described in Section 2.3.3, because in order to evaluate the GA's ability to discard irrelevant attributes from a data set, which is one the objectives of these experiments. Moreover, the UCI data sets are known to have few irrelevant attributes.

Table 5.1 summarizes the data set characteristics: the number of examples, the number of attributes, and the number of classes.

From an statistical point of view, most of the UCI data sets were not sufficiently large. Hence, the experiments with each UCI data set were run as 5-fold cross-validation[33], repeated 10 times for different seeds of the random number generator. This corresponds to a total of 50 experiments per data set for each technique. The statistical significance of the differences in performance among the various techniques was then tested by the *paired t-test* at a 5% confidence level.

[33] K-fold cross-validation is a technique for estimating the performance of a predictive model. A data set is randomly shuffled and then split into K equal parts. Then, $K-1$ parts are used for training the predictive model and the remaining part for evaluating the model. This process is repeated K times, each time using a different partition for evaluation of the model.

Table 5.1: Characteristics of the experimental UCI data sets in Testbed Problem 1.

Data Sets	No. of Classes	No. of Examples	No. of Original Attributes	No. of Original+Artificial Attributes
abalone	28	4177	8	24
balance	3	625	4	12
breast	2	683	9	27
bupa	2	345	6	18
car	4	1728	6	18
cmc	3	1473	9	27
crx	2	690	15	45
derm	6	358	34	102
glass	6	215	9	27
haber	2	306	3	9
heart	2	270	13	39
ion	2	351	34	102
iris	3	150	4	12
kr-vs-kp	2	3196	36	108
mushroom	2	5644	22	66
newthy	3	215	5	15
pima	2	768	8	24
segment	7	2310	19	57
sonar	2	208	60	180
vote	3	435	16	48
wdbc	2	569	30	90
wine	3	178	13	39
wpbc	2	194	33	99
yeast	10	1484	8	24

5.1.2 Reference Techniques

The primary goal of the experiments is to show that "instinct-based" mating indeed speeds-up convergence when implemented as an improvement to the baseline *RK-GA*$_0$. In this respect, *RK-GA*$_0$, which relies on the conventional mating strategy, is the reference point against which the performance of the *IM-GA* mating strategies will be gauged.

At the same time, it is imperative to compare the performance of *RK-GA*$_0$ against that of well-established techniques (although it is clearly not possible to make comparisons with every single existing technique) in order to ensure that *RK-GA*$_0$ is not a *weak* GA that could be easily improved by the *IM-GA* mating strategies.

From among existing GA-based approaches, the *CHC* (Eshelman 1991), *PBIL* (Cano, Herrera, and Lozano 2003), and *HT-GA*(Ishibuchi and Nakashima 2000) were chosen as representative techniques. These GAs were briefly described in Section 3.3.2.

To evaluate *RK-GA*$_0$'s ability to discard irrelevant attributes, the performance of its induced 1-NN classifiers were compared the that of the C4.5 Decision Trees program (Quinlan 1993)), which is known to be particularly good at this task. Also, following the example of Kuncheva and Jain (1999), the example-selection techniques E-NN and C-NN with the attribute-selection technique of Sequential Forward Selection (SFS) were combined sequentially to perform 1-NN Tuning. Kuncheva and Jain (1999) reported that the sequential application of these three techniques yielded a good balance between accuracy and data set reduction. This combination will be referred to by the acronym *WHSFS*.

Table 5.2 summarizes all the techniques employed in the experiments. In this table, the baseline *RK-GA*$_0$ is the technique sought to be improved in this work by the application of "instinct-based" mating in Testbed problem 1. Also in this table, the *RK-GA*$_1$, *RK-GA*$_2$, *RK-GA*$_3$, *RK-GA*$_4$, and *RK-GA*$_5$ (defined in Section 4.4.4) correspond to the five different combinations of *RK-GA*$_0$ with each of the five *IM-GA* "instinct-based" mating strategies proposed in Sections 4.2 and 4.3. All other remaining techniques were previously published in literature and are used in the experiments as benchmarks against which the performance of *RK-GA*$_0$ (with the conventional mating strategy) is compared to.

Every effort has been made to tune the reference techniques for their best performances, to make sure that the comparisons are fair.[34] Thus Rozsypal and Kubat (2003) reported good results of their original *RK-GA* when using 100 generations and population size $N_P = 30$. Therefore, the same values were used in *RK-GA*$_0$ (in the case of the *IM-GA* multi-population GA mating strategy, *IM-MP*, each of the three populations had $N_P/3 = 10$ specimens). In the case of *CHC*, the number of children per generation can vary from 0 to N_P. Hence, *CHC* was allowed to run for as many generations as needed to reach the same number of fitness-function evaluations as the other GAs ($N_P \times 100 = 3000$).

Following the suggestion from (Rozsypal and Kubat 2003), the *RK-GA*$_0$ was initialized by filling the example chromosomes with 10 uniformly distributed random numbers (from 1 to N_{ET}), and the attribute chromosomes with an ordered set of

[34]Tuning is the setting of classifier parameter values to optimize performance (e.g. fixing GA population size and fixing number of trees in decision forest).

Table 5.2: A summary of all GA-based and non-GA-based techniques used in the experiments for Testbed Problem 1.

Methods	**Acronyms**
***RK-GA*$_0$**	The algorithm developed by Rozsypal and Kubat (2003) and enhanced in Section 4.4.2 to use automatic weight selection. This technique relies on the *conventional mating strategy* and it is improved in this work by the application of the *IM-GA* "instinct-based" mating strategies in Testbed problem 1
***RK-GA*$_1$**	This GA is the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 1 (*IM-R-H*): *Instinct-based Mating* using the "Hamming" measure and "Random" (stochastic) selection of the first parents.
***RK-GA*$_2$**	This GA is the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 2 (*IM-D-H*): *Instinct-based Mating* using the "Hamming" measure and "Deterministic" selection of the first parents.
***RK-GA*$_3$**	This GA is the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 3 (*IM-R-CMW*): *Instinct-based Mating* using the novel "*Correct-My-Wrongs*" measure and "Random" (stochastic) selection of the first parents.
***RK-GA*$_4$**	This GA is the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 4 (*IM-D-CMW*): *Instinct-based Mating* using the novel "*Correct-My-Wrongs*" measure and "Deterministic" selection of the first parents.
***RK-GA*$_5$**	This GA is the combination of the baseline *RK-GA*$_0$ with *IM-GA* Strategy 5 (*IM-MP*): *Instinct-based Mating* using Multiple-Populations.
CHC	Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation genetic algorithm (Eshelman 1991).
PBIL	Population Based Incremental Learning genetic algorithm (Cano, Herrera, and Lozano 2003).
HT-GA	The GA based, 1-NN tuning algorithm developed by Ishibuchi and Nakashima (2000).
C4.5	The C4.5 Decision Trees program developed by Quinlan (1993).
WHSFS	Heuristic combination of Wilson's E-NN, proceeded by Hart's C-NN, and proceeded by Sequential Forward Selection (Kuncheva and Jain 1999).
1-NN	The simple, non-edited *nearest-neighbor* classifier (Cover and Hart 1967).

integers from 1 to N_{AT}. As for *CHC* and *HT-GA*, their chromosomes were initialized with the same mechanism. Note that if *CHC* been initialized as suggested by their authors, the *RK-GA*$_0$ would have conferred an unfair advantage (i.e. 10 instances vs. 50% of the data set on average) due to the large number of instances in the benchmark data sets. On the other hand, *PBIL* does not initially generate a population of chromosomes, but populates each new generation by sampling from a vector of probabilities (Cano, Herrera, and Lozano 2003). Hence, no interference was made to its initial probabilities setup. The *positive learning rate* in *PBIL* was set to 0.1 and the *negative learning rate* was set to 0.075. Also, the *mutation shift* was set to 0.05 and the mutation rate was set to 0.02 (Cano, Herrera, and Lozano 2003).

In *HT-GA*, the attribute bit mutation rate was set to 0.01 and the instance bit *1-to-0* and *0-to-1* mutation rates were set to 0.1 and 0.01, respectively. The weight W_{PF} was set to 5, and both weights W_F and W_P were set to 1 (Ishibuchi and Nakashima 2000).

In the C4.5 Decision Trees program, the default parameter values were used (including pruning). This enabled C4.5 to retain fewer attributes without compromising performance. The minimum number of instances per leaf was set to 2 and the number of folds for reduced error pruning was set to 3.

The mutation rate in *RK-GA*$_0$ was set to 5% for all experiments *with* and *without* the *IM-GA* mating strategies. This value was advocated by Rozsypal and Kubat (2003), who mentioned their results were unaffected by small variations of this value. For *CHC*, the population restart mutation rate was set to 35%.

5.1.3 Accelerated GA Convergence

The experimental results presented here compare the *convergence speed* of the baseline *RK-GA*$_0$ with (1) the conventional mating strategy versus with (2) the *IM-GA* "instinct-based" mating strategies. Recall from the discussions in Section 2.3.1 that the *convergence speed* is measured as the *time* required for the *average specimen accuracy*[35] of *RK-GA*$_0$ to reach a certain *convergence target*. *Time* is measured by the number of fitness-function evaluations. The *convergence target* for Testbed Problem 1 is set as the accuracy corresponding to the *non-edited* 1-NN classifier's accuracy. The *non-edited* 1-NN classifier's accuracy is obtained from Table 5.6, which presents averaged results from multiple cross-validation runs. Moreover, using the GA population's average accuracy is more objective than using the top-fitness specimen's accuracy because the latter can be good by mere chance.

Note that that the performance of *RK-GA*$_0$ is occasionally better than those reported in (Rozsypal and Kubat 2003) because of the more realistic weight parameter values used in the fitness-function (see a detailed explanation of this improvement in Section 4.4.2). Also, since the goal of this work is to improve *RK-GA*$_0$'s performance with "instinct-based" mating, the *convergence speed* of *RK-GA*$_0$ was not compared to that of the other GA-based methods (*CHC, PBIL* and *HT-GA*) so as not to mislead further discussion.

For each UCI data set and each strategy, Table 5.3 gives the time needed by the *average specimen* to reach the non-edited-1-NN accuracy from Table 5.6. The

[35]The average specimen accuracy is measured as the average classification accuracy of the classifiers represented by the GA population

bullet symbol ("•") indicates that the value achieved by an *IM-GA* mating strategy is statistically better than that achieved by *RK-GA*$_0$ (which relies on the conventional mating strategy). Conversely, the checkbox symbol ("☒") identifies those cases where *IM-GA* performed statistically worse than *RK-GA*$_0$.

The table shows that strategies *IM-D-CMW* (*RK-GA*$_4$), *IM-D-H* (*RK-GA*$_2$), *IM-R-CMW* (*RK-GA*$_3$), *IM-R-H* (*RK-GA*$_1$), and *IM-MP* (*RK-GA*$_5$) converged statistically faster than *RK-GA*$_0$ in nineteen, ten, seven, one, and thirteen data sets, respectively. The only case where the *convergence speed* of *RK-GA*$_0$ with an "instinct-based" mating strategy was slower than with the conventional mating strategy was when *IM-R-CMW* (*RK-GA*$_3$) was applied to the `cmc` data set. In all of the remaining cases, the *convergence speed* of "instinct-based" mating was comparable to that of *RK-GA*$_0$.

The conclusion is that the *IM-GA* "instinct-based" mating strategies speed up convergence in many UCI data sets, although different strategies exhibit somewhat different behavior. Also, the *IM-D-CMW* strategy (*RK-GA*$_4$) outperformed all other techniques. In five of the twenty-four data sets, no significant acceleration was observed.

5.1.4 Performance on Auxiliary Criteria

Having shown that the *IM-GA* "instinct-based" mating strategies tend to make the baseline *RK-GA*$_0$ convergence faster, it is imperative to ensure that this improvements do not come at the cost of lower quality of the induced 1-NN classifier. To achieve this,

the performance of *RK-GA*$_0$ *with* and *without* "instinct-based" mating is investigated along the three criteria that were described in Section 2.3:

1. Classification accuracy: The percentage of correctly classified testing examples (i.e. independent data set);

2. Example set reduction: The ability to choose a small training set, measured as the percentage of examples retained at the end of the GA run, and;

3. Attribute set reduction: The ability to remove irrelevant attributes from the training set, measured as the percentage of attributes retained at the end of the GA run.

Recall that the "original" attributes in the UCI data sets may be relevant, irrelevant, or redundant. However, the randomly generated attributes that were "artificially" added to the UCI data sets are *always* irrelevant. The 1-NN classifier classification accuracy may be adversely affected by the removal of the "original" attributes, but probably not by the removal of "artificial" attributes.

Note that that the performance of *RK-GA*$_0$ is occasionally better than those reported in (Rozsypal and Kubat 2003) because of the more realistic weight parameter values used in the fitness-function (see Section 4.4.2). The experimental results presented here primarily compare the behaviors of the proposed *IM-GA* "instinct-based" mating strategies to that *RK-GA*$_0$. To ensure that *RK-GA*$_0$ is not a *weak* GA that can be easily improved upon, performance comparisons of *RK-GA*$_0$ with some well-established techniques are also presented.

Classification Accuracy

Table 5.4 shows the classification accuracy achieved in the UCI data sets by the 1-NN classifiers obtained by *RK-GA*$_0$ with different *IM-GA* mating strategies. The bullet ("•") indicates that the value achieved by the given strategy is statistically better than that achieved by *RK-GA*$_0$ according to the paired t-test. The checkbox symbol ("☒") identifies those cases where the given strategy performed statistically worse than *RK-GA*$_0$ (according to the paired t-test).

The results indicate that all *IM-GA* mating strategies reach classification performances comparable to that of *RK-GA*$_0$. To be more specific, *IM-D-H* (*RK-GA*$_2$) significantly outperformed *RK-GA*$_0$ in five data sets, *IM-R-CMW* (*RK-GA*$_3$) and *IM-D-CMW* (*RK-GA*$_4$) outperformed *RK-GA*$_0$ in four data sets, *IM-R-H* (*RK-GA*$_1$) outperformed *RK-GA*$_0$ in three data sets, and *IM-MP* (*RK-GA*$_5$) outperformed *RK-GA*$_0$ in two data sets. Only in the `wpbc` data set were three out of five *IM-GA* mating strategies statistically worse. In all other cases, the results of the *IM-GA* "instinct-based" mating strategies were comparable to those of *RK-GA*$_0$.

The conclusion is that the *IM-GA* mating strategies are capable of optimizing the set of stored examples and their descriptions in a way that does not compromise the classification performance of the 1-NN classifier that uses them. Occasionally, the *IM-GA* mating strategies lead to classifiers that are more accurate than those obtained by *RK-GA*$_0$.

Tables 5.5 and 5.6 compare *RK-GA*$_0$'s classification performance with other GA-based and non-GA-based techniques, respectively. The results are very favorable for

RK-GA$_0$. The best of the "competitors" is the C4.5 Decision Trees program (C4.5) that performs statistically better in five data sets, worse in eleven, and comparably in eight.

Table 5.6 shows that both C4.5 and *WHSFS* perform extremely well in comparison to the simple, non-edited 1-NN classifier. Against this background, *RK-GA*$_0$'s ability to improve classification accuracy by removing less valuable examples and attributes is very impressive.

All of these results lead to the conclusion that the 1-NN classifiers obtained by *RK-GA*$_0$ outperform those obtained by the other investigated benchmark techniques. The experiments also indicated that the *IM-GA* mating strategies most of the time statistically outperformed all other techniques in terms of classification accuracy.

Example Set Reduction

The next task is to ensure that the *IM-GA* mating strategies do not produce larger sets of retained examples. Note that comparisons to both C4.5 and the non-edited 1-NN classifier are omitted because neither of these techniques attempt to reduce the set of training examples.

Table 5.7 summarizes the results of experiments that compare *RK-GA*$_0$ and the *IM-GA* mating strategies in terms of the average numbers of retained examples. Notice that the sizes of the resulting example subsets are in all data sets below 5.2%. Most of the time, only 1-2% of the examples were retained. All of the *IM-GA* mating strategies seem to be less successful in this task than *RK-GA*$_0$. However, this may be due to the bias toward *high classification accuracy* specimens in the "instinct-based

mating" strategies, which explain the higher accuracies of the *IM-GA* mating strategies in Table 5.4. However, the observed differences, while sometimes statistically significant, are small enough to be tolerated in practical applications. In addition, the observed differences are compensated by faster convergence and improved classification performance.

Table 5.8 compares the results of *RK-GA*$_0$ with those of GA-based and non-GA-based techniques. While all GA-based techniques exhibit impressive ability to reduce the size of the training set (e.g. the highest value is 21% of examples retained), *RK-GA*$_0$ managed to significantly outperform all of these techniques in all UCI data sets. Surprisingly, the performance of *WHSFS* were rather disappointing.

The conclusion is that *RK-GA*$_0$ performs better than the other techniques. While the *IM-GA* mating strategies performed tolerably worse than *RK-GA*$_0$ along this criterion, their *accelerated convergence* and improved classification accuracy outweigh the slightly increased training set size.

Attribute Set Reduction

Finally, the ability of all the techniques to discard irrelevant attributes is investigated. In the modified data sets used in the experiments, this ability can be observed along two different criteria: 1) how many of the "original" attributes are retained, and 2) how many of the "artificially" added (or irrelevant) attributes are retained.

Table 5.9 shows how *RK-GA*$_0$ and all *IM-GA* mating strategies optimized the "original" set of attributes. Notice that in comparison to *RK-GA*$_0$, the number of attributes retained by any *IM-GA* mating strategy is slightly (though always signif-

icantly) higher. This may be explained by the higher classification accuracies of the *IM-GA* mating strategies in Table 5.4.

The situation is different when it comes to the ability to remove "artificially" added irrelevant attributes. The results given in Table 5.10 indicate that the *IM-GA* strategies perform very well, discarding almost all of these irrelevant attributes. The only exception was the case where *IM-D-H* ($RK\text{-}GA_2$) performed worse than $RK\text{-}GA_0$ in the `wpbc` data set. In the `bupa` data set, *IM-R-H* ($RK\text{-}GA_1$) performed significantly better than $RK\text{-}GA_0$.

The results summarized so far reveal that the proposed *IM-GA* mating strategies tend to converge to specimens whose second chromosomes contain relevant attributes (i.e. containing some information about the class label). To corroborate this hypothesis, the data sets were discretized using the MDL discretization technique recommended by Fayyad and K.Irani (1993), and the *information gain* of the individual attributes were then computed. The attributes having the highest *information gain* were indeed found to be included in almost all of the final solutions discovered by any *IM-GA* mating strategy. It seems reasonable to assume that the GA should converge to this solution because, in 1-NN classifiers, such attributes yield the optimal separation between examples of different classes.

Tables 5.11 and 5.12 show that none of the GA-based and non-GA-based benchmark techniques performed as well as $RK\text{-}GA_0$ in handling the attributes set. Table 5.11 shows that the only statistically better result was when C4.5 was applied to the `derm` data set, while comparable results are seen in all benchmark techniques in the `balance` data set, for *PBIL* in the `iris` data set, for *WHSFS* in the `mushroom`

data set, and for the C4.5 Decision Trees Program in the `wine` data set. In all other data sets, their results are statistically worse than those of *RK-GA*$_0$.

Similarly, none of the GA-based and non-GA-based benchmark techniques performed as well as *RK-GA*$_0$ it comes to dealing with irrelevant attributes. *CHC*, *PBIL*, *HT-GA*, *WHSFS*, and C4.5 perform comparably to *RK-GA*$_0$ in one, eight, two, one, and eight data sets, respectively. Additionally, *PBIL* performed statistically better than *RK-GA*$_0$ in the `car` data set. All other results were statistically worse than those of *RK-GA*$_0$.

The conclusion from all these results is that the *IM-GA* "instinct-based" mating strategies perform only slightly worse than the baseline *RK-GA*$_0$ in retaining "original" attributes. However, this is compensated by the improved accuracy of the induced 1-NN classifiers and faster GA convergence. In addition, the results also suggest that *RK-GA*$_0$ performs significantly better than all the other tested techniques, GA-based and non-GA-based, in the task of optimizing the training data sets used to induce 1-NN classifiers.

5.1.5 Results Tables for Testbed Problem 1

All of the tables of results referenced in Sections 5.1.3 and 5.1.4 are presented in the following pages. The tables are organized in the order referred to in the paper.

Table 5.3: Time taken by the average specimen of the baseline $RK\text{-}GA_0$ versus the *IM-GA* mating strategies to reach the non-edited 1-NN classifier target accuracy given in Table 5.6.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	$\boldsymbol{RK\text{-}GA_1}$	$\boldsymbol{RK\text{-}GA_2}$	$\boldsymbol{RK\text{-}GA_3}$	$\boldsymbol{RK\text{-}GA_4}$	$\boldsymbol{RK\text{-}GA_5}$
abalone	65 ± 12.5	66 ± 12.7	63 ± 10.6	62 ± 9.5	59 ± 9.2 •	64 ± 10.3
balance	190 ± 35.0	194 ± 30.7	190 ± 33.0	195 ± 32.5	162 ± 23.7 •	180 ± 32.6
breast	131 ± 31.2	123 ± 24.7	120 ± 23.7 •	126 ± 24.5	112 ± 20.8 •	123 ± 21.6
bupa	8 ± 5.9	9 ± 6.8	9 ± 6.8	8 ± 6.2	10 ± 7.4	8 ± 6.4
car	46 ± 7.3	47 ± 7.6	48 ± 7.1	47 ± 5.7	46 ± 5.4	44 ± 6.5
cmc	64 ± 11.5	66 ± 11.5	65 ± 9.7	72 ± 13.5 ⊠	61 ± 10.7	60 ± 9.5 •
crx	130 ± 24.9	127 ± 23.6	122 ± 21.8 •	127 ± 22.6	105 ± 21.1 •	118 ± 21.4 •
derm	187 ± 28.2	193 ± 33.0	182 ± 26.0	173 ± 26.1 •	159 ± 21.4 •	166 ± 19.8 •
glass	253 ± 58.1	240 ± 61.1	232 ± 47.1 •	237 ± 43.4	210 ± 49.3 •	214 ± 59.2 •
haber	51 ± 12.0	55 ± 12.6	49 ± 7.3	53 ± 15.0	50 ± 9.4	54 ± 13.2
heart	73 ± 28.4	71 ± 27.2	69 ± 29.6	73 ± 30.0	61 ± 19.7 •	75 ± 26.2
ion	182 ± 34.7	172 ± 39.5	155 ± 33.5 •	161 ± 32.5 •	145 ± 31.4 •	144 ± 27.6 •
iris	90 ± 24.6	91 ± 22.8	89 ± 25.4	83 ± 17.7 •	78 ± 18.2 •	95 ± 22.7
kr-vs-kp	208 ± 42.0	201 ± 30.7	190 ± 24.9 •	193 ± 36.9 •	171 ± 26.2 •	182 ± 29.4 •
mushrrom	245 ± 29.6	237 ± 29.7	235 ± 32.3	237 ± 27.1	210 ± 21.6 •	215 ± 24.9 •
newthy	229 ± 46.3	210 ± 37.5 •	208 ± 33.5 •	215 ± 27.0 •	187 ± 31.3 •	202 ± 45.1 •
pima	106 ± 35.0	101 ± 27.1	97 ± 27.7	96 ± 21.9 •	86 ± 18.6 •	89 ± 23.3 •
segment	415 ± 50.8	407 ± 54.2	383 ± 41.0 •	415 ± 53.2	346 ± 52.0 •	373 ± 51.8 •
sonar	339 ± 71.1	336 ± 96.5	300 ± 73.6 •	315 ± 102.0	270 ± 81.7 •	320 ± 60.7
vote	174 ± 23.5	172 ± 19.2	166 ± 20.8 •	169 ± 18.3	152 ± 17.1 •	145 ± 17.0 •
wdbc	159 ± 34.2	161 ± 35.5	154 ± 36.4	166 ± 32.5	137 ± 31.8 •	163 ± 34.7
wine	209 ± 38.4	207 ± 39.9	199 ± 38.7	208 ± 38.7	184 ± 34.0 •	181 ± 34.0 •
wpbc	16 ± 10.8	15 ± 11.2	16 ± 9.6	13 ± 10.5	17 ± 9.6	15 ± 9.7
yeast	221 ± 42.6	213 ± 26.9	201 ± 27.0 •	204 ± 32.0 •	181 ± 30.6 •	189 ± 28.7 •

Table 5.4: Classification accuracy of the baseline $RK\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	$\boldsymbol{RK\text{-}GA_1}$	$\boldsymbol{RK\text{-}GA_2}$	$\boldsymbol{RK\text{-}GA_3}$	$\boldsymbol{RK\text{-}GA_4}$	$\boldsymbol{RK\text{-}GA_5}$
abalone	26.1 ± 1.8	26.0 ± 1.4	26.0 ± 1.5	26.2 ± 1.2	26.0 ± 1.3	26.1 ± 1.3
balance	87.1 ± 3.3	87.1 ± 3.5	87.8 ± 2.7	87.7 ± 2.8	87.4 ± 3.0	87.8 ± 3.3
breast	96.1 ± 1.6	96.2 ± 1.6	96.1 ± 1.8	96.3 ± 1.6	96.0 ± 1.5	96.1 ± 1.3
bupa	58.4 ± 7.5	65.0 ± 8.1 •	61.9 ± 9.2 •	64.3 ± 6.7 •	61.3 ± 8.2 •	59.6 ± 8.5
car	70.3 ± 2.7	72.0 ± 4.0 •	72.6 ± 4.9 •	72.2 ± 4.3 •	73.0 ± 4.9 •	70.8 ± 3.5
cmc	52.7 ± 4.0	53.6 ± 2.4	53.4 ± 3.0	53.3 ± 2.8	53.7 ± 2.9	53.4 ± 3.1
crx	86.4 ± 2.8	86.0 ± 2.8	86.0 ± 2.7	86.2 ± 2.5	85.9 ± 2.9	86.3 ± 2.3
derm	91.3 ± 3.5	91.5 ± 3.7	92.5 ± 4.1	91.9 ± 4.1	92.2 ± 3.6	92.3 ± 3.2
glass	65.3 ± 6.3	67.3 ± 7.1	67.2 ± 7.1	65.6 ± 7.5	66.4 ± 7.5	65.6 ± 6.6
haber	73.5 ± 4.9	74.0 ± 4.8	73.8 ± 4.9	74.1 ± 5.5	73.8 ± 5.2	72.7 ± 5.9
heart	80.8 ± 4.8	80.6 ± 5.2	80.4 ± 5.1	81.2 ± 5.3	80.6 ± 5.3	81.1 ± 4.8
ion	88.0 ± 3.2	88.6 ± 4.0	87.2 ± 3.8	88.1 ± 3.8	88.3 ± 3.9	88.7 ± 3.1
iris	93.8 ± 3.2	93.7 ± 3.6	93.8 ± 3.2	93.5 ± 3.9	93.3 ± 4.1	94.1 ± 3.4
kr-vs-kp	90.9 ± 5.7	92.3 ± 3.7	92.6 ± 3.1 •	93.6 ± 1.4 •	91.3 ± 5.1	91.5 ± 4.4
mushrrom	98.9 ± 0.6	99.5 ± 0.5 •	99.6 ± 0.4 •	99.5 ± 0.5 •	99.2 ± 0.6 •	99.2 ± 0.6 •
newthy	93.2 ± 4.6	93.2 ± 4.1	93.2 ± 3.9	92.7 ± 4.1	93.2 ± 4.1	93.6 ± 3.8
pima	75.3 ± 3.7	75.1 ± 3.6	75.7 ± 2.8	75.4 ± 2.5	75.6 ± 3.6	75.5 ± 3.0
segment	91.8 ± 1.6	92.1 ± 1.2	92.3 ± 1.2 •	92.2 ± 1.4	92.2 ± 1.2 •	92.3 ± 1.4 •
sonar	72.3 ± 6.2	72.3 ± 6.1	73.1 ± 7.0	73.1 ± 6.4	72.3 ± 6.9	72.2 ± 6.4
vote	97.0 ± 2.1	97.0 ± 2.3	97.0 ± 2.5	97.0 ± 2.3	97.0 ± 2.3	97.0 ± 2.4
wdbc	95.9 ± 1.8	96.0 ± 1.7	95.9 ± 1.8	95.8 ± 2.1	95.9 ± 1.7	95.9 ± 2.0
wine	93.8 ± 4.6	94.2 ± 4.6	93.4 ± 5.0	94.4 ± 4.2	93.6 ± 3.9	94.5 ± 4.0
wpbc	76.0 ± 7.1	73.3 ± 6.0 ☒	74.0 ± 5.8	72.5 ± 6.4 ☒	72.8 ± 6.0 ☒	76.0 ± 6.0
yeast	55.4 ± 3.0	55.9 ± 2.8	55.7 ± 2.6	55.6 ± 3.1	55.6 ± 3.2	55.7 ± 3.0

Table 5.5: Classification accuracy of the baseline $RK\text{-}GA_0$ versus the GA-based techniques *CHC*, *PBIL*, and *HT-GA*.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	CHC	PBIL	HT-GA
abalone	26.1 ± 1.8	22.0 ± 1.6 ☒	21.3 ± 1.4 ☒	18.6 ± 1.1 ☒
balance	87.1 ± 3.3	82.9 ± 4.7 ☒	85.0 ± 3.4 ☒	84.5 ± 3.4 ☒
breast	96.1 ± 1.6	95.8 ± 1.7	96.4 ± 1.2	95.6 ± 1.5 ☒
bupa	58.4 ± 7.5	54.2 ± 5.6 ☒	58.8 ± 6.4	53.9 ± 6.4 ☒
car	70.3 ± 2.7	81.0 ± 6.7 •	84.9 ± 3.2 •	69.8 ± 2.2
cmc	52.7 ± 4.0	42.6 ± 3.2 ☒	48.8 ± 3.0 ☒	41.2 ± 2.8 ☒
crx	86.4 ± 2.8	70.6 ± 5.8 ☒	84.5 ± 2.5 ☒	65.1 ± 4.4 ☒
derm	91.3 ± 3.5	65.8 ± 9.5 ☒	90.6 ± 4.2	39.6 ± 5.6 ☒
glass	65.3 ± 6.3	54.7 ± 9.6 ☒	67.2 ± 7.2	48.4 ± 7.4 ☒
haber	73.5 ± 4.9	69.9 ± 5.5 ☒	73.3 ± 5.1	72.3 ± 5.6
heart	80.8 ± 4.8	79.8 ± 4.7	81.7 ± 5.1	77.3 ± 4.6 ☒
ion	88.0 ± 3.2	83.3 ± 4.7 ☒	85.8 ± 4.5 ☒	81.3 ± 4.5 ☒
iris	93.8 ± 3.2	95.5 ± 3.7 •	94.6 ± 3.9	82.3 ± 8.8 ☒
kr-vs-kp	90.9 ± 5.7	53.6 ± 1.4 ☒	86.9 ± 8.5 ☒	54.4 ± 1.8 ☒
mushroom	98.9 ± 0.6	94.5 ± 3.7 ☒	99.9 ± 0.1 •	100.0 ± 0.0 •
newthy	93.2 ± 4.6	94.2 ± 4.1	93.4 ± 4.2	89.4 ± 4.3 ☒
pima	75.3 ± 3.7	69.5 ± 3.3 ☒	73.4 ± 3.2 ☒	70.2 ± 3.7 ☒
segment	91.8 ± 1.6	77.3 ± 4.6 ☒	93.9 ± 1.4 •	92.5 ± 1.2 •
sonar	72.3 ± 6.2	64.7 ± 7.1 ☒	67.4 ± 6.8 ☒	64.0 ± 6.9 ☒
vote	97.0 ± 2.1	89.6 ± 5.4 ☒	96.5 ± 2.8	79.6 ± 6.2 ☒
wdbc	95.9 ± 1.8	92.3 ± 2.7 ☒	94.4 ± 2.5 ☒	92.0 ± 2.7 ☒
wine	93.8 ± 4.6	93.5 ± 4.9	94.4 ± 3.9	88.2 ± 4.9 ☒
wpbc	76.0 ± 7.1	70.9 ± 6.5 ☒	73.8 ± 6.1 ☒	74.0 ± 7.1
yeast	55.4 ± 3.0	50.5 ± 4.3 ☒	53.6 ± 2.4 ☒	50.1 ± 4.7 ☒

Table 5.6: Classification accuracy of the baseline $RK\text{-}GA_0$ versus the non-GA-based techniques *WHSFS*, *C4.5*, and non-edited 1-NN classifier.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	WHSFS	C4.5	1-NN
abalone	26.1 ± 1.8	18.7 ± 1.4 ☒	24.1 ± 1.3 ☒	16.4 ± 1.1 ☒
balance	87.1 ± 3.3	78.4 ± 5.4 ☒	77.6 ± 3.6 ☒	70.2 ± 3.3 ☒
breast	96.1 ± 1.6	95.8 ± 1.6	94.4 ± 2.1 ☒	94.0 ± 1.5 ☒
bupa	58.4 ± 7.5	51.2 ± 5.4 ☒	61.1 ± 5.6 •	50.3 ± 6.4 ☒
car	70.3 ± 2.7	70.0 ± 6.7	87.3 ± 2.2 •	60.1 ± 2.4 ☒
cmc	52.7 ± 4.0	38.5 ± 3.1 ☒	50.5 ± 2.7 ☒	39.4 ± 2.7 ☒
crx	86.4 ± 2.8	79.7 ± 4.7 ☒	85.9 ± 2.4	62.8 ± 4.1 ☒
derm	91.3 ± 3.5	89.2 ± 4.5 ☒	92.5 ± 4.1	41.8 ± 5.5 ☒
glass	65.3 ± 6.3	50.3 ± 7.2 ☒	63.3 ± 6.8	51.3 ± 6.3 ☒
haber	73.5 ± 4.9	72.0 ± 4.6	71.0 ± 6.4 ☒	69.1 ± 6.0 ☒
heart	80.8 ± 4.8	77.7 ± 5.9 ☒	76.3 ± 6.1 ☒	70.6 ± 5.8 ☒
ion	88.0 ± 3.2	84.7 ± 5.7 ☒	88.7 ± 4.1	74.6 ± 5.6 ☒
iris	93.8 ± 3.2	93.5 ± 3.9	93.9 ± 3.9	75.6 ± 7.3 ☒
kr-vs-kp	90.9 ± 5.7	95.1 ± 1.7 •	98.9 ± 0.6 •	54.4 ± 1.9 ☒
mushroom	98.9 ± 0.6	99.6 ± 0.7 •	100.0 ± 0.1 •	79.7 ± 1.1 ☒
newthy	93.2 ± 4.6	91.3 ± 4.2 ☒	90.8 ± 4.6 ☒	86.2 ± 5.5 ☒
pima	75.3 ± 3.7	70.0 ± 3.1 ☒	72.4 ± 3.8 ☒	67.3 ± 3.6 ☒
segment	91.8 ± 1.6	94.5 ± 1.2 •	94.7 ± 1.1 •	64.6 ± 2.2 ☒
sonar	72.3 ± 6.2	75.8 ± 7.0 •	68.0 ± 7.2 ☒	64.6 ± 6.4 ☒
vote	97.0 ± 2.1	95.5 ± 3.7 ☒	96.8 ± 2.5	79.0 ± 5.4 ☒
wdbc	95.9 ± 1.8	93.5 ± 2.6 ☒	93.3 ± 2.7 ☒	88.0 ± 2.8 ☒
wine	93.8 ± 4.6	94.2 ± 3.4	89.2 ± 7.0 ☒	85.4 ± 5.2 ☒
wpbc	76.0 ± 7.1	73.9 ± 6.9	74.6 ± 6.7	62.7 ± 6.0 ☒
yeast	55.4 ± 3.0	42.2 ± 2.9 ☒	56.1 ± 3.0	38.6 ± 2.8 ☒

Table 5.7: Percentage of examples retained by the baseline $RK\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	$\boldsymbol{RK\text{-}GA_1}$	$\boldsymbol{RK\text{-}GA_2}$	$\boldsymbol{RK\text{-}GA_3}$	$\boldsymbol{RK\text{-}GA_4}$	$\boldsymbol{RK\text{-}GA_5}$
abalone	0.4 ± 0.1	0.4 ± 0.1 ☒	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1
balance	1.2 ± 0.5	1.2 ± 0.5	1.4 ± 0.6	1.3 ± 0.6	1.3 ± 0.4	1.3 ± 0.6
breast	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1 ☒	0.4 ± 0.1 ☒	0.4 ± 0.1	0.4 ± 0.1 ☒
bupa	0.9 ± 0.3	1.3 ± 0.4 ☒	1.2 ± 0.4 ☒	1.3 ± 0.4 ☒	1.3 ± 0.5 ☒	1.3 ± 0.5 ☒
car	0.1 ± 0.0	0.1 ± 0.1 ☒	0.2 ± 0.2 ☒	0.1 ± 0.1 ☒	0.2 ± 0.1 ☒	0.1 ± 0.1
cmc	0.8 ± 0.3	0.8 ± 0.4	1.0 ± 0.4 ☒	0.9 ± 0.4	0.8 ± 0.4	0.8 ± 0.3
crx	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1	0.4 ± 0.1
derm	4.3 ± 0.9	4.4 ± 0.9	4.6 ± 1.1 ☒	4.4 ± 1.0	4.3 ± 0.9	4.2 ± 1.0
glass	4.7 ± 1.3	5.2 ± 1.3 ☒	5.2 ± 1.3	5.1 ± 1.3	5.1 ± 1.1	4.6 ± 1.1
haber	0.8 ± 0.3	1.0 ± 0.3 ☒	0.9 ± 0.3 ☒	1.0 ± 0.2 ☒	1.0 ± 0.3 ☒	0.9 ± 0.3 ☒
heart	1.0 ± 0.3	1.0 ± 0.2	1.1 ± 0.2	1.1 ± 0.3	1.1 ± 0.3	1.1 ± 0.3
ion	1.5 ± 0.4	1.7 ± 0.4 ☒	1.8 ± 0.6 ☒	1.8 ± 0.6 ☒	1.7 ± 0.4 ☒	1.7 ± 0.4
iris	2.5 ± 0.0	2.5 ± 0.0	2.5 ± 0.0	2.5 ± 0.1	2.5 ± 0.0	2.5 ± 0.0
kr-vs-kp	0.2 ± 0.1	0.3 ± 0.1 ☒	0.2 ± 0.1	0.3 ± 0.3 ☒	0.2 ± 0.2	0.2 ± 0.1
mushrrom	0.2 ± 0.1	0.3 ± 0.1 ☒	0.3 ± 0.1 ☒	0.3 ± 0.2 ☒	0.2 ± 0.1 ☒	0.2 ± 0.1
newthy	1.9 ± 0.3	1.9 ± 0.3	1.9 ± 0.4	2.0 ± 0.3	1.9 ± 0.3	1.9 ± 0.3
pima	0.4 ± 0.2	0.5 ± 0.2 ☒	0.5 ± 0.2	0.5 ± 0.2 ☒	0.5 ± 0.2 ☒	0.5 ± 0.2
segment	2.4 ± 0.6	2.8 ± 0.6 ☒	2.6 ± 0.5	2.8 ± 0.6 ☒	2.7 ± 0.7 ☒	2.6 ± 0.6
sonar	1.4 ± 0.3	1.4 ± 0.4	1.4 ± 0.4	1.4 ± 0.4	1.4 ± 0.4	1.5 ± 0.5
vote	1.1 ± 0.0	1.1 ± 0.0	1.1 ± 0.0	1.1 ± 0.0	1.1 ± 0.0	1.1 ± 0.0
wdbc	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1
wine	2.2 ± 0.2	2.2 ± 0.3	2.3 ± 0.3	2.2 ± 0.3	2.2 ± 0.2	2.2 ± 0.3
wpbc	0.7 ± 0.3	1.2 ± 0.6 ☒	1.3 ± 0.7 ☒	1.3 ± 0.5 ☒	1.2 ± 0.6 ☒	0.7 ± 0.2
yeast	1.4 ± 0.4	1.7 ± 0.6 ☒	1.7 ± 0.5 ☒	1.7 ± 0.5 ☒	1.8 ± 0.6 ☒	1.6 ± 0.5 ☒

Table 5.8: Percentage of examples retained by the baseline $RK\text{-}GA_0$ and the GA-based techniques *CHC*, *PBIL*, and *HT-GA* and the non-GA-based technique *WHSFS*.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	CHC	PBIL	HT-GA	WHSFS
abalone	0.4 ± 0.1	0.5 ± 0.1 ☒	18.8 ± 0.6 ☒	5.6 ± 1.3 ☒	15.5 ± 0.7 ☒
balance	1.2 ± 0.5	7.2 ± 2.3 ☒	10.6 ± 1.0 ☒	7.7 ± 1.3 ☒	75.6 ± 1.7 ☒
breast	0.4 ± 0.1	1.9 ± 0.7 ☒	6.0 ± 0.8 ☒	5.1 ± 0.7 ☒	75.1 ± 13.7 ☒
bupa	0.9 ± 0.3	6.8 ± 3.6 ☒	10.6 ± 1.7 ☒	7.3 ± 1.4 ☒	49.9 ± 2.6 ☒
car	0.1 ± 0.0	11.3 ± 2.0 ☒	15.0 ± 1.4 ☒	1.8 ± 1.4 ☒	63.8 ± 1.4 ☒
cmc	0.8 ± 0.3	1.4 ± 0.2 ☒	16.3 ± 1.0 ☒	6.7 ± 0.6 ☒	40.0 ± 1.0 ☒
crx	0.4 ± 0.1	8.6 ± 4.2 ☒	11.3 ± 1.3 ☒	7.3 ± 1.1 ☒	64.8 ± 1.6 ☒
derm	4.3 ± 0.9	21.0 ± 2.8 ☒	16.8 ± 1.9 ☒	10.6 ± 1.6 ☒	41.9 ± 2.6 ☒
glass	4.7 ± 1.3	15.4 ± 2.8 ☒	13.2 ± 2.4 ☒	11.6 ± 1.7 ☒	48.7 ± 3.5 ☒
haber	0.8 ± 0.3	4.2 ± 1.9 ☒	3.7 ± 1.1 ☒	5.1 ± 1.3 ☒	63.9 ± 4.1 ☒
heart	1.0 ± 0.3	5.7 ± 2.1 ☒	6.2 ± 1.4 ☒	7.5 ± 1.5 ☒	71.3 ± 3.0 ☒
ion	1.5 ± 0.4	9.5 ± 2.2 ☒	11.3 ± 1.5 ☒	8.5 ± 1.4 ☒	58.6 ± 11.2 ☒
iris	2.5 ± 0.0	3.5 ± 1.1 ☒	2.9 ± 0.5 ☒	10.4 ± 2.4 ☒	74.0 ± 5.8 ☒
kr-vs-kp	0.2 ± 0.1	0.7 ± 0.1 ☒	19.2 ± 0.6 ☒	7.2 ± 0.4 ☒	56.4 ± 0.4 ☒
mushroom	0.2 ± 0.1	13.7 ± 2.7 ☒	15.1 ± 0.3 ☒	6.5 ± 0.3 ☒	84.5 ± 0.3 ☒
newthy	1.9 ± 0.3	3.5 ± 1.2 ☒	3.8 ± 1.0 ☒	6.4 ± 1.7 ☒	67.6 ± 12.0 ☒
pima	0.4 ± 0.2	2.4 ± 1.2 ☒	11.3 ± 1.1 ☒	6.8 ± 1.2 ☒	69.0 ± 2.5 ☒
segment	2.4 ± 0.6	15.0 ± 1.2 ☒	16.0 ± 0.7 ☒	9.6 ± 0.7 ☒	66.8 ± 1.2 ☒
sonar	1.4 ± 0.3	16.6 ± 3.6 ☒	12.4 ± 2.3 ☒	9.9 ± 1.9 ☒	67.4 ± 3.0 ☒
vote	1.1 ± 0.0	5.6 ± 2.5 ☒	5.3 ± 1.2 ☒	9.5 ± 1.5 ☒	78.9 ± 2.9 ☒
wdbc	0.5 ± 0.1	4.8 ± 1.3 ☒	9.0 ± 1.1 ☒	6.3 ± 0.9 ☒	84.5 ± 4.3 ☒
wine	2.2 ± 0.2	4.6 ± 1.7 ☒	5.1 ± 1.2 ☒	9.1 ± 1.8 ☒	86.9 ± 3.7 ☒
wpbc	0.7 ± 0.3	5.2 ± 1.9 ☒	5.1 ± 1.7 ☒	4.4 ± 2.4 ☒	48.3 ± 16.1 ☒
yeast	1.4 ± 0.4	13.9 ± 3.6 ☒	16.2 ± 0.9 ☒	8.2 ± 0.8 ☒	39.0 ± 1.4 ☒

Table 5.9: Percentage of original attributes retained by the baseline $RK\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	$\boldsymbol{RK\text{-}GA_1}$	$\boldsymbol{RK\text{-}GA_2}$	$\boldsymbol{RK\text{-}GA_3}$	$\boldsymbol{RK\text{-}GA_4}$	$\boldsymbol{RK\text{-}GA_5}$
abalone	25.3 ± 15.5	24.5 ± 10.7	23.0 ± 13.4	24.3 ± 12.7	27.0 ± 16.2	26.8 ± 15.8
balance	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
breast	42.7 ± 8.2	42.9 ± 7.8	44.4 ± 9.5	43.1 ± 9.2	42.7 ± 9.1	43.1 ± 9.2
bupa	24.3 ± 14.4	35.7 ± 10.1 ☒	34.0 ± 12.1 ☒	37.0 ± 11.3 ☒	33.7 ± 14.5 ☒	30.7 ± 12.8 ☒
car	8.3 ± 9.7	15.3 ± 15.7 ☒	18.3 ± 20.3 ☒	16.0 ± 18.7 ☒	20.0 ± 19.9 ☒	11.0 ± 13.3
cmc	30.9 ± 9.1	32.2 ± 8.2	34.0 ± 8.2 ☒	33.3 ± 8.4	32.9 ± 7.8	34.2 ± 10.0 ☒
crx	8.5 ± 4.7	9.2 ± 5.7	9.3 ± 5.2	10.4 ± 6.0 ☒	9.9 ± 5.6	7.5 ± 3.2
derm	30.9 ± 9.2	29.5 ± 7.7	29.5 ± 7.5	34.5 ± 9.5 ☒	29.4 ± 7.9	30.5 ± 7.3
glass	30.2 ± 9.3	31.3 ± 8.0	32.0 ± 8.0	31.6 ± 8.2	32.2 ± 9.0	32.7 ± 9.6
haber	34.7 ± 17.8	41.3 ± 18.5 ☒	43.3 ± 16.8 ☒	43.3 ± 15.4 ☒	42.7 ± 15.1 ☒	38.7 ± 21.7
heart	30.5 ± 8.1	32.8 ± 8.0	30.6 ± 7.4	32.9 ± 8.2	32.2 ± 9.3	31.4 ± 6.9
ion	7.5 ± 1.8	8.0 ± 2.4	8.5 ± 2.3 ☒	8.4 ± 2.0 ☒	8.1 ± 2.2	8.2 ± 1.8 ☒
iris	31.0 ± 10.8	34.0 ± 12.1	32.5 ± 11.6	34.0 ± 12.1	35.0 ± 12.4 ☒	31.5 ± 11.1
kr-vs-kp	10.8 ± 2.2	12.4 ± 3.3 ☒	12.2 ± 3.4 ☒	13.2 ± 3.1 ☒	11.9 ± 4.3	11.4 ± 2.3
mushrrom	11.2 ± 10.1	16.5 ± 6.2 ☒	18.0 ± 6.6 ☒	17.5 ± 8.1 ☒	14.9 ± 9.1 ☒	12.3 ± 6.2
newthy	47.6 ± 11.3	52.4 ± 13.3 ☒	53.6 ± 11.7 ☒	52.0 ± 12.8 ☒	51.6 ± 10.0 ☒	52.8 ± 12.0 ☒
pima	32.3 ± 10.4	36.3 ± 9.5 ☒	35.8 ± 10.1 ☒	35.5 ± 10.2	37.0 ± 10.7 ☒	34.8 ± 8.9
segment	33.8 ± 8.0	33.4 ± 7.4	34.9 ± 7.8	34.2 ± 7.2	32.1 ± 5.4	33.7 ± 6.3
sonar	3.9 ± 1.1	4.6 ± 1.6 ☒	4.4 ± 1.3 ☒	4.8 ± 1.4 ☒	4.1 ± 1.5	4.5 ± 1.7 ☒
vote	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0
wdbc	13.3 ± 3.5	14.4 ± 3.5	13.9 ± 3.2	14.1 ± 3.1	13.9 ± 3.9	14.9 ± 3.9 ☒
wine	31.4 ± 5.6	31.4 ± 4.6	30.8 ± 5.6	32.3 ± 5.8	32.5 ± 6.3	31.7 ± 5.7
wpbc	1.6 ± 1.8	3.6 ± 2.8 ☒	3.1 ± 3.0 ☒	4.0 ± 2.7 ☒	3.8 ± 2.8 ☒	1.5 ± 1.8
yeast	62.5 ± 9.4	66.3 ± 10.5 ☒	65.0 ± 11.3	69.8 ± 10.4 ☒	65.8 ± 10.4	66.3 ± 10.8 ☒

Table 5.10: Percentage of "artificial" (irrelevant) attributes retained by the baseline $RK\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$\boldsymbol{RK\text{-}GA_0}$	$\boldsymbol{RK\text{-}GA_1}$	$\boldsymbol{RK\text{-}GA_2}$	$\boldsymbol{RK\text{-}GA_3}$	$\boldsymbol{RK\text{-}GA_4}$	$\boldsymbol{RK\text{-}GA_5}$
abalone	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
balance	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 1.8
breast	0.3 ± 1.3	0.4 ± 1.5	0.2 ± 1.1	0.1 ± 0.8	0.4 ± 1.5	0.1 ± 0.8
bupa	2.8 ± 4.3	1.2 ± 4.1 •	1.5 ± 4.0	1.7 ± 4.8	1.8 ± 5.7	3.8 ± 7.4
car	4.5 ± 4.2	3.3 ± 4.1	3.8 ± 4.2	4.0 ± 4.2	3.3 ± 4.1	4.0 ± 4.2
cmc	0.1 ± 0.8	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.8	0.0 ± 0.0
crx	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
derm	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
glass	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
haber	10.0 ± 8.9	12.7 ± 7.9	11.7 ± 9.7	11.0 ± 8.7	10.3 ± 8.8	7.3 ± 8.4
heart	0.4 ± 1.2	0.5 ± 1.3	0.4 ± 1.4	0.5 ± 1.3	0.2 ± 0.8	0.5 ± 1.3
ion	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
iris	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
kr-vs-kp	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
mushrrom	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
newthy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
pima	0.9 ± 2.5	0.4 ± 1.5	1.0 ± 2.6	0.8 ± 2.4	0.5 ± 1.7	0.6 ± 1.9
segment	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
sonar	0.0 ± 0.1	0.0 ± 0.2	0.0 ± 0.1	0.0 ± 0.1	0.0 ± 0.1	0.0 ± 0.1
vote	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
wdbc	0.1 ± 0.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.2	0.1 ± 0.3	0.0 ± 0.0
wine	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
wpbc	0.8 ± 0.8	0.8 ± 0.8	1.2 ± 0.9 ⊠	0.9 ± 0.9	1.0 ± 1.3	0.9 ± 0.8
yeast	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

Table 5.11: Percentage of original attributes retained by the baseline *RK-GA*$_0$ versus the GA-based techniques *CHC*, *PBIL*, and *HT-GA* and non-GA-based techniques *WHSFS* and *C4.5*.

Data Set	***RK-GA***$_0$	CHC	PBIL	HT-GA	WHSFS	C4.5
abalone	25.3 ± 15.5	100.0 ± 0.0 ☒	76.8 ± 14.1 ☒	99.5 ± 2.5 ☒	99.5 ± 2.5 ☒	99.8 ± 1.8 ☒
balance	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
breast	42.7 ± 8.2	80.2 ± 11.9 ☒	54.9 ± 11.5 ☒	98.4 ± 3.9 ☒	75.8 ± 19.5 ☒	46.9 ± 11.9 ☒
bupa	24.3 ± 14.4	87.3 ± 19.8 ☒	49.3 ± 11.6 ☒	92.0 ± 10.8 ☒	94.3 ± 14.5 ☒	68.7 ± 19.8 ☒
car	8.3 ± 9.7	80.0 ± 14.3 ☒	64.7 ± 6.4 ☒	98.0 ± 6.4 ☒	94.0 ± 11.5 ☒	92.0 ± 8.4 ☒
cmc	30.9 ± 9.1	100.0 ± 0.0 ☒	49.8 ± 11.3 ☒	92.9 ± 8.3 ☒	99.3 ± 2.7 ☒	94.9 ± 7.2 ☒
crx	8.5 ± 4.7	78.5 ± 16.8 ☒	28.7 ± 7.5 ☒	92.7 ± 5.9 ☒	48.4 ± 17.8 ☒	28.1 ± 14.4 ☒
derm	30.9 ± 9.2	71.5 ± 7.5 ☒	50.8 ± 6.8 ☒	93.6 ± 3.8 ☒	47.2 ± 18.8 ☒	19.8 ± 2.0 •
glass	30.2 ± 9.3	71.8 ± 9.3 ☒	52.0 ± 11.3 ☒	95.1 ± 6.0 ☒	93.6 ± 13.7 ☒	61.6 ± 9.3 ☒
haber	34.7 ± 17.8	64.0 ± 23.2 ☒	52.0 ± 21.5 ☒	92.7 ± 13.9 ☒	93.3 ± 13.5 ☒	77.3 ± 31.2 ☒
heart	30.5 ± 8.1	60.8 ± 9.5 ☒	46.2 ± 7.5 ☒	93.7 ± 7.1 ☒	78.0 ± 16.0 ☒	42.9 ± 10.7 ☒
ion	7.5 ± 1.8	54.4 ± 8.7 ☒	33.6 ± 6.6 ☒	91.5 ± 4.3 ☒	27.7 ± 20.5 ☒	10.5 ± 4.6 ☒
iris	31.0 ± 10.8	50.5 ± 13.8 ☒	31.5 ± 11.1	98.0 ± 6.9 ☒	47.5 ± 18.4 ☒	63.5 ± 13.6 ☒
kr-vs-kp	10.8 ± 2.2	100.0 ± 0.0 ☒	47.9 ± 8.8 ☒	92.6 ± 7.3 ☒	50.5 ± 7.5 ☒	50.2 ± 5.5 ☒
mushroom	11.2 ± 10.1	95.5 ± 4.1 ☒	62.7 ± 12.2 ☒	80.5 ± 8.0 ☒	12.7 ± 1.8	30.9 ± 2.8 ☒
newthy	47.6 ± 11.3	73.2 ± 13.2 ☒	53.6 ± 11.0 ☒	98.4 ± 5.5 ☒	59.2 ± 14.0 ☒	71.6 ± 17.2 ☒
pima	32.3 ± 10.4	99.0 ± 4.9 ☒	49.8 ± 11.7 ☒	98.3 ± 4.4 ☒	98.8 ± 5.2 ☒	55.0 ± 16.4 ☒
segment	33.8 ± 8.0	87.8 ± 5.1 ☒	49.4 ± 10.2 ☒	75.4 ± 8.1 ☒	42.7 ± 22.7 ☒	52.9 ± 7.6 ☒
sonar	3.9 ± 1.1	36.2 ± 5.9 ☒	26.9 ± 5.1 ☒	83.3 ± 4.8 ☒	70.8 ± 24.9 ☒	5.6 ± 2.6 ☒
vote	6.3 ± 0.0	51.0 ± 7.7 ☒	10.0 ± 4.6 ☒	95.5 ± 5.4 ☒	16.6 ± 23.0 ☒	13.6 ± 3.3 ☒
wdbc	13.3 ± 3.5	67.6 ± 6.9 ☒	37.1 ± 6.6 ☒	90.4 ± 4.6 ☒	52.9 ± 31.2 ☒	14.8 ± 3.8 ☒
wine	31.4 ± 5.6	57.1 ± 10.0 ☒	40.9 ± 8.3 ☒	95.4 ± 6.6 ☒	46.3 ± 13.9 ☒	30.6 ± 6.5
wpbc	1.6 ± 1.8	14.8 ± 7.0 ☒	7.3 ± 4.3 ☒	47.3 ± 24.7 ☒	35.3 ± 31.5 ☒	4.5 ± 2.7 ☒
yeast	62.5 ± 9.4	93.8 ± 8.5 ☒	80.5 ± 8.8 ☒	95.5 ± 7.0 ☒	100.0 ± 0.0 ☒	97.0 ± 6.9 ☒

Table 5.12: Percentage of "artificial" (irrelevant) attributes retained by the baseline *RK-GA*$_0$ versus the GA-based techniques *CHC*, *PBIL*, and *HT-GA* and non-GA-based techniques *WHSFS* and *C4.5*.

Data Set	***RK-GA***$_0$	CHC	PBIL	HT-GA	WHSFS	C4.5
abalone	0.0 ± 0.0	100.0 ± 0.0 ☒	2.9 ± 4.0 ☒	86.8 ± 10.5 ☒	99.3 ± 2.4 ☒	98.8 ± 2.8 ☒
balance	0.0 ± 0.0	14.8 ± 32.2 ☒	0.0 ± 0.0	12.0 ± 22.0 ☒	52.8 ± 37.2 ☒	11.8 ± 11.4 ☒
breast	0.3 ± 1.3	31.1 ± 17.6 ☒	1.8 ± 3.8 ☒	69.8 ± 13.2 ☒	25.8 ± 16.1 ☒	1.2 ± 3.2 ☒
bupa	2.8 ± 4.3	71.8 ± 36.2 ☒	11.2 ± 11.6 ☒	92.2 ± 8.1 ☒	89.0 ± 13.1 ☒	11.8 ± 11.1 ☒
car	4.5 ± 4.2	3.8 ± 6.3	0.2 ± 1.2 •	97.2 ± 6.0 ☒	77.8 ± 39.9 ☒	24.7 ± 11.3 ☒
cmc	0.1 ± 0.8	100.0 ± 0.0 ☒	2.9 ± 5.3 ☒	92.0 ± 7.1 ☒	99.4 ± 2.0 ☒	54.4 ± 18.1 ☒
crx	0.0 ± 0.0	47.9 ± 37.2 ☒	0.7 ± 1.6 ☒	83.3 ± 7.7 ☒	8.6 ± 18.0 ☒	1.4 ± 2.3 ☒
derm	0.0 ± 0.0	12.5 ± 5.3 ☒	0.4 ± 0.8 ☒	75.0 ± 6.6 ☒	1.4 ± 1.2 ☒	0.1 ± 0.3
glass	0.0 ± 0.0	15.2 ± 13.6 ☒	0.2 ± 1.1	83.7 ± 9.2 ☒	59.9 ± 25.9 ☒	4.1 ± 5.7 ☒
haber	10.0 ± 8.9	39.3 ± 25.8 ☒	10.0 ± 8.9	85.7 ± 15.8 ☒	75.0 ± 29.8 ☒	10.0 ± 14.7
heart	0.4 ± 1.2	18.0 ± 8.7 ☒	3.2 ± 3.1 ☒	79.2 ± 10.9 ☒	40.7 ± 16.9 ☒	2.2 ± 2.9 ☒
ion	0.0 ± 0.0	23.4 ± 6.3 ☒	4.1 ± 2.7 ☒	76.0 ± 6.7 ☒	5.3 ± 11.4 ☒	0.2 ± 0.6 ☒
iris	0.0 ± 0.0	0.8 ± 3.0 ☒	0.0 ± 0.0	62.8 ± 27.1 ☒	5.5 ± 8.4 ☒	0.0 ± 0.0
kr-vs-kp	0.0 ± 0.0	100.0 ± 0.0 ☒	2.5 ± 4.3 ☒	87.4 ± 3.3 ☒	1.2 ± 0.6 ☒	0.3 ± 0.6 ☒
mushroom	0.0 ± 0.0	22.7 ± 7.7 ☒	1.3 ± 1.7 ☒	0.0 ± 0.0	1.8 ± 0.9 ☒	0.0 ± 0.0
newthy	0.0 ± 0.0	3.4 ± 4.8 ☒	0.0 ± 0.0	61.2 ± 22.2 ☒	2.4 ± 4.8 ☒	0.4 ± 2.0
pima	0.9 ± 2.5	96.8 ± 13.1 ☒	5.0 ± 7.4 ☒	84.5 ± 11.0 ☒	92.0 ± 13.3 ☒	11.1 ± 13.6 ☒
segment	0.0 ± 0.0	23.1 ± 9.3 ☒	0.1 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	1.9 ± 3.0 ☒
sonar	0.0 ± 0.1	20.8 ± 5.4 ☒	10.4 ± 2.8 ☒	78.1 ± 4.7 ☒	17.8 ± 16.3 ☒	0.4 ± 0.8 ☒
vote	0.0 ± 0.0	9.3 ± 6.2 ☒	0.1 ± 0.4	81.1 ± 9.1 ☒	2.3 ± 6.6 ☒	0.3 ± 0.9 ☒
wdbc	0.1 ± 0.3	36.5 ± 7.3 ☒	4.7 ± 2.8 ☒	80.0 ± 6.4 ☒	8.2 ± 10.4 ☒	0.1 ± 0.5
wine	0.0 ± 0.0	7.5 ± 5.7 ☒	0.2 ± 0.9 ☒	68.4 ± 13.2 ☒	2.9 ± 4.4 ☒	0.0 ± 0.0
wpbc	0.8 ± 0.8	14.8 ± 4.9 ☒	4.9 ± 2.8 ☒	48.5 ± 25.2 ☒	8.7 ± 10.7 ☒	0.8 ± 1.5
yeast	0.0 ± 0.0	17.4 ± 23.2 ☒	0.1 ± 0.9	26.3 ± 22.4 ☒	99.5 ± 2.1 ☒	40.3 ± 20.2 ☒

5.2 Performance of IM-GA in Testbed Problem 2

The experimental results presented in this section support the following three claims about the impact of the proposed *IM-GA* mating strategies in the baseline *TM-GA*$_0$ when applied to Testbed Problem 2:

1. Faster Convergence: The *IM-GA* "instinct-based" mating strategies lead *TM-GA*$_0$ to faster convergence in Testbed Problem 2.

2. Improved Quality of Solutions: The *IM-GA* "instinct-based" mating strategies *do not impact* the accuracy of the decision forests induced by *TM-GA*$_0$, occasionally improving *TM-GA*$_0$'s abilities to discard noisy/irrelevant attributes and build more compact decision forests (i.e. having decision trees with less nodes and requiring less tests to classify unknown examples).

3. *TM-GA*$_0$ performs comparably or better than the well-established, non-GA-based, classical ensemble generation techniques of Bagging, AdaBoost, and Random Forests in terms of classification accuracy, when the classical techniques are built with ensembles of both 25 and 50 decision trees. Also, *TM-GA*$_0$ generates smaller decision forests having significantly lower memory footprint (i.e. number of nodes), requiring significantly less tests to classify unknown examples (i.e. lower classification costs), and being significantly more *interpretable* than those generated by the classical techniques. This establishes that *TM-GA*$_0$ is not a *weak* GA and that the improvements achieved by the application of the *IM-GA* mating strategies in *TM-GA*$_0$ are real improvements.

Section 5.2.1 presents the UCI benchmark data sets used in all experiments with Testbed Problem 2. Section 5.2.2 list all the techniques, GA-based and non-GA-based, used in the experiments and discusses the parameters used to tune all of them for their best performances. Then, results for the *convergence speed*, which is the main performance criteria used to evaluate the impact of the *IM-GA* mating strategies in Testbed problem 2, are presented in Section 5.2.3. The experimental results reported in Section 5.2.4 show that the improvements in *convergence speed* achieved by the *IM-GA* mating strategies were not achieved at the cost of other performance criteria. Finally, the same section also shows that *TM-GA*$_0$ outperforms other well-established, classical techniques for ensemble generation along the various performance criteria discussed in Section 2.3.

In those result tables comparing the performance of *TM-GA*$_0$ *with* and *without* "instinct-based" mating, the bullet symbol ("•") indicates that the value achieved by an *IM-GA* strategy is statistically better than that achieved by the conventional mating strategy. Conversely, the checkbox symbol ("☒") identifies those cases where *IM-GA* performed statistically worse than the conventional mating strategy. Similarly, in those result tables comparing the performance of *TM-GA*$_0$ against the non-GA-based, classical benchmark techniques, the bullet symbol ("•") indicates that the performance of a benchmark technique is statistically better than that of *TM-GA*$_0$, while the checkbox symbol ("☒") indicates that a benchmark technique performed statistically worse than *TM-GA*$_0$.

Also, recall from the discussion in Section 4.5.3, that the impact of *IM-GA* Strategies 2, 4, and 5 on the performance of *TM-GA*$_0$ are investigated in Testbed Problem

2 (ignoring *IM-GA* Strategies 1 and 3) because of their superior performances in Testbed Problem 1.

Finally, for convenience, see Section 5.2.5 for all the tables of results.

5.2.1 Experimental Data

Similar to what was done for Testbed Problem 1, a total of 22 data sets from the UCI Repository (Newman and Merz 1998) were chosen to conduct the experiments with Testbed Problem 2. Due to limitations in the availability of computing power[36], it was not possible to evaluate *TM-GA*$_0$ on a few of the larger UCI data sets that were used in the evaluation of Testbed Problem 1. However, the total number of benchmark data sets used here is still large (i.e. 22 UCI data sets). These data sets were chosen because of their previous use in Testbed Problem 1 and also because of their diversity (i.e. varying in terms of number of examples, attributes, and classes).

Prior to experimentation, the data sets were modified as was done for Testbed Problem 1. Irrelevant attributes were added to each data set to evaluate *TM-GA*$_0$'s ability to discard noisy/irrelevant attributes from the data sets. This modification was done, following the example from previous works (Rozsypal and Kubat 2003; Quirino and Kubat 2010), because it follows that UCI data sets have mostly relevant attributes. The modification process is as follows: 1) all examples with unknown attribute values were removed from each data set, 2) all numerical attributes ("artificial" included) were normalized to mean 0 and standard deviation 1, and 3) irrelevant (or "artificial") attributes were added to each data set; where a data set contained

[36]This research challenge was detailed in Section 1.2.

N_A "original" attributes, $2 \cdot N_A$ irrelevant attributes were appended as randomly generated values from the *standard uniform distribution*. Table 5.13 summarizes the characteristics of all data sets: the numbers of examples, classes, original attributes, and original plus synthetic attributes.

Table 5.13: Characteristics of the experimental UCI data sets in Testbed Problem 2.

Data Sets	No. of Classes	No. of Examples	No. of Original Attributes	No. of Original+Artificial Attributes
balance	3	625	4	12
breast-c	2	277	9	27
breast-w	2	683	9	27
bupa	2	345	6	18
car	4	1728	6	18
cmc	3	1473	9	27
crx	2	690	15	45
derm	6	358	34	102
glass	6	215	9	27
haber	2	306	3	9
heart	2	270	13	39
ion	2	351	34	102
iris	3	150	4	12
newthy	3	215	5	15
pima	2	768	8	24
solar	6	1389	12	36
sonar	2	208	60	180
tic-tac-toe	2	958	9	27
vote	3	435	16	48
wdbc	2	569	30	90
wine	3	178	13	39
wpbc	2	194	33	99

As was the case in Testbed Problem 1, most of the UCI data sets were not sufficiently large from an statistical point of view. To account for this, all data sets were cross-validated[37] during experimentation. Each data set was randomly arranged 10

[37]N-fold cross-validation is the process of splitting a data set into N equal parts and subsequently training a classifier with N-1 parts and evaluating it on the remaining part, repeated N times.

times; the order of the examples was changed. For each random arrangement, 5-fold cross-validation was used. This resulted in a total of 50 experiments per data set for each technique employed in the experiments.

After experimentation, the chosen performance metrics described in Section 2.3 (i.e. classification accuracy, ability to remove irrelevant attributes, and decision forest size[38]) were validated statistically, to ensure the differences in performance among the techniques were not due to mere chance. To this end, the *paired t-test* at a 5%-confidence level was employed to assess the significance of performance differences. In summary, the experimental data sets choice, modification, 5-fold cross-validation, and statistical validation make up the platform upon which the experiments for Testbed Problem 2 are conducted.

5.2.2 Reference Techniques

The primary goal of the experiments is to show that "instinct-based" mating indeed speeds-up convergence when implemented as an improvement to the baseline *TM-GA*$_0$. In this respect, *TM-GA*$_0$, which relies on the conventional mating strategy, is the reference point against which the performance of the *IM-GA* mating strategies will be gauged.

At the same time, just as was done for the baseline *RK-GA*$_0$ in Testbed Problem 1, it is imperative to also compare the performance of the baseline *TM-GA*$_0$ against that of well-established techniques (although it is clearly not possible to make comparisons

[38]The size of a decision forest is measured using four criteria: 1) number of trees in the decision forest, 2) total number of nodes, 3) total number of leaves , and 4) average number of tests required to classify an unknown example.

with every single existing technique) in order to ensure that *TM-GA*$_0$ is not a *weak* GA that could be easily improved by the *IM-GA* mating strategies.

To this end, *TM-GA*$_0$ was compared to three well-known, classical techniques of classifier ensemble generation: 1) Bagging (Breiman 1996), 2) AdaBoost (Freund and Schapire 1996), and 3) Random Forest (Breiman and Schapire 2001). These renowned non-GA-based classical techniques generate accurate decision forests. Thus, *TM-GA*$_0$'s classification accuracy (as well as other performance criteria) is compared to that of each of these techniques to show that any improvement in *TM-GA*$_0$ due to the *IM-GA* mating strategies is a real improvement. Additionally, to measure accuracy, *TM-GA*$_0$ was compared to a fourth classical method of decision trees generation, the C4.5 Decision Trees program (Quinlan 1993) (C4.5). In addition to being used by both *TM-GA*$_0$ and the classical techniques to induce decision trees for their decision forests, C4.5 is also used as a baseline for comparison against that of the novel *TM-GA*$_0$ to ensure that the decision forests generated by *TM-GA*$_0$ are always more accurate than a single decision tree.

Table 5.14 summarizes all the techniques employed in the experiments. In this table, the baseline *TM-GA*$_0$ is the technique sought to be improved in this work by the application of "instinct-based" mating in Testbed Problem 2. Also in this table, the *TM-GA*$_2$, *TM-GA*$_4$, and *TM-GA*$_5$(defined in Section 4.5.3) correspond to the three combinations of *TM-GA*$_0$ with the *IM-GA* mating strategies that best performed in Testbed Problem 1; *IM-D-H*(single-population Strategy 2), *IM-D-CMW*(single-population Strategy 4), and *IM-MP*(multi-population Strategy 5), which were defined in Sections 4.2 and 4.3. All other remaining techniques were previously published in

literature and are used in the experiments as benchmarks against which the performance of *TM-GA*$_0$ (with the conventional mating strategy) is compared to.

Table 5.14: A summary of all GA-based and non-GA-based techniques used in the experiments for Testbed Problem 2.

Methods	**Acronyms**
***TM-GA*$_0$**	The novel GA for the discovery of optimal (accurate and compact) decision forests developed in this work in Section 4.5.1. This technique relies on the *conventional mating strategy* and it is improved in this work by the application of the *IM-GA* "instinct-based" mating strategies in Testbed problem 2
***TM-GA*$_2$**	This GA is the combination of the baseline *TM-GA*$_0$ with *IM-GA* Strategy 2 (*IM-D-H*): *Instinct-based Mating* using the "Hamming" measure and "Deterministic" selection of the first parents.
***TM-GA*$_4$**	This GA is the combination of the baseline *TM-GA*$_0$ with *IM-GA* Strategy 4 (*IM-D-CMW*): *Instinct-based Mating* using the novel "*Correct-My-Wrongs*" measure and "Deterministic" selection of the first parents.
***TM-GA*$_5$**	This GA is the combination of the baseline *TM-GA*$_0$ with *IM-GA* Strategy 5 (*IM-MP*): *Instinct-based Mating* using Multiple-Populations.
Bagging	The Bootstrap Aggregating classifier ensemble generation technique developed by Breiman (1996).
AdaBoost	The Adaptive Boosting classifier ensemble generation technique developed by Freund and Schapire (1996).
RF	The Random Forests classifier ensemble generation technique developed by Breiman and Schapire (2001).
C4.5	The C4.5 Decision Trees program developed by Quinlan (1993).

Every effort has been made to tune the reference techniques for their best performances, to make sure that the comparisons are fair.[39] In particular, *TM-GA*$_0$ was tuned to optimize three criteria: 1) accuracy, 2) removal of irrelevant attributes, and 3) decision forest size. The three classical techniques (Bagging, AdaBoost, and Random Forests) and the C4.5 Decision Trees program were tuned to optimize only accuracy. The tuning process described below was done prior to running each technique.

[39]Tuning is the setting of classifier parameter values to optimize performance (e.g. fixing GA population size and fixing number of trees in decision forest).

The novel *TM-GA*$_0$ was tuned as follows: The GA population size was set to 60 specimens [40], $N_P = 60$. Each of the 60 specimens correspond to a decision forest, where each forest was initially set to 7 trees. Each tree then corresponds to a chromosome-pair (see Section 4.5.1). At any iteration of *TM-GA*$_0$ decision forests having less than 3 decision trees were discarded. It was observed through rigorous experimentation that decision forests having less than 3 decision trees were unable to "complement", and thereby correct, each others' classification errors.

Next, both the example and attribute chromosomes were initialized. The example chromosomes were initialized with 1% of randomly selected training examples (random integers ranging from 1 to N_{ET}, the data set size). The values chosen to initialize all attribute chromosomes were the same, the set of integers from 1 to N_{AT} (all attributes in a data set). At each iteration of *TM-GA*$_0$, 5% of randomly selected example and attribute chromosomes from all generated *children* decision forests were mutated to promote diversity[41].

In addition to initializing the chromosomes, the fitness-function was then tailored by setting coefficient values c_1-c_6 to "1" (see **Equation 4.21**). This neutral coefficient value was chosen because the choice of initial conditions for the chromosome-pairs of the specimens promotes initially large differences among the terms of the fitness-function given by **Equation 4.21** (see Section 4.5.1 for a detailed explanation). This large differences generate *natural weights* that sufficiently guide the genetic search.

[40] This is twice that of previous works (Rozsypal and Kubat 2003; Quirino and Kubat 2010) because this new problem has a much larger search space than 1-NN tuning (see Section 2.2), hence, a larger population was needed to avoid premature convergence.

[41] Previous works (Rozsypal and Kubat 2003; Quirino and Kubat 2010) found results were unaffected by small variations of this value. Larger values prevented the GA from converging to a solution.

After initialization and tailoring of coefficient values, a stopping criteria for the *TM-GA*$_0$ run was set. The GA run was stopped when the top fitness value of the population could not be further improved for at least 1,000 generations[42].

To tune the three classical techniques of decision forest generation (Bagging, AdaBoost, and Random Forests) a parameter common to all three was first set: the decision forest size, which must be set manually. Breiman (1996) says that in experiments with UCI data sets, improvements over single classifiers were evident when ensembles of 25 classifiers were used, and that no significant accuracy gains were attained from larger ensembles. Hence, relying on the findings of previous works, the size of the ensembles generated by Bagging, AdaBoost, and RandomForest was set to 25 decision trees. In addition, ensembles of 50 decision trees were also considered for the purpose of comparing the classification accuracy of *TM-GA*$_0$ against that of the classical techniques. The purpose of increasing the ensemble size beyond what was suggested in the literature (i.e. doubling from 25 to 50 decision trees) was to ensure that the three classical techniques were indeed tuned for their best performances.

Having set the forest size, we further tuned each individual technique. For Bagging and AdaBoost, the example sampling bag size was set to (default) 100% of training set size. For Random Forest, the number of randomly selected attributes used to build forests was set as suggested by Breiman and Schapire (2001): $log_2(N_{AT})$, where N_{AT} is the total number of attributes in the training data set.

[42]Better solutions were not generated beyond this threshold of fitness stagnation in either small or large UCI data sets. However, larger data sets required more generations to converge.

Finally, C4.5 Decision Trees program was tuned. Its purpose in these experiments is two-fold: 1) as an external "decision tree builder" for all techniques, new and classical, and 2) as a benchmark whose accuracy was compared against that of the baseline *TM-GA*$_0$. For the first purpose, C4.5 was tuned such that each decision forest generation technique optimized the decision trees on their own, instead of relying on C4.5 to reduce over-fitting. This is an important step in evaluating *TM-GA*$_0$'s ability to discard irrelevant attributes in the data sets. To this end, the "unpruned" option[43] with minimum 2 examples per leaf[44] was used in order to just build trees, not optimize them. Note that this setup should not impact the performance of the classical techniques since the *majority voting* scheme used in the classification process of ensemble methods (i.e. output "averaging") has an equivalent effect on the performance of ensemble systems to that of pruning a decision tree: reducing training set over-fitting.

However, when the C4.5 Decision Trees program is used for the second purpose (accuracy comparison against *TM-GA*$_0$) it was tuned for optimal accuracy by using the default program options. These options included decision tree pruning using the "subtree raising" operation (to reduce over-fitting), a confidence factor of 0.25 for pruning decision rules[45], and a minimum 2 examples per leaf.

In summary, the classical techniques used to validate the performance of *TM-GA*$_0$ were tuned using parameter values reported in the literature, and therefore provide a reliable basis upon which to show that *TM-GA*$_0$ is not a *weak* GA that can be easily

[43]Unpruned trees corresponds to C4.5 Decision Trees program option "-U".
[44]Number of leaves is set in the C4.5 Decision Trees program by (default) option "-M 2".
[45]Confidence factor is set in C4.5 Decision Trees program by (default) option "-C".

improved upon. Therefore, *TM-GA*$_0$ can be used with confidence to investigate the optimization abilities of the *IM-GA* "instinct-based" mating strategies when applied to Testbed Problem 2.

5.2.3 Accelerated GA Convergence

The experimental results presented here compare the *convergence speed* of the baseline *TM-GA*$_0$ with (1) the conventional mating strategy versus with (2) the *IM-GA* "instinct-based" mating strategies. Recall from the discussions in Section 2.3.1 that the *convergence speed* is measured as the *time* required for the *average specimen accuracy*[46][47] of *TM-GA*$_0$ to reach a certain *convergence target*. *Time* is measured by the number of fitness-function evaluations. The *convergence target* for Testbed Problem 2 is set as the accuracy corresponding to the *95$_{th}$-percentile* (i.e. 95%) of the C4.5 Decision Trees program's accuracy. C4.5's accuracy is obtained from Table 5.17, which presents averaged results from multiple cross-validation runs. This *convergence target* prevents *TM-GA*$_0$ from undershooting the target accuracy, since *TM-GA*$_0$ *explicitly* avoids over-fitting the decision trees during the genetic search process.

For each UCI data set and each strategy, Table 5.15 gives the time needed by the *average specimen* to reach the *95$_{th}$-percentile* of the C4.5 Decision Trees program's accuracy from Table 5.17. The bullet symbol ("•") indicates that the value achieved by an *IM-GA* mating strategy is statistically better than that achieved by *TM-GA*$_0$ (which relies on the conventional mating strategy). Conversely, the checkbox symbol

[46]The average specimen accuracy is measured as the average classification accuracy of the classifiers represented by the GA population.

[47]Using the GA population's average accuracy is more objective than using the top-fitness specimen's accuracy because the latter can be good by mere chance.

("⊠") identifies those cases where *IM-GA* performed statistically worse than *TM-GA*$_0$.

The table shows that strategies *IM-D-H* (*TM-GA*$_2$), *IM-D-CMW* (*TM-GA*$_4$), and *IM-MP* (*TM-GA*$_5$) converged statistically faster than *TM-GA*$_0$ in thirteen, twelve, and twelve data sets, respectively. In all of the remaining cases, the *convergence speed* of "instinct-based" mating was comparable to that of *TM-GA*$_0$.

The conclusion is that the *IM-GA* "instinct-based" mating strategies speed-up convergence in many UCI data sets. Also, all strategies performed comparably under this criterium. In 8 of the 22 data sets, no significant acceleration was observed.

5.2.4 Performance on Auxiliary Criteria

Having shown that the primary goal of *accelerated convergence* by the *IM-GA* "instinct-based" mating strategies on the baseline *TM-GA*$_0$ has been achieved, it is imperative to ensure that this improvements do not come at the cost of lower quality of the induced decision forest classifiers. To ascertain this, the performance of *TM-GA*$_0$ *with* and *without* "instinct-based" mating is investigated along the six criteria that were described in Section 2.3:

1. Classification accuracy: The percentage of correctly classified testing examples by the induced decision forests (i.e. independent data set);

2. Attribute set reduction: The ability to remove irrelevant attributes from the training data set, measured as the percentage of attributes retained at the end of the GA run by the induced decision forests, and;

3. Decision forest complexity: The size of an induced decision forest is measured using four relevant criteria: a) the number of trees making up the decision forest (or ensemble size), b) the total number of nodes in the decision forest, c) the total number of leaves (or decision rules) in the decision forest, and d) the sum of the "*average number of tests required to classify an unknown example*" by each decision trees in the decision forest.

Recall that the "original" attributes in the UCI data sets may be relevant, irrelevant, or redundant. However, the randomly generated attributes that were "artificially" added to the UCI data sets are *always* irrelevant. The accuracy of decision forests may be adversely affected by the removal of the "original" attributes, but probably not by the removal of "artificial" attributes.

The experimental results presented here primarily compare the behaviors of the proposed *IM-GA* "instinct-based" mating strategies to that of *TM-GA*$_0$. However, performance comparisons between *TM-GA*$_0$ and (1) the three classical ensemble generation techniques of Bagging, AdaBoost, and Random Forests and (2) the C4.5 Decision Trees program are also presented to ensure that *TM-GA*$_0$ is not a *weak* GA that can be easily improved upon by the *IM-GA* mating strategies.

Classification Accuracy

Table 5.16 shows the classification accuracy achieved by decision forests generated by *TM-GA*$_0$ with different *IM-GA* mating strategies in the UCI data sets . The results indicate that all *IM-GA* mating strategies reach classification accuracies comparable to that of *TM-GA*$_0$. The conclusion is that the *faster convergence* promoted by the

IM-GA "instinct-based" mating strategies on *TM-GA*$_0$ does not come at the cost of lower classification accuracy of the induced decision forests.

Tables 5.17 and 5.18, respectively, compare *TM-GA*$_0$'s classification accuracy against that achieved by the three classical techniques when using ensembles of both 25 and 50 decision trees. Table 5.17 also shows the C4.5 Decision Trees program accuracy for comparison against the baseline *TM-GA*$_0$. The bullet ("•") indicates that the value achieved by a given classical technique is statistically better than that achieved by *TM-GA*$_0$ according to the paired t-test. The checkbox symbol ("☒") identifies those cases where a given technique performed statistically worse than the *TM-GA*$_0$.

In Table 5.17, the results are very favorable for *TM-GA*$_0$, who outperforms all three classical techniques built ensembles of 25 decision trees. *TM-GA*$_0$ outperforms Bagging, AdaBoost, Random Forests, and C4.5 in four, six, seven, and twenty data sets, respectively. The best of the "competitors" are Adaboost and Random Forests, which outperformed *TM-GA*$_0$ in two and three data sets respectively. In all other cases, the performances are comparable. Notice that the decision forests induced by *TM-GA*$_0$ almost always outperform the C4.5.

However, the results in Table 5.18 are more balanced since the three classical techniques now employ ensembles of 50 decision trees instead of 25. Overall, the accuracy of all classical techniques improve with the larger ensemble size. However, *TM-GA*$_0$ still outperforms Bagging and AdaBoost in three and six data sets, respectively, being outperformed by AdaBoost in only four data sets. Also, *TM-GA*$_0$ ties with Random Forests, performing both statistically better and worse in five data sets. Note an im-

portant detail in these results: increasing the ensemble size of the classical techniques beyond what was suggested in the literature only *marginally* improved their classification accuracies. This is the reason why Breiman (1996) suggested ensembles of 25 decision trees in experiments with UCI data sets, because he found that this choice of ensemble size provided a good trade-off between ensemble accuracy and size; in some experiments, he reported that larger ensembles actually damaged performance.

At the same, the classification costs of the classical techniques *skyrocketed* by doubling their ensemble sizes from 25 to 50 decision trees, while the *interpretability* of their results were further severely worsened. However, it can be argued that even with ensembles of 25 decision trees, the *interpretability* of the decision forests generated by the classical techniques has already been irrecoverably damaged Breiman (1996). The next few sections will reveal that the results achieved by *TM-GA*$_0$ are done so with significantly smaller ensembles (in the order of 3 to 7 decision trees) requiring only a slim fraction of the classifications costs of the other techniques. *TM-GA*$_0$ can achieve that because it approaches the ensemble learning problem from multiple perspectives and not only accuracy, as is done by the classical techniques.

The conclusion from these results is that the *IM-GA* mating strategies do not impact the classification accuracy of the decision forests induced by *TM-GA*$_0$. Also, *TM-GA*$_0$ outperforms well-established classical techniques of ensemble generation in terms of classification accuracy, performing particularly well in *head-to-head* comparisons against ensembles of 25 decision trees and comparably in comparisons with larger ensembles of 50 decision trees.

Attribute Set Reduction

The next task is to ensure that the *IM-GA* mating strategies do not impact *TM-GA*$_0$'s ability to discard irrelevant attributes. In the modified data sets used in the experiments, this ability is observed along two different criteria: 1) how many of the "original" attributes are retained by the generated decision forests, and 2) how many of the "artificially" added (or irrelevant) attributes are retained by the generated decision forests.

Table 5.19 gives the percentage of "original" attributes retained by *TM-GA*$_0$ and all *IM-GA* mating strategies. The performances are similar, indicating that the *IM-GA* mating strategies do not impact *TM-GA*$_0$'s ability to optimize the "original" set of attributes. In particular, the *IM-D-H* (*TM-GA*$_2$) and *IM-MP* (*TM-GA*$_5$) strategies outperformed *TM-GA*$_0$ in two and one data sets, respectively. The only statistically worse result by an *IM-GA* strategy was when *IM-D-CMW* (*TM-GA*$_4$) was applied to the `breast-c` data set.

The results are similar when it comes to the ability to remove "artificially" added irrelevant attributes. Table 5.20 indicates that the *IM-GA* strategies perform as well as *TM-GA*$_0$ in this task, *IM-D-H* (*TM-GA*$_2$) and *IM-MP* (*TM-GA*$_5$) occasionally improving the results. The only exceptions were *IM-D-CMW* (*TM-GA*$_4$) when applied to the `crx` data set and *IM-MP* (*TM-GA*$_5$) when applied to the `breast-c` data set. Notice also that the increase in irrelevant attributes retained *IM-D-CMW* (*TM-GA*$_4$) is tolerable for practical applications (only 0.3%).

Table 5.21 shows that none of the classical techniques (using ensembles of 25 decision trees) performed as well as *TM-GA*$_0$ in handling the attributes set, the only exceptions being where all techniques were applied to the `balance` and `tic-tac-toe` data sets. The results were worse for ensembles of 50 decision trees. Table 5.22 shows that all classical techniques performed statistically worse in removing "artificially" added irrelevant attributes in all data sets. This result illustrates a major issue in classical ensemble generation techniques: their lack of provisions for optimizing the training data sets used to induce the classifiers. *TM-GA*$_0$, which relies on a multi-objective genetic search, has an advantage in that aspect. The classical techniques, in contrast, are simply statistical tools that require large number of classifiers to work properly, while having no provisions for optimizing the individual classifiers.

The conclusion from these results is that the *IM-GA* "instinct-based" mating strategies do not impact *TM-GA*$_0$'s ability to optimize the attribute set and discard irrelevant attributes. The results also show that *TM-GA*$_0$ statistically outperforms the other tested techniques in distinguishing between relevant and irrelevant attributes in a data set.

Classification Costs Reduction

Finally, let us ensure that the *IM-GA* mating strategies do not produce larger decision forests having higher classification costs and memory footprint. Also, let us investigate the complexity (or size) of the decision forests induced by the baseline *TM-GA*$_0$ when compared to those induced by the three classical techniques (using ensembles of 25 decision trees). The criteria used are: 1) the decision forest size (i.e. number

of decision trees), 2) the total number of nodes in the decision forest, 3) the total number of leaves in the forest, and 4) the total number of *average tests* required by the individual decision trees in a forest to classify an unknown example.

As an illustration of the average size of the decision forests induced by the different techniques, Table 5.23 gives the number of decision trees making up the decision forests induced by *TM-GA*$_0$ and all *IM-GA* strategies. No statistical significance comparisons are made because the other criteria will more adequately capture the structural complexity of the decision forests. Notice from Table 5.23 that the average size of the decision forests vary from 3 to 7 decision trees, most decision forest having an average of 3 decision trees. In contrast, the classical techniques required at least 25 to 50 decision trees to achieve classification accuracies similar to that of *TM-GA*$_0$. This results indicates that the decision forests induced by *TM-GA*$_0$ are more accurate (i.e. less over-fit) and more capable of correcting each others' classification errors (i.e. more diverse) than those induced by the classical techniques. This is a result of *TM-GA*$_0$'s ability to optimize the training data sets used to induce the individual decision trees.

Table 5.24 compares the total number of nodes in the decision forests induced by the baseline *TM-GA*$_0$ and the *IM-GA* mating strategies. The results are comparable, with all *IM-GA* strategies occasionally reducing the total number of nodes in their induced decision forests. More specifically, *IM-D-H* (*TM-GA*$_2$), *IM-D-CMW* (*TM-GA*$_4$), and *IM-MP* (*TM-GA*$_5$) outperformed *TM-GA*$_0$ in this task in four, three, and two data sets, respectively. *IM-D-CMW* (*TM-GA*$_4$) performs statistically worse in in the `crx` data set and *IM-MP* (*TM-GA*$_5$) in the `wine` data set. Table 5.25 indicates

that the results are identical when it comes to *IM-GA*'s ability to reduce the number of leaves in the decision forests. The *IM-GA* strategies occasionally reduce the total number of leaves in their induced decision forests when compared to *TM-GA*$_0$.

Tables 5.26 and 5.27 reveal that none of the three classical techniques (using 25 decision trees) are capable of inducing decision forests as *compact* as those induced by *TM-GA*$_0$. Instead, the decision forests induced by *TM-GA*$_0$ correspond to small fractions of the memory footprint (i.e. number of nodes and leaves) of those induced by the classical techniques. The classical techniques perform statistically worse than the baseline *TM-GA*$_0$ in all data sets when it comes to the ability to minimize the memory footprint of their induced decision forests. This is an intuitive result because, unlike *TM-GA*$_0$, the classical techniques have no provisions for optimizing the training data sets of their individual classifiers.

Finally, Table 5.28 compares the ability of *TM-GA*$_0$ and *IM-GA* mating strategies to reduce the classification costs of their induced decision forests. This table gives the sum of the *average number of tests required to classify an unknown example* by each decision tree in a decision forest. Once again, the *IM-GA* mating strategies perform comparably to *TM-GA*$_0$, occasionally generating decision forests having statistically lower classification costs. The only cases where an *IM-GA* strategy faired statistically worse was where *IM-D-H* (*TM-GA*$_2$) was applied to the `heart` data set and the *IM-D-CMW* (*TM-GA*$_4$) was applied to the `breast-c` and `solar` data sets. Conversely, *IM-D-H* performed statistically better in four data sets, *IM-D-CMW* in three data sets, and *IM-MP* (*TM-GA*$_5$) in two data sets. All other results were statistically comparable.

The results are very different when the classification costs of the classical techniques are compared to that of *TM-GA*$_0$. Table 5.29 shows that the classical techniques perform statistically worse in all data sets. The decision forests induced by *TM-GA*$_0$ require only a fraction of the average number of tests required by the decision forests induced by the classical techniques. Breiman (1996) has pointed out that the interpretability of the results of classifier ensembles is lost in exchange for higher classification accuracies achieved by lather ensembles. *TM-GA*$_0$ is able to recover that lost interpretability by inducing both accurate and compact, or optimal, ensembles of decision trees.

All the results for Testbed Problem 2 presented here lead to the conclusion that the *IM-GA* mating strategies are capable of accelerating the convergence speed of *TM-GA*$_0$ without impacting the accuracy or size of the induced decision forests, occasionally generating more compact decision forests. In addition, the results show that *TM-GA*$_0$ outperforms three classical techniques of Bagging, AdaBoost, and Random Forests in terms of accuracy (when the latter rely on both 25 and 50 decision trees) and also in terms of the ability to generate compact decision forests, having accurate and diverse decision trees with small memory footprint and reduced classification costs. In addition, the decision forests generated by *TM-GA*$_0$ always outperform the C4.5 classifier in terms of classification accuracy. The advantages of the decision forests generated by *TM-GA*$_0$, versus those generated by the classical techniques, are:

1. They are compact, making results easy to interpret when compared to large forests generated by classical methods;

2. They have small memory footprint, even when converted into decision rules (i.e. reduced number of leaves) and can be stored in memory-constrained micro-chips, and;

3. They have a fast response time (i.e. reduced classification costs) that allow them to be used in applications where real-time response is critical, and otherwise large decision forests could not practically be employed.

5.2.5 Results Tables for Testbed Problem 2

All of the tables of results referenced in Sections 5.2.3 and 5.2.4 are presented in the following pages. The tables are organized in the order referred to in the paper.

Table 5.15: Time taken by the average specimen of the baseline $TM\text{-}GA_0$ versus the *IM-GA* mating strategies to reach the 95_{th}-percentile of the C4.5 Decision Trees target accuracy given in Table 5.17.

Data Set	$\boldsymbol{TM\text{-}GA_0}$	$\boldsymbol{TM\text{-}GA_2}$	$\boldsymbol{TM\text{-}GA_4}$	$\boldsymbol{TM\text{-}GA_5}$
balance	1165.9 ± 341.2	1012.1 ± 291.7 •	981.4 ± 329.3 •	1049.8 ± 320.3 •
breast-c	2.2 ± 8.9	1.5 ± 6.0	3.1 ± 11.3	2.1 ± 6.8
breast-w	37.5 ± 19.5	39.9 ± 17.1	37.3 ± 18.5	34.7 ± 18.1
bupa	74.0 ± 40.7	71.6 ± 17.7	73.2 ± 18.6	71.6 ± 16.8
car	7451.6 ± 3810.1	5899.4 ± 2855.4 •	5945.2 ± 2852.4 •	6061.8 ± 2415.0 •
cmc	1167.5 ± 519.8	1000.4 ± 329.1 •	1009.1 ± 413.7 •	1049.6 ± 396.3
crx	66.6 ± 18.8	59.0 ± 18.3 •	63.4 ± 20.3	61.7 ± 21.6
derm	4929.4 ± 721.4	4293.6 ± 689.6 •	4457.3 ± 789.9 •	4483.6 ± 750.0 •
glass	5580.2 ± 1658.8	4796.3 ± 1421.8 •	4848.7 ± 1898.7 •	4906.5 ± 1656.3 •
haber	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
heart	518.2 ± 76.5	486.4 ± 64.4 •	459.4 ± 53.0 •	470.5 ± 75.4 •
ion	1029.3 ± 211.0	903.5 ± 116.6 •	913.5 ± 146.4 •	912.9 ± 170.9 •
iris	1658.2 ± 194.5	1473.9 ± 199.6 •	1471.9 ± 156.1 •	1453.6 ± 195.3 •
newthy	3401.1 ± 806.4	2895.5 ± 667.4 •	3009.0 ± 730.4 •	2996.5 ± 1091.1 •
pima	27.0 ± 21.1	27.0 ± 21.2	28.6 ± 21.7	27.0 ± 18.7
solar	214.0 ± 52.0	197.2 ± 53.9	186.6 ± 44.2 •	180.1 ± 43.3 •
sonar	761.6 ± 132.0	686.0 ± 124.3 •	685.8 ± 111.6 •	673.9 ± 104.0 •
tic-tac-toe	17908.1 ± 7329.8	14850.8 ± 6172.4 •	15844.8 ± 8454.3	14513.4 ± 5209.6 •
vote	701.7 ± 98.4	700.5 ± 70.0	676.8 ± 71.2	684.0 ± 81.7
wdbc	128.3 ± 20.4	128.9 ± 22.4	129.3 ± 23.2	135.5 ± 23.0
wine	2521.9 ± 449.6	2325.7 ± 448.3 •	2166.6 ± 359.2 •	2083.5 ± 314.6 •
wpbc	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

Table 5.16: Classification accuracy of the baseline $TM\text{-}GA_0$ versus the $IM\text{-}GA$ mating strategies.

Data Set	$\boldsymbol{TM\text{-}GA_0}$	$\boldsymbol{TM\text{-}GA_2}$	$\boldsymbol{TM\text{-}GA_4}$	$\boldsymbol{TM\text{-}GA_5}$
balance	81.7 ± 3.4	81.6 ± 3.9	81.7 ± 3.3	81.5 ± 4.1
breast-c	72.4 ± 5.7	72.1 ± 5.5	72.0 ± 5.5	71.2 ± 5.8
breast-w	96.2 ± 1.6	95.8 ± 1.7	96.0 ± 1.4	96.2 ± 1.2
bupa	65.7 ± 5.6	65.9 ± 5.4	64.5 ± 5.7	66.3 ± 5.9
car	91.9 ± 1.9	91.4 ± 3.0	92.3 ± 1.2	91.2 ± 2.3
cmc	53.4 ± 3.6	54.2 ± 3.4	53.7 ± 3.3	53.7 ± 3.7
crx	86.4 ± 2.4	86.4 ± 2.9	86.2 ± 2.9	86.4 ± 2.5
derm	96.4 ± 2.2	96.4 ± 2.3	96.2 ± 2.0	96.3 ± 2.1
glass	69.2 ± 6.8	69.6 ± 8.0	70.4 ± 6.8	69.5 ± 7.7
haber	73.1 ± 5.5	72.9 ± 4.4	72.2 ± 5.1	73.3 ± 5.0
heart	81.5 ± 4.7	80.7 ± 5.0	80.9 ± 4.6	81.4 ± 4.7
ion	92.1 ± 2.9	91.7 ± 3.4	91.6 ± 4.1	92.3 ± 3.2
iris	93.4 ± 4.8	93.2 ± 4.0	93.3 ± 4.4	94.0 ± 4.1
newthy	93.1 ± 4.2	92.9 ± 3.7	93.4 ± 3.7	93.0 ± 4.2
pima	74.8 ± 2.8	74.5 ± 3.6	74.5 ± 3.6	74.0 ± 3.4
solar	73.6 ± 2.9	73.5 ± 2.1	73.3 ± 2.7	73.7 ± 2.6
sonar	73.3 ± 6.7	74.5 ± 6.1	72.3 ± 7.0	73.8 ± 7.4
tic-tac-toe	88.5 ± 2.9	87.3 ± 4.8	88.4 ± 3.3	87.9 ± 3.9
vote	97.0 ± 2.1	97.0 ± 2.2	97.0 ± 2.4	97.0 ± 2.0
wdbc	94.8 ± 1.8	94.6 ± 2.1	95.0 ± 1.6	95.1 ± 1.7
wine	94.8 ± 3.8	94.2 ± 4.9	94.9 ± 3.3	95.1 ± 4.2
wpbc	76.2 ± 6.4	75.9 ± 6.3	76.1 ± 6.3	76.3 ± 6.2

Table 5.17: Classification accuracy of the baseline *TM-GA*$_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25) and C4.5.

Data Set	***TM-GA*$_0$**	**Bagging**	**AdaBoost**	**RF**	**C4.5**
balance	81.7 ± 3.4	81.3 ± 3.9	82.8 ± 3.1	82.0 ± 2.5	77.6 ± 3.6 ☒
breast-c	72.4 ± 5.7	68.8 ± 7.0 ☒	68.7 ± 5.3 ☒	70.7 ± 4.3 ☒	70.0 ± 5.8 ☒
breast-w	96.2 ± 1.6	96.0 ± 1.6	96.6 ± 1.4	96.8 ± 1.4 •	94.4 ± 2.1 ☒
bupa	65.7 ± 5.6	67.4 ± 5.2	65.3 ± 4.4	65.9 ± 5.0	61.1 ± 5.6 ☒
car	91.9 ± 1.9	91.5 ± 1.5	92.8 ± 1.5 •	90.1 ± 1.6 ☒	87.3 ± 2.2 ☒
cmc	53.4 ± 3.6	52.3 ± 2.9	51.3 ± 3.0 ☒	50.1 ± 2.7 ☒	50.5 ± 2.7 ☒
crx	86.4 ± 2.4	85.7 ± 2.6	85.7 ± 2.8	84.7 ± 3.0 ☒	85.9 ± 2.4
derm	96.4 ± 2.2	95.8 ± 2.3	96.2 ± 2.6	95.6 ± 2.2 ☒	92.5 ± 4.1 ☒
glass	69.2 ± 6.8	67.8 ± 6.4	69.5 ± 6.3	70.5 ± 7.4	63.3 ± 6.8 ☒
haber	73.1 ± 5.5	71.7 ± 4.5	69.4 ± 5.2 ☒	71.5 ± 5.4	71.0 ± 6.4 ☒
heart	81.5 ± 4.7	79.6 ± 5.4 ☒	80.0 ± 4.8	80.7 ± 5.8	76.3 ± 6.1 ☒
ion	92.1 ± 2.9	91.7 ± 3.3	92.3 ± 3.7	92.3 ± 2.5	88.7 ± 4.1 ☒
iris	93.4 ± 4.8	93.5 ± 4.5	94.1 ± 3.9	94.4 ± 3.1	93.9 ± 3.9
newthy	93.1 ± 4.2	93.7 ± 3.5	93.8 ± 2.9	94.8 ± 3.6 •	90.8 ± 4.6 ☒
pima	74.8 ± 2.8	74.2 ± 3.6	74.3 ± 3.0	74.8 ± 3.4	72.4 ± 3.8 ☒
solar	73.6 ± 2.9	72.4 ± 2.2 ☒	72.3 ± 2.5 ☒	73.1 ± 1.6	70.3 ± 2.6 ☒
sonar	73.3 ± 6.7	74.6 ± 5.4	75.4 ± 6.4	75.0 ± 6.4	68.0 ± 7.2 ☒
tic-tac-toe	88.5 ± 2.9	87.6 ± 2.6	82.5 ± 3.0 ☒	77.4 ± 3.3 ☒	81.3 ± 3.6 ☒
vote	97.0 ± 2.1	96.0 ± 2.5 ☒	96.3 ± 2.4	95.1 ± 3.2 ☒	96.8 ± 2.5
wdbc	94.8 ± 1.8	95.0 ± 1.9	95.8 ± 2.1 •	95.4 ± 1.8	93.3 ± 2.7 ☒
wine	94.8 ± 3.8	95.4 ± 2.9	95.5 ± 3.8	97.4 ± 2.3 •	89.2 ± 7.0 ☒
wpbc	76.2 ± 6.4	74.6 ± 7.2	73.6 ± 7.2 ☒	76.3 ± 6.6	74.6 ± 6.7

Table 5.18: Classification accuracy of the baseline *TM-GA*$_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 50).

Data Set	*TM-GA*$_0$	Bagging	AdaBoost	RF
balance	81.7 ± 3.4	81.7 ± 2.9	84.2 ± 3.7 •	83.0 ± 3.5 •
breast-c	72.4 ± 5.7	71.7 ± 5.7	69.1 ± 4.7 ⊠	71.3 ± 5.1
breast-w	96.2 ± 1.6	96.3 ± 1.5	96.5 ± 1.5	97.0 ± 1.2 •
bupa	65.7 ± 5.6	67.1 ± 6.3	67.2 ± 5.5	64.8 ± 5.4
car	91.9 ± 1.9	91.8 ± 1.5	93.6 ± 1.3 •	90.9 ± 1.6 ⊠
cmc	53.4 ± 3.6	52.6 ± 3.0	51.4 ± 2.7 ⊠	51.0 ± 2.8 ⊠
crx	86.4 ± 2.4	85.9 ± 3.1	86.0 ± 3.3	85.4 ± 2.7 ⊠
derm	96.4 ± 2.2	95.4 ± 2.7 ⊠	96.6 ± 2.0	96.6 ± 2.0
glass	69.2 ± 6.8	67.9 ± 6.5	68.4 ± 6.6	71.5 ± 7.3
haber	73.1 ± 5.5	70.2 ± 4.3 ⊠	70.3 ± 5.5 ⊠	72.4 ± 6.0
heart	81.5 ± 4.7	79.9 ± 4.8 ⊠	80.5 ± 5.5	81.6 ± 4.7
ion	92.1 ± 2.9	91.9 ± 3.4	92.6 ± 2.7	92.9 ± 3.0
iris	93.4 ± 4.8	94.6 ± 3.7	94.0 ± 4.8	94.5 ± 3.5
newthy	93.1 ± 4.2	93.9 ± 3.4	94.3 ± 3.3	94.9 ± 3.7 •
pima	74.8 ± 2.8	74.5 ± 2.7	74.9 ± 3.1	75.7 ± 3.4
solar	73.6 ± 2.9	73.3 ± 2.1	72.4 ± 2.6 ⊠	73.5 ± 2.0
sonar	73.3 ± 6.7	75.4 ± 6.3	77.0 ± 8.3 •	78.1 ± 6.2 •
tic-tac-toe	88.5 ± 2.9	88.4 ± 2.7	83.3 ± 2.6 ⊠	78.9 ± 3.4 ⊠
vote	97.0 ± 2.1	96.3 ± 2.5	95.9 ± 2.2 ⊠	95.0 ± 3.1 ⊠
wdbc	94.8 ± 1.8	95.2 ± 2.2	95.8 ± 2.1 •	95.4 ± 2.0
wine	94.8 ± 3.8	94.5 ± 3.8	95.5 ± 4.0	97.9 ± 2.2 •
wpbc	76.2 ± 6.4	75.5 ± 6.3	75.4 ± 6.3	76.3 ± 5.6

Table 5.19: Percentage of original attributes retained by the baseline $TM\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$\boldsymbol{TM\text{-}GA_0}$	$\boldsymbol{TM\text{-}GA_2}$	$\boldsymbol{TM\text{-}GA_4}$	$\boldsymbol{TM\text{-}GA_5}$
balance	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
breast-c	29.6 ± 10.7	30.2 ± 10.5	33.1 ± 7.6 ⊠	29.8 ± 11.1
breast-w	50.0 ± 11.9	49.3 ± 12.3	49.1 ± 12.9	53.1 ± 11.1
bupa	69.3 ± 17.0	68.0 ± 20.4	73.0 ± 19.0	73.0 ± 17.8
car	92.7 ± 9.0	92.3 ± 9.0	91.7 ± 8.4	93.7 ± 8.8
cmc	44.2 ± 18.3	43.6 ± 13.8	44.7 ± 15.0	49.3 ± 18.4
crx	7.1 ± 1.6	6.7 ± 0.0 •	8.4 ± 6.1	6.8 ± 0.9
derm	30.1 ± 3.4	29.3 ± 3.9	29.7 ± 4.3	29.6 ± 4.5
glass	75.8 ± 11.4	73.6 ± 15.2	72.7 ± 11.9	76.4 ± 14.1
haber	28.0 ± 31.8	30.0 ± 33.2	24.0 ± 29.4	23.3 ± 30.3
heart	39.5 ± 7.9	41.2 ± 8.0	38.3 ± 7.7	41.1 ± 8.7
ion	19.9 ± 4.5	19.1 ± 5.0	18.4 ± 4.7	19.2 ± 4.4
iris	40.0 ± 13.4	38.5 ± 13.6	40.0 ± 13.4	41.5 ± 12.0
newthy	46.8 ± 11.9	44.0 ± 13.4	46.8 ± 13.2	46.8 ± 13.2
pima	45.0 ± 15.2	42.3 ± 17.3	46.5 ± 17.5	46.5 ± 20.2
solar	34.8 ± 10.2	31.7 ± 9.7	36.5 ± 14.1	32.8 ± 8.8
sonar	16.5 ± 7.1	11.9 ± 7.1 •	14.3 ± 7.3	13.3 ± 7.1 •
tic-tac-toe	99.3 ± 4.7	96.9 ± 9.8	99.3 ± 4.7	98.9 ± 6.4
vote	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0	6.3 ± 0.0
wdbc	18.7 ± 4.8	17.3 ± 5.0	18.0 ± 4.6	18.3 ± 5.1
wine	29.4 ± 6.2	28.2 ± 4.8	27.8 ± 4.9	27.7 ± 4.9
wpbc	0.0 ± 0.0	0.2 ± 1.3	0.2 ± 1.3	0.2 ± 1.7

Table 5.20: Percentage of "artificial" (irrelevant) attributes retained by the baseline $TM\text{-}GA_0$ and *IM-GA* mating strategies.

Data Set	$\boldsymbol{TM\text{-}GA_0}$	$\boldsymbol{TM\text{-}GA_2}$	$\boldsymbol{TM\text{-}GA_4}$	$\boldsymbol{TM\text{-}GA_5}$
balance	2.5 ± 5.6	3.3 ± 6.1	1.0 ± 3.4	4.3 ± 6.5
breast-c	11.7 ± 5.4	11.6 ± 5.8	13.2 ± 5.9	14.2 ± 5.8 ☒
breast-w	1.1 ± 2.5	1.4 ± 2.7	1.6 ± 3.0	1.1 ± 2.2
bupa	14.3 ± 15.4	17.5 ± 17.0	17.7 ± 20.7	18.8 ± 18.8
car	37.5 ± 17.6	30.7 ± 19.4 •	33.3 ± 20.7	42.5 ± 16.6
cmc	15.2 ± 26.9	15.7 ± 29.0	19.2 ± 29.9	22.7 ± 33.7
crx	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.9 ☒	0.0 ± 0.0
derm	0.5 ± 1.0	0.4 ± 0.9	0.3 ± 1.0	0.5 ± 1.1
glass	10.2 ± 7.1	8.2 ± 7.0	8.6 ± 6.7	12.1 ± 8.0
haber	11.3 ± 14.1	12.3 ± 15.7	12.0 ± 16.5	11.0 ± 16.4
heart	3.9 ± 3.0	4.0 ± 3.3	3.7 ± 3.5	3.8 ± 3.2
ion	0.7 ± 1.1	0.9 ± 1.4	0.8 ± 1.3	0.8 ± 1.2
iris	1.3 ± 3.8	1.0 ± 3.4	1.0 ± 4.3	1.0 ± 3.4
newthy	2.4 ± 4.3	3.0 ± 4.6	2.6 ± 4.9	4.0 ± 5.7
pima	9.1 ± 11.2	8.0 ± 11.4	9.9 ± 16.3	13.3 ± 18.9
solar	16.8 ± 26.5	13.0 ± 20.1	23.0 ± 33.6	12.3 ± 18.4
sonar	1.6 ± 1.4	0.9 ± 1.2 •	1.2 ± 1.2	1.1 ± 1.4 •
tic-tac-toe	32.8 ± 15.7	31.0 ± 16.9	35.1 ± 13.7	36.2 ± 13.6
vote	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
wdbc	0.7 ± 1.1	0.5 ± 1.2	0.5 ± 1.0	0.8 ± 1.3
wine	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
wpbc	0.0 ± 0.0	0.0 ± 0.2	0.1 ± 0.6	0.2 ± 1.3

Table 5.21: Percentage of original attributes retained by the baseline *TM-GA*$_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25).

Data Set	***TM-GA***$_0$	**Bagging**	**AdaBoost**	**RF**
balance	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
breast-c	29.6 ± 10.7	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
breast-w	50.0 ± 11.9	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
bupa	69.3 ± 17.0	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
car	92.7 ± 9.0	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
cmc	44.2 ± 18.3	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
crx	7.1 ± 1.6	93.3 ± 0.0 ☒	93.3 ± 0.0 ☒	100.0 ± 0.0 ☒
derm	30.1 ± 3.4	72.5 ± 7.2 ☒	92.5 ± 7.9 ☒	100.0 ± 0.0 ☒
glass	75.8 ± 11.4	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
haber	28.0 ± 31.8	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
heart	39.5 ± 7.9	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	99.8 ± 1.1 ☒
ion	19.9 ± 4.5	83.1 ± 6.2 ☒	82.3 ± 16.4 ☒	96.9 ± 0.6 ☒
iris	40.0 ± 13.4	94.0 ± 12.9 ☒	93.5 ± 14.1 ☒	100.0 ± 0.0 ☒
newthy	46.8 ± 11.9	100.0 ± 0.0 ☒	99.2 ± 4.0 ☒	100.0 ± 0.0 ☒
pima	45.0 ± 15.2	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
solar	34.8 ± 10.2	91.7 ± 0.0 ☒	100.0 ± 0.0 ☒	99.8 ± 1.2 ☒
sonar	16.5 ± 7.1	80.8 ± 5.1 ☒	73.7 ± 10.4 ☒	97.8 ± 1.6 ☒
tic-tac-toe	99.3 ± 4.7	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
vote	6.3 ± 0.0	43.6 ± 10.2 ☒	68.3 ± 23.4 ☒	98.5 ± 2.7 ☒
wdbc	18.7 ± 4.8	86.1 ± 6.0 ☒	85.0 ± 13.8 ☒	100.0 ± 0.0 ☒
wine	29.4 ± 6.2	83.2 ± 8.3 ☒	77.8 ± 21.8 ☒	100.0 ± 0.0 ☒
wpbc	0.0 ± 0.0	79.2 ± 6.5 ☒	67.6 ± 17.7 ☒	99.5 ± 1.1 ☒

Table 5.22: Percentage of "artificial" (irrelevant) attributes retained by the baseline *TM-GA*$_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25).

Data Set	***TM-GA*$_0$**	**Bagging**	**AdaBoost**	**RF**
balance	2.5 ± 5.6	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
breast-c	11.7 ± 5.4	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
breast-w	1.1 ± 2.5	92.3 ± 7.0 ☒	99.8 ± 1.1 ☒	100.0 ± 0.0 ☒
bupa	14.3 ± 15.4	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
car	37.5 ± 17.6	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
cmc	15.2 ± 26.9	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
crx	0.0 ± 0.0	99.7 ± 0.9 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
derm	0.5 ± 1.0	20.1 ± 5.2 ☒	53.4 ± 11.5 ☒	99.5 ± 0.8 ☒
glass	10.2 ± 7.1	99.7 ± 1.3 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
haber	11.3 ± 14.1	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
heart	3.9 ± 3.0	98.4 ± 2.3 ☒	99.9 ± 0.5 ☒	100.0 ± 0.0 ☒
ion	0.7 ± 1.1	54.9 ± 6.8 ☒	73.8 ± 18.0 ☒	95.3 ± 2.8 ☒
iris	1.3 ± 3.8	49.3 ± 23.5 ☒	74.5 ± 33.1 ☒	99.8 ± 1.8 ☒
newthy	2.4 ± 4.3	48.0 ± 17.6 ☒	82.0 ± 24.1 ☒	99.6 ± 2.0 ☒
pima	9.1 ± 11.2	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
solar	16.8 ± 26.5	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
sonar	1.6 ± 1.4	43.4 ± 4.7 ☒	51.6 ± 7.8 ☒	85.1 ± 3.4 ☒
tic-tac-toe	32.8 ± 15.7	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒	100.0 ± 0.0 ☒
vote	0.0 ± 0.0	37.1 ± 9.9 ☒	65.5 ± 26.9 ☒	100.0 ± 0.0 ☒
wdbc	0.7 ± 1.1	43.1 ± 7.1 ☒	71.3 ± 15.0 ☒	92.2 ± 4.0 ☒
wine	0.0 ± 0.0	6.8 ± 4.9 ☒	26.2 ± 16.3 ☒	88.7 ± 6.4 ☒
wpbc	0.0 ± 0.0	80.1 ± 4.5 ☒	78.1 ± 19.9 ☒	97.1 ± 1.8 ☒

Table 5.23: Number of decision trees in the decision forests induced by the baseline $TM\text{-}GA_0$ versus the *IM-GA* mating strategies.

Data Set	$TM\text{-}GA_0$	$TM\text{-}GA_2$	$TM\text{-}GA_4$	$TM\text{-}GA_5$
balance	3.0 ± 0.1	3.0 ± 0.1	3.0 ± 0.1	3.2 ± 0.5
breast-c	3.0 ± 0.2	3.2 ± 0.8	3.1 ± 0.3	3.1 ± 0.6
breast-w	3.0 ± 0.0	3.0 ± 0.2	3.1 ± 0.2	3.0 ± 0.1
bupa	3.0 ± 0.2	3.0 ± 0.2	3.0 ± 0.1	3.0 ± 0.1
car	3.0 ± 0.0	3.0 ± 0.2	3.0 ± 0.0	3.1 ± 0.3
cmc	3.2 ± 0.5	3.1 ± 0.4	3.1 ± 0.3	3.2 ± 0.4
crx	3.9 ± 1.0	3.7 ± 0.8	4.0 ± 1.1	3.7 ± 0.9
derm	3.1 ± 0.2	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0
glass	3.1 ± 0.3	3.1 ± 0.3	3.1 ± 0.2	3.1 ± 0.4
haber	5.3 ± 1.9	5.1 ± 2.0	5.3 ± 2.0	5.5 ± 1.9
heart	3.0 ± 0.1	3.0 ± 0.1	3.0 ± 0.1	3.0 ± 0.0
ion	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.1
iris	3.6 ± 0.5	3.5 ± 0.5	3.5 ± 0.5	3.4 ± 0.5
newthy	3.1 ± 0.2	3.1 ± 0.3	3.0 ± 0.1	3.0 ± 0.2
pima	3.1 ± 0.2	3.1 ± 0.3	3.1 ± 0.3	3.0 ± 0.2
solar	3.0 ± 0.2	3.2 ± 0.4	3.2 ± 0.4	3.1 ± 0.3
sonar	3.0 ± 0.1	3.0 ± 0.0	3.0 ± 0.1	3.0 ± 0.2
tic-tac-toe	3.0 ± 0.3	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.3
vote	3.3 ± 0.6	3.2 ± 0.4	3.3 ± 0.5	3.3 ± 0.5
wdbc	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.2
wine	3.0 ± 0.1	3.0 ± 0.1	3.0 ± 0.2	3.0 ± 0.0
wpbc	7.0 ± 0.0	6.9 ± 0.7	6.9 ± 0.7	6.9 ± 0.6

Table 5.24: Total nodes in decision forest for the baseline *TM-GA*$_0$ versus the *IM-GA* mating strategies.

Data Set	***TM-GA***$_0$	***TM-GA***$_2$	***TM-GA***$_4$	***TM-GA***$_5$
balance	88.4 ± 41.5	82.4 ± 45.9	76.7 ± 41.2	87.8 ± 40.1
breast-c	27.0 ± 10.3	26.5 ± 10.3	27.1 ± 7.7	25.7 ± 10.0
breast-w	21.7 ± 3.5	22.6 ± 5.6	22.1 ± 4.3	21.5 ± 4.3
bupa	59.8 ± 39.6	68.9 ± 48.8	67.8 ± 47.1	68.4 ± 46.8
car	498.2 ± 65.7	481.6 ± 108.3	505.3 ± 30.5	493.8 ± 108.2
cmc	252.5 ± 409.6	239.4 ± 409.8	259.2 ± 384.9	308.2 ± 456.7
crx	12.1 ± 3.1	11.2 ± 2.4	13.9 ± 6.8 ⊠	11.3 ± 2.7
derm	79.6 ± 8.1	78.5 ± 3.6	78.7 ± 4.8	77.7 ± 5.3
glass	118.7 ± 17.0	113.7 ± 15.7	116.1 ± 14.2	118.4 ± 23.1
haber	13.9 ± 9.5	14.2 ± 9.1	14.3 ± 11.2	12.9 ± 9.2
heart	19.0 ± 6.5	22.1 ± 11.9	19.1 ± 6.7	20.4 ± 7.8
ion	35.3 ± 8.6	35.9 ± 9.5	33.4 ± 7.7	34.6 ± 8.2
iris	21.4 ± 3.6	20.7 ± 3.8	20.1 ± 3.2 •	20.3 ± 3.1 •
newthy	26.5 ± 4.2	25.4 ± 4.6	25.0 ± 4.5 •	26.4 ± 5.2
pima	38.3 ± 35.5	35.2 ± 39.0	42.8 ± 48.8	46.8 ± 57.9
solar	144.6 ± 181.5	124.2 ± 135.8	201.6 ± 250.8	111.5 ± 107.2
sonar	42.0 ± 21.8	28.0 ± 20.2 •	34.8 ± 21.4	32.3 ± 20.7 •
tic-tac-toe	424.0 ± 67.8	387.8 ± 119.8 •	425.2 ± 63.8	426.5 ± 74.7
vote	10.0 ± 1.7	9.5 ± 1.1 •	9.9 ± 1.6	9.9 ± 1.6
wdbc	33.4 ± 5.4	29.0 ± 7.8 •	29.8 ± 6.7 •	31.9 ± 9.5
wine	23.5 ± 3.0	23.5 ± 3.1	23.8 ± 3.7	24.6 ± 2.8 ⊠
wpbc	7.0 ± 0.0	7.2 ± 1.1	7.3 ± 1.7	7.4 ± 2.8

Table 5.25: Total leaves in decision forest for the baseline *TM-GA*$_0$ versus the *IM-GA* mating strategies.

Data Set	***TM-GA***$_0$	***TM-GA***$_2$	***TM-GA***$_4$	***TM-GA***$_5$
balance	45.7 ± 20.7	42.7 ± 23.0	39.8 ± 20.6	45.5 ± 20.2
breast-c	17.9 ± 7.5	17.1 ± 6.9	18.2 ± 5.5	16.6 ± 6.1
breast-w	12.3 ± 1.8	12.8 ± 2.9	12.6 ± 2.2	12.3 ± 2.1
bupa	31.4 ± 19.8	36.0 ± 24.4	35.4 ± 23.6	35.7 ± 23.4
car	351.0 ± 46.7	341.3 ± 76.8	356.4 ± 22.1	346.2 ± 76.0
cmc	139.1 ± 218.1	131.4 ± 214.3	141.9 ± 203.1	168.2 ± 239.9
crx	8.1 ± 2.1	7.5 ± 1.6	9.3 ± 4.6 ⊠	7.5 ± 1.8
derm	60.2 ± 6.1	59.4 ± 2.7	59.6 ± 3.5	58.8 ± 3.9
glass	60.9 ± 8.6	58.4 ± 7.9	59.6 ± 7.2	60.8 ± 11.7
haber	9.6 ± 4.1	9.7 ± 3.9	9.8 ± 4.9	9.3 ± 4.1
heart	11.0 ± 3.2	12.6 ± 6.0	11.0 ± 3.3	11.7 ± 3.9
ion	19.2 ± 4.3	19.4 ± 4.8	18.2 ± 3.9	18.8 ± 4.1
iris	12.5 ± 1.9	12.1 ± 2.0	11.8 ± 1.7 •	11.9 ± 1.7 •
newthy	14.8 ± 2.2	14.3 ± 2.3	14.0 ± 2.2 •	14.7 ± 2.6
pima	20.7 ± 17.7	19.1 ± 19.5	23.0 ± 24.4	24.9 ± 29.0
solar	91.8 ± 94.6	81.4 ± 71.4	121.5 ± 131.8	74.4 ± 55.8
sonar	22.5 ± 10.9	15.5 ± 10.1 •	18.9 ± 10.7	17.7 ± 10.4 •
tic-tac-toe	277.5 ± 44.0	253.9 ± 78.0 •	277.4 ± 41.1	278.5 ± 48.8
vote	6.6 ± 1.1	6.3 ± 0.7 •	6.6 ± 1.1	6.6 ± 1.1
wdbc	18.2 ± 2.7	16.0 ± 3.9 •	16.4 ± 3.3 •	17.5 ± 4.8
wine	13.3 ± 1.5	13.2 ± 1.6	13.4 ± 1.9	13.8 ± 1.4 ⊠
wpbc	7.0 ± 0.0	7.0 ± 0.3	7.1 ± 0.6	7.2 ± 1.1

Table 5.26: Total nodes in decision forest for the baseline $TM\text{-}GA_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25).

Data Set	$TM\text{-}GA_0$	Bagging	AdaBoost	RF
balance	88.4 ± 41.5	2318.6 ± 80.2 ☒	2491.8 ± 66.6 ☒	4241.3 ± 85.6 ☒
breast-c	27.0 ± 10.3	2075.8 ± 108.7 ☒	2743.5 ± 124.8 ☒	3199.2 ± 117.6 ☒
breast-w	21.7 ± 3.5	607.7 ± 43.9 ☒	654.6 ± 36.7 ☒	1140.7 ± 56.6 ☒
bupa	59.8 ± 39.6	1417.6 ± 40.5 ☒	1098.2 ± 30.8 ☒	2329.1 ± 42.0 ☒
car	498.2 ± 65.7	4608.5 ± 82.9 ☒	6670.5 ± 137.0 ☒	10277.0 ± 284.4 ☒
cmc	252.5 ± 409.6	9283.7 ± 107.6 ☒	10471.0 ± 103.7 ☒	15885.2 ± 135.8 ☒
crx	12.1 ± 3.1	2183.0 ± 137.3 ☒	4010.2 ± 155.5 ☒	4374.2 ± 186.4 ☒
derm	79.6 ± 8.1	971.6 ± 49.6 ☒	1124.8 ± 248.7 ☒	2976.6 ± 133.1 ☒
glass	118.7 ± 17.0	1004.4 ± 36.7 ☒	988.4 ± 34.9 ☒	1910.3 ± 55.2 ☒
haber	13.9 ± 9.5	1547.8 ± 65.8 ☒	1576.8 ± 88.9 ☒	2575.8 ± 95.1 ☒
heart	19.0 ± 6.5	874.0 ± 40.8 ☒	805.4 ± 32.0 ☒	1507.5 ± 47.8 ☒
ion	35.3 ± 8.6	541.3 ± 29.7 ☒	505.4 ± 124.8 ☒	1174.3 ± 46.3 ☒
iris	21.4 ± 3.6	204.1 ± 21.7 ☒	186.4 ± 101.1 ☒	456.4 ± 38.5 ☒
newthy	26.5 ± 4.2	321.0 ± 19.3 ☒	271.0 ± 84.5 ☒	534.1 ± 39.6 ☒
pima	38.3 ± 35.5	2356.1 ± 49.8 ☒	2012.9 ± 45.9 ☒	4208.8 ± 74.6 ☒
solar	144.6 ± 181.5	5987.0 ± 129.9 ☒	7904.0 ± 172.5 ☒	12292.0 ± 386.0 ☒
sonar	42.0 ± 21.8	544.4 ± 18.5 ☒	448.4 ± 62.2 ☒	1094.8 ± 24.0 ☒
tic-tac-toe	424.0 ± 67.8	3736.2 ± 59.7 ☒	4674.3 ± 99.1 ☒	6886.0 ± 131.6 ☒
vote	10.0 ± 1.7	209.1 ± 24.6 ☒	243.6 ± 109.9 ☒	809.3 ± 69.3 ☒
wdbc	33.4 ± 5.4	483.5 ± 26.4 ☒	486.5 ± 98.9 ☒	1049.2 ± 53.7 ☒
wine	23.5 ± 3.0	245.0 ± 15.4 ☒	186.9 ± 89.6 ☒	590.0 ± 29.3 ☒
wpbc	7.0 ± 0.0	531.7 ± 25.3 ☒	423.8 ± 120.0 ☒	989.0 ± 47.3 ☒

Table 5.27: Total leaves in decision forest for the baseline $TM\text{-}GA_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25).

Data Set	$TM\text{-}GA_0$	Bagging	AdaBoost	RF
balance	45.7 ± 20.7	1171.8 ± 40.1 ☒	1258.4 ± 33.3 ☒	2133.2 ± 42.8 ☒
breast-c	17.9 ± 7.5	1581.2 ± 94.2 ☒	2198.1 ± 113.4 ☒	2328.6 ± 102.3 ☒
breast-w	12.3 ± 1.8	316.4 ± 21.9 ☒	339.8 ± 18.3 ☒	582.8 ± 28.3 ☒
bupa	31.4 ± 19.8	721.3 ± 20.3 ☒	561.6 ± 15.4 ☒	1177.1 ± 21.0 ☒
car	351.0 ± 46.7	3118.1 ± 59.0 ☒	4364.4 ± 86.2 ☒	6369.1 ± 172.4 ☒
cmc	139.1 ± 218.1	5253.0 ± 71.8 ☒	6277.9 ± 81.6 ☒	8660.6 ± 90.7 ☒
crx	8.1 ± 2.1	1553.6 ± 114.1 ☒	3168.1 ± 140.7 ☒	2813.0 ± 139.1 ☒
derm	60.2 ± 6.1	716.2 ± 36.9 ☒	813.5 ± 179.2 ☒	1992.8 ± 92.7 ☒
glass	60.9 ± 8.6	514.7 ± 18.3 ☒	506.7 ± 17.5 ☒	967.7 ± 27.6 ☒
haber	9.6 ± 4.1	1062.9 ± 56.6 ☒	1071.9 ± 95.0 ☒	1604.6 ± 82.3 ☒
heart	11.0 ± 3.2	449.5 ± 20.4 ☒	415.2 ± 16.0 ☒	766.2 ± 23.9 ☒
ion	19.2 ± 4.3	283.2 ± 14.9 ☒	264.5 ± 65.2 ☒	599.6 ± 23.2 ☒
iris	12.5 ± 1.9	114.6 ± 10.9 ☒	102.1 ± 55.0 ☒	240.7 ± 19.3 ☒
newthy	14.8 ± 2.2	173.0 ± 9.7 ☒	146.8 ± 45.7 ☒	279.5 ± 19.8 ☒
pima	20.7 ± 17.7	1190.5 ± 24.9 ☒	1019.0 ± 22.9 ☒	2116.9 ± 37.3 ☒
solar	91.8 ± 94.6	3715.7 ± 100.6 ☒	5101.4 ± 139.6 ☒	7300.2 ± 254.9 ☒
sonar	22.5 ± 10.9	284.7 ± 9.2 ☒	236.5 ± 32.8 ☒	559.9 ± 12.0 ☒
tic-tac-toe	277.5 ± 44.0	2294.5 ± 38.5 ☒	2885.1 ± 60.9 ☒	3908.3 ± 77.5 ☒
vote	6.6 ± 1.1	117.1 ± 12.3 ☒	131.9 ± 59.2 ☒	417.1 ± 34.6 ☒
wdbc	18.2 ± 2.7	254.3 ± 13.2 ☒	255.3 ± 51.8 ☒	537.1 ± 26.9 ☒
wine	13.3 ± 1.5	135.0 ± 7.7 ☒	103.3 ± 49.4 ☒	307.5 ± 14.7 ☒
wpbc	7.0 ± 0.0	278.3 ± 12.7 ☒	223.4 ± 63.2 ☒	507.0 ± 23.7 ☒

Table 5.28: Average number of tests required to classify an "unknown example" for the baseline *TM-GA*$_0$ versus the *IM-GA* mating strategies.

Data Set	***TM-GA*$_0$**	***TM-GA*$_2$**	***TM-GA*$_4$**	***TM-GA*$_5$**
balance	11.0 ± 2.4	10.8 ± 2.6	10.5 ± 2.6	11.6 ± 2.9
breast-c	4.2 ± 1.0	4.3 ± 1.5	4.6 ± 0.9 ⊠	4.5 ± 1.2
breast-w	5.5 ± 0.6	5.7 ± 1.0	5.6 ± 0.7	5.5 ± 0.6
bupa	9.3 ± 3.4	10.0 ± 3.6	9.7 ± 3.7	9.8 ± 3.5
car	8.2 ± 0.4	8.2 ± 0.7	8.2 ± 0.2	8.3 ± 1.0
cmc	13.6 ± 7.2	12.8 ± 7.3	13.2 ± 5.9	13.9 ± 7.4
crx	4.0 ± 1.0	3.7 ± 0.8	4.2 ± 1.2	3.7 ± 0.9
derm	11.2 ± 1.1	11.0 ± 0.7	10.9 ± 0.7	10.9 ± 0.6
glass	14.6 ± 1.8	14.5 ± 1.7	14.4 ± 1.5	14.9 ± 2.3
haber	2.1 ± 2.5	2.1 ± 2.4	2.1 ± 2.5	1.8 ± 2.3
heart	4.8 ± 0.9	5.2 ± 1.6 ⊠	4.8 ± 0.9	5.0 ± 1.1
ion	10.5 ± 1.6	10.5 ± 1.9	10.1 ± 1.5	10.4 ± 1.6
iris	6.6 ± 0.9	6.4 ± 0.9	6.3 ± 0.8 •	6.3 ± 0.8 •
newthy	7.2 ± 0.9	7.0 ± 0.9	6.8 ± 0.7 •	7.0 ± 0.7
pima	6.6 ± 2.6	6.3 ± 2.7	6.6 ± 3.0	6.9 ± 3.1
solar	6.5 ± 2.2	6.6 ± 1.6	7.6 ± 3.6 ⊠	6.3 ± 1.6
sonar	8.0 ± 2.8	6.2 ± 2.7 •	7.2 ± 2.8	6.8 ± 2.7 •
tic-tac-toe	12.3 ± 1.6	11.6 ± 2.2 •	12.2 ± 1.1	12.3 ± 1.5
vote	3.3 ± 0.6	3.2 ± 0.4 •	3.3 ± 0.5	3.3 ± 0.5
wdbc	6.3 ± 0.5	6.0 ± 0.8 •	6.0 ± 0.7 •	6.3 ± 1.0
wine	6.6 ± 0.7	6.5 ± 0.6	6.7 ± 0.8	6.7 ± 0.5
wpbc	0.0 ± 0.0	0.1 ± 0.6	0.1 ± 0.6	0.1 ± 0.7

Table 5.29: Average number of tests required to classify an "unknown example" for the baseline *TM-GA*$_0$ versus the non-GA-based techniques of Bagging, AdaBoost, and Random Forests (ensemble size 25).

Data Set	***TM-GA*$_0$**	**Bagging**	**AdaBoost**	**RF**
balance	11.0 ± 2.4	676.5 ± 13.3 ☒	882.1 ± 20.5 ☒	839.7 ± 11.2 ☒
breast-c	4.2 ± 1.0	504.6 ± 26.6 ☒	643.9 ± 19.9 ☒	563.6 ± 21.2 ☒
breast-w	5.5 ± 0.6	372.2 ± 15.2 ☒	517.0 ± 28.6 ☒	516.0 ± 16.5 ☒
bupa	9.3 ± 3.4	759.7 ± 34.2 ☒	735.4 ± 24.2 ☒	891.1 ± 32.4 ☒
car	8.2 ± 0.4	359.0 ± 1.5 ☒	562.6 ± 18.2 ☒	546.1 ± 13.9 ☒
cmc	13.6 ± 7.2	1204.4 ± 17.7 ☒	1390.5 ± 26.9 ☒	1129.5 ± 17.8 ☒
crx	4.0 ± 1.0	546.9 ± 17.8 ☒	829.9 ± 21.8 ☒	681.1 ± 26.1 ☒
derm	11.2 ± 1.1	515.7 ± 11.3 ☒	510.1 ± 113.8 ☒	572.1 ± 10.6 ☒
glass	14.6 ± 1.8	730.3 ± 26.1 ☒	750.3 ± 30.9 ☒	767.5 ± 16.8 ☒
haber	2.1 ± 2.5	370.1 ± 15.6 ☒	398.3 ± 33.9 ☒	489.5 ± 24.6 ☒
heart	4.8 ± 0.9	542.5 ± 19.2 ☒	634.4 ± 22.3 ☒	665.1 ± 15.3 ☒
ion	10.5 ± 1.6	583.6 ± 26.8 ☒	547.9 ± 135.4 ☒	737.0 ± 19.1 ☒
iris	6.6 ± 0.9	257.6 ± 11.1 ☒	219.8 ± 124.2 ☒	384.1 ± 16.4 ☒
newthy	7.2 ± 0.9	419.1 ± 22.5 ☒	345.4 ± 111.0 ☒	489.9 ± 23.4 ☒
pima	6.6 ± 2.6	773.4 ± 18.2 ☒	935.1 ± 24.2 ☒	949.8 ± 16.5 ☒
solar	6.5 ± 2.2	601.2 ± 7.5 ☒	971.1 ± 27.2 ☒	849.0 ± 26.4 ☒
sonar	8.0 ± 2.8	520.8 ± 22.7 ☒	493.7 ± 70.8 ☒	631.8 ± 13.4 ☒
tic-tac-toe	12.3 ± 1.6	598.8 ± 15.4 ☒	972.0 ± 36.8 ☒	875.9 ± 31.8 ☒
vote	3.3 ± 0.6	214.2 ± 15.3 ☒	265.9 ± 124.2 ☒	481.3 ± 22.7 ☒
wdbc	6.3 ± 0.5	381.2 ± 16.2 ☒	446.0 ± 90.1 ☒	515.4 ± 17.6 ☒
wine	6.6 ± 0.7	304.8 ± 10.4 ☒	243.8 ± 119.9 ☒	453.1 ± 15.0 ☒
wpbc	0.0 ± 0.0	489.2 ± 27.8 ☒	467.8 ± 135.6 ☒	625.7 ± 25.0 ☒

CHAPTER 6

Conclusion and Ideas for Future Research

This work proposed that the search speed of the Genetic Algorithm (GA) could be accelerated, with the quality of its generated solutions improved, by replacing the long-standing conventional mating strategy, which is based on random selection of mating partners, with more sophisticated mating strategies based on the Darwinian evolutionary principle of "opposites-attract".

To this end, a total of five (5) novel "instinct-based" mating strategies were proposed, four of them as single-population GA strategies and one as multi-population GA strategy. The proposed strategies improved over the conventional mating strategy, where specimens have no mating choice, by "endowing" specimens in the GA population with "mating-instincts" that guide them in their selection of mating partners capable of "complementing" their own behaviors along some problem-dependent optimization criteria.

To test the proposed ideas, two well-known and complex optimization problems from the domain of supervised classification were chosen: 1) the 1-NN Tuning problem, and 2) the Optimal Decision Forests problem. Since mating is only a component of the GA search process, two GA's were designed and implemented to solve the two

testbed problems; one GA was re-implemented from the literature and further improved, the other was entirely designed for the purposes of this work. These two GAs initially relied on the conventional mating strategy and their experimental results served as baseline against which the performance of the proposed mating strategies were verified.

Rigorous cross-validated experiments were run using 24 benchmark data sets from the renowned UCI Machine Learning Repository. In the first round of experiments, the GAs were run with the conventional mating strategy and their results were recorded. Then, the conventional mating strategy was replaced with each of the five (5) novel "instinct-based" mating strategies and the experiments were rerun.

The experimental results confirmed that the GA search speed was indeed accelerated in both testbed problems by the application of the five (5) proposed "instinct-based" mating strategies when compared to the conventional mating strategy. In addition, the experimental results also revealed that the increased search speed promoted by the new mating strategies (1) required only negligible additional computational power and (2) did not come at the cost of lower quality of the GA-generated solutions. In fact, the quality of the GA-generated solutions often improved.

In addition, the experimental results also showed that the performance of the two GAs used in this work, one improved from the literature and the other designed entirely in this work, surpassed that of well-established, GA-based and non-GA-based, classical techniques. The results revealed that the supervised classifiers induced by the two GAs (i.e. the 1-NN and decision forests classifiers) were simultaneously

more accurate and more compact than those attained by the well-established classical techniques.

In particular, the decision forests (i.e. ensembles of C4.5 decision trees) induced by the GA designed entirely in this work, the baseline *TM-GA*$_0$, were both significantly more accurate and compact than those induced by classical ensemble generation techniques. As a consequence, the decision forests induced by *TM-GA*$_0$ were characterized by a significantly improved interpretability of their predictions when compared to those induced by the classical techniques. This is an important result because the with the advent and wide adoption of classical ensemble generation techniques over the past two decades, the "*interpretability* of predictions" became a lost feature. Classical techniques trade-off the size of the decision forests for higher classification accuracy, which severely damages the ""*interpretability* of predictions"". In contrast, *TM-GA*$_0$ was capable of optimizing both the accuracy and size of the decision forests when applied to the Optimal Decision Forests problem (Testbed Problem 2).

The promising results attained in this work inspire further research. Numerous other real-world optimization problems can benefit from an increased GA search speed that is coupled with negligible additional computational requirements. For example, Albert (2002) reports that in some industrial applications of the GA, evaluating the fitness (i.e. quality) of even a single GA-generated solution may require minutes, sometimes even days! Such applications can benefit tremendously from an improved version of the GA that is capable of finding good solutions while requiring significantly less fitness evaluations. The principles of "instinct-based" mating that were proposed in this work, and also verified experimentally in Chapter 5, can pave the way to new

implementations of the GA that are capable of more efficiently and diversely sampling the vast search spaces of solutions found in numerous real-world optimization problems.

An example of such a real-world problem is the Traveling Salesman Problem (TSP) (Moon, Kim, Choi, and Seo 2002), a well-known NP-Hard combinatorial optimization problem. TSP finds several industrial applications in areas such as automated planning, logistics, and computer hardware design. Historically, the GA has been successfully applied in the solving of complex TSP formulations comprised of millions of interconnected nodes. Given this historical importance, investigating the impact of "instinct-based" mating strategies on GA convergence speed when applied to TSP is a promising area of future research; the potential impact encompasses numerous real-world applications.

In addition to extending the principles of "instinct-based" mating to other various real-world problems, another promising area of future research involves extending "instinct-based" mating into other existing GA frameworks, such as the"gendered" GA. As was discussed in Chapter 3, the "gendered" GA is a representative GA framework that has been applied to various optimization problems. It is characterized by the use of multiple fitness-functions that each seek to optimize a different subset of objectives. Thus, inspired by the results presented in this work, a research question that naturally arises is whether "instinct-based" mating strategies can also significantly impact the GA performance when applied to frameworks that adopt multiple fitness-functions into the genetic search process, such as the "gendered" GA framework.

Similarly, the potential impact of this future research topic encompasses numerous real-world applications.

Other promising areas of future research involve a more detailed look into the theoretical implications of the results presented in this work; simultaneous improvements in both GA convergence speed and quality of GA-generated solutions. Recall that Section 1 elaborated on the importance of GA mechanisms that maintain the diversity of the GA population during genetic search and, thus, deter the GA from converging prematurely to suboptimal solutions. In a parallel fashion, the experimental results presented in this work suggest that "instinct-based" mating strategies may be capable of more efficiently maintaining the diversity of the GA population when compared to the conventional mating strategy. Thus, a promising area for future research involves a detailed look into the impact of "instinct-based" mating on the *time-evolution* of the GA population diversity during genetic search. Promising findings may inspire future research on the design of new mechanisms, and improvements of existing ones, that more efficiently deter premature convergence during genetic search.

Finally, statistical modeling of the GA is another promising area of future research with significant theoretical implications. A statistical model of the GA could be used as a tool to judge the potential impact of various "instinct-based" mating strategies on the convergence properties of a particular GA framework. He and Kang (1999, Ming et al. (2006, Lin et al. (2010) have demonstrated that Markov chains can be used as a tool to appropriately model the convergence properties of particular GA frameworks. The task, of course, is rather non-trivial, with every GA component (i.e. initialization, mating, recombination, mutation, and survival) contributing in a

very complex fashion to the Markov Chain's estimates of *state transition probabilities*. Such a novel tool, however, would allow researchers to theoretically determine which combination of GA framework and mating strategy performs best for a particular application. In addition, promising results would emphasize the impact of improved mating strategies on the performance of the GA and encourage further research on this topic.

Bibliography

Albert, L. 2002. Efficient genetic algorithms using discretization scheduling. Tech. rep., University of Illinois at Urbana-Champaign.

Alkhalid, A., Chikalov, I., and Moshkov, M. 2011. Decision tree construction using greedy algorithms and dynamic programming - comparative study. In *CSP2011: Proceedings of the International Workshop on Concurrency, Specification and Programming 2011*. 1–9.

Alpaydin, E. 2004. *Introduction to Machine Learning*. MIT Press.

Angiulli, F. 2005. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 25–32.

Angluin, D. and Smith, C. 1983. Inductive inference: Theory and methods. *Comput. Surveys 15*, 237–269.

Bentley, P. J. 1999. *Evolutionary Design by Computers with CDROM*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Breiman, L. 1996. Bagging predictors. In *Machine Learning*. Kluwer Academic Publishers, 123–140.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. 1984. *Classification and Regression Trees*. Wadsworth and Brooks/Cole Advanced Books and Software, Monterey, CA.

Breiman, L. and Schapire, E. 2001. Random forests. In *Machine Learning*. Kluwer Academic Publishers, 5–32.

Cano, J., Herrera, F., and Lozano, M. 2003. Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *Evolutionary Computation, IEEE Transactions on 7,* 6 (Dec.), 561–575.

Cantu-paz, E. 1997a. Designing efficient master-slave parallel genetic algorithms.

Cantu-paz, E. 1997b. A survey of parallel genetic algorithms. *Calculateurs Paralleles 10*, 141–171.

Cantu-paz, E. 1999. Topologies, migration rates, and multi-population parallel genetic algorithms.

Chandra, A. and Yao, X. Multi-objective ensemble construction, learning and evolution. In *In Proceeding of the PPSN workshop on Multiobjective Problem Solving from Nature, year = 2006, month = Sept, location = Reykjavik, Iceland, url= http://dbkgroup.org/knowles/MPSN3/mpsn3chandra.pdf,*.

Cheatham, M. and Rizki, M. 2006. Feature and prototype evolution for nearest neighbor classification of web documents. In *Proceedings of Third International Conference on Information Technology: New Generations*. IEEE Computer Society, Las Vegas, NV, 364–369.

Chen, J.-H., Chen, H.-M., and Ho, S.-Y. 2005. Design of nearest neighbor classifiers: multi-objective approach. *International Journal of Approximate Reasoning 40*, 3–22.

Chikalov, I. 2011. *Intelligent Systems Reference Library: Average Time Complexity of Decision Trees*. Vol. 21. Springer-Verlag Berlin Heidelberg.

Coello, C. and Cortes, N. 2005. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines 6,* 2, 163–190.

Coello, C. A. C. 1999. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems 1,* 3, 129–156.

Cover, T. and Hart, P. E. 1967. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on 13,* 1, 21–27.

Cristianini, N. and Taylor, J. S. 2000. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-2. *Evolutionary Computation, IEEE Transactions on 6,* 2, 182–197.

Dietterich, T. 1995. Overfitting and undercomputing in machine learning. *ACM Computing Surveys 27*, 326–327.

Dietterich, T. 1997. Machine-learning research: four current directions. *AI Magazine 18*, 97–136.

E. Cantu-Paz, S. Newsam, C. K. 2004. Feature selection in scientific applications. In *Proceedings of the ACM SIGKDD Internations Conference on Knowledge Discover and Data Mining*. ACM, New York, USA, 788–793.

Eshelman, L. J. 1991. *The Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination*. Morgan Kauffman Inc., San Mateo, CA.

Fayyad, U. and K.Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1022–1027.

Fix, E. and J. L. Hodges, J. 1989. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review 57,* 3 (Dec.), 238–247.

Freidman, J. H., Bentley, J. L., and Finkel, R. A. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw. 3,* 3, 209–226.

Freund, Y. and Schapire, E. 1996. Experiments with a new boosting algorithm. In *ML96: Proceedings of the 13th National Conference on Machine Learning*. 148–156.

Fuller, R., Groom, G., and Jones, A. 1994. Photogramm. engg. *Remote Sensing 60*, 553–556.

Gates, G. 1972. The reduced nearest neighbor rule. *Information Theory, IEEE Transactions on 18,* 3 (May), 431–433.

Gil-Pita, R. and Yao, X. 2007. Using a genetic algorithm for editing k-nearest neighbor classifiers. *Intelligent Data Engineering and Automated Learning*, 1141–1150.

Goh, K. S., Lim, A., and Rodrigues, B. 2003. Sexual selection for genetic algorithms. *Artif. Intell. Rev. 19,* 2, 123–152.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gong, M., Jiao, L., Du, H., and Bo, L. 2008. Multiobjective immune algorithm with nondominated neighbor-based selection. *Evolutionary Computation, IEEE Transactions on 16,* 2, 225–255.

Gorman, R. and Sejnowski, T. 1988. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks 1*, 75–89.

Grosso, P. B. 1985. Computer simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model. In *Doctoral Dissertation*. The University of Michigan.

Gunter, S. and Bunke, H. 2004. Feature selection algorithms for the generation of multiple classifier systems and their application to handwritten word recognition. *Pattern Recognition Letters 25,* 11, 1323–1336.

H. Zhu, L. Jiao, J. P. 2006. Multi-population genetic algorithm for feature selection. *Natural Computation Techniques Applications*, 480–487.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl. 11*, 10–18.

Hall, M. and Smith, L. 1998. Practical feature subset selection for machine learning. In *ACSC98: Proceedings of the 21st Australasian Computer Science Conference*. Springer Berlin, 181–191.

Hart, E. and Timmis, J. 2008. Application areas of ais: The past, the present and the future. *Applied Soft Computing 8,* 1, 191–201.

Hart, P. 1968. The condensed nearest-neighbor rule. In *Information Theory, IEEE Transactions on*. Vol. IT-14. 515–516.

Haupt, R. and Haupt, S. 2004. *Practical Genetic Algorithms, 2nd Edition*. John Wiley and Sons, Inc.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

He, J. and Kang, L. 1999. On the convergence rates of genetic algorithms. *Theoretical Computer Science 229*, 23–29.

Ho, S.-Y., Liu, C.-C., and Liu, S. 2002. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters 23,* 13, 1495–1503.

Ho, T. 1998a. C4.5 decision forests. In *Pattern Recognition 1998: Proceedings of the Fourteenth International Conference on Pattern Recognition*. Vol. 1. 545–549.

Ho, T. 1998b. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 832–844.

Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MA, USA.

Horowitz, E., Sahni, S., and Rajasekaran, S. 1997. *Computer Algorithms*. Computer Science, New York, USA.

Hu, Q., Yu, D., and Wang, M. 2005. Constructing rough decisionforests. In *Proceedings of the 10th International Conference on rough sets, fuzzy sets, data mining, and granular computing*. RSFDGrC 2005. Springer Berlin, 147–156.

Hyafil, L. and Rivest, R. 1976. Constructing optimal binary decision trees is np-complete. *Information Processinf Letters 5,* 1, 15–17.

I. Witten, E. Frank, M. H. 2011. *Data Mining: Practical Machine Learning Tools And Techniques Third Edition*. Morgan Kaufmann.

Ishibuchi, H. and Nakashima, T. 2000. Multi-objective pattern and feature selection by a genetic algorithm. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*. 1069–1076.

J. Yao, N. Kharma, P. G. 2005. Bmpga: a bi-objectvive multi-population genetic algorithm for multi-modal function optimization. In *Evolutionary Computation, IEEE Transactions on*. Vol. 1. 816–823.

Karr, L. and Freeman, M. 1998. *Industrial Applications of Genetic Algorithms*. CRC Press LLC.

Knowles, J. and Corne, D. 2000. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation, IEEE Transactions on 8,* 2, 149–172.

Konak, A., Coit, D., and Smith, A. 2006. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety 91,* 9, 992–107.

Kreyszig, E. 1999. *Advanced Engineering Mathematics*. John Wiley and Sons, Inc.

Kuncheva, L. 2003. That elusive diversity in classifier ensembles. In *Pattern Recognition and Image Analysis*, F. Perales, A. Campilho, N. Blanca, and A. Sanfeliu, Eds. Lecture Notes in Computer Science, vol. 2652. Springer Berlin Heidelberg, 1126–1138.

Kuncheva, L. and Bezdek, J. 1998. Nearest prototype classification: clustering, genetic algorithms, or random search? *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 28,* 1 (Feb), 160–164.

Kuncheva, L. and Jain, L. 1999. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters 20*, 1149–1156.

Kuncheva, L. and Whitaker, C. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning 51*, 181–207. 10.1023/A:1022859003006.

Kuncheva, L. I. Reducing the computational demand of the nearest neighbor classifier.

Liepins, G., Hillard, M., Richardson, J., and Palmer, M. 1990. Genetic algorithms application to set covering and traveling salesman problems. *Operations research and Artificial Intelligence: The integration of problem-solving strategies*, 29–57.

Lim, T. and Shih, Y. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning 40*, 203–229.

Lin, F., Zhou, C., and Chang, K. 2010. Convergence rate analysis of allied genetic algorithm. In *Decision and Control (CDC), 2010 49th IEEE Conference on*. 786–791.

Lin, S., Punch, W., and Goodman, E. 1994. Coarse-grain parallel genetic algorithms: categorization and new approach. In *Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing*. IEEE Computer Society Press, Los Angeles, CA, USA.

Lis, J. and Eiben, A. 1996. A multi-sexual genetic algorithm for multiobjective optimization. In *Proceedings of the 1996 International Conference on Evolutionary Computation*, T. Fukuda and T. Furuhashi, Eds. IEEE, Nagoya, Japan, 59–64.

Llora, X. and Guiu, J. M. 2003. Prototype induction and attribute selection via evolutionary algorithms. *Intell. Data Anal. 7,* 3, 193–208.

Lu, Z., Wu, X., Zhu, X., and Bongard, J. 2010. Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '10. ACM, New York, NY, USA, 871–880.

Man, K., Tang, K., and Kwong, S. 1996. Genetic algorithms: Concepts and applications. *IEEE Transactions On Industrial Electronics 43,* 5 (October), 519–534.

Ming, L., Wang, Y., and Cheung, Y. 2006. On convergence rate of a class of genetic algorithms. In *Automation Congress, 2006. WAC '06. World*. 1–6.

Moon, C., Kim, J., Choi, G., and Seo, Y. 2002. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research 140,* 3, 606–617.

Munetomo, M., Takai, Y., and Sato, Y. 1993. An efficient migration scheme for subpopulation-based asynchronously parallel genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 649.

Murthy, S. K. and Salzberg, S. 1995. Decision tree induction: How effective is the greedy heuristic? In *KDD95: Proceedings of the 1st International Conference On KDD and DM*. 222–227.

Nene, S. A. and Nayar, S. K. 1997. A simple algorithm for nearest neighbor search in high dimensions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*.

Newman, C. B. D. and Merz, C. 1998. UCI repository of machine learning databases.

Nilsson, N. 1998. *Introduction to Machine Learning: An Early Draft of a Proposed Textbook*.

Okun, O., Valentini, G., and Re, M. 2011. *Studies in Computational Intelligence: Ensembles in Machine Learning Applications*. Vol. 373. Springer-Verlag Berlin Heidelberg.

Oza, N. and Tumer, K. 2008. Classifier ensembles: Select real-world applications. *Information Fusion 9,* 1, 4–20.

Parvin, H., Alizadeh, H., and Bidgoli, B. 2009. Using clustering for generating diversity in classifier ensemble. *JDCTA 3,* 1, 51–57.

Podgorelec, V. and Kokol, P. 2001. Evolutionary decision forests - decision making with multiple evolutionary constructed decision trees. *Problems In Applied Mathematics and Computational Intelligence*, 97–103.

Popescu, M., Popescu, L., and Mastorakis, N. 2009. Applications of genetic algorithms. *WSEAS Trans. Info. Sci. and App. 6*, 1782–1791.

Quinlan, J. R. 1993. *Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Quirino, T. and Kubat, M. 2010. Instinct-based mating in genetic algorithms applied to the tuning of 1-nn classifiers. *Knowledge and Data Engineering, IEEE Transactions on 22,* 12, 1724–1737.

Raguwanshi, M. M. and Kakde, O. G. 2006. Genetic algorithm with species and sexual selection. In *CIS-RAM 06: IEEE International Conferences on Cybernetics, Intelligent systems, Robotics, Automation, and Mechatronics*. Bangkok, Thailand.

Raymer, M., Punch, W., Goodman, E., Kuhn, L., and Jain, A. 2000. Dimensionality reduction using genetic algorithms. *Evolutionary Computation, IEEE Transactions on 4,* 2, 164–171.

Ritzel, B., Eheart, J., and Ranjithan, S. 1994. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research 30,* 5, 1589–1603.

Rodriguez, J., Kuncheva, L., and Alonso, C. 2006. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28,* 10 (Oct), 1619–1630.

Rokach, L. 2008. An evolutionary algorithm for constructing a decision forest: Combining the classification of disjoints decision trees. *International Journal of Intelligent Systems 23,* 4, 455–482.

Rokach, L. 2010. *Pattern Classification Using Ensemble Methods*. World Scientific Publishing Co.

Rozsypal, A. and Kubat, M. 2003. Selecting representative examples and attributes by a genetic algorithm. *Intell. Data Anal. 7,* 4, 291–304.

Rukovansky, I. 2009. Optimization of the throughput of computer network based on parallel ea. In *Proceedings of the World Congress on Engineering and Computer Science, WCECS-09*. Vol. 2. San Francisco, CA, 1038–1043.

S. Chen, D. Whitley, S. S. 1999. Fast and accurate feature selection using hybrid genetic strategies. In *CEC99: Proceedings of the Congress on Evolutionary Computation*. 177–184.

Safe, M., Carballido, J., Ponzoni, I., and Brignole, N. 2004. On stopping criteria for genetic algorithms. *Advances in Artificial Intelligence SBIA 2004 3171*, 405–413.

Schaffer, J. 1985. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. 1st International Conference on Genetic Algorithms*. Lawrebce Erlbaum, 93–100.

ScienceDaily. 2009. Opposites attract: How genetics influences humans to choose their mates. *European Society of Human Genetics*.

Shamir, N., Saad, D., and Marom, E. 1993. Preserving the diversity of a genetically evolving population of nets using the functional behavior of neurons. *Complex Systems 7,* 5, 327–346.

Skurichina, M. and Duin, R. 2002. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis and Applications 5*, 121–135.

Soryani, M. and Rafat, N. 2006. Application of genetic algorithms to feature subset selection in a farsi ocr. In *Proceedings of World Academy of Scinece, Engineering and Technology*. Vol. 18. IEEE Computer Society, 364–369.

Sproull, R. F. 1991. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica 6,* 4, 579–589.

Srinivas, N. and Deb, K. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, IEEE Transactions on 2,* 3, 221–248.

Stender, J. 1993. *Parallel Genetic Algorithms: Theory and Applications*. IOS Press.

Tan, P., Steinbach, M., and Kumar, V. 2005. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Tomek, I. 1976. Two modifications of cnn. *Systems Man and Communications, IEEE Transactions on*, 769–772.

Tsymbal, A., Pechenizkiy, M., and Cunningham", P. 2005. Diversity in search strategies for ensemble feature selection. *Information Fusion 6,* 1, 83–98.

Velazco, J. and Bullinaria, J. 2003a. Gendered selection strategies in genetic algorithms for optimization. In *Proceedings of the Proceedings of the UK Workshop on Computational Intelligence*. University of Bristol, Bristol, UK, 217–223.

Velazco, J. and Bullinaria, J. 2003b. Sexual selection with competitive/co-operative operators for genetic algorithms. *International Conference on Neural Networks and Computational Intelligence*, 217–223.

Wilson, D. L. 1972. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on 2,* 3, 408–421.

Wu, Y. and Zhang, A. 2004. Feature selection for classifying high-dimensional numerical data. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*. IEEE Computer Society, Washington, DC, USA, 251–258.

X. Yu, M. G. 2010. *Introduction To Evolutionary Algorithms*. Springer London Heidelberg.

Zhou, Z. 2009. Ensemble learning. *Encyclopedia of Biometrics*, 270–273.

Zhu, X., Wu, X., and Chen, Q. 2003. Eliminating class noise in large datasets. In *ICML2003: In Proceeding of International Conference on Machine Learning*. 920–927.

Zhu, Y., Yang, Z., and Song, J. 2006. A genetic algorithm with age and sexual features. In *ICIC*. 634–640.

Zitzler, E., Laumanns, M., and Thiele, L. 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on 3,* 4, 257–271.