January 2012

Mining Aircraft Telemetry Data With Evolutionary Algorithms

Kirk Anders Ogaard

MINING AIRCRAFT TELEMETRY DATA WITH EVOLUTIONARY ALGORITHMS

by

Kirk A. Ogaard
Bachelor of Science, University of North Dakota, 1999
Master of Science, University of North Dakota, 2008

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Grand Forks, North Dakota
May
2012

This dissertation, submitted by Kirk A. Ogaard in partial fulfillment of the requirements for the Degree of Doctor of Philosophy from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done, and is hereby approved.

Ronald Marsh

Hassan Reza

Emanuel Grant

Timothy Young

Elizabeth Bjerke

This dissertation is being submitted by the appointed advisory committee as having met all of the requirements of the Graduate School at the University of North Dakota and is hereby approved.

Wayne Swisher

4/10/2012

Title Mining Aircraft Telemetry Data with Evolutionary Algorithms

Department Computer Science

Degree Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in his absence, by the Chairperson of the department or the dean of the Graduate School. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my dissertation.

Kirk A. Ogaard
March 30, 2012

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Ronald Marsh, for the honor of being the first student to graduate from the doctoral program in scientific computing. I would also like to thank my committee members—Dr. Hassan Reza, Dr. Emanuel Grant, Dr. Elizabeth Bjerke, and Dr. Timothy Young—for their valuable feedback on my dissertation. And finally, I would like to thank Jim Higgins for providing me with the aircraft telemetry data.

To my wife, Tina

ABSTRACT

The Ganged Phased Array Radar – Risk Mitigation System (GPAR-RMS) was a mobile ground-based sense-and-avoid system for Unmanned Aircraft System (UAS) operations developed by the University of North Dakota. GPAR-RMS detected proximate aircraft with various sensor systems, including a 2D radar and an Automatic Dependent Surveillance – Broadcast (ADS-B) receiver. Information about those aircraft was then displayed to UAS operators via visualization software developed by the University of North Dakota. The Risk Mitigation (RM) subsystem for GPAR-RMS was designed to estimate the current risk of midair collision, between the Unmanned Aircraft (UA) and a General Aviation (GA) aircraft flying under Visual Flight Rules (VFR) in the surrounding airspace, for UAS operations in Class E airspace (i.e. below 18,000 feet MSL). However, accurate probabilistic models for the behavior of pilots of GA aircraft flying under VFR in Class E airspace were needed before the RM subsystem could be implemented.

In this dissertation the author has documented his research on a novel application of an ant colony algorithm to the synthesis of aircraft telemetry data, which was then data mined to discover probabilistic models of the behavior of pilots of GA aircraft flying under VFR in Class E airspace. The results of data mining an aircraft telemetry data set from a consecutive nine month period in 2011 are presented. This aircraft telemetry data set consisted of Flight Data Monitoring (FDM) data obtained from Garmin G1000 devices which are onboard every Cessna 172 in the University of North Dakota's training fleet. Data from aircraft which were potentially within the controlled airspace

surrounding controlled airports were excluded. Also, GA aircraft which operated in Class E airspace were assumed to have been flying under VFR, which is a valid assumption for most training flights. First, complex subpaths were discovered from the aircraft telemetry data set using a novel application of an ant colony algorithm. Then, probabilistic models were data mined from those subpaths using extensions of the Genetic K-Means (GKA) and Expectation-Maximization (EM) algorithms.

The results obtained from the subpath discovery and data mining suggest: 1) at both low altitudes (between 597 and 3,589 feet MSL) and high altitudes (between 3,590 and 12,860 feet MSL) a pilot flying a GA aircraft near to an uncontrolled airport will perform different maneuvers than a pilot flying a GA aircraft far from an uncontrolled airport and 2) when only maneuvers with a duration of one minute or longer were considered, all variations of left turns were performed less frequently than 1%, while many variations of straight flight and right turns were performed more frequently than 1%. However, since only aircraft telemetry data from the University of North Dakota's training fleet were data mined, these results are not likely to be applicable to GA aircraft operating in a non-training environment.

CHAPTER I

INTRODUCTION

An Unmanned Aircraft (UA) is any aircraft designed to fly without an onboard pilot or crew. At present, UAs are designed to be remotely piloted from the ground via direct and/or indirect wireless communication links. There are many different types of UAs designed for many different types of applications in military and/or civilian operational environments. Civilian applications include surveillance of dangerous environments (e.g. flooded rivers) and acquisition of aerial imagery for scientific or industrial endeavors (e.g. assessing crop growth in farm fields).

In (Loegering and Evans 1999), the design process for the Global Hawk UA is described. The Global Hawk UA is designed for extended aerial surveillance from altitudes as high as 65,000 feet above Mean Sea Level (MSL). Thus, the UA's design includes redundant flight controls to increase its reliability during long flights. All the control surfaces for the UA are split, with each half controlled by a separate actuator. The UA also has redundant avionics computers. For each split control surface on the UA, one avionics computer controls the actuator for one half, and the other avionics computer controls the actuator for the other half. Thus, if one of the avionics computers and/or actuators for a control surface fails, the UA can still remain airborne by relying on the other redundant component.

Civilian Applications of Unmanned Aircraft

In (Nonami 2007), several different types of civilian UAs, including Yamaha's RMAX unmanned helicopter, Aerosonde's UA, and General Atomic's Altair, Altus I, and Altus II are described. Research into the semi-autonomous operation of UAs in the National Airspace System (NAS), i.e. airborne sense-and-avoid technology is also described in (Nonami 2007). Current problems with semi-autonomous UAs (e.g. collision avoidance) are assumed in (Nonami 2007) to be solvable by further advances in the computational power of CPUs. Ground-based sense-and-avoid technologies for UAs, such as the Ganged Phased Array – Risk Mitigation System (GPAR-RMS) described in (Marsh, et al. 2011), are not discussed in (Nonami 2007).

In (Hunt, et al. 2010), an application of the Vector-P UA is described in which the UA provided aerial imagery for assessing crop growth. The UA was flown with the MicroPilot Ground Control Station (GCS) software. Due to the UA's small size, the digital camera attached to the UA needed to be light-weight. The aerial imagery obtained during the UA's two test flights provided an adequate assessment of the crop growth for two farm fields. However, the Federal Aviation Administration (FAA) currently restricts Unmanned Aircraft System (UAS) operations in the NAS, thus limiting the feasibility of assessing crop growth with UAs (Hunt, et al. 2010).

Restrictions on Civilian Operation of Unmanned Aircraft

While UAs have been remarkably successful in U.S. military operations (Weibel and Hansman 2004), FAA safety regulations impose strict limitations on their civilian operation in the NAS (Dalamagkidis, Valavanis, and Piegl 2008). In (Dalamagkidis, Valavanis, and Piegl 2008), current regulatory problems in the U.S. regarding UAs flying

in the NAS are described. Many federal agencies could benefit from civilian UASs operating in the NAS, including the U.S. Coast Guard, the U.S. Customs and Border Protection, and the U.S. Department of Agriculture (Dalamagkidis, Valavanis, and Piegl 2008). However, the FAA will not permit unfettered UAS operations in the NAS, until those UAS operations are demonstrated to have an equivalent level of safety to manned aircraft systems (Dalamagkidis, Valavanis, and Piegl 2008).

Collision risk is defined in (Dalamagkidis, Valavanis, and Piegl 2008) as the potential for a UA to damage people and/or property as a result of a midair collision or ground collision. Furthermore, (Dalamagkidis, Valavanis, and Piegl 2008) states the risk of a particular collision is determined by the probability of the collision occurring and the amount of damage it would cause to people and/or property. Probabilistic models for General Aviation (GA) aircraft, such as those presented in chapter 4, could only be used to calculate the first type of collision risk, i.e. the risk of a midair collision between a UA and a manned aircraft. Any probabilistic models for calculating the second type of collision risk (i.e. the risk of a UA colliding with the ground) would need to be based on entirely different factors, and are beyond the scope of this research.

Current FAA regulations only permit UAS operations in the NAS on a case-by-case basis via Certificates of Authorization (COAs). A COA designates a restricted airspace for an authorized organization to fly a particular type of UA. While the authorized organization is flying its UA, manned aircraft are not permitted to enter the COA's restricted airspace. COAs usually expire after a year (Dalamagkidis, Valavanis, and Piegl 2008).

According to FAA guidelines, severe collision events (i.e. collisions resulting in numerous casualties and/or annihilation of aircraft) for manned aircraft should occur less frequently than 10^{-9} occurrences per flight hour (Federal Aviation Administration 2008). Furthermore, (Federal Aviation Administration 2008) states that severe collision events should rarely occur during the entire service of an aircraft. Since current FAA regulations require the safety levels for UAS operations in the NAS to be equivalent to the safety levels for manned aircraft systems, UAS operations in the NAS must be demonstrated to result in less than 10^{-9} severe collision events per flight hour (Dalamagkidis, Valavanis, and Piegl 2008).

Proposed Collision Avoidance Strategies

Since many different types of UAs exist, each with different performance characteristics, (Dalamagkidis, Valavanis, and Piegl 2008) suggests categorizing UAs based on the altitude ranges which they operate at most frequently—low altitudes (e.g. the Vector-P UA), medium altitudes (e.g. the RQ-1 Predator), or high altitudes (e.g. the Global Hawk). Specific collision-avoidance strategies are suggested for UASs operating in each of these three altitude ranges. Two general strategies are also identified in (Dalamagkidis, Valavanis, and Piegl 2008) for avoiding midair collisions between a UA and a manned aircraft: 1) maintain adequate horizontal and vertical separation between the UA and proximate aircraft or 2) if adequate separation cannot be maintained, the UA must activate an airborne sense-and-avoid system. However, if adequate separation between the UA and proximate aircraft can be consistently maintained with a ground-based sense-and-avoid system, such as GPAR-RMS, an airborne sense-and-avoid system is not necessary. Probabilistic models for GA aircraft, such as those discovered using the

methodology discussed in chapter 3, could have enabled GPAR-RMS to estimate the risk of a midair collision between the UA and other GA aircraft in the surrounding airspace. With a current and accurate estimate of the risk of a midair collision, the Range Safety Operator (RSO) could have immediately landed the UA if the estimated risk of a midair collision was too great (Marsh, et al. 2011).

Although many commercial aircraft are equipped with collision avoidance technology based on Automatic Dependant Surveillance – Broadcast (ADS-B) and/or Traffic Collision Avoidance System (TCAS), neither of these technologies are currently available for most UAs (Dalamagkidis, Valavanis, and Piegl 2008). Furthermore, most GA aircraft in the U.S. currently do not have ADS-B or TCAS capability. Thus, other systems for avoiding midair collisions between a UA and proximate aircraft (i.e. ground-based or airborne sense-and-avoid systems) need to be developed (Dalamagkidis, Valavanis, and Piegl 2008).

In (Weibel and Hansman 2004), a preliminary assessment of the risk of a midair collision between a UA and a manned aircraft is described. The likelihood of such a midair collision is estimated in (Weibel and Hansman 2004) using a gas model simulation. The gas model simulation described in (Weibel and Hansman 2004) characterizes the UA solely by its mass, since aircraft mass is considered a crucial factor in midair collisions. Furthermore, the masses of different types of UAs vary significantly depending on their intended applications (Weibel and Hansman 2004). The density of manned aircraft in the gas model simulation was calculated from a single day of data logged by the FAA's Enhanced Traffic Management System (ETMS). ETMS is used by the FAA to archive data from the ground-based radars used by Air Traffic Control (ATC)

towers at controlled airports. In (Weibel and Hansman 2004), the UA was considered to have a uniform probability of occupying any position in the airspace, regardless of its mass.

The results from the gas model simulation in (Weibel and Hansman 2004) suggest the risk of midair collision between a UA and a manned aircraft is much higher when the density of manned aircraft is greater, e.g. within FAA-designated airways. The results also suggest smaller UAs (i.e. with less mass) are much less likely to cause severe collision events than larger UAs (Weibel and Hansman 2004). Although the preliminary results from (Weibel and Hansman 2004) provide some insight into the total risk of midair collisions between UAs and manned aircraft in the NAS, these findings are not applicable to the risk of midair collisions for specific airspace configurations.

In (Marsh, et al. 2011), the visualization software developed for GPAR-RMS is described. GPAR-RMS was a mobile ground-based sense-and-avoid system developed at the University of North Dakota. GPAR-RMS was composed of proprietary hardware and software systems for monitoring UAS operations. These hardware and software systems were installed in a fifth-wheel trailer which was quickly transportable to an area for field deployment of the UAS. The hardware system was composed of two rack-mounted sets of high-speed, multi-core servers and a set of external sensors, including a Garmin GDL 90 ADS-B transceiver, a DeTect Harrier 2D radar (which was networked to the fifth-wheel trailer via a wireless bridge), a Davis Weather Monitor II weather station, and a Garmin Global Positioning System (GPS) puck. The hardware system also included one or more GCSs for controlling the UA(s), i.e. an Insitu ScanEagle GCS and/or a MicroPilot GCS.

The software system was composed of the following separate components: 1) a Sensor Fusion System (SFS) which received data streams from the external sensors (i.e. the Garmin GDL 90 ADS-B transceiver, the DeTect Harrier 2D radar, the GCS(s), and the Davis Weather Monitor II weather station), fused the aircraft telemetry data, and multicast the fused data and the meteorological measurements from the weather station to one or more Information Display Systems (IDSs), 2) the Risk Mitigation (RM) subsystem which estimated the risk of collision for the UA based on the current airspace configuration, 3) the Health Monitor which displayed the overall system health for GPAR-RMS, including the health of the external sensors and the internal temperatures of the rack-mounted servers, 4) the Range Control Center (RCC) IDS which displayed the information received from SFS in a suitable format for the RSO, and 5) the Ground Observer (GO) IDS which displayed the information received from SFS in a suitable format for a ground observer or as an additional display for the UA pilot. Whereas the RCC IDS displayed a top-down view of the current airspace configuration, the GO IDS displayed a view of the current airspace configuration which was centered on the UA (Marsh, et al. 2011).

The centralized control architecture of GPAR-RMS was effective for UAS operations involving one or possibly two UAs, but a distributed control architecture would be more effective for controlling and monitoring UAS swarms (Reza and Ogaard 2011). UAS swarms consist of many small UAs flying together. However, the problem of ground-based sense-and-avoid for UAS swarms is beyond the scope of this research.

Problem Statement

A critical part of GPAR-RMS which was unfinished was the RM subsystem (Marsh, et al. 2011). Estimating the risk of midair collision for a specific airspace configuration is algorithmically complex. The risk must be estimated for every possible midair collision between the UA and every other GA aircraft in the surrounding airspace. Furthermore, the estimated risk must be accurate and updated in near real time. Thus, in order for an algorithm (such as one that could have been integrated into the RM subsystem) to estimate the risk of a midair collision between the UA and another GA aircraft, the algorithm needs the probabilities of the pilot of that particular type of GA aircraft performing various maneuvers during the next minute. The intent of the UA pilot would obviously be known.

Although the set of basic maneuvers (e.g. straight ascents or descents, ascending or descending turns, and level turns) available to pilots of GA aircraft are known, the specific flight path a pilot chooses for an aircraft can be composed of any combination of these basic maneuvers. Also, many variations exist for each of the basic maneuvers. The pilot of an aircraft may, for instance, perform a level turn at different rates, such as 2° per second or 3° per second. Class E airspace is defined in (Federal Aviation Administration 2011) as national airspace below 18,000 feet MSL which is not already Class A, B, C, D, or G airspace. Furthermore, (Federal Aviation Administration 2011) states pilots flying aircraft under Visual Flight Rules (VFR) in Class E airspace are not required to communicate with ATC. Hence, probabilistic models were needed to predict the behavior of pilots of those types of manned aircraft which typically operate under VFR in Class E

airspace (i.e. GA aircraft). The problem of how accurate probabilistic models for the behavior of pilots of GA aircraft flying under VFR in Class E airspace can be discovered is the focus of this dissertation.

The methodology discussed in chapter 3, which was used to discover probabilistic models for the behavior of pilots of GA aircraft, involves data mining massive aircraft telemetry data sets—specifically Flight Data Monitoring (FDM) data sets. These aircraft telemetry data sets contain very accurate data about the flight paths of FDM-capable GA aircraft over a specific period of time. The positions reported by the telemetry devices on these GA aircraft are provided by the Wide Area Augmentation System (WAAS), which augments the accuracy of GPS in order to meet FAA requirements (Federal Aviation Administration 2010). WAAS provides a horizontal positional accuracy of about 1 meter and a vertical positional accuracy of about 1.5 meters in most cases (Federal Aviation Administration 2006). With such horizontal and vertical positional accuracy, discovering accurate probabilistic models of GA pilot behavior by data mining massive aircraft telemetry data sets becomes feasible.

These probabilistic models could then be used to estimate the total risk of a midair collision for a UA flying in Class E airspace needed by the RM subsystem of GPAR-RMS. First, the RM subsystem would filter out any GA aircraft in the surrounding airspace which were determined to be: a) within the controlled airspace surrounding any nearby controlled airport(s) and/or b) flying at such a distance and velocity from the UA to not be considered a possible conflict. Then, the RM subsystem would determine the applicable probabilistic model for each remaining GA aircraft in the surrounding airspace based on the GA aircraft's altitude above MSL and its proximity to the nearest

uncontrolled airport. Thus, each potentially conflicting GA aircraft would have an associated probabilistic model. Finally, the maneuver probabilities from these probabilistic models associated with the potentially conflicting GA aircraft in the surrounding airspace would be used to estimate the probability of any of those GA aircraft colliding with the UA in the next minute.

CHAPTER II

BACKGROUND

Data mining algorithms are a very diverse class of algorithms used to search for meaningful patterns in data sets. Metaheuristic algorithms (which are related to data mining algorithms) are a broader class of algorithms used to solve combinatorial optimization problems. Heuristic algorithms, like metaheuristic algorithms, can also be used to solve combinatorial optimization problems. Many data mining algorithms are actually heuristic algorithms. However, although heuristic algorithms typically have faster run times, such algorithms are susceptible to converging to suboptimal solutions. These suboptimal solutions are locally optimal, but not globally optimal, solutions to the combinatorial optimization problems. Such locally optimal solutions may be invalid solutions to the combinatorial optimization problems. Certain classes of metaheuristic algorithms, e.g. canonical genetic algorithms which save their most optimal solutions, have been proved to eventually converge to the globally optimal solution to any combinatorial optimization problem (Rudolph 1994). Thus, hybrid algorithms which combine data mining algorithms with metaheuristic algorithms are pertinent to discovering globally optimal probabilistic models of the behavior of pilots of GA aircraft flying under VFR in Class E airspace.

Metaheuristic Algorithms

In (Yagiura and Ibaraki 2001), metaheuristic algorithms are discussed. A combinatorial optimization problem involves searching for the globally optimal solution

in some countable set of candidate solutions (Bianchi et al. 2009). A heuristic algorithm, e.g. the K-Means Algorithm (KMA), iteratively improves an approximate solution (i.e. a candidate solution) to a combinatorial optimization problem using some predetermined optimization criteria (Yagiura and Ibaraki 2001). The heuristic algorithm runs until no further improvements to its candidate solution are possible with the given optimization criteria (Greiner 1996). However, a heuristic algorithm is not guaranteed to discover the globally optimal solution to a combinatorial optimization problem (Greiner 1996). A heuristic algorithm may discover different locally optimal solutions when run with different initial conditions (Greiner 1996). Metaheuristic algorithms, e.g. the Genetic K-Means Algorithm (GKA), extend the combinatorial optimization capabilities of traditional heuristic algorithms by performing a broader search for the globally optimal solution (Yagiura and Ibaraki 2001).

Some examples of metaheuristic algorithms are multi-start local search, Tabu search, genetic algorithms, and simulated annealing (Yagiura and Ibaraki 2001). A multi-start local search algorithm is an extension of the local search algorithm (Yagiura and Ibaraki 2001). A local search algorithm is a heuristic algorithm which iteratively improves its candidate solution by replacing it with neighboring candidate solutions which are more optimal according to the predetermined optimization criteria (Arya, et al. 2004). The definition of the neighborhood for a candidate solution depends on the particular combinatorial optimization problem to be solved (Arya, et al. 2004). However, a local search algorithm is only capable of searching for optimal solutions from a small subset of the entire set of candidate solutions (Arya, et al. 2004). This subset of candidate solutions is determined by: a) the initial conditions and b) the optimization criteria (Arya,

et al. 2004). A multi-start local search algorithm extends a local search algorithm by searching a larger subset of the entire set of candidate solutions (Yagiura and Ibaraki 2001). It iteratively performs local searches with different initial conditions (Yagiura and Ibaraki 2001).

Another extension of the local search algorithm is the Tabu search algorithm (Yagiura and Ibaraki 2001). A Tabu search algorithm maintains a Tabu list, i.e. a fixed-length history of previous candidate solutions which are considered suboptimal (Glover 1989; Glover 1990). If the optimization criteria cannot generate a candidate solution which is not already in the Tabu list, the Tabu search algorithm uses some predetermined method for generating a new candidate solution, e.g. randomly perturbing the current candidate solution (Glover 1989; Glover 1990).

A genetic algorithm mimics biological evolutionary processes. It starts by generating a random population of candidate solutions to the combinatorial optimization problem (Yagiura and Ibaraki 2001). Then, it iteratively evolves each new generation of candidate solutions through the stochastic application of genetic operators (e.g. inheritance, mutation, and crossover) to selected candidate solutions from the previous generation (Yagiura and Ibaraki 2001).

A simulated annealing algorithm mimics the metallurgical technique of annealing. It repeats the following steps to search for the globally optimal solution to the combinatorial optimization problem (Yagiura and Ibaraki 2001): 1) A candidate solution is randomly generated; 2) Let Δ be the difference between the optimality of the candidate solution and the current solution S; 3) If $\Delta \leq 0$ (i.e. the candidate solution is at least as optimal as S), the candidate solution will always be selected to replace S; and 4) If $\Delta > 0$

(i.e. the candidate solution is less optimal than S), the candidate solution will be selected to replace S with a probability of $e^{\frac{-\Delta}{T}}$ where T is the current value of the temperature parameter (Yagiura and Ibaraki 2001). Thus, the value of the temperature parameter T affects the probability of a suboptimal candidate solution being selected to replace S (Yagiura and Ibaraki 2001). A simulated annealing algorithm typically starts with a large value for T, and then decreases it by a small amount with each iteration (Yagiura and Ibaraki 2001). Thus, the simulated annealing algorithm tries to avoid converging to a locally optimal solution by occasionally choosing suboptimal candidate solutions (Yagiura and Ibaraki 2001).

Although most metaheuristic algorithms randomly generate the initial candidate solution(s), some metaheuristic algorithms try to find better initial candidate solution(s) using more sophisticated techniques, e.g. a greedy search algorithm (Yagiura and Ibaraki 2001). Also, since metaheuristic algorithms do not perform exhaustive searches of the entire set of candidate solutions, such algorithms typically have termination criteria such as: a) terminating after a predetermined number of iterations or b) terminating after the metaheuristic algorithm has not significantly improved the candidate solution(s) for a predetermined number of iterations (Yagiura and Ibaraki 2001). Finally, metaheuristic algorithms should be designed to search a statistically diverse sample of the entire set of candidate solutions (Yagiura and Ibaraki 2001).

Genetic Algorithms

Genetic algorithms are a class of metaheuristic algorithms modeled after evolutionary processes that occur in nature. Using homogeneous finite Markov chain analysis, (Rudolph 1994) proves, for the general case, that a genetic algorithm will

eventually converge to the globally optimal solution if: a) The genetic algorithm has mutation, crossover, and proportional selection operators; b) The mutation operator is applied separately to each component of each candidate solution with some nonzero probability; c) The genetic algorithm tracks the most optimal candidate solution discovered during its entire execution; and d) The genetic algorithm solves a static combinatorial optimization problem. Markov chain analysis is also used in (Eiben, Aarts, and Van Hee 1991) to prove, for the general case, that a genetic algorithm in which the fittest candidate solutions are always selected for reproduction will eventually converge to the globally optimal solution. Whether a specific instance of a genetic algorithm will eventually converge to the globally optimal solution mainly depends on: a) the correctness of its fitness function, since the fitness function is crucial to eliminating unproductive searches for the globally optimal solution (Eiben, Aarts, and Van Hee 1991), and b) whether its mutation operator is implemented as defined in (Rudolph 1994), i.e. there is a nonzero mutation probability for each component of a candidate solution which is independent of the mutation probabilities for the other components.

In (Snyder and Daskin 2005), a genetic algorithm for solving the generalized Traveling Salesman Problem (TSP) is discussed. This problem belongs to the Nondeterministic Polynomial Hard (NP-Hard) class of computational problems. The genetic algorithm in (Snyder and Daskin 2005) encodes its candidate solutions with random keys, and also applies a local improvement heuristic to its candidate solutions. The maximum run time for the genetic algorithm in (Snyder and Daskin 2005) during 41 test cases was 10.1 seconds, which was significantly faster than the five other non-genetic

algorithms that were tested. Furthermore, the genetic algorithm in (Snyder and Daskin 2005) is algorithmically simpler than non-genetic algorithms designed to solve the generalized TSP.

In (Kim, Abraham, and Cho 2007), a hybrid algorithm is discussed which combines a genetic algorithm with a bacterial foraging algorithm. The hybrid algorithm was used to tune a Proportional-Integral-Derivative (PID) controller for an automatic voltage regulator in a simulated environment (Kim, Abraham, and Cho 2007). Bacterial foraging algorithms mimic the way bacteria, such as *E. coli*, search for food while simultaneously avoiding toxic environments (Kim, Abraham, and Cho 2007). According to (Kim, Abraham, and Cho 2007), the *E. coli* bacterium uses the following four foraging strategies: 1) If the bacterium is in a non-toxic environment, it wanders randomly in search of food; 2) If the bacterium is swimming towards an increasingly nutritious environment (or a decreasingly toxic environment), it engages in food-seeking behavior; 3) If the bacterium is swimming towards a decreasingly nutritious environment (or an increasingly toxic environment), it engages in harm-avoidance behavior; and 4) In some situations the bacterium releases chemicals that will attract nearby *E. coli* bacteria and cause them to clump together to protect themselves from a toxic environment.

An important consideration in the design of a genetic algorithm is its constraint handling (Kim, Abraham, and Cho 2007). When a genetic algorithm applies genetic operators such as crossover and/or mutation, it may produce invalid candidate solutions (Kim, Abraham, and Cho 2007). Thus, a genetic algorithm which applies crossover and/or mutation operators must also have some form of constraint handling to eliminate these invalid candidate solutions (Kim, Abraham, and Cho 2007). A bacterial foraging

algorithm can efficiently perform such constraint handling, since foraging bacteria are essentially solving a constrained combinatorial optimization problem (Kim, Abraham, and Cho 2007). Thus, a hybrid algorithm combining these two classes of algorithms is useful for solving certain constrained combinatorial optimization problems (Kim, Abraham, and Cho 2007). In the simulated tuning of a PID controller for an automatic voltage regulator in (Kim, Abraham, and Cho 2007), the hybrid algorithm converged to the globally optimal solution in fewer generations than the non-hybrid genetic algorithm.

In (Pizzuti 2008), the GA-Net genetic algorithm for discovering communities in graphs is discussed. A community is defined in (Pizzuti 2008) as a set of vertices in the graph where the vertices are densely connected to other vertices within the set, but only sparsely connected to vertices not in the set. The number of communities is automatically determined by a genetic algorithm (Pizzuti 2008). In the test in (Pizzuti 2008), which used an artificial data set, a population of 300 candidate solutions was evolved for 30 generations. A reproductive mutation rate of 20% was used during the test (Pizzuti 2008). GA-Net detected communities in the artificial data set with an accuracy rate of about 80%. Thus, GA-Net's accuracy rate when tested with an artificial data set was comparable to equivalent non-genetic algorithms for discovering communities in graphs (Pizzuti 2008), such as (Newman and Girvan 2004).

In (Pandy and Padhy 2008), the run-time performance of genetic algorithms and Particle Swarm Optimization (PSO) algorithms for designing a Flexible AC Transmission System (FACTS) are compared. A FACTS is typically used to increase the reliability of a power grid (Pandy and Padhy 2008). Genetic algorithms have proven effective at finding the optimal parameters for control systems, especially when traditional optimization

methods are too cumbersome (Pandy and Padhy 2008). In a PSO algorithm, each particle in the swarm is considered a separate candidate solution (Pandy and Padhy 2008). The PSO algorithm starts with the particles randomly wandering through the search space (Pandy and Padhy 2008). Less successful particles try to improve their candidate solutions by imitating more successful particles (Pandy and Padhy 2008). Each particle also remembers the best solution it has hitherto discovered (Pandy and Padhy 2008).

In order to compare the two algorithms, the genetic algorithm and the PSO algorithm in (Pandy and Padhy 2008) were subjected to identical tests. The results of the tests indicate that although the PSO algorithm in (Pandy and Padhy 2008) converges to the globally optimal solution in fewer iterations (i.e. generations), it is more CPU-intensive than the genetic algorithm. However, both classes of algorithms exhibited acceptable performance at optimizing designs for FACTS (Pandy and Padhy 2008).

Ant Colony Algorithms

Ant colony algorithms are a class of metaheuristic algorithms that mimic the path-building behavior of ants in nature (Gutjahr 2000). In (Botee and Bonabeau 1998), a hybrid algorithm is discussed for solving specific instances of TSP. The hybrid algorithm in (Botee and Bonabeau 1998) combines an ant colony algorithm with a genetic algorithm. Ant colony algorithms mimic the ability of foraging ants to discover the shortest (i.e. the most optimal) path to a food source (Botee and Bonabeau 1998). The ant that finds the shortest path to the food source will also be the first ant to successfully return to the ant hill with food (Botee and Bonabeau 1998). Once this ant finds the food source, it will return to the ant hill along the same path, thus doubling the strength of its pheromone trail (Botee and Bonabeau 1998). The strength of its pheromone trail

increases faster than any other pheromone trails leading to that food source, resulting in other ants preferentially following its pheromone trail (Botee and Bonabeau 1998). Furthermore, suboptimal pheromone trails are quickly eliminated by pheromone evaporation (Botee and Bonabeau 1998).

The hybrid algorithm in (Botee and Bonabeau 1998) uses a genetic algorithm to find the optimal parameters for the ant colony algorithm to use for solving a specific instance of the TSP. The hybrid algorithm in (Botee and Bonabeau 1998) results in the ant colony algorithm converging to the globally optimal solution faster. However, combining an ant colony algorithm with a genetic algorithm also resulted in a slower overall run time for the hybrid algorithm (Botee and Bonabeau 1998).

In (Gutjahr 2000), the Graph-Based Ant System (GBAS) ant colony algorithm is discussed. The GBAS ant colony algorithm represents a combinatorial optimization problem as a construction graph (Gutjahr 2000). A construction graph is defined in (Gutjahr 2000) as a special type of directed graph where every path shares a common start node. Time is represented by cycles, which are defined in (Gutjahr 2000) as complete traversals of the construction graph by all the ants (i.e. the agents). The weight assigned to an edge is the probability of an ant traversing it (Gutjahr 2000). These weights are calculated from the digital pheromone strength (which evaporates at a rate directly proportional to the number of cycles) and the utility of ants traversing the edge (Gutjahr 2000). The utility of ants traversing a particular edge depends on the type of combinatorial optimization problem being solved (Gutjahr 2000). The GBAS ant colony algorithm in (Gutjahr 2000) was proved to converge to an optimal solution when certain

criteria are met. However, the GBAS ant colony algorithm was not implemented to confirm this convergence proof (Gutjahr 2000).

The runtime complexities of two ant colony algorithms are analyzed in (Gutjahr 2008), the GBAS (Gutjahr 2000) and Ant System (Dorigo, Maniezzo, and Colorni 1996) algorithms. Both ant colony algorithms were shown in (Gutjahr 2008) to find an optimal solution to a test problem in linearithmic time, i.e. $O(n \cdot \log n)$. However, since the run-time complexities of the ant colony algorithms were only analyzed with a single test problem, the results of the analysis may not be applicable to other types of problems (Gutjahr 2008).

In (Han and Shi 2007), a hybrid algorithm is discussed for image segmentation which combines an ant colony algorithm with fuzzy clustering. The probability of an ant selecting a particular digital pheromone trail to follow is directly proportional to the length of the digital pheromone trail and the strength of the trail's digital pheromones (Han and Shi 2007). Typically, the ant also gives preference to digital pheromone trails which are closer to it (Han and Shi 2007). For example, to segment an image with respect to its gray values using the hybrid algorithm in (Han and Shi 2007), the following steps are performed: 1) The pixels of the target image are preprocessed into three-dimensional data points, where each data point consists of the gray value, the gradient, and the weight for the pixel; 2) A gray-scale histogram is constructed for the image; 3) The number and values of the peaks in the histogram are used to determine the number of clusters and the gray values for their initial centroids; 4) The gradients for the initial centroids are calculated; 5) The weights for the initial centroids are calculated; 6) The ants probabilistically construct paths through the 3D Euclidean space (defined by the data

points) based on the lengths and strengths of previous digital pheromone trails; and 7) An ant is assigned to whichever cluster has the closest centroid to its current position. When the hybrid algorithm in (Han and Shi 2007) was tested with other traditional image segmentation algorithms, e.g. Sobel edge detection, the hybrid algorithm in (Han and Shi 2007) was found to be more effective at extracting interesting features from the image. However, the tests discussed in (Han and Shi 2007) were only performed on a limited number of images.

In (Parunak, Purcell, and O'Connell 2002), an ant colony algorithm for controlling UA swarms is discussed. The current velocity for each UA in the swarm is determined by a vector field (Parunak, Purcell, and O'Connell 2002). The vectors in the vector field simultaneously direct the UAs in the swarm towards the desired target and away from hazards in the environment (Parunak, Purcell, and O'Connell 2002). The ant colony algorithm in (Parunak, Purcell, and O'Connell 2002) dynamically constructs a vector field from digital pheromones deposited by UAs in the swarm. Thus, if several UAs in the swarm find the same path to a particular target, other UAs in the swarm will also tend to follow that digital pheromone trail to the target (Parunak, Purcell, and O'Connell 2002). Similarly, if several UAs find the same path around a particular hazard in the environment, other UAs in the swarm will tend to follow it too (Parunak, Purcell, and O'Connell 2002). The ant colony algorithm in (Parunak, Purcell, and O'Connell 2002) performs the following steps: 1) The target and hazards in the environment are designated by a human operator at the GCS; and 2) The UAs in the swarm autonomously discover an optimal path to the target which avoids the hazards.

In (Ma, Duan, and Liu 2007), an ant colony algorithm for controlling a UA is discussed. The ant colony algorithm in (Ma, Duan, and Liu 2007) constructs a path the UA can follow to a target while avoiding stationary hazards (e.g. ground-based radars) in the environment. Since this is a combinatorial optimization problem, an ant colony algorithm (or other metaheuristic algorithms) can be used to solve it (Ma, Duan, and Liu 2007). An ant colony algorithm is essentially a positive feedback loop, where the useful behaviors of ants reinforce each other (Ma, Duan, and Liu 2007). Each stationary hazard is assumed in (Ma, Duan, and Liu 2007) to have an associated cost function. Thus, the task of the ant colony algorithm in (Ma, Duan, and Liu 2007) is to find a path that minimizes the cost functions associated with the stationary hazards in the environment. The results from running the ant colony algorithm in (Ma, Duan, and Liu 2007) in several simulated environments indicate an ant colony algorithm can efficiently find a path to a target that avoids stationary hazards in the environment. However, the ant colony algorithm in (Ma, Duan, and Liu 2007) is not applicable to UAs flying in environments with mobile hazards (i.e. dynamic operational environments).

Data Mining Algorithms

Data mining algorithms can be classified as analytical algorithms, supervised learning algorithms, or unsupervised learning algorithms (Painter, et al. 2006). Although there are many analytical algorithms used for data mining, linear regression and principal component analysis algorithms are two of the more popular choices. Supervised learning algorithms (i.e. classification algorithms) and unsupervised learning algorithms (i.e. clustering algorithms) are two important non-analytical approaches to data mining.

Analytical Algorithms

Analytical algorithms for data mining involve the application of statistical theory to data mining problems. In (Chen, et al. 2002), an algorithm for real-time multidimensional linear regression of stream data is discussed. Stream data are a type of dynamic data continuously produced in profuse quantities by some source (Chen, et al. 2002), e.g. a planetary orbiter. Due to their extremely high production rate, it is not feasible to archive stream data for offline data mining (Chen, et al. 2002). Thus, mining of stream data must occur in real time (Chen, et al. 2002). Furthermore, run-time efficiency is essential to any such algorithm for mining stream data (Chen, et al. 2002). The linear regression algorithm in (Chen, et al. 2002) uses a data cube model in order to conserve memory. A data cube has separate dimensions for each category in the data set (Chen, et al. 2002). The length of each dimension in the data cube is typically some statistical measure, such as the mean or variance (Chen, et al. 2002). Each dimension in the data cube represents, either directly or indirectly, some important feature in the stream data (Chen, et al. 2002). Thus, it is only necessary for the algorithm to store the data cube itself in memory, not the entire set of stream data (Chen, et al. 2002).

A linear regression algorithm searches for the globally optimal linear function for approximating the relationship between certain features in the data set (Chen, et al. 2002). A linear regression algorithm does this by minimizing the sum of squared errors between the estimated and actual values of the dependent variable (Chen, et al. 2002). This linear function is thus an approximation of the relationship between the dependent variable and the independent variables in the feature set (Chen, et al. 2002).

KMA is in the NP-hard time complexity class (Drineas, et al. 2004). Thus, the time complexity of KMA scales poorly as the size of the data set increases. Since KMA is an integer programming algorithm, however, it can be approximated with a more efficient linear programming algorithm by relaxing the constraints on the original combinatorial optimization problem (Fisher 1981). In (Drineas, et al. 2004), an approximation algorithm for KMA which uses linear programming relaxation is discussed. The approximation algorithm in (Drineas, et al. 2004) is in the Polynomial (P) time complexity class. The approximation algorithm in (Drineas, et al. 2004) uses singular value decomposition to find solutions which are, on average, half as optimal as solutions found by KMA for the same data set. If there are N data points, each with D dimensions, the data points can be partitioned into K clusters (Drineas, et al. 2004): 1) by constructing an N×D matrix containing the data points and 2) by finding the singular value decomposition of the N×D matrix. Each row in the N×D matrix corresponds to a single data point in the data set (Drineas, et al. 2004). Although the singular value decomposition algorithm in (Drineas, et al. 2004) typically finds less optimal solutions than KMA, it runs asymptotically faster. Thus, it can efficiently find approximate solutions to clustering problems involving massive data sets (Drineas, et al. 2004).

In (Lughofer 2008), a hybrid algorithm is discussed which improves the vector quantization algorithm. A vector quantization algorithm is a clustering algorithm which incrementally partitions a data set into K clusters (Lughofer 2008). First, K data points are selected from the data set to be the initial centroids for the clusters (Lughofer 2008). Then, the data points are incrementally processed in fixed-sized accretions (Lughofer 2008). For each unprocessed data point P in the current accretion, the following steps are

performed (Lughofer 2008): 1) Using a predetermined distance metric, the distance between the data point P and each of the K centroids is calculated; 2) The cluster C whose centroid is the minimum distance from the data point P is chosen; and 3) The centroid for cluster C is moved closer to the data point P by some fixed amount between 0 and 1. These steps are repeated until: a) All the data points in the data set have been processed; and b) No more significant movements of cluster centroids occur (Lughofer 2008).

The traditional vector quantization algorithm has several disadvantages (Lughofer 2008): 1) The same data points are scanned during every iteration; and 2) The number of clusters must be determined prior to starting the vector quantization algorithm. The hybrid algorithm in (Lughofer 2008) addresses these disadvantages by incorporating an Adaptive Resonance Theory (ART) neural network. When the hybrid algorithm in (Lughofer 2008) and similar clustering algorithms were tested, the hybrid algorithm produced more accurate results than the other clustering algorithms.

In (Parente, et al. 2011), the Vector Quantization Principal Component Analysis (VQPCA) algorithm is discussed. VQPCA is a hybrid algorithm for data mining experimental results from Moderate or Intense Low-oxygen Dilution (MILD) combustion (Parente, et al. 2011). Principal component analysis assumes a linear relationship between the variables in the data set (Parente, et al. 2011). Furthermore, the resulting principal components are often difficult to interpret due to their formulaic complexity (Parente, et al. 2011). The hybrid algorithm in (Parente, et al. 2011) tries to address these problems. The VQPCA hybrid algorithm combines principal component analysis and vector quantization algorithms (Parente, et al. 2011). First, the data set is partitioned into

clusters with vector quantization (Parente, et al. 2011). Then, each cluster is subjected to principal component analysis (Parente, et al. 2011). When the VQPCA hybrid algorithm was compared with a traditional principal component analysis algorithm, the VQPCA hybrid algorithm produced a more accurate characterization of the MILD combustion process (Parente, et al. 2011).

Supervised Learning Algorithms

Supervised learning is an appropriate technique when the classes are explicitly known prior to data mining, but the rules for classifying the data are unknown (Duda, Hart, and Stork 2001). In supervised learning algorithms, the task is to discover sets of classification rules from the labeled training data set (Duda, Hart, and Stork 2001). Using these rule sets, classifiers are constructed to efficiently and accurately classify similar, unlabeled data sets (Duda, Hart, and Stork 2001).

In (Agrawal, Imielinski, and Swami 1993), classification algorithms for databases which use decision trees are discussed. Classification algorithms partition data sets into separate classes using rule-based classifiers (Agrawal, Imielinski, and Swami 1993). The rule sets for the classifiers are discovered from the training data set (Agrawal, Imielinski, and Swami 1993). Thus, each rule in the rule set is supported by a certain number of data points in the training data set (Agrawal, Imielinski, and Swami 1993). The support for a given rule is a measure of its statistical significance (Agrawal, Imielinski, and Swami 1993). If the support for a rule does not exceed some predetermined threshold, the

classification algorithm discards it (Agrawal, Imielinski, and Swami 1993). Since the main task of a classification algorithm is rule discovery, it is vital for the algorithm to discover rules efficiently through effective use of disk and CPU resources (Agrawal, Imielinski, and Swami 1993).

In (Mehta, Agrawal, and Rissanen 1996), the Supervised Learning In Quest (SLIQ) algorithm is discussed. SLIQ is a classification algorithm scalable to massive data sets (Mehta, Agrawal, and Rissanen 1996). Unlike many classification algorithms, the SLIQ algorithm does not load the entire training data set into memory before constructing its classifiers (Mehta, Agrawal, and Rissanen 1996). This allows the SLIQ algorithm to construct classifiers for massive data sets for which the training data sets are too large to load into memory (Mehta, Agrawal, and Rissanen 1996). Large training data sets are desirable because they can be used to construct more accurate classifiers (Mehta, Agrawal, and Rissanen 1996).

The SLIQ algorithm was designed to construct decision tree classifiers (Mehta, Agrawal, and Rissanen 1996). Decision tree classifiers can be constructed quickly, and are easily translated into Structured Query Language (SQL) queries (Mehta, Agrawal, and Rissanen 1996). During tests run on publicly available data sets, the SLIQ classifiers had accuracy levels comparable to classifiers constructed by similar algorithms (Mehta, Agrawal, and Rissanen 1996). It was notable, however, that although the SLIQ classifiers were slower for the smallest data set tested, their run times were significantly faster for larger data sets than the classifiers constructed by other classification algorithms (Mehta, Agrawal, and Rissanen 1996).

In (Lewis 1998), naïve Bayes algorithms for classifying text are discussed. A naïve Bayes classifier uses Bayes' theorem on conditional probabilities to classify data points:

$$P(C \mid X) = \frac{P(C)P(X \mid C)}{P(X)}$$

First, the naïve Bayes classifier estimates the following probabilities from the training data set (Lewis 1998): 1) the unconditional probability of the data point belonging to class C, 2) the unconditional probability of the data point having feature set X, and 3) the conditional probability of the data point having feature set X (given it belongs to class C). After determining these three probabilities, the naïve Bayes classifier estimates a fourth probability—the conditional probability of the data point belonging to class C given it has feature set X (Lewis 1998). Finally, the estimated value for this fourth probability is used to classify the data point (Lewis 1998). Since there may be many such conditional probabilities for class C, each estimated from different feature sets, only the highest conditional probability for class C is used to classify the data points (Lewis 1998).

A naïve Bayes classifier uses Bayes' theorem to indirectly estimate the conditional probability of a data point belonging to class C, given it has feature set X, instead of the more difficult task of directly estimating this probability (Lewis 1998). A naïve Bayes classification algorithm provides a simple and efficient means for classifying text (Lewis 1998). However, other supervised learning algorithms often produce more accurate classifiers if provided with massive training data sets (Lewis 1998).

In (Boros, et al. 2000), the Logical Analysis of Data (LAD) algorithm is discussed. The LAD algorithm classifies data points as positive or negative results using sets of discovered rules (Boros, et al. 2000). The LAD algorithm discovers patterns in the

training data set which are associated with positive and negative results (Boros, et al. 2000). A positive pattern is a combination of features which is only found in data points associated with positive results (Boros, et al. 2000). A negative pattern is a combination of features which is only found in data points associated with negative results (Boros, et al. 2000). The LAD algorithm represents the combination of features for each pattern as a Boolean expression in first-order logic (Boros, et al. 2000). The Boolean expression for a pattern evaluates to 1 if a data point matches the pattern or 0 if it does not match (Boros, et al. 2000).

Patterns are discovered using either a top-down or bottom-up approach (Boros, et al. 2000). In the top-down approach, the Boolean expressions for patterns are discovered (Boros, et al. 2000): 1) by constructing Boolean expressions where each pattern feature is represented by a separate term and 2) by simplifying the resultant Boolean expressions by applying identities from first-order logic. In the bottom-up approach, the Boolean expressions for patterns are discovered (Boros, et al. 2000): 1) by enumerating Boolean expressions for every possible combination of features and 2) by discarding any Boolean expressions which do not result in the desired classification. Classifiers can then be constructed based on the patterns discovered in the training data set (Boros, et al. 2000).

In (Ruggieri 2002), the Efficient C4.5 (EC4.5) algorithm for classification is discussed. The EC4.5 classification algorithm is faster than the traditional C4.5 classification algorithm when classifying certain types of data sets (Ruggieri 2002). The C4.5 classification algorithm uses decision trees (constructed from the training data set) and information theory to determine the optimal classifications for unlabeled data points (Duda, Hart, and Stork 2001). Decision trees are an efficient means of classifying

nonmetric data sets (Duda, Hart, and Stork 2001). The traditional classification algorithm, which calculates the distance between scores for two data points (using some predetermined distance metric, e.g. Euclidean distance) to assess their similarity, is not applicable to nonmetric data sets (Duda, Hart, and Stork 2001). In nonmetric data sets, features for data points are qualitative attributes instead of quantitative measurements (Duda, Hart, and Stork 2001). For example, a data set pertaining to flowering plants might contain attributes such as whether the plant is annual or perennial, the color of its flowers, and its indigenous climate. Obviously, such features are nonmetric, and cannot be mapped to points in Euclidean space.

A decision tree algorithm classifies each unlabeled data point by applying a decision tree to it (Ruggieri 2002). The decision tree evaluates a single attribute at each interior node (Ruggieri 2002). The decision tree algorithm then picks a child node based on its evaluation of the data point's value for that attribute (Ruggieri 2002). These steps are repeated until a leaf node in the decision tree is reached (Ruggieri 2002). The data point is classified with the same label as that leaf node (Ruggieri 2002).

The splitting criterion for a decision tree algorithm is crucial, since it determines how decision trees will be constructed (Ruggieri 2002). In the C4.5 algorithm, the data points at each interior node are split based on whichever attribute will result in the maximum information gain ratio (Ruggieri 2002). Information gain (Kullback and Leibler 1951) is a measure of the difference in entropy between two probability distributions—the underlying probability distribution for the data set (which is assumed to be unknown in any data mining task) and another probability distribution which

estimates the underlying distribution. By maximizing the information gain ratio at each interior node, the C4.5 classification algorithm tries to insure that the children nodes are as dissimilar from each other as possible (Ruggieri 2002).

While efficient at accurately classifying unlabeled data points, the traditional C4.5 algorithm's approach to constructing decision trees from the training data set is inefficient at finding thresholds for continuous attributes (Ruggieri 2002). Interior nodes in a decision tree evaluate continuous attributes using predetermined thresholds (Ruggieri 2002). The traditional C4.5 algorithm is inefficient at constructing decision trees for such data sets because it uses a linear search to find the thresholds (Ruggieri 2002). The EC4.5 algorithm is more efficient at constructing decision trees for these types of data sets because it uses a binary search to find the thresholds (Ruggieri 2002). When comparative tests were run between the EC4.5 algorithm and the traditional C4.5 algorithm, the EC4.5 algorithm constructed decision trees in less time than the traditional C4.5 algorithm in most cases (Ruggieri 2002). Thus, the EC4.5 algorithm may be a better choice for classifying nonmetric data sets with continuous attributes (Ruggieri 2002).

In (Ivanciuc 2007), Support Vector Machine (SVM) algorithms are discussed. SVM algorithms are supervised learning algorithms which classify data points using a binary classification scheme (Ivanciuc 2007). First, the SVM algorithm constructs a classifier for the labeled training data set (Ivanciuc 2007). Each data point is assigned coordinates in a coordinate system (Ivanciuc 2007). The classifier is then constructed by finding the two hyperplanes that demarcate the boundaries between the two classes (Ivanciuc 2007). The data points which define those two hyperplanes are called the support vectors (Ivanciuc 2007). The SVM algorithm uses the constructed classifier to

classify similar, unlabeled data sets (Ivanciuc 2007). It is also possible to design an SVM algorithm for classifying data points which cannot be bounded by two hyperplanes (Ivanciuc 2007). In such an algorithm, a nonlinear function maps the coordinates assigned to the data points into one of the two bounded regions (Ivanciuc 2007).

In (Cortez, et al. 2009), the results of data mining a massive data set consisting of objective (i.e. physical and chemical) and subjective (e.g. taste) evaluations of various Portuguese wines are discussed. The same data set was mined by three different algorithms—a linear regression algorithm, a neural network algorithm, and an SVM algorithm (Cortez, et al. 2009). Each of the three algorithms was used to discover correlations between the objective properties of the wine (which can be accurately measured) and the subjective properties of the wine (which can only be determined by wine connoisseurs) (Cortez, et al. 2009). When the results from the three algorithms were compared, the SVM algorithm was found to be more accurate than the other two algorithms (Cortez, et al. 2009). Furthermore, while an SVM algorithm is guaranteed to eventually converge to the globally optimal solution, a neural network algorithm could converge to a solution that is only locally optimal (Cortez, et al. 2009).

In (Weinberger and Saul 2009), an improvement to the accuracy of the K-nearest neighbor algorithm for classification is discussed. The traditional K-nearest neighbor algorithm is improved in (Weinberger and Saul 2009) by using a different distance metric—a Mahalanobis distance metric. A K-nearest neighbor algorithm classifies unlabeled data points using a majority vote from the data point's K nearest neighbors in the test data set (Weinberger and Saul 2009). As an example, consider a test data set comprised of two classes of labeled data points, A and B. A K-nearest neighbor algorithm

would classify an unlabeled data point P (Weinberger and Saul 2009): 1) by finding the K data points from the test data set which were closest to P (according to some predetermined distance metric) and 2) by classifying P according to whichever of the two classes was found more frequently among the K data points. Obviously, an odd number should be used for K to avoid ties.

Like most data mining algorithms, the traditional K-nearest neighbor algorithm uses a Euclidean distance metric (Weinberger and Saul 2009). This typically involves mapping data points into a Euclidean space using a fixed number of features from the data points, i.e. the feature set (Weinberger and Saul 2009). Distance metrics are used in data mining algorithms to measure the similarity between any two data points (Weinberger and Saul 2009). However, for some data sets, a Euclidean distance metric may not be a sufficiently accurate measure of the similarity between two data points (Weinberger and Saul 2009). A common approach for such data sets is to use a distance metric learning algorithm to construct a better distance metric from the training data set (Weinberger and Saul 2009).

The distance metric learning algorithm in (Weinberger and Saul 2009) constructs a Mahalanobis distance metric from the training data set. A Mahalanobis distance metric minimizes distances between neighbors belonging to the same class and maximizes distances between neighbors belonging to different classes (Weinberger and Saul 2009). When compared with K-nearest neighbor algorithms which used other distance metrics, the Mahalanobis distance metric used in (Weinberger and Saul 2009) for their K-nearest

neighbor algorithms had better classification accuracy. However, Mahalanobis distance metrics have unacceptable run times when applied to data sets with large numbers of dimensions (Wu et al. 2010).

Unsupervised Learning Algorithms

Unsupervised learning is an appropriate technique when the classes for the data are not explicitly known prior to data mining, i.e. there is no labeled training data set (Duda, Hart, and Stork 2001). In unsupervised learning algorithms, the task is to classify the data into a predetermined number of classes based on some similarity metric, e.g. Euclidean distance (Duda, Hart, and Stork 2001). Applications of unsupervised learning techniques to data mining include clustering with self-organizing maps (Zhang, et al. 2009), the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm (Zhang, Ramakrishnan, and Levy 1996), wavelet-based clustering (Sheikholeslami, Chatterjee, and Zhang 2000), medical research (Cho, et al. 2010; Hybels, et al. 2009; van Rooden, et al. 2010; Jiang, Tang, and Zhang 2004; Chaussabel, et al. 2008), and atmospheric science research (Leckebusch, et al. 2008; Camargo, et al. 2007a; Camargo, et al. 2007b).

The first formal description of KMA is in (MacQueen 1967). KMA was designed to efficiently group data points into K clusters based on some similarity measure (MacQueen 1967). KMA tries to minimize the Total Within-Cluster Variation (TWCV) for those K clusters (MacQueen 1967). However, KMA often produces spurious results due to (Velmurugan and Santhanam 2010): a) its extreme sensitivity to initial conditions (i.e. the initial centroids selected for the K clusters), b) its sensitivity to outliers (e.g. a data point in a cluster whose score is much higher than the other data points in the cluster

will distort the centroid for the cluster), and c) its tendency to find solutions that are only locally optimal. Data mining algorithms have inductive biases which result in different types of algorithms favoring different types of solutions (Freitas 2002). KMA, in particular, has an inductive bias towards spherical-shaped clusters (Wagstaff, et al. 2001).

KMA is a non-hierarchical clustering algorithm which partitions its input data set into K clusters, where K is a predetermined constant. Since KMA cannot determine the optimal value for K itself, the optimal value of K for KMA must be estimated using some other technique. Conversely, hierarchical clustering algorithms produce cluster hierarchies, where the number of clusters is automatically determined by the algorithm itself. Applications of hierarchical clustering algorithms include agglomerative hierarchical clustering algorithms (Teh, Daumé, and Roy 2008; Chang, et al. 2010; Lai and Huang 2011) and divisive hierarchical clustering algorithms (Sorzano, et al. 2010; Kim and Billard 2011).

GKA is discussed in (Krishna and Murty 1999). By hybridizing a genetic algorithm with KMA, GKA is guaranteed to eventually converge to the globally optimal solution (Rudolph 1994). GKA does not directly address the cluster initialization problem (Krishna and Murty 1999). However, its use of a population of many candidate solutions (instead of the single candidate solution in KMA) results in reduced sensitivity to initial conditions.

In (Velmurugan and Santhanam 2010), KMA and the K-medoids algorithm are compared. The K-medoids algorithm, originally described in (Kaufman and Rousseeuw 1990), is a variation on KMA which addresses KMA's sensitivity to outliers (Velmurugan and Santhanam 2010). The K-medoids algorithm uses medoids, instead of

means, for the centroids for clusters (Velmurugan and Santhanam 2010). A medoid for a cluster is the data point which is the most similar to the other data points in the cluster according to the distance metric (Struyf, Hubert, and Rousseeuw 1997). Thus, unlike the mean, the medoid is always an actual data point in the cluster. The K-medoids algorithm is thus less sensitive to outliers (Velmurugan and Santhanam 2010). A hybrid algorithm which combines a genetic algorithm with the K-medoids algorithm is discussed in (Sheng and Liu 2006).

In (Pelleg and Moore 2000), the X-means algorithm is discussed. The X-means algorithm is an extension of KMA which automatically improves suboptimal choices for the number of clusters (Pelleg and Moore 2000). After each iteration of KMA, the Bayesian Information Criterion (BIC) is evaluated for each cluster to determine whether the cluster should be split into two subclusters in order to more accurately represent the naturally occurring clusters in the data (Pelleg and Moore 2000). Instead of requiring a single value for K to be selected beforehand, like KMA, the X-means algorithm only requires a range of possible values for K (Pelleg and Moore 2000). The BIC is evaluated for each of the possible values for K, and only the value for K which has the highest BIC is used for the final clusters (Pelleg and Moore 2000).

In (Chaturvedi, Green, and Carroll 2001), the K-modes algorithm is discussed. According to the theory of scale types in (Stevens 1946), there are only four scales (i.e. classes) of empirical measurements: nominal scale, ordinal scale, interval scale, and ratio scale. Whereas KMA was designed for clustering interval scale data and the K-medians algorithm for ordinal scale data, the K-modes algorithm is an adaptation of KMA for nominal scale data (Chaturvedi, Green, and Carroll 2001). Since KMA optimizes the sum

of squared errors for the data points, KMA is not applicable to nominal scale data (i.e. categorical data) (Chaturvedi, Green, and Carroll 2001). However, the K-modes algorithm optimizes the L_0 norm for data points instead, thus making it suitable for nominal scale data (Chaturvedi, Green, and Carroll 2001). To validate the clusters produced by the K-modes algorithm, (Chaturvedi, Green, and Carroll 2001) tested the K-modes algorithm on an artificial data set. The results indicate the validity of clusters produced by the K-modes algorithm is comparable to that of clusters produced by an equivalent algorithm (Chaturvedi, Green, and Carroll 2001). However, the mean run-time for the K-modes algorithm was significantly faster than the mean run-time for the other algorithm (Chaturvedi, Green, and Carroll 2001). The K-modes algorithm (like KMA) is susceptible to finding solutions that are only locally optimal (Chaturvedi, Green, and Carroll 2001). Also, unlike some algorithms, there is no obvious means of finding the optimal number of clusters to use for the K-modes algorithm (Chaturvedi, Green, and Carroll 2001).

In (Roy and Sharma 2010), the Genetic K-Modes (GKMODE) hybrid algorithm is discussed. The GKMODE hybrid algorithm combines a genetic algorithm with the K-modes algorithm discussed in (Chaturvedi, Green, and Carroll 2001). The GKMODE hybrid algorithm in (Roy and Sharma 2010) is intended to combine the global optimization capabilities of genetic algorithms (Rudolph 1994) with the run-time efficiency of the K-modes algorithm, much like GKA does with the K-means algorithm (Krishna and Murty 1999). In order to validate their results, (Roy and Sharma 2010) ran the GKMODE hybrid algorithm on publicly available data sets containing a mixture of numeric and categorical data. Then, the clusters produced by the GKMODE hybrid

algorithm were compared with the correct classes for those publicly available data sets (Roy and Sharma 2010). Since there was a significant amount of overlap between the two result sets, the clusters produced by the GKMODE hybrid algorithm were considered to be valid (Roy and Sharma 2010).

In (Lu, et al. 2004a), the Fast Genetic K-Means Algorithm (FGKA) is discussed. FGKA tries to improve the run-time efficiency of GKA with various techniques (Lu, et al. 2004a). For example, when invalid candidate solutions are generated, they are given the lowest possible fitness values (Lu, et al. 2004a). Thus, invalid candidate solutions will not be selected for reproduction, and will be eliminated from succeeding generations (Lu, et al. 2004a). In contrast, GKA explicitly scans for invalid candidate solutions, which increases its overhead (Lu, et al. 2004).

In (Lu, et al. 2004b), the Incremental Genetic K-Means Algorithm (IGKA) and Hybrid Genetic K-Means Algorithm (HGKA) are discussed. IGKA improves the run-time efficiency of FGKA (Lu, et al. 2004a) by incrementally updating clusters during each KMA iteration instead of reassigning all the data points with each iteration of KMA. However, FGKA outperforms IGKA for small numbers of iterations (Lu, et al. 2004b). Also, IGKA is more efficient than FGKA only when small mutation probabilities are used (Lu, et al. 2004b). HGKA uses a combination of both FGKA and IGKA to further increase run-time efficiency (Lu, et al. 2004b). HGKA starts by running FGKA on the data set, and then switches to running IGKA after the number of iterations exceeds some predetermined threshold (Lu, et al. 2004b).

In (Al-Shboul and Myaend 2009), the Genetic Algorithm Initialized K-Means (GAIK) hybrid algorithm is discussed. The GAIK algorithm is a hybrid algorithm which

addresses the cluster initialization problem of KMA (Al-Shboul and Myaend 2009). The GAIK hybrid algorithm uses a genetic algorithm to find initial centroids that are close to global extrema (Al-Shboul and Myaend 2009). Then, KMA is run with these optimized initial centroids (Al-Shboul and Myaend 2009). This reduces the extreme sensitivity of KMA to initial conditions (Al-Shboul and Myaend 2009). However, the hybridization of KMA with a genetic algorithm makes it more CPU-intensive (Al-Shboul and Myaend 2009). Also, the GAIK hybrid algorithm only uses a genetic algorithm for finding optimal initial centroids. The GAIK hybrid algorithm does not combine a genetic algorithm with KMA like GKA does (Krishna and Murty 1999; Al-Shboul and Myaend 2009). Thus, it is not guaranteed to converge to the globally optimal solution (Rudolph 1994).

In (Chander, Kumar, and Kumar 2011), the Partition-Based Genetic Algorithm Initialized K-Means (PGAIK) hybrid algorithm is discussed. Like the GAIK hybrid algorithm (Al-Shboul and Myaend 2009), the PGAIK algorithm is a hybrid algorithm which addresses the cluster initialization problem of KMA. The PGAIK hybrid algorithm uses a genetic algorithm to find optimal initial centroids, and then runs KMA using those optimized initial centroids (Chander, Kumar, and Kumar 2011). The PGAIK hybrid algorithm partitions the data set into K subsets (Chander, Kumar, and Kumar 2011). It then selects one initial centroid from each subset (Chander, Kumar, and Kumar 2011). This avoids the case where all the initial centroids are very close to each other, which produces a suboptimal distribution of data points among the clusters (Chander, Kumar, and Kumar 2011). The PGAIK hybrid algorithm was also shown in (Chander, Kumar, and Kumar 2011) to produce more compact clusters than the GAIK hybrid algorithm.

However, like the GAIK hybrid algorithm, the PGAIK hybrid algorithm in (Chander, Kumar, and Kumar 2011) is more CPU-intensive than KMA. Furthermore, since the PGAIK hybrid algorithm in (Chander, Kumar, and Kumar 2011) does not retain the most optimal candidate solution it discovers during its execution, it is not guaranteed to converge to the globally optimal solution (Rudolph 1994). A way to measure cluster validity, which in this case is considered to be directly related to cluster compactness, is also proposed in (Chander, Kumar, and Kumar 2011). Cluster compactness can be measured using a within-cluster scatter matrix (Chander, Kumar, and Kumar 2011).

In (Manning and Schütze 1999), the Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977; Moore 1999; Fraley and Raftery 2002; Plant and Böhm 2010) is discussed. The EM algorithm is similar to KMA (Manning and Schütze 1999; Plant and Böhm 2010), except the EM algorithm produces fuzzy clusters instead of crisp clusters. The EM algorithm assigns each data point an estimated probability of membership in each of the K clusters (Manning and Schütze 1999). Then, the EM algorithm iteratively improves these membership probability estimates until a locally or globally optimal solution is reached (Manning and Schütze 1999).

In (Kumar, Satoor, and Buck 2009), an extension of the EM algorithm for parallel execution on NVIDIA's Compute Unified Device Architecture (CUDA) is discussed. The run-time performance of the parallelized EM algorithm in (Kumar, Satoor, and Buck 2009) improved when the number of available Graphics Processing Units (GPUs) increased. The probability model used for the parallelized EM algorithm in (Kumar, Satoor, and Buck 2009) is a Gaussian mixture model. Since it was designed specifically for the CUDA architecture, the parallelized EM algorithm in (Kumar, Satoor, and Buck

2009) has the best performance when used for CPU-intensive applications (as opposed to I/O-intensive applications). The data parallelism of the algorithm in (Kumar, Satoor, and Buck 2009) was maximized when the clusters were small enough to fit in main memory. The largest data set the parallelized EM algorithm in (Kumar, Satoor, and Buck 2009) was tested on contained 230,400 data points. Some notable disadvantages of the parallelized EM algorithm in (Kumar, Satoor, and Buck 2009) were frequent memory conflicts between its threads, and the sensitivity of its performance to the number of threads that were used.

In (Feng and Wang 2011), a hybrid genetic algorithm (PGKM) is discussed which combines a genetic algorithm with a variant of KMA. Instead of requiring that the number of clusters that KMA will use to be known *a priori*, the PGKM hybrid algorithm uses a genetic algorithm to try to automatically determine the optimal number of clusters to use (Feng and Wang 2011). The cluster initialization problem is also addressed by finding initial centroids that are far apart from each other (according to the distance metric) and within areas of high density in the data set (Feng and Wang 2011). The PGKM hybrid algorithm in (Feng and Wang 2011) may be useful for data mining massive data sets where the optimal number of clusters is expected to be large. In such cases, iteratively testing different numbers of clusters with KMA is infeasible (Feng and Wang 2011). However, the optimal number of clusters found by the PGKM hybrid algorithm could be inaccurate in such cases, e.g. the PGKM hybrid algorithm finds a number of clusters that is only locally optimal, not globally optimal (Feng and Wang 2011).

In (Handl, Knowles, and Kell 2005), techniques for cluster validation are discussed. The clusters produced by a clustering algorithm can be validated by demonstrating the clusters correspond to meaningful patterns in the data set, i.e. it is extremely unlikely the clusters were produced by chance (Handl, Knowles, and Kell 2005). Cluster validation is necessary to insure the clusters produced by a clustering algorithm are semantically valid (Handl, Knowles, and Kell 2005). Clusters produced by different clustering algorithms may exhibit different qualities, such as compactness or connectedness (Handl, Knowles, and Kell 2005). Compact clusters are defined in (Handl, Knowles, and Kell 2005) as clusters produced with minimal TWCV. Connected clusters are defined in (Handl, Knowles, and Kell 2005) as clusters produced by grouping together data points in the same neighborhood as each other. KMA searches for the most compact clusters for a data set, while density-based algorithms such as the Density-Based Scan (DBSCAN) algorithm (Ester, et al. 1998) search for the most connected clusters (Handl, Knowles, and Kell 2005).

Clusters can be validated either internally or externally (Handl, Knowles, and Kell 2005). Techniques for internal validation of clusters include stability validation techniques which measure the consistency of clusters produced by a clustering algorithm that is iteratively applied to similar data sets (Handl, Knowles, and Kell 2005). Unlike other techniques for internal validation of clusters, stability validation techniques are not biased towards any particular clustering algorithm (Handl, Knowles, and Kell 2005). However, like any technique for internal validation of clusters, stability validation techniques cannot distinguish between locally and globally optimal clusters produced by a clustering algorithm (Handl, Knowles, and Kell 2005). Techniques for external

validation of clusters include comparing them with a similar, labeled data set (Handl, Knowles, and Kell 2005). However, if no labeled data set exists for a particular type of data set, then some other internal or external validation technique must be used (Handl, Knowles, and Kell 2005).

In (Halkidi, Batistakis, and Vazirgiannis 2001), techniques for cluster validation are also discussed. An important technique for external validation of clusters is testing the statistical significance of the clusters produced by the clustering algorithm (Halkidi, Batistakis, and Vazirgiannis 2001). Clusters are tested for statistical significance by showing that the probability of the clustering algorithm producing the clusters by chance does not exceed some predetermined significance level, e.g. a significance level of 5% (Halkidi, Batistakis, and Vazirgiannis 2001).

Different criteria exist for assessing cluster optimality (Halkidi, Batistakis, and Vazirgiannis 2001). The criteria suggested in (Berry and Linoff 1996) are compactness and separation (Halkidi, Batistakis, and Vazirgiannis 2001). A clustering algorithm can ensure cluster compactness by minimizing the TWCV (Halkidi, Batistakis, and Vazirgiannis 2001). Cluster separation can be ensured by various techniques, including maximizing the distance between the centroids for the clusters (Halkidi, Batistakis, and Vazirgiannis 2001). Since the problem of simultaneously minimizing the TWCV (i.e. ensuring cluster compactness) and maximizing the distance between the centroids for the clusters (i.e. ensuring cluster separation) is likely to be intractable, clustering algorithms typically will only optimize one of these criteria.

The normalized Hubert statistic (Γ) is useful for validating cluster compactness (Halkidi, Batistakis, and Vazirgiannis 2001):

$$\Gamma = \frac{1}{M}\sum_{i=0}^{N-1}\sum_{j=i+1}^{N}\frac{(P(i,j)-\mu_p)(Q(i,j)-\mu_q)}{\sigma_p^2\sigma_q^2}$$

where $M = \frac{N(N-1)}{2}$, P is a matrix where element (i,j) is the Euclidean distance between data points i and j, Q is a matrix where element (i,j) is the Euclidean distance between the centroids of the clusters containing data points i and j, μ_p and μ_q are the means of the P and Q matrices, respectively, and σ_p^2 and σ_q^2 are the variances of the P and Q matrices, respectively. A cluster can be considered to be compact if it has a large normalized Hubert statistic (Halkidi, Batistakis, and Vazirgiannis 2001). Furthermore, the optimal number of clusters to use for a specific data set can be determined by finding the number of clusters which maximizes the value for the normalized Hubert statistic (Halkidi, Batistakis, and Vazirgiannis 2001).

Mining Vehicle Telemetry Data

Numerous algorithms have been developed for mining telemetry data from land, sea, and air vehicles. These data mining algorithms for telemetry data sets can be classified as either descriptive or predictive algorithms according to the models the algorithms use. Applications of descriptive algorithms include discovering patterns in animal migrations and automobile traffic (Li, et al. 2010) and coastal surveillance (Dahlbom and Niklasson 2007). Applications of predictive algorithms include improving commercial airline safety (McFadden and Towell 1999; Callantine 2001) and improving the safety of aircraft in the U.S. Navy (Haas, Walker, and Kough 2008).

In (Zhang, Zhang, and Hu 2007), a feature extraction algorithm for classifiers for operational data sets obtained from military aircraft is discussed. A feature extraction

algorithm searches for the smallest possible feature set necessary for data mining (Zhang, Zhang, and Hu 2007). The feature extraction algorithm in (Zhang, Zhang, and Hu 2007) extracts relevant features from the data in two phases: 1) performing an Artificial Neural Network Weight Analysis (ANNWA) of a multilayer neural network trained on the data set and 2) applying a genetic algorithm to find the optimal feature set for constructing a classifier for the data set. The weights in a multilayer neural network can be considered to be a ranking of the relevance of data points in the training data set (Zhang, Zhang, and Hu 2007). If a particular data point (i.e. feature) in the training data set results in a strongly weighted connection in the multilayer neural network, then that data point is likely to be highly relevant to the output signal from the multilayer neural network (Zhang, Zhang, and Hu 2007).

ANNWA was performed prior to running the genetic algorithm because the time complexity of ANNWA is considerably less than the time complexity of a genetic algorithm (Zhang, Zhang, and Hu 2007). Thus, by first reducing the possible feature set with ANNWA, the total number of features the genetic algorithm had to operate on was significantly reduced (Zhang, Zhang, and Hu 2007). Two data sets were used to test the feature extraction algorithm in (Zhang, Zhang, and Hu 2007). Both of the data sets used in (Zhang, Zhang, and Hu 2007) consisted of engine performance data for military aircraft (e.g. the oil pressure for the turbine engines). The results from testing the feature extraction algorithm in (Zhang, Zhang, and Hu 2007) suggest that smaller feature sets produce more accurate classifiers.

Descriptive Algorithms

Descriptive algorithms for mining vehicle telemetry data try to discover patterns in the data sets which can be used to describe vehicle movements. A three-step process for fuzzy clustering of trajectories is discussed in (Pelekis, et al. 2011). A trajectory is the chronologically-ordered sequence of the positions of a moving object, e.g. the complete flight path of an aircraft. A trajectory has a fixed starting position (at time zero) and ending position. A trajectory clustering algorithm measures the similarity between two trajectories based on the proximity of the objects to each other during similar time frames (Pelekis, et al. 2011).

Unlike other trajectory clustering algorithms, the three step process in (Pelekis, et al. 2011) corrects for uncertainty in the trajectory data. A trajectory database stores the discrete positions of an object at varying times during its trajectory (Pelekis, et al. 2011). Various types of uncertainty may be present in a trajectory database, for instance, the small amount of positional uncertainty intrinsic to any GPS-based data set (Pelekis, et al. 2011). The first algorithm in (Pelekis, et al. 2011) preprocesses the trajectory database into a more suitable format for trajectory clustering. The preprocessing algorithm in (Pelekis, et al. 2011) segments the discrete positions for the trajectory using time intervals with some fixed duration D. After the preprocessing, each interval of the trajectory with duration D is represented by a single data point (Pelekis, et al. 2011). Thus, each original trajectory is preprocessed into an approximate form for processing by the next two algorithms (Pelekis, et al. 2011). For example, an object may have been contained in some geographic region R for 10 minutes during its trajectory.

The second algorithm discussed in (Pelekis, et al. 2011), called the CenTra algorithm, determines the centroid trajectories for each of the clusters. The third algorithm discussed in (Pelekis, et al. 2011), called the Time-Relaxed CenTra (TX-CenTra) algorithm, performs fuzzy clustering of the preprocessed data points based on their distances from the centroid trajectories. The TX-CenTra algorithm merges reoccurring chronological sequences of successive data points in the trajectories (Pelekis, et al. 2011). Thus, the results from the TX-CenTra algorithm are simplified and easier to interpret (Pelekis, et al. 2011). Although the three step process of trajectory clustering in (Pelekis, et al. 2011) had acceptable run-time performance during testing, it also exhibited high sensitivity to initial conditions.

Predictive Algorithms

Predictive algorithms for mining vehicle telemetry data try to discover probabilistic models from the data sets which can be used to predict vehicle movements. In (McCall, et al. 2007), an algorithm for predicting the behavior of automobile drivers is discussed. The predictive algorithm in (McCall, et al. 2007) uses computer vision algorithms to: a) detect the automobile's position within the traffic lane and b) detect changes in the driver's lateral head motion. Furthermore, the predictive algorithm in (McCall, et al. 2007) uses data from the automobile's internal sensors to determine its velocity. To predict the automobile's future path, a Kalman filter (Kalman 1960) is applied to its reported velocity and its estimated position within the traffic lane (McCall, et al. 2007). Sparse Bayesian learning (Tipping 2001) is used to predict whether the driver intends to initiate a lane change based on changes in the driver's head motion and the automobile's estimated position within the traffic lane (McCall, et al. 2007). The

reliability of the predictive algorithm in (McCall, et al. 2007) was acceptable when tested in scenarios which were similar to the scenarios present in the training data set. However, the predictive algorithm in (McCall, et al. 2007) was less reliable in scenarios which differed significantly from those present in the training data set.

In (Taniar and Goh 2007), several data mining algorithms are discussed which could be used to discover movement patterns of GPS-enabled mobile device users (e.g. smart phone users). First, the GPS positions of the mobile device users are sampled at some constant rate and stored in a database (Taniar and Goh 2007). Then, the database is normalized by discarding irrelevant data, e.g. errors in the positions reported by the GPS receivers (Taniar and Goh 2007). If the mobile device user stayed near a particular GPS position for some predetermined duration, then that GPS position is considered to be significant (Taniar and Goh 2007). For example, the mobile device user may have stopped at a restaurant for an hour. A movement pattern of a mobile device user is defined in (Taniar and Goh 2007) as a path which starts and ends at significant GPS positions. These significant GPS positions in a mobile device user's path are correlated to nearby Locations Of Interest (LOI), e.g. a department store, that were assumed to have been interesting to the mobile device user (Taniar and Goh 2007).

The support for a movement pattern is defined in (Taniar and Goh 2007) as the frequency at which the movement pattern occurs in the database. Furthermore, the confidence in the significance of the movement pattern is defined in (Taniar and Goh 2007) as the relative frequency of the movement pattern in the database with respect to similar movement patterns (Taniar and Goh 2007). If the support and/or confidence for a movement pattern in the database do not exceed certain predetermined thresholds, the

movement pattern is not considered to be significant (Taniar and Goh 2007). Any movement patterns discovered in the database which do not meet the minimum criteria for significance (i.e. exceed the minimum thresholds for support and confidence) are excluded from the output of the data mining algorithms (Taniar and Goh 2007).

The data mining algorithms in (Taniar and Goh 2007) were tested on three artificial data sets constructed by hand with Microsoft Excel (Taniar and Goh 2007). The results from the tests indicate the time complexities of the data mining algorithms in (Taniar and Goh 2007) grow exponentially as the size of the data sets (i.e. the number of data points) increases. Since the data mining algorithms in (Taniar and Goh 2007) were only tested with artificial data sets, their accuracy at discovering movement patterns in real data sets could not be verified.

In (Maedar, Morari, and Baumgartner 2011), an algorithm for predicting maneuvers of GA aircraft is discussed. The predictive algorithm discussed in (Maedar, Morari, and Baumgartner 2011) was implemented as part of the FLARM collision avoidance system. FLARM (Flarm Technology 2010) is a cooperative collision avoidance system (i.e. its collision avoidance algorithm is dependent on communication with FLARM devices in other GA aircraft) designed for use by GA aircraft (Maedar, Morari, and Baumgartner 2011). Using its GPS receiver to determine the GA aircraft's position and velocity, the onboard FLARM device estimates its intended flight path (Maedar, Morari, and Baumgartner 2011). The onboard FLARM device then wirelessly transmits the estimated flight path for its GA aircraft to any nearby GA aircraft which

may also be equipped with FLARM devices (Maedar, Morari, and Baumgartner 2011). If a potential conflict is detected, the onboard FLARM device warns the pilot of the possibility of a midair collision (Maedar, Morari, and Baumgartner 2011).

The predictive algorithm in (Maedar, Morari, and Baumgartner 2011) has three steps: 1) estimating the current state of the GA aircraft, 2) attempting to classify the current maneuver being performed by the pilot (e.g. turning or straight and level flight), and 3) predicting the future flight path of the GA aircraft. During the first step, an Interacting Multiple Model (IMM) algorithm based on an Extended Kalman Filter (EKF) is used to estimate the current state of the GA aircraft using historical GPS data and an estimation of current wind conditions (Maedar, Morari, and Baumgartner 2011). During the second step, the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) attempts to classify the current maneuver using a static classification scheme based on observation of common maneuvers performed by GA pilots. Finally, during the third step, the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) estimates the flight path of the GA aircraft during the next 20 seconds based on the estimate of its current state (from the first step) and the estimate of the current maneuver the pilot is performing (from the second step).

Although the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) is based on traditional techniques for predictive modeling of nonlinear systems, such as EKF, the algorithm uses a static classification scheme based on observation of common maneuvers performed by pilots of GA aircraft. Thus, their classification scheme may not include unusual maneuvers that pilots of GA aircraft may occasionally perform, which could be discovered by mining massive aircraft telemetry data sets. Furthermore, the

static classification scheme used by the algorithm in (Maedar, Morari, and Baumgartner 2011) cannot detect ascending and descending maneuvers, e.g. a descending right turn. The curvature of the Earth is also not considered by the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) when transforming the geographic coordinates provided by GPS into a Cartesian coordinate system. However, the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) tries to estimate the current wind conditions in the GA aircraft's environment, which can significantly impact maneuvers by GA aircraft at lower speeds, e.g. 80 knots.

To test their predictive algorithm, (Maedar, Morari, and Baumgartner 2011) used a synthetic data set. The results in (Maedar, Morari, and Baumgartner 2011) indicate: a) Their algorithm accurately predicted the turn rate for pilot maneuvers, with a maximum estimation error of about 7°; and b) Their algorithm accurately predicted the speed of the GA aircraft when estimates of the current wind conditions were included. Thus, the predictive algorithm in (Maedar, Morari, and Baumgartner 2011) can accurately predict some of the more common maneuvers performed by pilots of GA aircraft in level flight.

CHAPTER III

METHODOLOGY

The Aircraft Data Miner (ADM) was developed to data mine ADS-B, and later FDM, data. ADM was implemented with the C++ language in the Linux operating system environment. ADM was used to mine a large FDM data set to discover probabilistic models of pilot behavior as a function of the aircraft's performance (e.g. a Cessna 172), altitude, and proximity to the nearest uncontrolled airport. The FDM data were obtained exclusively from the University of North Dakota's training fleet. Thus, the maneuvers performed by those student pilots are only likely to be used in a training environment. The behavior of the pilot of a GA aircraft flying under VFR in Class E airspace may have also been influenced by hazardous conditions in the operational environment, such as a mechanical failure in the aircraft. However, such hazardous conditions in the operational environmental only occur rarely, and their consideration is beyond the scope of this research.

FDM data obtained from the Garmin G1000 are also typically stored in Comma-Separated Value (CSV) files for later analysis. The data sets mined with ADM consist of a large number of flat files (in the CSV format) containing the raw data from FDM data archived by Garmin G1000 units from many different aircraft over an extended period of time. Those FDM data are stored in the flat files in the same chronological order that the data were logged by the Garmin G1000. Many of the data contained in FDM data sets are not relevant to this analysis. Also, the data streams logged by Garmin G1000 units are

time-ordered sequences of discrete 3D GPS positions occupied by FDM-capable aircraft, whereas the continuous flight paths of the aircraft (rather than the discrete positions along those flight paths) are more important to the analysis of pilot maneuvers. Thus, it is necessary to extensively preprocess the raw FDM data into a more useful format prior to data mining.

Data Preprocessing

ADM performs all of its data preprocessing via SQL commands that operate on tables in a relational database, as recommended in (Segal 2010). Since the data preprocessing algorithms are I/O-intensive, proper caching of tables in the data preprocessing database is crucial to the performance of the data preprocessing algorithms. ADM performs five phases of data preprocessing on the FDM data for each performance class in the data set (see figure 1): 1) extracting the FDM data from the flat files and importing the relevant data into a relational database, 2) normalizing the data in the database, 3) constructing normalized flight paths from the discrete 3D GPS positions in the database, 4) discovering digital pheromone trails by finding subpaths which are common to multiple normalized flight paths, and 5) dynamically segmenting the normalized flight paths into subpaths using those digital pheromone trails. These five phases of data preprocessing must be completed prior to data mining.

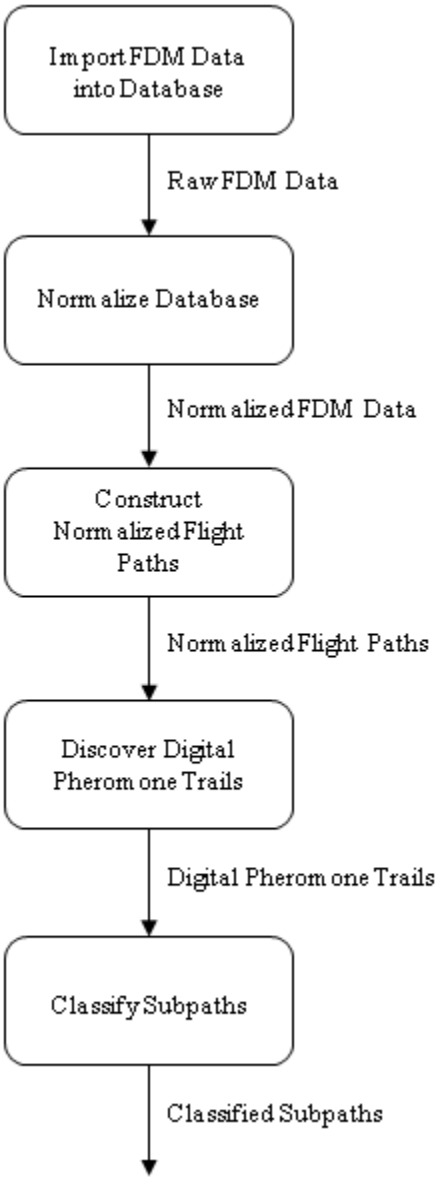

Figure 1. The flowchart for the five phases of data preprocessing.

Three of the phases of data preprocessing have their own data models—the normalized flight path construction phase, the digital pheromone trail discovery phase, and the subpath classification phase. The aircraft telemetry data import phase and the normalization phase are the only phases which share the same data model. See figure 2 for the data structure diagram (DeMarco 1979) of the data preprocessing phases.

The output data tables from a given data preprocessing phase are the input data tables for the following phase. All tables used during the data preprocessing phases are stored in the same relational database. During the aircraft telemetry data import phase, FDM data are extracted from flat files and imported into tables in the relational database. This is the only phase of data preprocessing which operates on flat files. The remaining phases operate exclusively on tables in the relational database. Each table used during data preprocessing contains data for only one performance class. Thus, as FDM data are extracted from the flat files, each datum is imported into the corresponding table for its performance class. A special table in the data preprocessing database, the aircraft data index, specifies which performance classes have been imported into the database, as well as other database metadata. This database schema facilitates the later data mining phases, where the subpaths for each performance class are data mined separately.

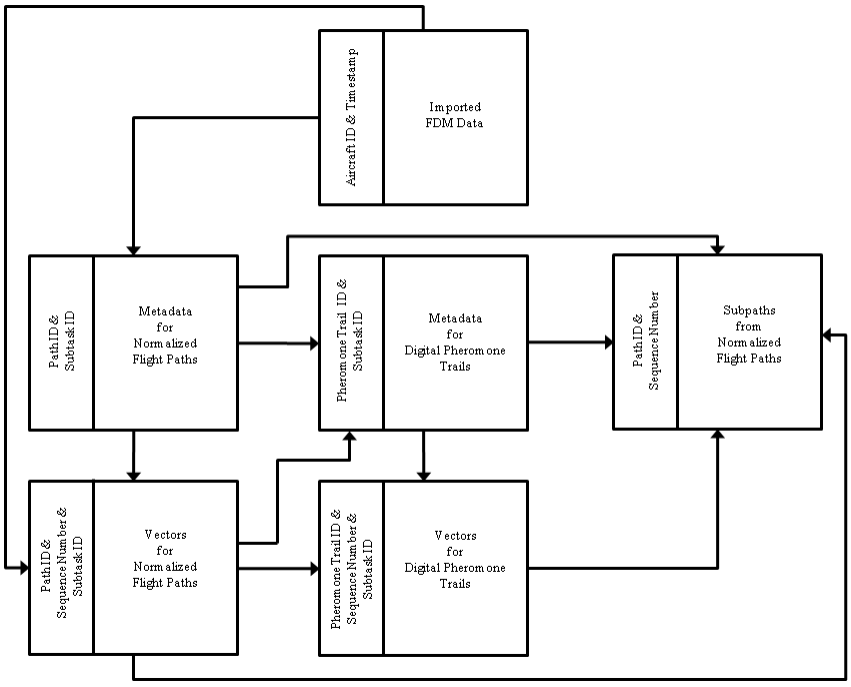

Figure 2. The data structure diagram for the data preprocessing database.

The normalized flight path construction phase constructs normalized flight paths from the discrete 3D aircraft positions stored in the tables from the previous phases. These normalized flight paths are jointly specified by two types of data models—the path metadata model and the path vector data model. The path metadata model specifies general information about the normalized flight paths, such as the unique identifiers for the paths and the lengths of the paths. The path vector data model specifies the ordered 3D vectors of which the paths are composed. Each path vector has an associated identifier and sequence number which indicates which path it pertains to and its position within that path, respectively. The starting latitude, longitude, and altitude for the path vectors, as well as their yaw (i.e. heading) and pitch (i.e. ascent) angles, are also specified by the path vector data model.

The digital pheromone trail discovery phase discovers digital pheromone trails (i.e. subpaths which are common to multiple normalized flight paths) using the path metadata and path vector tables generated during the previous phase. The discovered digital pheromone trails are jointly specified by two types of data models—the digital pheromone trail metadata model and the digital pheromone trail vector model. The digital pheromone trail metadata model specifies information about the digital pheromone trails themselves, such as the unique identifiers and the strengths of the digital pheromone trails. The digital pheromone trail vector data model specifies the ordered 3D vectors of which the digital pheromone trails are composed. Each digital pheromone trail vector has an associated identifier and sequence number which indicates which digital pheromone trail it pertains to and its position within that digital pheromone trail, respectively. Unlike the path vector data model, however, the digital pheromone trail vector model only

specifies the yaw and pitch angles of the digital pheromone trail vectors. It does not specify the starting latitude, longitude, or altitude. Thus, every digital pheromone trail shares the same start point in the digital pheromone trail vector data model. The start point for each successive vector in the digital pheromone trail is the end point for the previous vector.

The subpath classification phase segments the normalized flight paths (specified by the path metadata and path vector tables generated during the third phase) into subpaths using the digital pheromone trail metadata and digital pheromone trail vector tables generated during the previous phase. The subpaths are specified by a single data model. The subpath data model specifies information about each subpath, such as the normalized flight path it pertains to, its sequence number, and—if it has a matching digital pheromone trail in the relational database—the unique identifier for that digital pheromone trail, i.e. the digital pheromone trail which is the best classifier for the subpath.

Constructing Normalized Flight Paths

Every data preprocessing phase except the subpath classification phase operates incrementally. The input tables to two of these phases—the normalization and normalized flight path construction phase—have special fields which indicate whether a specific row has been processed by a specific phase. Thus, those two phases only need to process those rows which have not already been processed, instead of completely processing both the rows for the new and the old data whenever any new data are imported into the relational database.

The first phase of data preprocessing (see figure 3) involves reading the raw FDM data obtained from the Garmin G1000 units, discarding irrelevant data, insuring each aircraft has a unique numeric identifier so it is easily tracked, and importing the relevant data from each aircraft into the relational database. Data for any flight segments outside of Class E airspace are discarded. FDM data do not specify whether aircraft are on the ground or airborne. Thus, it is not possible to insure that all FDM data from aircraft on the ground are completely excluded from the automated analysis. ADM retains the following fields from FDM data: a) the aircraft's latitude, longitude, and altitude, b) the aircraft's heading, c) the aircraft's horizontal velocity, d) the aircraft's ascent angle, and e) the time of reception for the datum. The aircraft's tail number and performance class are specified manually when the FDM data are archived.

Input: Set *D* of aircraft telemetry data files.
Output: Set *R* of rows in the data preprocessing database containing aircraft telemetry data.

for each file *F* **in** *D* **do**
  for each data point *P* **in** *F* **do**
    if not *has_missing_features(P)* **then**
      if $P.altitude < 18000$ **then**
        $S \leftarrow lookup\_table\_for(P.performance\_class)$
        Insert row for *P* into table *S*.
Mark any rows for normalized paths for corresponding performance class as incomplete.

Figure 3. The algorithm for importing aircraft telemetry data into the data preprocessing database.

Since integer-based algorithms are typically faster than equivalent string-based algorithms, a unique identifier is generated for each aircraft in the data set which is based on the aircraft's tail number. First, each digit in the aircraft's tail number is replaced with its two digit representation, so a "0" is replaced with "00", a "1" with "01", and so forth. Then, each letter in the tail number is also replaced with a two digit representation—"A"

is replaced with “10”, “B” is replaced with “11”, and so forth. Finally, a “1” is inserted at the beginning of the digit string to create the unique identifier. For example, using this algorithm the tail number N1657U would be mapped to the unique identifier 1230106050730.

The second phase (see figure 4) involves normalizing the data in the database. Redundant data, e.g. an aircraft maintaining the same GPS position for several seconds, and data for all flight segments outside of the U.S. are discarded. The specific volume of controlled airspace around controlled airports varies from airport to airport. However, the only publicly available data which specify these specific controlled airspace volumes around controlled airports is only available in the Portable Document Format (PDF), which is not amenable to the automatic processing required by data mining algorithms. The maximum volume of controlled airspace around any airport in the NAS is 27,780 meters horizontally and 3,000 meters vertically above MSL. This is the maximum possible volume for Class B airspace. ADM guarantees the data points used to reconstruct the normalized flight paths are not inside controlled airspace around any controlled airports by discarding any data points from aircraft within this maximum volume of controlled airspace around any controlled airport.

```
Input: Set R of unprocessed rows in the data preprocessing database containing aircraft telemetry data.
Output: Set R′ of normalized rows in the data preprocessing database containing aircraft telemetry data.

Delete rows in R for data points outside of the NAS.
for each aircraft A in R
    N ← number of data points for A
    for i = 1 to N
        if is_possibly_near_controlled_airport(A[i]) then
            Delete row for A[i].
        else
            if i > 1 and A[i].id ≠ A[i-1].id then
                Delete all rows for A in R.
                goto end_of_outer_loop
            if A[i].ascent_angle = null then
                A[i].ascent_angle ← tan^-1 (A[i].horizontal_velocity, A[i].vertical_velocity)
            if i > 1 and A[i-1].heading = null then
                if A[i].latitude ≠ A[i-1].latitude or A[i].longitude ≠ A[i-1].longitude then
                    A[i-1].heading ← tan^-1 (A[i].longitude, A[i].latitude)
                else
                    Delete row for A[i-1].
            if i > 1 and A[i-1].heading = null then
                Delete row for A[i-1].
    label end_of_outer_loop
Mark rows in R′ as completely processed by the normalization phase.
```

Figure 4. The algorithm for normalizing the aircraft telemetry data in the data preprocessing database.

The third phase (see figure 5) involves constructing vectors from the discrete 3D GPS positions for aircraft in the database. Consecutive data points with the same heading and ascent angles (i.e. the yaw and pitch angles, respectively) are merged to form the longest possible vectors. The magnitude of these vectors is measured in time, not distance, because while aircraft can fly at different speeds (and thus cover different distances in the same amount of time), it is reasonable to assume that the pilots of those aircraft require about the same amount of time to perform the same types of maneuvers.

There may have been other factors affecting the aircraft's heading, ascent angle, and altitude than just the pilot's control inputs, e.g. air turbulence. Also, the uncertainties in the aircraft's horizontal and vertical position (inherent to GPS-based telemetry devices such as the Garmin G1000) introduce a measurable amount of error. To correct for these anomalies, ADM rounds up the aircraft's heading and ascent angles to the nearest

multiple of 2°, and the aircraft's altitude to the nearest multiple of 2 meters. Thus, the values for the heading, ascent angle, and altitude of the aircraft are considered to be accurate indicators of the pilot's intent to within ±1° for headings and ascent angles, and to within ±1 meter for altitudes.

Input: Set R' of unprocessed rows in the data preprocessing database containing normalized aircraft telemetry data.
Output: Set P of rows in the data preprocessing database containing normalized flight paths.

for each aircraft A **in** R'
  $N \leftarrow$ number of data points for A
  $p \leftarrow$ new path
  for $i = 1$ **to** N
    if *length*(p) = 0 **then**
      $normalized\_heading \leftarrow 90$
      $\Delta \leftarrow 90 - A[i].heading$
      $v \leftarrow$ vector with $heading = normalized\_heading$, $ascent\_angle = A[i].ascent\_angle$, $magnitude = 0$
    else if $A[i].timestamp - A[i\text{-}1].timestamp < 300$ seconds **then**
      if $A[i].heading = A[i\text{-}1].heading$ **then**
        $normalized\_heading \leftarrow 90$
        $\Delta \leftarrow 90 - A[i].heading$
      else
        $normalized\_heading \leftarrow A[i].heading + \Delta$
      if $A[i].heading = A[i\text{-}1].heading$ **and** $A[i].ascent\_angle = A[i\text{-}1].ascent\_angle$ **then**
        $v.magnitude \leftarrow v.magnitude + A[i].timestamp - A[i\text{-}1].timestamp$
      else
        Insert v into p.
        $v \leftarrow$ vector with $heading = normalized\_heading$, $ascent\_angle = A[i].ascent\_angle$, $magnitude = 0$
    else
      Insert v into p.
      $S \leftarrow lookup\_tables\_for(A.performance\_class)$
      Insert vectors for p into *S.vector_table*.
      Insert metadata for p into *S.metadata_table*.
      $p \leftarrow$ new path
  if $A[i].timestamp - A[i\text{-}1].timestamp < 300$ seconds **then**
    if $A[i].heading = A[i\text{-}1].heading$ **then**
      $normalized\_heading \leftarrow 90$
      $\Delta \leftarrow 90 - A[i].heading$
    else
      $normalized\_heading \leftarrow A[i].heading + \Delta$
    if $A[i].heading = A[i\text{-}1].heading$ **and** $A[i].ascent\_angle = A[i\text{-}1].ascent\_angle$ **then**
      $v.magnitude \leftarrow v.magnitude + A[i].timestamp - A[i\text{-}1].timestamp$
    else
      Insert v into p.
      $v \leftarrow$ vector with $heading = normalized\_heading$, $ascent\_angle = A[i].ascent\_angle$, $magnitude = 0$
  else
    Insert v into p.
    $S \leftarrow lookup\_tables\_for(A.performance\_class)$
    Insert rows for vectors in p into *S.vector_table*.
    Insert row for metadata of p into *S.metadata_table*.
    $p \leftarrow$ new path
Age any digital pheromone trails in the database by an amount proportional to *size(R')*.
Mark rows for digital pheromone trails for corresponding performance classes as incomplete.
Mark rows in R' as completely processed by the normalized flight path construction phase.

Figure 5. The algorithm for constructing normalized flight paths from the aircraft telemetry data in the data preprocessing database.

Since an aircraft's heading is represented as a compass direction in FDM data, it is necessary during this phase to normalize the aircraft headings obtained from the FDM data. The headings are normalized with respect to the heading currently considered the straight-flying direction for the aircraft. If the aircraft flew with the same heading H for two or more consecutive data points, then H would be considered its current straight-flying direction. Thus, to normalize the straight-flying heading H to an angle of 90°, it is rotated by 90-H degrees. Likewise, all other aircraft headings are rotated by the same amount until the straight-flying direction changes. If the angles of ascent for an aircraft were unavailable (e.g. in ADS-B data), these values could be calculated from the aircraft's horizontal and vertical velocities.

Also, if there is a time difference of more than 300 seconds between two consecutive data points for an aircraft, then these data points are considered to belong to separate normalized flight paths. Thus, the second data point will be used to start a new normalized flight path. Once all the normalized flight paths have been constructed from the discrete 3D positions of the aircraft, any existing digital pheromone trails in the relational database are aged by an amount S which is proportional to the size of the input data set for the phase. The digital pheromone trails in the relational database are aged by: 1) subtracting S from the digital pheromone strength for each of the digital pheromone trails and 2) deleting any digital pheromone trails which, as a result, have a digital pheromone strength which is no longer greater than zero.

Discovering Digital Pheromone Trails

The fourth phase (see figure 6) involves the discovery of digital pheromone trails in the normalized flight paths for the aircraft using an ant colony algorithm. In order to dynamically discover classes of maneuvers frequently performed by pilots of GA aircraft, each normalized flight path is considered a separate digital pheromone trail deposited by the aircraft. If every normalized flight path is compared with every other normalized flight path, then subpaths can be discovered that are common to multiple paths. These common subpaths are the areas where digital pheromones from normalized flight paths are reinforcing each other (see figure 7). For example, if a subpath is common to two normalized flight paths, its corresponding digital pheromone trail (see figure 8) will have a strength of 2.

Input: Set P of unprocessed rows in the data preprocessing database containing normalized flight paths.
Output: Set T of rows in the data preprocessing database containing digital pheromone trails.

$N \leftarrow$ number of rows in P
$D \leftarrow$ maximum diffusion distance
$t \leftarrow$ new digital pheromone trail
for $p = 1$ **to** $N - 1$
    for each vector $\vec{u}$ **in** path p
        $u_e \leftarrow$ endpoint of $\vec{u}$
        for $q = p + 1$ **to** N
            for each vector $\vec{v}$ **in** path q
                $v_e \leftarrow$ endpoint of $\vec{v}$
                if *distance*$(u_e, v_e) > D$ **then**
                    if *length*$(t) > 0$ **then**
                        if t is not in database **then**
                            Insert t into database.
                        $t \leftarrow$ new digital pheromone trail
                else
                    Append $\vec{u}$ to t.
    if *length*$(t) > 0$ **then**
        if t is not in database **then**
            Insert t into database.
        $t \leftarrow$ new digital pheromone trail
Mark rows in P as completely processed by the digital pheromone trail discovery phase.

Figure 6. The algorithm for discovering digital pheromone trails from the normalized flights paths in the data preprocessing database.

Digital pheromone trails with greater strengths are thus more likely to represent actual maneuvers performed by pilots of GA aircraft. Furthermore, since two subpaths do

not have to be identical to be considered a match, only within some maximum diffusion distance D of each other, either of the subpaths could be selected as representative of the corresponding digital pheromone trail. In such cases, ADM arbitrarily selects the first subpath as the subpath which is representative of the digital pheromone trail.

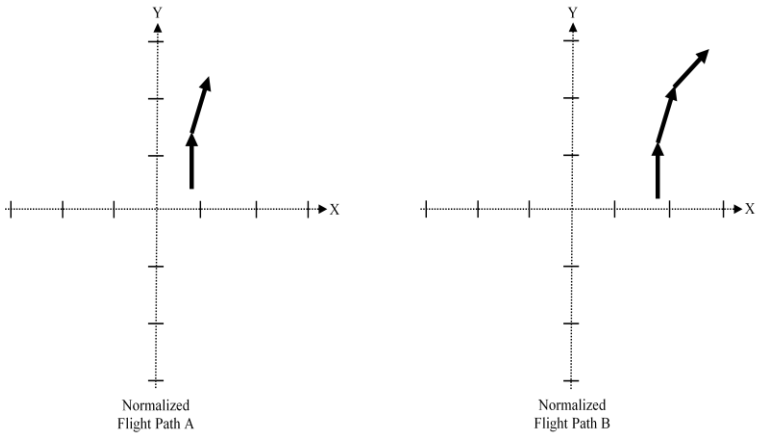

Figure 7. Normalized flight paths (projected in 2D) which have one common subpath (when D = 1).

During digital pheromone trail discovery, the shapes of the digital pheromone trails are important—not the GPS positions of their endpoints. Thus, each digital pheromone trail is assigned the same starting point in the internal coordinate system. Also, normalized flight paths are represented internally as ordered sequences of unit vectors to facilitate comparisons during this phase.

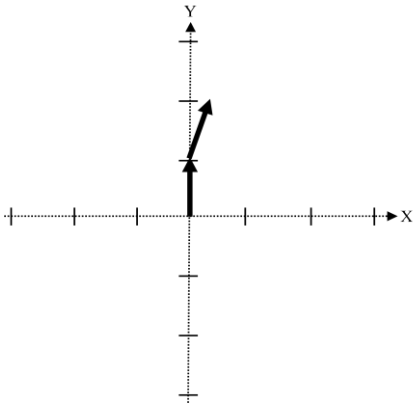

Figure 8. The digital pheromone trail discovered from normalized flight paths A and B (when D = 1). Its digital pheromone strength is 2.

Every normalized flight path for a given performance class is compared with every other normalized flight path for that performance class on a vector by vector basis. If two vectors are within some maximum diffusion distance D of each other, then they are considered a match. Since only the distances between unit vectors will be calculated, D must have a value less than 2. Any value for D greater than or equal to 2 will result in a case where every vector is within diffusion distance of every other vector. Also, any digital pheromone trails with lengths of less than 60 are discarded, because the RM subsystem is primarily concerned with the trajectories of GA aircraft over the next minute (i.e. 60 seconds). When a shorter minimum length of 5 was used for the digital pheromone trails, the discovered digital pheromone trails with the greatest relative subpath frequencies had insufficient lengths for accurately predicting trajectories of GA aircraft over the next minute.

The algorithm for digital pheromone trail discovery performs many distance calculations. To improve its run-time efficiency, a vector proximity map is constructed. A vector proximity map is a 2D Boolean array. It can determine if any two unit vectors are within some maximum diffusion distance D of each other in constant run-time. The angles (i.e. heading and ascent angles) for each of the vectors are encoded as integers. These encoded integers are then used as indexes into the vector proximity map. The element in the vector proximity map corresponding to those two vectors is 1 if the vectors are within diffusion distance of each other (i.e. the vectors match), or 0 otherwise.

The strength of a digital pheromone trail is a potential indicator of the frequency at which pilots performed this type of maneuver. In ant colony algorithms, the strengths of digital pheromone trails decrease over time through evaporation unless the digital

pheromone trails are continually reinforced by new digital pheromones. However, the time elapsed according to the system clock is not a useful control variable for this problem, since FDM data are not necessarily imported into the relational database at a constant rate. Thus, a better control variable for this problem is the amount of new FDM data that are being imported into the relational database. The discovery of digital pheromone trails thus ultimately results in a set of frequently occurring subpaths in the FDM data for a specific performance class.

Classifying Subpaths

The fifth and final phase of data preprocessing (see figure 9) involves segmenting the normalized flight paths into subpaths using the digital pheromone trails discovered during the previous phase. This is the only phase of data preprocessing which is nonincremental, because any insertions or deletions of digital pheromone trails that occur during the digital pheromone trail discovery phase necessitate the reclassification of all subpaths in the relational database. The subpaths identified during this phase are stored in the database for later retrieval during the data mining phases. Each normalized flight path P is segmented into subpaths in an iterative manner, starting with the first vector in the path. The current vector V_p from P is compared with the first vector V_t from every digital pheromone trail in the database whose length is less than or equal to the length of P. If the vectors V_p and V_t are within some maximum diffusion distance D of each other, then the digital pheromone trail is a potential match. If the vectors V_p and V_t are not within some maximum diffusion distance D of each other, then the digital pheromone trail is not a potential match.

```
Input: Set P of unprocessed rows in the data preprocessing database containing normalized flight paths.
Output: Set U of rows in the data preprocessing database containing subpaths.

for each path p in P
   for i = 1 to length(p)
      m ← longest digital pheromone trail with greatest strength which matches subpath p[i..length(m)]
      if not is_inside_controlled_airspace(p[i]) then
         u ← new subpath
         u.starting_altitude ← p[i].altitude
         u.starting_proximity_to_private_airport ← p[i].proximity_to_uncontrolled_airport
         u.vectors ← p[i..length(m)]
         if m = null then
            u.maneuver_type ← unknown
         else
            u.maneuver_type ← m.ID
         S ← lookup_table_for(p.performance_class)
         Insert subpath u into table S.
Mark any rows for data mining results for corresponding performance classes as incomplete.
```

Figure 9. The algorithm for classifying subpaths using the digital pheromone trails in the data preprocessing database.

Next, each of the potentially matching digital pheromone trails is compared with every subpath in P of the same length which has the same starting vector. This determines if any subpath of P completely matches one of the digital pheromone trails. From the set of complete matches to subpaths in P, the digital pheromone trail with the greatest strength is used to classify the subpath. If there are multiple such digital pheromone trails, the longest digital pheromone trail from the set of strongest matches is used to classify the subpath. The matching digital pheromone trail, having some length L_t, will thus match some subpath from P of length L_t. The next subpath segmented from P will start immediately after the end of the previous subpath. If there are only a few normalized flight paths for a particular performance class, it is possible for a normalized flight path to have subpaths that are not common to any other normalized flight paths. Such subpaths are defined by exclusion and cannot be classified.

The set of digital pheromone trails discovered for a particular performance class thus form a dynamic set of classes of maneuvers a pilot is likely to perform when flying a GA aircraft with that performance class. Each digital pheromone trail is a potentially

unique type of maneuver that was intentionally performed by pilots. In some cases, there are may be two or more digital pheromone trails with different strengths that match the same subpath in a normalized flight path. In these cases the digital pheromone trail with the greatest strength (i.e. the one that occurs most frequently in the data) is always used. However, since the actual intentions of the pilot are unknown, it is possible (though unlikely) that the pilot actually performed a maneuver corresponding to a digital pheromone trail of lesser strength.

Searching for Proximate Uncontrolled Airports

The aircraft's altitude, proximity to the nearest controlled airport, and proximity to the nearest uncontrolled airport at the start of each subpath are calculated and/or stored in the relational database for later retrieval. The proximity to the nearest controlled or uncontrolled airport is the geodesic distance from the aircraft's latitude/longitude position to the latitude/longitude position of the nearest controlled or uncontrolled airport, respectively. Proximity to the nearest airport is calculated by searching an airport database for controlled or uncontrolled airports, respectively, which are near the aircraft, finding the distance from the aircraft to each of the nearby controlled or uncontrolled airports, respectively, and then selecting the minimum of those distances.

If a high degree of accuracy is desired for geodesic distances, Vincenty's inverse method (Vincenty 1975) is preferred, since the geodesic distances calculated by Vincenty's inverse method are accurate to within half a millimeter. Although very accurate, algorithms based on Vincenty's inverse method can also be very CPU-

intensive. Thus, the algorithm for calculating the aircraft's proximity to an uncontrolled or controlled airport was designed to reduce the total number of distance calculations performed with Vincenty's inverse method.

The airport search algorithm represents the contiguous area of the NAS as a large grid of cells. A cell in the grid measures 50 meters on each side. Since the curvature of the Earth over a 50 square meters area is negligible, any curvature within the cells can be ignored. The number of cells in every column in the grid is 55,121 cells (which is equivalent to 2,756,050 meters). The number of cells in the rows of the grid varies from 82,259 cells (or 4,112,950 meters) to 117,052 cells (or 5,852,600 meters), depending on the row's latitude. To map a point specified as latitude and longitude to a point within the grid, the airport search algorithm only needs to calculate two distances using Vincenty's inverse method—from the western edge of the grid to the point and from the southern edge of the grid to the point.

After calculating those two distances, the airport search algorithm divides both distances by the length of a side of a grid cell (i.e. 50 meters), rounding down to the nearest integer, to obtain the X and Y coordinates for the point within the grid. The grid positions for all the airports (both uncontrolled and controlled) in the airport database are calculated prior to the subpath classification phase, and stored for later retrieval. The airport's type (i.e. either uncontrolled or controlled) is stored along with its grid position. Then, during the subpath classification phase, each latitude/longitude point along the aircraft's path is mapped to its corresponding point within the grid.

Since the grid positions for all the airports in the airport database are calculated offline, the airport search algorithm can efficiently determine which airports of a specific type are near the aircraft using the aircraft's current position within the grid. First, the algorithm searches for airports of that specific type within the same cell as the aircraft. If there aren't any airports of that specific type in the same cell as the aircraft, the algorithm searches all the cells that border that cell, and so on, until cells containing one or more airports of that specific type are found. Once the airport search algorithm finds the cell(s) containing the airports of that specific type which are closest to the aircraft's cell, it calculates the distance between the aircraft and each of those airports, and uses the minimum for the aircraft's proximity measurement.

Data Mining

ADM performs all of its data mining via SQL commands that operate on tables in a relational database. All tables used during the data mining phases are stored in the same relational database. ADM performs its data mining in two phases (see figure 10): 1) altitude mining and 2) proximity mining. These two data mining phases are performed separately for the data from each performance class in the FDM data set(s). The phases must occur in sequence to produce correct results.

Both of the data mining phases operate on the same relational database. This relational database stores all of the candidate solutions generated during both phases of data mining. Data are grouped into tables based on their respective performance classes. Since the subpath classification phase is nonincremental, all phases of data mining are also nonincremental. Accessing massive tables in a relational database is more costly

than accessing many smaller tables. Thus, each respective table in the relational database only stores data points for the clusters for the candidate solutions from a single generation.

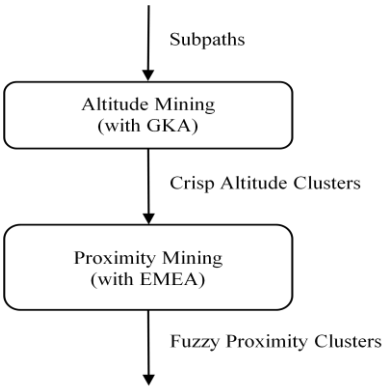

Figure 10. The flowchart for the two phases of data mining.

ADM uses a database-oriented implementation of GKA for altitude mining. GKA produces compact clusters by minimizing the TWCV. See figure 11 for the data structure diagram (DeMarco 1979) of the data mining phases. ADM uses the Expectation-Maximization Evolutionary Algorithm (EMEA) for proximity mining. EMEA is a database-oriented hybrid algorithm which combines the EM clustering algorithm with a genetic algorithm. EMEA uses a Gaussian mixture model for its probability distributions. Cluster compactness is not a very useful optimality criterion for fuzzy clustering algorithms, such as EMEA. If EMEA produces compact clusters, the centroids for these clusters could still be close to each other, causing the clusters to overlap. Thus, instead of producing compact clusters by minimizing the TWCV like GKA, EMEA produces clusters with sufficient separation by maximizing the distance between the centroids for the clusters.

The two phases of data mining share the same data model. The output data tables from the first data mining phase are the input data tables for the second data mining

phase. The aircraft data index used during the data preprocessing phases is also used during the data mining phases to determine which performance classes have data which are ready for data mining (i.e. completely preprocessed).

The altitude mining phase mines the subpaths stored in the data preprocessing database during the final phase of data preprocessing (i.e. the subpath classification phase). Then, the proximity mining phase mines the crisp clusters from the most optimal candidate solution stored in the data mining database by the altitude mining phase. The clusters produced during both phases of data mining are jointly specified by three types of data models—the cluster metadata model, the cluster data model, and the candidate solution metadata model. The cluster metadata model specifies general information about the clusters, such as their unique identifiers and important cluster statistics (e.g. the variation within each cluster). The cluster data model specifies the subpaths from which the clusters are composed. Each subpath in the cluster data model has an associated identifier which specifies its containing cluster. The candidate solution metadata model specifies general information about candidate solutions needed by GKA, such as the fitness values. EMEA does not use the candidate solution metadata model.

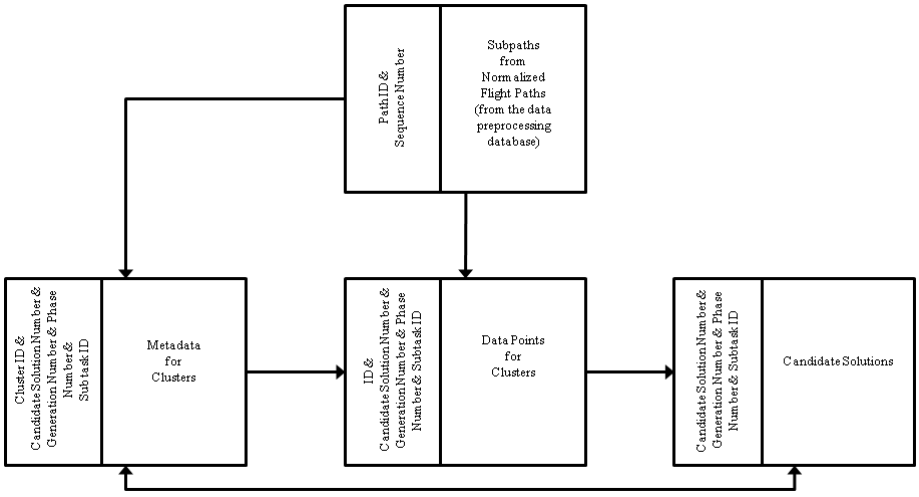

Figure 11. The data structure diagram for the data mining database.

If the aircraft telemetry data were mined jointly with respect to both the aircraft's altitude and its proximity to the nearest uncontrolled airport, e.g. by using 2D feature vectors of the form (A, P) where A is the aircraft's altitude and P is the aircraft's proximity, there would have to be a linear relationship between the aircraft's altitude and its proximity to the nearest uncontrolled airport. Either the aircraft's altitude would need to be dependent on the proximity, or the aircraft's proximity would need to be dependent on its altitude.

Using a 2D Euclidean-based distance metric typically requires minimizing a relaxed form of the 2D Euclidean distance function, such as:

$$f(x,y)=(x_1-x_2)^2+(y_1-y_2)^2$$

Thus, the closer the X and Y coordinates are between the two points, the greater their similarity will be according to a 2D Euclidean-based distance metric. However, this apparent linear relationship can be disproved with a counterexample. Consider two aircraft, A_1 and A_2. Aircraft A_1 is flying at 500 meters MSL and aircraft A_2 is flying at 3,000 meters MSL. If both aircraft are near uncontrolled airports, e.g. within a few kilometers, the maneuvers performed by the pilots of the aircraft are likely to be influenced by the proximity of their aircraft to the uncontrolled airports. However, the altitudes of the aircraft differ by 2,500 meters. Thus, their corresponding data points will be assigned to different clusters, even though the maneuvers performed by the pilots of these aircraft are likely to be very similar.

The behavior of the pilot of an aircraft with respect to variations in the aircraft's altitude is much easier to predict than the pilot's behavior with respect to variations in the aircraft's proximity to the nearest uncontrolled airport. At higher altitudes, terrain

features have less influence over the maneuvers performed by the aircraft's pilot. Conversely, at low altitudes, terrain features are one of the predominant factors influencing the maneuvers performed by an aircraft's pilot. Thus, data mining with respect to an aircraft's altitude should only result in a few altitude clusters, e.g. a cluster for high altitudes and a cluster for low altitudes, which should be compact clusters with crisp boundaries. This suggests a crisp clustering algorithm (i.e. GKA) should be used to mine aircraft telemetry data with respect to the aircraft's altitude.

The behavior of the pilot of an aircraft with respect to variations in the aircraft's proximity to the nearest uncontrolled airport is more complex. Thus, it is more difficult for a crisp clustering algorithm to correctly cluster the aircraft telemetry data with respect to proximity. The proximity clusters are likely to be less compact and have fuzzy boundaries. This suggests a fuzzy clustering algorithm (i.e. EMEA) should be used to mine aircraft telemetry data with respect to the aircraft's proximity to the nearest uncontrolled airport.

Clustering algorithms such as GKA and EMEA try to produce clusters which satisfy certain optimality criterion, such as cluster compactness and cluster separation. Furthermore, crisp clustering algorithms produce more compact clusters than fuzzy clustering algorithms. If a crisp clustering algorithm is used for the first phase of data mining, this will create highly compact clusters. Since the second phase will further refine the highly compact clusters produced during the first phase, if a fuzzy clustering algorithm is used for the second phase of data mining, it will be more likely to produce highly compact clusters. Thus, a crisp clustering algorithm (i.e. GKA) should be used during the first phase of data mining, and a fuzzy clustering algorithm (i.e. EMEA)

should be used during the second phase of data mining. The two phases of data mining result in probabilistic decision tree models (see figure 12) which will be applicable to GA aircraft flying outside the controlled airspace surrounding controlled airports.

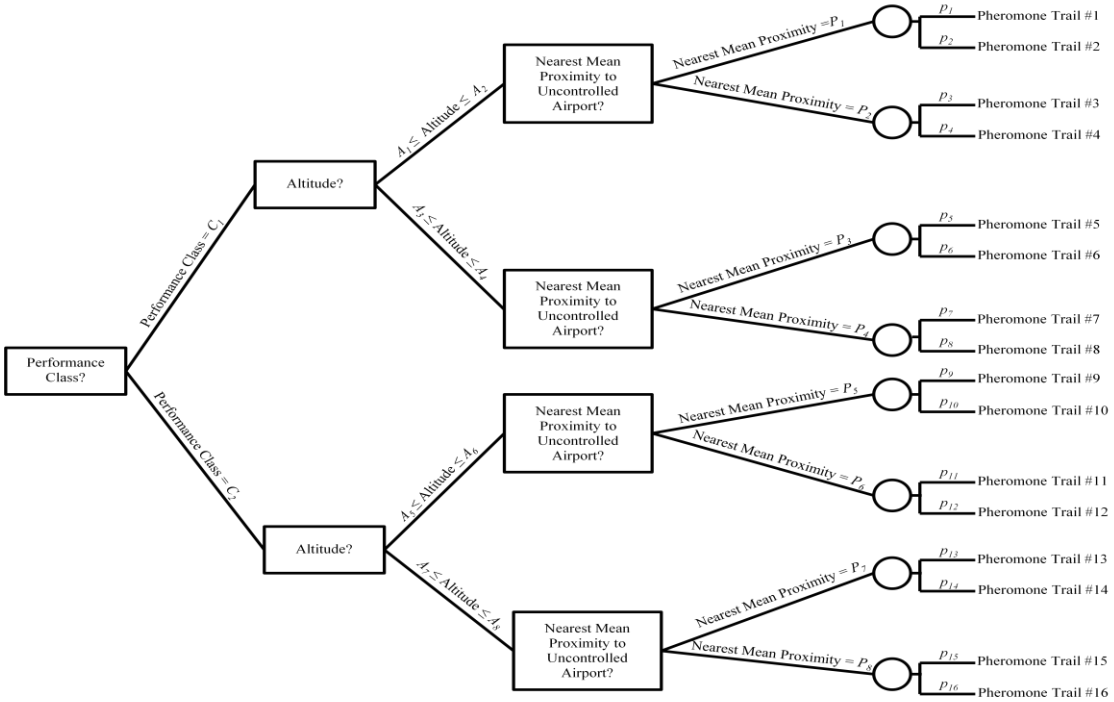

Figure 12. An example of a decision tree model based on data mining results.

Unlike GKA, the EM algorithm (and hence EMEA) produces fuzzy clusters where each data point has a probability of membership in each of the clusters. These fuzzy clusters do not have crisp boundaries that can be used in the resultant decision tree model to determine which cluster is most similar to a data point. Instead, the decision based on the aircraft's proximity to the nearest uncontrolled airport uses the fuzzy cluster whose mean proximity value is nearest to the proximity value of the aircraft's data point.

Mining Altitude Features

In the first phase of data mining, the subpaths from the preprocessed data for a performance class are automatically clustered into K_1 clusters with respect to the altitude at the start of each subpath. GKA (the data mining algorithm used during this phase) is a

hybrid algorithm which combines KMA (see figure 13) with a genetic algorithm. Since GKA is a crisp clustering algorithm, GKA tends to produce highly compact clusters, which is important during the first phase of data mining.

Input: Set *U* of rows in the data preprocessing database containing subpaths.
Output: Set *U′* of rows in the data preprocessing database containing subpaths mapped to crisp clusters.

$K_1 \leftarrow$ number of clusters
$D \leftarrow$ number of dimensions in a feature vector
$N \leftarrow$ number of rows in set *U*
$C_j \leftarrow$ size of the *j*th cluster
$\mu_j \leftarrow$ centroid of the *j*th cluster
$X_i \leftarrow i$th subpath from set *U*

for $j = 1$ **to** K_1

$$\mu_j \leftarrow \frac{\sum_{i=1}^{Cj} X_i}{Cj}$$

for $i = 1$ **to** N
  $d \leftarrow \infty$
  for $j = 1$ **to** K_1
    $\overline{d} \leftarrow \sum_{k=1}^{D} \left(X_{ik} - \mu_{jk}\right)^2$
    if $\overline{d} < d$ **then**
      $d \leftarrow \overline{d}$
      $G \leftarrow j$
  Assign X_i to cluster *G*.

Figure 13. The K-Means Algorithm (KMA).

GKA (see figure 14) takes several input parameters, including the number of clusters (K_1), the fitness constant to use (typically 1.5), the probability of a mutation occurring during reproduction, the number of candidate solutions in each generation of the population, and the total number of generations to produce. Since GKA uses a one-point crossover operator, a minimum of 3 candidate solutions must be used. Also, the parallelized GKA does not allow crossover between slave nodes in the computational cluster. Whether GKA is guaranteed to converge to the globally optimal solution depends on its implementation (Eiben, Aarts, and Van Hee 1991; Rudolph 1994).

Input: Set *U* of rows in the relational database containing subpaths.
Output: Set *L* of rows in the relational database containing subpaths mapped to crisp clusters for each candidate solution.

$K_1 \leftarrow$ number of clusters

$C_j \leftarrow$ size of the *j*th cluster

$\mu_j \leftarrow$ centroid of the *j*th cluster

$F_c \leftarrow$ fitness constant

$M \leftarrow$ number of candidate solutions in the population

$P \leftarrow$ *M* random mappings of the subpaths in set *U* to the clusters

$G \leftarrow$ number of generations

$P_m \leftarrow$ mutation probability

for $t = 1$ **to** G

do

$r_1 \leftarrow$ random number between 0 and 1 exclusive

$r_2 \leftarrow$ random number between 0 and 1 exclusive

while $r_1 = r_2$

for $s = 1$ **to** M

$$T_s \leftarrow \sum_{j=1}^{K_1} \sum_{i=1}^{C_j} \left(X_i - \mu_j\right)^2$$

$$T_{\max} \leftarrow \sum_{s=1}^{M} T_s$$

for $s = 1$ **to** M

$F_s \leftarrow F_c \times T_{max} - T_s$

$$F_{\max} \leftarrow \sum_{s=1}^{M} F_s$$

for $s = 1$ **to** M

$$\overline{F_s} \leftarrow \frac{F_s}{F_{\max}}$$

Sort normalized fitness values for candidate solutions in *P* in descending order.

$a \leftarrow 0$

for $s = 1$ **to** M

$a \leftarrow a + \overline{F_s}$

$A_s \leftarrow a$

Sort candidate solutions in *P* in ascending order by accumulated normalized fitness value.

for $s = 1$ **to** M

if $A_s > r_1$ **then**

father $\leftarrow S_s$

else if $A_s > r_2$ **then**

mother $\leftarrow S_s$

$S_{t+1} \leftarrow$ *mate*(*mother*, *father*)

$r \leftarrow$ random number between 0 and 1 inclusive

if $r < P_m$ **then**

$S_{t+1} \leftarrow$ mutated S_{t+1}

Run Algorithm K-Means on S_{t+1}.

Figure 14. The Genetic K-Means Algorithm (GKA) implemented with a generational population model and rank-based selection.

Since the FDM data from the Garmin G1000 only specify an aircraft's altitude above MSL, and not its altitude Above Ground Level (AGL), aircraft altitudes above MSL are used during the altitude mining phase. An aircraft's altitude above the terrain,

which may have more directly influenced the maneuvers chosen by the aircraft's pilot than its altitude above MSL, is thus not available during the altitude mining phase. If the aircraft's altitude AGL was available, it could be substituted for the aircraft's altitude above MSL without any modification to the altitude mining algorithm. High-resolution 3D terrain data, such as the Shuttle Radar Topography Mission (SRTM) terrain data obtained by the National Aeronautics and Space Administration (NASA) (National Aeronautics and Space Administration 2009), could be used to estimate the aircraft's altitude AGL from its altitude above MSL. However, estimates of an aircraft's altitude AGL using SRTM terrain data would have limited accuracy, since the resolution of SRTM terrain data is only 1 arcsecond (United States Geological Survey 2009).

Mining Proximity Features

In the second phase of data mining, the most optimal candidate solution generated for a given performance class during the first phase is data mined further. Each of the K_1 clusters from that candidate solution is data mined with respect to the aircraft's proximity to the nearest uncontrolled airport at the start of each subpath. This partitions each of the K_1 altitude clusters into K_2 proximity clusters, resulting in a probabilistic decision tree model.

EMEA (the data mining algorithm used during this phase) is a fuzzy clustering algorithm. Thus, EMEA (see figure 15) tends to produce less compact clusters. However, since the behavior of a pilot with respect to variations in the aircraft's proximity to the nearest uncontrolled airport is complex, estimating the correct number of crisp proximity clusters can be difficult. EMEA estimates the probability of each data point belonging to

each of the K_2 clusters, instead of assigning each data point to only one cluster. Thus, EMEA is less dependent on a correct estimate of the number of clusters in the data set to produce valid results.

Input: Set L' of rows in the relational database containing subpaths mapped to a crisp cluster from the most optimal candidate solution in L.
Output: Set L'' of rows in the relational database containing subpaths mapped to fuzzy clusters for each candidate solution.

$K_2 \leftarrow$ number of clusters
$M \leftarrow$ number of candidate solutions in the population
$P \leftarrow M$ mappings of the subpaths to clusters with random parameters
$G \leftarrow$ number of generations
$P_m \leftarrow$ mutation probability
$X_i \leftarrow i$th subpath from set L'
$H_{ij} \leftarrow$ probability of X_i being a member of jth cluster
$\mu_j \leftarrow$ mean for jth cluster
$\sigma_j^2 \leftarrow$ variance for jth cluster
$\pi_j \leftarrow$ weight for jth cluster
$\eta_{ij} \leftarrow$ Gaussian function evaluated for X_i considered as a member of jth cluster

for $t = 1$ **to** G
  for each candidate solution S_t **in** P

$$\eta_{ij} \leftarrow \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{\frac{-(X_i-\mu_j)^2}{2\sigma_j^2}}$$

$$H_{ij} \leftarrow \frac{\pi_j \cdot \eta_{ij}}{\sum_{r=1}^{K_2} \pi_r \cdot \eta_{ir}}$$

$$\mu_j \leftarrow \frac{\sum_{s=1}^{N} H_{sj} \cdot X_s}{\sum_{s=1}^{N} H_{sj}}$$

$$\sigma_j \leftarrow \frac{\sum_{s=1}^{N} H_{sj}(X_s-\mu_j)^2}{\sum_{s=1}^{N} H_{sj}}$$

$$\pi_j \leftarrow \frac{\sum_{s=1}^{N} H_{sj}}{\sum_{r=1}^{K_2}\sum_{s=1}^{N} H_{sr}}$$

  for each candidate solution S_t **in** P
    $r \leftarrow$ random number between 0 and 1
    if $r < P_m$ **then**
      $S_{t+1} \leftarrow$ randomly perturbed S_t
    else
      $S_{t+1} \leftarrow S_t$

Figure 15. The Expectation-Maximization Evolutionary Algorithm (EMEA).

EMEA takes several input parameters, including the number of clusters (K_2), the probability of a mutation, the number of candidate solutions in each generation of the population, and the total number of generations to produce. Since EMEA does not use a

crossover operator, it does not have any restrictions on the number of candidate solutions which may be used. Due to its lack of a crossover operator, EMEA is not guaranteed to converge to the globally optimal solution (Eiben, Aarts, and Van Hee 1991; Rudolph 1994). However, EMEA's use of a mutation operator and a set of multiple candidate solutions decreases the likelihood that EMEA will converge to a locally optimal solution.

Exploiting Data Parallelism

The data preprocessing and data mining algorithms exploit data parallelism by using a Beowulf computational cluster and the Parallel Virtual Machine (PVM) system (Geist, et al. 1994) for parallel computing. Since the master process and the database server both run on the master node, the master process needs to limit its CPU usage. Thus, the master process does not perform any data preprocessing or data mining itself. It merely divides the task and spawns slave processes on all the nodes in the computational cluster. In order to prevent key conflicts, temporary keys are used during the normalized flight path construction and digital pheromone trail discovery phases of data preprocessing, as well as both of the data mining phases. Each slave process spawned during these phases is assigned its own temporary key for its tables. After all slave processes for the phase are finished, the master process merges the temporary keys for the tables by: 1) adding the cumulative size of previous table partitions to each primary key field and 2) setting the temporary keys to null values. An advantage of temporary keys is separate tables do not need to be opened for each individual slave process. This typically results in less overhead and more cache hits for the database server.

For each of the parallelized data preprocessing algorithms, the master process performs the following steps: 1) divides the data preprocessing task by partitioning the input data set, 2) spawns slave processes for each of the partitions in the data set via calls to the PVM library, 3) marks the phase as complete after all the slave processes have successfully completed their tasks and notified the master process, and 4) merges the results from the slave processes.

The first phase of data preprocessing (i.e. the aircraft telemetry data import phase) is the only phase of data preprocessing or data mining which was not parallelized. If a centralized database server is used, there is no advantage to multiple slave processes importing data into the database simultaneously. When new data are imported into the database, the normalization, normalized flight path construction, digital pheromone trail discovery, and subpath classification phases of data preprocessing, as well as both of the data mining phases, are marked as incomplete.

During the second phase of data preprocessing (i.e. the normalization phase), the data for each table are partitioned for slave processes with respect to the aircraft associated with those data. Each slave process thus operates on its own set of aircraft. Since a composite primary key is used for the aircraft data tables, and no new rows are generated during this phase, temporary keys are not used.

During the third phase of data preprocessing (i.e. the normalized flight path construction phase), each normalized aircraft data table is partitioned with respect to the aircraft associated with the data. Thus, each slave process operates on its own set of aircraft. However, since each slave process constructs its own normalized flight paths,

temporary keys are used during this phase to avoid key conflicts. Furthermore, since digital pheromone trails are aged during this phase, if any existing digital pheromone trails in the relational database evaporate, then all normalized flight paths associated with those digital pheromone trails will need to be reprocessed during the subpath classification phase.

During the fourth phase of data preprocessing (i.e. the digital pheromone trail discovery phase), each path table and path vector table is partitioned with respect to its normalized flight paths. Each slave process thus operates on its own set of normalized flight paths. However, since each slave process can potentially discover new digital pheromone trails, temporary keys are used during this phase to avoid key conflicts.

During the fifth phase of data preprocessing (i.e. the subpath classification phase), each path table and path vector table is partitioned with respect to its normalized flight paths. Each slave process thus operates on its own set of normalized flight paths. Since a composite primary key is used for the subpath tables, which is based on the unique path identifiers used in the path and path vector tables, temporary keys are not used during this phase.

For each of the parallelized data mining algorithms, the master process performs the following steps: 1) divides the data mining task by assigning a fraction of the total candidate solutions to each node in the computational cluster, 2) spawns slave processes to data mine the subpopulations via calls to the PVM library, 3) marks the phase as complete after all the slave processes have successfully completed their tasks and notified the master process, and 4) merges the results from the slave processes. Since the data mining occurs in two phases (altitude mining followed by proximity mining), it is

necessary for ADM to wait for all the slave processes from the altitude phase to complete before any slave processes are spawned for the proximity mining phase. See figure 16 for the UML activity diagram for parallelized data mining using 2 slave processes.

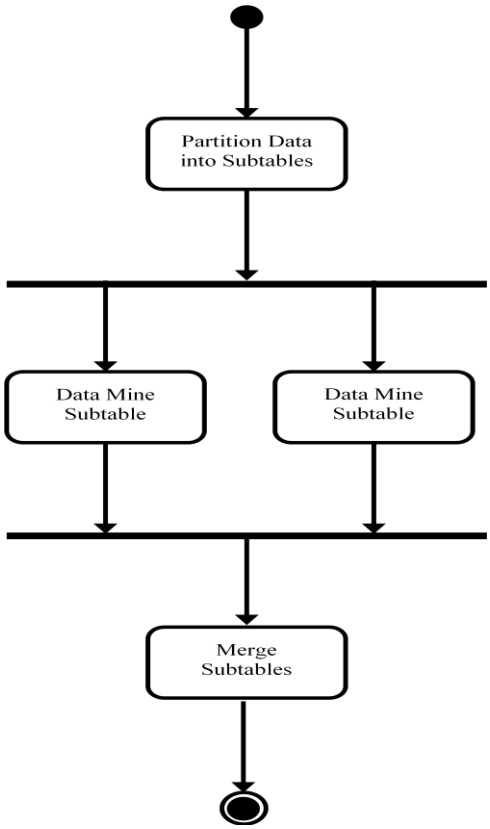

Figure 16. The UML activity diagram for the parallel data mining algorithms.

If a centralized database server is used during parallel data mining, explicit locking via communication between the master and slave nodes is necessary for balancing the load on the centralized database server. Thus, these parallel data mining algorithms exhibit coarse-grained parallelism when used with a centralized database server. However, the data parallelism of these data mining algorithms can be significantly improved by using a distributed database server (Lee, et al. 2000; Cheng, Lee, and Wong 2002; Ismail 2012).

CHAPTER IV

RESULTS

ADM was used to data mine the FDM data set S using two types of data preprocessing—nonincremental and incremental data preprocessing. Data set S consists of approximately 104 gigabytes of data archived between 3/13/2011 and 11/23/2011. These FDM data were obtained exclusively from 61 Cessna 172 planes in the University of North Dakota's training fleet. The data points in S have latitudes ranging from 35.47° to 49.91° and longitudes ranging from -108.56° to -81.85°.

In order to test the correctness of the incremental data preprocessing algorithms, as well as verify the data mining results through stability testing (Handl, Knowles, and Kell 2005), data set S was partitioned into two subsets, data set A and data set B. For the incremental data preprocessing: 1) Data set A was preprocessed; 2) Data set B was incrementally preprocessed and integrated with data set A; and 3) The subpaths from the combined data sets A and B were data mined with respect to altitude, and then with respect to proximity to uncontrolled airports.

Nonincremental Data Preprocessing

ADM sequentially preprocessed data set S as follows: 1) The raw FDM data from data set S, consisting of 136,287,621 data points, were imported into the database; 2) The imported data from data set S were normalized, deleting 3,655,423 data points which were potentially inside controlled airspace (of which 6,423 data points were within the Class B airspace surrounding a Class B airport); 3) The remaining 132,622,198 data

points from data set S were used to reconstruct 1,963 normalized flight paths; 4) From these normalized flight paths, 7,229 digital pheromone trails were discovered when D = 1; and 5) Of these digital pheromone trails, 1,197 digital pheromone trails were used to classify 27,188 subpaths (with 3,795,093 unclassifiable data points). See figures 17 and 18 for histograms of the subpaths with respect to starting altitude and starting proximity to uncontrolled airports, respectively.

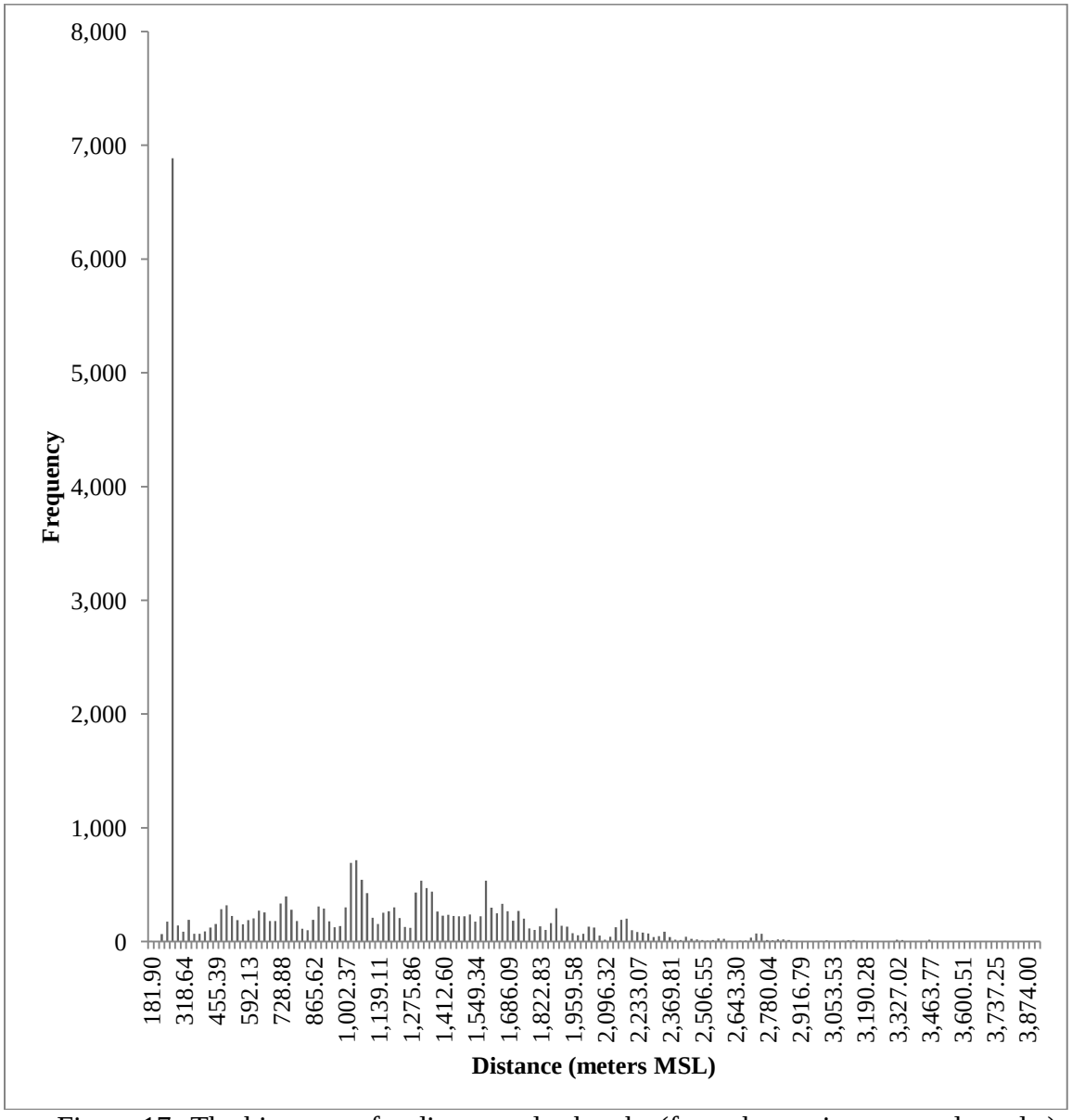

Figure 17. The histogram for discovered subpaths (from the nonincremental results) with respect to starting altitudes.

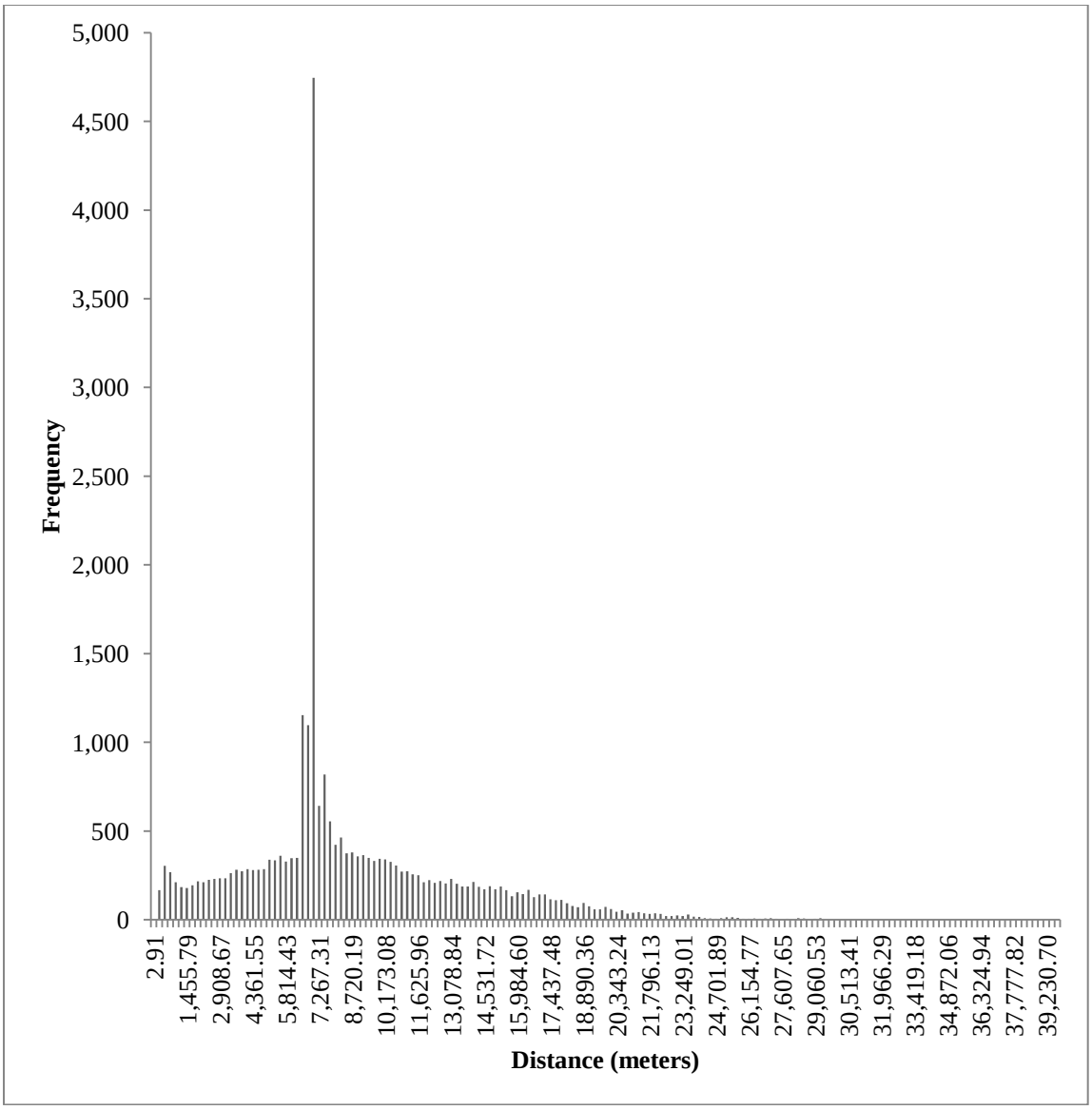

Figure 18. The histogram for discovered subpaths (from the nonincremental results) with respect to starting proximities to uncontrolled airports.

Although 7,229 digital pheromone trails were discovered, only 1,197 digital pheromone trails were actually used to classify subpaths. These 1,197 digital pheromone trails had the greatest digital pheromone strengths out of those digital pheromone trails which matched the given subpaths. Of those 1,197 digital pheromones trails, the two digital pheromone trails with a relative subpath frequency of at least 10% were digital pheromone trails #N56 (see table 1) and #N3882 (see table 2).

Table 1. The vector sequence for digital pheromone trail #N56.

Vector Sequence #	Heading (degrees)	Ascent Angle (degrees)	Duration (seconds)
1	90°	2°	63

Table 2. The vector sequence for digital pheromone trail #N3882.

Vector Sequence #	Heading (degrees)	Ascent Angle (degrees)	Duration (seconds)
1	90°	0°	1
2	88°	-2°	1
3	90°	-2°	1
4	88°	-2°	1
5	90°	-2°	3
6	88°	-2°	1
7	90°	-2°	1
8	88°	-2°	1
9	90°	-2°	3
10	88°	-2°	1
11	90°	-2°	3
12	88°	-2°	1
13	90°	-2°	2
14	88°	-2°	1
15	90°	-2°	1
16	88°	-2°	1
17	90°	-2°	1
18	88°	-2°	1
19	90°	-2°	3
20	88°	-2°	1
21	90°	-2°	2
22	88°	-2°	1
23	90°	-2°	1
24	88°	-2°	1
25	90°	-2°	1
26	88°	-2°	1
27	90°	-2°	1
28	88°	-2°	1
29	90°	-2°	1
30	88°	-2°	1
31	90°	-2°	1
32	88°	-2°	1
33	90°	-2°	1
34	88°	-2°	1
35	90°	-2°	1
36	88°	-2°	1
37	90°	-2°	1
38	88°	-2°	1
39	90°	-2°	1
40	88°	-2°	1
41	90°	-2°	1
42	88°	-2°	1
43	90°	-2°	2
44	88°	-2°	1
45	90°	-2°	3
46	88°	-2°	1
47	90°	-2°	1

After sequentially preprocessing the raw FDM data set, ADM performed parallel data mining of the resultant subpaths on a Beowulf computational cluster with the Network File System (NFS) and a centralized MySQL database server. ADM evolved 6 candidate solutions for 200 generations during both phases of data mining using a mutation probability of 17% and a crossover probability of 100%. To test ADM's parallel data mining algorithms, the data mining was performed on 3 nodes of the Beowulf cluster. ADM's master process did not perform any data mining itself. Instead, the master process merely divided the task and spawned slave processes on the remaining nodes in the Beowulf cluster. Thus, each of the 2 slave nodes was assigned 3 candidate solutions for parallel data mining.

During the first phase of data mining (i.e. altitude mining), the minimum TWCV of 3,519,079,060.43 was reached on the 52nd generation by candidate solution #2. During the second phase of data mining (i.e. proximity mining), the maximum cluster separation of 1,682.57 was reached for the first altitude cluster on the 7th generation (by candidate solution #6) and the maximum cluster separation of 6,875.62 was reached for the second altitude cluster on the 36th generation (by candidate solution #2). The data mining results are shown in tables 3 and 4. These results have a significance level of 0.05% when statistically validated with Pearson's chi-square test (Pearson 1900; Handl, Knowles, and Kell 2005).

ADM first categorized the data points into one of 2 clusters based on the aircraft's altitude. Two clusters were used for the first phase of data mining because the subpath altitudes appear to be grouped into two large clusters with a boundary near approximately 980 meters (or 3,215 feet) MSL (see figure 17). This resulted in a cluster of low altitudes

(containing 15,851 data points) and a cluster of high altitudes (containing 11,337 data points). Then, for each of these clusters, ADM categorized the data points within the cluster into one of 2 subclusters based on the aircraft's proximity to the nearest uncontrolled airport. Two proximity subclusters were used for each altitude cluster because the subpath proximities appear to be grouped into four clusters with possible boundaries at approximately 1,215 meters, 6,060 meters, and 12,110 meters (see figure 18). Since EMEA was used for proximity mining, the proximity clusters produced were not crisp clusters.

Table 3. The relative subpath frequencies and membership probabilities for the maneuvers (with a relative frequency of at least 1%) discovered in the first altitude cluster (with starting altitudes from 181.9 to 1,093.9 meters MSL) from the nonincremental results (for Cessna 172 aircraft).

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
N7216	4	1.03%	55.61%	44.39%
N7222	2	1.03%	51.05%	48.95%
N2050	31	1.05%	43.26%	56.74%
N3243	30	1.08%	55.62%	44.38%
N3231	94	1.23%	55.71%	44.29%
N3237	80	1.26%	55.54%	44.46%
N3232	50	2.41%	54.64%	45.36%
N3880	56	2.43%	42.32%	57.68%
N6919	149	2.73%	61.95%	38.05%
N699	880	3.33%	55.21%	44.79%
N3887	36	3.62%	44.75%	55.25%
N2008	82	3.64%	44.8%	55.2%
N7210	7	3.88%	50.67%	49.33%
N3878	97	3.92%	42.21%	57.79%
N362	1,642	4.02%	67.38%	32.62%
N101	1,243	4.91%	64.76%	35.24%
N725	924	5.03%	59.06%	40.94%
N3882	118	8.95%	44.66%	55.34%
N56	19,810	22.56%	66.42%	33.58%

Table 4. The relative subpath frequencies and membership probabilities for the maneuvers (with a relative frequency of at least 1%) discovered in the second altitude cluster (with starting altitudes from 1,094.14 meters to 3,919.58 meters MSL) from the nonincremental results (for Cessna 172 aircraft).

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
N743	1,049	1.03%	48.06%	51.94%
N725	924	1.09%	64.04%	35.96%
N6919	149	1.16%	60.58%	39.42%
N3232	50	1.23%	49.66%	50.34%
N3923	2	1.4%	54.42%	45.58%
N7210	7	1.59%	60.1%	39.9%
N699	880	1.6%	59.71%	40.29%
N2050	31	1.68%	56.47%	43.53%
N3880	56	4.9%	54.83%	45.17%
N3887	36	5.48%	57.14%	42.86%
N2008	82	5.92%	57.75%	42.25%
N3878	97	9.79%	55.1%	44.9%
N3882	117	13.11%	55.63%	44.37%

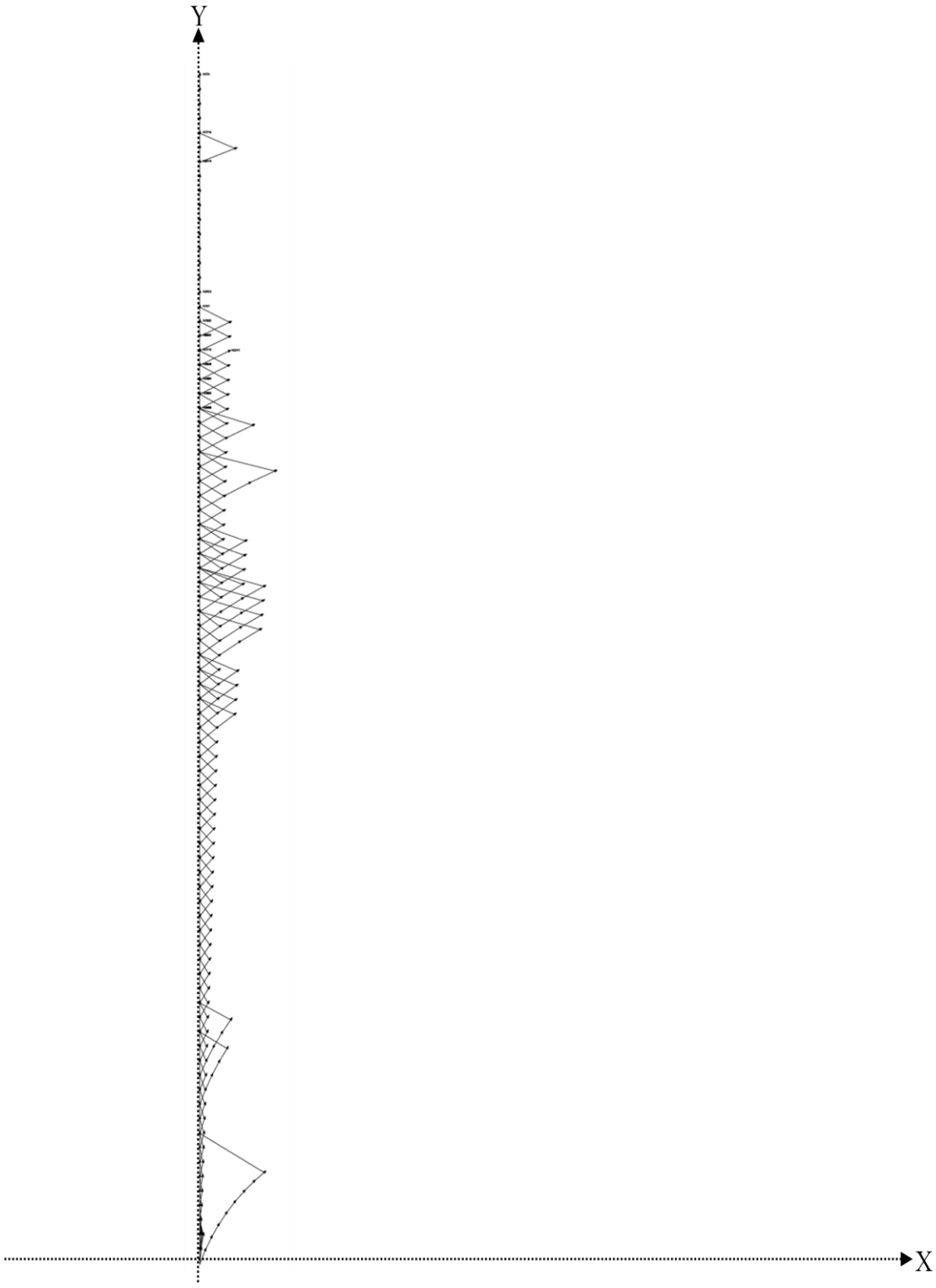

Figure 19. The digital pheromone trails (projected in 2D) used to classify the subpaths from nonincremental preprocessing. Only those with relative subpath frequencies of at least 1% are shown. The positive Y axis points in the aircraft's forward direction, and the positive X axis points to the right.

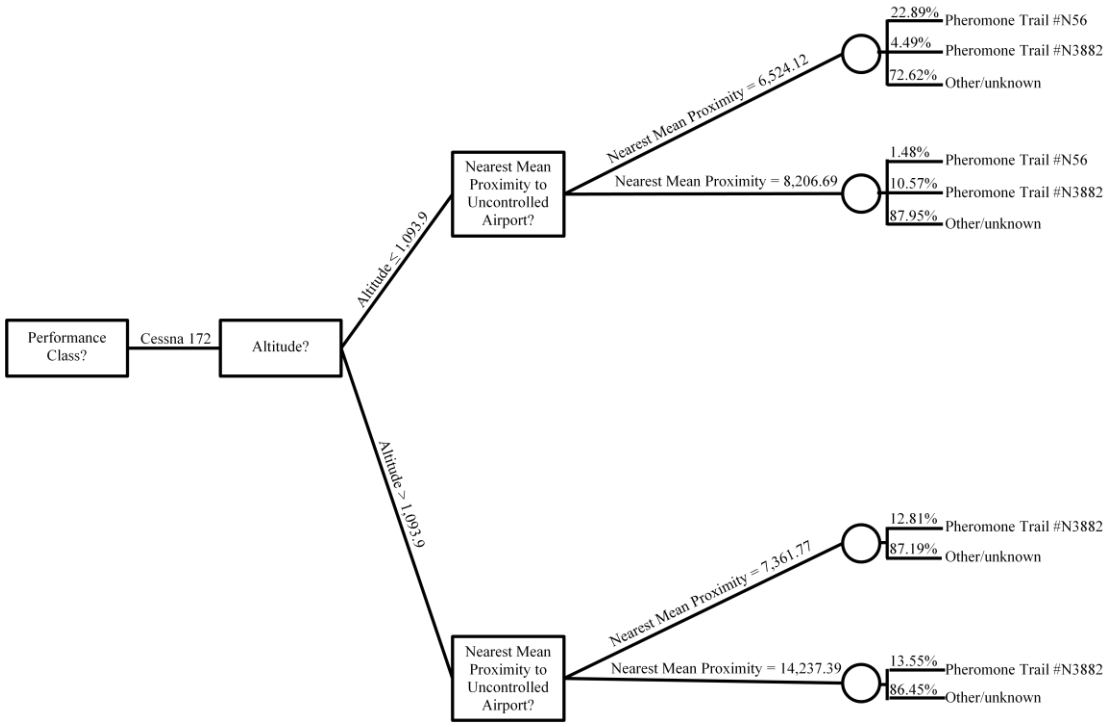

Figure 20. The decision tree model from data mining the results of nonincremental preprocessing. Only digital pheromone trails with a relative subpath frequency of at least 10% are shown. Subpaths for both altitude clusters were grouped with the cluster with the closest mean proximities.

Incremental Data Preprocessing

ADM sequentially preprocessed data set A as follows: 1) The raw FDM data from data set A, consisting of 67,687,525 data points, were imported into the database; 2) The imported data from data set A were normalized, deleting 1,755,356 data points which were potentially inside controlled airspace; 3) The remaining 65,932,169 data points from data set A were used to reconstruct 936 normalized flight paths; 4) From these normalized flight paths, 2,734 digital pheromone trails were discovered when D = 1; and 5) Of these digital pheromone trails, 781 digital pheromone trails were used to classify 10,198 subpaths (with 2,033,648 unclassifiable data points).

After sequentially preprocessing data set A, data set B was incrementally preprocessed and integrated with data set A as follows: 1) The raw FDM data from data set B, consisting of 68,600,096 data points, were imported into the database; 2) The imported data from data set B were normalized, deleting 1,897,471 data points which were potentially inside controlled airspace; 3) The remaining 66,702,625 data points from data set B were used to construct an additional 1,000 normalized flight paths; 4) After these normalized flight paths were constructed, 1,367 digital pheromone trails from data set A were evaporated using a proportional evaporation constant of 1,000; 5) From these normalized flight paths, 4,002 additional digital pheromone trails were discovered when D = 1; and 6) Of these digital pheromone trails, 853 digital pheromone trails were used to classify 17,136 additional subpaths (with 1,927,902 unclassifiable data points). The histograms of the subpaths with respect to starting altitude and starting proximity to uncontrolled airports are similar to the respective histograms from the nonincremental results.

Of those 1,634 digital pheromones trails (from the union of data sets A and B), the three digital pheromone trails with a relative subpath frequency of at least 10% are digital pheromone trails #I276 (see table 5), #I456 (see table 6), and #I4949 (see table 7).

Table 5. The vector sequence for digital pheromone trail #I276.

Vector Sequence #	Heading (degrees)	Ascent Angle (degrees)	Duration (seconds)
1	90°	-2°	61

Table 6. The vector sequence for digital pheromone trail #I456.

Vector Sequence #	Heading (degrees)	Ascent Angle (degrees)	Duration (seconds)
1	90°	2°	62
2	88°	4°	1

Table 7. The vector sequence for digital pheromone trail #I4949.

Vector Sequence #	Heading (degrees)	Ascent Angle (degrees)	Duration (seconds)
1	90°	0°	1
2	88°	-2°	1
3	90°	-2°	1
4	88°	-2°	1
5	90°	-2°	3
6	88°	-2°	1
7	90°	-2°	1
8	88°	-2°	1
9	90°	-2°	3
10	88°	-2°	1
11	90°	-2°	3
12	88°	-2°	1
13	90°	-2°	2
14	88°	-2°	1
15	90°	-2°	1
16	88°	-2°	1
17	90°	-2°	1
18	88°	-2°	1
19	90°	-2°	3
20	88°	-2°	1
21	90°	-2°	2
22	88°	-2°	1
23	90°	-2°	1
24	88°	-2°	1
25	90°	-2°	1
26	88°	-2°	1
27	90°	-2°	1
28	88°	-2°	1
29	90°	-2°	1
30	88°	-2°	1
31	90°	-2°	1
32	88°	-2°	1
33	90°	-2°	1
34	88°	-2°	1
35	90°	-2°	1
36	88°	-2°	1
37	90°	-2°	1
38	88°	-2°	1
39	90°	-2°	1
40	88°	-2°	1
41	90°	-2°	1
42	88°	-2°	1
43	90°	-2°	2
44	88°	-2°	1
45	90°	-2°	3
46	88°	-2°	1
47	90°	-2°	1

After incrementally and sequentially preprocessing data sets A and B, ADM performed sequential data mining of the resultant subpaths on a quad-core 64-bit server with a centralized MySQL database server. ADM evolved 6 candidate solutions for 200 generations during both phases of data mining using a mutation probability of 17% and a crossover probability of 100%. During the first phase of data mining (i.e. altitude mining), the minimum TWCV of 3,528,381,952.68 was reached on the 18th generation by candidate solution #2. During the second phase of data mining (i.e. proximity mining), the maximum cluster separation of 1,647.34 was reached for the first altitude cluster on the 12th generation (by candidate solution #5) and the maximum cluster separation of 6,888.42 was reached for the second altitude cluster on the 36th generation (by candidate solution #1). The data mining results are shown in tables 15 and 16. These results have a significance level of 0.05% when statistically validated with Pearson's chi-square test (Pearson 1900; Handl, Knowles, and Kell 2005).

ADM first categorized the data points into one of 2 clusters based on the aircraft's altitude. This resulted in a cluster of low altitudes (containing 15,970 data points) and a cluster of high altitudes (containing 11,364 data points). Then, for each of these clusters, ADM categorized the data points within the cluster into one of 2 subclusters based on the aircraft's proximity to the nearest uncontrolled airport.

Table 8. The relative subpath frequencies and membership probabilities for the maneuvers (with a relative frequency of at least 1%) discovered in the first altitude cluster (with starting altitudes from 181.9 to 1,090.3 meters MSL) from the incremental results (for Cessna 172 aircraft).

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
I1815	2	1.12%	47.01%	52.99%
I4344	30	1.15%	45.32%	54.68%

Table 8. Cont.

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
I1811	12	1.36%	44.27%	55.73%
I4346	78	1.56%	43.05%	56.95%
I4351	50	2.22%	43.79%	56.21%
I4957	54	2.31%	56.97%	43.03%
I1580	32	3.09%	39.77%	60.23%
I4971	36	3.63%	54.88%	45.12%
I3224	82	3.8%	55.67%	44.33%
I4952	93	3.93%	58.13%	41.87%
I1807	26	4.08%	49.44%	50.56%
I4949	113	9.19%	54.49%	45.51%
I276	8,312	20.76%	39.28%	60.72%
I456	16,162	23.03%	34.22%	65.78%

Table 9. The relative subpath frequencies and membership probabilities for the maneuvers (with a relative frequency of at least 1%) discovered in the second altitude cluster (with starting altitudes from 1,090.57 meters to 3,954.81 meters MSL) from the incremental results (for Cessna 172 aircraft).

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
I4351	50	1.08%	51.61%	48.39%
I1580	32	1.28%	59.46%	40.54%
I4969	2	1.42%	55.01%	44.99%
I1807	26	1.61%	59.51%	40.49%
I4957	54	4.83%	54.23%	45.77%
I276	8,312	5.41%	57.22%	42.78%
I4971	36	5.76%	58.02%	41.98%
I3224	82	5.79%	58.84%	41.16%
I4952	93	9.78%	54.82%	45.18%
I4949	113	12.91%	55.65%	44.35%

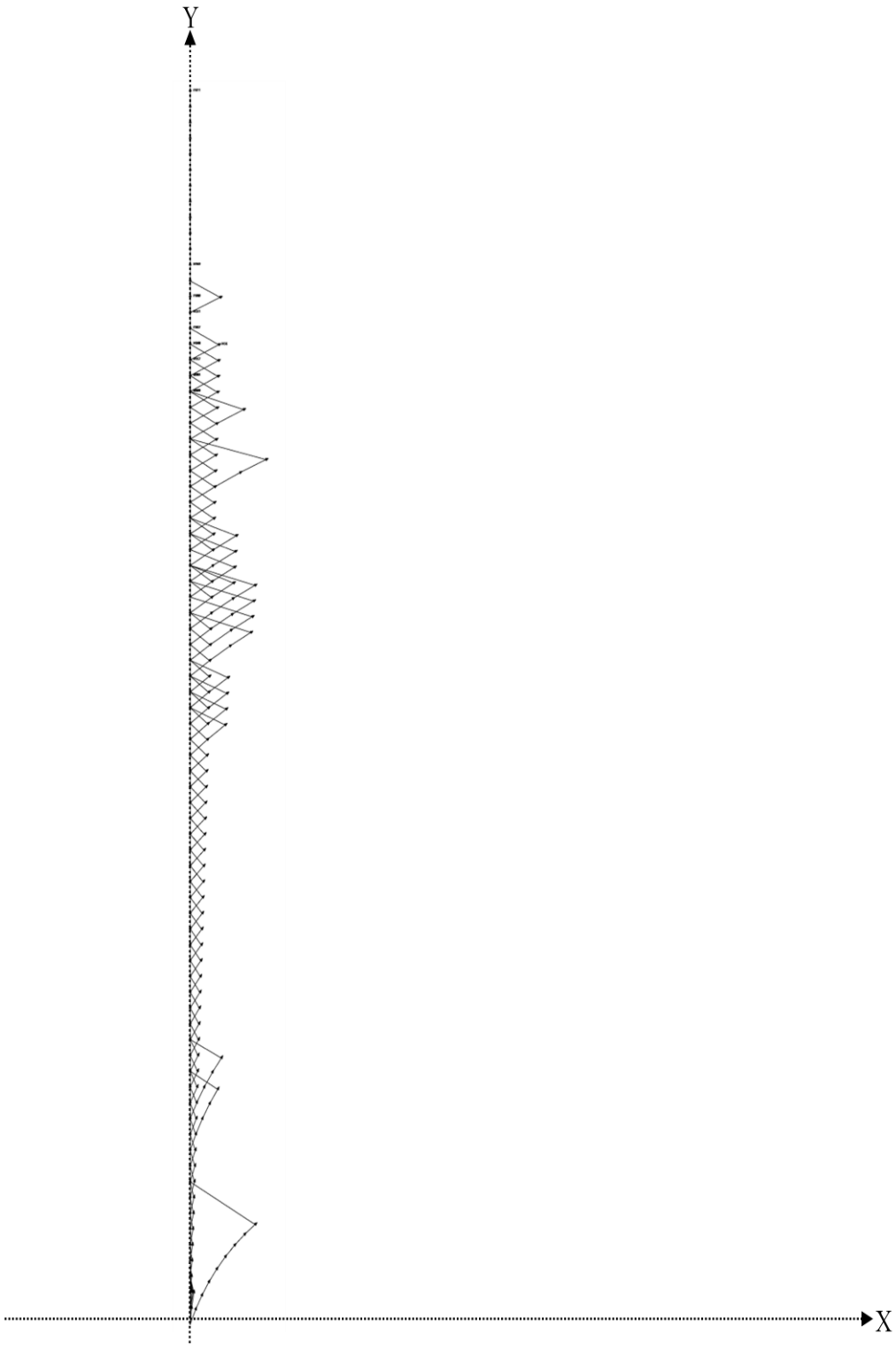

Figure 21. The digital pheromone trails (projected in 2D) used to classify the subpaths from incremental preprocessing. Only those with relative subpath frequencies of at least 1% are shown. The positive Y axis points in the aircraft's forward direction, and the positive X axis points to the right.

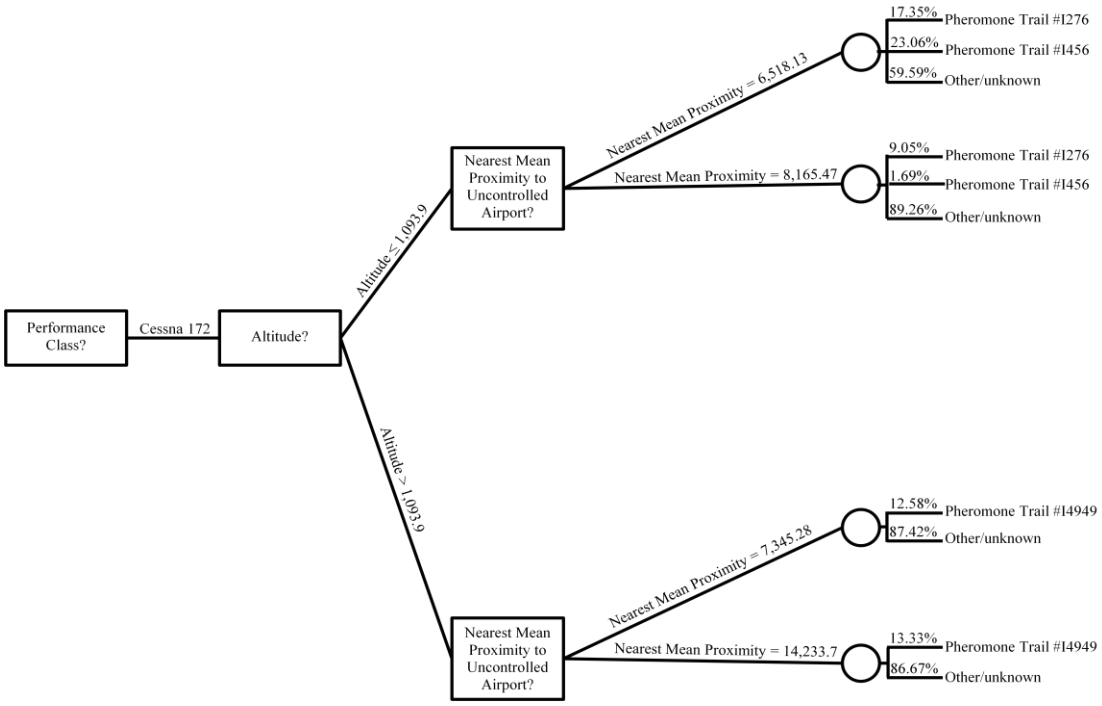

Figure 22. The decision tree model from data mining the results of incremental preprocessing. Only digital pheromone trails with a relative subpath frequency of at least 10% are shown. Subpaths for both altitude clusters were grouped with the cluster with the closest mean proximities.

Table 10. The run times for the five phases of sequential data preprocessing of a 1 gigabyte FDM data set.

Data Preprocessing Phase	Run time (seconds)
Importing Aircraft Telemetry Data	680
Normalizing Database	6,701
Constructing Normalized Flight Paths	40
Discovering Digital Pheromone Trails	7
Classifying Subpaths	140

Table 11. The run times for the two phases of sequential data mining of the preprocessed data set.

Data Mining Phase	Run time (seconds)
Altitude Mining	8,218
Proximity Mining	12,886

Table 12. The run times for the two phases of parallel data mining of the preprocessed data set using a centralized database server.

Data Mining Phase	Run time (seconds)
Altitude Mining	25,969
Proximity Mining	35,955

Table 13. The run times for the two phases of parallel data mining of the preprocessed data set using a distributed database server.

Data Mining Phase	Run time (seconds)
Altitude Mining	11,048
Proximity Mining	20,373

Validation with Synthetic Data

Synthetic FDM data were used to separately validate the sequential, nonincremental data preprocessing and data mining algorithms. To validate the sequential, nonincremental data preprocessing algorithms (see table 14), eleven CSV data files were constructed by hand to simulate FDM data from eleven different GA aircraft. Ten of these synthetic CSV files contained identical paths composed of alternating left and right turns. The other synthetic CSV file contained a different path composed of straight and level flight. This was intended to test the ability of the ant colony algorithm to correctly discover the shape, length, and strength of the path which was common to 10 of the 11 synthetic CSV files (see figure 23).

Table 14. The expected and actual results from validating the sequential, nonincremental data preprocessing algorithms using synthetic FDM data.

Data Preprocessing Phase	Expected Result	Actual Result
Importing Aircraft Telemetry Data	Imported 990 data points.	Imported 990 data points.
Normalizing Database	Deleted 0 data points.	Deleted 0 data points.
Constructing Normalized Flight Paths	Constructed 11 paths.	Constructed 11 paths.
Discovering Digital Pheromone Trails	Discovered 2 trails.	Discovered 2 trails.
Classifying Subpaths	Successfully classified 11 subpaths.	Successfully classified 17 subpaths. Seventeen subpaths could not be classified.

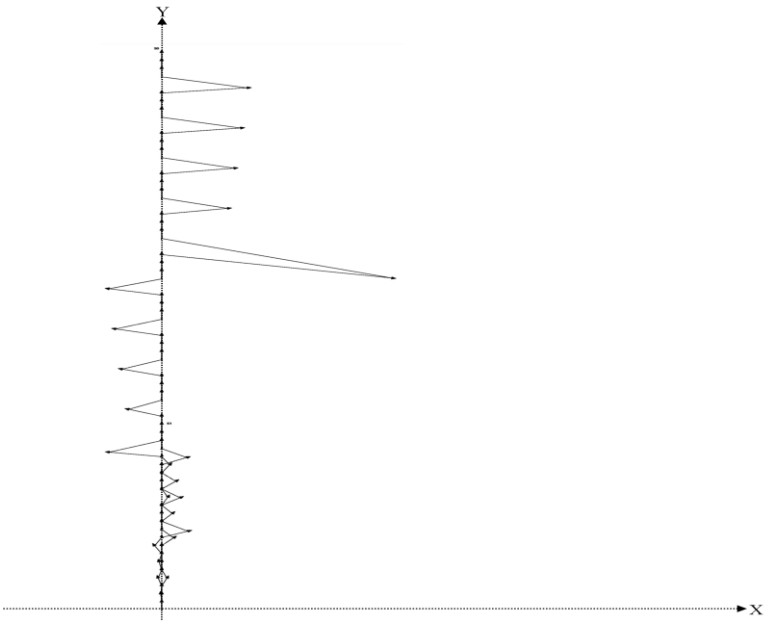

Figure 23. The digital pheromone trails (projected in 2D) used to classify the subpaths from the synthetic FDM data.

Table 15. The expected and actual results from validating the sequential data mining algorithms using synthetic FDM data.

Data Mining Phase	Expected Result	Actual Result
Altitude Mining	Produced 2 non-overlapping crisp clusters—a cluster of low starting altitudes and a cluster of high starting altitudes.	Produced 2 non-overlapping clusters—a cluster of low starting altitudes (from 91.44 to 96.44 meters MSL) and a cluster of high altitudes (from 194.488 to 210.51 meters MSL).
Proximity Mining	Produced 2 well-separated fuzzy clusters.	Produced 2 well-separated fuzzy clusters (with a maximum total separation of 5,842.17).

Table 16. The relative subpath frequencies and membership probabilities for the maneuvers discovered in the first altitude cluster from the synthetic FDM data.

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
S0	90	100%	66.66%	33.33%

Table 17. The relative subpath frequencies and membership probabilities for the maneuvers discovered in the second altitude cluster from the synthetic FDM data.

Pheromone Trail ID	Pheromone Strength	Relative Subpath Frequency	Probability of Membership in Cluster 1	Probability of Membership in Cluster 2
S0	90	72.73%	62.5%	37.5%
S1	20	27.27%	100%	0%

The subpaths produced by the sequential, nonincremental data preprocessing of the synthetic FDM data were used to construct synthetic data to validate the sequential data mining algorithms. The starting altitudes and proximities to uncontrolled airports for the subpaths in the second synthetic data set were modified to conform to the desired probability distribution. During sequential data mining of the synthetic FDM data, the minimum TWCV of 366.42 was reached by all candidate solutions on the 1st generation. The first altitude cluster contained 6 subpaths, with starting altitudes from 91.44 meters (or 300 feet) MSL to 96.44 meters (or 316 feet) MSL, and the second altitude cluster contained 28 subpaths, with starting altitudes from 194.88 meters (or 639 feet) MSL to 210.51 meters (or 691 feet) MSL. The maximum cluster separation of 5,842.17 was reached on the 7th generation by candidate solution #3. This resulted in a decision tree model (see figure 24).

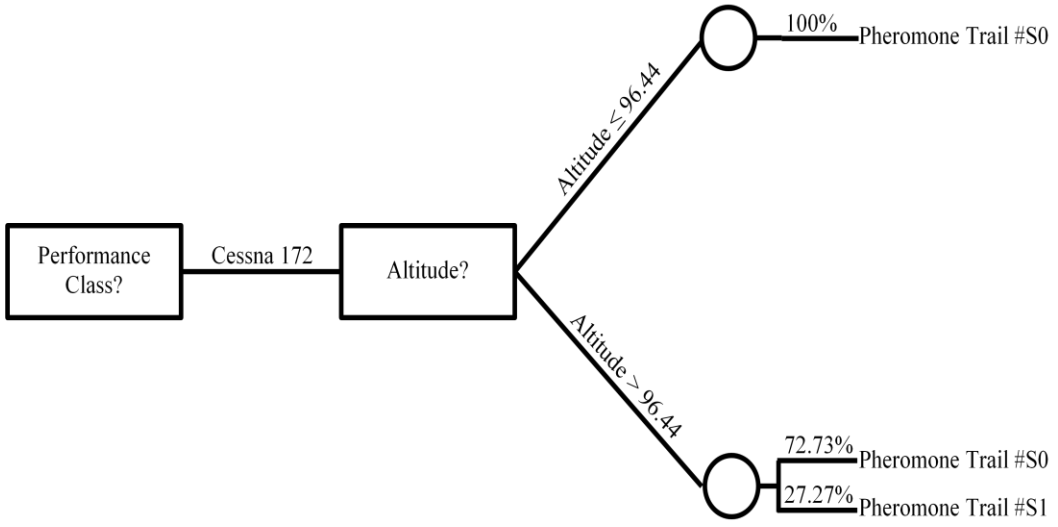

Figure 24. The decision tree model from data mining the synthetic FDM data.

Accuracy and Performance Testing with Real Data

To evaluate the accuracy and performance of a predictive algorithm based on the decision tree model from the nonincremental results, two simple test programs were implemented for testing the accuracy and performance, respectively. Approximately 133 megabytes of FDM data (consisting of 253,088 data points) obtained from flights for a single Cessna 172 aircraft during early 2012 were used to test both the accuracy and performance during sequential execution of the two test programs. The average error in the predicted values for the aircraft's heading angles when compared with the actual values was about 21°, and the average error in the predicted values for the aircraft's ascent angles when compared with the actual values was about 4°. After running on a single CPU for 13,315 seconds, the performance test program had processed 82,802 relevant data points, giving an average performance of 0.38 seconds per data point. The remaining 170,286 data points in the FDM data were skipped due to invalid data points and data points which were potentially inside Class B airspace around a controlled airport.

CHAPTER V

ANALYSIS

The membership probabilities for digital pheromone trails in the first altitude cluster from the nonincremental results (see table 3) do not show a strong bias towards either of the possible proximity clusters. This suggests that, at relatively low altitudes, the maneuvers naturally form into two proximity clusters—a cluster of aircraft near to an uncontrolled airport and a cluster of aircraft far from an uncontrolled airport. The membership probabilities for the second altitude cluster from the nonincremental results (see table 4) also do not show a strong bias towards either of the possible proximity clusters. This suggests that, at higher altitudes, pilot maneuvers also naturally group into two proximity clusters—a cluster near to and a cluster far from an uncontrolled airport. The membership probabilities for digital pheromone trails from the incremental results (see tables 8 and 9) suggest this as well.

Verification Results

Although the nonincremental and incremental data preprocessing used the same raw FDM data as input, the two methods did not preprocess those data identically. For example, the incremental data preprocessing resulted in the evaporation of 1,367 digital pheromone trails (those with the lowest pheromone strengths), while the nonincremental data preprocessing obviously did not involve the evaporation of any digital pheromone trails. Thus, the nonincremental and incremental data preprocessing should not be expected to produce identical results. Furthermore, the stochastic nature of the data

mining algorithms (i.e. GKA and EMEA) resulted in small variations in the data mining results between the two result sets (i.e. the nonincremental and incremental preprocessing results). Also, since the particular implementations of GKA and EMEA are not guaranteed by (Rudolph 1994) to converge to the globally optimal solution, either data mining phase could have produced a locally optimal solution. However, the results from both result sets are similar, which suggests stability in the results derived from the two different types of preprocessing algorithms.

For the nonincremental result set, 2 digital pheromone trails with relative frequencies of 10% or greater were used to classify subpaths—digital pheromone trails #N56 and #N3882. Digital pheromone trail #N56 (see table 1) could represent, for example, a maneuver where the aircraft maintained a straight heading for 63 seconds while ascending at an angle of 2°, while digital pheromone trail #N3882 (see table 2) could represent a maneuver where the aircraft turned to the right at an average turn rate of 2° per second. For the incremental result set, 3 digital pheromone trails with relative frequencies of 10% or greater were used to classify subpaths—digital pheromone trails #I276 (see table 5), #I456 (see table 6), and #I4949 (see table 7).

Thus, more digital pheromone trails with relative frequencies of 10% or greater were used to classify subpaths in the incremental result set than in the nonincremental result set. This is likely due to the evaporation of digital pheromone trails that occurs during incremental preprocessing. Evaporation of digital pheromone trails deletes a certain number of digital pheromone trails with the lowest pheromone strengths. The number of digital pheromone trails deleted is proportional to the size of the data set which is incrementally preprocessed. Evaporation also decreases the pheromone strengths

of all the remaining digital pheromone trails. Thus, evaporation tends to decrease the dispersion of pheromone strengths in the remaining digital pheromone trails.

Both the nonincremental (see figure 19) and incremental (see figure 21) result sets contain similar sets of digital pheromone trails with relative subpath frequencies of 1% or greater, consisting of straight flight and right turns. While digital pheromone trails representing left turns were discovered, those digital pheromone trails had relative subpath frequencies of less than 1%. This indicates that left turns with a duration of one minute or longer were performed less frequently than straight flight and right turns. Each arrow in figures 19, 21, and 23 represents a 1 second segment of the digital pheromone trail. The "jumps" in the digital pheromone trails from both result sets representing right turns (see figures 19 and 21) indicate the aircraft resumed straight flight, i.e. the aircraft heading was normalized to 90° after the "jump" in the digital pheromone trail. The decision tree models for the nonincremental (see figure 20) and incremental (see figure 22) result sets are also similar, with each cluster having one maneuver which was performed much more frequently than the other maneuvers.

Validation Results

During the test with synthetic data, the ant colony algorithm correctly discovered the shape (see figure 23), length (70), and strength (90) of the identical paths in 10 of the 11 synthetic CSV files. The ant colony algorithm also correctly discovered the shape (see figure 23), length (24), and strength (20) of a repeating subpath of the different path in the other synthetic CSV file. The strength of the digital pheromone trail discovered for the 10 synthetic CSV files containing an identical path is 90 because, during the digital pheromone discovery phase of data preprocessing, the strength for a digital pheromone

trail is incremented by 2 for each path which contains the common subpath. The strength of the digital pheromone trail discovered for the other synthetic CSV is 20 because the digital pheromone trail was a common subpath to all of the identical paths in the 10 synthetic CSV files. However, since the digital pheromone trail which was common to all of the 11 synthetic CSV files was shorter than the digital pheromone trail which was common to the 10 synthetic CSV files with identical paths, it was not used to classify those paths. The expected number of subpaths differed from the actual number of subpaths (see table 20) because, instead of the digital pheromone discovery algorithm representing each complete path as a single digital pheromone trail (i.e. the expected result), a shorter digital pheromone trail was discovered which was common to the paths in all 11 synthetic CSV files. Since this digital pheromone trail was shorter than the length of the different path, 17 subpaths were classified instead of 11.

Although the results from testing the decision tree model for the nonincremental results (see figure 20) with a simple test program indicated its predicted values for heading angles had somewhat limited accuracy (with an average heading error of 21°), these results also indicated that the predicted values for ascent angles were highly accurate (with an average ascent angle error of 4°). The accuracy of the decision tree model at predicting ascent angles during the test suggest the decision tree model would be a better predictor of ascending and descending maneuvers performed by pilots of GA aircraft (e.g. descending turns) than the predictive algorithm in (Maedar, Morari, and

Baumgartner 2011). Also, the performance of the test program when applying the decision tree model was adequate, requiring an average time of only 0.38 seconds to generate a predicted maneuver when provided with the aircraft's current altitude above MSL and proximity to the nearest uncontrolled airport.

Thus, a predictive algorithm based on the decision tree model would have good performance scalability with increasing numbers of GA aircraft in the surrounding airspace. As the complexity of the decision tree model for such a predictive algorithm increases, more floating-point comparisons will be required for predicting the future path of each GA aircraft. However, modern CPUs, and especially GPU-based parallel architectures such as NVIDIA's CUDA, are highly efficient at floating-point comparisons. Furthermore, if a massive aircraft telemetry data set (e.g. a terabyte or more of raw data) was mined using these data preprocessing and mining algorithms, the accuracy of these probabilistic models could be substantially improved.

CHAPTER VI

CONCLUSION

Probabilistic models for the behavior of pilots of GA aircraft flying in Class E airspace were obtained by data mining a large FDM data set (i.e. an aircraft telemetry data set). The FDM data set was preprocessed separately using both nonincremental and incremental data preprocessing algorithms. The nonincremental result set was data mined on a Beowulf computational cluster using parallelized versions of GKA and EMEA. The incremental result set was data mined sequentially on a quad-core server.

The membership probabilities for digital pheromone trails (discovered with a novel application of an ant colony algorithm) in the clusters of lower and higher altitudes from both the nonincremental and incremental result sets did not show a strong bias towards either of the two proximity clusters. This suggests that, at both low and high altitudes, different sets of maneuvers would be performed by a pilot flying a GA aircraft depending on whether the GA aircraft is near to or far from an uncontrolled airport. The decision tree models for both result sets were similar. In each cluster in both of the decision tree models, one maneuver was performed much more frequently than any of the other maneuvers. Thus, the two result sets were similar which indicates stability in the results produced by the two different types of preprocessing algorithms (i.e. nonincremental and incremental preprocessing).

Future work for this research could include the following: 1) The FDM data set consisted exclusively of aircraft telemetry data from the University of North Dakota's flight training program. Thus, the set of discovered maneuvers may not be equivalent to the set of maneuvers which could be discovered from a national aircraft telemetry data set containing data for GA aircraft. 2) The volumes of controlled airspace surrounding airports were only approximated, since any GA aircraft within the maximum possible volume of controlled airspace surrounding any controlled airport was considered to be in controlled airspace. These controlled airspace volumes could be more accurately represented by separately specifying the FAA-mandated controlled airspace volumes for each controlled airport in the airport database. 3) The aircraft's altitude above mean sea level (MSL) was used as a data mining parameter, since this is the only type of altitude provided by FDM data. However, an aircraft's altitude above ground level (AGL) is likely to be a variable which is more strongly correlated with the maneuvers performed by the pilot of a GA aircraft. This could be estimated using a terrain database such as NASA's SRTM terrain database. 4) Not all variables which may have been correlated were considered, e.g. the prevailing weather conditions, the possibility of a mechanical failure, or the possibility of pilot error. Other data, such as METARs or an aircraft maintenance database could be correlated to the aircraft telemetry data set to include these parameters of the aircraft's operational environment. 5) The predictive power of the decision tree models could not be validated in collision avoidance scenarios, since the GPAR-RMS research project was discontinued prior to completion of this research. 6) GKA was not implemented with a mutation operator which is independently applied to each component of each candidate solution. If the implementation of GKA was extended

to use such a mutation operator, GKA could be guaranteed to eventually converge to the globally optimal solution for any input data set. 7) The parallelized GKA could be extended to allow crossover between candidate solutions on distinct nodes in the Beowulf computational cluster. 8) Since EMEA does not allow crossover between candidate solutions, EMEA is not guaranteed to eventually converge to the globally optimal solution. Although extending EMEA to include a crossover operator would be non-trivial, it would ensure EMEA would eventually converge to the globally optimal solution to the combinatorial optimization problem. 9) Using a distributed database server during parallel data mining, instead of a centralized database server, would increase data parallelism, thus considerably improving the performance of the parallelized GKA and EMEA.

In this dissertation, the author has documented his research on a novel application of an ant colony algorithm to the synthesis of aircraft telemetry data, which was then data mined to discover probabilistic models of the behavior of pilots of GA aircraft flying under VFR in Class E airspace. This is a novel application of an ant colony algorithm because existing research on ant colony algorithms has focused on their application to combinatorial optimization problems, not their application to pattern discovery. Ant colony algorithms could potentially be used to discover reoccurring patterns in any time series data, e.g. electromagnetic signals or financial markets.

Two clustering algorithms were studied for this research: 1) the Genetic K-Means algorithm, which combines a crisp clustering algorithm with a genetic algorithm, and 2) the Expectation-Maximization algorithm, which is a fuzzy clustering algorithm that estimates the probability of each data point belonging to each of the K clusters. The

Genetic K-Means algorithm was found to produce compact clusters without converging to local optima. The Expectation-Maximization algorithm was found to quickly produce fuzzy clusters, but also to frequently converge to local optima. However, the membership probabilities determined by the Expectation-Maximization algorithm provided good estimates of the number of naturally occurring clusters in the data sets.

APPENDICES

Appendix A
Glossary of Acronyms

ADM — Aircraft Data Miner

ADS-B — Automatic Dependent Surveillance – Broadcast

AGL — Above Ground Level

ANNWA — Artificial Neural Network Weight Analysis

ART — Adaptive Resonance Theory

ATC — Air Traffic Control

BIC — Bayesian Information Criterion

BIRCH — Balanced Iterative Reducing and Clustering using Hierarchies

COA — Certificate of Authorization

CSV — Comma-Separated Values

CUDA — Compute Unified Device Architecture

DSD — Data Structure Diagram

EC4.5 — Efficient C4.5

EKF — Extended Kalman Filter

EM — Expectation-Maximization

EMEA — Expectation-Maximization Evolutionary Algorithm

ETMS — Enhanced Traffic Management System

FAA — Federal Aviation Administration

FACTS — Flexible AC Transmission System

FDM — Flight Data Monitoring

FGKA — Fast Genetic K-Means Algorithm

GA — General Aviation

GAIK — Genetic Algorithm Initialized K-Means algorithm

GBAS — Graph-Based Ant System

GCS — Ground Control Station

GKA — Genetic K-Means Algorithm

GKMODE — Genetic K-Modes algorithm

GO — Ground Observer

GPAR-RMS — Ganged Phased Array – Risk Mitigation System

GPS — Global Positioning System

HGKA — Hybrid Genetic K-Means Algorithm

IDS — Information Display System

IGKA — Incremental Genetic K-Means Algorithm

IMM — Interacting Multiple Model

KMA — K-Means Algorithm

LAD — Logical Analysis of Data

LOI — Location of Interest

MILD — Moderate or Intense Low-oxygen Dilution

MSL — Mean Sea Level

NAS — National Airspace System

NASA — National Aeronautics and Space Administration

NFS — Network File System

NP — Nondeterministic Polynomial

P — Polynomial

PDF — Portable Document Format

PGAIK — Partition-Based Genetic Algorithm Initialized K-Means

PID — Proportional-Integral-Derivative

PSO — Particle Swarm Optimization

PVM — Parallel Virtual Machine

RCC — Range Control Center

RM — Risk Mitigation

RSO — Range Safety Officer

SFS — Sensor Fusion System

SLIQ — Supervised Learning In Quest

SQL — Structured Query Language

SRTM — Shuttle Radar Topography Mission

SVM — Support Vector Machine

TCAS — Traffic Collision Avoidance System

TSP — Traveling Salesman Problem

TWCV — Total Within-Cluster Variation

UA — Unmanned Aircraft

UAS — Unmanned Aircraft System

VFR — Visual Flight Rules

VQPCA — Vector Quantization Principal Component Analysis

WAAS — Wide Area Augmentation System

XML — Extensible Markup Language

Appendix B
Glossary of Aviation Terms

ADS-B — A protocol for regularly transmitting and receiving GPS-based telemetry data between proximate transceivers, such as aircraft with onboard ADS-B transceivers or base stations.

Class B Airspace — Largest airspace class which can surround a controlled airport.

Class E Airspace — National airspace below 18,000 feet MSL which is not Class G airspace.

Controlled Airport — An airport with an ATC tower.

FDM — A process whereby GPS-based telemetry data and other performance data for aircraft are archived by onboard equipment, such as the Garmin G1000, for later analysis.

General Aviation — A flight conducted by a private pilot, i.e. a pilot who is not associated with the military or a commercial airline company.

Instrument Flight Rules — A set of flight rules where the pilot uses only avionics instruments and directions from ATC to avoid potential conflicts with other aircraft.

Uncontrolled Airport — An airport without an ATC tower.

Visual Flight Rules — A set of flight rules for pilots where the pilot visually searches for nearby aircraft and avoids any potential conflicts with other aircraft.

REFERENCES

Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami. 1993. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering* 5, no. 6 (December): 914–925.

Al-Shboul, Bashar, and Sung-Hyon Myaeng. 2009. Initializing K-means using genetic algorithms. *World Academy of Science, Engineering, and Technology* 54 (June): 114–118.

Arya, Vijay, Naveen Garg, Rohit Khandekar, Adam Myerson, Kamesh Munagala, and Vinayaka Pandit. 2004. Local search heuristics for K-median and facility location problems. *SIAM Journal on Computing* 33, no. 3:544–562.

Berry, Michael J., and Gordon Linoff. 1996. *Data mining techniques for marketing, sales, and customer support*. New York: John Wiley and Sons, Inc.

Bianchi, Leonora, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing* 8, no. 2:239–287.

Boros, Endre, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. 2000. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering* 12, no. 2 (March/April): 292–306.

Botee, Hozefa M., and Eric Bonabeau. 1998. Evolving ant colony optimization. *Advances in Complex Systems* 1:149–159.

Callantine, Todd J. 2001. Analysis of flight operational quality assurance data using model-based activity training. *Proceedings of the Advances in Aviation Safety Conference and Exposition* (September).

Camargo, Suzana J., Andrew W. Robertson, Scott J. Gaffney, Padhraic Smyth, and Michael Ghil. 2007. Cluster analysis of typhoon tracks. Part I: General properties. *Journal of Climate* 20, no. 14:3635–3653.

—. 2007. Cluster analysis of typhoon tracks. Part II: Large-scale circulation and ENSO. *Journal of Climate* 20, no. 14:3654–3676.

Chandar, Kailash, Dinesh Kumar, and Vijay Kumar. 2011. Enhancing cluster compactness using genetic algorithm initialized K-means. *International Journal of Software Engineering Research and Practices* 1, no. 1 (January): 20–24.

Chang, Chih-Tang, Jim Z. C. Lai, and M. D. Jeng. 2010. Fast agglomerative clustering using information of K-nearest neighbors. *Pattern Recognition* 43:3958–3968.

Chaturvdei, Anil, Paul E. Green, and J. Douglas Caroll. 2001. K-modes clustering. *Journal of Classification* 18, no. 1:33–55.

Chaussabel, Damien, Charles Quinn, Jing Shen, Pinakeen Patel, Casey Glaser, Nicole Baldwin, Dorothee Stichweh, Derek Blankenship, Lei Li, Indira Munagala, Lynda Bennett, Florence Allantaz, Asuncion Mejias, Monica Ardura, Ellen Kaizer, Laurence Monnet, Windy Allman, Henry Randall, Diane Johnson, Aimee Lanier, Marilynn Punaro, Knut M. Wittkowski, Perrin White, Joseph Fay, Goran Klintmalm, Octavio Ramilo, A. Karolina Palucka, Jacques Banchereau, and Virginia Pascual. 2008. A modular analysis framework for blood genomics studies: Application to systemic lupus erythematosus. *Immunity* 29, no. 1 (July): 150–164.

Chen, Yixin, Guozhu Dong, Jaiwei Han, Benjamin W. Wah, and Jianyong Wang. 2002. Multi-dimensional regression analysis of time-series data streams. *Proceedings of the International Conference on Very Large Databases* (August): 323–334.

Cheng, Chun-Hung, Wing-Kin Lee, and Kam-Fai Wong. 2002. A genetic algorithm-based clustering approach for database partitioning. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* 32, no. 3 (August): 215–230.

Cho, Michael H., George R. Washko, Thomas J. Hoffmann, Gerard J. Criner, Eric A. Hoffman, Fernando J. Martinez, Nan Laird, John J. Reilly, and Edwin K. Silverman. 2010. Cluster analysis in severe emphysema subjects using phenotype and genotype data: An exploratory investigation. *Respiratory Research* 11, no. 1 (March): 1–9.

Cortez, Paulo, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems* 47 (May): 547–553.

Dahlbom, Anders and Lars Niklasson. 2007. Trajectory clustering for coastal surveillance. *Proceedings of the International Conference on Information Fusion* (July): 1–8.

Dalamagkidis, K., K. P. Valavanis, and L. A. Piegl. 2008. On unmanned aircraft systems issues, challenges, and operational restrictions preventing integration into the national airspace system. *Progress in Aerospace Sciences* 44, no. 7-8 (October/November): 503–519.

DeMarco, Tom. 1979. *Structured analysis and system specification*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society—Series B* 39, no. 1:1–38.

Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. 1996. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* 26, no. 1:1–13.

Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay. 2004. Clustering large graphs via the singular value decomposition. *Machine Learning* 56:9–33.

Duda, Richard O., Peter E. Hart, and David G. Stork. 2001. *Pattern classification*. 2nd ed. New York: John Wiley and Sons, Inc.

Eiben, A. E., E. H. L. Aarts, and K. M. Van Hee. 1991. Global convergence of genetic algorithms: A Markov chain analysis. *Lecture Notes in Computer Science* 496:3–12.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Federal Aviation Administration. Electronic code of federal regulations: Aeronautics and space. 2011. http://ecfr.gpoaccess.gov/cgi/t/text/text-idx?c=ecfr&tpl=/ecfrbrowse/Title14/14tab_02.tpl (accessed September 4, 2011).

—. Navigation services: Wide area agumentation system (WAAS). 2010. http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/waas (accessed September 7, 2011).

—. System safety handbook. 2008. http://www.faa.gov/library/manuals/aviation/risk_management/ss_handbook (accessed September 4, 2011).

—. Wide-area augmentation system performance analysis report. 2006. http://www.nstb.tc.faa.gov/REPORTS/waaspan17.pdf (accessed September 7, 2011).

Feng, Min, and Zhenyan Wang. 2011. A genetic K-means clustering algorithm based on the optimized initial centers. *Computer and Information Science* 4, no. 3 (May): 88–94.

Fisher, Marshall L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science* 27, no. 1 (January): 1–18.

Flarm Technology. Homepage. 2010. http://flarm.com (accessed March 24, 2012).

Fraley, Chris, and Adrian E. Raftery. 2002. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association* 97, no. 458 (June): 611–631.

Freitas, Alex A. 2002. *Data mining and knowledge discovery with evolutionary algorithms*. Berlin: Springer-Verlag.

Geist, Al, Adam Bequelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidyalingam S. Sunderam. 1994. *PVM: Parallel Virtual Machine: A user's guide and tutorial for network parallel computing*. Cambridge, Massachusetts: The MIT Press.

Glover, Fred. 1989. Tabu search—Part I. *ORSA Journal on Computing* 1, no. 3:190–206.

—. 1990. Tabu search—Part II. *ORSA Journal on Computing* 2, no. 1:4–32.

Greiner, Russell. 1996. PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence* 84 (July): 177–208.

Gutjahr, Walter J. 2000. A graph-based ant system and its convergence. *Future Generation Computer Systems* 16, no. 9 (June): 873–888.

—. 2008. First steps to the runtime complexity analysis of ant colony optimization. *Computers and Operations Research* 35, no. 9 (September): 2711–2727.

Haas, David, Joel Walker, and Lawrence Kough. 2008. Using flight data to improve operational readiness in naval aviation. *Proceedings of the American Helicopter Society Annual Forum* (May).

Hababeh, Ismail. 2012. Improving network systems performance by clustering distributed database sites. *Journal of Supercomputing* 59:249–267.

Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis. 2001. On clustering validation techniques. *Journal of Intelligent Information Systems* 17, no. 2 (December): 107–145.

Han, Yanfang, and Pengfei Shi. 2007. An improved ant colony algorithm for fuzzy clustering in image segmentation. *Neurocomputing* 70 (October): 665–671.

Handl, Julia, Joshua Knowles, and Douglas B. Kell. 2005. Computational cluster validation in post-genomic data analysis. *Bioinformatics* 21, no. 15 (May): 3201–3212.

Hunt, Jr., E. Raymond, W. Dean Hively, Stephen J. Fujikawa, David S. Linden, Craig S. T. Daughtry, and Greg W. McCarty. 2010. Acquisition of NIR-green-blue digital photographs from unmanned aircraft for crop monitoring. *Remote Sensing* 2, no. 1 (January): 290–305.

Hybels, Celia F., Dan G. Blazer, Carl F. Pieper, Lawrence R. Landerman, and David C. Steffens. 2009. Profiles of depressive symptoms in older adults diagnosed with major depression: A latent cluster analysis. *American Journal of Geriatric Psychiatry* 17, no. 5 (May): 387–396.

IIvanciuc, Ovidiu. 2007. Applications of support vector machines in chemistry. *Reviews in Computational Chemistry* 23:291–400.

Jiang, Daxin, Chun Tang, and Aidong Zhang. 2004. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 16, no. 11 (November): 1370–1386.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineers—Journal of Basic Engineering (Series D)* 82 (March): 35–45.

Kaufman, Leonard, and Peter J. Rousseeuw. 1990. *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley and Sons, Inc.

Kim, Dong Hwa, Ajith Abraham, and Jae Hoon Cho. 2007. A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences* 177, no. 18 (September): 3918–3937.

Kim, Jaejik, and L. Billard. 2011. A polythetic clustering process and cluster validity indexes for histogram-valued objects. *Computational Statistics and Data Analysis* 55:2250–2262.

Krishna, K., and M. Narasimha Murty. 1999. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* 29, no. 3 (June): 433–439.

Kullback, S. and R. A. Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* 22, no. 1 (March): 79–86.

Kumar, N. S. L. Phani, Sanjiv Satoor, and Ian Buck. 2009. Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA. *Proceedings of the IEEE International Conference on High Performance Computing and Communications* (June): 103–109.

Lai, Jim Z. C., and Tsung-Jen Huang. 2011. An agglomerative clustering algorithm using a dynamic K-nearest neighbor list. *Information Sciences* 181:1722–1734.

Leckebusch, Gregor C., Andreas Weimer, Joaquim G. Pinto, Mark Reyers, and Peter Speth. 2008. Extreme wind storms over Europe in present and future climate: A cluster analysis approach. *Meteorologische Zeitschrift* 17, no. 1:67–82.

Lee, H., Y. K. Park, G. Jang, and S. Y. Huh. 2000. Designing a distributed database on a local area network: A methodology and decision support system. *Information and Software Technology* 42:171–184.

Lewis, David D. 1998. Naïve (Bayes) at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science* 1398:4–15.

Li, Zhenhui, Jae-Gil Lee, Xiaolei Li, and Jiawei Han. 2010. Incremental clustering for trajectories. *Lecture Notes in Computer Science* 5982:32–46.

Loegering, G., and D. Evans. 1999. The evolution of the Global Hawk and MALD avionics systems. *Proceedings of the Digital Avionics Systems Conference*.

Lu, Yi, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J. Brown. 2004. FGKA: A fast genetic K-means clustering algorithm. *Proceedings of the ACM Symposium on Applied Computing* (March).

—. 2004. Incremental genetic K-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics* 5, no. 172 (October). http://www.biomedcentral.com/content/pdf/1471-2105-5-172.pdf (accessed May 30, 2011).

Lughofer, Edwin. 2008. Extensions of vector quantization for incremental clustering. *Pattern Recognition* 41, no. 3 (March): 995–1011.

Ma, Guanjun, Haibin Duan, and Senqi Liu. 2007. Improved ant colony algorithm for global optimal trajectory planning of UAV under complex environment. *International Journal of Computer Science and Applications* 4, no. 3:57–68.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability* 1:281–297.

Maeder, Urban, Manfred Morari, and Thomas Ivar Baumgartner. 2011. Trajectory prediction for light aircraft. *Journal of Guidance, Control, and Dynamics* 34, no. 4 (July/August): 1112–1119.

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge, Massachusetts: The MIT Press.

Marsh, Ronald, Kirk Ogaard, Micah Kary, John Nordlie, and Chris Theisen. 2011. Development of a mobile information display system for UAS operations in North Dakota. *International Journal of Computer Information Systems and Industrial Management Applications* 3:435–443.

McCall, Joel C., David P. Wipf, Mohan M. Trivedi, and Bhaskar D. Rao. 2007. Lane change intent analysis using robust operators and sparse Bayesian learning. *IEEE Transactions on Intelligent Transportation Systems* 8, no. 3 (September): 431–440.

McFadden, Kathleen L., and Elizabeth R. Towell. 1999. Aviation human factors: A framework for the new millennium. *Journal of Air Transport Management* 5:177–184.

Mehta, Manish, Rakesh Agrawal, and Jorma Rissanen. 1996. SLIQ: A fast scalable classifier for data mining. *Proceedings of the International Conference on Extending Database Technology* (March): 18–32.

Moore, Andrew. 1999. Very fast EM-based mixture model clustering using multiresolution kd-trees. *Proceedings of the Conference on Advances in Neural Information Processing Systems* 11 (November): 543–549.

National Aeronautics and Space Administration. Shuttle radar topography mission. 2009. http://www2.jpl.nasa.gov/srtm (accessed November 18, 2011).

Newman, M. E. J., and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics* 69, no. 2 (February).

Nonami, Kenzo. 2007. Prospect and recent research and development for civil use autonomous unmanned aircraft as UAV and MAV. *Journal of System Design and Dynamics* 1, no. 2:120–128.

Painter, Michael K., Madhav Erraguntla, Gary L. Hogg, Jr., and Brian Beachkofski. 2006. Using simulation, data mining, and knowledge discovery techniques for optimized aircraft engine fleet management. *Proceedings of the Winter Simulation Conference* (December): 1253–1260.

Panda, Sidhartha, and Narayana Prasad Padhy. 2008. Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design. *Applied Soft Computing* 8, no. 4 (September): 1418–1427.

Parente, A., J. C. Sutherland, B. B. Dally, L. Tognotti, and P. J. Smith. 2011. Investigation of the MILD combustion regime via principal component analysis. *Proceedings of the Combustion Institute* 33, no. 2:3333–3341.

Parunak, H. Van Dyke, Michael Purcell, and Robert O'Connell. 2002. Digital pheromones for autonomous coordination of swarming UAVs. *Proceedings of the AIAA Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference* (May).

Pearson, Karl. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine, Series 5* 50, no. 302:157–175.

Pelekis, Nikos, Ioannis Kopanakis, Evangelos E. Kotsifakos, Elias Frentzos, and Yannis Theodoridis. 2011. Clustering uncertain trajectories. *Knowledge and Information Systems* 28, no. 1 (July): 117–147.

Pelleg, Dan, and Andrew W. Moore. 2000. X-means: Extending K-means with efficient estimation of the number of clusters. *Proceedings of the International Conference on Machine Learning*.

Pizzuti, Clara. 2008. GA-Net: A genetic algorithm for community detection in social networks. *Proceedings of the International Conference on Parallel Problem Solving from Nature* 5199 (September): 1081–1090.

Plant, Claudia, and Christian Böhm. 2010. Parallel EM-clustering: Fast convergence by asynchronous model updates. *Proceedings of the IEEE International Conference on Data Mining Workshops* (December): 178–185.

Reza, Hassan, and Kirk Ogaard. 2011. Modeling UAS swarm system using conceptual and dynamic architectural modeling concepts. *Lecture Notes in Computer Science* 6828:331–338.

van Rooden, Stephanie M., Willem J. Heiser, Joost N. Kok, Dagmar Verbaan, Jacobus J. van Hilten, and Johan Marinus. 2010. The identification of Parkinson's disease subtypes using cluster analysis: A systematic review. *Movement Disorders* 25, no. 8 (June): 969–978.

Roy, Dharmendra K., and Lokesh K. Sharma. 2010. Genetic K-means clustering algorithm for mixed numeric and categorical data sets. *International Journal of Artificial Intelligence and Applications* 1, no. 2 (April): 23–28.

Rudolph, Günter. 1994. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks* 5, no. 1 (January): 96–101.

Ruggieri, Salvatore. 2002. Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering* 14, no. 2 (March/April): 438–444.

Segal, Paul. 2010. Analyzing extremely large volumes of business data: Practical considerations. Invited talk at the International Colloquium on Data Sciences, Knowledge Discovery, and Business Intelligence, Sydney, Australia. December 14.

Sheikholeslami, Gholamhosein, Surojit Chatterjee, and Aidong Zhang. 2000. WaveCluster: A wavelet-based clustering approach for spatial data in very large databases. *The Very Large Database Journal* 8, no. 3:289–304.

Sheng, Weiguo, and Xiaohui Liu. 2006. A genetic K-medoids clustering algorithm. *Journal of Heuristics* 12:447–466.

Snyder, Lawrence V., and Mark S. Daskin. 2006. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research* 174, no. 1 (October): 38–53.

Son, Jin Hyun, and Myoung Ho Kim. 2004. An adaptable vertical partitioning method in distributed systems. *The Journal of Systems and Software* 73:551–561.

Sorzano, C. O. S., J. R. Bilbao-Castro, Y. Shkolnisky, M. Alcorlo, R. Melero, G. Caffarena-Fernández, M. Li, G. Xu, R. Marabini, and J. M. Carazo. 2010. A clustering approach to multireference alignment of single-particle projections in electron microscopy. *Journal of Structural Biology* 171:197–206.

Stevens, S. S. 1946. On the theory of scales of measurement. *Science* 103, no. 2684 (June): 667–680.

Struyf, Anja, Mia Hubert, and Peter J. Rousseeuw. 1997. Clustering in an object-oriented environment. *Journal of Statistical Software* 1, no. 4:1–30.

Taniar, David, and John Goh. 2007. On mining movement patterns from mobile users. *International Journal of Distributed Sensor Networks* 3, no. 1 (January): 69–86.

Teh, Yee Whye, Hal Daumé III, and Daniel Roy. 2008. Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems* 20.

Tipping, Michael E. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1 (June): 211–244.

United States Geological Survey. SRTM quick start guide. 2009. http://dds.cr.usgs.gov/srtm/version2_1/Documentation/Quickstart.pdf (accessed November 18, 2011).

Velmurugan, T., and T. Santhanam. 2010. Computational complexity between K-means and K-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of Computer Science* 6, no. 3:363–368.

Vincenty, Thaddeus. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review* 23, no. 176 (April): 88–93.

Wagstaff, Kiri, Claire Cardie, Seth Rogers, and Stefan Schroedl. 2001. Constrained K-means clustering with background knowledge. *Proceedings of the International Conference on Machine Learning* (June/July): 577–584.

Weibel, Roland E., and Jr., R. John Hansman. 2004. Safety considerations for operation of different classes of UAVs in the NAS. *Proceedings of the AIAA Unmanned Unlimited Technical Conference, Workshop, and Exhibit*.

Weinberger, Kilian Q., and Lawrence K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10 (December): 207–244.

Yagiura, Mutsunori, and Toshihide Ibaraki. 2001. On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan* 32, no. 3:33–55.

Zhang, Liang, Fengming Zhang, and Yongfeng Hu. 2007. A two-phase flight data feature selection method using both filter and wrapper. *Proceedings of the International Conference on Software Engineering, Artificial Intelligence, and Parallel/Distributed Computing* 1 (August): 447–452.

Zhang, Shu Jia, Markus Hagenbuchner, Ah Chung Tsoi, and Alessandro Sperduti. 2009. Self organizing maps for the clustering of large sets of labeled graphs. *Lecture Notes in Computer Science* 5631:469–481.

Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An efficient data clustering method for very large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (June): 103–114.