Real-Time Analysis of Physiological Data and Development of Alarm Algorithms for Patient Monitoring in the Intensive Care Unit

by

Ying Zhang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

August 2003
[September 2003]

Author_____
Department of Electrical Engineering and Computer Science
August 29, 2003

Certified by_____
Peter Szolovits
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Real-Time Analysis of Physiological Data and Development of Alarm Algorithms
for Patient Monitoring in the Intensive Care Unit
by
Ying Zhang

Submitted to the
Department of Electrical Engineering and Computer Science

August 2003

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The lack of effective data integration and knowledge representation in patient monitoring limits its utility to clinicians. Intelligent alarm algorithms that use artificial intelligence techniques have the potential to reduce false alarm rates and to improve data integration and knowledge representation. Crucial to the development of such algorithms is a well-annotated data set. In previous studies, clinical events were either unavailable or annotated without accurate time synchronization with physiological signals, generating uncertainties during both the development and evaluation of intelligent alarm algorithms.

This research aims to help eliminate these uncertainties by designing a system that simultaneously collects physiological data and clinical annotations at the bedside, and to develop alarm algorithms in real time based on patient-specific data collected while using this system.

In a standard pediatric intensive care unit, a working prototype of this system has helped collect a dataset of 196 hours of vital sign measurements at 1 Hz with 325 alarms generated by the bedside monitor and 2 instances of false negatives. About 89% of these alarms were clinically relevant true positives; 6% were true positives without clinical relevance; and 5% were false positives. Real-time machine learning showed improved performance over time and generated alarm algorithms that outperformed the previous generation of bedside monitors and came close in performance to the new generation.

Results from this research suggest that the alarm algorithm(s) of the new patient monitoring systems have significantly improved sensitivity and specificity. They also demonstrated the feasibility of real-time learning at the bedside. Overall, they indicate that the methods developed in this research have the potential of helping provide patient-specific decision support for critical care.

Thesis Supervisor: Peter Szolovits, Ph.D.
Title: Professor of Computer Science and Engineering

To my grandparents

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Peter Szolovits. Every time I came to an unexpected result or an obstacle, Professor Szolovits had the power to get to its essence and to invigorate my research in a new light. He made me think hard about research questions in the most encouraging spirit, and he "finely combed" through drafts of this thesis to give me a valuable learning experience in scientific writing. His wisdom, style, and dedication to the education and growth of students tell much about why MIT is a special place. I am truly grateful for his guidance and support in my endeavors. I also want to especially thank Christine L. Tsien. Chris took me on as an UROP student and introduced me to the area of patient monitoring. She has been an exceptional mentor, big sister, and friend. Her work has generated much research interest in intelligent patient monitoring and inspired several ideas in this thesis. Her faith in me and her unwavering support have made the "little bumps on the road" easier to go over.

I am very grateful to Adrienne Randolph for giving me the opportunity to conduct research at the bedside at Children's Hospital in Boston. Dr. Randolph has been a principle investigator with genuine professionalism and valuable insights for our study and a role model for me. This thesis work would not have been realized without her continuing support. I would also like to thank Isaac Kohane. His faith in students and his zest for research have powered many to achieve their best. I cannot thank Dr. Kohane enough for his teaching, his interest in my education, and his support for this research at Children's Hospital. Each problem with the bedside monitor seemed to "melt away" as soon as David Martin started to tackle it. I am deeply indebted to Mr. Martin; without his expertise and continuing support for this research, we may not be able to get the physiological data from the bedside monitor. I want to sincerely thank all the nurses who have helped me in annotating the clinical events at the bedside. Their expertise and work ethics make me wish that every child who needs intensive care could have nurses like them. I am truly grateful to the patients and their families who have participated in our study. They kindly allowed me to sit by their bedside and showed great interest and support for the study. Simply thinking about them motivates me to do more research, to do good work.

I would like to thank Roger Mark, my graduate counselor, an invaluable mentor in patient monitoring, and the first person who showed me the similarities between physiological systems and electrical systems. He gave me the opportunity to attend Computers in Cardiology 2000 Conference, which opened my eyes to physiological signal analysis and intelligent patient monitoring. I am very grateful for Professor Mark's teaching, advice, and support in my education. I also want to thank John Guttag for teaching the seminar class *Medical Innovation and Engineering Research*, which motivated me to look beyond the problem of false alarms and into data integration, analysis, and knowledge representation for patient monitoring. I am genuinely grateful to John Wang, Larry Nielsen, and Mohammed Saeed for giving me the opportunity to work with them in the Patient Monitoring Division of then Agilent Technologies (now Philips Medical Systems). My summer internship allowed me to investigate the patient monitoring system from inside the box and to learn from them as well as from Andres Aquirre, Joanne Foster, Scott Kresge, and Susan Shorrock.

When I was in high school, I envisioned a life at MIT as studying in the library or working in lab until exhaustion, then taking a nap on a bench nearby, and getting up to work again. Only after getting to know Gerald Sussman did I truly understand what nerd pride really means. Professor Sussman taught me how to formulate a good research problem and to have a clear goal. He has shown me the power of having a broad range of knowledge, the zest for teaching, and

great humanity. I am deeply touched by his dedication to the education and growth of every student, and I am forever grateful for his teaching and guidance. I also want to especially thank Dennis Freeman, my undergraduate academic advisor. Professor Freeman effectively helped my transition from biology to engineering. His teaching in *Quantitative Physiology* gave me the first drill in scientific writing. As I was writing this thesis, I recalled several techniques that I had learned from him. Professor Freeman's advise, encouragement, and unwavering support have contributed much to my education and growth. I am very grateful to Arthur Smith, who gave me helpful advice on several occasions and is genuinely dedicated to both the undergraduate and graduate education in EECS. I also would like to thank the entire staff of EECS Undergraduate Office, especially Anne Hunter, Vera Sayzew, and Linda Sullivan, and of EECS Graduate Office, especially Marilyn Pierce. They really think for and care about students in imaginable and unimaginable ways.

I would like to thank all members, past and present, of my research group MEDG, who have each helped in their individual ways. I am particularly thankful of Patrick Cody, Fern DeOliveria, Meghan Dierks, Jon Doyle, Hamish Fraser, Ronida Lacson, William Long, Michael McGeachie, Andrew Nakrin, Lik Mui, Delin Shen, Yao Sun, Stanley Trepetin, and Min Wu for making me feel welcomed and for always being happy to help. I also want to especially thank Mojdeh Mohtashemi, whose doctoral thesis defense made me want to deliver my own thesis defense one day, who later became my officemate, a big sister, and a friend, and whose care and wisdom made all the difference.

I am very fortunate to have come to know Raymond Chan, Thomas Heldt, Ramakrishna Mukkamala, Shunmugavelu Sokka, and Wei Zong. Along with Mohammed Saeed, they have genuinely cared about my education and growth, and believed in me even when I was not so perfect. I look up to each of them in many ways, and I will always treasure their kindness and friendships.

I am eternally grateful to Farita McPherson for saving my foot just in time from being crashed by a utility vehicle ten months ago. I want to sincerely thank Deborah Brown, the orthopedic specialist at MIT Medical, and Michael Cassanni, my physical therapist at Kennedy Brothers, for getting me back on my feet and to walk again.

I would like to truly thank my parents for their unconditional love and unwavering support in everything I do. I also want to thank all my friends, especially Wesley Watters for his enduring faith in me and his unconditional friendship, which has really been a gift from heaven. This thesis work sprang during a period of tremendous growth, maturation, and discovery. I do not know how to thank enough all the people who have contributed to it, either directly or indirectly, or have touched my life in some way.

My work has been carried out in fond memories of He Jingzhi and Wu Xiangen, and with wonderful inspirations from Zhang Kongjia and Qin Quan. My grandparents brought me up since infancy, taught me to be a genuine person, and encouraged me to keep going forward in education and in doing something useful for others. I would like to dedicate this thesis to them.

This thesis is based upon work supported in part by a DARPA Research Grant and the Health Science and Technology Medical Engineering and Medical Physics Fellowship.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

In the intensive care unit (ICU) and other critical care settings, patients' physiological state needs to be monitored, but medical staff do not have the human resources and technical capabilities to perform this task continuously. Since the technology of monitoring astronauts' vital signs in space was transferred to the bedside in the 1960s, patient monitoring systems have become an indispensable part of critical care. Today, these systems can gather multiple physiological signals simultaneously and derive clinically important parameters.

Although the amount of information patient monitoring systems provide to medical professionals is more than ever before and still on the increase with improvements in computation power, memory, storage capability, and networking, the usability and usefulness of the information are less than desirable. The raw data contains measurement errors and noise from biosensors. Corrections for these errors and elimination of noise are difficult and limited without concurrent improvements of the measurement devices. Data integration and multi-parameter data analysis may be able to extract useful information from the imperfect raw data, but the state-of-the-art monitoring systems carry out limited data integration and analysis for effective decision support.

One symptom of this lack of data integration and analysis is the generation of false alarms. Patient monitoring systems for critical care should alert caregivers when the patient requires immediate attention. Several studies in the 1990's, however, indicated that the vast majority of the alerts generated by automated monitors were inappropriate. A study in the multidisciplinary ICU of a pediatric teaching hospital, however, showed that 86% of total 2942 alarms during 298 monitoring hours over a ten-week period were false positives; an additional 6% were found to be clinically irrelevant true alarms; only 8% of all alarms were true alarms with clinical significance [45]. Another study in a similar pediatric ICU found that 68% of alarms were false, 26.5% were induced by medical procedures, and only 5.5% were significant true alarms that resulted in

change in therapy. [25] In critical care settings for adults, false alarm rate could be even higher – as high as 94% was reported in a standard cardiac ICU. [21]

To reduce false alarm rates in the ICU, researchers have been pursuing two paths: (a) creating better sensors that reduce measurement noise and that "notice" systematic faults such as wires being disconnected; and (b) developing "intelligent" alarm algorithms for patient monitoring. Methods such as neural networks, classification trees, fuzzy logic, and other artificial intelligence techniques have shown potential for reducing false alarm rates. These techniques may also be used in improving more general aspects of patient monitoring, such as data integration and analysis, prognosis generation, and decision support.

1.2 Problem Statement

The conventional approach to developing, evaluating, and refining physiological models or algorithms for decision support is based on a retrospective analysis of physiological data with clinical annotations that were collected around the same time as the data. There are several limitations to this approach. First, physiological data and clinical annotations are collected by separate mechanisms and often poorly synchronized as a result. Second, because physiological data and clinical annotations have different granularity, and the time range of a clinical event is often difficult to capture, even with time synchronization, correlation between two different types of data can be ambiguous. Third, in the critical care setting, it is difficult to record everything that can potentially be useful in retrospective research, so clinical annotations are collected based on assumptions about future research needs, and retrospective studies often find that they need additional clinical information and thus cannot reconstruct the necessary clinical context to interpret a past event properly. Thus, most developments of "intelligent alarm algorithms" contain significant uncertainties and assumptions that may not be clinically valid; as a result, evaluations of these algorithms also yield results that are still speculative.

To address these problems, we have developed a system that enables data analysis and algorithm development for patient monitoring in real time. In this thesis, we first demonstrate the feasibility of real-time data analysis at the bedside and concurrent clinical annotation. Then we present the development and evaluation of alarm algorithms in real-time, using machine learning techniques.

1.3 Thesis Organization

In the remainder of this thesis, we begin by motivating and describing a system for synchronized collection of physiological signals and clinical annotations in Chapter 2. We will also discuss design considerations, constraints on such systems, and its utility. Chapter 3 presents real-time modeling at the bedside. It describes methods for developing alarm algorithms using machine learning techniques in real time. Then, in Chapter 4, we review related work. Chapter 5 concludes this thesis with a summary of the studies and findings in our research. We will also discuss questions that have arisen form our research and ideas for future work.

Chapter 2

A System for Synchronized Collection of Physiological Signals and Clinical Annotations

This chapter describes a system for synchronized collection of physiological signals and clinical annotations at a bedside at a standard pediatric intensive care unit. Its design purpose is to support real-time analysis of physiological signals and real-time development of alarm algorithms for patient monitoring in critical care settings.

2.1 Motivation

To develop and evaluate models and algorithms for intelligent patient monitoring, we must have real patient data and a way to reconstruct the clinical context under which these data are generated. In other words, we need to obtain physiological measurements or signals from the patient's monitor and to know what is going on with the patient when these measurements become available. Yet, the reconstruction of the clinical context is a nontrivial task. Although experienced physicians can form a hypothesis about the patient's state or what could be happening to the patient by examining physiological data such as an electrocardiagram and blood pressure readings, only with adequate clinical information, such as the course of therapy and events at the bedspace, can he or she validate this hypothesis. Thus, a well-annotated dataset that contains both physiological data and clinical annotations is key to the development of intelligent patient monitoring systems.

There are two major requirements for a well-annotated dataset. First, physiological data must be accompanied by clinical information that enables the reconstruction of clinical events that could affect the current and future values of the data or could provide explanations for the physiological data from the past. Second, both the physiological data and clinical information should be time-stamped such that they are synchronized in time and can be accurately correlated.

In previous studies, data acquisition in the intensive care unit focused primarily on collecting physiological signals from the bedside monitors. Little information about the state of the patient and clinical events at the bedside were recorded. The reasons are straightforward. First, only until recently has the revolution in computation power and storage capacity enabled researchers to record large amount of data and to allow computers to analyze these data in a timely fashion. Second, many researcher did not realize the importance of clinical information to modeling physiological systems until they had encountered the limitations of retrospective annotation by human experts. The third reason, which still hinders much research today, is the difficulty of accessing clinical information, due to either practical reasons (e.g. clinical event recording requires a trained person and is labor intensive) or legal concerns (e.g. clinical annotations could contain confidential patient information).

A study by Moody *et al.* foresaw the importance of clinical information and recorded clinical data such as laboratory reports, physicians' and nurses' progress notes, and administration of medications through the hospital's clinical information systems [30] Another study by Tsien *et al.* prospectively recorded clinical events at the bedside. [44] In both studies, however, clinical information was recorded separately from the physiological data and time-stamped by different clocks. As a result, the exact correlation between physiological data and clinical information could not be achieved. Assumptions about the correlation between the two forms of data had to be introduced, but they could not remove the uncertainties in the development and evaluation of models and algorithms based on these data.

Difficulties in synchronizing physiological data and clinical information come from two sources. First, when a patient's condition deteriorates, the clinicians are fully occupied caring for the patient instead of writing notes. In fact, they write progress notes only when the patients do not need their attentions or at the end of their shifts. (Personal observation) Thus, the event recordings in the physicians' and nurses' notes usually lag behind the actual events and cannot be accurately correlated with physiological data. Furthermore, when a clinician records an event that happened hours before, his or her memory of it might be less vivid, and the information that gets recorded about the event might lack useful details. A trained observer, however, could take advantage of the fact that the clinicians usually could talk when they carry out procedures to avoid information loss. The observer could sit at the bedside to record the clinicians' response to and verbal description of a clinical event as it happens. He or she could also ask for additional

information that might be useful for reconstructing the event. The design of our system supports the use of such observer-recorded annotations.

The second source arises because it is difficult during an alarm event to determine the significance of the alarm, which may become apparent only some time after the event ends. The user interface that collects clinical information, therefore, must remain available to the user after the event. However, because alarms may follow each other in succession, it may happen that multiple alarm events are awaiting entry of their clinical interpretation at the same time. Therefore, the user interface must keep clear to its users just which annotation corresponds to which event and time interval. Our system design also addresses this requirement.

In the first part of this research, we addressed the problem of data synchronization by building a system for synchronized data collection and clinical annotations. We designed this system to be used by a trained observer to collect data at the bedside in real time. We also describe the expansions of this system for real-time trials of alarm algorithms. In the rest of this thesis, we refer to the entire system as PAAT, for Prospective Alarm Algorithm Trial system.

2.2 Methods

2.2.1 Overview

The "ideal" data acquisition system for our purposes is a powerful workstation with full networking capabilities for use with any bedside monitor in any critical care setting. It can communicate with different brands of bedside monitors and obtain physiological data using common standards through an RS232 interface or from the ICU's information system. It has enormous bandwidth on both the serial and the Ethernet lines to receive all the data that a bedside monitor has, and possibly from nearby monitors as well. It has enough computational power to receive, store to the database, and simultaneously analyze and learn from the data, all at once.

Not surprisingly, all the assumptions of our "ideal" system are violated in the actual situation for which our system has been used. In this section, we describe the actual situation we faced and how we tackled each constraint to achieve the purpose of our system. Figure 2.1 is a block diagram of the system's components:

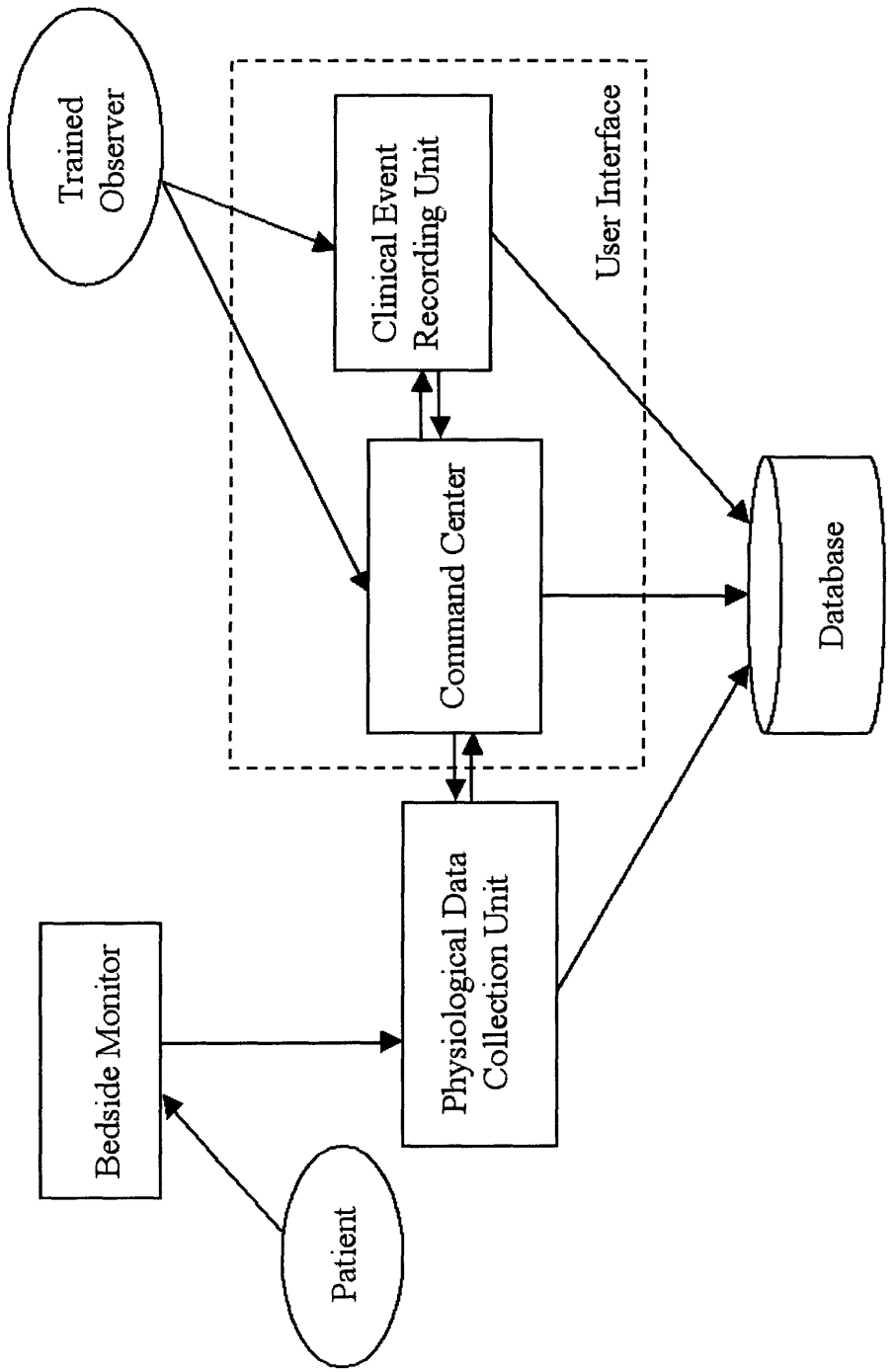


Figure 2.1 System diagram / Data flow chart

Our system is divided by function into three components: a physiological data collection unit, a clinical event recording unit, database, and a command center. Design considerations include system synchronization, speed, modularity, and multiple tasking.

2.2.2 Physiological Data Collection

2.2.2.1 Bedside monitor

We carried out this research in collaboration with a standard pediatric medical ICU. Because the patient monitoring systems of each brand have their own proprietary platform, operating system, and network, we had to design our system specifically for the bedside monitor that was designated for our research in this ICU. This monitor belongs to the HP Viridia Neonatal Component Monitoring System (CMS) series. It was manufactured by the former medical instruments division of Hewlett Packard, which then became part of Agilent Technologies, and now Philips Medical Systems. This monitor has optimized features for neonatal care and is configurable for pediatric and adult patient monitoring.

2.2.2.2 Data Access

In an ideal situation, we would like to access data from any one of the eighteen bedside monitors in our collaborating ICU, but the ICU's central information system was not available for use by our project; thus, we designed our system to communicate with the bedside monitor that was designated for our research via the RS232 interface. An RS232 dual interface card (Option 13 for CMS Model 1077A) was installed in the monitor. A variety of HP printers can connect to it to produce paper reports. With a special cable and a set of software that correctly configures the connection, it also allows a personal computer to access all waveforms (e.g. electrocardiogram), numerics (e.g. heart rate), and alarm status data from the monitor.

A printer cable with a 25 pin D-type female connector and a 9 pin female connector for IBM AT-LASER plotter has been used to connect the RS232 interface card with the serial port of a standard laptop. Although this particular kind of cable might not be the only kind that can

establish a connection with the correct interface lines between the serial port and the RS232 interface card, among many cables that we have tested, it is the only one that works.

The manufacturer of this monitor provides a programming guide, a library file, some source files, and a demo program that shows how to obtain data through the RS232 interface. Based on the source code of the demo program, we built an application, named CMSCOM, to configure the RS232 connection and to receive data. The library file, which is called *mecif.lib,* contains the definition of each data structure and functions for communication with the RS232 interface. Because the data structures were defined in a 16-bit format and compilation in a 32-bit operating system prevented CMSCOM from correctly interpreting the data values after linking to the library file, CMSCOM was compiled in 16-bit Turbo C in Windows 95.

CMSCOM first communicates with the bedside monitor and checks which data are available for access. All data are transmitted in packets called messages. Each message type is identified by a message ID. Message ID is uniquely specified by six items, as listed in Table 2.1.

SourceID	Specifies the measurement module that transmits the message (e.g. ECG module, invasive pressure module)
SourceNo	Differentiates between different modules with the same SourceID (e.g. the patient has two different pressure sensors/measurement modules)
ChannelID	Specifies the output quantity (e.g. ECG wave or heart rate numeric)
ChannelNo	Differentiates between different channels (e.g. ECG wave may outputs three wave channels, each of which represents a different ECG lead)
MsgType	Contains the syntax and semantic of the message (i.e. specify the data structure)
Layer	Differentiates between messages that have the same meaning in different steps of data processing

Table 2.1: Message ID structure [17]

After getting a list of message Ids for available data, it sends a request to the monitor for continuously receiving a set of desired available data.

When a message is received from the monitor, it is identified by its message ID, and the parameter value contained in the message is saved into a designated text file. We chose to use text files to transfer data instead of directly transferring data into the database because CMSCOM is a 16-bit application and its direct compatibility with various database engines is limited.

2.2.2.3 Bandwidth Consideration

Ideally, we would like to collect all the physiological data that the monitor could provide. However, the amount of data we can collect is limited by the bandwidth of the communication channel. The type and the number of physiological signals that can be collected from the bedside monitor via its RS232 interface are governed by bandwidth cost of each signal, the baudrate of the RS232 ports on the monitor, the baudrate of the serial port on the laptop, and the number of escape sequences in the messages. Exceeding the allowable bandwidth could cause an overflow of the monitor's transmission buffer and loss of data.

The CMS monitor collects two forms of physiological signals: waveforms and numerics. Waveforms are sampled at either 500 Hz (electrocardiogram) or 125 Hz (pressures, arterial oxygen saturation, respiration) Numerics (e.g. heart rate, respiratory rate) are derived from the waveforms once every 1024 milliseconds. Table 2.2 lists the types of monitor data that are available for access and their minimum bandwidth cost.

Message Type	Period (milliseconds)	Length (bytes)	Minimum Bandwidth Cost (bytes/second)
Waveform	32	19-43	1376
Waveform Support	1024	33-133	133
Numeric	1024	39-135	135
Alarm Information	1024	13-61	61

Table 2.2: Bandwidth cost summary for each data type [17]

There are two RS232 ports on each RS232 card, and each monitor can have two cards. Each RS232 port could be set at one of the three baud rates: 9,600 baud, 19,200 baud, and 38,400 baud, with the constraint that if one port is set at the maximum baud rate, the other port on the same

card would only support data output up to 600 bytes/sec. Since the amount of escape sequences is unknown, the maximum amount of data CMSCOM requests from the monitor should be under the byte rate limit for the baud rate setting of the monitor. These limits are listed in Table 2.3. (HP Programming Guide)

Selected Baudrate (bits/sec)	Byte Rate Limit (bytes/sec)
9600	920
19200	1850
38400	3750

Table 2.3 Limits on byte rates [17]

Table 2.2 and Table 2.3 show that in order to collect all available physiological signals, there must be two RS232 interface cards to provide 4 ports, two of which must be set at the maximum baud rate. However, we have only one serial port on our laptop, and this standard serial port has a minimal buffer. If the incoming data arrives at 38,400 bits/second, it would overflow the buffer and lose data.

The four RS232 ports in combination can transmit 8700 bytes every second. This byte rate allows a maximum of six waveforms that are sampled at 500 Hz being collected simultaneously or five such waveforms plus all the numeric data. Since most patients usually have no more than four or five such waveforms being measured or displayed simultaneously on the monitor, a previous study was able to receive most data from the bedside monitor for most patients using two RS232 cards and an expanded serial connection with smart serial cards on a workstation. [30]

We considered a similar expansion, but since, for research at the bedside, we were restricted to using only a laptop, which limits the extent of expansion and computational power, we were not able to receive and analyze all the waveforms in real time. Thus, we focused on getting numeric data and alarm information as a first step. The final baud rate was set at 19,200 bits/second for one RS232 connection. In the future, we would like to upgrade our system to receive all available physiological data.

2.2.3 Command Center

The functions of the command center include data transfer, data synchronization, and central control of all the components of PAAT. It features the main user interface, which hosts patient information entry and command buttons. Figure 2.2 illustrates the main user interface.

The command center is written in Visual Basic .NET. The choice of this language was made based on the ease of creating and modifying the user interface, availability of multiple timers, and direct access to SQL server and other database engines using the functionality ADO.NET.

At selected clock times, the command center obtains physiological data from data text files and transfers the data into the database. This clock is based on the system time of the laptop. The command center uses this clock to check if any physiological data have been received in the past two seconds. If yes, it time-stamps each set of data with the time of this clock and records the data into the database. Otherwise, it substitutes a special value for missing data, time stamps it, and records it in the database.

2.2.4 Clinical Event Recording

2.2.4.1 Event Annotation

Clinical event recording is done in four cases: 1) the bedside monitor sounds an alarm; 2) an alarm algorithm under investigation displays an alarm; 3) the patient becomes irritated and requires immediate attention when no alarm occurs; 4) drugs are being administered or discontinued.

CMS Alarm Event. When the bedside monitor sounds an alarm and sends alarm information to the system, an annotation box, as in Figure 2.3, is displayed on the user interface.

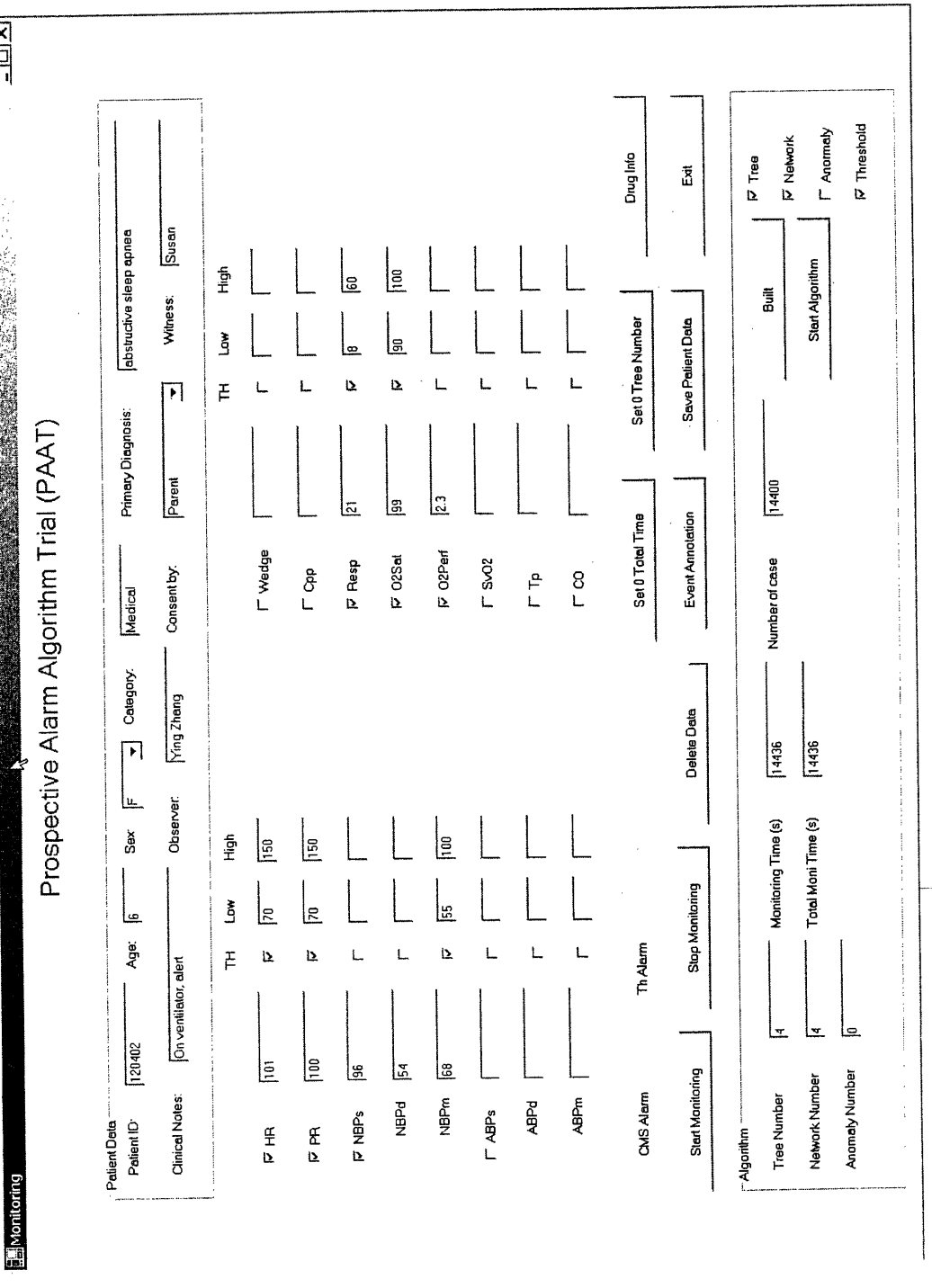


Figure 2.2 Main user interface

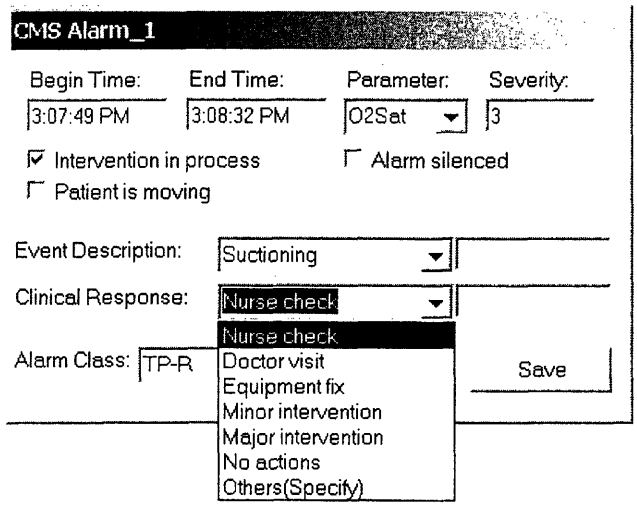


Figure 2.3 CMS alarm message box

The field "Begin Time", "Alarm Type" and "Alarm Severity" are automatically filled when the annotation box appears. "Begin Time" is the time point when the alarm starts. "Alarm Type" is the physiological parameter that triggers the alarm. "Alarm Severity" is given by the monitor according to the following prioritization scheme [17]:

3: Red Alarms – identify apnea, extreme bradycardia, extreme SpO2 desaturation, FiO2 low oxygen, asystole, ventricular fibrillation and pressure disconnect conditions;

2: Yellow alarm – alert clinicians when the preset alarm limits are exceeded;

1: Short yellow alarm – alert clinicians of abnormal but not life-threatening arrhythmia.

Technical alerts (INOPS), which are triggered by signal quality problems, equipment malfunction, a measurement setup problem, or an ongoing calibration, are recorded into the database but not annotated because they are treated differently by caregivers and have distinct sounds and known causes.

During or after the alarm, a trained observer at the bedside records whether the patient is moving, whether a medical procedure is in process, and how the medical staff responds to the alarm, such as checking the patient, adjusting sensors, or silencing the alarm without any intervention.

"End Time" is automatically filled in when the alarm stops. At this time, a trained observer asks the nurse or physician at the bedside to classify the alarm into one of the three categories, as adopted from work by Tsien [48]:

- TP-R (True Positive, Clinically Relevant)

  These alarms are appropriate given the actual data value, and the patient's condition requires prompt attention. For example, a patient suddenly develops a dysrhythmia with a heart rate of 200 beats per minute (bpm), the ECG measures 200 bpm, the monitor is set with an upper threshold at 160 bpm, and the monitor sounds an alarm.

- TP-I (True Positive, Clinically Irrelevant)

  These alarms are appropriate given the input data value as compared to the set threshold value, but the patient's condition had not changed in a way that required additional medical attention. The sounding of the alarm thus has no clinical relevance. For example, a patient suddenly has a heart rate of 200 beats per minute (bpm), the ECG measures 200 bpm, the monitor is set with an upper threshold at 160 bpm, but the increase in heart rate is due to patient's excitement upon seeing his or her family, so he or she does not require medical attention. For this example, we should note that children, especially sick children, generally have a higher heart rate and a higher tolerance of fluctuation in heart rate than adults do.

- FP (False Positive)

  These alarms are inappropriate given the input data value. For example, a patient has a heart rate of 80 bpm. The ECG electrodes are manipulated. Although the patient's heart rate stays at 80 bpm throughout this period, an alarm sounds. The alarm was false because the reported value did not reflect the patient's condition.

This classification can be revised if any events in the next 30 minutes indicate the classification is incorrect. The need for this update mechanism will be elaborated in the discussion on the gold standard for alarm classification in section 2.2.7.

When the entire message box is completed, the observer would save the annotation and close the message box.

Algorithm Alarm Event. When an alarm algorithm under investigation generates an alarm and reports it, an annotation box, as in Figure 2.4, is displayed on the user interface.

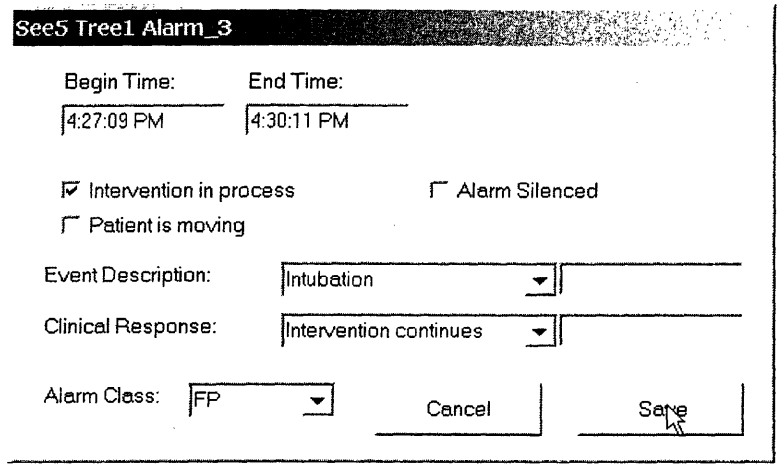


Figure 2.4 Algorithm alarm message box

The field "Begin Time" is automatically filled when the annotation box appears. "Begin Time" is the time point when the alarm starts. The annotation is done in the same manner as that for the CMS alarm message box except there are no parameter and severity specifications because the alarm algorithms that we have developed classify the patient's condition instead of an individual physiological signal, and the alarms generated by these algorithms do not have a priority assignment. In future research where new algorithms generate more refined decisions or require additional information, the algorithm alarm message box can easily be modified to facilitate necessary annotations.

Non-Alarm Event. If the patient starts to show irritation or serious discomfort when no alarms are generated, the trained observer starts a Non-Alarm event annotation entry box to record this event. If the nurse deems this event to be clinically relevant, this event would be classified as a false negative. Figure 2.5 shows the layout of this annotation entry box.

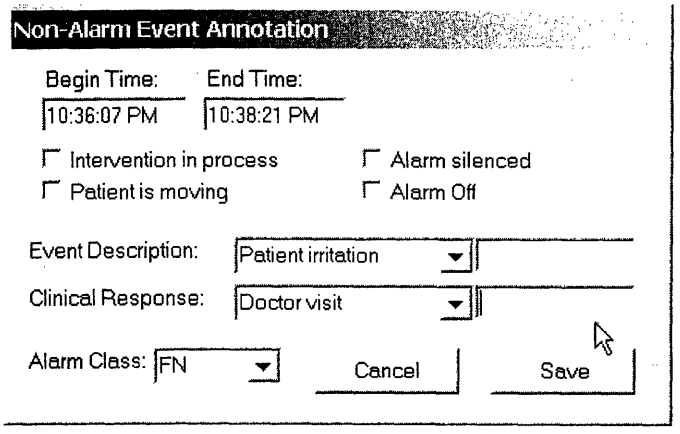


Figure 2.5 Non-Alarm event annotation entry box

Medication Annotation. An annotation entry box is also available for the observer to record when a drug is administered, which could significantly alter some of the physiological data.

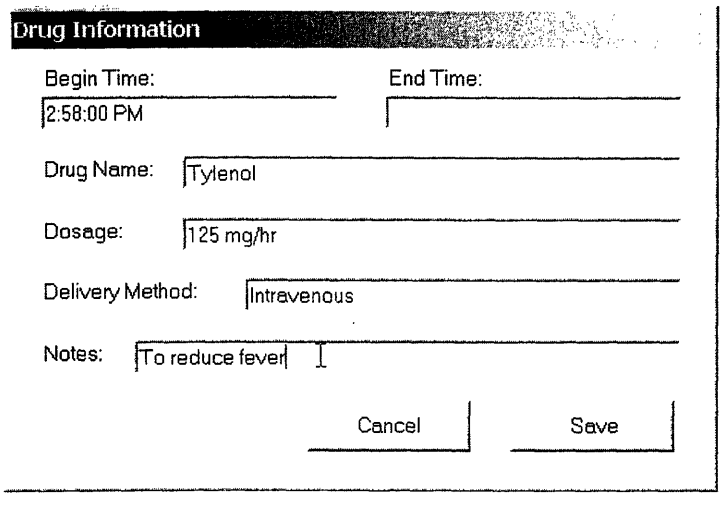


Figure 2.6 Drug information entry box

2.2.4.2 Multiple threading

Events, especially clinically relevant events, are likely to occur multiple times. They might be concurrent, overlapping, or in series. Especially during such times, the caregivers are busily involved in taking care of the patients, so they might not be available to answer questions or they do not have adequate information to classify the alarm yet. Thus, our system must allow multiple alarm messages to coexist and multiple annotation boxes to remain open in order to facilitate the annotation process.

The design of multiple threading in PAAT is illustrated in Appendix A. Figure A.1 shows the primary thread and the main thread. The primary thread serves the main user interface, initializes new threads, and relays commands to other threads in the system. The main thread serves the monitoring functions of the system. It has a timer running at one second and automatically reads, processes, and stores the physiological data. It also generates new threads for the annotation processes for the monitor's alarms and the algorithms' alarms during each cycle.

As shown in Figure A.2, there is a thread for each alarm-generating entities (i.e. the bedside monitor and each alarm algorithm). PAAT also runs a threshold alarm algorithm in parallel with the CMS alarm system to mimic the alarm algorithm in the previous generation of patient monitoring systems. It immediately generates an alarm when a measurement value exceeds the parameter's upper or lower threshold. The purpose of running this algorithm will be elaborated in Chapter 3.

When an alarm is generated, a new thread is initialized and a message box appears on the screen. This thread will remain active until the user finishes entering the annotations about the alarm and closes the message box. Thus, concurrent or overlapping events can be annotated simultaneously and as clinical information becomes available. Figure A.3 illustrate this capability of PAAT for CMS alarm annotations and the threshold alarm annotations. Figure A.4 gives the similar illustration for two sample alarm algorithms.

Each Non-Alarm event annotation and drug information entry also has its own thread. The purpose of these thresholds is to allow multiple concurrent annotation sessions as well as separation from the automated monitoring process.

2.2.5 Database

The database is designed using SQL server desktop version. SQL Server provides more capability and reliability than database engines such as Microsoft Access and is simpler to use than engines such as Oracle. It is easy to backup and restore different databases, allowing data from each collection session to exist in one database. More importantly, SQL Server has enforced transactions mechanism, which ensures the physical integrity of each transaction. For example, it provides lock facilities that preserve transaction isolation and logging facilities that ensure transaction durability. Even if the server hardware, operating system, or SQL Server itself fails, SQL Server uses the transaction logs, upon restart, to automatically roll back any uncompleted transactions to the point of the system failure. Transaction management features enforce transaction atomicity and consistency. After a transaction has started, it must be successfully completed, or SQL Server undoes all of the data modifications made since the transaction started. These features ensure data integrity and prevent partial or corrupted record from being stored.

Table 2.4 lists the tables in the database for the physiological data collection unit. Table 2.5 lists all the tables for clinical event recordings.

HR (heart rate)
PR (pulse rate)
RR (respiratory rate)
ABP (arterial blood pressure)
NBP (cuff blood pressure)
O2Sat (arterial oxygen saturation)
Perf (oxygen perfusion)
Tp (temperature)
CO (cardiac output)
Wedge (pulmonary wedge pressure)
CO2 (carbon dioxide)
AllNu (all numeric)
AllNuAvg (features derived from the numeric)

Table 2.4 Physiological data tables

CMS Alarm
Tree1 Alarm
Tree 2 Alarm
.
. (Algorithm alarms)
.
Threshold Alarm

Table 2.5 Clinical event recordings tables

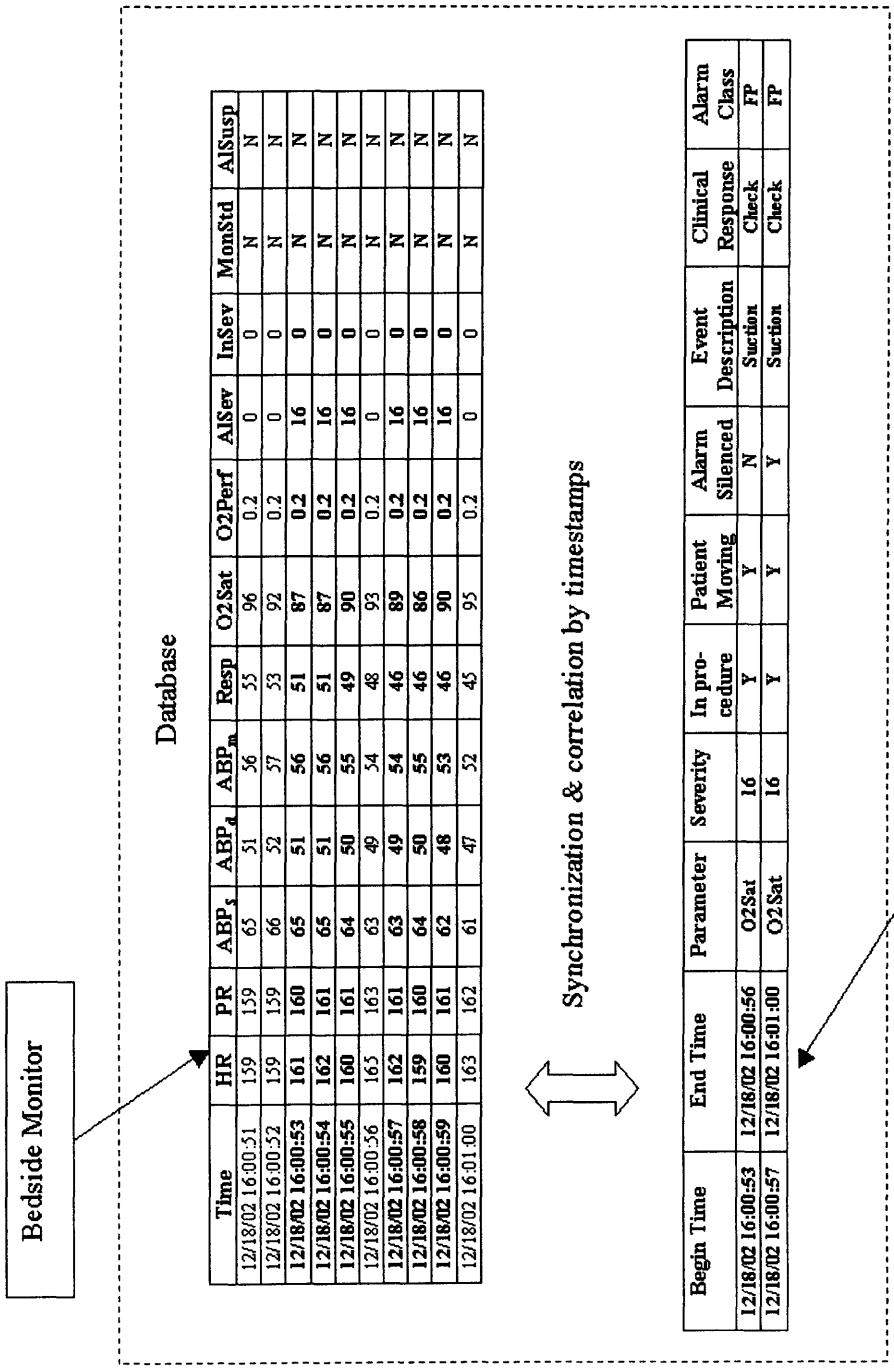
2.2.6 Time Synchronization

In the previous section, we described the time stamps for all the data records. Time synchronization between two different physiological parameters is achieved by matching their time stamps. Time synchronization between event recordings is achieved by comparing the beginning times and the two ending times. Synchronization between physiological data and event recording is achieved by going over all the time intervals defined by the begin and end time of each event and identifying those events whose time interval spans the time stamp of the physiological data, as illustrated in Figure 2.7.

2.2.7 Gold Standard for Alarm Classification

The most difficult part of annotating an alarm-sounding event in the ICU is the accurate classification of the alarm. Even if the definitions of each class, such as those described earlier in the chapter (i.e. TP-R, TP-I, or FP), are clearly disjoint, the complexity of the event or how an alarm is classified, by whom, and on what basis could make the classification process ambiguous.

In their studies, Tsien, Lawless, and Koski all asked the patient's nurse to classify the alarms. The gold standard for alarm classification is then the nurses' comments. From personal communication with the director of a standard pediatric ICU, we learned that the patient's nurse may not serve as a sufficient gold standard because nurse training focuses on routine care and reaction to clinical events rather than on knowledge that enable nurses to detect trends and to



Time	HR	PR	ABP_s	ABP_d	ABP_m	Resp	O2Sat	O2Perf	AlSev	InSev	MonStd	AlSusp
12/18/02 16:00:51	159	159	65	51	56	55	96	0.2	0	0	N	N
12/18/02 16:00:52	159	159	66	52	57	53	92	0.2	0	0	N	N
12/18/02 16:00:53	**161**	**160**	**65**	**51**	**56**	**51**	**87**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:00:54	**162**	**161**	**65**	**51**	**56**	**51**	**87**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:00:55	**160**	**161**	**64**	**50**	**55**	**49**	**90**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:00:56	165	163	63	49	54	48	93	0.2	0	0	N	N
12/18/02 16:00:57	**162**	**161**	**63**	**49**	**54**	**46**	**89**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:00:58	**159**	**160**	**64**	**50**	**55**	**46**	**86**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:00:59	**160**	**161**	**62**	**48**	**53**	**46**	**90**	**0.2**	**16**	**0**	**N**	**N**
12/18/02 16:01:00	163	162	61	47	52	45	95	0.2	0	0	N	N

Begin Time	End Time	Parameter	Severity	In procedure	Patient Moving	Alarm Silenced	Event Description	Clinical Response	Alarm Class
12/18/02 16:00:53	**12/18/02 16:00:56**	**O2Sat**	**16**	**Y**	**Y**	**N**	**Suction**	**Check**	**FP**
12/18/02 16:00:57	**12/18/02 16:01:00**	**O2Sat**	**16**	**Y**	**Y**	**Y**	**Suction**	**Check**	**FP**

Figure 2.7 Synchronization between physiological data and event annotations

foresee a deterioration in the patient's condition. Other studies propose to verify the nurse's classification with the attending physician, but since the attending physician may not have followed the course of this particular patient on a minute-to-minute basis, the physician's judgment may not be a good gold standard either.

So far, there is not a reliable gold standard for alarm classification in the literature. Thus, in this study, we construct a 2-tier gold standard: we first ask the patient's nurse or a physician who has been closely following the patient to classify the alarm sounding event. Then, we use the patient's condition in the next 30 minutes to revise the human expert's classification as needed. For example, if an alarm for bradycardia (i.e. heart rate less than the low limit) is classified as a false positive, and the patient becomes persistently hypotensive in the next 30 minutes, we would revise the classification to either clinically relevant true positive or a class chosen by all physicians and nurses at the bedside.

2.2.8 Evaluation Procedure

Evaluation of the system was done in three stages: simulation, implementation, and usage. Testing was done at each stage. During simulation, both unit level testing and system-level testing were performed. A simulation database was constructed as the source of patient data. At the beginning of implementation, the RS232 connection was first tested separately from the rest of the system. After it was shown to function normally, system-level testing was performed. After this test succeeded, the system was used to run alarm algorithms in real time in a completely functional phase, which facilitated stress testing.

2.2.9 Implementation

This system has been implemented and used at the Multidiciplinary Intensive Care Unit (MICU) of Children's Hospital in Boston. The study was approved as a part of a research protocol by the Institutional Review Board of Children's Hospital. A patient consent form is required to ensure

that patients and their families are comfortable with the presence of the observer. Patient confidentiality and privacy have been protected according to the hospital guidelines.

A sample protocol for each session is as follows:

1) After obtaining patient's/guardian's consent, the investigator connects the portable computer for the study to the bedside monitor via a cable that is installed and tested by technicians from the Biomedical Department of Children's Hospital.
2) The software-based system on the portable computer collects numeric physiological data (e.g. heart rate, blood pressures, oxygen saturation) every second, monitors alarm and algorithm alarm recordings as alarms occurs, and gathers annotations of any clinical events (e.g. suction, drug injection, change of sensors). The investigator also obtains the patient's age, gender, primary reason for MICU stay, and the settings of monitor alarm limits from the nurse, and records this information into the database.
3) A half hour or one hour into data collection, intelligent alarm algorithms start to analyze the collected data in real-time to generate visual alarms on the screen of the investigator's computer. When such an alarm appears, the investigator will record the time, duration, and cause of these generated alarms, along with the nurse's comment regarding the patient's state. The performance of these algorithms may be evaluated either in real time or after the session ends.
4) At the end of each session (2-12 hours after the beginning of the session), collected data and annotations may remain stored on the investigator's computer or may be transferred at a remote location to a file server.

2.3 Results

2.3.1 System Evaluation

We tested our system according to the evaluation procedure described in the last section on a Pentium 4 professional notebook with 2.2 GHz CPU speed, 1024 MB memory, 60 GB storage disk space, and 64 MB video memory. It was used to collect patient data over 300 hours in

sessions from 2 to 12 hours in duration, and demonstrated no major problem in the last 200 hours of data collection.

CMSCOM was the most robust and independent component of the system. Its function was not disrupted by disturbances or faults in other parts of the system. However, to achieve this robustness, it required the highest priority among all running applications. Otherwise, data arrival from the bedside monitor would be aperiodic and may cause data loss at unpredictable times.

The number of concurrent multiple threads was tested up to 70, which allowed message boxes for 70 alarms to be active at the same time. Because this number far exceeded the amount of annotations the observer could handle concurrently, higher numbers were not tried. The system may be able to handle more than 70 concurrent annotations given that the system resources such as memory and screen size were sufficiently large.

As a data collection system, the system's performance was consistent. For the evaluation of alarm algorithms, the system performed at a normal level for up to 10 algorithms. When trials of more than 10 alarm algorithms were carried out simultaneously, the amount of computation and memory usage started to put a burden on the system, resulting in lower performance and data loss. Such problems were alleviated by decreasing the rate of data collection and analysis (e.g. from 1 Hz to 0.5 Hz).

The system slowed down globally in response to the notebook heat-up problem. Nevertheless, all processes continued in a synchronized manner. This problem was also alleviated by decreasing the rate of data collection and analysis.

2.3.2 Data Collection

The MICU provides intensive care to a diverse group of children. Its patient population consists of all critically ill children admitted to Children's Hospital except those who have primary cardiac disease and require treatments in the Coronary Intensive Care Unit (CCU). Thus, patients come to the MICU for a variety of reasons, such as recovery from surgery, treatments for specific diseases, close monitoring and diagnosis. They range from neonates to adolescents. Their stays at the MICU last from a few hours to a few months. A nurse simultaneously cares for at most

two patients. One or sometimes two nurses are assigned to a patient who needs frequent interventions.

Different patients were monitored for different sets of physiological parameters. Some parameters, such as heart rate, were measured in every patient, while others were measured only in some patients. Table 2.6 lists the numeric parameters according to how frequently they were measured during this research.

Always Monitored parameters	Heart rate Pulse rate Respiratory rate Arterial oxygen saturation
Frequently monitored parameters	Arterial blood pressure (systolic, diastolic, mean) Noninvasive blood pressure (systolic, diastolic, mean) Oxygen perfusion Venous oxygen saturation
Less frequently monitored parameters	Temperature Central venous pressure Carbon dioxide level
Rarely monitored parameters	Wedge pressure Cardiac output Temperature difference

Table 2.6 Monitored numeric parameters

During 196 monitoring hours, collected between 8 AM to 2 AM, in sessions of 2-12 hours, the bedside monitor sounded 325 clinical alarms. Of these alarms, 290 were true positives with clinical relevance, 20 were true positives but clinically irrelevant, and 15 were false positives. Two instances of false negatives were observed.

Arterial blood pressure alarms had a false positive rate at 20%, but the other 80% were all true positives with clinical relevance. While 13 out of 15 false alarms were arterial oxygen saturation alarms, over 80% of the other arterial oxygen saturation alarms were true positives, and about 80% of which were clinical relevant, and the rest were clinically irrelevant true positives. Heart rate alarms, respiratory rate alarms, and noninvasive blood pressure alarms were mostly true positives without clinical relevance. There was no oxygen perfusion alarm because the nursing

staff did not set limits or turn on the alarm system for this parameter. Other rarely monitored parameters did not generate alarms.

Besides alarms that alert the caregiver of possible deterioration in the patient's condition, the CMS monitor generated another kind of audio and visual alerts, called INOPs. During the 196 monitoring hours, 1768 INOPs were generated. They were used to alert the caregivers of setup or hardware faults and the system being unable to process signals properly. Unlike the alarms, they were not specifically indicative of the patient's condition. For clarity, in the rest of this thesis, we refer to regular alarms as alarms, INOP alerts as INOPs, and alarms and INOPs together as alerts.

According to our observations at the bedside, the MICU staff responded to the two kinds of alerts with different senses of urgency. Alarms and INOPs were sounded with different tones. Upon hearing an alarm, the nurses would immediately check the patient and the values displayed on the monitor's screen. During the 196 monitoring hours, ten out of the 325 alarms were silenced by the MICU staff. At no times did the MICU staff suspend the monitor's alarm system. For the INOPs, the nurses could wait from seconds to minutes before checking the monitor and making appropriate adjustments of sensors and equipment connections. After identifying the cause of an INOP, they usually silenced the audio alert but rarely suspended it.

Not only were the nurses' prioritizations of the INOPs different from that of the alarms, but also the rates of the two types of alerts varied widely from session to session and patient to patient. Figure 2.8 is a histogram of the number of patients in each bracket of different alarm rates. Out of the 16 patients included in this study, five patients were in especially critical conditions, so that the study sessions had to end and the observer had to withdraw from the bedside within the first two hours. These patients had the highest alarm rate and contributed to the distribution above 10 alarms per hour. However, no INOPs were generated during the observation of these patients. The other eleven patients were followed in sessions of 2-12 hours in the total 196 monitoring hours. Figure 2.9 is the histogram of the number of patients in each bracket of different alarm rates for the eleven patients. It is in fact a zoomed-in view of Figure 2.8 for the patients whose alarm rate was below 10 alarms per hour.

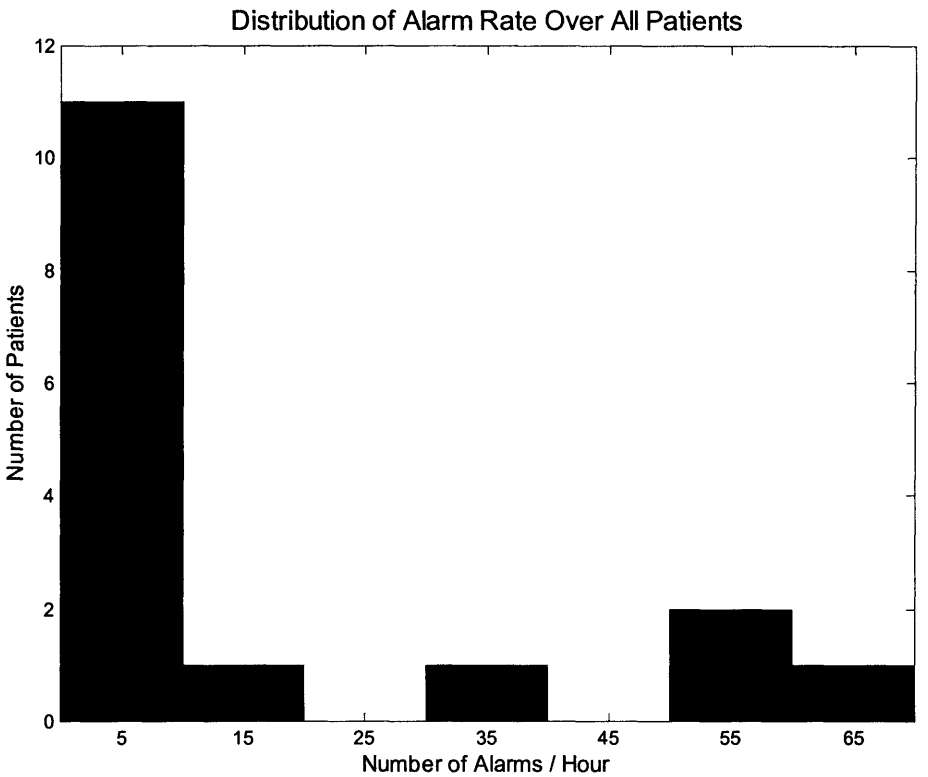


Figure 2.8 Distribution of alarm rate over all patients

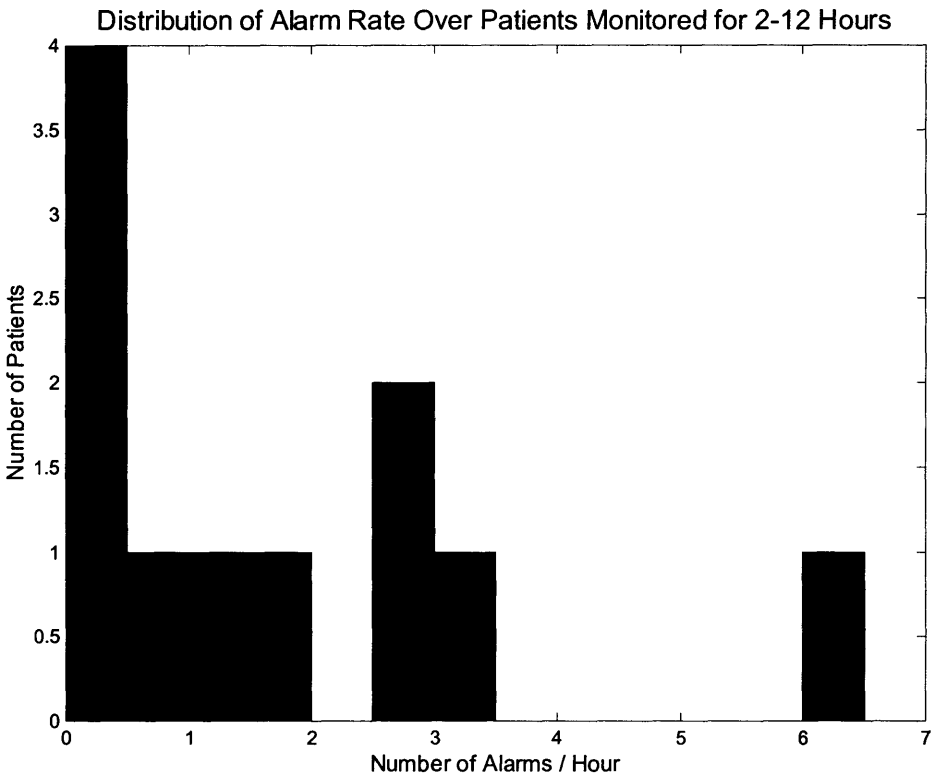


Figure 2.9 Distribution of alarm rate over the patients monitored for 2-12 hours

Three of these eleven patients had no alarms. Others' alarm rates were relatively distributed closed to the mean of 1.7 alarms/hour. The total 1768 observed INOPs were randomly distributed among these patients.

2.4 Discussion

This system for synchronized collection of physiological signals and clinical annotations has been shown to synchronize the physiological data and clinical event recordings in a consistent manner. Although its performance could be influenced by hardware capabilities, it is robust in achieving real-time data collection at the bedside.

Using this system, we were able to collect a set of annotated physiological data with more certainty in data correlation than previous studies had. It can be easily backed up or restored. The data storage format also allows easy access during both prospective and retrospective data analysis.

A major new finding in this study is the unexpectedly low volume of clinical alarms generated by the bedside monitor. In the study by Tsien, there were 2942 alarms during 298 monitored hours: about 9.9 alarms/hour. [45] In the similar study by Lawless, there were 2176 alarms during 928 monitoring hours: about 2.3 alarms/hour. [25] Our results show only 325 clinical alarms during 196 monitoring hours: about 1.7 alarms/hour.

One explanation for the low volume of clinical alarms in comparison to the study by Tsien is that Tsien's study counted what we now call INOPs as clinical alarms. That study obtained its data from the previous generations of patient monitoring systems, which did not have the capability to distinguish the abnormal measurements that were due to hardware or operational malfunction of the monitor itself from those that were due to some physiological changes in the patient. In fact, the number of alarms and INOPs per hour is about 10.7 alerts/hour, which is similar and even slightly higher than the alarm rate of 9.9 alarms/ hour that was found by Tsien.

This reasoning may also help explain the unexpected low rate of false alarms. Comparing to previous studies in which the false alarm rate is as high as 86%, a false alarm rate of 4.6% found in this study suggests a significant reduction in false alarm rate with the newer monitoring system. [45] Since all the technical alarms are false alarms by our alarm classification and we

exclude them from the alarm count, our false alarm rate should be lower than those in previous studies.

Table 2.7 gives a breakdown of alert categorization with two counting schemes. The first scheme includes alarms only, and the second scheme includes both alarms and INOPs, where INOPs are considered as false positive alarms.

Study	Average Alarm Rate (Number of Alarms / Hour)			
	Total	TP-R	TP-I	FP
Tsien	9.9	0.79	0.59	0.79
Lawless	2.3	0.12	0.62	0.12
This study (W/O INOPs)	1.7	1.49	0.10	0.05
This Study (W/ INOPs)	10.7	1.49	0.10	9.14

Table 2.7 Frequencies of different types of alerts

Including the INOPs in the alarm number count has no effect on the rate of true positive alarms; only the false positive rate and total alert rate are changed.

As in the study by Tsien, Lawless' alarm statistics include alarms that were strictly due to hardware or operational malfunctions; yet, its alarm rate, 2.3 alarms/hour, is only about 0.5 alarms higher than this study's alarm rate that does not include the INOPs. One explanation may be that the actual number of alarms is higher than the recorded alarm rate. Lawless obtained his alarm statistics by asking the pediatric ICU staff to voluntarily record the source, time (shift), and the number of alarm soundings for each patient during a 7-day period. [25] Given the intense workload in an environment such as the ICU, and for reasons that we will elaborate in Chapter 3, the nurses may not had time to record all the occurrences of alarms at the bedside.

Another reason for our significantly different results compared to previous studies may be the differences among the three patient populations. One characteristic of a patient population is the critical nature of the patient's state. We could expect that a population of less severely ill patients would have a lower average alarm rate. The number of clinically relevant true positive alarms

serves as a measure of this characteristic. Table 2.7 shows that our study has the highest rate of clinically relevant true positive alarms among the three studies. The TP-R rate in Tsien's study is about a half of what our study observed. Lawless' TP-R rate is the lowest. Thus, the results suggest that the patient populations for the three studies are different, but the lower alarm rate in this study is not necessarily a result of having a population of less clinically critical patients. On the contrary, the relatively high rate of clinically relevant true positive alarms suggests that the patient population in this study overall may be more critical than that in the previous two studies.

Although the alarm statistics are dependent on the overall patient population, we must emphasize that individual patients within the study may be clinically very different from one another. We do not have the distribution of patient population over the true alarm rates from the previous studies, but the patient distribution in our study, as illustrated by Figure 2.8 and Figure 2.9, suggests that the alarm rate could vary significantly within the patient population for a study. For example, four patients had an alarm rate of less than 0.5 alarms per hour, while five patients had more than 10 alarms per hour. Even within the range of 10 alarms per hour, seven out of the eleven patients had different alarm rates.

Table 2.7 shows another major new finding in this study: the rate of clinically irrelevant true positive alarms is lower than that in the previous studies by about six fold. This decrease is further illustrated by the statistics of the TP-I alarms in Table 2.8.

Study	Percentage of Total Number of Alarms in Each Class		
	TP-R	TP-I	FP
Tsien	8%	6%	86%
Lawless	5.5%	26.5%	68%
This study (W/O INOPs)	89.2%	6.2%	4.6%
This Study (W/ INOPs)	13.8%	1.0%	85.2%

Table 2.8 Distribution of the alarms among different alarm classes

Statistics of different types of alerts averaged over 196 monitoring hours. The alarm classification methods for Tsien's study and this study are the same, as described in Section 2.2. Lawless' categorization of TP-R and TP-I is slightly different. His TP-R statistics are for alarms that lead to change in therapy, while TP-I alarms are true alarms that do not lead to change in therapy.

Counting the INOPs, only 1% of the alarms in this study were clinically irrelevant true positives. This percentage is far less than that found by Lawless, and this result may be due to the different definitions for TP-I alarms and TP-R alarms in the two studies. However, this percentage is also less than that found by Tsien, who used the same alarm classification method and found a similar false positive alarm rate as did this study, with INOPs counted as false alarms. Thus, the results overall suggest that the newer generation of patient monitoring systems has improved data analysis and alarm algorithms.

The significant reduction in the clinically irrelevant true alarms is not the only evidence for the improvements in the new monitors. Without counting the INOPs, about 89.2% of all alarms are true positives with clinical relevance. Even with the INOPs counted (and thus the false alarm rate being comparable to those in the previous studies), the percentage of clinically significant true alarms under the same classification method has improved from 8% to 13.8%.

Due to intensive competition in the market of patient monitoring systems, individual vendors guard their special techniques as proprietary information, so the new data analysis techniques and specialized algorithms in the current generation of monitors have not been made public. We hypothesize that more sophisticated signal processing is applied to physiological signals to reduce noise and to derive more accurately the numeric values. Data analysis that incorporates information from multiple parameters may also facilitate the observed improvements. Furthermore, specialized algorithms, such as that for ST elevation detection, may also contribute to the increased specificity of alarm sounding decisions.

Given these unexpected results, we then would like to ask: is there still room or a need to improve patient monitoring systems? The answer seems to be yes. Why? Data from this study only represents a subset of patients in one pediatric ICU. Adult patient population may have a wider range of disease courses. Moreover, the purpose of patient monitoring systems is not only to generate sensitive and specific alarms but also to make patient data more readily useable and informative to the medical professionals. Thus, much more work still need to be carried out.

Things we have learned from this study help us to find the direction for this future work. In the next chapter, we explore some of the ideas in this direction.

Chapter 3
Real-Time Development of Alarm Algorithms

3.1 Motivation

Machine learning has typically been used to learn patterns in the data collected from multiple patients in medical applications. As a result, the models learned represent the average patient. In clinical medicine, however, many patients behave in highly individual ways that might deviate significantly from the average. Even what counts as "normal" for a specific patient may be highly abnormal if seen in another, and patients' dynamic responses to changing circumstances also vary greatly from individual to individual. One of our main goals, therefore, is to develop learning methods that will create models specific to the individual patient. There is, innately, a trade-off in choosing this strategy, however. The amount of data that is available about a large population is typically far richer than what can be gathered about an individual patient. Therefore, the population model is more likely to include aspects that represent rare events, which are likely to have been seen in the population but not in the individual. Nevertheless, we have chosen to try to learn individualized models in order to explore our ability to recognize events that are most relevant to the specific individual.

A previous study that applies a machine learning-based event discovery process to detecting "true alarm" events from physiological data of multiple patients showed that some models perform significantly worse on data from two other groups of patients. After adjusting the thresholds in these models to levels that were more suited to 9% of one evaluation dataset, the refined models' performances on the rest of that dataset showed significant improvement. [48] This finding suggests that a robust model must have a component that "adapts" to the targeted patient population or to individual patient whenever possible.

In patient monitoring, there could be significant patient variability within the same patient population; thus, adaptation to individual patient is necessary. Yet, how to incorporate patient-specific data into modeling? How to design a model component that changes with the patient? In this research, we carried out an exploratory study towards helping answer these questions. The study examined the feasibility and potential of developing alarm algorithms, using machine learning techniques, based on patient-specific data, in real time at the bedside. If these algorithms could achieve acceptable performance, our methodology could be used in constructing comprehensive models that generalize over both disease processes and patient population.

3.2 Methods

In this section, we describe our methods in four parts: system requirements, training alarm algorithms, evaluating these algorithms, and learning incrementally in real time. We also explain the rationale for choosing the tools that we used and the methodologies that we adopted.

As for any supervised learning in machine learning, the learning process consists of training/model building and testing/evaluation. The core learning algorithm takes in a set of examples with input-output pairs of the target function to be learned and generates all the parameters of the model (weights and biases in the case of neural network models).

3.2.1 System Requirements

In order to develop alarm algorithms in real time, we must have a system that obtains physiological data from a patient monitoring system, that collects clinical annotations at the bedside, whose applications generate alarm algorithms within a bounded time, and that includes a mechanism to evaluate these algorithms on subsequent incoming data.

We expanded our system for synchronized collection of physiological signals and clinical annotations to include modules for training and evaluating alarm algorithms. Alarms generated by these algorithms were annotated in a similar manner as for recording alarm events generated

by the bedside monitor. To maintain modularity, each new application was designed to function on its own processing thread and to be independent of a maximum number of other system components.

3.2.2 Real-Time Training of Alarm Algorithms

In this study, we experimented with two machine learning techniques that had shown potential in generating intelligent alarm algorithms for patient monitoring systems: classification tree learning and neural network learning. We focused on classifying each stream or instance of physiological data into an "alarm class," if it corresponds to an adverse event for the patient, or a "non-alarm class," if it represents some normal patient condition.

This section describes the two machine learning techniques and their implementation in our study. At the end of the section, we also elaborate on key issues that we have encountered in real-time training of alarm algorithms.

3.2.2.1 Classification Tree Learning

3.2.2.1.1 Overview of Classification Tree Learning

Classification tree learning is a method for inductive inference that takes on continuous-valued, discrete-valued, and/or Boolean inputs and generates a discrete-valued output, and the learned function is represented by a classification tree or a set of if-then rules. It is one of the most widely used classifiers, and it classifies each instance by sorting it down a tree from the root, through branching nodes that depend on the values of the instance's attributes, to a leaf node, which represents the instance's class. It also forms a disjunction of conjunction of ranges of attribute values of training instances.

3.2.2.1.2 Why Choose Classification Tree Learning?

Classification tree learning is a useful classification method for many practical problems. It is generally best suited to problems in which instances are represented by a fixed set of attributes; the classification output is discrete-valued; disjunctive description may be required; training data may contain errors; and the training data may contain missing attribute values. [28] The problem of deciding whether to sound an alarm has all of these characteristics. Thus, classification tree learning is particularly useful for the classification of patient state as needing prompt medical attention or not.

Another reason for choosing classification tree learning is that a previous study showed that classification trees are effective in classifying ICU data from a pediatric patient population into the alarm class and non-alarm class. [47] It demonstrated the potential of classification tree learning in providing decision support.

Furthermore, the core algorithm for classification tree learning is well developed and readily available. It is often referred to as ID3 in the literature, and it conducts a top-down, greedy search through the space of possible classification trees. [33] ID3's successors C4.5 and C5 have been widely used. [28] The software that implements C5 can be called in batch-mode as an independent, embedded application, running at very high speed, with almost no limit on the amount of training data or test data. This feature allows the software to be incorporated into our system.

3.2.2.1.3 Real-Time Generation of Classification Trees

For this study, we obtained the software See5, the Windows version of C5. (RuleQuest Research, St. Ives, Australia) [41] To run See5 in batch-mode for training, two text files of specified format were required. One file, called the Name file, specified the number and type of each attribute, the type of the output, the order of attributes for the input data, and special training options, such as boosting, cross-validation, and pruning. The other file, called the Data file, contained the training data in the specified order. A third text file, called the Cost file, could be used, but is not required, to specify an asymmetric cost function.

Before a trained observer initializes training from the command center, he or she can specify a training size, which is the number of data points to be written to the data file. After initialization,

PAAT automatically creates the Name file based on the available physiological data specified on the user interface, writes the most recent data of the specified training size into the Data file, and then calls See5 to process the training data and generate a classification tree. The resulting classification tree is stored in a file called the Tree file.

3.2.2.1.4 Classification Tree Training Options

There are several options available for modifying the training process in classification tree learning. The ones used in See5 are boosting, cross-validation, differential misclassification costs, fuzzy thresholds, local pruning, global pruning, final minimum cases, winnowing, subset, and sampling. The following subsections provide an overview of each option and specify how each was set in this study.

Boosting. When this option is selected with the specification of the number of trials, the training process builds a classifier in each trial. When an instance is classified, each classifier votes for its predicted class with a weight equal to the confidence of its prediction, and the class assignment that has the weighted majority vote from all the classifiers is the final decision. The classification tree model is then this set of classifiers. Depending on the nature of the data, boosting often gives higher predictive accuracy but at the expense of increased training time. In our study, we used the standard setting of 10 trials for boosting.

Cross-validation. This option can be used to estimate the accuracy of the classifier even when there are no separate test cases. The training data are divided into a user-specified number of blocks, and this number is referred to as the number of folds. Each block contains approximately the same number of cases and the same distribution of classes. A classifier is built for data in the other blocks and then tested on the data in the block. Because the data division changes each time a cross-validation is carried out, the results also change each time for the same data.

We experimented with cross-validations with 5 to 15 folds. This option further lengthened the training time. Since we were going to estimate the accuracy of the classifier(s) on separate data instances, we chose not to carry out cross-validation in real time.

Differential misclassification costs. Without this option, by default, the core learning algorithm assumes that the costs of all types of misclassification are equal. Some problems, however, have an asymmetric cost function. For example, in medicine, the cost of diagnosing a patient with a disease as not having the disease is much higher than the cost of diagnosing the patient positive for the disease when the patient in fact does not have the disease. Thus, we first used differential misclassification costs in training. However, due to the bias this cost assignment created, as to be discussed in the later sections under the topic of handling imbalanced dataset, we decided not to assign differential misclassification costs.

Local Pruning. To prevent the classification tree from overfitting the training data, local pruning examines each subtree and replaces it by a leaf or a sub-branch if by doing so the entire tree's estimated predictive accuracy would be improved. The Pruning CF parameter for this option controls the extent of local pruning. A lower value causes more extensive simplification. In this study, we used the default Pruning CF value of 25%.

Global pruning. Global pruning looks at the tree as a whole and may prune relatively weak subtrees to prevent the final tree from overfitting the training data. No parameter specifies the extent of global pruning in See5. We used this option to obtain shorter trees, which may generalize better over instances not in the training data.

Final Minimum Cases. The parameter for this option also influences the complexity of the classifier. A higher minimum number of cases restricts the tests that can be used near the leaves of the classification tree and usually leads to smaller classification trees. In this study, we used the default parameter value of 2 cases.

Winnowing. Selecting this option directs the core learning algorithm to analyze the training cases and discard attributes that are only marginally relevant to classification prior to constructing a classifier. In this study, we assumed that all data could potentially be useful for classification, and this option was not selected.

Subset. By default, the core learning algorithms deals separately with each value of an unordered discrete attribute. If this option were chosen, these values would be grouped into subsets. In this study, all attributes are continuous-valued, so we did not select this option.

Sampling. Selecting this option causes only the specified percentage of the training instances to be used for constructing the classifier. Samples are redrawn every time a classifier is constructed. Since we could lose interesting patterns or informative training instances in sampling, we chose not to use this option. This rationale is elaborated further in section 3.3.

3.2.2.2 Neural Network Learning

3.2.2.2.1 Overview of Neural Network learning

Neural Network learning is inspired in part by the complex webs of interconnected neurons in biological learning systems in the brain. It is a general, practical method for learning real-valued, discrete-valued, and vector-valued nonlinear functions. It has the structure in Figure 3.1.

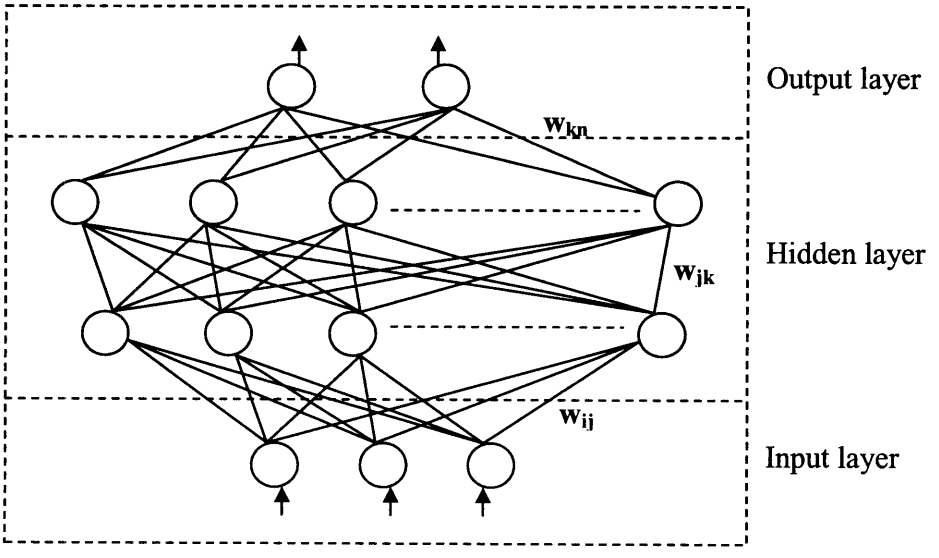


Figure 3.1 Neural network structure

A neural network consists of a layer of input nodes, a user-specified number of layers of hidden nodes, and a layer of output nodes. Each input node outputs the value of the corresponding attribute, and this output feeds into every node in the next layer with a specific weight, and this weighted value becomes the input of this next node. A bias may be added to each hidden layer. The values of the output layer nodes specify the output of the network. The number of nodes in each hidden layer can be user-specified or optimized during learning. Each hidden node transfers its input to the output with a specific transfer function.

3.2.2.2.2 Why Choose Neural Network Learning?

Neural network learning is among the most effective methods for learning to interpret complex real-world sensor data and also applicable to problems for which more symbolic representations are often used. With the commonly-used transfer functions, it is inherently nonlinear and is useful especially to capture nonlinear patterns. The training examples may contain errors, and evaluation of the learned target function is fast. [28] These capabilities work well for our problem.

Another reason for choosing neural network learning is that a previous study showed that neural network classifiers are effective in classifying ICU data from a pediatric patient population into the alarm class and non-alarm class. [48] It demonstrated the potential of neural network learning in providing decision support.

This machine learning technique does have the drawbacks of possible long training times and difficulty with interpretation of the input-output relations. Because we were using data from one patient and the estimated data size was not significantly large, we expected the training time to be tolerable. Also, at this stage in our research, we would like to focus on pattern recognition first before going onto pattern elucidation. Thus, these drawbacks would not prevent us from achieving our current research goals.

3.2.2.2.3 Real-time Generation of Neural Network

We use the backpropagation algorithm as our core learning algorithm. Each hidden node is designed as a sigmoid threshold unit, which is best illustrated by Mitchell, as adopted in Figure 3.2.

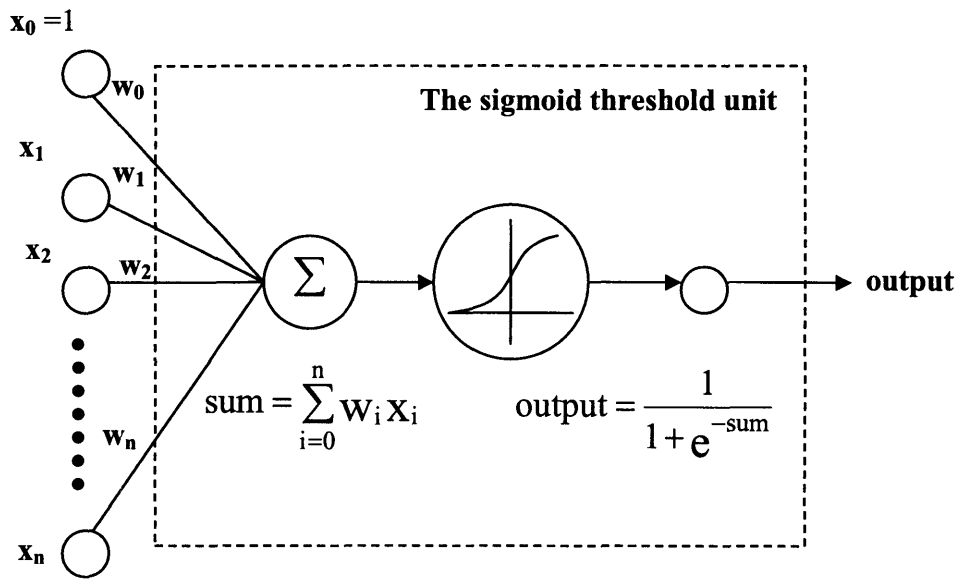


Figure 3.2 The primitive unit for the neural networks' hidden nodes

We used EasyNN-plus, a software package that employs BACKPROPAGATION as the core learning algorithm, to build neural networks. To run EasyNN-plus in batch-mode for training, two text files of specified format are required. One file, called the Script file, specifies the number and type of each attribute, the type of the output, the order of attributes in the training dataset, and options for the training process. The other file, called the Data file, contains the training data.

Before a trained observer initializes training from the command center, he or she specifies a training size, the number of data points to be written to the data file. After initialization, PAAT automatically creates the Script file based on the available physiological data specified on the

user interface, writes the most recent data of the specified training size into the Data file, and then calls EasyNN to process the training data and generate a neural network. The resulting neural network is stored in a file called the Network file.

3.2.2.2.4 Neural Network Learning Specifications

Learning rate. Learning rate determines the step size in the gradient descent search of the BACKPROPAGATION algorithm by moderating the degree to which weights are changed at each step. The trade-off on learning rate is that a smaller rate enables a more thorough learning process but lengthens the training time. In our study, training data varied from patient to patient and even over different monitoring periods of the same patient, so an "optimal" learning rate could not be pre-determined. We balanced over this trade-off in real time by setting the initial learning rate at 1.0, letting the core learning algorithm optimize it during training, and making it decay over training cycles. EasyNN-plus optimizes the learning rate by running a few learning cycles with different values prior to actual training to determine a learning rate that allows quick learning and convergence. Then, the learning rate is automatically reduced if erratic learning or oscillations in training error occur.

Momentum. Momentum is used to alter the weight-update rule in the BACKPROPAGATION algorithm to speed up convergence. For our purposes, we first set the momentum to be 0.8 and then let the core learning algorithm to optimize it and to allow it to decay over training cycles. EasyNN-plus optimizes the momentum by running a few learning cycles with different values prior to the actual training and then automatically reduces the value during learning if oscillations in training error occur.

Network reconfiguration. This option allows the number of nodes in a specified hidden layer to grow during training. A previous study had found that neural networks with one hidden layer are sufficient to capture the complexity in the physiological data. [48] Thus, we started with one hidden layer in neural network learning. The number of nodes in each hidden layer was set to be the number of inputs and then allowed to grow during training.

Validation. To help prevent the neural network from becoming overfitted to the training data, we designated 30% of the training data for validation. The learning and validating process was set to stop after 100% of the validating examples were correct after rounding or if the number of correctly classified validating instances decreased.

Target error stops. The core algorithm can stop the learning after a target error is reached. In our study, we set the average error of 0.1 as the target error.

Fixed period stops. The core algorithm allows the user to preset a time limit on the training. To allow adequate training while staying within the time needed for real-time training, we set the upper bound for training time at 120 seconds.

3.2.2.3 Key issues in real-time training

3.2.2.3.1 Handling Imbalanced Dataset

As discussed in Chapter 2, clinical events were few in number and sparse in time at the bedside. The available dataset for training alarm algorithms was highly imbalanced or skewed. As a result, the classification trees or neural networks derived from such imbalanced training dataset were biased towards classifying every new instance as belonging to the non-alarm class. We could in theory have an alarm algorithm that would never generate an alarm.

To address this problem with the imbalanced dataset, we experimented with three ways to reduce its effects. One method was to introduce an asymmetrical cost function: the cost of a false negative greatly exceeds the cost of a false positive. This method was applicable in classification tree learning but not in neural network learning. The second method was to include only a sample of the non-alarm instances in the training dataset such that the two classes were equally represented during training. The third method was is to duplicate the instances in the alarm class multiple times in the training dataset to reduce the imbalance.

3.2.2.3.2 Feature Derivation

Features of clinical time-series data are information derived from the data either over time or from multiple time-series or both. Extracting features that effectively characterize the trends in the training data is an important step in building generalized models for patient monitoring. A number of features have been derived from clinical time-series in previous studies, but their techniques for feature derivation cannot be readily employed in real-time learning, which requires a large memory space to keep track of the data over time and fast calculations. As the number of features and the complexity of the features increase, the time needed to derive these features can increase exponentially. Thus, in this study, we used both the actual value of the measurements and simple features such as time averages.

3.2.2.3.3 Handling Missing Values

The problem of missing values occurs in two forms. The first form usually results from unavailability of the measurements (i.e. the caregiver removes a sensor from the patient). The parameter with missing values might not have any measurement readings for a period of time or the rest of the session. With the second form, the measurement of a particular parameter is not available or the monitor does not generate a value occasionally because of corrupted signals.

The trained observer could readily detect the second form at the bedside. Our action in this case was to exclude the attribute(s) with missing values from the training of new algorithms, but we continued to run existing algorithms through the incoming data. For the second form, we chose to let the core learning algorithms' built-in functions handle the missing values.

The built-in function in See5 estimates a probability for each of the possible value of the attribute at each sample based on the observed frequencies among the training instances and substitutes the missing value with the value with the highest probability. To classify new instances whose attribute values are unknown, See5 computes the most probable classification by summing the weights of the instance fragments classified according to different path down the tree at the leaf nodes. [34]

The built-in function for handling missing values in EasyNN-plus substitutes the missing value with the median of the affected attribute in the training dataset. This method is used during both training and evaluation. [12]

3.2.3 Real-Time Evaluation of Alarm Algorithms

After developing the alarm algorithms, we evaluated them in real time because real-time trials of alarm algorithms have three major advantages. First, they run the alarm algorithm as soon as the physiological data and clinical annotations are collected. Therefore, physiological data, clinical annotations, and alarms generated by the alarm algorithms are synchronized and time-stamped within the same time frame. This synchronized setup allows better correlation between alarm algorithm output and the clinical data.

Second, when an alarm algorithm generates an alarm, the trained observer can immediately annotate what is going on at the bedspace and proceed to classify the alarm. He or she can seek and verify information about the patient's state with the medical professionals. In contrast, the conventional approach to evaluating alarm algorithms might not have recorded this information, especially if the clinical staff did not observe anything unusual. Thus, evaluating alarm algorithms in real-time provides more informative assessment of the algorithms.

Third, the real time trials allow the detection of false negatives when the bedside monitor and the alarm algorithms all fail to detect these clinical significant events. Unlike in retrospective evaluations where the access to clinical information is limited to what has been recorded, the observer can obtain the information about these false negatives from the ICU staff during real-time evaluation. This information is valuable in either refining the alarm algorithm under study or re-tuning of the learning process.

After the trained observer initializes the evaluation of the classification tree, a new data stream is written into a file called the Cases file. The application then uses the Tree file and the Cases file to classify the data stream and outputs the classification result into a file called the Class file.

For the evaluation of the neural network, a new data stream is input into an application that obtains the weights and bias values from the Network file and calculates the output. We created this application independently of EasyNN-plus because that software's memory requirement may potentially slow down other processes.

3.2.4 Incremental Learning

In this part of the study, we would like to explore the question: how much data is needed to capture the states of a patient over time. In the framework of our research, how much data is needed to develop a patient-specific yet adequate model of the patient state in real time?

As the first step, we built classification trees and neural networks in incremental periods at 30 minutes, 1 hour, 2 hours, 4 hours, and 8 hours after the beginning of 12-hour sessions. Each model was evaluated immediately on the incoming data in the parallel processes illustrated in Figure A.2 to A.4. Model evaluation first used the clinical annotations to determine the actual class of each instance in the database. Then, four performance metrics were calculated for each model, based on the classification given by the model and the actual class of each instances, and averaged over 10 study sessions.

The four performance metrics are sensitivity, specificity, positive predictive value (PPV), and accuracy. Sensitivity measures the percentage of actual alarm-class instances that are correctly classified by the model. Sensitivity measures the percentage of actual no-alarm class instances that are correctly classified by the model. Positive predictive value measures the percentage of true positives of all the instances that are classified by the model as in the alarm-class. It is one minus the false positive rate. Accuracy measures the percentage of correctly classified instances by the model over all instances. Below is an illustration of the four performance metrics, based on an illustration adopted from Tsien [48]:

$$\text{Sensitivity} = \frac{d}{b+d} = \frac{\text{number of correct model} - \text{classified alarm instance}}{\text{total number of instances}}$$

$$\text{Specificity} = \frac{a}{a+c} = \frac{\text{number of correct model} - \text{classified non} - \text{alarm instance}}{\text{total number of non} - \text{alarm instances}}$$

$$\text{PPV} = \frac{d}{c+d} = \frac{\text{number of correct model} - \text{classified alarm instance}}{\text{total number of model} - \text{classified alarm instance}}$$

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{\text{total number of correct model} - \text{classified instances}}{\text{total number of instances}}$$

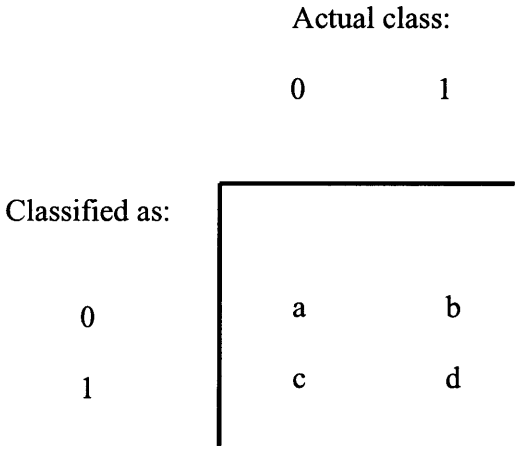


Figure 3.3 Performance metrics illustration

3.3 Results

3.3.1 Training Time Assessment

The duration of the training process varies with the training size and training options. Table 3.1 lists the training time for classification trees during a typical 12-hour study session in the ICU.

Number of Examples	Training Time for Classification Tree Learning (seconds)		
	10 Boosting Trials 10-fold Cross-validation Global Prunning 25%	10-fold Cross-validation Global Prunning 25%	Global Prunning 25%
1800	0.1	0.1	< 0.1
3600	2.5	0.2	0.1
7200	3.2	0.7	0.2
14400	11.0	2.0	0.5
28800	53.5	6	1.1

Table 3.1 Classification tree training time

The actual training time for each study could vary within ±50% from the figures given in Table 3.1. When applications other than those of the system are also running, the training time could lengthen significantly.

Training neural networks in general took more time than training classification trees. EasyNN-plus does not keep track training time; thus, precise estimates of training times are not available for neural network learning. A rough estimate based on repeated observations was in the ranges of a few seconds to several minutes for 1800 to 28800 examples, corresponding to 30 minutes to 8 monitoring hours. Training time could vary significantly with different training specifications, such as learning rate, momentum, the number of validation cycles, and the target error for each cycle. EasyNN-plus was set to stop training at 120 seconds. Thus, the comparatively long training time for neural network learning did not disrupt the overall system, although at the expense of modeling accuracy.

3.3.2 Sample Classification Tree

An example of a classification tree model for detecting true alarms is shown in Figure 3.4. This tree was the classifier that had the smallest training error among a community of classifiers that were built on 14400 training examples (data from 4 monitoring hours) with 10 trials of boosting. Cross-validation and asymmetric cost function are not used. In the classification tree, true-alarm class is labeled as "1" and the non-alarm class is labeled as "0". The parentheses after the class label indicate the number of training examples that arrived at that node, followed by the number out of these examples that were incorrectly classified at that node. These numbers can have decimals because of pruning the tree.

To read this tree, we start with the first line. If the arterial oxygen saturation is greater than 91, the instance is classified into the non-alarm class. There were 10645.6 training examples that were classified into the non-alarm class at this node, and 187.7 of these examples were classified incorrectly. If the arterial oxygen saturation is less than 87, the instance is classified into the true-alarm class. There were 130.2 training examples that were classified into the true-alarm class at this node, and 0.3 of which were classified incorrectly. If the arterial oxygen saturation is greater than or equal to 87 but less than 91, we look at the heart rate: if the heart rate is greater than 211,

```
O2Sat > 91: 0 (10645.6/187.7)
O2Sat <= 91:
:.....O2Sat <= 87: 1 (130.2/0.3)
     O2Sat > 87:
  :.....HR > 211: 1 (47.9/0.4)
        HR <= 211:
    :.....PRAvg > 117.4: 1 (252.4/52.1)
          PRAvg <= 117.4:
      :.....O2SatAvg > 89.2: 0 (1185.1/47.5)
            O2SatAvg <= 89.2:
        :.....O2Perf > 4.1: 1 (499.5/41)
              O2Perf <= 4.1:
          :.....PRAvg > 110.8: 1 (76.9/0.2)
                PRAvg <= 110.8:
            :.....HRAvg > 111.6: 1 (58.6/2.5)
                  HRAvg <= 111.6:
              :.....O2PerfAvg <= 2.74: 1 (262.2/114)
                    O2PerfAvg > 2.74: 0 (1241.6/123)
```

Figure 3.4 An example of classification tree.

the instance is in the true-alarm class; otherwise, we look at the time average of the pulse rate, and so forth.

3.3.3 Sample Neural Network

An example of neural network model for alarm detection contains 16 input nodes, 1 hidden layer with 18 nodes, and one output node. The number of training examples was 14400. During network training, the optimal step size was 0.2, momentum was 0.8, target error for stopping each training cycle was 0.1.

3.3.4 Imbalanced Dataset

Using an asymmetric cost function to rebalance the dataset caused about a 4 to 10 fold increase in training error for cost ratios (false positive: false negative) from 1:10 to 1:1000. The classification trees generated with such cost function generated at least twice more false alarms than those with a symmetric function. The increase in the false alarm rate depended on the nature

of the training data, so only a lower bound has been observed. For two classification trees, the false alarm rate jumped by 50 fold.

Rebalancing the dataset by sampling instances in the larger class also increased the false alarm rate, although not as dramatically as we have seen with the first approach. The increase depended on both the training data and the evaluation data.

Rebalancing the dataset by duplicating the instances in the smaller class multiple times showed improved sensitivity and no general increase of the false alarm rate. However, the training time for a larger training dataset could slow down the system. Some classifiers also showed the effects of overfitting the duplicated training examples. The classification tree in Figure 3.5 illustrates some of these effects.

```
O2SatAvg > 89.2:
:.....PRAvg <= 117.4: 0 (14227.6/5)
:     PRAvg > 117.4:
: :.....PRAvg <= 118: 1 (13)
:       PRAvg > 118: 0 (24.1)
O2SatAvg <= 89.2:
:.....O2Perf > 4.1: 1 (35.1/1.1)
     O2Perf <= 4.1:
  :.....O2Sat <= 87: 1 (12)
       O2Sat > 87:
    :.....PRAvg > 110.8: 1 (12)
          PRAvg <= 110.8:
      :.....O2Perf <= 2.5: 1 (9/1)
            O2Perf > 2.5: 0 (67.1/7)
```

Figure 3.5 An example of an overfitted classification tree.

3.3.5 Feature Derivation

Although both classification tree models and neural network models varied from patient to patient and time interval to time interval, time averages were selected more frequently for the node in classification trees than the actual parameter. In neural network models, they in general had larger weights.

3.3.6 Incremental Learning

The performances of classification tree models averaged over 10 sessions are presented in Table 3.2.

Performance Metrix	Performance Averaged Over 10 Sessions (~120 Hours)						
	CMS	Threshold	Tree1	Tree2	Tree3	Tree4	Tree5
Sensitivity	1	1	0.0036	0.0083	0.1111	0.4286	0.8432
Specificity	0.9913	0.8765	0.9991	0.9843	0.9794	0.9828	0.9829
PPV	0.8235	0.6958	0.625	0.1667	0.1429	0.3749	0.7214
Accuracy	0.9917	0.9621	0.9575	0.9510	0.9533	0.9700	0.9751

Table 3.2 Performance comparison of classification tree models

The column with the header "CMS" is for the alarm algorithm(s) of the bedside monitor. "Threshold" stands for the standard threshold alarm algorithm. "TreeX" stands for the Xth classification tree model in a session. Tree1 were usually built with 30 minutes or 1800 instances of training data, Tree2 with 60 minutes or 3600 instances, Tree3 with 2 hours or 7200 instances, Tree4 with 4 hours or 14400 instances, and Tree5 with 8 hours or 28800 instances.

The performances of neural network models averaged over 10 sessions are presented in Table 3.3.

Performance Metrix	Performance Averaged Over 10 Sessions (~120 Hours)						
	CMS	Threshold	ANN1	ANN2	ANN3	ANN4	ANN5
Sensitivity	1	1	0.0501	0.2301	0.5948	0.8034	0.9562
Specificity	0.9913	0.8765	0.9940	0.9704	0.9822	0.9856	0.9873
PPV	0.8235	0.6958	0.7000	0.1598	0.3719	0.7134	0.7928
Accuracy	0.9917	0.9621	0.9606	0.9515	0.9605	0.9742	0.9861

Table 3.3 Performance comparison of neural network models

The column with the header "CMS" is for the alarm algorithm(s) of the bedside monitor. "Threshold" stands for the standard threshold alarm algorithm. "ANNX" stands for the Xth neural network model in a session. ANN1 were usually built with 30 minutes or 1800 instances of training data, ANN2 with 60 minutes or 3600 instances, ANN3 with 2 hours or 7200 instances, ANN4 with 4 hours or 14400 instances, and ANN5 with 8 hours or 28800 instances.

Figures 3.6 to 3.9 present the performance comparison of the CMS alarm algorithm(s), the standard threshold alarm algorithm, the classification tree models, and the neural network models.

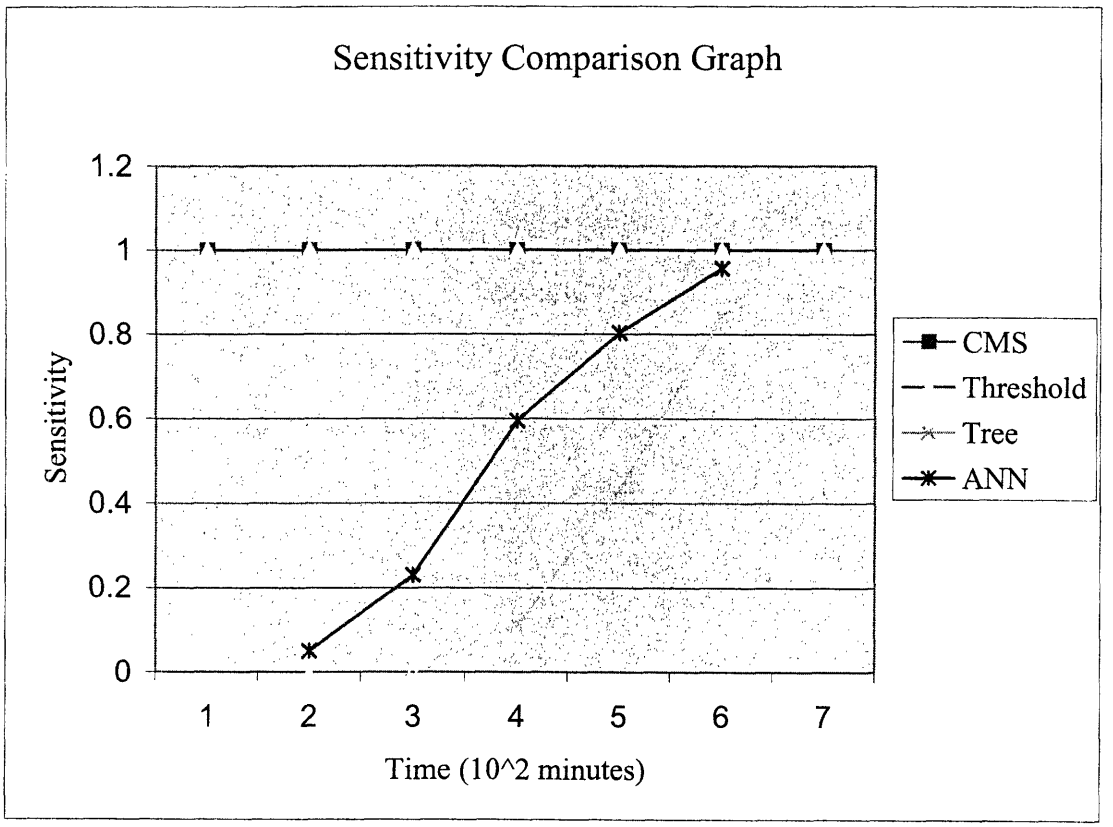


Figure 3.6 Sensitivity comparison graph

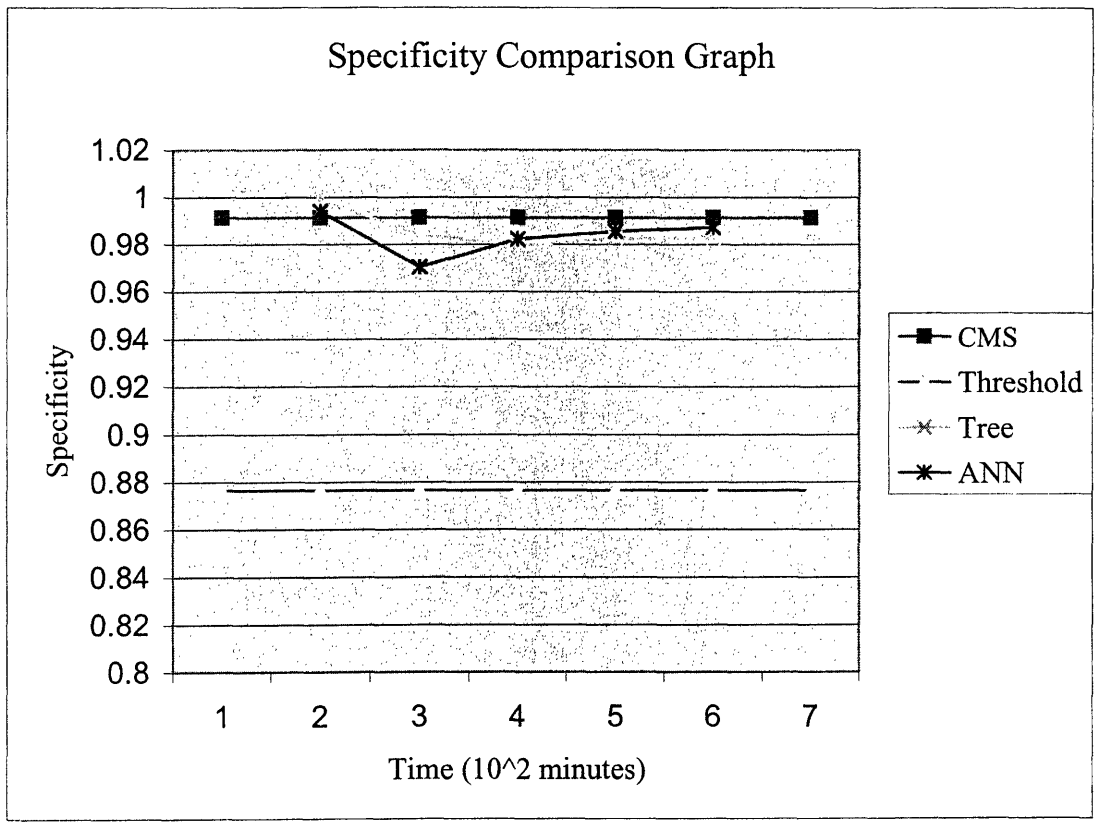


Figure 3.7 Specificity comparison graph

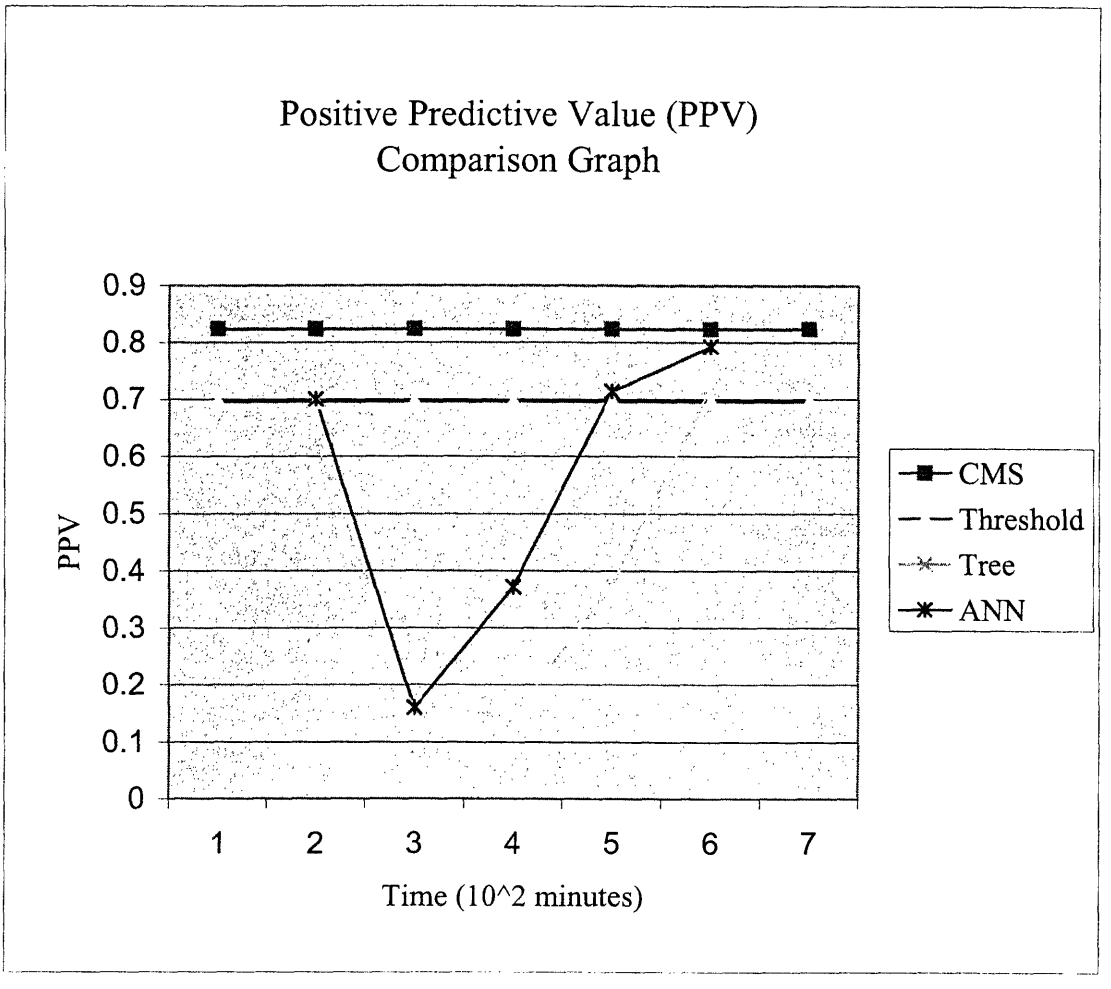


Figure 3.8 Positive predictive value comparison graph

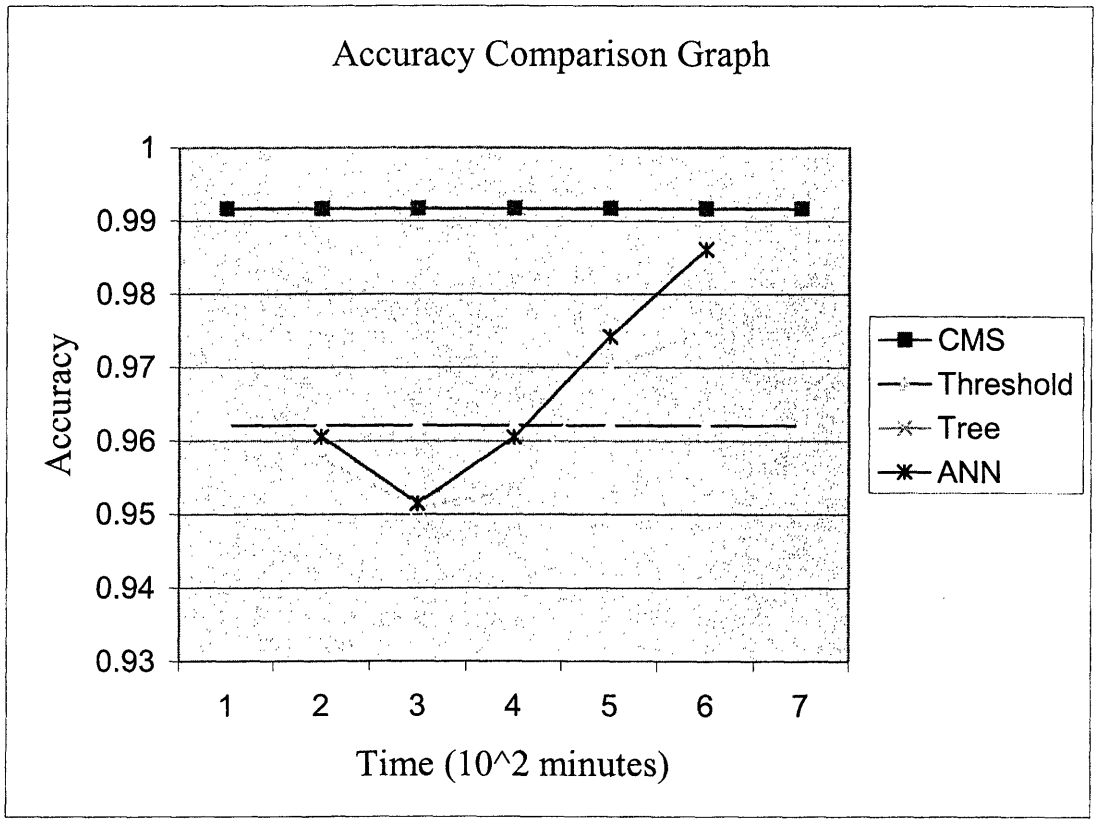


Figure 3.9 Accuracy comparison graph

3.4 Discussion

The results show that the development of alarm algorithms based on machine learning is feasible in real time. Furthermore, these algorithms are useful for integrating multiple physiological signals in detecting adverse clinical events. We conclude this chapter with a discussion of the results and relevant issues arisen during our research.

3.4.1 Real-Time Development of Models

In real-time learning, specifically in developing alarm algorithms, we found that the training time is determined by the type of learning, the core learning algorithm, the number of training examples, and the options for fine-tuning the training process. For example, training neural networks takes more time than training classification trees in general, and this training time increases fast for neural networks as the number of training examples increases. Training options such as boosting and cross-validation can significantly increase the training time, and because the intermediate models during boosting and cross-validation can be different from trial to trial, the training time has a larger variance than training without these options.

Uncertainties in the training time may not affect the overall process if the system does not require instantaneous model generation. However, even in such systems, the allowed timeframe is within a few minutes. Thus, any artificial intelligence techniques that require training time that is over minutes to hours cannot be used for this purpose.

3.4.2 Imbalanced Dataset

Using asymmetric cost function to rebalance the training dataset did not show promising results. Experiments demonstrate that using the cost function in training classification trees not only leads to a higher training error on the training set but also produces a high rate of false alarms. This observation is consistent independent of other training specifications.

In sampling the larger class, the amount of information available for learning the patterns of normal patient conditions could be lost. Thus, the models derived from the final training set are more likely to misclassify a non-alarm instance as in the alarm class.

The results suggest that rebalancing the dataset by expanding the rare class could produce more sensitive and specific models. However, this method is not guaranteed to do so. For example, in Figure 3.x, we saw a classification tree that was overfitted to the training data. Because the training examples in the smaller class were duplicated multiple times, the mechanisms that prevent overfitting, such as pruning of the classification trees, were unable to minimize overfitting to these training examples. Another drawback is that the amount of training data is increased by thousand and tens of thousand fold. The overall effect on the system might not be tolerable in real-time learning.

3.4.3 Feature Selection

Data averages over a time window of the raw data can help eliminate some of the artifacts that create spikes of abnormally high or abnormally low values. As a result, they more accurately reflect the actual value of the parameters. Both the classification tree models and neural network models indicate that the time averages have a larger information gain despite the natural bias in the information gain measure that favors attributes with many values over those with few values.

To improve our system, we have implemented slopes as a new feature. The results are yet to be obtained. The hypothesis is that they could be useful in detecting trends that involve multiple parameters over time. For example, sudden decoupling between two parameters that are highly correlated under normal physiological conditions would suggest the presence of disease processes, adverse reaction to therapy, or both.

3.4.4 Incremental Learning

Comparison of the sensitivity of different alarm algorithms showed that the first models are not sensitive in detecting alarm events. One reason may be that for most session, there were no alarm events during the first 30 minutes. Thus, these models automatically classify all data instances

into the non-alarm class. Even though there were a few sessions with true clinical events during the first 30 minutes and the resulting classification trees and neural networks had better performance, their sensitivity was still below 50%. When averaged over 10 sessions, the overall sensitivity was low.

While the first models were not sensitive, their specificity was higher than any other models, CMS's alarm algorithm(s), and threshold alarm algorithm. This high specificity came from their ability to classify most non-alarm instances correctly. This result is expected, because sensitivity and specificity are inversely related. Also, due to their high specificity, the first trees managed to have a relative high positive predictive value.

Subsequent learning models improved steadily in sensitivity, as on a learning curve. The improvement was moderate for Tree2 and Tree3 and for ANN2, and their specificity and positive predictive value were low. One reason for this phenomenon may be that the models have learned the patterns in the alarm instances, but because the training data did not capture all the patterns of the normal physiological conditions, when these models encounter a new non-alarm event, they are likely to classify it as abnormal, alarm instance.

The fourth and fifth models showed improvements in sensitivity, specificity, and positive predictive value. These models were trained with more data; hence, they may capture more patterns, both of the normal physiological conditions and of abnormal patient states.

The performance of Tree5 and ANN5 approached that of the CMS's alarm algorithm in all four performance metrics. Their positive predictive values also exceed that of the standard threshold alarm algorithm, which represents the alarm algorithms in the last generation of patient monitoring system. This result suggests that more training data may help build a more robust model for the patient, and 8 hours of training data may be adequate.

Overall, the neural network models performed better than the classification trees. They also seemed to improve faster than classification tree learning did. This result may be attributed to the stronger learning capabilities for nonlinear problems.

There are several reasons for the learned models not to perform as well as CMS's alarm algorithm(s). First, alarm detection in the newer generation of patient monitoring system is more sophisticated than that in the last generation. Although the details of the new algorithm(s) have not been disclosed, from publicly available information we know that they include sensitive artifact detection, noise elimination, pattern recognition of specific disease conditions, such as ventricular fibrillation, and some multiple-signal/multi-channel data analysis. For example, from

observations at the bedside and retrospective data analysis, we found that when the value of a parameter went beyond the thresholds, the CMS's alarm system would wait for at least 6 seconds before sounding an alarm.

Another reason is that each product line of patient monitoring systems has its unique platform and operating system that allow fast, real-time processing of data in formats that take up little memory. The monitoring system itself has more processing power than a standard professional laptop. Furthermore, our alarm algorithms were developed from only the numeric data, while the bedside monitor had more information than just the numerics. Thus, it is not surprising that our models did not outperform the CMS's alarm algorithm(s).

Even with much less data and processing power, our algorithms' performances did come close to that of the CMS's algorithms. Furthermore, these models are intended to be part of a larger decision support system for detecting and predicting adverse events. Their value lies in their specificity to individual patient's baseline values and adaptivity to the course of patient outcome. Thus, it is reasonable to predict that with more information, such as waveforms and their features, classification tree learning and neural network learning have promising potential in helping generate intelligent alarm sounding decisions.

Chapter 4
Related Work

Efforts to develop better, more intelligent patient monitoring systems have been ongoing, both in academia and industry. In this chapter, we present related work in four areas: data acquisition in the ICU, understanding patient monitoring and alarms in critical care, intelligent patient monitoring, and real-time systems and design issues. There are necessarily areas of overlap among these topics.

4.1 Data Acquisition in the ICU

Data acquisition is an essential step in the development and evaluation of patient monitoring systems. Due to restricted accessibility of patient data, building systems that collect physiological signals and record clinical information has become a serious research topic. This section presents some of the data collection / annotation systems in the literature.

Tsien and Fackler developed and implemented a prospectively annotated data collection system to support intelligent analysis of intensive care unit data at Children's Hospital, Boston. This system had two components: automated collection of bedside monitor data and computer-based, human operated recording of the clinical events at the bedside. The first component collected numerical data directly from SpaceLab bedside monitor into a laptop via a serial connection at a rate of every 5-6 seconds. This physiological data was saved in a text file. The second component recorded detailed annotations of relevant clinical events such as alarm soundings, staff interventions, equipment malfunction, etc. These annotations were stored in a Microsoft Access database. All the data were time-stamped to allow retrospective analysis and correlation between the clinical events and the actual numeric data during these events. The system was used to collect annotated data over 298 hours. One area for improvement, as noted by the authors, is to make sure that physiological data and clinical annotations are time-stamped in a synchronized manner. [48]

Moody *et al.* developed and implemented a data collection system that acquired signals from the bedside monitors (Hewlett Packard Merlin CMS) in the medical, surgical, and cardiac ICUs of Boston's Beth Israel Hospital. This system communicated with the bedside monitors via two RS232 interface links to a Digiboard PC/4e "smart" serial interface on a standard PC at the monitor's maximum output baudrate. It allowed the recording of up to three ECG signals that were sampled at 500 Hz and four or five other waveforms that were sampled at 125 Hz, or two ECG signals and six other signals. This system also recorded numeric data, the monitor's status messages, and alarm information. Clinical information from the patient's medical records and from the hospital's clinical computing system was recorded separately into three relational databases. The physiological data and clinical information together make up a database called MIMIC, which supports research on multi-parameter intelligent monitoring for intensive care. [30]

Sukuvaara *et al.* constructed a signal preprocessor called DataLog, which continuously collected patient's vital signs from bedside monitors via an RS232 interface. Their system also included a patient data management system (PDMS) that directly connected to the bedside monitor. It provided clinical information for a knowledge-based alarm system, called InCare, which we will describe in section 4.2. [43]

As shown in the three systems described above, Chambrin made the observation that the RS232 interface has been widely used in collecting data from bedside monitors. [6] Eddleman also found that, although more and more bedside monitors are networked and connected to a hospital's LAN system, the RS232 interface has continued to be used in the development of prototype patient monitoring systems because of its position as a *de facto* standard in the medical community. [11]

We do, however, anticipate that future generations of patient monitoring systems will have more networking and data sharing capabilities to allow easy remote access to patient data from the bedside. As a pilot study, Wang *et al.* built a real-time patient monitoring system on the Internet to facilitate intelligent on-line monitoring for the intensive care unit and showed promising test and evaluation results. One of the major concerns for such system is the security of data over the Internet. [51]

Goldstein *et al.* also built a remote data acquisition system for the study of disease dynamics in the intensive care unit. They acquired physiological data from multiple bedside monitors at different ICUs simultaneously via the hospital intranet and the Internet. The advantages of this

system include remote access to data, some patient selection (from any 6 out of the 16 PICU bedside), and continuous data capture on a 24-hour by 7 days schedule. Their study went a step further to analyze the physiological data. Both linear and nonlinear analysis methods in the time and frequency domains were used; most outputs were signal visualizations of data transformations such as autocorrelation, power spectral density, and wavelet correlation. These analyses were carried out retrospectively off-line, and the clinical annotations were being appended to the physiological data retrospectively as well. [14]

Another data acquisition system that employs networking capabilities and the hospital's information system is the data collection system for a temporal ICU patient database called MIMIC II. Saeed *et al.* recorded physiological data and clinical information from patients admitted to an 8-bed medical intensive care unit and an 8-bed coronary care unit from the patient's admission to his or her discharge. The physiological data (2 ECG leads, arterial blood pressure waveform, pulmonary arterial pressure waveform, and 30 1-minute parameters such as heart rate) as well as monitor-generated alarms and INOPs were collected from Philips CMS bedside monitors via an information center database server. Clinical information was automatically captured from the ICU's information system (CareVue by Philips Medical Systems) using the Philips Information Support Mart (ISM). A web-enabled rational database was used to transfer, store, and manage the clinical data. A text search engine facilitated queries of the database. Some of the clinical information was aligned with the 1-minute physiological data, which was also processed using wavelet analysis techniques. [38]

4.2 Understanding Patient Monitoring and Alarms

In general, researchers in patient monitoring believe that continuous monitoring of vital signs and other relevant physiological variables is indispensable in critical care, as the caregiver cannot be at the bedside at all times and measure all signals. [43] However, it was generally acknowledged that the previous generation of patient monitoring systems generated false alarms at an undesirably high rate, as we have noted earlier. Although our studies have shown that the false alarm rate is significantly reduced by distinguishing INOPs from clinical alarms and by more sophisticated alarm algorithms, examining how researchers have tackled the previous

problem with false alarms in the ICU might help us understand the requirements of patient monitoring and the decision process for alarm generation.

For the previous generation of patient monitoring system, there were many factors that contribute to the high false alarm rate. First, these monitoring systems used single variable limit alarms: an alarm was generated if a patient parameter exceeds a threshold determined separately and independently for each signal. Some essential problems with this approach were the difficulty of determining appropriate alarm thresholds, irritations resulting from frequent false positive alarms, and the fact that each signal is assessed separately without considering the clinical context (e.g. the values of other signals or recent medications). Thus, a conventional alarm often could not give appropriate information about the patient's state. It may even be "meaningless in the context of all information about the patient's state." [39] Koski also found that an individual limit alarm does not infer specific physiologic deterioration. [22]

Arnell and Koski proposed that a combination of several signals should be used to generate alarms that are more context relevant. [2, 22] Schecke *et al.* constructed AES-2, a knowledge-based decision support system for patient monitoring in cardioanesthesia. This system incorporated multiple signals and provided an interactive user interface via a touch-screen color graphics monitor. A fuzzy set approach was applied to the uncertainty in decision-making. [39] Bloom demonstrated that the interpretation of any single physiologic variable could be improved by examining its interrelationships with other variables, and that the significance of the composite of these variables varied depending on the clinical state of the patient. [5]

Multivariable monitoring does not imply that using more variables would necessarily lead to better patient monitoring. Mylrea *et al.* showed that sometimes when a smaller number of parameters were considered at the same time, the monitoring produced better results. They examined 10 published reports of operating room critical incidents, accidents, and deaths since 1975 to determine the percentages of anesthesia-related critical incidents that could have been detected with knowledge of patient airway variables. Their estimates indicated that integrated monitoring of "only 5 patient airway variables could potentially identify and warn when 50 to 60% of the preventable anesthesia mishaps occur." [32]

Some variables may also be more informative of a patient's state than others. Koski *et al.* were interested in clinicians' opinions on alarm limits and urgency of therapeutic responses. They found that clinicians view heart rate, end tidal CO2, and systemic arterial blood pressure as the most important vital signs and place less emphasis on the pulmonary arterial pressures. [22]

In addition to multivariable monitoring, safe ranges of monitored vital signs must always be determined based on the individual patient's state. [22] In Makivirta's study, physicians' choice of alarm limits did not vary significantly from the mean limit values observed in postoperative cardiac patients. However, as the variables contain recurrent transient excursions beyond the threshold, these limit values would result in the relatively frequent occurrence of false alarms. [26] This study suggests that the alarm thresholds should be set according to the patient's state instead of completely based on established limits.

Fewer false alarms may be possible by automatically changing alarm limits during a procedure. Shifts also can occur with drug administration, incision, etc. Automated event recognition and limit adjustment, using multivariable analysis, can provide these corrections. Recognition of intubation has been accomplished using measured variables from clinical monitors interfaced to a personal computer. Recorded data from 20 general surgical cases were used to test a rule-based algorithm for detecting changes in oxygen level, breathing rate, and heart rate to identify preoxygenation, start of intubation, and completion of intubation. Fifteen of 19 intubations were recognized, producing a 42% reduction in "low CO2" false alarms. "Determination of appropriate limits could be further enhanced using information on initial patient status and diseases." [32] Moreover, automatically setting the limits at predetermined ranges from the latest variable levels can speed up the adjustment and promote the use of alarm limits. [22]

Manually changing the alarm limits could also increase the positive predictive value of the alarms. Schoenberg *et al.* developed an algorithm that allowed users to first define a physiological trend, such as the difference between the heart rate variability over one minute in the last minute and that from three minutes ago. Then, a threshold for this trend was chosen to yield one of the three outcomes: above the threshold, below the threshold, or unknown due to missing values. Each outcome was assigned a score, and the sum of the scores was compared with a second threshold to determine whether an alarm should sound or not. This algorithm required an expert to manually set its parameters, but it showed that the positive predictive value of the algorithm's alarm exceeded that of the bedside monitor's alarm by 31.9%. [40]

Another area for improvement is the detection and removal of noise from physiological signals. Real-time systems for monitoring and therapy planning, which receive their data from on-line monitoring equipment and computer-based patient records, require reliable data. Horn suggested that "data validation has to utilize and combine a set of fast methods to detect,

eliminate, and repair faulty data, which may lead to life-threatening conclusions. The strength of data validation resulted from the combination of numerical and knowledge-based methods applied to both continuously assessed high-frequency data and discontinuously assessed data. The data validation benefited from the temporal data-abstraction process, which provides automatically derived qualitative values and patterns. The temporal abstraction was oriented on a context-sensitive and expectation-guided principle." [18]

Horn *et al.* constructed VIE-VENT, an open-loop, knowledge-based monitoring and therapy-planning system for artificially-ventilated newborn infants. It consisted of data selection, data validation, data abstraction, data interpretation and therapy planning. The strength of the system was that it used time-point, time-interval, and trend-based methods to validate data. Automatic elimination of invalid measurements resulted in reduced false positive alarm rate. [18]

To increase noise immunity, alarm delays and trend analysis are usually added. [32] Tsien studied four algorithms that filter output signals of a bedside monitor: moving average, moving median, delay, and sampling rate. Moving average algorithms and delay algorithms decreased false alarms up to a particular window size. Moving median algorithms seemed more likely to eliminate true alarms than false alarms. The sampling rate algorithm showed no consistent effect on the positive predictive value of the alarms. [46]

From interviews with neonatal ICU staff, ward observations, and experimental techniques for investigating the role of computerized monitoring in neonatal intensive care, Alberdi *et al.* found that the monitors played a secondary role in the clinicians' decision making and that the ICU staff used the information resources provided by the monitors less often than expected. The study suggested that computerized monitoring could improve through the development of intelligent algorithms, systematic staff training, integration and presentation of clinical information, and better user interfaces. [1]

4.3 Intelligent Patient Monitoring

Studies in patient monitoring have been applying techniques in artificial intelligence and related disciplines to improve patient monitoring systems. A collection of such studies is presented here to give an overview of the area.

Cohn *et al.* modeled the progression of hemodynamic abnormality by a sequence of clinical phases or "scenes", which reflected the predominant physiologic process involved (e.g. increased pericardial pressure, vasodilation, hypotension). A prototype intelligent cardiovascular monitor, DYNASCENE, implemented this paradigm as a parallel process lattice running on a multiprocessor. [8] Bloom used cluster analysis, discriminant analysis, and statistical predictors to identify changes in clinical context. [5]

Sukuvaara *et al.* developed and tested a knowledge-based alarm system for monitoring cardiac operated patients. It consisted of two parts: DataLog and InCare. DataLog was a signal preprocessor for the continuously monitored patient signals. InCare was a knowledge based alarm system that implements 87 rules that helped deduce a specific pathological condition from a combination of measured signals and estimated trends. InCase could continue to operate with incomplete data. Rules used both numeric data and detected trends in the numerical data over a time window. Multiple rules and multiple conditions in the rules were combined by logical OR operator to maintain the reliability of the system when data was incomplete. [43] During a 171.9-hour trial with 35 patients, the sensitivity of the system was 100%, and the specificity was 71%. In the second phase of their study, with 73 cases, the sensitivity of the system remained at 100%, and the specificities for the alarms and for the alerts were 73.9% and 70.0%. [23]

Mylrea *et al.* suggested that addition of pattern recognition capability using neural networks would allow the development of systems that would meet the stringent and complex requirements of the medical environment. Neural networks could be more easily updated than rule-based systems. Two neural networks could also be operated in parallel. [32]

In the words of Kickert and Mamdani, fuzzy control is "the incorporation of the experience of a human process operator; the description of the operator's control strategy by linguistic rules where the words are defined as fuzzy sets; and the main advantage of this approach is the possibility of implementing rules of thumb, experience, intuition, and heuristics without the need for a mathematical model." [20] Fuzzy control has wide applications in industry, because computational algorithms with fuzzy control can derive inferences from vague data using vague logical statements. In medicine, fuzzy control has been used to control pacemaker rate [42] and to monitor left ventricular assist device (LVAD) controller [53]. Since determining the appropriate threshold for individual signals in monitoring devices is difficult, the fuzzy inference approach may be useful in dealing with the vagueness of a precise threshold and in modeling physicians' decision-making process.

In a study by Rau *et al.*, 14 experienced cardioanesthetists formulated a set of defined terms, membership functions for the input parameters, and a knowledge base, which had 188 fuzzy rules. The rule of inference was compositional. [35] Zong et al. used a fuzzy logic approach to analyze the relationship between electrocardiogram and arterial blood pressure waveform in an effort to reduce false arterial blood pressure alarms in the ICU. A fuzzy variable, called "Signal_quality_good" (SQG), was derived from the linguistic variables for describing local waveform characteristics, and it was used to describe the quality of the arterial blood pressure signal. [54]

Another study detected time-varying relationships between physiological variables using graphical modeling. It explored the statistical methodology of graphical models based on partial correlations between different signals. It showed that distinct clinical states of a patient were characterized by distinct partial correlation structures. [19]

Tsien *et al.* used classification tree induction on multiple signals to detect false alarms in the intensive care unit. Features such as the maximum, minimum, range, mean, median, linear regression slope, absolute value of this slope, and standard deviation were calculated for successively overlapping time windows of three-minute, five-minute, and ten-minute durations as inputs into the decision tree learning algorithm C4.5. This study showed that using machine learning techniques such as decision tree induction on derived features from physiological data may be a viable approach to distinguishing false alarms from true positives in the ICU. [47] Tsien went further to detect "true alarm" situations in the ICU using a pipeline for event discovery in medical time-series data. This study demonstrated that machine learning techniques, such as decision tree classifiers, neural networks, logistic regression, radial basis function networks, and support vector machines, were useful in discovering knowledge from physiological data and their correlation with clinical events. [48, 49]

4.4 Real-Time Systems, Design Issues, and Decision Support

According to Laplante, a real-time system is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failure. It is one whose logical correctness is based on both the correctness of the outputs and their timeliness. [24]

Some of the design issues are: 1) the selection of hardware and software; 2) the decision to take advantage of a commercial real-time operating system or to design a special operating system; 3) the selection of an appropriate software language for system development; 4) the maximizing of system fault tolerance and reliability through careful design and rigorous testing; 5) the design and administration of tests, and the selection of test and development equipment. [24]

For real-time systems, a major problem is maintaining consistency, both temporal and among the data, between the computerized model and the process to be managed. The first question is one of precision in the representation of time. For ONCOCIN, the maximally precise unit of time was the day. SEPIA's precision was one minute. GUARDIAN selected its unit of timeaccording to the working context of the system. The second question rises from the synchronization of the computer process and the real process being monitored: gaps can occur if the system must wait too long for a piece of information, or if the computer crashes. [31]

Sampling rate of physiological signals must be high enough to yield useful information about the patient's state. Schecke *et al.* suggested that their approach required a very precise data recording; sampling interval of vital signs must be considerably shorter than 1 minute. A comprehensive semi-automatic anesthesia information and documentation system is a prerequisite. [39]

User-interface is an important part of designing patient monitoring systems. Coiera proposed a user and dialogue modeling approach for the development of user interfaces for intelligent patient monitoring systems. He believed that this method could facilitate communication between human and computer. He pointed out that the emphasis should be on the process of development of an interface rather than the final product. Furthermore, the process of development should follow the user's natural cognitive processes and structures. [7]

Very few references on real-time learning on medical data or real-time decision support for critical care have been found in the literature. One study presented the architecture of an intelligent alarm system for patient monitoring during anesthesia, called the Adaptive Real-Time Anesthesiologist Associate (ARTAA). It planned to implement a hybrid expert system based on neural networks and fuzzy logic theories for real-time and adaptive detection of which monitor is connected to the patient and common machine malfunctions. [15] Another study by Fried *et al.* compared autoregressive models, phase space models, and dynamic linear models for online detection of artifacts, baseline change, and trends that could help classify the patient's state in

retrospective case studies on physiological data from 19 critically ill patients. They showed that no single statistical methodology could model all the patterns in physiological time-series data and suggested a combination of methods and pattern-specific models could achieve better results. [13]

Chapter 5
Conclusion

In this thesis, we have presented the design, implementation, and evaluation of a system for synchronized collection of physiological signals and clinical annotations, and then described an expansion of this system for real-time learning in the application of developing alarm algorithms for patient monitoring systems. We conclude with a summary of our studies and findings, and questions for future research.

5.1 Studies and Findings

We began, in Chapter 2, by outlining the structure of the system for synchronized collection of physiological signals and clinical annotations and describing the design of its functional components. During the evaluation of this system and the subsequent study, we found that this system achieved its design goals and enabled time synchronization and accurate correlation between physiological data and clinical annotations.

Using this system, we collected numerical time-series data from the patient monitoring system and clinical event annotations at a bedside in a typical pediatric medical ICU. We found that the new monitor generated alarms at a frequency much lower than what had been reported in the literature. The finding that very few of these alarms were false positives further contrasts with the high false alarm rates observed in previous studies.

In discussion, we attempted to explain why our results gave a seemingly complete picture of the state of patient monitoring. One of our hypotheses was tested in the subsequent study of this research; it clearly showed that the new generation of patient monitoring systems, including the one used in our research, has significantly improved data analysis capabilities and alarm algorithms.

In Chapter 3, we presented learning in real time as a novel approach to help develop patient-specific algorithms for patient monitoring. We first explored the feasibility of this approach in the critical care setting by training and evaluating classification tree models and neural network models to detect adverse events at the bedside. Then, we assessed the utility of this approach by carrying out classification tree learning and neural network learning at incremental time intervals in an adaptive manner.

Our expanded system of real-time data collection and algorithm development demonstrated that learning in real time is a feasible approach to developing alarm algorithms. Performances of the trained classification trees and neural networks were consistent with the course of a generalized learning process. The ones that are trained with eight hours of monitored numerics data outperformed the standard threshold alarm algorithm, which represented the alarm algorithms in previous generations of patient monitoring systems, and came close in performance to the alarm algorithm(s) in the new-generation monitors. Contrary to our initial expectations, our individualized learned monitoring algorithms did not improve on the current generation of alarm methods incorporated in proprietary monitoring equipment.

5.2 Questions for Future Research

In this section, we elaborate four key questions that have arisen from our studies. We will also discuss relevant ideas for future research in patient monitoring.

What should be the gold standard for event classification in the ICU?

Correct event classification is central to obtaining a useful annotated dataset and developing intelligent alarm algorithms. As described in Chapter 2 and Chapter 4, event classification has relied on human experts as the gold standard. Yet, we know that asking either the nurses to classify the alarm at the bedside or experienced physicians to do so retrospectively based on known patterns in the physiological data could bias the annotation toward misclassifying a true alarm as a false positive because of inadequate medical knowledge or clinical information. Decisions made with this gold standard are also subject to inter-observer variability.

In our study, we used a combination of human experts at the bedside and patient outcome within a windowed timeframe after the event as a new gold standard for event classification. While our results contain no instances where future patient outcome was used to revise the classification by human experts, they do not suggest that incorporating patient outcome as a part of the gold standard is not necessary nor that classification by human experts alone is sufficient.

Since future patient outcome could yield significant information about the current patient state because living system and disease processes are causal and memory driven, one idea for future research is to carry out a rigorous evaluation of the proposed gold standard, a study that goes beyond the scope of this research. Another idea is to identify new sources of information that could either facilitate or validate event classification. It is also important to develop a gold standard that allows comparison of results across studies.

How to best deal with the missing value problem in real time?

As described in Chapter 3 and Chapter 4, the data from patient monitoring systems often have missing values for one or more parameters, either transiently in time or consistently over minutes and hours. These missing values cannot be simply ignored or set to zero; parameters that frequently have missing values cannot be simply removed from the study. Thus, we need a systematic approach to deal with the missing value problem.

In our study, we recorded both the transient and consistent missing values during data collection. We relied on the core learning algorithms' specialized mechanisms for dealing with missing values in real-time learning. We need to investigate whether there are more effective ways to handle missing values other than what we have tried. These methods should offer a general, systematic approach to the problem and yet could be easily implemented for specific studies.

What features of clinical time-series data are most informative of patient condition?

As we discussed in Chapter 3, features that are derived from clinical time-series data may capture patterns in the data and allow easy detection of adverse events by machines or humans. There are many features that can be derived, such as averages, slopes, statistical characteristics, frequency characteristics, etc., and many more have yet to be derived and examined. Researchers seem to

know that a wealth of information in clinical time-series data have not been extracted and utilized, but there is not a systematic way of deriving and selecting new features.

Due to the time constraint on feature derivation, we used the basic and widely used feature, time averages, in our study. While we are currently experimenting with slopes and simple statistical characteristics, we would like to have in the near future a repertoire of features that yield high information gains and are quick to derive. Furthermore, we would like to know which features to use in what kind of problem.

In real-time learning, is it more effective to adaptively update existing models or to build a new set of models with newly obtained patient data?

Real-time learning is a potentially useful approach to discover knowledge and to provide decision support in many real-world applications. It is a new concept, and the methodologies for this approach are yet to be formulated and tested. In Chapter 3, we presented a study that explored real-time learning in the medical domain. Two machine learning techniques were employed to build models that help generate alarm-sounding decisions in incremental time intervals, successively with more patient data. Our results showed that learning from scratch requires patient-specific data from a sufficient period of time. Thus, in providing clinical decision support for a specific patient, we still need to use the knowledge or models from a larger, relevant patient population during this period.

An interesting exploration would be to adaptively update existing models instead of building new ones from scratch. The questions would then be how to do it, which of the two approaches has more advantages and fewer limitations, and are their performances highly dependent on clinical context.

One idea for updating existing models is to use N hours of training data from a group of patients with similar conditions and add the data from the current patient to the training dataset. This method might dramatically improve the performance of the learned models before all the data from this patient can capture possible clinical states. The hypothesis is that the models derived from a larger patient population are better than the relative ignorance of the newly learned models derived from only limited amounts of data from this particular patient. Nevertheless, as we monitor this patient for more and more hours, at the Nth hour, we would have about equal amounts of population and individual data to train from, at 2xNth hour, we would

have twice as much data from the individual. Thus, we should eventually get a very specific model for your patient. While the details of this method are yet to be worked out, some interesting questions already arise: how to select the initial N hours of training data; what value should N take on, is it dependent on the clinical context, and if so, how to determine it under the constraints of time and data processing power for real-time learning.

5.3 Summary

From an engineering standpoint, this research developed a computer-based system and realized real-time learning using artificial intelligence techniques for a real-world application. From a clinical standpoint, it facilitated the understanding of physiological signals at the bedside and may help improve clinical decision-support systems. Overall, this research demonstrated that obtaining useful information in an environment that is over-loaded with data is a challenging but feasible task. It also showed that unexpected results could lead to a better understanding of and creative ways to tackle a multidisciplinary problem. Our challenges in medical engineering research will continue to be formulating problems arising from specific applications such as patient monitoring in terms of more general engineering problems and balancing between application and theory to produce specific engineering solutions for patient care.

References

1. Alberdi E, Gilhooly K, Hunter J, Logie R, Lyon A, McIntosh N, Reiss J. Computerisation and decision making in neonatal intensive care: A cognitive engineering investigation. *Journal of Clinical Monitoring and Computing*. 2000; 16: 85-94.

2. Arnell WJ, Schultz DG. Computers in anesthesiology - a look ahead. *Medical Instrumentation*. 1983; 17: 385-93.

3. Becker K, Thull B, Kasmacher-Leidinger H, Stemmer J, Rau G, Kalff G, Zimmermann HJ. Design and validation of an intelligent patient monitoring and alarm system based on a fuzzy logic process model. *Artificial Intelligence in Medicine*. 1997; 11(1): 33-53.

4. Beneken JEW and van der Aa JJ. Alarms and Their Limits in Monitoring. *Journal of Clinical Monitoring*. 1989; 5(3): 205-210.

5. Bloom MJ. Techniques to identify clinical contexts during automated data analysis. *International Journal of Clinical Monitoring and Computing*. 1993; 10(1): 17-22.

6. Chambrin MC, Ravaux P, Chopin C, Mangalaboyi J, Lestavel P, Fourrier F. Computer-assisted evaluation of respiratory data in ventilated critically ill patients. *International Journal of Clinical Monitoring and Computing*. 1989; 6(4); 211-215.

7. Coiera E. Incorporating user and dialogue models into the interface design of an intelligent patient monitor. *Medical Informatics*. 1991; 16(4): 331-346.

8. Cohn AI, Rosenbaum S, Factor M, Miller PL. *Methods of Information in Medicine*. 1990; 29(2); 122-131.

9. Coleman WP, Siegel JH, Giovannini I, Sanford DP, Gaetano AD. Computational logic: a method for formal analysis of the ICU knowledge base. *International Journal of Clinical Monitoring and Computing*. 1993; 10(1), 67-69.

10. Cunningham S, Deere S, Elton RA, McIntosh N. Neonatal physiological trend monitoring by computer. *International Journal of Clinical Monitoring and Computing*. 1992; 9(4): 221-227.

11. Eddleman DW, Tucker DM, McEachern M. A patient monitoring system designed as a platform for application development. *International Journal of Clinical Monitoring and Computing*. 1990; 7(4), 233-240.

12. EasyNN-plus Help Manual. Neural Planner Software. 2003.

13. Fried R, Gather U, Imhoff M. Online Pattern Recognition in Intensive Care Medicine. *Proceedings of American Medical Informatics Association Annual Fall Symposium.* 2001; 184-188.

14. Goldstein B, McNames J, McDonald BA, Ellenby M, Lai S, Sun Z, Krieger D, Sclabassi RJ. Physiologic data acquisition system and database for the study of disease dynamics in the intensive care unit. *Critical Care Medicine.* 2003, Feburary; 31(2): 433-441.

15. Guez A, Nevo I. Neural networks and Fuzzy logic in clinical laboratory computing with application to integrated monitoring. *Clinica Chimica Acta (International Journal of Clinical Chemistry).* 1996, Nov; 8(6): 543-576.

16. Haimowitz IJ. Intelligent Diagnostic Monitoring Using Trend Templates. *In:* Symposium on Computer Applications in Medical Care. American Medical Informatics Association, 1994, pp 702-708.

17. *HP Viridia Component Monitoring System RS232 Computer Interface Programming Guide.* 4th Ed. Hewlet Packard. 1998.

18. Horn W, Miksch S, Egghart G, Popow C, Paky F. Effective data validation of high-frequency data: time-point-, time-interval-, and trend-based methods. Computers in Biology and Medicine. 1997; 27(5): 389-409.

19. Imhoff M. Detecting Relationships Between Physiological Variables Using Graphical Modeling. *Proceedings of American Medical Informatics Association Annual Fall Symposium.* 2002; 340 –344.

20. Kickert WJM, Mamdani Ej. Analysis of a fuzzy logic controller. *Fuzzy Sets Systems.* 1978; 1: 29-44.

21. Koski EMJ, Makivirta A, Sukuvaara T, Kari A. Frequency and reliability of alarms in the monitoring of cardiac postoperative patients. *International Journal of Clinical Monitoring and Computing.* 1990; 7: 129-133.

22. Koski EMJ, Makivirta A, Sukuvaara T, Kari A. Clinicians' opinions on alarm limits and urgency of therapeutic responses. *International Journal of Clinical Monitoring and Computing.* 1995; 12(2): 85-88.

23. Koski EMJ, Sukuvaara T, Makivirta A. A knowledge-based alarm system for monitoring cardiac operated patients---assessment of clinical performance. *International Journal of Clinical Monitoring and Computing.* 1994; 11(2): 79-83.

24. Laplante PA. *Real-Time Systems Design and Analysis.* IEEE: New York, 1997.

25. Lawless ST. Crying wolf: False alarms in a pediatric intensive care unit. *Critical Care Medicine.* 1994; 22(6): 981-985.

26. Makivirta A, Koski EMJ. Alarm-inducing variability in cardiac postoperative data and the effects of prealarm delay. *Journal of Clinical Monitoring*. 1994; 10: 153-162.

27. Meredith C and Edworthy J. Are there too many alarms in the intensive care unit? An overview of the problems. *Journal of Advanced Nursing*. 1995; 21: 15-20.

28. Mitchell TM. *Machine Learning*. WCB McGraw-Hill: Boston, 1997.

29. Momtahan K, Hetu R, Tansley B. Audibility and identification of auditory alarms in the operating room and intensive care unit. *Ergonomics*. 1993; 36(10): 1159-1176.

30. Moody GB, Mark RG. A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring. Computers in Cardiology. 1996; 23: 657-660.

31. Morice V, Seroussi B, Boisvieux JF. A Real Time Control Architecture for Continuously Managing Patients in a Care Unit. *Methods of Information in Medicine*. 1995; 34(5): 475-488.

32. Mylrea KC, Orr JA, Westenskow DR. Integration of monitoring for intelligent alarms in anesthesia: neural networks – can they help? *Journal of Clinical Monitoring*. 1993; 9(1): 31-37.

33. Quinlan, JR. Induction of decision trees. *Machine Learning*. 1986; 1(1), 81-106.

34. Quinlan, JR. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

35. Rau G, Becker K, Kaufmann R, Zimmermann HJ. Fuzzy Logic and Control: Principal Approach and Potential Applications in Medicine. *Artificial Organs*. 1995; 19(1): 105-112.

36. Russ, TA. *Reasoning with time dependent data*. PhD thesis, Massachusetts Institute of Technology, August 1991.

37. Russ, TA. Use of data abstraction methods to simplify monitoring. *Artificial Intelligence in Medicine*. 1995; 7(6): 497-514.

38. Saeed M, Lieu C, Raber G, Mark RG. MIMIC II: A massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in Cardiology*. 2002; 29: 641-644.

39. Schecke T, Langen M, Popp HJ, Rau G, Kasmacher H, Kalff G. Knowledge-based decision support for patient monitoring in cardioanesthesia. *International Journal of Clinical Monitoring and Computing*. 1992; 9(1): 1-11.

40. Schoenberg R, Sands DZ, Safran C. Making ICU alrms meaningful: a comparison of traditional vs. trend-based algorithms. *Proceedings of American Medical Informatics Association Fall Symposium.* 1999; 379-383.

41. See5 Help Manual. RuleQuest Research. 2003.

42. Siguira T, Mizushina S, Kimura M, Fukui Y, Harada Y. A fuzzy approach to the rate control in an artificial cardiac pacemaker regulated by respiratory rate and temperature: A preliminary report. *Journal of Medical Engineering and Technology.* 1991; 15: 107-110.

43. Sukuvaara T, Koski EMJ, Makivirta A, Kari A. A knowledge-based alarm system for monitoring cardiac operated patients - technical construction and evaluation. *International Journal of Clinical Monitoring and Computing.* 1993; 10(2): 117-126.

44. Tsien CL and Fackler JC. An annotated data collection system to support intelligent analysis of intensive care unit data. *In:* Advances in Intelligent Data Analysis. Liu X, Cohen P, Berthold M (editors). Berlin, Springer-Verlag, 1997, pp. 111-121.

45. Tsien CL and Fackler JC. Poor prognosis for existing monitors in the intensive care unit. *Critical Care Medicine*, vol. 25, no. 4, April 1997, pp. 614-619.

46. Tsien CL. Reducing false alarms in the intensive care unit: a systematic comparison of four algorithms. *Proceedings of the American Medical Informatics Association Annual Fall Symposium*, October 1997.

47. Tsien CL, Kohane IS, McIntosh N. Multiple signal integration by decision tree induction to detect artifacts in the neonatal intensive care unit. *Artificial Intelligence in Medicine.* 2000, Jul; 19(3): 189-202.

48. Tsien CL. TrendFinder:Automated detection of alarmable trends. Laboratory for Computer Science Technical Report 809, Massachusetts Institute of Technology. July 2000.

49. Tsien CL. Event discovery in medical time-series data. *Proceedings of American Medical Informatics Association Fall Symposium.* 2000; 858-62.

50. Uckun S. Intelligent systems in patient monitoring and therapy management. *International Journal of Clinical Monitoring and Computing.* 1994; 11(4): 241-253.

51. Wang K, Kohane I, Bradshaw KL, Fackler J. "A real time patient monitoring system on the World Wide Web." Proceedings of the American Medical Informatics Association Annual Fall Symposium, 1996.

52. Westenskow DR, Orr JA, Simon FH, Ing D, Bender HJ, Frankenberger H. Intelligent Alarms Reduce Anesthesiologist's Response Time to Critical Faults. *Anesthesiology.* 1992; 77(6); 1074-1079.

53. Yoshizawa M, Kuramoto K, Takeda H, Miura M, Yambe T, Nitta S. An automatic monitoring and estimation tool for the cardiovascular dynamics under ventricular assistance. *Proceedings of the 14th International Conference on IEEE Engineering and Medical Biology*. 1992; 1: 364-366.

54. Zong W, Moody GB, Mark RG. Reduction of False Blood Pressure Alarms by use of Electrocardiogram Blood Pressure Relationships. *Computers in Cardiology*. 1999; 26: 305-308.

Appendix A

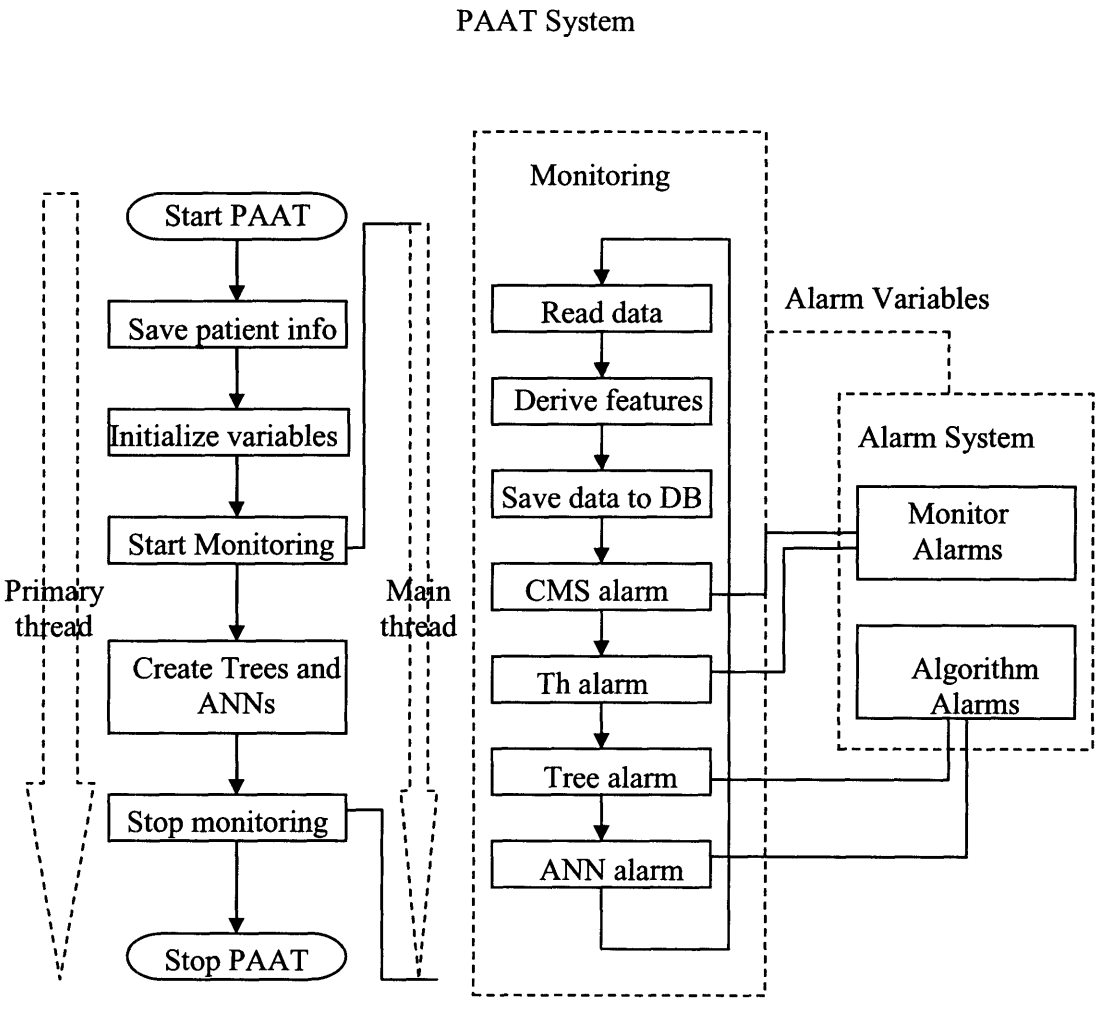


Figure A.1 The primary thread and main thread in PAAT

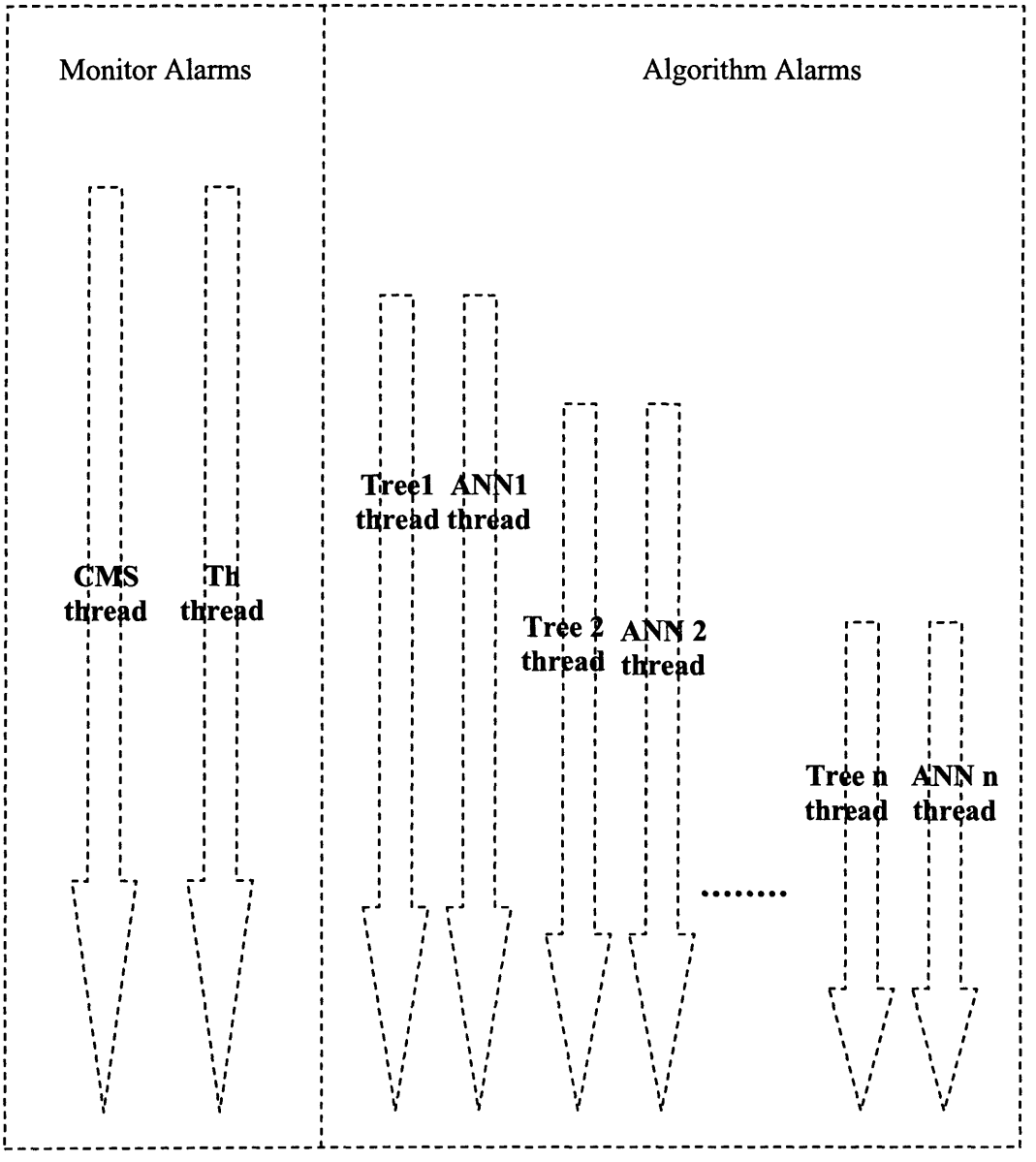


Figure A.2 Multiple threads for incremental learning

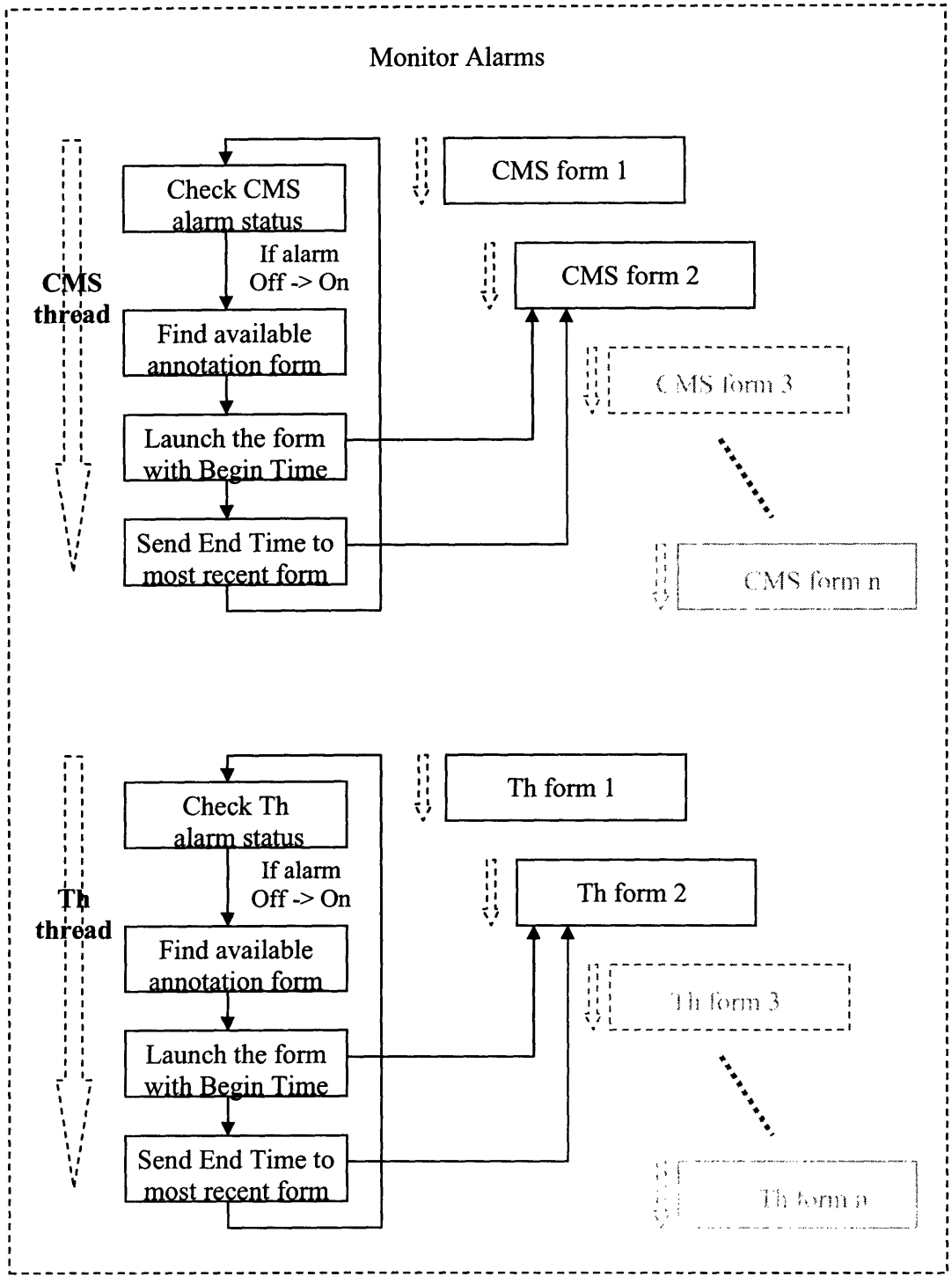


Figure A.3 Threads for CMS alarm annotations and threshold alarm annotations

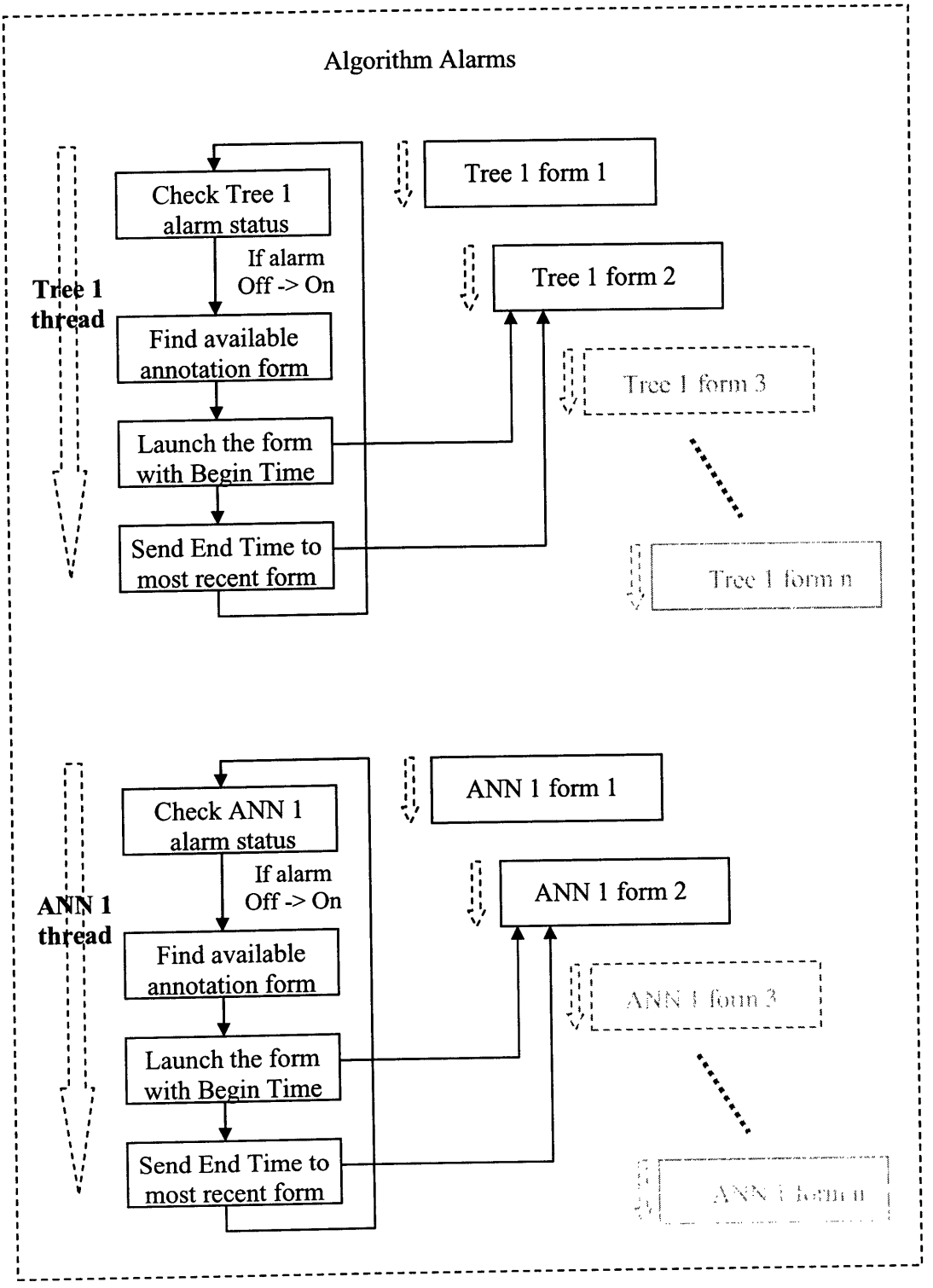


Figure A.4 Multiple threads for algorithms' alarm annotations