A PREDICTIVE LEARNING CONTROL SYSTEM FOR AN ENERGY CONSERVING THERMOSTAT

by

DAVID WAYNE PAYTON

B.S.E.E., University of California, Los Angeles
(1979)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1981

Signature of Author _____
Department of Electrical Engineering and Computer Science
August 28, 1981

Certified by _____
Edward Fredkin
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Departmental Committee on Theses

A PREDICTIVE LEARNING CONTROL SYSTEM FOR AN ENERGY CONSERVING THERMOSTAT

by

DAVID WAYNE PAYTON

Submitted to the Department of Electrical Engineering And Computer Science
on August 31, 1981 in partial fulfillment of
requirements for the Degree of Master of Science in
Electrical Engineering and Computer Science

ABSTRACT:

It has been well established that energy can be saved in heating or cooling buildings if comfortable temperatures are maintained only when people are present. Many systems have been developed which use timers to control temperature settings according to people's daily cycles, but these often require excessive interaction with the user.

This thesis presents a novel application of machine learning concepts to enable a system to automatically learn and predict people's behavior patterns. With sensors to detect the presence of people in different parts of a building, the predictive ability is used to anticipate where people will be so as to accomodate for the delay between the time the temperature setting is changed and the time the desired temperature is finally obtained. Since the system learns behavior patterns automatically, it requires very little user interaction to do a good job of saving energy and maintaining user comfort.

A computer simulation of this system has been developed and tested on genuine behavior data. Results from these tests show that the system can save from two to three times the energy saved with night setback alone while providing very reasonable comfort levels as well.

Thesis Supervisor: Edward Fredkin
Title: Professor of Electrical Engineering and Computer Science

CONTENTS

FIGURES

ACKNOWLEDGMENTS

I would like to thank the many people who have helped me on this work. I am especially grateful to Edward Fredkin for providing valuable guidance as well as the initial inspiration for the concept of an intelligent thermostat. It was his insistence on the many practical considerations of this project that has helped to make it more than just a study of machine learning.

I would also like to thank the people from Honeywell Inc. for a great deal of advice related to making an effective energy conserving thermostat. In particular, Lorni Nelson, Ron Benton, Tom Beckey, Cliff Nelson, John Fechter, Ralph Torborg, Roger Moen and Dr. Gerald Dinneen provided valuable insights into the many practical details of thermostat design. I am particularly grateful to Lorni Nelson for allowing me to copy graphs from his article on "Energy Savings Through Thermostat Setback".

I appreciate the valuable discussions and comments from Robert Gianseracusa, Tommaso Toffoli, Ramesh Patil, and Marvin Minsky on the technical aspects of my approach to learning.

Special thanks also goes to the people who helped me obtain my test data. Karen Weininger, Andreas Mortensen, and my parents each graciously went through the tedium of recording their daily behavior patterns for more than two months. Without this help, I would never have been able to confirm the validity of my algorithm.

I am also grateful to the MIT Artificial Intelligence Laboratory whose tremendous resources made this work possible. With the powerful programming environment provided by the LISP Machine and the LISP language, this work progressed much faster than it could have under ordinary conditions.

My financial support has been generously provided by the Hughes Aircraft Corporation through their Howard Hughes Fellowship program.

1. INTRODUCTION

A wide variety of automatic temperature control systems have been available for homes and industry for quite some time. These systems range from simple thermostat controls to sophisticated computerized timing systems. In all systems there is a trade-off between energy consumption and user comfort. Occasionally, this trade-off may be altered by reducing energy consumption while still maintaining user comfort, but this generally involves greater system complexity. Such complexity is often self defeating because it makes the system more difficult to use due to unwieldy human interface requirements. I am proposing a control system which uses learning to tune itself to the user's behavior patterns. This permits significant energy savings with a minimal sacrifice to user comfort and allows the user interface to remain fairly simple.

Simple thermostat controls offer a good level of user comfort, but can be fairly wasteful of energy. These systems may either be manually set to maintain a desired temperature or they may be turned off entirely. If the system is not turned off or the temperature set back when people are not present, then energy may be wasted heating or cooling unoccupied space. If a conscientious user turns down the temperature setting when he knows he will not be present, he will have to endure a period of discomfort after turning the setting back up when he returns.

Systems which can be programmed to regulate temperature according to the time of day and the day of the week can make a noticeable improvement in energy consumption over standard thermostats. This is because these systems can be made to take advantage of times when no heat is needed without requiring constant user interaction. A programmed thermostat can be set to automatically turn down the temperature setting at night when people are sleeping and in the daytime when people are at work. The savings that can result from such a scheme have been shown to be fairly significant [Nelson and MacArthur,78] [Beckey and Nelson,81]. Furthermore, if people's behavior patterns are sufficiently regular,

then these systems can be set to turn up the temperature setting before people arrive so that the proper temperature has been reached when people need it. The difficulty these systems have is that human behavior is not completely dependent on the time of day. This makes it hard for people to accurately anticipate their behavior patterns since they cannot account for the many irregularities that might occur. Furthermore, many people do not wish to spend the effort required to obtain optimal settings for their personal needs. Without this effort, much of the potential benefit from these systems is lost.

There is potential for even greater savings if programmable thermostats could be used in conjunction with a zoning system [Tobias,75]. This would allow specific zones to be heated only when they are used, reducing the amount of energy wasted on heating areas that are rarely used. Unfortunately, when we add more zones to a programmable thermostat system, we make the interface problems much worse. Not only would the user have many more timers to set, but the patterns that he had to anticipate would be much more complicated and irregular.

It is clear that temperature control systems can be significantly improved by providing thermostats with advance knowledge of when various areas in a building will or will not be occupied. This would allow thermostat settings to automatically be turned up before areas are occupied and turned back down when areas are vacated, providing people with an ideal temperature upon entry and still minimizing the energy wasted on empty space. For such systems to be truly useful, however, it it is necessary to obtain this knowledge with a minimum amount of user interaction. A partial solution to this problem may be obtained through the use of people sensing devices such as common burglar alarms which could tell the system which zones are occupied at any given time without the need for user interaction. This solution is insufficient for two reasons. First, it fails to provide the desired advance warning. This means that although energy can be saved on all unoccupied zones, people would always have to endure a period of discomfort if they should enter any area that had not been recently

used. A second problem is that a control system that relies only on the presence of people to make its temperature settings will cycle every time people enter and leave the controlled area. If this cycling occurs frequently, it can be both wasteful of energy and detrimental to the heating or cooling unit [Bonne,Janssen,Nelson,and Torborg,76]. It is the goal of my predictive learning control system to provide both the needed advance warning and the short term stability to improve comfort and to reduce unnecessary cycling in a system using people sensors. This will be done by predicting near term future behavior by learning from past observed behavior patterns.

Before we may actually predict future behavior from a knowledge of past behavior, we must consider the reasonable assumptions that may be made about human behavior patterns that permit such prediction. One assumption is that people tend to engage in certain time dependent activities. Whether it be sleeping, eating, or going to work, people often have definite time constraints for the things they do. Another assumption is that people have many sequential behavior patterns as well. Someone who always takes a shower in the morning before going to the kitchen for breakfast exhibits a perfect example of this type of pattern. Knowledge of a sequential pattern like this should enable us to get a better estimate of when the kitchen will be occupied by observing when the bathroom was first entered. Other sequential patterns may be envisioned based on the typical duration of various activities and the interaction between several individuals. A final assumption is that behavior patterns will repeat, though not necessarily on a day to day basis. This means, for example, that what happens on a work day may be different from what happens on a weekend, but most work days will be similar, and certain weekends will be similar to one another as well. More specifically, we may consider each individual as having a repertoire of many classes of behavior patterns from which he chooses one on any particular day. The choice of behavior class on a given day should have a certain degree of regularity. This choice may be related to behavior on the previous day, the previous week, or it may have some more obscure periodicity.

Because most of the information my system needs will be supplied by the people detection units, the user interface requirements for my system can be extremely simple. The major requirements of my system will be an input for temperature settings, an input of the user's subjective preference for comfort versus efficiency, and a method for the user to inform the system about special days such as holidays and vacations. The comfort versus efficiency control will serve to bias whatever prediction errors the system makes either toward those that have the temperature setting turned up more often than needed, or toward those that have it turned down too often. In the extremes, this will cause the system to operate either as a standard thermostat, or as a system depending solely on people sensors. The special day information will allow the system to anticipate unusual behavior patterns. This information will be simple to obtain since the user will only have to provide a date and an arbitrary name if he wishes to input a special day to the system. This task is made even simpler by the fact that the manufacturer can initialize the system with a knowledge of all legal holidays. Since the information requirements of this system are limited and at a level that the user can easily understand, the interface problem of timed thermostats is overcome.

The system discussed in this thesis is intended to be easily implemented at a reasonable cost. There are several factors which should make this possible. First, the system employs classification procedures which form generalizations from daily patterns, thereby reducing the amount of memory required. Second, even though the algorithm I use is computationally intensive, the problem requires very little speed since the time between important events is on the order of many minutes. Without a speed requirement, the entire job could be handled by a fairly inexpensive microprocessor. This leads to the final point which is that the cost of the microprocessors and memory that would be needed for the implementation of this system is still falling rapidly. Already, microprocessors are available complete with memory and interface circuitry all on a single chip. Since the price of these devices is dropping so quickly, it is important to realize that powerful software provides the key for taking advantage of

this tremendous resource. With the cost of computational power continuously falling, and the cost of energy continuously rising, it is clear that the cost effectiveness of any advanced temperature control system will depend more on its ability to conserve energy than on its computational requirements.

The learning control system I have designed consists of three main parts. The first is a method for obtaining a probabilistic representation from the behavior patterns of a given day such that if these patterns were to be repeated, an accurate prediction of which areas in a building should be regulated could be obtained. The second part of my system is a method for classifying days according to similarities in their probabilistic representations. This allows similar days to be grouped into a single class, and permits the compression of data from many days into a simple, compact form. The third part is a method for predicting which class to use in the decision process on an upcoming day, combined with a technique for altering that prediction should it prove to be a poor one. This involves periodic analysis of the sequence of classes, analysis of interrelationships between classes, as well as analysis of relationships between classes and user specified special days such as holidays and vacations.

The system I have developed employs several different ideas about learning. A low level probabilistic form of learning is used to interpret basic behavior patterns. Some special aspects of my use of this method are the implementation of a dynamically adjusted weight factor which maintains the proper tradeoff of comfort versus savings, and the use of the probabilistic representation as a classification vehicle. At a higher level, a learning system is used to predict sequences of classes. This system learns with the aid of knowledge that is provided by the programmer and by the user of the system. When combined, these two approaches to learning result in an effective and versatile predictive learning system.

2. THE DECISION FUNCTION

The decision to set the thermostat up or down in a zone can be based on probabilistic considerations. The two key influencing factors in these considerations are the time of day and the occupancy patterns of the recent past. The time of day is important because many activities are completely time dependent. For activities which depend more on the involvement in other activities beforehand, it is likely that the occupancy patterns of the recent past will have the most significance. If a probabilistic relationship between these factors and the decision choices can be obtained, then a useful decision algorithm can be derived.

For any given time of day, it is possible to determine the a priori probability that the thermostat should be set up at that time. This probability is based on the number of occasions that the higher setting has been required at the given time on previous days. In addition to this, knowledge of the recent occupancy patterns for each zone will also be available. Using information accumulated from past experience, it is possible to estimate the probability of seeing the recently observed patterns given that the thermostat must indeed be set up. Combining this probability with the a priori probability, we obtain the likelihood that the thermostat should be set up. When this likelihood is compared with the likelihood that the thermostat should be set back, an appropriate choice of action can be made. This decision method is very appropriate for formulation as a maximum likelihood decision algorithm [Minsky and Selfridge,60] [Papert,61] [Minsky and Papert,69] [Edwards,72].

In this technique, we must define a set of binary partial predicates or evidence factors $\phi_{i,l}$ which are statistically independent for each possible decision type F_j. In this particular problem, there are only two decision types of concern:

F_1 = The decision to turn up the thermostat because people are present or will be soon.
F_2 = The decision to turn down the thermostat because no need is expected.

Note that we assume here, as we will for the remainder of this discussion, that we are dealing with only a heating problem. This is done merely to simplify the description of the problem,

and it should remain clear that everything that applies to heating will apply to cooling as well.

I have chosen to use a set of evidence factors which are based on the past 24 hours of observed behavior in each zone. All we may know about past behavior will be whether or not each zone was occupied by one or more people at a given time, and even this data will quantized into short time slices to simplify machine representation. The quantization will be performed such that occupation of a zone is registered during a given time slice only if the zone is in fact occupied for more than a fixed percentage of that time. Each evidence factor $\phi_{i,l}$ then, is associated with a particular zone and a time slice which occurred a fixed number of time quanta prior to the present time, where the i subscript refers to the number of time quanta and the l subscript refers to the zone. An evidence factor is assigned a value of "1" or "0" depending on whether or not its associated zone was occupied during its associated time slice. Thus, $\phi_{4,1}$ will have a value of "1" if zone 4 was occupied 1 time slice prior to the present time, and $\phi_{6,3}$ will have a value of "0" if zone 3 was unoccupied 6 time slices prior to the present time. Note that since each $\phi_{i,l}$ is defined in terms of time slices relative to the present, their values will be changing with time.

Since this algorithm theoretically requires the evidence factors to be independent over both decision types F_1 and F_2, we must at least convince ourselves that they are approximately independent for the problem at hand. For the independence assumption to be valid, it is necessary that the knowledge that certain zones have been occupied by one or more people during past time slices does not provide any new information about which other zones might have been occupied given that we already know the decision type. The decision type alone can provide a great deal of information about what the evidence factor values would probably be, but if we then learn the value of one of the evidence factors, this should tell us nothing new about the values of the other evidence factors. If, for example, occupancy of one zone always meant that another zone had to be unoccupied, the corresponding evidence factors would

not be independent. Since knowing either of these evidence factors would indicate the value of the other factor exactly, the only way that one could claim independence of the factors would be if this information could be determined entirely from the decision class alone.

There are several reasons why the evidence factors used in my system will be reasonably independent. First, the total number of people in a building is not fixed. This means that at any time a zone can become occupied or unoccupied by a new arrival or departure, with no dependence on what is happening in the other zones. Second, it is possible for one or more people to occupy an arbitrary number of zones during a single time slice of the day. Many people can occupy a single zone during a time slice, or one person can occupy several zones during a time slice if he moves between zones. This clearly implies that the occupancy status of any given zone should tell us very little about the occupancy status of the other zones regardless of the total number of occupants in the building. Although these points do not guarantee independence, they do indicate that the assumption is an acceptable approximation.

Now that we have defined the evidence factors $\phi_{i,l}$, we may establish a method for making a maximum likelihood choice between decision types F_1 and F_2.

First we define the following:

$$\begin{aligned} \mathrm{p}'_{j,k} &= \mathrm{P}(F_j \mid k); \\ \mathrm{p}_{i,l,j} &= \mathrm{P}(\phi_{i,l} = 1 \mid F_j); \\ \mathrm{q}_{i,l,j} &= \mathrm{P}(\phi_{i,l} = 0 \mid F_j) = 1 - \mathrm{p}_{i,l,j} \end{aligned}$$

Where $\mathrm{P}(F_j \mid k)$ is the a priori probability that F_j is the correct decision type at a given time of the day denoted by k. For the given time slice k, and a corresponding set of evidence factors, we can decide whether the temperature should be set up or down depending on which j maximizes the expression:

$$\mathrm{p}'_{j,k} \prod_{\phi_{i,l}=1} \mathrm{p}_{i,l,j} \prod_{\phi_{i,l}=0} \mathrm{q}_{i,l,j}$$

If $j = 1$ maximizes the expression, then decision type F_1 is chosen and the temperature setting should be turned up. If $j = 2$ maximizes the expression, then decision type F_2 is

chosen and the temperature setting should be turned down. This choice is still not final since there will be a cost factor which may modify it.

This maximum likelihood decision algorithm can be understood in terms of a choice based on the probabilities for F_1 and F_2 given the state of all the pattern features, and the time slice k. We may formulate these probabilities as follows:

$$\mathrm{P}(F_j \mid \Phi \text{ and } k) = \frac{\mathrm{P}(F_j \mid k)\ \mathrm{P}(\Phi \mid F_j)}{\mathrm{P}(\Phi \mid F_1) + \mathrm{P}(\Phi \mid F_2)}$$

where Φ is the state vector for all $\phi_{i,l}$. Thus, we have:

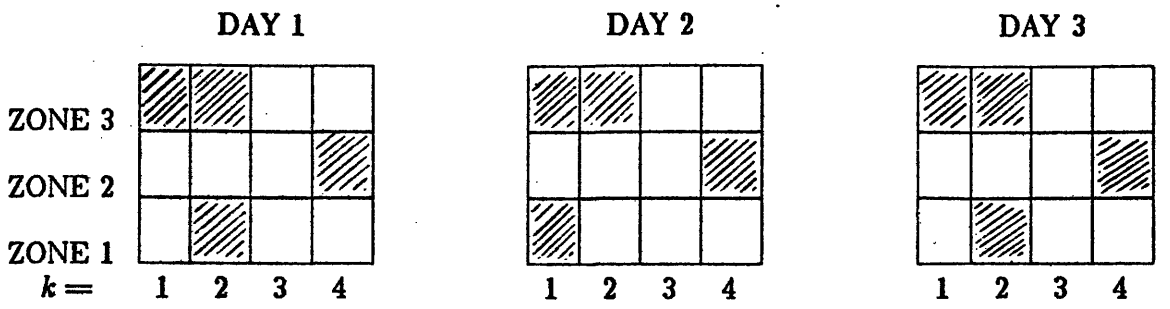
$$\mathrm{P}(\Phi \mid F_j) = \mathrm{P}(\phi_{1,1} \bigwedge \phi_{1,2} \bigwedge \phi_{1,3} \bigwedge \phi_{2,1} \bigwedge \phi_{2,2} \bigwedge \quad \cdots \mid F_j)$$

Now, if the $\phi_{i,l}$ components are all independent, then we may simplify this expression to

$$\mathrm{P}(\Phi \mid F_j) = \mathrm{P}(\phi_{1,1} \mid F_j)\ \mathrm{P}(\phi_{1,2} \mid F_j)\ \mathrm{P}(\phi_{1,3} \mid F_j)\ \mathrm{P}(\phi_{2,1} \mid F_j)\ \mathrm{P}(\phi_{2,2} \mid F_j) \qquad \cdots$$

In order to compare $\mathrm{P}(F_1 \mid \Phi \text{ and } k)$ with $\mathrm{P}(F_2 \mid \Phi \text{ and } k)$ we may compute them both explicitly from the formulas above (only one of these terms actually needs to be computed since F_1 and F_2 are mutually exclusive and collectively exhaustive), or we may simply compare the numerator $\mathrm{P}(F_1 \mid k)\mathrm{P}(\Phi \mid F_1)$ with the numerator $\mathrm{P}(F_2 \mid k)\mathrm{P}(\Phi \mid F_2)$ to see which is larger. The maximum likelihood decision method is equivalent to this latter comparison, using the independence assumption to simplify the $\mathrm{P}(\Phi \mid F_j)$ terms. This method is computationally efficient since it allows a direct comparison between the sum of the logarithms of the $\mathrm{P}(\phi_{i,l} \mid F_1)$ terms with that of the $\mathrm{P}(\phi_{i,l} \mid F_2)$ terms.

As a simple example of the decision mechanism, we may examine a hypothetical home with only 3 zones and with the day quantized into six hour time slices. The actual algorithm uses much smaller time slices, but the principles of this example still apply. A typical occupancy history taken over three days might appear as follows:



The shaded regions represent time slices during which the corresponding zones are occupied for more than one hour. The first time slice covers the time period between midnight and 6 AM, the second goes from 6 AM to noon, the third from noon to 6 PM, and the fourth time slice goes from 6 PM to midnight. In the patterns shown, we might consider zone 3 to be the bedroom, zone 2 to be the living room, and zone 1 to be the kitchen. Other rooms such as nearby bathrooms and hallways may be associated with each zone as well. The first of these patterns would then correspond to a day in which someone sleeps sometime between midnight and noon, spends at least an hour in the kitchen between the hours of 6 AM and noon, goes out in the afternoon, and comes back sometime after 6 PM and stays in the living room until about midnight. Note how more than one zone has been counted as occupied during a single time slice since a person has apparently spent at least an hour in several zones during the six hour period. This would be seen if shorter time slices had been used as well.

Suppose we were controlling the temperature for zone 2. Suppose also, for explanation purposes, that the time constant for heating zone 2 is six hours. Now assume that we are just starting day 2. Since our evidence factors $\phi_{i,l}$ depend on the past 24 hours of occupancy, we must use data from day 1 to make our choice between F_1 and F_2. The matrix of evidence factors will be as follows:

l \ i	4	3	2	1
3	1	1	0	0
2	0	0	0	1
1	0	1	0	0

$\phi_{i,l}$ for time slice $k = 1$ of day 2.

Six hours later, we reach the beginning of time slice $k = 2$ of day 2, and again we must decide whether the temperature should be turned up or down. This time though, the past 24 hours consist of the first time slice of day 2 and the last three time slices of day 1. This gives us a new evidence factor matrix as shown:

l \ i	4	3	2	1
3	1	0	0	1
2	0	0	1	0
1	1	0	0	1

$\phi_{i,l}$ for time slice $k = 2$ of day 2.

Note that we are essentially shifting a fixed length feature grid over a pattern of indefinite length. The evidence factor matrices for time slices 3 and 4 of day 2 will be as follows:

l \ i	4	3	2	1
3	0	0	1	1
2	0	1	0	0
1	0	0	1	0

$\phi_{i,l}$ for time slice $k = 3$ of day 2.

l \ i	4	3	2	1
3	0	1	1	0
2	1	0	0	0
1	0	1	0	0

$\phi_{i,l}$ for time slice $k = 4$ of day 2.

Assuming that we already know the values of $p'_{j,k}$, $p_{i,l,j}$, and $q_{i,l,j}$, we can see how a choice between F_1 and F_2 can be made at the beginning of each time slice of day 2 using the above evidence factor matrices and the maximum likelihood decision rule. The $p'_{j,k}$ term in the decision rule represents the a priori probability that the temperature in a given zone should be turned up or down during time slice k. This term completely ignores behavior characteristics of the recent past, and is intended to represent people's tendencies to consistently be in the given zone at certain times of the day. On the other hand, the $p_{i,l,j}$ and $q_{i,l,j}$ terms are based entirely on the recent past. These terms are intended to help identify behavior patterns that

are only weakly related to the exact time of day, but still have a great deal of regularity in terms of zone transitions and duration. If, for example, a person tends to come home from work in the evenings anywhere from 6 PM to 8PM, but once home he has a consistent pattern of going first into the bedroom and then into the kitchen, then the important criterion for heating the kitchen will be that the bedroom is occupied rather than the precise time of day.

3. ESTIMATION OF PARAMETERS

Now that a prediction technique has been established, we face the problem of determining the values of the parameters $p'_{j,k}$, $p_{i,l,j}$, and $q_{i,l,j}$ such that the predictor will be effective. First we will discuss how these parameters are evaluated for a single day, then, a method for classification of days which will allow the merger of data from similar days will be discussed.

From a Bayesian point of view, it is useful to treat the unknown parameters $p'_{j,k}$, $p_{i,l,j}$, and $q_{i,l,j}$ as random variables [Drake,67] [Lindley,72]. Then, with the proper data from a given day and the day preceding it, we can estimate both the probability density functions and the expected values for these random variables. The expected values of the parameters may then be used directly by the decision algorithm on future days of the same class. Because we are only interested in some general properties of these random variables, we want to represent them with probability density functions which are easily parameterizable from the available data. A good choice for this is the Beta PDF defined as:

$$B_p(P_0 \mid m_0, n_0) = C(m_0, n_0) P_0^{m_0-1}(1-P_0)^{n_0-m_0-1} \qquad \begin{cases} m_0 > 0 \\ n_0 > m_0 \\ 0 \le P_0 \le 1 \end{cases}$$

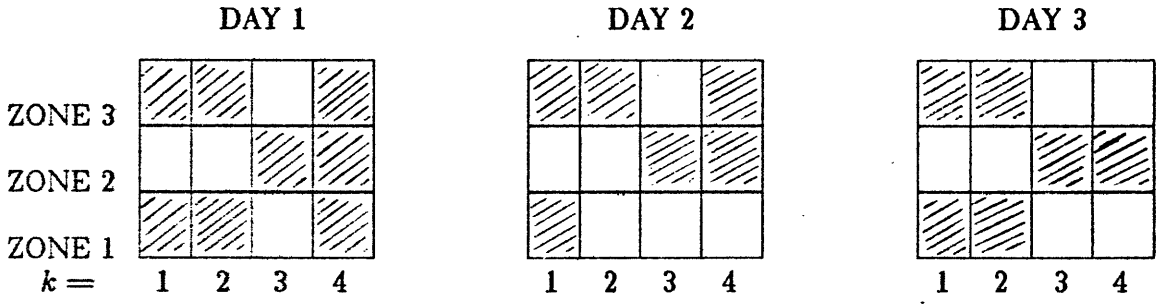
$$\text{where:} \quad C(m_0, n_0) = \left[\int_0^1 P_0^{m_0-1}(1-P_0)^{n_0-m_0-1} \, dP_0 \right]^{-1}$$

This PDF is selected because it conveniently has an expected value of P_0 equal to m_0/n_0, and because it is useful in a method for classifying days.

Next, we must concern ourselves with how the raw data from a given day will be processed to obtain the desired parameters. As discussed earlier, the raw data will consist of a simple quantized occupancy history from every zone. The processing for each zone is independent of the others except that this raw data must be shared between them.

The first processing that is performed on the raw data for each zone is to decide which decision type F_1 or F_2 should have been chosen at the beginning of each time slice for the

day. The decision type is definitely F_1 for time slices when the zone was occupied, but it is also F_1 for time slices that precede an occupation of a zone by less than the time required to warm up the zone. All other time slices belong to the decision type F_2. Using the occupancy patterns from the previous discussion, the time slices when the temperature should have been turned up either because of actual occupancy or expected occupancy may be seen below.



Here, the shaded regions represent times when the temperature should have been turned up. Note that because of the exaggerated six hour heating time constant that we are using in this example, each zone that was occupied during a particular time slice had to be heated during the time slice directly preceding the occupied period. In reality, the heating time constants are much shorter, but so are the time slices. Hence, the actual effect of the time constant is very similar to what is shown, although on a much shorter time scale.

From these heating patterns alone, we may estimate $p'_{j,k} = P(F_j \mid k)$. To do this, we begin with an initial estimate of each $p'_{j,k}$ at 0.5 using $m_0 = 1$ and $n_0 = 2$ in the Beta PDF. In this way, we start with a value of $p'_{j,k}$ that gives us the minimum information about the behavior patterns, and a Beta PDF that gives us the minimum information about $p'_{j,k}$. We then update each $p'_{j,k}$ for each time slice k by setting each n_0 to 3, and incrementing m_0 to 2 for each $p'_{1,k}$ whose time slice k is decision type F_1, and each $p'_{2,k}$ whose time slice k is decision type F_2. Note that $p'_{2,k} = 1 - p'_{1,k}$ for each k, so we can actually get by with only keeping track of $p'_{1,k}$.

Let us proceed with an example of how the $p'_{j,k}$ terms are to be estimated given the above heating patterns. For this example, we will start with day 2, assuming that we have no prior

knowledge about what the patterns should be like. First, we shall examine the $p'_{j,k}$ terms for zone 2. Initially, as stated above, all $p'_{j,k}$ terms will start out at a value of $\frac{1}{2}$. The first time slice of the second day is decision type F_2 for zone 2 since there is no need for the temperature to be turned up in that zone during that period. Thus, the parameter $p'_{1,1}$ will be updated to the value $\frac{1}{3}$ with the denominator incremented once for one trial, and the numerator left unchanged because of a failure to find decision type F_1 on that trial. Similarly, the parameter $p'_{1,2}$ will also be updated to $\frac{1}{3}$. The parameters $p'_{1,3}$ and $p'_{1,4}$ will both be updated to the value $\frac{2}{3}$, with the denominator incremented once for one trial, and the numerator incremented once because the decision type was F_1. On the following day, the same procedure could be used to give a value of $\frac{1}{4}$ to parameters $p'_{1,1}$ and $p'_{1,2}$, while giving a value of $\frac{3}{4}$ to parameters $p'_{1,3}$ and $p'_{1,4}$. Note how the effect of the initial bias diminishes as more information is obtained.

The parameters $p_{i,l,j}$, and $q_{i,l,j}$ are estimated only over time slices corresponding to their associated decision types j rather than over all time slices. As we have seen earlier, there is a matrix of evidence factors $\phi_{i,l}$ for each time slice of the day. Therefore, for the two sets of time slices consisting of F_1 type time slices and F_2 type time slices, there are two sets of $\phi_{i,l}$ matrices. The $p_{i,l,1}$ values are estimated from the set of $\phi_{i,l}$ matrices associated with the F_1 type time slices, and the $p_{i,l,2}$ values are estimated from the set of $\phi_{i,l}$ matrices associated with the F_2 type time slices. The estimation is performed as follows:

> For any given $p_{i,l,j}$, start with an initial estimate of its Beta PDF using parameters $m_0 = 1$, and $n_0 = 2$. As with the variable $p'_{j,k}$, these parameters represent the minimum possible information state. Now, out of the U samples for the evidence factors $\phi_{i,l}$ that are associated with decision type F_j, only V of these will be ones. We may consider these V ones as V successes out of U trials when dealing with $p_{i,l,j}$, and therefore increment n_0 by U and m_0 by V. This will give a new expected value for $p_{i,l,j}$ of : $\frac{V+1}{U+2}$. Each $p_{i,l,j}$ term is evaluated in this manner to obtain a $p_{i,l,1}$ matrix and a $p_{i,l,2}$ matrix. It is

not necessary to determine the $q_{i,l,j}$ values since $q_{i,l,j} = 1 - p_{i,l,j}$.

Continuing the previous example, we can consider how the $p_{i,l,j}$ parameters will be affected by the given occupancy patterns. We shall first consider the $p_{i,l,j}$ parameters pertaining to the pattern for zone 2 on the second day. The first two time slices in zone 2 of the second day are of decision type F_2 since the temperature should have been set back during these periods. Therefore, the parameters $p_{i,l,1}$ will be unaffected by the information from the first two time slices, and only the $p_{i,l,2}$ parameters will be changed. If we look back at the values of $\phi_{i,l}$ for time slice $k = 1$ of day 2, we find that only $\phi_{1,2}, \phi_{3,1}, \phi_{3,3}$, and$\phi_{4,3}$ are equal to 1, and the rest are equal to zero. As a result of the first time slice then, the parameters $p_{1,2,2}$, $p_{3,1,2}$, $p_{3,3,2}$, and $p_{4,3,2}$ will each have their denominators and numerators incremented by 1, while the other $p_{i,l,2}$ parameters will have only their denominators incremented. Thus, the parameters that correspond to features that showed positive occupancy will all become $\frac{2}{3}$, and the parameters corresponding to features that showed negative occupancy will all become $\frac{1}{3}$. This is of course assuming that they all started at the initial lowest information state of $\frac{1}{2}$. These changes can be most clearly understood from the parameter matrices shown below:

$p_{i,l,1}$ parameters

l \ i	4	3	2	1
3	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

$p_{i,l,2}$ parameters

l \ i	4	3	2	1
3	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$
1	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

On time slice 2 of the second day, the decision type in zone 2 is again F_2, so again only the $p_{i,l,2}$ parameters will be changed. Now, however, the feature array has been shifted in time, and so the parameter changes will be different. This time, the parameters $p_{1,1,2}$, $p_{1,3,2}$, $p_{2,2,2}$, $p_{4,1,2}$, and $p_{4,3,2}$ will have both their numerators and denominators incremented by 1, and the remaining parameters will only have their denominators incremented. The resulting parameters are shown below.

$p_{i,l,1}$ parameters

l \ i	4	3	2	1
3	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

$p_{i,l,2}$ parameters

l \ i	4	3	2	1
3	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$
2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{2}{4}$
1	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$

On time slice 3 of the second day, the decision type in zone 2 is F_1, so now the $p_{i,l,1}$ parameters will be updated, and the $p_{i,l,2}$ will be left as they are. The parameters $p_{1,3,1}$, $p_{2,1,1}$, $p_{2,3,1}$, $p_{3,2,1}$ will be incremented by 1 in both the numerator and the denominator, and the others will only be incremented in the denominator. The new matrices from this step are shown below.

$p_{i,l,1}$ parameters

l \ i	4	3	2	1
3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$
2	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$

$p_{i,l,2}$ parameters

l \ i	4	3	2	1
3	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$
2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{2}{4}$
1	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$

On the last time slice of the second day, the decision type in zone 2 is again F_1, and the $p_{i,l,1}$ parameters will be updated once more. The resulting parameter arrays from these final updates are shown below.

$p_{i,l,1}$ parameters

l \ i	4	3	2	1
3	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{2}{4}$
2	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
1	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$

$p_{i,l,2}$ parameters

l \ i	4	3	2	1
3	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$
2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{2}{4}$
1	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{2}{4}$

These are the final $p_{i,l,j}$ parameter arrays for the second day. These two matrices in conjunction with a matrix of the $p'_{j,k}$ terms calculated earlier form the complete set of probability parameters that define a class for zone 2. Similar parameter arrays will be obtained for the other two zones using the same procedure. Each zone develops its own group of

classes, and the groups remain completely independent of one another in all respects. The complete definition of a class will consist of these parameter arrays, some terms relating to the weight factor, some error statistics and bounds used for classification purposes, and a set of confidence factors for the voting system which is used for selecting the class on a new day.

Now that we have generated the parameter arrays for first class for zone 2, we can see how this information will be used in the maximum likelihood decision algorithm to predict what the correct heating pattern should be on day 3. Normally, there will be a variety of classes to choose from at the beginning of a new day, and it will be the duty of the class prediction algorithm to select the correct class to use. For the purpose of this example, however, we will assume that the parameter array set we have just derived forms the only existing class for zone 2. To make a prediction of whether the temperature setting should be turned up or turned down in zone 2 for the duration of the first time slice of day 3, we compare the $p_{i,l,1}$ parameters with the $p_{i,l,2}$ parameters when they are applied to the occupancy pattern of day 2. We may begin by evaluating the expression:

$$p'_{1,1} \prod_{\phi_{i,l}=1} p_{i,l,1} \prod_{\phi_{i,l}=0} q_{i,l,1}$$

In this case, only $\phi_{1,2}$, $\phi_{3,3}$, $\phi_{4,1}$, and $\phi_{4,3}$ are equal to 1, and the remaining occupancy features are equal to zero. This means that the above expression may be transformed into the following:

$$\begin{aligned} & p'_{1,1} \times (p_{1,2,1} \times p_{4,1,1} \times p_{3,3,1} \times p_{4,3,1}) \\ & \quad \times (1 - p_{1,1,1}) \times (1 - p_{1,3,1}) \times (1 - p_{2,1,1}) \times (1 - p_{2,2,1}) \\ & \quad \times (1 - p_{2,3,1}) \times (1 - p_{3,2,1}) \times (1 - p_{3,1,1}) \times (1 - p_{4,2,1}) \end{aligned}$$

In this expression, each $q_{i,l,1}$ term has been replaced by its equivalent $(1 - p_{i,l,1})$ form. Now, by simply plugging in the proper value for $p'_{1,1}$, and the appropriate values from the $p_{i,l,1}$ parameter array, we obtain the following score in favor of turning the temperature up:

$$\begin{aligned} & \tfrac{1}{3} \times (\tfrac{1}{4} \times \tfrac{1}{4} \times \tfrac{2}{4} \times \tfrac{1}{4}) \\ & \quad \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \\ & \quad \times (1 - \tfrac{3}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) = 1.1 \times 10^{-5} \end{aligned}$$

We must now compare this result with the value we obtain from the expression

$$p'_{2,1} \prod_{\phi_{i,l}=1} p_{i,l,2} \prod_{\phi_{i,l}=0} q_{i,l,2}$$

For this computation, we have the same feature matrix as before, but we are using the $p_{i,l,2}$ parameter matrix instead of the $p_{i,l,1}$ parameter matrix and we use $p'_{2,1}$ instead of $p'_{1,1}$. Recalling that $p'_{2,k} = 1 - p'_{1,k}$ for all time slices k, the score in favor of turning the temperature down is:

$$\begin{aligned}(1-\tfrac{1}{3}) \times (\tfrac{2}{4} \times \tfrac{2}{4} \times \tfrac{2}{4} \times \tfrac{3}{4}) \\ \times (1-\tfrac{2}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{2}{4}) \\ \times (1-\tfrac{1}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) = 1.2 \times 10^{-3}\end{aligned}$$

Since the score in favor of turning the temperature down is higher than the score in favor of turning it up, the decision will be to turn down the temperature setting for the first time slice of day 3.

Once the first time slice of day 3 is complete, it will be necessary to decide how to set the temperature for the second time slice. At this time, the occupancy features $\phi_{1,3}$, $\phi_{2,2}$, and $\phi_{4,3}$ will be equal to 1, and the remaining features will be equal to zero. The expression for the score in favor of turning the setting up will therefore be:

$$\begin{aligned}p'_{1,2} \times (p_{1,3,1} \times p_{2,2,1} \times p_{4,3,1}) \\ \times (1 - p_{1,1,1}) \times (1 - p_{1,2,1}) \times (1 - p_{2,1,1}) \times (1 - p_{2,3,1}) \\ \times (1 - p_{3,1,1}) \times (1 - p_{3,2,1}) \times (1 - p_{3,3,1}) \times (1 - p_{4,1,1}) \times (1 - p_{4,2,1})\end{aligned}$$

By plugging in the appropriate numbers, we obtain:

$$\begin{aligned}\tfrac{1}{3} \times (\tfrac{2}{4} \times \tfrac{1}{4} \times \tfrac{1}{4}) \\ \times (1-\tfrac{1}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{3}{4}) \\ \times (1-\tfrac{2}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{2}{4}) = 3.4 \times 10^{-5}\end{aligned}$$

When we plug in the corresponding values from the $p_{i,l,2}$ parameter array, we obtain:

$$\begin{aligned}(1-\tfrac{1}{3}) \times (\tfrac{2}{4} \times \tfrac{2}{4} \times \tfrac{3}{4}) \\ \times (1-\tfrac{2}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{1}{4}) \\ \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{2}{4}) \times (1-\tfrac{1}{4}) = 1.2 \times 10^{-3}\end{aligned}$$

Again, the score in favor of turning the temperature down is larger than the score in favor of turning it up, so the temperature will remain turned down during the second time slice of day 3.

When predicting the proper temperature setting for the third time slice, the occupancy features $\phi_{1,1}$, $\phi_{1,3}$, $\phi_{2,3}$, and $\phi_{3,2}$ will be equal to 1, and the remaining features will be equal to zero. The expression for the score in favor of turning the setting up will therefore be:

$$\begin{aligned} & p'_{1,3} \times (p_{1,1,1} \times p_{1,3,1} \times p_{2,3,1} \times p_{3,2,1}) \\ & \quad \times (1 - p_{1,2,1}) \times (1 - p_{2,1,1}) \times (1 - p_{2,2,1}) \times (1 - p_{3,1,1}) \\ & \quad \times (1 - p_{3,3,1}) \times (1 - p_{4,1,1}) \times (1 - p_{4,2,1}) \times (1 - p_{4,3,1}) \end{aligned}$$

By plugging in the appropriate numbers, we obtain:

$$\begin{aligned} & \tfrac{2}{3} \times (\tfrac{1}{4} \times \tfrac{2}{4} \times \tfrac{3}{4} \times \tfrac{2}{4}) \\ & \quad \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) = 6.2 \times 10^{-4} \end{aligned}$$

When we plug in the corresponding values from the $p_{i,l,2}$ parameter array, we obtain:

$$\begin{aligned} & (1 - \tfrac{2}{3}) \times (\tfrac{2}{4} \times \tfrac{2}{4} \times \tfrac{1}{4} \times \tfrac{1}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{3}{4}) = 2.3 \times 10^{-5} \end{aligned}$$

This time, the decision will be to turn up the temperature setting since the score in favor of this decision is greater than the score opposing it.

At the beginning of the last time slice of the third day, the occupancy features $\phi_{2,1}$, $\phi_{2,3}$, $\phi_{3,3}$, and $\phi_{4,2}$ will be equal to 1, and the remaining features will be equal to zero. The expression for the score in favor of turning the setting up will therefore be:

$$\begin{aligned} & p'_{1,4} \times (p_{2,1,1} \times p_{2,3,1} \times p_{3,3,1} \times p_{4,2,1}) \\ & \quad \times (1 - p_{1,1,1}) \times (1 - p_{1,2,1}) \times (1 - p_{1,3,1}) \times (1 - p_{2,2,1}) \\ & \quad \times (1 - p_{3,1,1}) \times (1 - p_{3,2,1}) \times (1 - p_{4,1,1}) \times (1 - p_{4,3,1}) \end{aligned}$$

By plugging in the appropriate numbers, we obtain:

$$\begin{aligned} & \tfrac{2}{3} \times (\tfrac{2}{4} \times \tfrac{3}{4} \times \tfrac{2}{4} \times \tfrac{2}{4}) \\ & \quad \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{1}{4}) = 1.9 \times 10^{-3} \end{aligned}$$

When we plug in the corresponding values from the $p_{i,l,2}$ parameter array, we obtain:

$$\begin{aligned} & (1 - \tfrac{2}{3}) \times (\tfrac{1}{4} \times \tfrac{1}{4} \times \tfrac{2}{4} \times \tfrac{1}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{2}{4}) \\ & \quad \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{1}{4}) \times (1 - \tfrac{2}{4}) \times (1 - \tfrac{3}{4}) = 7.6 \times 10^{-6} \end{aligned}$$

Once more, the decision will be to keep the temperature setting turned up, since this is clearly favored in the likelihood analysis.

As we see from this example, the maximum likelihood estimation did a perfect job of predicting when the temperature had to be turned up, and when it had to be turned down. This is primarily due to the fact that the patterns for the different days were so similar to one another. With wider variations in the patterns, a single set of parameter arrays cannot be expected to deal with all the possible interrelationships. This is why there is a need for multiple classes.

For the sake of accuracy, it is important to point out some of the differences between this example and the actual system I have implemented. First, one important difference is the use of a weight factor to bias the decision. This factor is is a dynamically variable parameter which is multiplied with the $p_{i,l,1}$ parameters to bias the final decision one way or the other. The purpose of this factor is to balance the errors that the system makes according to the user's preferences for comfort and conserving energy. If the factor is greater than 1, we can see that a greater number of close decisions will be biased toward keeping the temperature turned up. If the factor is less than 1, these close decisions will be biased toward keeping the temperature turned down. The details of this cost factor will be discussed more thoroughly later.

Another important difference between the example above and the actual system is that in my system I divide the day into four distinct time segments. The segments are then subdivided into time slices like those in the example. Each segment has its own distinct set of classes which are independent from the classes used in other segments just as they are independent from the classes used for other zones. The reason the day needs to be divided into segments is that a single behavior sequence may have entirely different meanings at different times of the day. For example, when someone goes from the bedroom to the kitchen in the morning, they are likely to be having breakfast, and will probably be leaving for work soon

thereafter. On the other hand, if that same person goes from the bedroom to the kitchen in the evening, it is doubtful that they will be leaving the house afterwards. Thus, by having sets of classes that are only relevant at certain intervals during the day, the system has a better chance of extracting consistent patterns of movement between zones. By having four segments, there can be separate classes for the early morning hours between midnight and 6 AM, awakening hours between 6 AM and noon, working hours between noon and 6 PM, and evening hours between 6 PM and midnight. The only disadvantage of having the day segmented is that the class prediction system must make a selection four times a day for each zone rather than only once a day for each zone. This increases the chances that sometime during the day, an incorrect class will be chosen. This problem is dealt with by attempting to recognize an incorrect choice as quickly as possible, and by adjusting the weight factor of the decision function to minimize the potential loss due to an incorrect class selection.

4. THE COST FACTOR

Since we can never expect people's behavior to be perfectly regular, we can never expect a system which uses regularity as a basis for predictions to be perfectly accurate. There are two types of mistakes that my system can make — those which cause users to be physically uncomfortable because they move into zones where they are not expected, and those which cause energy to be wasted because people fail to show up when they are expected. It is important that the user be capable of influencing the balance between these two types of errors according to his personal priorities on comfort and saving energy. It is also very important that the user be capable of exercising his influence in a very simple and understandable manner.

To deal with this problem, my system allows the user to directly input what I call a relative cost factor. The relative cost factor is a parameter which represents the relationship between the user's subjective cost of physical discomfort and the monetary cost of wasted energy. This parameter can be translated directly into the preferred ratio of the two different types of errors. A relative cost of 2 to 1 states that physical discomfort is twice as expensive subjectively as wasted energy. This may be translated into the statement that two errors on the side of wasting energy may be equivalent to only one error on the side of user discomfort. It is intended for the system to bias its decision process such that whatever the average number of errors per day may be, the proportion of waste errors to discomfort errors will be maintained at the ratio specified by the relative cost factor. This way, if the relative cost is at the comfort extreme, the user will obtain performance nearly equivalent to a standard thermostat. Only in situations where the system is absolutely sure that a zone will be unoccupied will it consider turning the temperature setting down. On the other hand, if the relative cost is at the energy conservation extreme, it will function similar to a system which relies solely on people sensors to decide where to keep the temperature turned up. This will cause some discomfort when people first enter unoccupied zones, but it will maintain comfort

once zones are occupied. A zone which has people coming in and out of it need not cause any discomfort since there can be a delay between the time someone leaves a zone and the time the temperature is actually set back.

The most appropriate way to bias the decision function in favor of one type of error over another is by using a weight factor in the maximum likelihood algorithm. This factor is used to scale one side of the decision formula, thereby biasing the decision for or against that side depending on whether the factor is greater or less than one. The new form of the decision function becomes:

IF: $$\text{weight factor} \times p'_{1,k} \prod_{\phi_{i,l}=1} p_{i,l,1} \prod_{\phi_{i,l}=0} q_{i,l,1} \geq p'_{2,k} \prod_{\phi_{i,l}=1} p_{i,l,2} \prod_{\phi_{i,l}=0} q_{i,l,2}$$

THEN: Turn the temperature setting to the comfort level.

OTHERWISE: Turn the setting to the conservation level.

This weight factor is not the same as the relative cost factor since its value says nothing about the ratio at which the two types of errors will occur. The error ratio depends upon all the parameters in the class representation, and how they interact with the typical day to day irregularities in behavior. With the appropriate feedback, this weight factor can be automatically adjusted to a level which yields the desired average error ratio.

The feedback applied to the weight factor is obtained by comparing the relative cost ratio with the actual ratio of errors. When the two different types of errors are scaled by the relative cost ratio, the difference between the scaled values is taken to obtain a positive or negative change score. If this score is positive, then the weight factor should be increased, and if it is negative, then the weight factor should be decreased. To determine how much the weight factor should be changed, the change score is multiplied by a gain factor.

The gain factor is needed because it is important that the weight factor reach an appropriate level as quickly as possible. The gain factor serves to multiply the small change score to a level where it will make a significant change in the weight factor. However, the significance of a change in the weight factor depends on how close the weight factor is to its optimal value. Therefore, the gain factor cannot simply be a preset constant, it must vary

dynamically, increasing when the weight factor is far from a correct setting, and decreasing when the weight factor appears to be close to an optimal setting.

One way to determine whether or not the weight factor is near an optimal setting is to observe oscillations in the change score over consecutive days. If a given setting of the weight factor causes an excessive number of waste errors on one day, and then, after being changed to compensate for this error, it causes an excessive number of discomfort errors on the next day, then the weight factor was probably changed too much the first time. To avoid overcompensating for this excessive change by another excessive change, the gain factor should be reduced before the next change is made. On the other hand, if the ratio of waste errors to comfort errors is consistently larger or smaller than the relative cost ratio, then the weight factor is not being changed fast enough. In cases like this, where the sign of the change score is the same on two consecutive days, the gain factor will be increased.

An important exception to these rules occurs when there are very few errors. In such cases, the errors may be small enough to indicate that the weight factor is very close to a desirable level. It is then best to set the change score to zero to insure that the weight factor is not altered. I have chosen to make this exception when there are no occurrences of the most costly error type, and the occurrences of the other error type are less than the proportion that is allowed with one occurrence of the most costly error type. Here, the most costly error type is specified by whether the relative cost is greater or less than one. If, for example, the relative cost is 2 to 1, then comfort errors are the most costly. The weight factor would remain unchanged in this case if there were no comfort errors, and no more than two waste errors. If, on the other hand, the relative cost is 1 to 3, then waste errors are the most costly. In this case, the weight factor would remain unchanged if there were no waste errors, and no more than three cold errors.

The relative cost seems sufficiently intuitive to allow it to be directly adjusted by the user. The user need not know anything about the actual value of the relative cost to be able

to simply turn a knob in one direction if he finds himself uncomfortable far too often, and in another direction if he is satisfied with the comfort but would like to save more energy. Special care must be taken, however, to insure that a change in the relative cost adjustment causes an immediate change in performance. Each time a user changes the relative cost setting, the proper weight factors will have to be recomputed immediately. This may easily be done if several weeks of past behavior patterns are saved in memory. Then, if the relative cost is changed, the past data can be reprocessed to give new values for the weight factors. This computation should only be done after a person is sure where they want the new setting to be. It would certainly not be appropriate to begin this lengthy computation at the first movement of the setting. Instead, there must be a delay of a few minutes between the time that the setting was last changed and the time that the recomputation begins.

If this recomputation is not done, the user will never be able to obtain satisfactory performance from the system. Long delays between the changing of a setting and the final attainment of the desired performance would be sure to cause a great deal of confusion and frustration. Users who were not satisfied with the level of comfort would turn the knob a little to favor comfort. Then, when they found themselves uncomfortable on the following day, they would turn the knob even further. After several days, the knob would probably be turned much farther in the comfort direction than was necessary, and the weight factor would just be reaching a level corresponding to the first adjustment. Another week later, and the weight factor will be biased toward making waste errors far more often than necessary. When the user realizes the degree of excessive waste, he will start turning the knob in the other direction. Again, the delay will probably cause him to turn it too far. As we can see, it is doubtful that this process would ever end, so there is no choice but to perform the recomputation.

With the relative cost factor implemented as described, the system will have a very powerful interface mechanism. All a user has to do is turn a dial, and he will be able to save

money. If he wants to save money, he knows that he will have to sacrifice some comfort to do so, and if he gets tired of walking into cold rooms, he knows he will have to spend a little more money to get them heated sooner. Thus, in a very short amount of time, the user should be able to find the setting which minimizes the energy costs and still provides an acceptable level of comfort.

5. CLASSIFICATION OF DAYS

Knowing only how to estimate the parameters $p'_{j,k}$ and $p_{i,l,j}$ for a particular day does not always allow us to obtain general pattern representations. Behavior patterns on different days can be tremendously different, and we cannot expect a single set of parameters to be able to perform well on them all. To solve this problem, a method is needed for separating data from days that are significantly different while at the same time, combining data from days that are similar so as to improve the estimates of their parameters. This leads to the classification of days.

When considering the classification of days, it is important to determine a useful basis for classification. I have taken a functional point of view in this respect. Since our main reason for classifying days is that different events in different days will require different decisions to be made, it makes sense to classify days according to the functions which make these decisions. In this case, all of the decision making knowledge for any given day is contained in the PDFs for $p'_{j,k}$ and $p_{i,l,j}$. Therefore, only days for which these PDFs are similar should be placed in the same class.

Using this approach, after a day has ended and the PDFs for $p'_{j,k}$ and $p_{i,l,j}$ have been computed, these parameters are compared with those of all currently established classes. Also at this time, each class is evaluated on how well it would have performed had it been used for prediction on that day. This performance can be compared with statistics from previous days to make sure it is reasonable. If no class can pass the acceptability test for the new day, then a new class must be generated consisting only of the data from the one day. If any class should pass the acceptability test, then the new day will be classified as the class whose probability parameters matched it best. This class will then be augmented with the data from that day. Note that this method allows the system to generate a new class on the first day it sees, and then generate additional classes only as they are needed.

Since memory space is not infinite, the system will eventually run out of space for new

classes. This problem can be solved by merging classes that are very similar, by eliminating old unused classes, or by forcing new days into existing classes even when the acceptability tests fail. I have chosen the latter of these options for dealing with memory limitations in my simulation. This may not be the best method to use, but definitely the simplest to implement, and since I rarely run into any memory problems on my simulations, this is what I use. I limit the maximum number of classes that any single room day-segment may have to four. This is a totally arbitrary limitation, but I felt it was best to limit this number to a reasonably small value because the class sequence prediction algorithm will work better with fewer classes to choose from. As it turned out in my final tests, there was rarely a need for even four classes for any of the room day-segments.

Classification of a day-segment for a given room is based on several criteria. First, there are two tests to see whether or not a new class has to be generated, then if no new class is to be generated the best classification has to be determined. The two tests which determine whether or not a new class needs to be generated are called the acceptability tests. These tests are intended to limit the growth of classes as much as possible, while still allowing extremely deviant events to have their own classifications. This goal is best achieved if we prohibit the establishment of a new class if any of the existing classes should happen to give good performance during the new day-segment or match well with the parameters of the new day-segment. Performance is measured by an error score, and similarity of parameters is measured by a distance score.

5.1 The Error Score

One heating time constant after a day-segment is complete, the hypothetical performance of all existing classes during that day-segment is easy to determine. I say "hypothetical" because it is not necessary for a class to have actually been used during the day segment for its performance score to be calculated. Instead, each existing class may be run on the data for the past day-segment as if it had actually been used to control the temperature during

that period. Each class will make a certain number of errors when it is run for the given day-segment. These errors will be of two types. One type is to have the temperature turned up when it should have been turned down, and the other type is to have the temperature turned down when it should have been turned up. We can label these two types of errors as "waste errors" and "cold errors" respectively, since the first type will cause energy to be wasted, and the second type will cause the user to be uncomfortably cold for a short time. It is important to note that a failure to turn up the temperature in the current day-segment in anticipation of occupancy in the following day-segment is an error of the latter type. Since an error of this type occurring on the last time slice of the current day-segment can only be detected one heating time constant after the following day-segment has begun, the cumulative errors for the current day-segment cannot be evaluated until this time.

The errors resulting for each class can be checked against certain bounds associated with each class to determine whether or not they are within acceptable limits. The bounds that are used to check validity are based on the past errors that a class has produced when it was actually used. Each time a class is used, the number of incidences of both types of errors are recorded. The data base allows the storage of the cumulative number of incidences of each type of error as well as the cumulative square of these incidences. Since the number of times each class has been used is also known, it is possible to determine the average number of errors that may be expected in a single day, as well as the standard deviation for this number. However, before this information can be used to check the acceptability of new error results, some additional processing must be performed.

It is not reasonable to simply compare the two different types of errors separately to determine the acceptability of a class on a new day segment. If, for instance, a particular class averaged 10 waste errors but only 1 cold error each time it was correctly used, it is not clear whether or not a new day will fit acceptably into that class if the class yields no waste errors and 2 cold errors. The solution to this problem is to combine the two types of errors

in such a way that their relative significance is taken into account. Fortunately, the relative cost factor, an indicator of the user's preference for comfort or money, is a direct measure of the relative significance of these errors. This factor then, provides a key to combining the errors into a single score.

An error score is obtained by adding the two types of errors together with one type scaled down according to the size of the relative cost factor. The choice of which type of error to scale down depends on whether the relative cost factor is greater or less than one. If the relative cost factor is greater than one, this means that errors which cause discomfort are more costly to the user than errors which waste energy. A relative cost factor which is less than one implies that errors which waste energy are more costly to the user than errors which cause discomfort. In terms of the defined error types, this would mean that waste errors are more costly than cold errors. Scaling of the errors is performed by reducing the value of the least significant error type by an amount proportional to the relative cost ratio. Thus, if the relative cost factor is 2.0, then cold errors have the most significance, so waste errors are scaled down by a factor of two. On the other hand, if the relative cost factor is 0.5, then waste errors are the most significant, and so cold errors are scaled down by a factor of two (1/0.5). Once the appropriate scaling has been performed, the errors may be added to obtain the final error score.

The upper bound for an error score obtained on a new day must be determined from statistical properties of past errors that have been accepted for a class. Sufficient information is stored each time the classification of a new day is established to allow the calculation of the average number of errors per day as well as the mean squared error per day for both types of errors. This facilitates the use of a Gaussian approximation for the errors, and the standard deviations and means for each type of error may be determined.

This approximation is very useful since it allows a simple acceptability test. If the error score for any class on a new day is less than one standard deviation from the mean error

score for that class, then the new day is suitable for assignment to one of the existing classes. This criterion requires special calculation of the standard deviation of the error score from the standard deviations of the cold and waste errors. This is done by scaling the variances of the two error types in the same way that the errors themselves are scaled. By doing this, we may combine the two variances to obtain the variance of the combined error score. Taking the square root of this variance will give the standard deviation of the error score. The upper bound is established by adding this standard deviation to the mean error score. One standard deviation from the mean is a somewhat arbitrary limit, but it is a reasonable one since it will tend to become more restrictive with time, and not less restrictive.

The size of the bound determines its restrictiveness. If a bound for a class is very small, then only when the class yields very few errors on a new day will the acceptability test be satisfied. If a bound for a class is large, then even when the class yields many errors on a new day, that day might still be a potential member of the given class. Whenever a new day is accepted into a particular class, the errors made by that class on that day are entered into the error statistics for the class. If the new errors are within the initial error bound, the bound should get smaller, or more restrictive.

Since a brand new class has not accumulated any errors, it is important to initialize the error bound for a new class at a reasonable level. If the proper level for the bound is not chosen, future days will be either accepted or rejected too readily by the new class. Since the bound is frequently decreased but rarely increased by errors from accepted new days, it is important for the initial bound to be the maximum that should be acceptable. I have based my choice of the bound on the principle that any class, when used at the proper time, should always perform better than a function which randomly chooses when to turn the temperature setting up and when to turn it down. By taking a percentage of the errors that such a function might generate, a reasonable bound may be established.

The number of errors that the random control function would generate is determined

from the occupancy pattern that first causes the new class to be generated. This pattern is used because the class will presumably be used on similar patterns in the future. As an example, suppose that a particular day-segment is 12 time slices long. Also assume that during four of these time slices the heat should have been turned up, and on the remaining eight time slices the heat should have been turned down. The random control function will on the average have the heat turned up half the time and turned down half the time. Therefore, during half the time when the heat should be turned up, it will be turned down, and during half the time when the heat should be turned down, it will be turned up. This means that on the average, a random control would have produced two cold errors from not having the heat turned up at the right time, and four waste errors for having the heat turned up when it wasn't needed. In my system, I take 80 percent of the average random errors to be the initial maximum bound. The maximum bounds for cold errors and waste errors are kept separate so that the maximum error score will change appropriately if the relative cost factor is changed.

5.2 The Distance Score

The error bound is not the only criterion for acceptance of a new day into any of the available classes, there is also a distance measure which is used both for determining acceptability, and for finding the best classification for a new day. The distance measure is a direct comparison of the probabilistic representation that is generated for a new day with the probabilistic representations of the existing classes. Since each representation consists of an array of parameters which each correspond to a specific probability density function, the representation may be characterized by a single point in a multi-dimensional space where there is a unique dimension for each parameter. When two representations are characterized in this manner, the geometric distance between them is easy to determine. As a very simple example, we may consider three representations which consist of only two parameters. Let class A have the parameters [1/4, 3/4], class B have the parameters [3/4, 1/2], and an unclassified new day have parameters [1/2, 1/4]. These points are represented in the two-dimensional graph

in Figure 1 below:

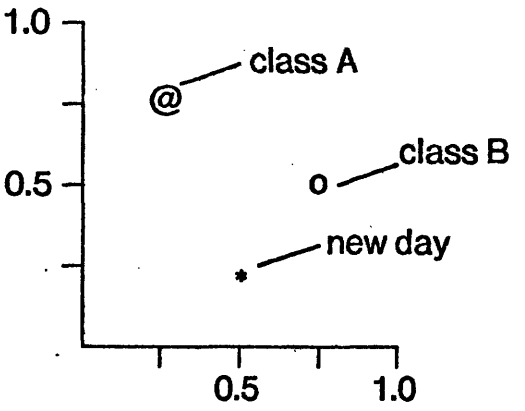


FIGURE 1: THE DISTANCE SCORE

The distance from the new day to class A would be $\sqrt{(\frac{1}{4}-\frac{1}{2})^2+(\frac{3}{4}-\frac{1}{4})^2} = 0.559$, and the distance from the new day to class B would be $\sqrt{(\frac{3}{4}-\frac{1}{2})^2+(\frac{1}{2}-\frac{1}{4})^2} = 0.354$. The parameters that are actually used for the distance measure are the $p'_{j,k}$, and $p_{i,l,j}$ terms in the pattern representation.

By keeping a maximum distance bound with each class, it is possible to determine whether data for a new day should be accepted into one of the existing classes or a new class should be generated. If the distance between the new day and each of the existing classes is beyond the maximum distance bounds corresponding to each class, then there will be reason for generating an new class. Otherwise, the best classification for the new day will be as the class which is closest to it. In the above example, provided that the new day lies within the distance bound of either class A or class B, it will be classified as class B. If the new day lies within the distance bound of class A, but not that of class B, it will still be classified as class B since it is closest to B. The reason for this is that if the new day is acceptable for class A, it must also be acceptable to any class which is closer to it than A. As a result of accepting the new day as part of class B, the distance bound for class B will be increased so as to insure that any future occurrence of a day exactly like the one that was just accepted will fall within the bound. This is to make sure that once a particular type of pattern is accepted into a given class, it will always be accepted by that class.

The two methods for determining whether a new day may be accepted into one of the

existing classes or whether a brand new class should be generated have now been described, but their interaction needs to be explained further. The basic rule I apply for deciding when to generate a new class is to only do so when the new day does not satisfy either of the two acceptability tests, and the total number of classes that have been generated is still below the preset limit. This way, if the new day is within the error bounds or the distance bounds of any of the existing classes, then it will be accepted. Once it has been decided that no new class should be generated, the error and distance bounds become irrelevant and the actual classification is based entirely on the relative distances between the new day and the existing classes. This approach is intended to minimize the development of new classes as much as possible while still providing a capability for generating those classes that are essential.

When the first class is first established, the error bound will be set at 80 percent of the average random error, and the distance bound will be set to zero. If the next day falls within the error bound, then the class will be augmented with the pattern from that day, and the distance bound will be increased to the distance between the class and the new pattern. The errors that were produced by the given class on that day will be added to the cumulative error statistics of the class. If the next day results in a pattern which causes the class to perform badly, then the error bound will not be satisfied. If at the same time, this pattern is not within the distance bound of the existing class, a new class will be established. This new class will be initialized in the same manner as the first class was. The error bound will of course be different depending on the actual pattern that caused the new class to be generated. On the following day, the pattern will first be checked to see if it passes either the error bound test or the distance bound test. If either test is passed for any of the established classes, then this will be sufficient to prevent a new class from being generated.

The next step is to determine the classification of the new day. The fact that the error test was passed only for the second class does not limit the selection to this class alone. Since a slight change in the weight factor for the first class might have made it perform much better

on the new pattern than the second class ever could, it is necessary to base the classification entirely on distance. Assuming that the new day is subsequently classified as the first class, the error statistics will be updated from the new day, and the maximum distance will be extended to reach the new point. The fact that data which did not pass the error test for the first class was still entered into the first class will cause the error bound to be increased. Since there is no control on how much this increase might be, it is important to always maintain the effective error bound at or below the initial level.

Once a given day has been classified, the selected class must be augmented with the data from that day. A class is defined by a set of Beta PDFs for the random variables $\mathbf{p}'_{j,k}$ and $\mathbf{p}_{i,l,j}$. Each PDF is represented simply by the two parameters M_c and N_c. We may recall that the Beta PDFs for random variables $\mathbf{p}'_{j,k}$ and $\mathbf{p}_{i,l,j}$ were also established from the data of the one day as well. Let these PDFs be characterized by the parameters M_d and N_d. All that is needed to augment the selected class then, is to subtract the bias that was initially inserted into M_d and N_d and add the resulting raw data to M_c and N_c. Thus:

$$M_c^{(p+1)} = M_c^{(p)} + (M_d - 1);$$
$$N_c^{(p+1)} = N_c^{(p)} + (N_d - 2).$$

It is clear that the process for augmenting classes cannot continue indefinitely without it consuming an infinite amount of memory. This problem can be avoided if we require that once M_c reaches a specified limit, we switch to a heuristic estimator of the form:

$$N_c^{(p+1)} = \mathrm{Int}\left[\frac{M_c^{(p)}}{M_c^{(p)} + (M_d - 1)}(N_c^{(p)} + (N_d - 2))\right];$$
$$M_c^{(p+1)} = M_c^{(p)}.$$

This estimator will keep M_c constant, and only alter N_c.

Using the classification and parameter augmentation procedures described here, several good things will happen. First, by grouping days which are similar but not identical into the same class, we will be able to isolate those features that are most relevant to the prediction problem, and ignore extraneous information. Also, class augmentation serves as a means for

compressing the pattern information from many days into a single representation, thereby reducing memory requirements. The ability to have several different representations is still very important, however, since human behavior is too diverse to be put into a single representation. This is what makes the classification algorithm so important, since it must make the choice of when to compact its representations and when to diversify them.

6. PREDICTION OF CLASSES

If more than one class exists for a given zone, then it will be necessary to predict which class to use at the beginning of a new day. A class which works well to predict a particular behavior pattern will be of little value if that class is not used when it is needed. Fortunately, there is certain information available to the system which should help in choosing the correct class. For instance, the system can know about the calendar, various special days, and all of the past classes that have been used. The designer of the system may also be able to provide the system with inexact knowledge of how these pieces of information might be related. The problem then boils down to a matter of using the inexact knowledge to interpret the given information, and thereby make a reasonable prediction of what to expect in the future.

This problem is of critical importance to work in artificial intelligence since many problems requiring intelligence involve the assimilation of incomplete data and uncertain knowledge for making decisions and predictions. In my approach, I have combined the virtues of two established techniques in a rather unique way. First, a knowledge base system similar to MYCIN is used to direct the processing of the available information according to inexact knowledge of how different pieces of information might be related [Shortliffe and Bunchman,75]. Second, a learning technique similar to that developed by Samuel in his learning checker player is used to help bias the influence of each knowledge source [Samuel,59]. This way, the initial knowledge of the system can be very general since the precise details may be learned from experience.

In a knowledge base system like MYCIN, the best of a given set of hypotheses is selected according to the significance of various pieces of evidence in favor or against each hypothesis. Evidence is obtained from inexact knowledge about the meaning of various elements of raw data, and the strength of this evidence is included as part of the knowledge. This type of system transforms directly into the problem of predicting future classes if we formulate a hypothesis for each candidate class stating that the associated class is the correct choice.

Inexact evidence is then obtained by using general knowledge about people's behavior patterns to analyze the record of their past behavior. The power of MYCIN to determine the validity of each hypothesis by combining inexact evidence thus provides a good framework for the class prediction problem.

The addition of learning to this system is essential because of the extreme diversity in different people's behavior. Even though a general base of knowledge can be established for understanding human behavior, the significance of any part of that knowledge for predicting behavior depends entirely on the individuals involved. Thus, the only way that such a general base of knowledge can be useful is if the system can learn which parts of its knowledge are most important merely by observing the behavior of the individuals who use it.

There are many forms of knowledge that might be considered useful for predicting future classes. We know that some forms of behavior may be fairly regular, repeating on a weekly or monthly basis, while others revolve around various special occasions. At other times, certain disruptions in normal behavior may foretell additional disruptions in the near future. Since classes are used to represent behavior patterns, these types of relationships between different behavior patterns should be reflected in similar relationships between the various classes. However, it is not possible to know in advance which classes will be associated in this manner. Therefore, the knowledge that the system contains can only be in a form which directs the system to look for such relationships between all classes and identify those classes which are consistently related.

To deal with the many possible ways that classes might be interrelated, the class prediction system uses independent experts that can vote on which classes are most likely and least likely to be needed. These experts each represent distinct pieces of knowledge since they analyze data and make predictions according to their own specific criteria. The experts begin with only general knowledge, but they are usually capable of gaining more knowledge with experience. A typical expert will initially know only to look for particular relationships

within the data, but with time, relationships will be found, and this knowledge will be used as a basis for making a vote. The vote can thus be considered as a piece of inexact evidence for the knowledge base system.

Each expert must consider each of the classes available for selection, but they need not vote on classes which are uncertain. If an expert does vote on a class, it may either vote for or against that class. There is no limit to the number of classes that a single expert may vote on. By allowing experts to abstain from voting, they need only vote on a class when the available evidence for or against that class is strong. This is important because it creates an implicit confidence threshold which requires that a degree of certainty be reached before a vote can be made.

When an expert gives a vote for or against a class, this vote must be weighted by a confidence factor just as inexact evidence is weighted for its reliability in MYCIN. In a standard knowledge base system, each confidence factor is known in advance and in fact, these factors represent a significant part of the system's knowledge. In my system, the confidence factors are all established dynamically. This is done by an external judge which evaluates the performance of each expert, and assigns appropriate confidence factors accordingly. This is the key to the learning ability of my system since it automatically finds the best experts to use for predicting the particular behavior patterns at hand.

The final class prediction is made by combining the confidence factors for each vote in the same manner that inexact evidence is combined in MYCIN. This method is designed to combine inexact evidence factors in such a way that positive evidence can only weigh in favor of a hypothesis however weak that evidence may be, and similarly, negative evidence can only weigh against a hypothesis. In my system, the set of hypotheses corresponds to the set of classes that might be used on the upcoming day segment. Once all of the evidence for each hypothesis has been combined, the best class can be chosen by selecting the one which has the most evidence in favor of it.

It is important to note that the concept used for combining evidence factors is very different from standard probabilistic methods. The approach I use corresponds directly to the idea of belief and disbelief factors used in MYCIN. In this method, votes in favor of a class will only increase our belief that the class is the best choice, and votes against a class will only decrease our belief that the class is the correct choice. This is intended in MYCIN to accommodate for the way that human experts think about the interaction of various pieces of evidence, but it also works well in the automatic procedures I have developed. The key advantage is that it allows the consideration of inexact and interdependent pieces evidence in a very straightforward manner. If a probabilistic approach were used instead, a vote in favor of a class with a strength of only 0.1 would end up being equivalent to a vote of strength 0.9 against the class. In addition, any interdependence of evidence factors would have to be dealt with explicitly. It is not reasonable to use an expert's vote in favor of a class as evidence against using that class simply because that expert is very unreliable, and it is not always possible to know in advance which evidence factors will be interdependent. Thus the concept of belief and disbelief factors provides considerable power and versatility to the system.

Since each hypothesis has its own set of evidence factors both for and against it, these factors must be combined in a manner which allows an overall comparison between the different classes. The method I use computes a single certainty score for each hypothesis and then chooses the hypothesis with the highest certainty score for its prediction. In the computation of the final certainty scores for the individual hypotheses, cumulative belief and disbelief factors are first computed separately, and then these factors are combined. The computation a cumulative belief factor is performed by taking all of the confidence factors of votes in favor the given hypothesis, and combining them with a function which is similar to a multi-valued form of a binary "or" function. The cumulative disbelief factors are computed in the same way, using the confidence factors of votes against the hypothesis rather than those in favor of it.

For convenience, we may call the function used to combine confidence factors a "numerical-or". The numerical-or of two numbers A and B which are both less than or equal to 1 would be as follows:

$$(\text{numerical-or} \quad A \quad B) = A + (1 - A) \times B$$

The interpretation of this function is that if A provides a fixed amount of confidence about a certain hypothesis, then B can at most only increase that confidence by the amount of uncertainty remaining after A is known. The actual increase will be a percentage of the remaining uncertainty which is proportional to the certainty that B provides by itself. Since the cumulative belief and disbelief factors for a hypothesis should not depend upon the order in which the evidence factors are combined, it is important that the "numerical-or" function be symmetrical. This symmetry can be seen when the function is expressed in the following equivalent form:

$$(\text{numerical-or} \quad A \quad B) = (A + B) - A \times B$$

Thus, it is clear that

$$(\text{numerical-or} \quad A \quad B) = (\text{numerical-or} \quad B \quad A)$$

Once the separate cumulative belief and disbelief factors have been calculated, the computation of the final certainty score is simple. All that needs to be done is to subtract the value of the disbelief factor from the value of the belief factor. This will yield a certainty score that is between plus and minus one. If the evidence against a class is stronger than the evidence in favor of it, then the final certainty score will be negative. If the evidence in favor of the class is stronger, then it will be positive. With one of these certainty scores associated with each of the candidate classes, the best choice may easily be obtained by selecting the class with the highest certainty.

6.1 The Confidence Factors

Since the confidence factors that correspond to each vote play a key role in the learning capability of the class predictor, it is important to know how they are evaluated. The confidence factors can be thought of as representing the level of certainty we may have about each of the votes that might be made. A simple but unsuccessful method for obtaining these factors is to update them after each prediction in such a way that they represent an approximate estimate of the probability that their associated votes will be correct. This estimate may be maintained by a limited memory estimator which is incremented each time a correct vote is made, and decremented each time an incorrect vote is made. The standard form for the estimation procedure is as follows:

If a vote was correct then: $\quad$ new factor $= (1 - \epsilon) \times$ old factor $+ \epsilon$

If a vote was incorrect then: $\quad$ new factor $= (1 - \epsilon) \times$ old factor

Where ϵ is a constant less than one.

Each confidence factor is initialized at a value of 1/2, and then either increased or decreased according to the performance of their corresponding expert. This basic scheme is very similar to that used by Samuel in his learning checker playing program [Samuel,1959].

Despite certain flaws, this method is intuitively satisfying since it will cause a confidence factor to increase if an expert makes many correct votes for a class, and decrease if the expert makes many incorrect votes for a class. The fact that the confidence factors have limited memory is also positive, since this places the greatest emphasis on the most recent performance. This way, when an expert gets better or worse with time, the confidence factor will be modified accordingly. Unfortunately, this method is not as ideal as it sounds since it produces terms that resemble probabilities much more than belief or disbelief factors. The key problem is that a mediocre expert which makes many predictions and happens to have a lucky streak can develop a higher confidence factor than a perfect expert which makes

infrequent predictions but is always correct. This imbalance may cause serious errors because the perfect expert will be ignored when it is most needed.

An improved method for updating confidence factors is to only modify them for experts which have been involved in an undesirable outcome. This is the case when either an incorrect vote is given too much emphasis, or a correct vote is given too little emphasis, resulting in the selection of an incorrect class. Under this approach, the confidence factors will only be modified if a final prediction is incorrect. When the class prediction for a day segment turns out to be correct, then none of the confidence factors should be modified since they can be considered to have been at the proper relative levels. This scheme will tend to give the most emphasis to those experts that are crucial to accurate decision making.

As an example, suppose we have three experts, and have only to decide between two different classes. Suppose also that class 1 will always occur on weekdays, and class 2 will always occur on weekends. Now, assume that the experts are found to vote as follows: Expert 1 will always vote for class 1 on weekdays, but it will abstain from voting on weekends. Expert 2 will vote for class 1 on every day of the week. Expert 3 will vote for class 2 on every day of the week. Now we may determine what would happen under the first method described by assigning reliability factors according to the probability of success of each expert. The probability of success for a vote in favor of class 1 by expert 1 is 1.0 since each time it makes this vote it is correct. The probability of success of vote in favor of class 1 by expert 2 is 5/7 since this expert is correct only five out of the seven times it votes. Similarly, the probability of success of a vote in favor of class 2 by expert 3 is 2/7. When the votes on a weekday are tallied, class 1 easily wins as it should. Unfortunately, when the votes for a weekend are tallied, class 1 still wins because expert 2 has a higher confidence factor than expert 3. This should not be allowed to happen because experts 2 and 3 are not supplying any valuable information at all. To make a decision on the basis of which of the two experts have been luckiest is not desirable in this case. The proper weighting of these experts would be to give

the highest weight to expert 1 since it seems to recognize a difference between weekdays and weekends, and the lowest weight to expert 2 since whenever it is not making a redundant vote it is actually making a harmful one. It should be clear that the improved updating method produces this desired result.

An additional improvement to the updating procedure is a method to account for the cost of choosing an incorrect class. Whenever an expert makes a vote on a particular class, there is always the chance that the vote will be incorrect. If an incorrect vote results in the wrong class being selected for a new day, then there is a good chance that more errors will be made than would have been made with the correct class. It is also true that some classes, even though they may not be ideal, may be better suited than others to provide reasonable performance on the new day. It is therefore desirable to bias the selection of classes toward those which would be most likely to provide reasonable performance even when they are not the best choice. This biasing may be reflected directly in the confidence scores for each of the experts. If an expert incorrectly votes in favor of the wrong class, or against the correct class, then the confidence factor for future votes of that type by that expert must be diminished in proportion to the severity of the error. On the other hand, if an expert makes a correct vote, but the final selection is not the same because of votes from the other experts, then confidence in this expert must be increased in proportion to the difference in error between the class that was actually selected, and the class that should have been selected.

It is possible to deal with cost considerations directly in the procedure for updating confidence scores. As mentioned earlier, confidence scores are only modified if the combined votes yield a final prediction which is incorrect. When this happens, there will be two types of correct votes, and two types of incorrect votes. The correct votes will be those in favor of the proper class as well as those against any other class. The incorrect votes will be all those against the proper class, and all those in favor of any other class. The confidence factors corresponding to the correct votes will be increased according to the formula:

$$\text{new factor} = \text{old factor} \times (1 - \epsilon) + \epsilon$$

And the confidence factors corresponding to the incorrect votes will be decreased according to the formula:

$$\text{new factor} = \text{old factor} \times (1 - \epsilon)$$

The only difference between these formulas and the those described earlier is that the size of ϵ will depend on the severity of the errors.

If an extremely costly mistake has been made, ϵ will be made large, causing a large change in the confidence factors for both the correct and the incorrect votes that have been made. The maximum value allowed for ϵ is 0.5, and the minimum value is zero. This is to insure that a single change in a confidence factor does not entirely ignore its previous value. When dealing with a vote that was made on a class that was not the correct class, the value assigned to ϵ is determined by the ratio between the distance scores of the correct class and the class that had been voted on. When dealing with a vote that was made on the correct class, the value of ϵ is determined by the ratio between the distance scores of the correct class and the class that was actually used. Recall that the distance score for a class is a measure in difference between the parameters for that class and the parameters that are generated from the new pattern. This distance score does not provide a direct measure of the errors a class would produce on the new pattern, but it does indicate this indirectly since the error score tends to increase with the distance score. In spite of this, the distance measure actually works better than the error score for scaling ϵ since it is insensitive to fluctuations in the weight factor of the decision function.

The formulas used to determine ϵ are as follows:

If the confidence factor being changed is for a vote on an incorrect class then:

$$\epsilon = 0.5 \times (1 - \frac{\text{The distance score for the correct class}}{\text{The distance score for the class that was voted on}})$$

If the confidence factor being changed is for a vote on the correct class then:

$$\epsilon = 0.5 \times (1 - \frac{\text{The distance score for the correct class}}{\text{The distance score for the class that was actually used}})$$

The distance scores in these formulas are the same as those used to determine the the best classification for the occupancy pattern of a new day. This distance score reflects the cost of a mistake in the sense that an incorrect class which has a distance score that is much larger than the distance score for the correct class is likely to yield many more errors than the correct class. On the other hand, if a new pattern lies right between the correct class and another class in parameter space, there will be no severe penalty on a vote for the other class since the choice was so close. By using this score to scale the modification of confidence factors, any prediction errors should be most likely to involve classes which are nearly equivalent to the correct class, and errors involving classes which are extremely different from the correct class should be rare. This way, the resulting errors in temperature control should be small even in the event that an incorrect class is used.

6.2 Control of the Class Predictor

There are certain special control procedures that are needed for the class predictor to function properly. These procedures are primarily concerned with choosing the correct defaults when limited information is available. These defaults are needed when the classes available for selection are limited, or when the actual classification of a preceding segment is not known before a vote has to be made.

When the system is just starting out on the first day, no normal classes will exist yet so the decision algorithm will have to use a default class called class 0. This class will force the decision algorithm to either keep the temperature set up for every time slice, or keep it set down for every time slice depending on the value of the relative cost factor. If the relative cost factor is greater than one, this means that the user puts a higher priority on comfort than on saving energy, so the temperature will remain set to the comfort level. If the relative cost is less than one, the decision algorithm will set the temperature to the lower conservation level, but the people sensors will still take care of heating zones that are occupied. This approach

is intended to guarantee either maximum user comfort or maximum energy savings during the early learning stages, depending on which is most important to the user.

In time, the system will collect pattern data and begin to establish groups of classes which are exclusive to particular zones of the building and segments of the day. If only one class has been established for a particular segment and zone, then it will not be necessary for the predictor to do any voting for that zone. Class 0 will never be chosen when there is another alternative, so the other class is the only remaining choice. When two or more classes have been established for a particular zone in the given day segment, then the predictor must vote on which class to select using its collection of independent voters. These voters include the periodic predictor and several correlation experts.

One of the correlation experts requires some special control considerations. This is the "previous segment" expert which tries to correlate the classes occurring on consecutive segments and predict the class for the upcoming segment based on the class of the previous segment. The problem with this expert is that it does not have all the information it needs to make an accurate prediction at the beginning of a segment. This is due to the fact that the final classification for the previous segment cannot be obtained at the beginning of the current segment. The correctness of the control decisions made at the end of the previous segment cannot be evaluated until the first few time slices of the current segment have passed, and this evaluation is crucial to the classification process. To accommodate for this lack of information, the class of the previous segment is assumed to be the class that was last in use when the segment ended. It is reasonable to make this guess since the last class in use will be the survivor of the self evaluation process. If this guess should turn out to be incorrect, the vote will have to be redone with the new information. If the overall vote is changed, then the new class will be used as if it had been selected from the start of the current day segment. This, of course, will not change any past control decisions, but it will effect the evaluation of the experts when the current segment is complete and it comes time to modify

their confidence factors.

7. THE VOTING EXPERTS

Now we may discuss the details of how the specific voting experts actually work. The experts that I use may be divided into two main types. The first type is capable of extrapolating a binary sequence on the basis of various periodic tendencies it finds within the sequence. This is valuable because the incidence of any given class over the lifetime of the system can be expressed as a binary sequence with each element of the sequence representing a different day. A "1" in the sequence can represent the fact that the given class was chosen for the corresponding day, and a "0" can represent the fact that another class was chosen for that day. If a class has any periodicity in its occurrence, this expert can find it and use it to guess when the class will occur again.

The second type of expert is capable of looking for correlations between the use of the different classes and various events. An event can be any arbitrary feature that the programmer wishes to specify. Each expert has its own special feature that it tries to correlate with the occurrence of different classes. The experts that I used are the following:

1. An expert to correlate the use of each class with the one used a week earlier.
2. An expert to correlate the use of each class with the one used a day earlier.
3. An expert to correlate the use of each class with the one used a segment earlier.
4. An expert to correlate the use of each class with the occurrence of any special days.

These are just a few of the possible correlation type experts that actually could have been employed.

7.1 The Periodic Expert

As stated earlier, the periodic expert looks at the record of classes that have been used in the past, and forms a binary sequence for each of the classes. The binary sequence for a given class will have an element for each day, and an element will be a "1" if that class was used on the corresponding day and a "0" if not. The periodic predictor is able to take the binary sequence for a class and make one of three choices regarding its vote for that class

on the upcoming day segment. The three choices are to vote for the class, against the class, or to simply abstain from voting. If abstention is chosen, the expert will have no influence on that class, and its confidence factor associated with that class will be unaffected by later evaluation procedures. This predictor operates independently on each of the candidate classes. An understanding of how this predictor works may be obtained through an understanding of how it looks for periodicities within a binary sequence and uses this to extrapolate the sequence into the future.

Periodic prediction is a problem common to many disciplines, but as yet it has no universal solution. Since this problem is so vast, I cannot begin to expect to obtain a perfect predictor, but at least I can obtain a reasonable one. This is all that is necessary anyway since the periodic predictor is only one of several experts in my system. In many approaches toward periodic prediction there is much emphasis on first obtaining a model of the sequence to be predicted, and then finding various parameters of that model from direct analysis of the sequence. If the model is valid, it should permit the extension of the sequence once it has the correct parameters. The model I use for predicting human behavior allows binary sequences of different periods to overlap in an "inclusive or" fashion to model the effect that interdependent behavior patterns of different people might have on the class sequence.

Because of this unusual model, I have taken a heuristic approach which tries to analyze a binary sequence in much the same way that a person might. One approach people might take when trying to find periodicities in such a sequence is to try to match templates with the sequence. For instance, if we think a binary sequence might have a period of ten elements, we could take a template that has a window every ten spaces, and place it over the sequence to see if each window had the same element in it. In fact, even though the template may not give good results for each of the possible starting points, we might still be able to extrapolate the sequence on the basis of those starting points that matched well. Ideally, we would like to find the fewest number of templates that can accurately describe the sequence.

The best way to visualize the template matching problem is to consider a binary sequence such as the following:

$$1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\cdots$$

Now suppose we had a template made of cardboard that could be placed over this sequence, and in this template, holes were cut at regular intervals. We can let another binary sequence

$$1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\cdots$$

represent this template if we say that there is cardboard where we see zeros, and holes where we see ones. The template shown above would be used to test for periodicities of seven. To use this template to test the sequence given above, we would begin by placing the start of the template at the start of the sequence and look to see if the elements in all the windows match. For the sample sequence above, each window would have a one in it so we would save the hypothesis that the ones in the sequence may have a component with period of seven and zero displacement. Next, we would displace the template by one element, and again we would notice that each window of the template had a one in it. This means we would also save the hypothesis that the ones in the sequence have a component with period of seven and a displacement of one. We continue to displace the template one step at a time up to one step less than the period length. When the template is displaced by two steps, we will find that the first window has a one, the second has a zero, the third has a one, and the fourth has a zero. This is not a satisfactory match for either the ones or the zeros. When the template is displaced by three steps, all of the windows will have zeros in them. This will allow us to hypothesize that the zeros in the sequence may have a period of seven with a displacement of three. Using this approach for all templates which can at least cover two elements of the sequence, we can generate a very large number of hypotheses about the possible periods and displacements of the sequence elements. The problem then becomes one of obtaining the most complete description of the sequence at hand with the fewest number of hypotheses.

It is important to consider how a description of a sequence may be obtained from a set of hypotheses of the form discussed above. Since each hypothesis specifies where a set of ones or zeros may be found at specific positions in a sequence, there must be a way of combining these hypotheses to obtain a description of all the elements of the sequence. For example, if the elements from all hypotheses are superimposed upon one another, the original sequence may be reproduced exactly except in places where no hypothesis could produce a match. The object however, is not merely to obtain a description of the original sequence, it is to predict the next element of that sequence as well. This means that we will want the representation to reflect real periodicities in the sequence as much as possible. Since there are generally many more hypotheses generated than are actually needed to describe the sequence, the problem becomes one of selecting the subset hypotheses which is most likely to use the real periodicities to represent the sequence. If real periodicities do exist, it is most likely that this subset will be the smallest possible subset that can still give a complete description of the sequence. This is because each hypothesis can be thought of as containing purely periodic information about the sequence. When hypotheses of different periods are superimposed upon one another however, a certain amount of aperiodic information is added, and the periodic information becomes less significant.

Since the number of possible combinations of hypotheses can become very large for even a moderate number of hypotheses, it is necessary to use certain heuristic techniques that will simplify the search for an optimal or nearly optimal subset. This is done in my system through several different methods. First, all of the components are sorted in order of the shortest period to the longest period. This helps to find the components which provide the most information about the sequence since they will have more elements that must fit into the sequence. The procedure that follows involves an attempt to build a complete representation of the given sequence by picking components out of this sorted list one at a time until either the representation is complete, or none of the remaining components can improve the representation.

The selection of a new component is based primarily on the amount of new information it provides to the representation. In this case, each element that a component can add that was previously unspecified constitutes new information. This permits the quantification of new information simply by counting the number of such elements that are added by a hypothesis.

The search for each new component starts with the smallest period component that is still unused. The amount of new information provided by each consecutive component in the sorted list can be computed, and a record of the best match can be maintained. When we get to components with periods that are so large that the maximum number of elements that they could match is less than the current high match score, then the search is complete. The component with the best match can then be added to the representation, and the search repeated for the next component.

There is a slight complication when there are several components which have equal match scores. If one of these components is just chosen arbitrarily, it is likely to overlap some of the elements that the others could have contributed, thereby reducing the value of components that would have otherwise been of equal value. The problem with this is that there may be a few components that can fit together with no overlap, and another which overlaps the others a great deal. If the overlapping component is added first, the other components will no longer appear useful. On the other hand, if one of the nonoverlapping components is added first, then the other nonoverlapping components will still be just as useful as before. The net match value of the given group of components will thus be much higher if the nonoverlapping components are used first. The components that most commonly fit together without overlapping are those with the same period. Therefore, when several components are found with equal match scores, they are first inspected to see if any of them have a common period. If this fails, they are inspected to see if any of them has a period which matches a period of one of the components that has already been added to the representation. If any components should satisfy these conditions, they will be added first. If several new components are found to have

the same period, they may actually be added to the representation all at the same time since they will have no effect on each others match scores.

The overall scheme for the periodic expert can be described as a generate and test algorithm. First, all of the possible hypotheses about periodicities in the sequence are generated, and then, these hypotheses are tested to see which of them provide the best description of the sequence. The net result is that we end up with a set of hypotheses about what the predominant periodicities of the sequence might be. The next problem is to use these periodic hypotheses to extrapolate the sequence. Here, it is important to decide how to combine the periodic components to make a reasonable extension of the sequence. If for instance, we used only the components for the ones in the sequence, and we superimposed them on top of one another, we could extrapolate the result to see all the likely places where ones might occur later in the sequence. To superimpose the components for the ones, we start with a sequence that is initially all zeros, and change any element to a one if it corresponds to a one in any of the components. If this is done, however, there will be a tendency for the extrapolation to be excessively biased in favor of ones. This is because the components will most likely have different periods, but have many instances where they overlap within the sequence. When we extend the sequence, however, there may not be so much overlapping, and the extended part of the sequence will be filled with many more ones than is appropriate. If we try superimposing the components which specify positions of the zeros in the sequence, we get the same problem, but this time, the result will be biased in favor of zeros.

The solution to this problem is found by combining the components for the ones with the components for the zeros in a reasonable manner. Within the known portion of the sequence, there will be no overlapping of the two different types of components. This is because the components are chosen such that they are completely accurate. Any template which was found to contain both ones and zeros would not have been selected as a potential candidate. There is a good chance however, that there will be some overlapping of ones and zeroes when

all the components are extended. It is then necessary to decide what the correct prediction should be when a certain number of hypotheses predict a zero should fall at a particular point in the sequence, and another set of hypotheses predict a one should fall at the same point. The simplest solution is to make a prediction according to whether there are more predictions for ones, or more predictions for zeros at the point in question. If the hypotheses predicting ones outnumber the hypotheses predicting zeros, then the final prediction will be a one. If the hypotheses predicting zeroes outnumber those predicting ones, then the final prediction will be a zero. If the number of hypotheses in favor of ones and zeros are equal, then no prediction will be made. These predictions are translated into votes by the periodic expert by assigning an affirmative vote for the class being analyzed if the prediction is a one and a negative vote for the class being analyzed if the prediction is a zero. If no prediction is made, then the expert will have to abstain from voting on that class.

7.2 Correlation Experts

All of the experts that try to use correlations as a basis for prediction use the same basic correlation functions but look for different features to be correlated. They all try to predict the upcoming class based on the correlations between the candidate classes and certain observable events. The set of known events can include things like the class that was used a week, day, or even just a segment of the day earlier, as well as calendar information such as the occurrence of user defined special days. If the coincidences of classes and known events such as these are recorded, then when an event is found to be highly correlated with the occurrence of a given class, or perhaps the non-occurrence of that class, then every time the event occurs, it will provide some information about which class to select.

Each correlation expert has its own special incidence matrix for each day segment within each zone. This is because the sets of candidate classes are grouped in these terms. Since an expert looks only at events of a single type, the incidence matrix for a given expert only needs to record the number of times each available class is selected when any of the events

of the specified type are also found. For example, suppose we have an expert that tries to correlate the class selected on each day with the class that was selected the day before. In this case, the event type is specified by the fact that the expert looks only at relationships between the present class and the class used one day earlier. The different events of this type correspond directly to the number of classes that exist for that zone and day segment. To see what a typical incidence matrix might look like in this case, we can study the following sequence of three classes:

1 3 1 1 3 1 1 3 1 1 3 1 2 3 1 1 3 1 1 2 1 2 3 1 1 2

The incidence matrix that would result from this sequence for an expert that looks for correlations between consecutive pairs of days is shown below:

		1	2	3	Present Class
Class used on	1	6	4	5	
the previous day	2	1	0	2	
	3	7	0	0	

This matrix is to be interpreted as follows:

Class 1 has been found to follow itself six times, class 2 one time, and class 3 seven times. Class 2 has been found to follow class 1 four times, itself zero times, and class 3 zero times. Class 3 has been found to follow class 1 five times, class 2 two times, and itself zero times.

A very simple way to use this matrix for prediction is first to find the row that corresponds to the current event, and then find the class which has had the maximum number of incidences and the class which has had the least. If we wanted to extend the above sequence, for example, we would look in the row labeled 2 since we want to predict which class will follow class 2 which is at the end of the sequence. Then, we would find the most incidences for class 3 which had two, and the least incidences for class 2 which had none. The predictor can then use this information to give a vote in favor of class 3, and against class 2. The predictor would have

to abstain from voting on class 1. This method is equivalent to calculating the probabilities that each class will occur given a particular class had occurred the day before, and choosing the classes with the maximum and minimum probabilities for the affirmative and negative votes respectively. If probabilities were to actually be used, the only difference would be that each row would be normalized such that the sum of the elements was equal to one. Since normalizing the elements of a row does not change their relative sizes, the classes with the maximum or minimum probabilities will be the same as the classes with the maximum or minimum number of incidences.

This simple scheme is incomplete because it produces a few subjectively unsatisfying results. The problem can be seen in the above example when we are trying to predict which class will follow class 1. Using the scheme explained above, the predictor will vote in favor of class 1 which has six incidences, and against class 2 which has only four incidences. It seems completely unreasonable, however, to vote against class 2 when the only times class 2 has been known to occur have been right after an occurrence of class 1. From this we see that the simple scheme fails due to the fact that the resulting vote is inappropriately biased against class 2 only because of its infrequent occurrence in the sequence.

To correct for this problem, we might consider an approach which compensates for the effect of having one class dominate the correlation prediction simply because it occurs most often. To do this, each column of the incidence matrix can be normalized to reflect the probability for each event having occurred given that a particular class is to be used afterwards. Note that this is calculating the probability of the cause given the effect whereas before we were looking at the probability of the effect given the cause. After the columns are normalized, the rows can then be searched for the classes with the highest and lowest probabilities. Where the classes with highest or lowest probabilities also have the highest or lowest coincidences, an affirmative or negative vote is justified. If the classes with bounding probabilities do not also have bounding coincidences, then it will be necessary to abstain from voting on them.

Because incidences are to be normalized in rows, and then compared along the columns, care must be taken to estimate the probabilities in a reasonable manner. If we were to merely divide each entry in the matrix by the sum of entries in the same column, we would not obtain a good probability estimate in cases where the total number of incidences in the column are small. For instance, it would not necessarily be a good estimate in the above example to say that the the probability of class 1 preceding class 2 is 1, and the probability of classes 2 and 3 preceding class 2 is 0. There is simply too little information available to place such extreme values on these probabilities. An alternative approach is to start with an a priori estimate of these probabilities, and use the accumulated information about incidences to bias these probabilities. This is very similar to the method used to estimate the parameters in the probabilistic representations for daily behavior patterns. If the a priori probabilities are to all be 1/2, each probability is computed by adding one to the number of incidences for the corresponding element in the matrix, and dividing it by two plus the total of all the incidences in the column of that element. In cases where the set of events are mutually exclusive, these probabilities must again be normalized such that the sum of all the probabilities in a column is equal to one.

Using the above example, and assuming that the events are not mutually exclusive even though they actually are in this case, we would obtain a matrix of probabilities as follows:

		1	2	3
		Present Class		
Class used on the previous day	1	$\frac{7}{16}$	$\frac{5}{6}$	$\frac{6}{9}$
	2	$\frac{2}{16}$	$\frac{1}{6}$	$\frac{1}{9}$
	3	$\frac{8}{16}$	$\frac{1}{6}$	$\frac{1}{9}$

Now, if we make the correction for the fact that the events are mutually exclusive since only one of the three classes can be used on a previous day, we obtain the following matrix of probabilities:

		1	2	3	Present Class
Class used on	1	$\frac{7}{17}$	$\frac{5}{7}$	$\frac{6}{10}$	
the previous day	2	$\frac{2}{17}$	$\frac{1}{7}$	$\frac{1}{10}$	
	3	$\frac{8}{17}$	$\frac{1}{7}$	$\frac{1}{10}$	

Looking at this matrix, we see that the first row has a maximum probability when the present class is class 2, and a minimum probability when the present class is class 1. The second row has a maximum probability when the present class is class 3 and minimum probability when the present class is class 1. Similarly, the third row has a maximum probability when the present class is class 1, and a minimum probability when the present class is class 3. These relationships are most easily seen when expressed in a matrix form as follows:

		1	2	3	Present Class
Class used on	1	min	max		
the previous day	2	min		max	
	3	max		min	

We can compare this with the maximums and minimums along each row of the original incidence matrix which may also be expressed in matrix form:

		1	2	3	Present Class
Class used on	1	max	min		
the previous day	2		min	max	
	3	max	min	min	

The voting algorithm only allows an expert to make a vote for elements where these two matrices agree. In this case there are only three elements with coinciding maximums or minimums in both the incidence matrix and the matrix of probabilities. These elements are shown below:

		1	2	3	Present Class
Class used on	1				
the previous day	2			max	
	3	max		min	

This result seems to satisfy the heuristic need to only allow votes on a class when there is a definite relationship between it and one of the events. A class should not have a greater chance of being correlated with any of the given events simply because the total number of occurrences of the class has been large. In the above example, we see that if we need to predict which class should follow class 1, no votes can be made. This is because even though class 1 has been found to follow itself most often, class 2 has followed class 1 every time it has been used. With this conflict, it is best to abstain from voting.

If we need to predict which class should follow class 2, we can vote in favor of class 3, but that is all. Class 3 has been found to follow class 2 most often in spite of the fact that this class 3 has occurred very few times. Even though class 2 has never been found to follow itself, there have not been enough occurrences of class 2 to justify a vote against this possibility.

If we must predict which class should follow class 3, we may make a vote in favor of class 1 and against class 3. Classes 2 and 3 have both actually had no incidences after class 3, but since class 3 has occurred more times overall than class 2, there is greater certainty that class 3 will not follow itself than there is that class 2 will not follow class 3.

When two or more classes are equal in both their probability score and their coincidence score for a particular event, these classes are considered to be tied, and the expert will be allowed to make the same vote on each of them. In the above example, when we are considering classes which should follow class 3, classes 2 and 3 are tied for the minimum number of coincidences but the probabilities associated with these classes are not equal. As a result, the two classes are not considered tied, and we are able to choose class 3 for a negative vote because its probability score is the lowest. It is useful to note that a tie can only occur, and

it must occur, when the incidences for two classes are equal in a given row and the sum of the incidences down the columns for these classes equal as well. In other words, a tie exists when two classes have been found to occur the same number of times after a given event as well as the same number of times overall.

The above example demonstrated how a vote would be made when the events that are used as the basis for prediction are mutually exclusive. When these events are not mutually exclusive, very little is changed except that the second normalization of the probability matrices is not performed. The only expert in the system which does not use mutually exclusive events is the one that deals with user defined special days. There is actually an expert for each one of the user defined days, but they all share the same incidence matrix. The only reason for having unique experts for each day type is so that they may each be evaluated for reliability independently. The incidence matrix that these experts share has a row for each of the user defined day types, and a column for each of the classes that might be used on those days. In other words, the day types are considered to be the events, and the experts must determine which classes are correlated with these events. Each special day expert is called upon only when the current date corresponds to its associated day type. When this happens, the expert or experts use the special day incidence matrix and each make their predictions according to the incidences that correspond to their associated special day. When the incidence matrix is to be updated after a day with multiple names, an update is made in each row that was named for that day.

The incidence matrix that the special day experts share is handled in a different manner than the incidence matrices for mutually exclusive events. Since the user defined special days are not mutually exclusive, a single day can be associated with several day names. When the incidence matrix is updated after the day is complete, the class column will be selected according to the class that was used on that day, and the incidences for that class will be augmented in each row that corresponds to a name that was used for that day. This

means that even though a class may have only occurred once, its total updates may still be greater than one. When the probability matrix is derived from the incidence matrix, it is not appropriate to approximate each probability by dividing the corresponding incidence by the total number of updates. Instead, the incidences should be divided by the total number of times the class has been used. Note that if the events had been mutually exclusive, the total number of updates for a class would be equal to the total number of times that class had been used. It is only because the events are not mutually exclusive that this distinction needs to be made.

As an example of why this is important, suppose that two different user defined names are always associated with the same dates. Also, suppose that the same class is always used on these dates. As a result of this, the column in the incidence matrix associated with this class will have the same number of updates for both of the day names. If the corresponding probabilities are calculated by the standard method discussed earlier, the individual updates will be divided by the total number of updates in the column. This will give probabilities of 1/2 for each element. This is not a valid result. Since the day names had both been active events each time the class was used, the probabilities associated with each of them should be close to 1. To permit this result to be obtained, it is necessary to keep track of the actual number of times the class has been used. When this is used instead of the total number of updates in a column, the result for non mutually exclusive events is much more reasonable, and the result for mutually exclusive events is unchanged.

8. SELF EVALUATION

Once the class predictor has made a decision on which class to use in each zone on an upcoming segment, the decision function will begin to apply the probabilistic parameters of those classes to determine the proper temperature setting in each zone during the next time slice. No matter how well the class prediction system performs, it is bound to make some mistakes since people's behavior patterns are never perfectly regular. Therefore, it is important that a facility be provided to allow the system to evaluate its performance incrementally at the end of each time slice to make sure that the initial class predictions were indeed reasonable.

The only criterion that the system can use to evaluate its performance incrementally is the number of errors the decision function has made while using the selected classes. The decision function has the potential for making only two different types of mistakes. It can decide to have the temperature setting turned up when there is actually no need for it, or it can decide to have the temperature setting turned down when there really is a need for it. The need for the temperature setting being turned up must be defined not only as the immediate occupancy of a zone, but also, as the occupancy of a zone within a time period shorter than that required to bring the zone to the desired temperature. This brings in a small complication to any attempt to evaluate current performance since the correctness of the most recent decisions will be unknown until the appropriate heating time constant has passed. In other words, if it takes an hour to bring a particular unoccupied zone to a comfortable temperature, then the correctness of any decision on whether or not to modify the temperature setting for that zone cannot be evaluated until one hour after the decision is made. This is not a problem when a zone is occupied. In this case, all decisions that were made up to and including the time slice of the most recent occupancy can be evaluated for their correctness. This is because unoccupied segments may or may not need regulation, depending on whether there will be an occupancy in the near future, while occupied time slices always

require regulation and always allow specification of the need for regulation in the time slices preceding them.

With a knowledge of the current errors that have been made while applying the parameters of a particular class to the behavior patterns of the current day, it is possible to compare these errors with certain standards to determine whether or not the current performance is acceptable. The standards that are most convenient to use are obtained from the average and maximum error information that is stored with each class. This information is the same as that used in the algorithm for determining the acceptability of a class during the final classification process. In fact, the same algorithm that is used to compute the final error acceptability may also be used to evaluate the intermediate acceptability with only a few modifications.

Modifications to the basic acceptability algorithm are necessary for intermediate evaluation because we have to compare partially accumulated error information with statistics for the total errors that have accumulated at the end of a day segment. It is therefore necessary to place limits on the accumulating errors to reflect the likelihood that the chosen class is correct given the average number of errors that are usually made with that class. To do this, the maximum allowable error score is reduced according to the number of time slices of the current day segment that have actually been used in calculating the current errors for the class.

There are many possible ways to rescale the maximum allowable error score. The more the error score is reduced, the more likely we are to reject a class that is being used when it shouldn't. Unfortunately, we are also more likely to reject a class that is being used correctly but has had most of its errors at the beginning of the day segment. To avoid rejection of valid classes, I have chosen a scaling function which sets the maximum allowable error to half its normal level at the beginning of a day segment, linearly increases the scale to unity as the day segment approaches its halfway point, and holds the scale at unity from the halfway

point to the end of the day segment.

When the errors for the currently applied class exceed the limit, the class is not immediately rejected. An appropriate replacement must be found first. To find a replacement, all classes that have not yet been rejected are evaluated just as though they had been selected at the start of the day segment. Just like the class that was actually used, these other classes will each have their own current error scores and maximum error limits. Therefore, the best replacement can be found from the class which has the lowest error score and still does not exceed its error limit. It is important that the replacement class not exceed its error limit since it would then be a candidate for rejection immediately after it was chosen. The current class will not be rejected unless the replacement class has a lower error score. This is to prevent replacing a class with one that actually performs worse simply because the error bound on the first was too stringent.

9. THE DATA BASE

A knowledge of how the data base for my system is organized should provide a better understanding of how everything works. The actual structures I have used may not be ideal for a practical implementation of the learning algorithm, but they have provided a very flexible and efficient framework for development of the system. The learning algorithm was implemented in the LISP programming language. Although practical implementations of this algorithm could be written in almost any language, the LISP language and the LISP machine which runs the language simplified the initial development tremendously. The LISP language is very useful because it supports a large variety of data structures, and it also permits a high degree of modularity. The primary structures I use are arrays and property lists. A property list provides a means for associating attributes with symbols. A symbol may have an arbitrary number of properties, and each property will have a name and an associated structure. The fact that LISP allows functions to be written and tested independently is very helpful in permitting the development of modular and hierarchical programs. This is also of definite value for program verification.

9.1 General Structure

The data base was implemented as a tree structure of arrays. This is intended to eliminate the use of global variables as much as possible by assigning each important variable its own special location in the data structure. This way, references to all elements of the structure may be made through the single topmost node. Of course, functions are allowed to pass pointers to intermediate levels of the data structure as arguments to other functions.

Since the same operations have to be performed on many different parts of the data base, it has been designed to have many identical sub-structures within the main structure. Each sub-structure may be constructed as many times it is needed by a single function. For example, the structures that store the data used to define a class are all identical to one another. This not only allows these structures to be assembled and initialized by a single

function, it also provides a uniform environment for the procedures which operate on the data within them. This way, the data for any class will be processed in the same way as any other class. If there is a need to process a certain sub-structure differently than others, the indicators for this may be stored within the structure itself.

At the highest level of the data structure, the most generally accessed data is stored. The top node of the data structure is labeled "top-frame" in Figure 2. As shown in the figure, this node has the following property names: "special-days", "current-day-names", "saved-day-names", "thermo-model", "day-segments", and "seg-elements". This node evaluates to an array containing the highest nodes of sub-structures which carry all of the data relevant to the individual zones.

The purpose of each property of the "top-frame" node is somewhat described by its name. As seen in Figure 3, the "special-days" property is actually a three dimensional array which is used to store all of the names of special days, and all of the dates that might be associated with them. The names are stored along one dimension of this array, and the dates are stored along the other. The third dimension is used to store the day, month, and year separately. The "current-day-names" property is a one dimensional array which is used to store all the special day names that correspond to the current date. Rather than actually storing the names in this array, only their index in the special-days array needs to be saved. The leader of the current-day-names array is used to store the current date. The "saved-day-names" property is also an array, and it is exactly the same as the current-day-names array. This array is used to save the special day names that were used on the previous day. This is necessary for the correct operation of the book keeping procedures of the class prediction algorithm. The "thermo-model" property is a structure which stores the parameters used for modeling the thermal transients and energy loss for the entire building. The "day-segment" property is a variable to indicate the number of segments each day will be divided into. This provides important information about how the system will behave as well as about the lower

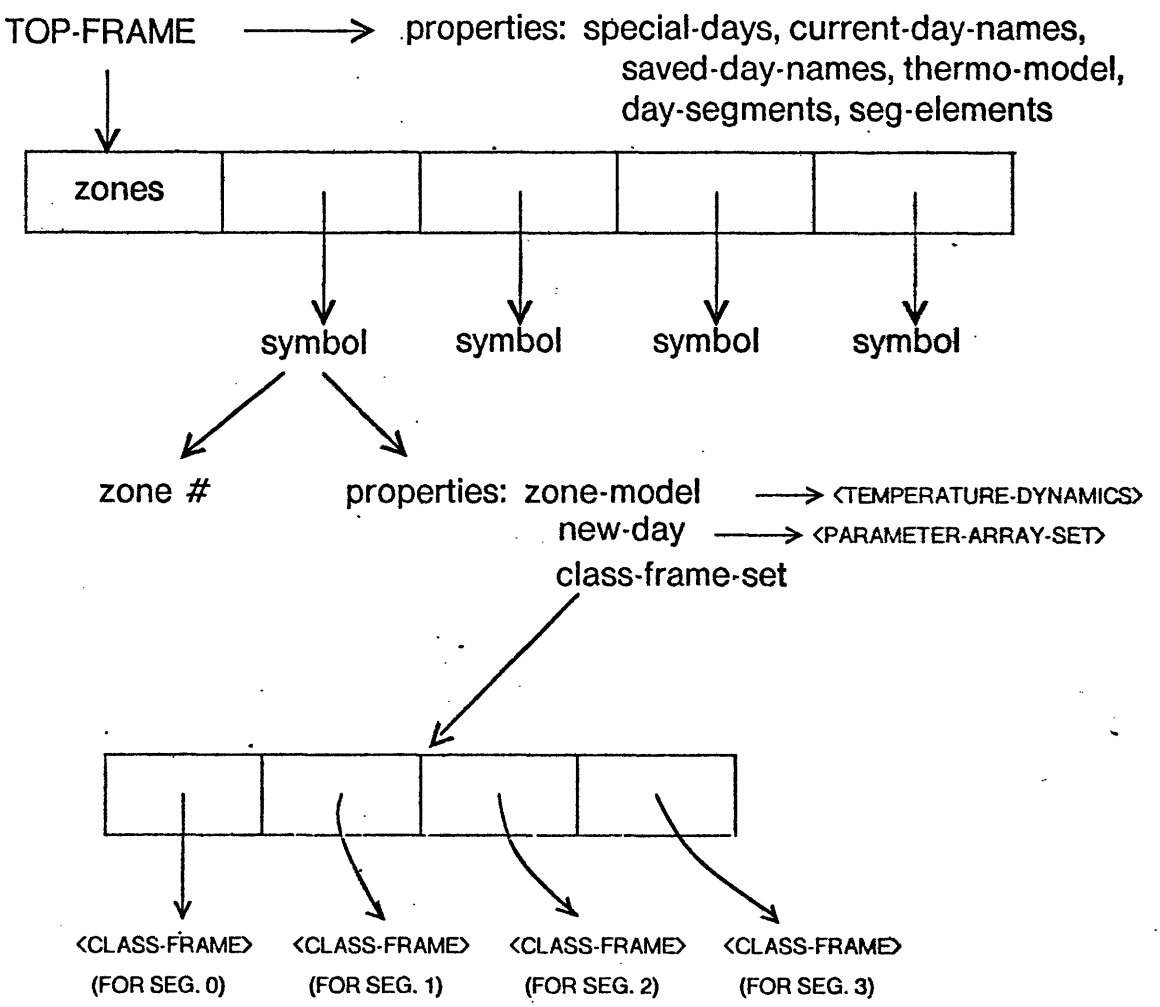
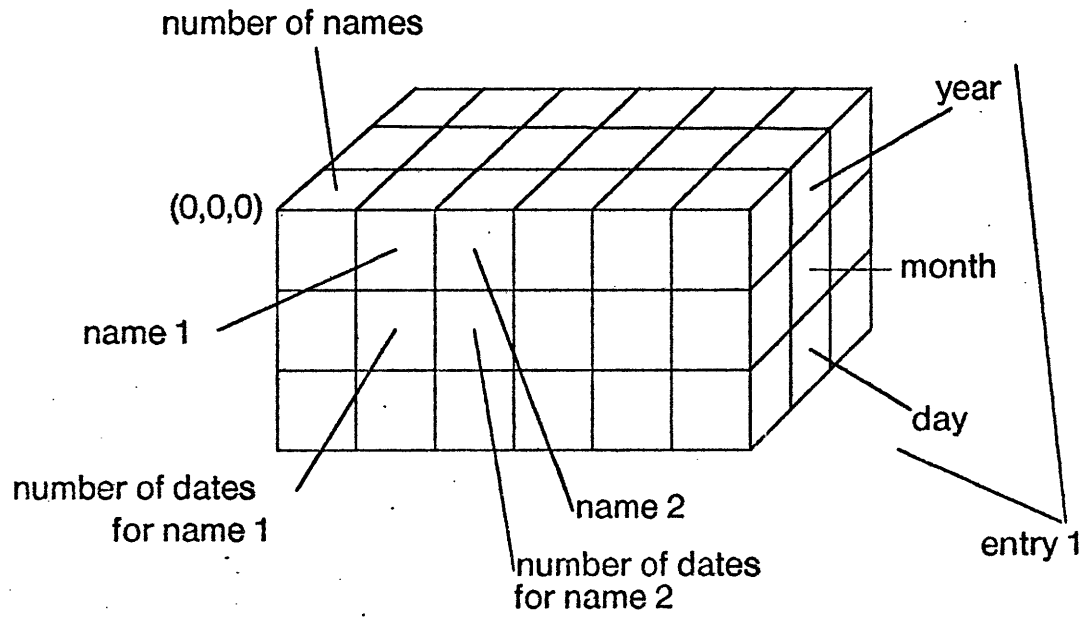


FIGURE 2: TOP LEVEL OF THE DATA BASE

SPECIAL-DAYS: (3-D array)



CURRENT-DAY-NAMES:

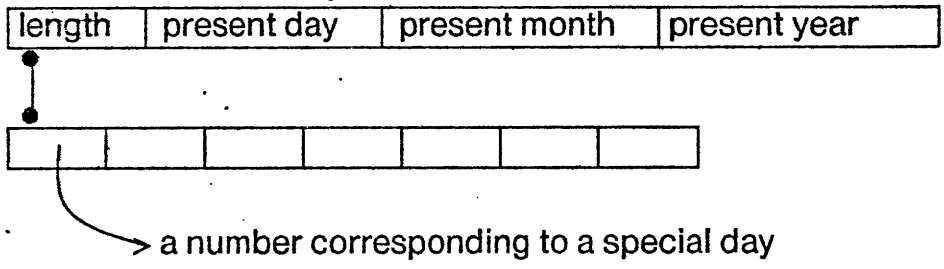


FIGURE 3: STRUCTURES FOR SPECIAL DAYS

levels of the data base. The "seg-elements" property is a variable to indicate the number of time slices that will be within each segment.

As seen in Figure 2, the top nodes of the sub-structures for the individual zones are symbols. each symbol evaluates to the zone number of the zone it represents, and has three properties. The property names belonging to these nodes are "zone-model", "new-day", and "class-frame-set". The "zone-model" property of the zone node is a structure which is used to model the thermal transients and energy loss within the given zone. This is equivalent to the structure associated with the "thermo-model" property of the "top-frame" node. The zone-model structures are needed to model a building with perfect isolation between zones, and the thermo-model structure is needed to model a building with no isolation between zones. The "new-day" property of the zone node is an array which I shall call a "parameter-array-set". The parameter-array-set contains all the parameters needed to define a pattern class. The parameter-array-set associated with the "new-day" property represents only the behavior pattern observed on the most recent day segment. When the most recent day segment is being classified, this parameter-array-set is compared with the parameter-array-set of each existing class for that day segment. If a new class is created, the new-day parameter-array-set is directly transferred down to the level of the other classes. The "class-frame-set" property is an array which has an element for each segment of the day. These elements may be called "class-frames" since they each contain all the class information associated with their corresponding day segment.

A "class-frame", shown in Figure 4, is an array which contains the parameter-array-sets of all the classes that belong to the particular zone and day segment grouping. In addition, the leader of a "class-frame" array contains several structures which are common to all of the existing classes within the array. One of these structures is called the "class-sequence-array", and the others may each be called "relation-arrays". There is one relation-array for each of the correlation experts of the class sequence predictor. Only one relation-array is needed,

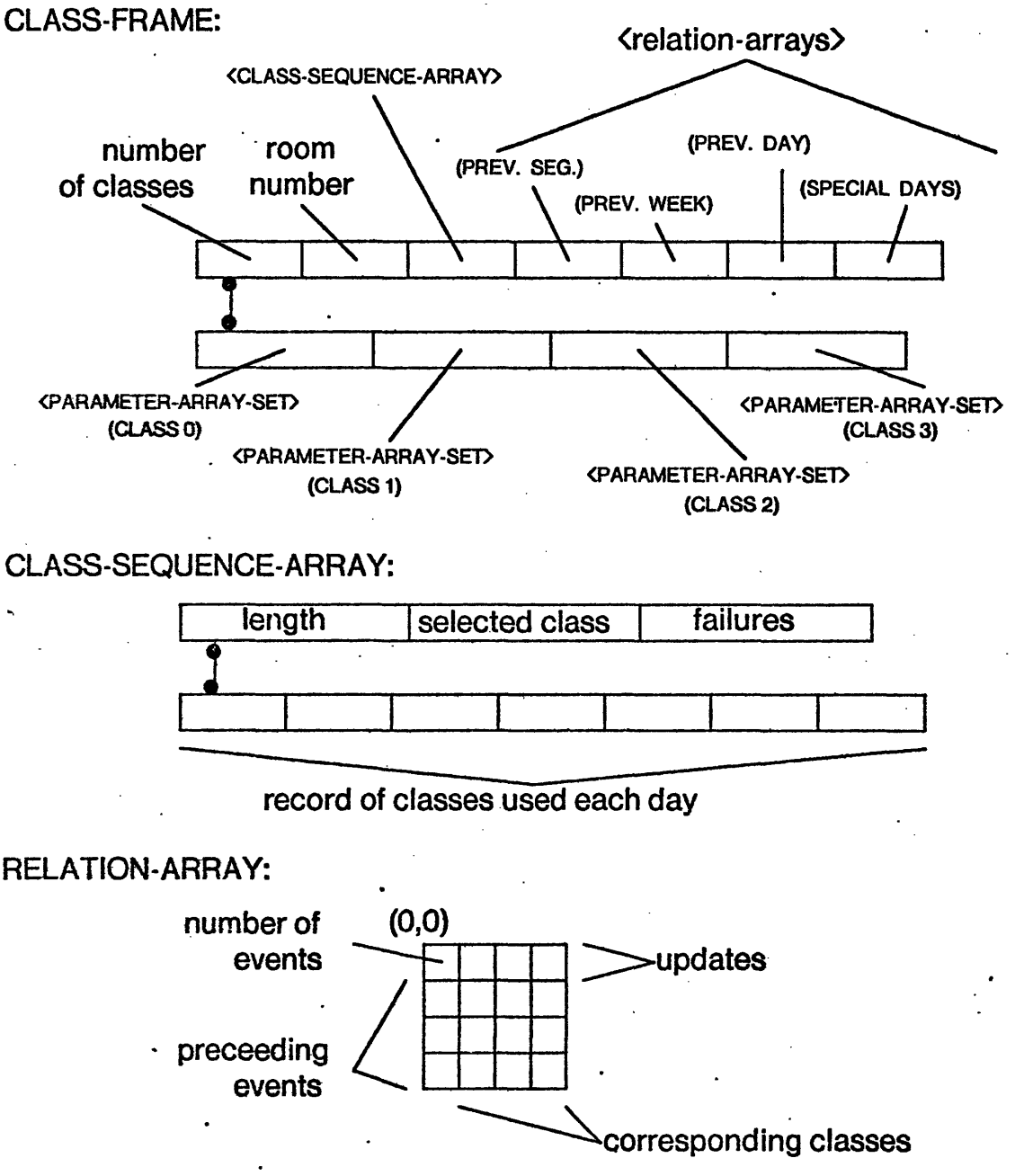
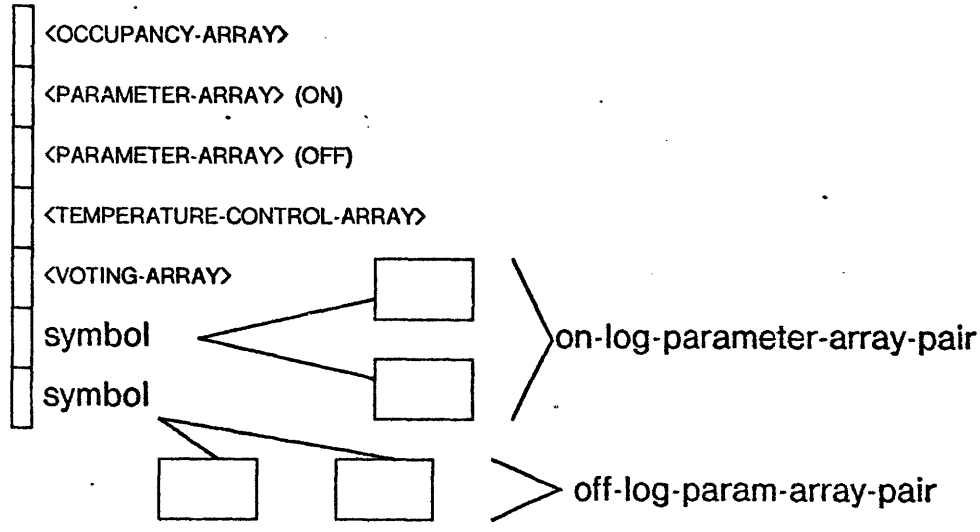


FIGURE 4: SUB-STRUCTURES FOR CLASS RELATED DATA

however, for all of the special day experts. The class-sequence-array is merely a record of the final classifications that were made on each day for the given day segment. This information is necessary for the proper functioning of the periodic predictor, and for keeping the relation-arrays correctly updated. The leader of the class-sequence-array is used to store the currently selected class, and the number of times the class selection is changed due to poor performance when the associated day segment is in progress. The relation-arrays are two dimensional arrays used to correlate particular events with the classes within the class-frame. Each row of a relation-array corresponds to a particular event, and each column corresponds to one of the classes in the class-frame. The number of times that each class has been found to correspond to each event is recorded in the corresponding row and column of the array. The details of this procedure have been discussed in the section on the voting experts.

The "parameter-array-set" is an array which contains the probabilistic representation for a single class as well as other class specific information. This array and its components are shown in Figures 5 and 6. The probabilistic representation consists of three arrays, two which are called "parameter-arrays", and one which is called the "occupancy-array". The "on" parameter-array contains the $p_{i,l,1}$ parameters which weight the evidence in favor of having the temperature set at comfort levels, and the "off" parameter array contains the $p_{i,l,2}$ parameters which weight the evidence in favor of having the temperature set at levels which will conserve energy. The "occupancy-array" contains the $p'_{j,k}$ parameters which represent the a priori probabilities of the zone needing heat at different times. In the leader of the occupancy-array, all of the special information for the class is stored. This includes the weight factor, the user specified relative cost, the last change and damping for the relative cost, and a cumulative record of the cold errors and waste errors that have occurred each time the class has been used. The squares of these errors are also stored for later computation of the standard deviations, along with the maximum allowable cold errors and waste errors and the maximum distance score to allow proper acceptability testing. When the class has actually

PARAMETER-ARRAY-SET:



OCCUPANCY-ARRAY:

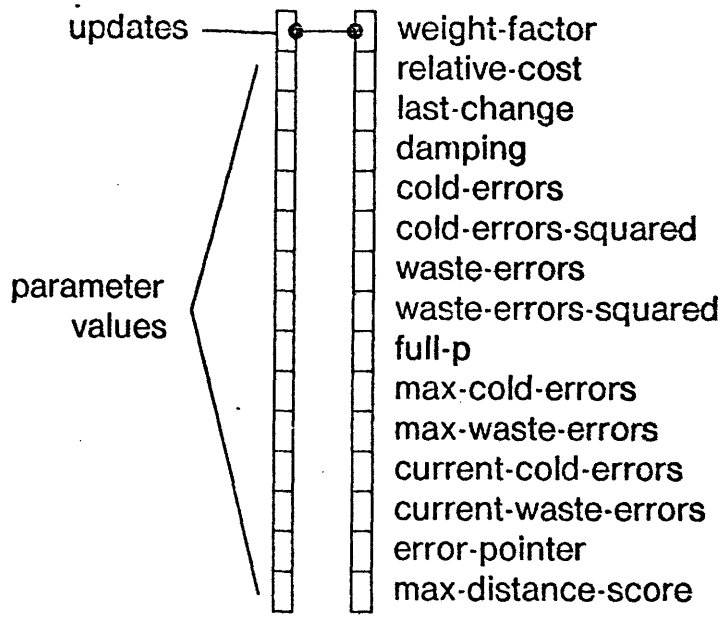
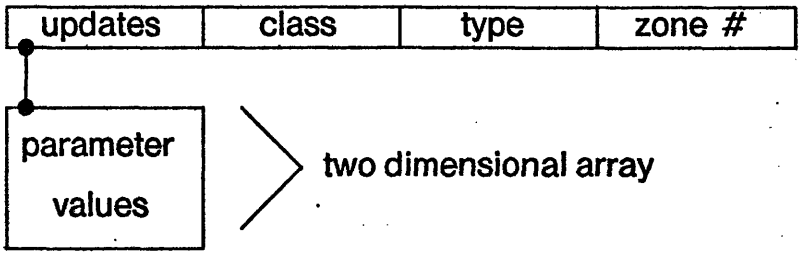


FIGURE 5: CLASS DEFINING DATA

PARAMETER-ARRAY:



VOTING-ARRAY:

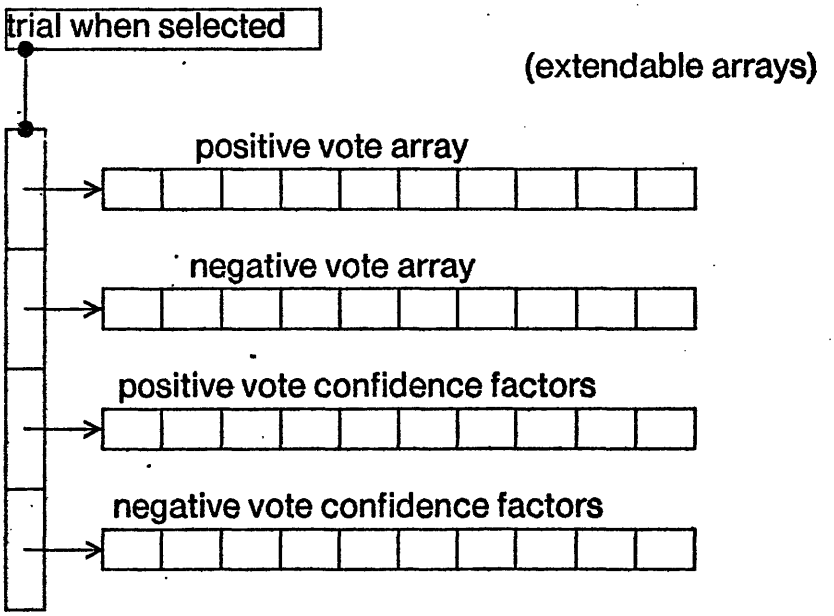


FIGURE 6: ADDITIONAL CLASS DEFINING INFORMATION

been selected on a particular day, the current cold errors and waste errors are stored and updated on each time slice, and an index of how many time slices have been processed in the evaluation of the current errors is maintained. A "full-p" flag is stored to tell the class augmenting procedures that the class has had its maximum allowable number of updates, and that the modified updating procedure must be used hereafter.

Additional structures stored in the parameter-array-set are the "temperature-control-array", the "voting-array", and the "log-parameter-arrays". The temperature-control-array stores a record of the output that the decision function would have produced had the associated class been used on the current day segment. This information is used by the intermediate self evaluation function and the final classification function to compare the classes on the basis of their performance even if they were not actually used.

The "voting-array" is composed of an array of four one dimensional arrays. These one dimensional arrays are used to store the votes that are made by each of the experts of the class prediction system along with the confidence factors for each vote. One of these arrays is used to save all the votes in favor of the associated class, and one is for all the votes against the associated class. The third array stores reliability factors for each of the experts should they make a positive vote, and the fourth array stores reliability factors for each of the experts should they make a negative vote. Note that independent reliability factors are used for the positive and negative votes for each expert for each class within each day segment. The leader of the voting-array is used to keep track of when a class was actually used by the decision function during a given day segment. This is primarily needed to insure that a class does not have to be reconsidered once it has been rejected due to excessive errors.

For the sake of increasing computational speed, the parameter-array-set also stores logarithmic representations of the parameter arrays. These are the "log-parameter-array-pairs". The logarithmic representations are composed within two pairs of arrays. One pair is derived from the "on" parameter-array, and the other is derived from the "off" parameter-

array. As previously stated, the arrays of "on" parameters and "off" parameters along with the time occupancy parameters are sufficient for use in the maximum likelihood decision algorithm. However, multiplying these parameters together as required by the algorithm can easily produce floating point numbers whose exponents are too large. To get around this problem, the logarithms of the parameters are computed and added together. Although only the $p_{i,l,j}$ parameters and not the $q_{i,l,j}$ parameters need to be stored in the parameter arrays, the $q_{i,l,j}$ terms must be determined in order to perform the maximum likelihood computation. When the parameter arrays are put into their logarithmic form, it is necessary to save the logarithms of both the $p_{i,l,j}$ and the $q_{i,l,j}$ parameters since it is not easy to get one from the other once they are in logarithmic form. This is why two pairs of arrays are used in the logarithmic representation. The first element of each pair stores the logarithm of the $p_{i,l,j}$ terms, and the second element of each pair stores the logarithm of the $q_{i,l,j}$ terms. The advantage of saving the logarithmic representation of a class is that the logarithms, which are essential in the maximum likelihood computation, only need to be recomputed when the class they represent is augmented with new data.

9.2 Memory Requirements

From a practical viewpoint, we may wish to consider the amount of memory that this data base might require. The main consumers of memory will be those sub-structures which occur frequently, and which store many floating point numbers. The primary consumers of memory in the structures I have described are the parameter-arrays, the time-occupancy-arrays, and the relation-arrays. If each floating point number consumes one word of memory, then the total number of words used for the parameter-arrays may be determined by multiplying the size of a single parameter-array by two since there are two parameter-arrays for each class, and then by the total number of classes per class-frame, the total number of segments per day, and finally, the total number of zones. In a typical application, the day might be divided into four segments, and 48 half hour time slices. There might typically be four zones, and we

may limit the total number of classes within each class-frame to four as well. Since the size of a parameter-array will then be 4×48, or 192 words, the total memory used by all of the parameter arrays will be: $192 \times 2 \times 4 \times 4 \times 4 = 24,576$ words.

The memory used by the time-occupancy-arrays is determined in a similar manner from the size of each array multiplied by the number of classes per zone, the number of segments, and the number of zones. In a typical case, this will be: $48 \times 4 \times 4 \times 4 = 3,072$ words.

The memory used by the relation-arrays is equal to the size of the relation-arrays times the number of experts, the number of segments, and the number of zones. With 5 by 5 relation arrays, and four experts, this will be: $25 \times 4 \times 4 \times 4 = 1,600$ words. Thus, the total memory used for these structures will be 29,248 words. The actual memory required by the data base will of course be larger, but this figure gives a reasonable approximation of what will be needed.

10. THE CONTROL STRUCTURE

The control structure of the simulator is intended to make the performance as realistic as possible. In particular, even though the raw data must be supplied in advance in order to run a simulation of many consecutive days, the program must deal with this data as if it were receiving information one time slice at a time. Much care must be taken to be sure that the program only accesses data that would normally be available in a real situation. To do this properly, it is necessary that certain special operations be performed at different phases of the day.

There are three main control phases which occur during each of the four segments of a day. In the first phase, the system uses an initial prediction of which class to use for temperature regulation, but it has no way to evaluate its performance. Next, the system will reach a point where it has sufficient information to evaluate and classify the behavior patterns of the previous segment. This phase involves the majority of the bookkeeping procedures which add to the system's knowledge of past human behavior and its own performance. The third phase involves self evaluation at each time step to verify that the correct classes are being used in the decision process.

In the first phase, which is at the start of each segment of the day, there are some special tasks which must be performed only at the beginning of a new day. In particular, there is some processing that needs to be done with the date of the new day. The actual date may be obtained either from the raw data itself, or from an internal calendar function which can provide the correct date if the date of the previous day is known. The date of a new day must be compared with all the dates in the special-days array to determine all of the special day names that are associated with that date. Whenever there is a match between the current date and a date stored in the special-days array, a number representing the special day name associated with the current date is placed on the current-day-names array. Before this is done, however, the contents of the current-day-names array is saved onto the saved-day-

names array because the special day names from the previous day will be needed later. The current-day-names array will then be loaded with a number for each of the special names that are associated with the current date. Each number is actually a reference to the special-days array, so there is a one to one correspondence between each number and each name.

If the new day is also the very first day in the sequence, a structure called the heat-on-array will be generated from the occupancy data of the new day. This array contains the ideal control pattern that the decision algorithm should produce. It indicates whether the temperature setting in each zone should be turned up or down for the duration of each time slice. Recall that the temperature should be set up during and before periods of occupation, and set down all other times. The number of time slices for which the temperature must be set up before a zone is occupied is determined from the heat time constant. The heat-on-array is used by the evaluation functions that determine how many errors the decision algorithm is making at any given time. Since this array contains advance knowledge of what the predictor should be doing, access to it is controlled carefully to insure that only information which would be available to a real time system is used in the simulator.

Regardless of whether a segment is at the beginning or in the middle of a day, there are several tasks which are always performed during the first phase. Before any temperature control decisions can be made, the class prediction algorithm must choose the pattern classes that will be used in each zone during the new segment. The class prediction algorithm will need to know the classifications for the patterns in each zone during the previous segment, but this information will not be available yet. Because of this, the prediction system will have to assume that the last classes that were used for prediction in the previous segment will be the same as the final classifications. If this assumption is incorrect, it will have to be rectified in the second phase. With all of the pattern classes for the new segment predicted, the system will proceed to use these classes in the decision algorithm to specify the temperature setting for the next time slice. This will continue until the duration of the heating time constant has

passed since the beginning of the segment. Once this time has elapsed, it will be possible to accurately evaluate the performance of the system during the previous segment.

When the system's performance during the previous segment can be evaluated, the second phase of operation will begin. This evaluation cannot be done earlier because the first few time slices of the current segment are needed to determine what the proper control decisions should have been at the end of the previous segment. As mentioned earlier, this is due to the delaying effect of the heat time constant. The second phase thus begins with the final classification of the patterns for each zone of the previous segment. The errors that would have been produced by each existing class for a zone are compared with the limiting values. The distances between each class and the new pattern are also compared with the current maximums. If both of these acceptability tests should fail for a given zone, a new class is established for that zone using the new pattern. Otherwise, class whose parameters are most similar to those of the new pattern will be augmented with the parameters from that pattern.

When an existing class is augmented with a new pattern, there are several bookkeeping procedures that are invoked. First, the maximum distance score of the selected class must be increased if the distance between the class and the new pattern is greater than the previous maximum. Then, the selected class is augmented with the parameters of the new day segment, and the weight factor for the class is adjusted according to the number and type of errors that would have been made by the class had it been applied throughout the entire segment. This factor is changed so as to help maintain the relative cost ratio specified by the user. Should the final classification be different from the class that was originally predicted at the beginning of the previous segment, the weight factor of that class which was predicted will also be adjusted. This is to reduce the amount of harm that can be done by a bad prediction, although it reduces the performance quality of correct predictions as a consequence. The certainty factors associated with each of the voting experts in the class prediction algorithm will be updated for each of the classes according to the correctness of each vote that was

made. Other historical details such as the average error scores, and the squared error scores for the augmented class are also updated at this time. Also, the name of the selected class is added to the class sequence list and the class-relation arrays of the corresponding zone are updated. Since this is the key information that was missing in the first phase, it is possible that the class prediction algorithm will need to correct its prediction of the current class. This will only be necessary however, if any of the classes that were actually in use at the end of the previous segment were not the same as those obtained in the final classification process.

If a new class is created during the second phase, then most of the above procedures will not be needed. The maximum distance score is initialized at zero and the weight factor is also initialized at zero. The weight factor of the class that had been predicted will not be disturbed and the certainty factors for the experts in the class prediction algorithm will remain unchanged because there was no way for the correct class to have been predicted. The error scores for the new class cannot be computed, but the initial error bounds may be computed from the occupancy pattern that formed the new class. It will also be possible to update the class sequence list and the class-relation arrays at this time.

In the third phase of operation, the system will continue to control the temperature setting at the beginning of each time slice, but it will now be able to evaluate its intermediate performance as well. If the system is on the first segment of a new day, then the heat-on-array from the previous day may be discarded, and a new one must be generated for the new day. This array will be used to evaluate performance throughout the entire day. Intermediate performance evaluation is intended to insure that the current classes being used for the decision algorithm are valid in the sense that they are not producing an unreasonable number of errors. The acceptable level of errors is determined from a function which scales the maximum error limit for an entire segment according to the actual proportion of the segment that has elapsed. If a class is found to be producing an excessively large number of errors,

the system will look for a class that would have had fewer errors in the same situation and begin using that class instead of the one that was originally selected. This process continues until the temperature has been set for the last time slice of the current day segment. When the next segment begins, the system returns to the first phase of operation, and the class prediction algorithm chooses a new set of parameter array classes for temperature control.

The control structure is implemented with several levels of nested loops. The topmost loop steps down the list of arrays containing the occupancy data for individual days. The next level steps through the four segments of each day, and the third level steps through the time slices of the current day segment. The lowest level loop executes the processing for each zone one at a time. The first three loops are needed to divide time into periods that require similar processing, and the innermost loop is needed to deal with the independent processing required by each zone. The three phases of operation are implemented by a conditional statements within the innermost loop that branch to various operations according to the state of the different loops.

11. HUMAN INTERFACE

One of the primary goals of my system is to provide a very simple and understandable human interface which will permit effective performance without a great deal of effort on the user's part. This is currently a problem with timed thermostats since performance is directly related to the accuracy and detail provided by the timer settings, yet the user is required to provide this accuracy and detail himself. For instance, if a timer is used that only has day and night settings, and no facility for handling different days of the week, then the system will perform poorly if the user's behavior changes on different days of the week. This performance can be improved if the timer allows different settings for each day of the week, but significantly more effort is required on the part of the user. The added versatility can therefore result in improved performance only if the user is willing to spend the extra time and effort needed to obtain reasonable settings.

Because of the increase in complexity of timed thermostats as they make more provisions for improved performance, much effort has been spent to make these devices as simple to use as possible. There are several important features that are employed to simplify user interaction. First, it is desirable to minimize the total number of buttons while still making the function of each button very clear to the user. For instance, when it is necessary for the user to enter a temperature setting, rather than having ten buttons for a decimal entry, two buttons can be used. Pushing one button can cause a displayed setting to be increased, and pushing the other button can cause the displayed setting to be decreased. If either button is held down for more than a brief moment, the setting can continue to change in the desired direction until the button is released. It is even possible to have the rate of change increase as the button is held down for a longer period of time since this implies that a large change in the setting is desired.

This simple entry method need not be used only for numerical entry. Other uses may be for selecting various features or options. The two button system can be used to search

through a list of menu items, with a third button to allow the user to indicate when the desired item has been found. Menus provide a very simple and yet versatile user interface. A menu system can allow the selection between an almost unlimited set of options, all with a very limited set of buttons. The only problem is that as the set of options gets larger, the time required to find the desired option will also get larger if the number of buttons is not increased. Additional buttons can be added in a useful way by permitting the user to select between mutually exclusive items at one time. In other words, it is possible to allow the user to first select a set of options that he is concerned with, and then allow him to select an item within that set by providing a button for each item. The advantage of this is that the same buttons are used for each set of options, so the user is essentially just changing the labels of these buttons along with their effective function.

By automatically dealing with the problem of deciding what the temperature should be at different times of the day, the interface requirements of my system can be tremendously simplified over a comparable timed thermostat. My system also permits an interface which is much more responsive to the way people think about their heating or cooling needs. The system is intended to deal with people on a very fundamental level by allowing them to directly express their desire for comfort and their desire to save money. These two factors, after all, are the primary issues of concern when considering any heating or cooling system. People may find it hard to anticipate their own behavior patterns, but they should have no problem deciding when they are unhappy with the comfort their system provides, or the amount of energy it is consuming.

There are several interface features that my system will require. I will first discuss what these features are and why they are needed, and then I will discuss how these features might best be implemented in a properly human engineered device. There are seven different types of information that a user may provide to my system to customize it to his personal needs. This information is listed below:

1. The normal comfort temperature
2. The desired setback temperature when a zone is unoccupied
3. An indication of the user's desire to save money over his need for comfort
4. Names of special days and their corresponding dates
5. The desired temperature for night time setback
6. Times for temperature setback at night
7. A setting to force the system into a standard thermostat mode

The normal comfort temperature of a zone is simply the temperature that the user wants to have in that zone whenever he is present. This corresponds to the temperature setting on most standard thermostats. It is important to provide some numerical scale for the temperature setting even though all the user really thinks about is whether he is hot or cold because there is always the possibility that one user might undo a change that had just been made by another user. The user who made the first change might then get upset at the system because the change he made never took effect, not knowing that someone else had tampered with his setting. If a user makes a choice on a fixed temperature scale, he can easily see if his setting has been changed by someone else, and the users can then deal with the problem of conflicting comfort requirements between themselves.

There is always a disadvantage to having users set their desired temperature directly since some people often turn the setting to an extreme when they are uncomfortable, expecting to obtain their desired temperature much faster that way. This usually does not provide comfort any faster than if the correct setting had been made since most heating or cooling systems will work at full capacity until the setting has been reached, regardless of how far the setting is from the actual temperature. Instead, some additional discomfort and waste will result because the actual temperature will often go beyond the desired set point before the user decides to turn it back to normal. With a properly engineered interface, this behavior can be discouraged while still retaining the advantages of allowing direct temperature setting.

The desired setback temperature for a zone when it is unoccupied should also be under direct user control. This temperature setting will be used whenever the decision function determines that the temperature should be turned down in an unoccupied zone. It is important

for this parameter to be under user control since the user may want strict limits on the amount of temperature variation allowed in certain zones, or, the user may wish to limit the amount of discomfort he will experience should he walk into a zone when he was not expected.

The user's desire to save money is a very important quantity for the system to know. Since very few individuals are completely predictable, there will always have to be a balance between mistakes that cause discomfort, and mistakes that cause energy to be wasted. If the system maintained comfortable temperatures all the time, then the user would not be uncomfortable, but no money would be saved. On the other hand, if the system regulated temperature only when it sensed the presence of people, a great deal of money could be saved, but users would have to be uncomfortable each time they entered an unoccupied room. My system is capable of finding a balance between these two extremes, but exactly where that balance should be depends entirely upon the user's preferences. By allowing the user to express his desire to save money, with the implicit understanding that saving money may sacrifice comfort, the system should be able to find a level of performance which maximizes the user's overall satisfaction. This must be done by interpreting the user's preferences into the relative cost factor.

A facility which allows users to inform the system about any unusual days that can be anticipated in advance will help the system deal with irregular variations in their behavior. To maintain all the aspects of very simple interaction that have been developed thus far, it is important that this facility does not require the user to provide an excessive amount of information. For instance, it would be inappropriate to require the user to tell the system exactly what will happen on an unusual day since this would place an undesirable burden on the user, and on top of that, the user might not even know what to expect on that day. If instead, the user is allowed to give a particular type of unusual day any name he likes, such as "vacation" or "holiday", and is then asked to provide only the dates on which this day type will occur, the system should be able to build an association between the day type, and

the behavior patterns which have occurred on the corresponding dates.

A special temperature setting for night time setback must be provided because people generally need less heat when they are sleeping. The prediction system I use would not be useful for turning the temperature down when people are asleep because it has no way to tell when they are actually sleeping. Since the system senses the presence of people, it knows that it must provide a comfortable temperature. If the system were to operate as if no one was present, it could allow the temperature to reach levels that are uncomfortable even for a sleeping person. What is needed therefore, is a special temperature setting just for sleeping people. This setting is used instead of the normal setting at certain hours of the day. Other zones which are unoccupied when people are sleeping will have their temperatures turned down to their unoccupied settings. These settings may deviate from comfortable temperatures much more than the sleep setting allowing much greater savings

With a special sleep temperature setting, it will be necessary for the user to be able to inform the system when to use this setting. This information can be provided with simple start and stop times similar to those used in standard timed thermostats. This timing will not be so crucial as to cause difficulty to the user since a sleep temperature setting is usually not so different from normal that it would cause great discomfort to a person who was not actually sleeping. Also, if a person is sleeping at unusual hours, the temperature will be kept at the normal level, causing a slight loss of energy, but no loss in comfort.

An alternative to having the user input time settings for sleeping hours is to have a bed sensor which detects when someone is in bed. This sensor could be used to turn the temperature to the sleep setting whenever someone was actually sleeping. The only problem with this is that there is a good chance that the sensor could make mistakes if people are only sitting on the bed or if heavy objects are placed on the bed.

The feature for allowing the user to switch the mode of the system from the energy conserving predictive mode to the standard thermostat mode is merely a failsafe to insure

that the user is not completely at the mercy of the automatic mechanism. This facility may be especially useful when the system is first installed, because when the system is first learning new behavior patterns, it is likely to make many mistakes. Rather than being forced to endure these mistakes, a user may prefer to simply have a standard thermostat until the performance has improved. An important implication of this is that the system will need to provide the user with some indication of how well it is performing. Without this indication, the user would have no way of knowing when it would be appropriate to switch from the standard thermostat mode to the predictive mode.

Below, in Figure 7, is an example of what a simple interface might look like for my system. It has an alpha-numeric display for indicating temperature settings, the actual temperature, the current mode of the system, and various prompts for special day input.

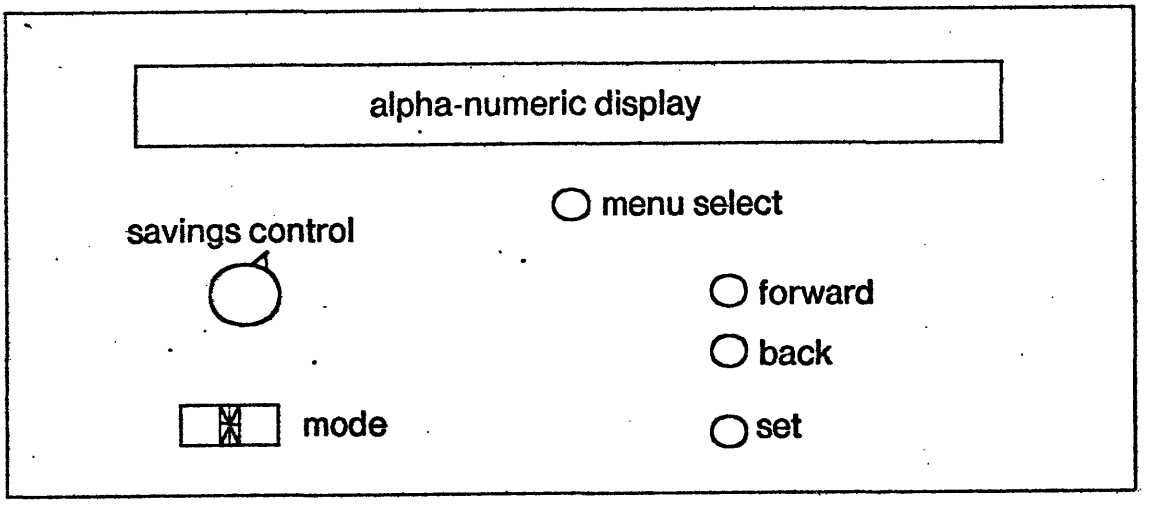


FIGURE 7: THE SIMPLE USER INTERFACE

The "savings control" dial will allow the user to directly input the relative cost factor. The user, however, need not have any understanding of how the relative cost factor is used to be able to operate this dial properly. All the user really needs to know is that turning the knob one way will guarantee comfort with the performance of a standard thermostat, and

turning it the other way will minimize expenses with the performance of a people sensing thermostat. It should then be easy for the user to select a point between these two extremes that will give him the greatest satisfaction.

The temperature settings may be changed by the "forward" and "back" buttons. A means to select which of the three possible settings are being adjusted is provided by the "select" button. The "forward" and "back" buttons will be for increasing or decreasing the selected temperature setting. When the "forward" button is pushed, the setting in the display will increase by just one degree. If the button is held down, the setting will begin to increase automatically until the button is released. The "back" button will work the same way except that it will cause the displayed temperature to decrease. A short time after either of the buttons are released, the choice will become permanent. The problem with users who set the temperature far beyond their actual needs in hopes of making the system respond faster to their request can be slightly reduced if an indicator is provided to inform the user that the system is operating at full capacity.

There are several ways that the user might be able to input special day names and their corresponding dates. One way is to provide alpha-numeric entry by slewing through the alphabet one letter at a time, using the "forward" and "back" buttons to move forward and backward through the alphabet, and the "set" button to indicate when the correct letter has been found. The "set" button may be pressed twice to indicate that the name is complete. It might be simpler though, if the system merely had numbers for each special day, and the user wrote the names that correspond to these numbers on an external note pad. The special days are common to all zones, and therefore only need to be entered once if each interface node shares its information with the others.

A good way for dates of the special days to be entered is also needed. On occasion, the user will need to see all the dates that correspond to a special day name. The entry of dates can be performed by having the system ask for month, day, and year one at a time. The

two button method discussed for temperature entry can be used here to enter numbers that correspond to month, day, and year. The system should be able to check the validity of any entry against a calendar, and ask the user to try again if there is an error. The user should be able to easily specify things like "every monday", or "July 4, any year", or "The first of every month" without having to make entries for each exact date that correspond to these statements.

12. A THERMAL MODEL

The results I have obtained are difficult to evaluate in their raw form. These results show where the prediction algorithm does well, and where it makes mistakes, but they do not provide a clear indication of the amount of savings the system can provide, or the amount of discomfort that the user will experience when mistakes are made. To obtain this information, we must estimate what the temperatures in the house will be at different times, and determine how long the user must withstand uncomfortable temperatures, and how long the room temperatures can be maintained at a low level to save energy.

We can approximate the amount of discomfort a person experiences at any given time as being proportional to the difference in the actual temperature and the desired temperature. Of course, this is not entirely accurate since the real discomfort people experience is highly dependent on what they are doing at the time. Physical activity and clothing play a very important part in the comfort of an individual at different temperature levels, but the available information does not allow these factors to be taken into account. People are also known to be easily satisfied within a range of room temperatures, but it is simplest to consider any deviation from the setpoint as a comfort error since small deviations within the comfort region will not figure significantly into this computation anyway [Egan,1975]. Using this approximation, the total discomfort experienced in a given day can be determined from the cumulative degree-minutes of discomfort. This would be equivalent to the difference between the actual temperature and the desired temperature in a zone integrated over the periods that the zone is occupied.

To do a perfect job of estimating energy savings, it would be necessary to carefully model the house and its surroundings, taking into account all sources of heat generation and heat loss, hourly changes in weather conditions, and all of the thermal properties of the structure under consideration. Since facilities and data for such a simulation are not available, I have had to simplify my simulation tremendously. To make sure my simplification is reasonable,

I have tried to make assumptions that agree fairly well with results that have been obtained from more detailed simulations.

Using graphic results from Honeywell simulations for the thermal properties of a house, I have been able to develop a simple program which can approximate these results fairly accurately [Nelson and MacArthur,78]. Using the same test conditions as they did, I was able to obtain the result that day-night setback achieves 2.1 times the savings of night setback alone. Since this agrees very well with the result they obtained with a much more detailed simulation, I believe my simulation will produce valid results for tests using different setback times determined by the prediction algorithm.

A special complication that makes my simulation different from the Honeywell model is the fact that I have divided the house into zones. The use of a zoned house requires that special care be taken when accounting for errors. First, to make the savings results comparable to those for a single zone building, the savings for each zone must be averaged. Also, we must be careful not to credit a person with more degree minutes of discomfort than he actually would experience. This can happen if a person spends time in several zones during a single time slice. If someone spends more than a given amount of time in a zone, they are counted as having occupied that zone during the entire period. If a person spends 10 minutes in a zone, he will be considered to have been in that zone for 30 minutes since that is the length of the time period. It would not be correct, however, to credit a person who spent 10 minutes in each of three zones that were each 5 degrees too cold with a total of 5×30 degree-minutes of discomfort (450 degree-minutes), when in reality the person only experienced 150 degree-minutes of discomfort. The solution to this problem is to divide the total degree-minutes accumulated by a person for a given period by the number of zones that they occupied during that period.

Special consideration also needs to be made to account for the fact that a multiple zoned system cannot have complete thermal isolation between zones. The simple model I have used

does not account for this because it was derived from a single zone system. To get around this problem and still be able to produce reasonable results, I study the performance of the system for the two extremes of coupling between zones. If we assume zero coupling between zones, this would be equivalent to having perfect insulation between zones. This is clearly unrealistic, but it gives a lower bound on the amount of energy that will be used for a given heating pattern. If we assume complete coupling between zones, this would be equivalent to having no insulation at all between zones. This too is unrealistic, but it gives an upper bound on the amount of energy that will be used. When we apply the complete coupling assumption, we essentially transform a multi-zoned system back into a single zone system since heating any single zone will force all of the others to be heated as well.

12.1 Details of the Model

To construct a simple model of the thermal properties of a house, I formulated a set heuristic rules for temperature variations based on the simple equivalent electric circuit model shown in Figure 8, and data from a detailed model that was studied at Honeywell. The circuit model I used consists of two interconnected capacitors, C_a, and C_w. Capacitor C_a represented the heat capacity of the internal air of the house, and C_w represented the heat capacity of the outer walls. These capacitors were connected to resistors R_1, R_2, and R_3. Resistor R_1 connects C_a to C_w, and represents the thermal conductivity between the indoor air and the wall. Resistor R_2 connects C_a to a node which represents the outside air temperature. This resistor represents the direct thermal conductivity between the inside air and the outside air which might result from leakage such as the airflow through cracks and windows. Resistor R_3 connects C_w to the outside air node. This represents the thermal conductivity between the wall and the outside air.

In this circuit model, voltages at the different nodes represent temperatures, and currents represent heat flow. The outside air is modeled as a voltage source, which in this model was held constant to simulate a constant outdoor temperature. For simplicity, the heat input is

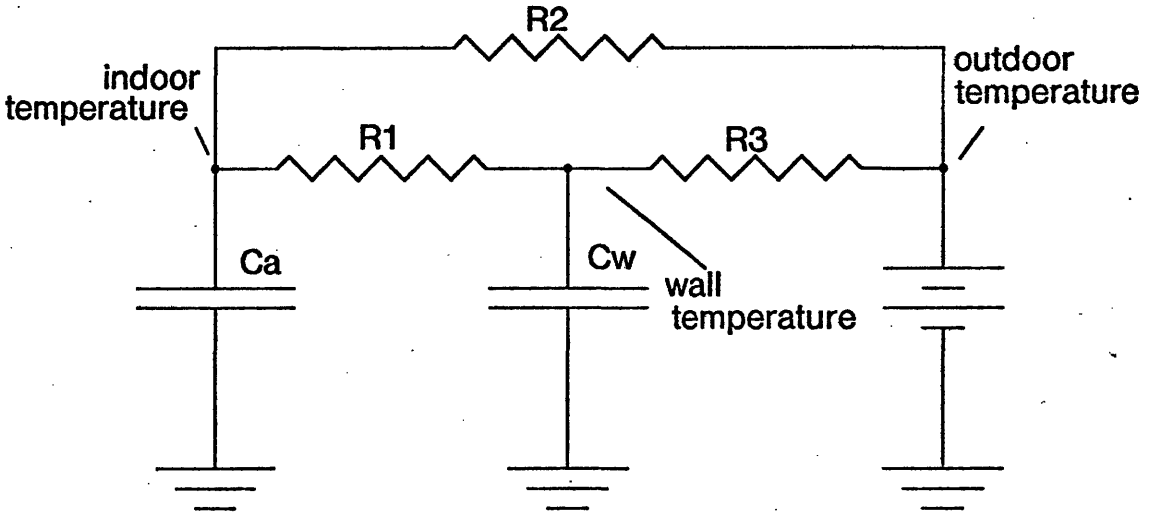


FIGURE 8: ELECTRIC EQUIVALENT CIRCUIT FOR THE THERMAL MODEL

modeled as a ramp voltage input at the node for the inside air. A constant current input would actually be more accurate here, but the simplification is close enough to real results that it is acceptable.

Since we may expect the leakage loss through R_2 to be negligible, it is reasonable to approximate the overall heat loss to be proportional to the current flowing through R_3 which is equivalent to the heat flow from the outer wall to the outside air. This means that the heat loss should be proportional to the temperature differential between the outer wall and the outside air. This observation is confirmed fairly well by results from Honeywell simulations. The model also tells us that for any given inside air temperature, the steady state wall temperature should be specified by the voltage divider formed by resistors R_1, and R_3. As a result, the steady state wall temperature is specified by the equation:

$$\text{wall temperature} = \frac{R_1 \times R_3}{R_1 + R_3} \times (\text{inside temp} - \text{outside temp}) + \text{outside temp}$$

In studying the simple circuit model in conjunction with the graphic results from more detailed simulations, three time constants are suggested. One time constant is needed for the rise of the wall temperature when the indoor air temperature has been increased by the heating system, one time constant is needed for the fall in air temperature towards the wall temperature once the heater has been turned off, and one time constant is needed for the fall of the wall temperature when the air temperature is falling. When the wall temperature is rising or falling, it should exponentially approach the steady state temperature specified by the indoor and outdoor temperatures. The falling rate, however, should be slightly slower than the rising rate due to the additional heat capacity of the indoor air. This leads to the first two time constants. If the wall temperature could somehow be held constant, the indoor air temperature would be seen to fall exponentially to that temperature. This leads to the third time constant. As stated earlier, the indoor air temperature is approximated as rising at a constant rate when the heat is on so there is no need for a fourth time constant.

Temperature changes are calculated using these exponential approximations over short

time steps. Since the time constant for the falling air temperature is considerably shorter than the time constant for the falling wall temperature, the wall temperature is first approximated as being constant during the time step, and the change in air temperature is calculated using the formula:

$$\text{new temp} = \text{current temp} + (\text{objective temp} - \text{current temp})(1 - \exp(-\frac{t}{\tau}))$$

In this formula, the objective temperature for the indoor air is equal to the wall temperature. Once the new air temperature is calculated, the average air temperature is approximated by taking the average of the old and new air temperatures. This average is then used to determine the objective wall temperature through the equation for the steady state wall temperature. The new wall temperature may then be calculated using the same formula above with a different value of τ to correspond to the time constant for cooling the wall.

Some special care must be taken when the indoor temperature reaches either the upper or lower set points. If the above calculations should have the indoor temperature exceeding either of these limits within a single time step, then some compensation must be made to eliminate this overshoot. In my model, if the calculated new temperature for the indoor air should exceed a set point, it is reset to that set point to simulate a change in action of the heater. The only problem with this is that it complicates the calculation of the average room temperature. The average of the old temperature and the new temperature is no longer a valid approximation because there will be a period during the time slice where the temperature had actually been constant. To correct for this problem, the average must be computed in two sections of time. The first section will be the time when the temperature is changing, and the second section will be the time when the temperature is held constant at the set point. The average temperature for the first section will be the average of the old temperature and the set point temperature. The average temperature for the second section will simply be the set point temperature itself. The final average must be determined from a weighted average of the first two averages. The weighting will be according to the relative durations of the

corresponding time sections.

As shown in Figure 9 below, the indoor air temperature was rising from the old temperature to some new temperature, until time t_1 when the upper set point was reached. Then, from time t_1 to t_2, the temperature remained at the set point. The length of the time interval t_0 to t_1 relative to the interval t_0 to t_2 may be approximated by the relative temperature difference between the set point and the old temp, and the new temp and the old temp. This will provide the proper weighting to obtain the average temperature during the interval t_0 to t_2.

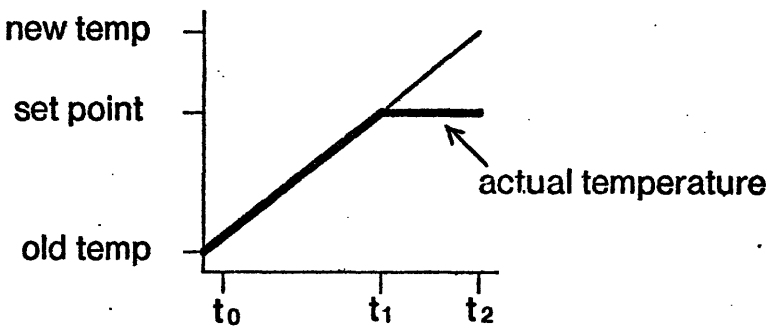


FIGURE 9: AVERAGE TEMPERATURE CALCULATION

The average temperature will be computed as follows:

$$\text{average temp} = \frac{T_{rel} \times \text{ave temp 1} + (1 - T_{rel}) \times \text{ave temp 2}}{2}$$

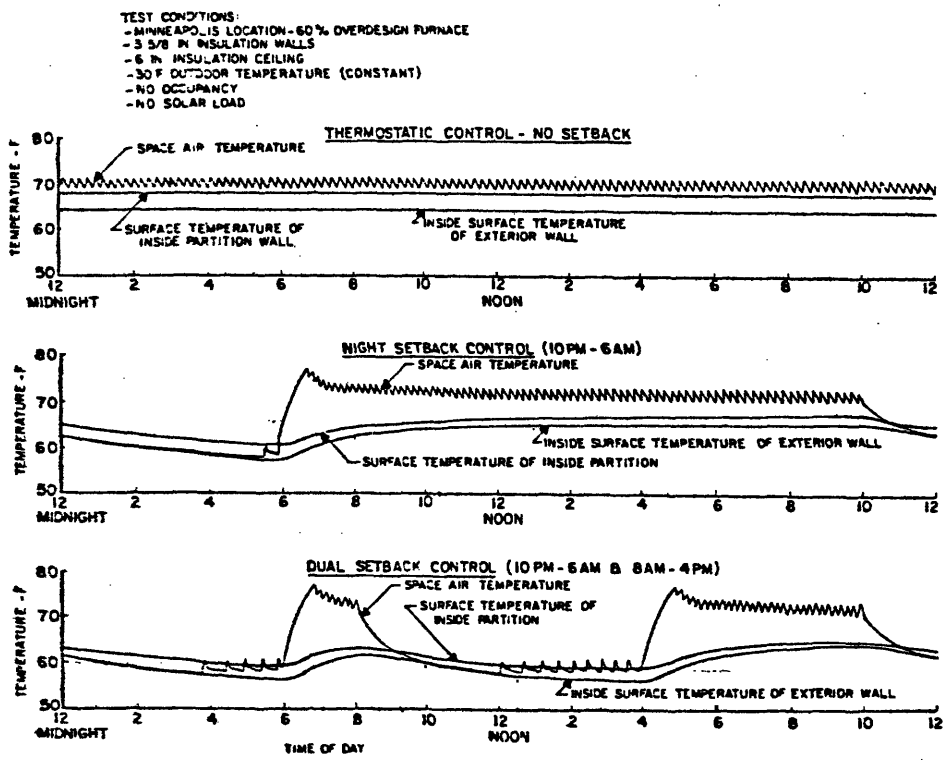
where:

$$\text{ave temp 1} = \frac{\text{set point} - \text{old temp}}{2}$$

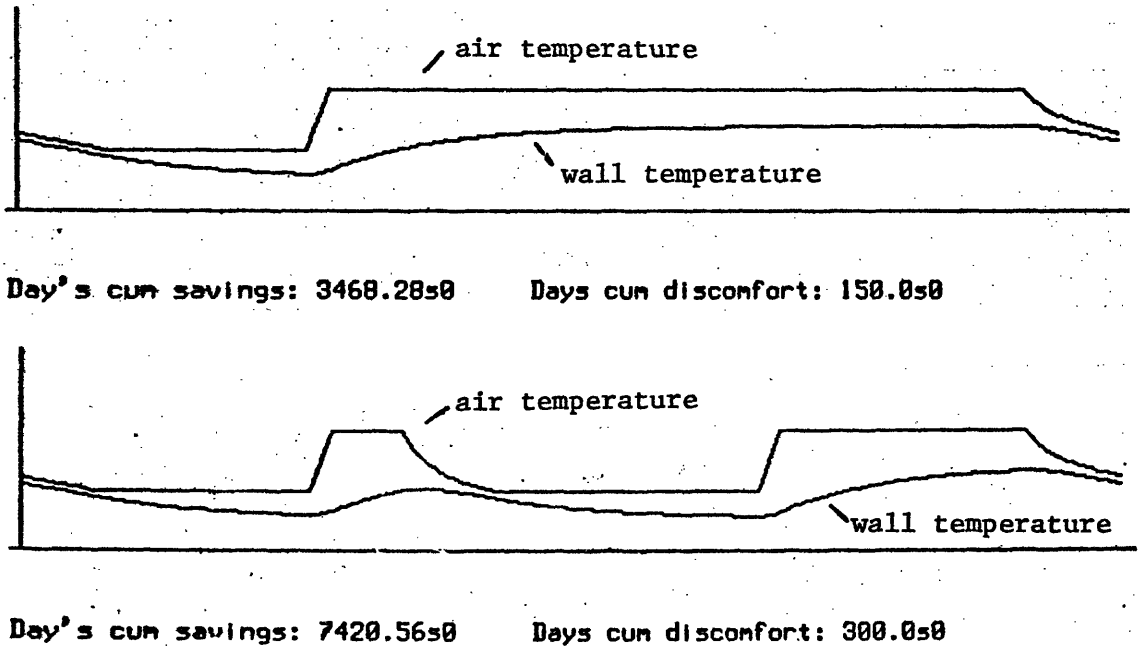
$$\text{ave temp 2} = \text{set point}$$

$$T_{rel} = \frac{\text{set point} - \text{old temp}}{\text{new temp} - \text{old temp}}$$

The graphs in Figure 10 show a comparison between my model, and the Honeywell model. They appear to be sufficiently similar to justify my model as a means to obtaining a reasonable approximation of the energy consumption and comfort levels for an arbitrary heating pattern.



Graphs from Honwywell's detailed thermal model of a house.



Graphs from my simplified model.

FIGURE 10: COMPARISON OF RESULTS FROM MY MODEL WITH THE HONEYWELL MODEL

Another test I have used to verify the validity of my model is to compare the relative energy savings that it predicts for different heating patterns with the savings that the Honeywell model predicts for the same patterns. As obtained in my model, 7420.56 degree-minutes of savings are obtained from day-night setback as opposed to only 3468.28 degree-minutes of savings from night setback alone. This means that day-night setback achieved 2.140 times the savings of night setback in my model. In the Honeywell model, 14.54% is saved by day-night setback, and 6.78% is saved by night setback alone. Thus, the Honeywell model shows that day-night setback saved 2.145 times the energy that was saved with only night setback. The agreement between these figures for relative savings is very convincing evidence that my model is reasonable. This data also provides a useful means for determining the percent of energy savings from the figures for savings in degree-minutes.

13. FUTURE DIRECTIONS

The predictive learning control system that has been described in this thesis has potential usefulness in many situations where pattern directed prediction is needed. There are also many modifications that can be made to the basic algorithm that should enhance performance in the temperature control application as well as many others. One of the keys to better performance should be the addition of more knowledge in the form of additional rules for the class prediction algorithm and additional pattern features for use by the decision function.

13.1 Improvements

In the control system that has been discussed thus far, predictions are always made on the basis of patterns, and these patterns are always of the form of past occupancy records from the zones within a building. It is important to recognize that the algorithm is not limited to only using this type of information as patterns, it is merely that this past behavior was thought to have a great deal of value for making predictions about future behavior. There are of course many other sources of information that might have a great deal of value in predicting future behavior. For example, the use of particular lights or appliances may be associated with certain common tasks that people perform. Knowledge of when these items are being used can provide helpful information to the system about the types of behavior to expect. Other useful information might be obtained from the status of various door locks within the building, as well as the open or closed state of doors, the settings of burglar alarms if there are any, and the use of various pieces of furniture such as beds and chairs.

If implemented properly, these features will serve the same purpose as occupancy information in the system's predictive function, except that unlike occupancy, the features themselves will only be sensed, and not predicted by the system. This is an important distinction because the system must use past occupancy data to predict what the proper temperature settings should be, but these settings are directly related to future occupancy. If other features are used to predict temperature settings, the settings will have no direct

relationship to the future states of those features. The system does not have to worry about when certain doors are likely to be open or closed, or when various appliances will be used, it merely notices when these things happen, and predicts occupancy behavior accordingly.

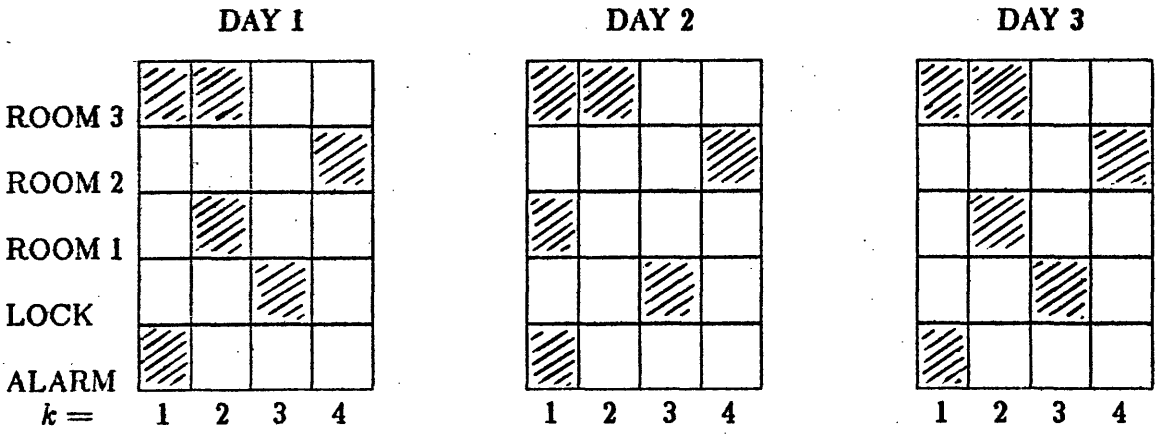
The inclusion of these extra pattern features in my system would actually be quite simple. In the basic system, predictions are made on the basis of an array of binary features. It happens that these features have been chosen to be associated directly with the occupancy of the different zones at different times with respect to the present. To include other features in this system, all that is needed is to add extra rows in the pattern array in the same way that we would add additional rows for new zones. Provided a feature is binary, the elements of a new row will contain the values of the feature at various times from the present just as the standard row contains the state of occupancy of its corresponding zone at various times from the present. It is important to note that in this format, a feature is used under the assumption that it has some time relational value in predicting behavior patterns. In other words, we expect any feature that is used to somehow be correlated with future behavior. If this is not true, no extreme harm will be done, but the system may take time to determine for itself that no relationship exists.

An example of a typical feature might be the state of a lock on a main door to a home. This feature would be highly correlated with future behavior since the door may only be locked when people are away or sleeping. This feature could be dealt with in just the same way that occupancy for a particular zone is handled. Just as we record when a zone was or was not occupied at certain times from the present, we would record when the lock was secured or open at certain times from the present.

Some special consideration might be required for certain features if full advantage is to be taken of their predictive value. An important example of this is in the use of alarm clock settings. An alarm clock setting should be very useful for predicting the beginning of normal morning behavior patterns since these patterns are often initiated by the activation of the

alarm. The reason that special consideration should be given an alarm setting is that the sounding of the alarm will result in immediate behavioral changes, so if the predictor reacts only to the actual sounding, then it may not have sufficient time to deal with the changes that will take place soon afterwards. For example, if a person always wakes up to an alarm and goes directly into the bathroom, then the system will not have sufficient advance warning to heat the bathroom before it is used. There is really no reason not to have the proper advance warning in this case, since the time the alarm would actually go off could be determined when the alarm was set. A properly chosen alarm clock feature will take advantage of the facts that behavior may be highly correlated with the sounding of the alarm, and that the exact time of the sounding may be known well in advance of its actual occurrence. A simple way to do this is to have a feature which signifies an alarm sounding one time constant before it actually occurs. This is not at all hard to do because of the advance knowledge of when the alarm will go off. It will allow the system to take full advantage of the correlation between the alarm and behavior patterns because it will know about the alarm before the corresponding behavior changes occur.

The feature arrays for three consecutive days for a house with three zones, a lock sensor, and an alarm clock monitor would appear as follows:



The rows for the different zones are the same as they were in an earlier example, with dark squares indicating times during the day when the zones where occupied. Two rows have been

added, one for the lock data, and one for the alarm data. In the row for the lock data, each dark square indicates a time when the lock was secured, and each light square indicates a time when the lock was open. In the row for the alarm data, there is a single dark square one time constant before each time that the alarm will go off. The arrays have been drawn this way to emphasize that dark squares in the alarm and lock rows are treated by the predictor in the nearly same way as the dark squares in the rows for the different zones. The only difference is that the predictor does not need to predict what will happen in the lock and alarm rows, it only uses this information to influence its prediction of what will happen in the other rows.

Another useful modification to the algorithm described in this thesis might be a facility for the system to choose between different temperature settings depending on the certainty of its predictions. This way, even if the predictor does not expect someone to be present, the temperature can be slightly increased just in case that prediction is wrong. This way, if the prediction is indeed wrong, people will not suffer as much discomfort as they would if the temperature had not been raised. If the prediction was incorrect, there would of course be some energy wasted from the temperature increase. Thus, the amount of alteration of the temperature from the value specified by the predictor must be dependent both on the reliability of the prediction, and the cost of a mistake to the user. One possible way to determine the reliability of a prediction is to look at an error score for the class that made the prediction. This score would have to be based on the past errors that the class has made in each time slice it was used.

The system I have proposed is designed to place fairly low requirements on the people sensing devices that are used. The sensors my system uses only need to detect the presence of people in various zones, and they do not have to detect how many people are in each zone, or who is in the different zones, or even the exact times that people move from one zone to another. It is conceivable, however, that sensors could be made to trace individuals, and

know exactly where they are at any given time. If, for example, each individual had his own personal transponder that continuously transmitted a unique code, it would not be difficult for sensors to pick up these signals and determine who was present by deciphering the code.

If this type of sensor system were used, many significant changes could be made in my system which would improve its performance, but it is likely that an entirely different approach might do better. The main modification to my system would be to have a separate predicting database for each individual. This way, overlapping behavior patterns which result from two or more people being in the same zone at the same time can be separated from one another, and the probabilistic patterns for each individual can be determined. Another modification to my system would be to permit personalization of all temperature settings. Since the system always knows where each individual is, and also is capable of making a prediction of where they each will be in the near future, the system should be able to make sure that the temperature in each zone is at a level that is most desirable to the particular occupants at any given time.

Sensors that can differentiate between individuals may allow the use of algorithms that are potentially more powerful than the one proposed in this thesis. If we know the sequence of zones that an individual goes through the course of a day, it is likely that we may be able to build up a repertoire of sub-sequences that the person uses at certain times of the day. For instance, the procedure of going from the bedroom to the bathroom, back to the bedroom, and then into the kitchen might be a common sub-sequence of room changes that a certain individual goes through on most mornings. A system which knows about this sub-sequence merely has to synchronize what it knows about the timing for each zone with the observed behavior to obtain a reasonable prediction of what the person will do next. This form of analysis is not possible when patterns are indistinguishably overlapping one another.

A very important area for future work on this system will be to actually construct a working prototype and test its performance in a real home. This is essential if the effectiveness

of the human interface is to be demonstrated, and is also very important because the evaluation criterion is very subjective. Only by having people use the system can we ascertain their ability to easily communicate their needs to the system, and only by having people live under the conditions created by the system can we determine their sensitivity to the inevitable errors that will be made.

In an actual implementation of my system, it would be very important to find a suitable people sensing device. This device would have to be able to sense the presence of people no matter what they are doing. This requirement poses a problem for some standard burglar alarm sensors since they commonly sense particular actions such as the interruption of a light beam or the disturbance of an ultrasonic or microwave field by people's motions. When people are not moving, the sensors may not be able to tell that anyone is present. This problem may be solved by having an array of different types of sensors ranging anywhere from simple microphones to sensitive scent detectors. Sensors must be selected which can complement the ability of others to detect the presence of a person. Ideally, in situations where one sensor will fail, another sensor should be able to take over.

A more appropriate approach would be to have an intelligent sensing system which uses certain basic knowledge about people to fill in gaps left by the sensors. This would include knowledge about the conservation of people, the restlessness of people, and the connectivity of zones. By conservation of people, I mean that people cannot simply disappear form a zone, they must physically move out through one of the available exits. Since sensors can easily detect this motion, It will be possible for the system to know that someone is in a zone even though it can't sense them simply because it hasn't seen them leave. A knowledge of the connectivity of the different zones can also be used in conjunction with this principle to permit the system to deduce that when it suddenly senses activity in a previously unoccupied zone, that a person must have come from one of the connected zones. This can be very useful information to one of the adjoining zones if it is trying to decide whether or not someone

has left. The fact that people cannot remain perfectly stationary or quiet for extremely long periods of time can also be useful to make sure the system does not easily mistake an empty zone for being occupied. If no activity is sensed for an extended period of time, the system can automatically assume that no one is present.

The key advantage to this approach is that it allows a system to be constructed from very inexpensive parts. Both the sensory and the computing equipment would cost very little. Inexpensive and unreliable sensors such as microphones could easily provide fairly reasonable performance. Also, the computing power needed for this problem would not be difficult to obtain since inexpensive microprocessors could handle all of the computational load. This is just another example of how money can be saved by taking full advantage of cheap computing resources.

13.2 Other Applications

My algorithm is almost directly applicable to controlling lighting just as it controls heating or cooling systems. In a lighting system, it is not sufficient to provide light only where people are at any exact moment and to switch lights the instant anyone moves form one zone to another. If this were done, people would become very annoyed with the constant flashing of lights whenever they moved anywhere. A good way to get around this problem is to anticipate where people will be in the near future, and have lights turned on for them before they get there. If the anticipation is incorrect, then some energy will be wasted, but on the average, energy will probably be saved since people often leave lights on in empty rooms only because they think they might be returning later. My system is perfectly suited for this problem, because it will not only provide the advance notice to allow lights to be turned on in the right zones, but it also allows the user to express his preference for lighting which tends to follow his motions closely and conserve a maximum amount of energy, or lighting which tends to be on more often than it is really needed, insuring comfortable movement between zones, but using more energy. The tradeoff here is the same as that in a heating or

cooling system. Errors which arise from a desire to conserve energy will cause discomfort, and errors which arise from a desire to maintain comfort will cause some energy to be wasted. If a person walks into a room where he was not expected, the only discomfort he will experience is that the lights will flash on. If, on the other hand, the no one ever shows up in a zone that is expected to be occupied, the lights will be on needlessly, and hence some energy will be wasted.

I believe that there are many other potential applications of the work that I have done. The overall concept of the system I have developed can be useful in many areas where pattern based prediction is needed. This is especially true if the details of the patterns are not known very well in advance, requiring learning to take place as the patterns are observed. It is likely that careful human analysis of patterns would work better in situations where it is possible, but in many situations, it is not practical for this to be done. My system takes a tremendous load away from the human users, with only a small sacrifice to performance. If the features provided by the zone occupancy data are changed, and the predictor output as well as the performance feedback are changed, the system takes on a very general character.

My learning system has four basic classes of inputs, and single class of output. One class of input is the pattern feature class, which in the heating control problem consists of all the occupancy information that is used for each prediction. Another class of input is the knowledge input which consists of the information that the user provides about special days. This is knowledge input because it lets the user communicate high level information to the system when certain unusual events may occur. A third class of input is a relative cost input. This is an input that lets the user specify the relative significance of the two types of errors the system can make. The fourth class of input is the performance input. This input lets the system know what it should have predicted so that it can evaluate its own performance, and learn from experience. The single output that the prediction system must produce is the prediction itself. In the thermostat implementation, that output also serves a control function

in the sense that when the system predicts that someone will be entering a zone within the next time constant, the temperature setting for that zone is changed appropriately.

The reason I have described the inputs and outputs of my system in terms of several general classes is only to point out that the particular details of what I have used as inputs and outputs for the thermostat implementation are not so important as the classes of inputs and outputs themselves. Other applications may require completely different pattern features than I have used, different types of knowledge, different means of determining the proper weighting of errors, and a different definition of what constitutes an error. Also, the predictive output may be used in a variety of different ways to either influence a decision or directly perform some control function.

In retrospect, the portion of the system that is used for class prediction is probably the most powerful component of the learning algorithm. The power of this algorithm comes from the fact that it can deal with high level knowledge in the same way that an expert system does, but it adds to this knowledge through learning as well. The knowledge in the system comes through the design of the voting experts. Each expert is capable of incorporating many forms of high level knowledge. In most of the experts I used, the knowledge was expressed in the form of rules for the kinds of correlations to look for. For example, the expert that looked for correlations between classes used on each day and the classes used a week earlier actually represented the knowledge that people tend to have weekly periodicities in their behavior. The advantage of this approach is that it allows the consideration of any such hypothesis since the value of each hypothesis is determined through learning, where learning is achieved by the appropriate automatic adjustment of the confidence factors for each expert. Furthermore, confidence factors are adjusted and combined in such a way that the result is fairly insensitive to the fact that some experts may not be independent. For example, if there are two identical experts in the system, their confidence factors will eventually be adjusted such that their combined vote has no more influence than one of them could have had if it

were alone. The extension of this learning approach to other applications might prove to be very interesting.

14. APPENDIX: A SAMPLE TEST

This section shows a sample of a typical simulation run from my system. At the beginning of a simulation run, the program is supplied with behavior pattern data, and a number which represents the thermal time constant in minutes for each zone. The behavior pattern data is supplied in the form of a list of two-dimensional binary arrays, each array containing the quantized occupancy history for a single day. The ones or zeros in a given row of one of these arrays represent occupancy or vacancy of a zone at different times of the day. In order to properly demonstrate the validity of my algorithm, it was essential that genuine behavior data be obtained, since any manufactured data would be certain to reflect the biases designed into this system.

The raw data for the simulation was obtained from an apartment with a single occupant, and this apartment was divided into four separate zones. The first zone was the living room area, the second zone was the kitchen and dining room area, the third zone was a dressing room and bathroom area, and the fourth zone was the bedroom. The subject recorded her movement throughout the apartment to an accuracy of plus or minus ten minutes. This was done by making marks on a time line that was constructed for each day. The raw data was then quantized into half hour time slices, with any zone which had an occupant for more than ten minutes credited with occupancy for the entire duration of the corresponding time slice. As a result of this quantization, it is not unusual to find several zones occupied by one person during a single time slice. This happens when the person is moving frequently from one zone to another, so it is proper to have each zone heated in such cases.

The only other information the system needs is a list of user defined special days and their corresponding dates. The system obtains this information through a special query routine. This program asks the user about any special days, and checks entries against an internal calendar to make sure each date is valid. If the input is reasonable, it loads the data into its appropriate position in the three dimensional special-days array. If there is anything un-

reasonable about the input, the system will ask the user to try again. This method allows the user to input data and examine the contents of the data base without requiring him to have any knowledge of structure of the data base. The combination of a high level user interface with special precautions to test the input for validity insures the integrity of the special day data. Even though the precise details of the user interaction would be much different in any practical implementation, the importance of this high level interaction still remains.

The subject used in this test had some very diverse behavior patterns. She was working as well as going to school part time at night. On a working day, she would typically get up around six, and leave for work around eight. If there was no school on that day, she could be home anywhere between three and six. On a school day, she would rarely be home before ten. On weekends or days off from work, she would frequently be at home studying at the dining room table.

During the middle of the data collection, a major change in behavior occurred due to a change in the subject's school schedule. Initially, the subject had evening classes on Wednesdays and Thursdays. Then, on Between May 21, and June 1, there was a break between semesters. From June 1 on, the subject had evening classes on Mondays and Thursdays. Due to the fact that the system could be given the name "school day" as a name for a special day, and that it could be told well in advance which days were school days, this major disruption in behavior caused only a brief disturbance in performance. The only reason for any change in performance at all was the fact that prior to the behavior disruption, the system had now way of knowing the significance of the "school day" name. The expert which made its prediction on the basis of the patterns observed the previous week was just as reliable as the special day expert until the first semester was over. At this point, the weekly regularity was broken, so those experts which made their predictions on this basis lost some influence, while the expert for school days gained some influence. If there were to be another change in the school schedule in the future, the altered influence of these experts should provide improved

performance.

There are several details of the following results that need to be explained before they can be completely understood. First, at the beginning of each day, the date is printed, and any special names that should correspond to the date are printed beside it. Below the date, the pattern to be predicted is printed as an array of ones and zeros. In this format, each row represents a particular zone of the apartment, and each column represents one of the half hour time slices. The pattern begins and ends at midnight. Each "1" in the pattern indicates that the corresponding zone was occupied during that time slice. Each "0" indicates that the zone was unoccupied.

Below the pattern to be predicted, we see a similar array which displays the performance of the decision algorithm. In addition to the ones and zeros seen before, this uses three other symbols as well. These are the tilde "~", the underline "_", and the asterisk " * ". These additional symbols are used to indicate errors made by the decision algorithm. The tilde is used at times when the heat was turned up when it was not needed. This symbol represents what has previously been called "waste errors". The underline and asterisk both represent times when the heat was turned down when it should have been turned up. The difference is that the underline corresponds to times when the heat should have been turned up because an unoccupied zone needed pre-heating, and the asterisk corresponds to an error that was made in an already occupied zone. It is important to note that from a comfort standpoint the asterisk carries much less significance than the underline. This is because the people sensors insure that an unoccupied zone will be heated, regardless of what the decision function says.

Next to each row of the performance display, the total degree-minutes of energy savings, and the total degree-minutes of discomfort are displayed for the corresponding zone. These figures are obtained from running the thermal model of the house on the heating pattern for each zone, assuming complete isolation of zones. The amount of energy savings shown is relative to the amount that would have to be used to keep the temperature at the comfort

level throughout the entire day. The degree-minutes of discomfort are determined from the difference in the current comfort temperature and the actual zone temperature. Note that the a correction is made here for moments when more that one zone is occupied during a single time slice.

The models all used an air temperature rise time of one degree every three minutes. The time constant for rising wall temperature was 140 minutes, and the time constant for falling wall temperature was 150 minutes. The comfort temperature setting was 70 degrees Farenheight, and the normal setback temperature was 60 degrees Farenheight. The night setback temperature for occupied zones was also 60 degrees, and the night setback period was from midnight to 6 AM. During the night setback period, the comfort temperature is redefined as the night setpoint. Thus, discomfort is only counted if the subject goes into a zone which is colder than this setpoint. Since a 70 degree comfort temperature is required in any occupied zone as soon as the night setback period is over, the temperature must be turned up in advance to avoid any discomfort. In this case, the temperature was turned up at 5:30 to provide the proper time for heating.

Below the performance display, we see a summary of the comfort and savings results. The savings figures are averaged, and translated into the percent savings using a constant derived from comparison of my model with the Honeywell model. The discomfort figures are totaled to give the total discomfort in degree-minutes for the day. Savings and comfort figures are also shown for the model which simulates no isolation between zones. This model was run in the same way as the zone models except it was considered unoccupied only if each zone was unoccupied, and it was unheated only if each zone was unheated. As a result, we see that with no isolation, there is greater comfort, but less energy is saved.

Finally, the output shows the performance of the class prediction algorithm. There are three matrices shown. One displays the predicted classes, one shows the classes that were last used during each day segment, and one shows the final classifications. Each matrix has

four rows and four columns. The rows correspond to the different zones, and the columns correspond to the different day segments. The classes for the first six hours are in the first column, the classes for the next six hours are in the second column, and so fourth. The matrix of predicted classes shows the selections of the class prediction algorithm obtained at the beginning of each segment. The matrix of last classes used shows which classes were in use at the completion of each segment. If a class that was initially predicted is rejected by the self evaluation algorithm, there will be a difference between the predicted class and the last class used. The final classification matrix shows the results of the pattern classification procedure. If the class predictor is doing its job properly, the predicted classes will be the same as the final classifications. In studying these results, it is important to note that the system is started with no past experience so it performs poorly at first, but improves over time.

14.1 Summary of Results

These two months of results may be summarized as follows:

The average percent savings per day with maximum isolation was: 10.76%

The average percent savings per day with minimum isolation was: 6.70%

The average discomfort per day with maximum isolation was: 98.9 degree-minutes

The average discomfort per day with minimum isolation was: 11.5 degree-minutes

Since the savings from night setback alone for the subject in this test was only 3.7%, my system will provide anywhere from 1.8 to 2.9 times the savings of night setback with an average discomfort of only 11.5 to 98.9 degree-minutes per day. This discomfort is almost negligible when we consider that it is equivalent to only one to ten minutes at a temperature ten degrees below the comfort setting.

```
(test-run kaw-day-list 59.)


OK, get to work !!


Do you want to input a special day? No

Present date:  4/22/1981     school day
Pattern to be predicted:

|000000000000110000000000000000000000000000011100|
|000000000000001000000000000000000000000001100000|
|000000000000011100000000000000000000000001110000|
|111111111110100000000000000000000000000001100011|


Performance:

|~~~~~~~~~~1111~~~~~~~~~~~~~~~~~~~~~~~~~~~~11111~~|  Savings: 1906   Discomfort: 0
|~~~~~~~~~~~~111~~~~~~~~~~~~~~~~~~~~~~~~~1111~~~~~|  Savings: 1906   Discomfort: 0
|~~~~~~~~~~~11111~~~~~~~~~~~~~~~~~~~~~~~~11111~~~~|  Savings: 1906   Discomfort: 0
|11111111111111~~~~~~~~~~~~~~~~~~~~~~~~~~1111~1111|  Savings: 1906   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 3.7%     Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 3.7%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  | 0   0   0   0  |                        | 0   0   0   0  |
  | 0   0   0   0  |                        | 0   0   0   0  |
  | 0   0   0   0  |                        | 0   0   0   0  |
  | 0   0   0   0  |                        | 0   0   0   0  |

                          Final Classifications:

                           | 1   1   1   1  |
                           | 1   1   1   1  |
                           | 1   1   1   1  |
                           | 1   1   1   1  |
```

```
Present date:  4/23/1981     school day
Pattern to be predicted:

|00000000000011000000000000000000000000000000000111|
|00000000000010010000000000000000000000000000000000|
|00000000000001110000000000000000000000000000001100|
|11111111111110000000000000000000000000000000001100|


Performance:

|00000000001111000000000000000000000000000~~11111|  Savings: 8582   Discomfort: 0
|0000000000__111*000000000000000000000000~~~~~00000|  Savings: 7945   Discomfort: 50
|00000000000_1111000000000000000000000000~~~~11**00|  Savings: 7684   Discomfort: 0
|1111111111111~000000000000000000000000000~~~111111|  Savings: 8175   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 15.9%    Total discomfort degree minutes: 50

                    Minimum Isolation of Zones
 Average Percent Savings: 14.4%    Total discomfort degree minutes: 0

   Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |

                          Final Classifications:

                           |  1   1   1   1   |
                           |  1   1   1   2   |
                           |  1   1   1   1   |
                           |  1   1   1   1   |
```

Present date: 4/24/1981
Pattern to be predicted:

```
|10000000000011000000000000000010000000000000000000|
|00000000000010100000000000000010001110111111111111|
|00000000000011110000000000000010001001000000010000|
|11111111111110000000000000000011100000000000000001|
```

Performance:

```
|*0000000001111~~00000000000__*000000000000~~~~~~|  Savings: 8397   Discomfort: 50
|0000000000__111~~0000000000__*0__*11_**11111*111|  Savings: 7489   Discomfort: 204
|0000000000__1111~0000000000__*0__*11*000~111~~~~|  Savings: 7164   Discomfort: 163
|1111111111111~~0000000000000__**1~~~00000~~~~111|  Savings: 6908   Discomfort: 150
```

Maximum Isolation of Zones
Average Percent Savings: 14.7% Total discomfort degree minutes: 566

Minimum Isolation of Zones
Average Percent Savings: 10.1% Total discomfort degree minutes: 150

Predicted Classes:

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Last Classes Used:

1	1	1	1
1	1	0	0
1	1	0	1
1	1	0	1

Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	1

```
Present date:  4/25/1981
Pattern to be predicted:

|00000000000000000000000000011111111100000000000|
|00000000000001000000110010010000000000000000000|
|00000000000000100000011100000000000100000000000|
|11111111111110111111100111110000000000000000000|


Performance:

|000000000~~~~~~~~0000000~1111111111100000000000~|  Savings: 5656   Discomfort: 0
|000000000~~111~~~~__*1111111~~000000~~~~~~~~~~~~|  Savings: 3444   Discomfort: 28
|000000000~~~111~~~~__*11~~~~~~000__*0000~~~~~~0~|  Savings: 4017   Discomfort: 77
|1111111111111111***1111111*~~0000000000000000~~|  Savings: 5211   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 9.0%     Total discomfort degree minutes: 105

                    Minimum Isolation of Zones
 Average Percent Savings: 4.4%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   1   1   1  |                     |  1   0   1   1  |
  |  1   1   1   1  |                     |  1   0   1   1  |
  |  1   1   1   1  |                     |  1   0   1   1  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   2   1   3   |
                         |  1   2   1   1   |
                         |  1   2   1   1   |
```

```
Present date:  4/26/1981
Pattern to be predicted:

|000000000000000000000010001110000000011000000111|
|000000011000000000000010001001111111100000000100|
|001100001100000000000010000111000000010000000000|
|001111110111111111111011110010000011110000000000|


Performance:

|~~~~~~~~~~~~~~~~~~~~111~11111~~~~~~11**0~~~~11111|  Savings: 3660   Discomfort: 0
|~~0~~1_**~~~~~~~~~~111~1_*1111111111~~~~~~~11111|  Savings: 2352   Discomfort: 6
|1111~~1_*1~~~~~~~~~~111~0__111~~~~~1_*0000000000|  Savings: 3941   Discomfort: 38
|111111111111111111111111__*000__*1**0~~~~~~~~1|  Savings: 4546   Discomfort: 71

                    Maximum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 115

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   0  |
  |  1   1   1   1  |                       |  0   1   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   2   1  |                       |  1   1   1   0  |

                        Final Classifications:

                        |  2   1   1   2  |
                        |  2   2   1   2  |
                        |  2   2   2   1  |
                        |  1   1   2   2  |
```

```
Present date:  4/27/1981
Pattern to be predicted:

|10000000000000000000000000000000100000000000111|
|11000000000000000000101110000011101111111111000|
|00000000000000000000000011100000000000001100001100|
|01111111111111111111111100000000000000000000000001|


Performance:

|1~~~~~~~00~~~~~~~~000~00~~~~~~~111~~0000000__*11|  Savings: 4105   Discomfort: 75
|11~~~~~~~~~~~~~0000__*1111~~~111111****1111111~~~|  Savings: 3520   Discomfort: 75
|~~~~~~~~~~~~~~~0000000__111~~~000000__*1~~1111~~|  Savings: 7517   Discomfort: 150
|11111111111111111111111~~~~~~~~00000000000000__*|  Savings: 4753   Discomfort: 75

                    Maximum Isolation of Zones
 Average Percent Savings: 9.7%     Total discomfort degree minutes: 375

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   2  |
  |  1   2   1   3  |                       |  1   1   1   3  |
  |  1   2   1   1  |                       |  1   2   1   0  |
  |  1   1   1   2  |                       |  1   1   1   2  |

                        Final Classifications:

                          |  2   1   1   2  |
                          |  2   2   3   2  |
                          |  3   2   2   1  |
                          |  1   2   2   2  |
```

```
Present date:  4/28/1981
Pattern to be predicted:

|0000000000000000000000000000000000000000000000000000|
|0000000000001100000000000001110011001111101111|
|0000000000001011000000000001100000001000000110|
|1111111111111000000000000000011100000000010000|


Performance:

|00000000000000000000000~00~~000000000000~~~~~~~~|  Savings: 9499   Discomfort: 0
|~000000000__**0~~0000000~0__*1111111111111111111|  Savings: 4763   Discomfort: 125
|0000000000_111*1~000000000__**0000~11100~~~1111~|  Savings: 5961   Discomfort: 75
|1111111111111~~~~~000000~000__*11000~~~~~111~~11|  Savings: 6527   Discomfort: 75

                    Maximum Isolation of Zones
 Average Percent Savings: 13.1%    Total discomfort degree minutes: 275

                    Minimum Isolation of Zones
 Average Percent Savings: 7.2%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   1   1   1  |
  |  2   2   1   3  |                       |  2   1   3   3  |
  |  1   2   1   1  |                       |  1   1   1   1  |
  |  1   1   1   2  |                       |  1   1   2   2  |

                        Final Classifications:

                        |  2   1   1   1  |
                        |  2   2   2   3  |
                        |  3   2   1   1  |
                        |  1   1   2   1  |
```

```
Present date:  4/29/1981     school day
Pattern to be predicted:

|000000000000000000000000000001101111110000011111|
|000000000000001000110000000010000000000001100000|
|000000000000000000001100000000000011000000001000|
|111111111111111111000000000001110000000000000001|


Performance:

|0000000000~~~00000000000~~11111111***000__*11111|  Savings: 5993   Discomfort: 52
|~00000~~~~~~111~1111~~00~~11*0000000~~~111100000|  Savings: 3839   Discomfort: 0
|000000000~~~~~~~~1111~~000000001111~000000__*000|  Savings: 5250   Discomfort: 75
|1111111111111111**00000000~1_**1~~~~000000000_11|  Savings: 5428   Discomfort: 23

                    Maximum Isolation of Zones
 Average Percent Savings: 10.0%    Total discomfort degree minutes: 150

                    Minimum Isolation of Zones
 Average Percent Savings: 4.4%     Total discomfort degree minutes: 0

   Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   1   2  |
  |  2   2   1   2  |                     |  2   2   1   2  |
  |  3   2   2   1  |                     |  3   2   2   1  |
  |  1   1   1   1  |                     |  1   1   2   1  |

                          Final Classifications:

                           |  1   1   1   2   |
                           |  1   2   1   1   |
                           |  1   2   1   1   |
                           |  1   2   2   1   |
```

```
Present date:  4/30/1981     school day
Pattern to be predicted:

|000000000000000000000000000000000000000000000000000|
|000000000011100000000000000000000000000000000000111|
|000000000001100000000000000000000000000000000000100|
|111111111111000000000000000000000000000000000000100|


Performance:

|000000000000~00000000000000~00000000000~~~~~~~~~~|  Savings: 8805   Discomfort: 0
|00000000_1111~0000000000~~~~~00000000~~~~~~__***|  Savings: 7637   Discomfort: 21
|~000000001111~00000000000~~~~~~~~~~~~~~~~~~~111~~|  Savings: 6436   Discomfort: 0
|111111111111~~~~~000000~0~~~~~~~~0~~0000~~~11111|  Savings: 5243   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 13.8%    Total discomfort degree minutes: 21

                    Minimum Isolation of Zones
 Average Percent Savings: 6.6%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  2   1   1   1  |                     |  2   1   1   1  |
  |  1   2   1   1  |                     |  1   2   1   2  |
  |  1   2   2   1  |                     |  1   2   2   1  |
  |  1   1   2   1  |                     |  1   1   2   1  |

                        Final Classifications:

                         |  2   1   1   1  |
                         |  2   2   1   1  |
                         |  3   2   1   1  |
                         |  1   1   1   1  |
```

Present date: 5/1/1981
Pattern to be predicted:

```
|0000000000000100000000000000000000000001100000000|
|0000000000001000000000000000000001000100111111111|
|0000000000000110000000000000000001000100000000000|
|1111111111111000000000000000000001111000000000000|
```

Performance:

```
|0000000000~11*000000000000000000000001111~~~~~~~~|  Savings: 7871   Discomfort: 0
|0~~~~~~~~~~111~~~00000000~~~~00__*~11111111111111|  Savings: 5079   Discomfort: 45
|0000000000~1111~00000000000000__*0111~00~~~~~~~~|  Savings: 6329   Discomfort: 50
|1111111111111~~~~~000000000000__*111000000~~~~~1|  Savings: 6562   Discomfort: 50
```

Maximum Isolation of Zones
Average Percent Savings: 12.7% Total discomfort degree minutes: 145

Minimum Isolation of Zones
Average Percent Savings: 7.8% Total discomfort degree minutes: 121

Predicted Classes:

1	1	1	1
2	2	1	3
1	2	1	1
1	2	1	1

Last Classes Used:

1	1	1	1
2	2	2	3
1	2	1	1
1	1	2	1

Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	1

Present date: 5/2/1981
Pattern to be predicted:

```
|000000000000000000000011000000000011110000000000000|
|100000000000000000001111111111111111110000111111111|
|000000000000000000001100000000000011111111000000000|
|011111111111111111111110000000000111110000000000000|
```

Performance:

```
|00000000000~~~00000__*1~~~~~~~111111~~~000000000|  Savings: 4725   Discomfort: 50
|*000000000~~~~~~~11111**11111111111001111111111|  Savings: 2398   Discomfort: 0
|00000000000~~~~~~1111~~~~~~~~~1111111***~~~~~~~~|  Savings: 2375   Discomfort: 0
|11111111111111111***1**~~~000~1111110000000000__|  Savings: 4430   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 6.8% Total discomfort degree minutes: 50

Minimum Isolation of Zones
Average Percent Savings: 4.4% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	1	2
1	1	1	1
1	1	2	1

Last Classes Used:

1	0	1	1
1	1	1	2
1	1	1	0
1	0	2	1

Final Classifications:

1	1	1	2
1	2	3	2
1	2	1	2
1	2	2	2

```
Present date:  5/3/1981
Pattern to be predicted:

|000000000000000110000000000000000000111111000000|
|000000000000000110000111100011011111111110111110|
|000000000000000000000000000000011000111110000000|
|111111111111111101111100111110001100000000000011|


Performance:

|~000000000000__*1~~~~~~~~~~~~~~~~~~1111***1~~~~~~|  Savings: 4783   Discomfort: 50
|0000000000~~0__*1~~111111~11111111111****111111~|  Savings: 2900   Discomfort: 50
|~~~~~~00000~00000~~~~~~~~~~~~~1111~11111**00~~~~~|  Savings: 3495   Discomfort: 0
|111111111111111111111111***1~1111~~~00000000__*1|  Savings: 5218   Discomfort: 75

                    Maximum Isolation of Zones
 Average Percent Savings: 8.0%     Total discomfort degree minutes: 175

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

   Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   0   1   0  |
  |  2   2   1   1  |                       |  2   1   1   2  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   2   1  |                       |  1   1   1   2  |

                          Final Classifications:

                           |  2   2   1   2  |
                           |  2   2   3   2  |
                           |  3   2   1   2  |
                           |  1   2   1   2  |
```

Present date: 5/4/1981
Pattern to be predicted:

```
|00000000000010000000000000000000110111000000000011|
|00000000000010000000000000000000110011101111111100|
|00000000000001100000000000000000100111011000000000|
|11111111111100000000000000000001011111000000000000|
```

Performance:

```
|0000000000__1~~00000000000000__*11111~~~~~~~1111|  Savings: 6305   Discomfort: 150
|0000000000_11~~~000000000000__*111111111111111~~|  Savings: 5426   Discomfort: 50
|000000000001_1*~000000000000~111~111111**0~~~~~~~|  Savings: 5028   Discomfort: 60
|111111111111~~~000000000~00011111111*0000~~~~~11|  Savings: 6550   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 11.4% Total discomfort degree minutes: 260

Minimum Isolation of Zones
Average Percent Savings: 8.4% Total discomfort degree minutes: 0

Predicted Classes:

2	1	1	1
2	2	1	3
3	2	1	1
1	1	2	2

Last Classes Used:

2	1	0	1
2	2	2	3
3	1	1	1
1	1	2	2

Final Classifications:

2	2	1	1
2	2	2	3
3	2	1	1
1	2	2	1

```
Present date:  5/5/1981
Pattern to be predicted:

|0000000000000110000000000000000111000011110000111|
|0000000000000100000000000000000101001100001111100|
|0000000000000110000000000000000010000000000000000|
|1111111111111000000000000000000011100000000000000|


Performance:

|00000000~~~1111~~00000000000_1111~~111111~~11111|  Savings: 4941   Discomfort: 0
|0000000~~~~111~000000000~~~~__*111111~~1111111~~|  Savings: 4567   Discomfort: 40
|000000000~~1111~00000000000~~~111~~~~~~000~~~~~0|  Savings: 5011   Discomfort: 0
|1111111111111~00000000000000~~11111~0000000~~~11|  Savings: 6854   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 10.5%    Total discomfort degree minutes: 40

                    Minimum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   1   1   1  |
  |  2   2   1   3  |                       |  2   2   2   3  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   2   1  |                       |  1   1   2   1  |

                          Final Classifications:

                           |  2   1   1   1  |
                           |  1   2   2   1  |
                           |  1   2   1   1  |
                           |  1   2   2   1  |
```

```
Present date:  5/6/1981      school day
Pattern to be predicted:

|0000000000000100000000000000000000000000000000000000|
|0000000000001100000000000000000000000000000111110|
|0000000000000100000000000000000000000000000110000|
|1111111111111000000000000000000000000000000110001|


Performance:

|0000000000~111~~~~000000000000~~~000~~~~~~~~~~~~~|  Savings: 5411   Discomfort: 0
|000000000~1111~000000000~~~~00000000~0~~1111111~|  Savings: 6099   Discomfort: 0
|000000000~~~111~0000000000000~~~~~~~~~~~1111~~~~|  Savings: 5652   Discomfort: 0
|1111111111111~~0000000000000~~~~~~~~0000__11~111|  Savings: 6741   Discomfort: 49

                    Maximum Isolation of Zones
 Average Percent Savings: 11.7%    Total discomfort degree minutes: 49

                    Minimum Isolation of Zones
 Average Percent Savings: 6.5%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   2   1  |                       |  1   1   2   2  |

                          Final Classifications:

                           |  2   1   1   1  |
                           |  1   2   1   1  |
                           |  1   2   1   1  |
                           |  1   1   1   1  |
```

```
Present date:  5/7/1981     school day
Pattern to be predicted:

|00000000000011000000000000000000000000000000000011|
|00000000000010000000000000000000000000000000000110|
|00000000000011000000000000000000000000000000000110|
|11111111111100000000000000000000000000000000000111|


Performance:

|00000000001111~~~~0000000000000000000~~~~~~~~1111|  Savings: 6407   Discomfort: 0
|000000000~111~~000000000~~~00000000000~~~~~1111~|  Savings: 6909   Discomfort: 0
|0000000000~1111~00000000000000~~~~~~~~0~~~~111*0|  Savings: 5538   Discomfort: 0
|111111111111~~~000000000000000000000000~~~11111|  Savings: 8515   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 13.4%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 7.5%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   1   1   1   |
                         |  1   1   1   1   |
                         |  1   1   1   1   |
```

```
Present date:  5/8/1981
Pattern to be predicted:

|00000000000011000000000000001110000001000000011|
|00000000000011110000000000001111001111111111100|
|00000000000001110000000000000010001111100000000|
|11111111111110000000000000000011111111000000000|


Performance:

|000000000~1111~~~000000000__*11~~~~~111~~~~~1111|  Savings: 4819   Discomfort: 75
|000000000~1111110000000000__*1111111111111111~~|  Savings: 5212   Discomfort: 75
|000000000~~11111~00000000000_11~111111*000~~~~00|  Savings: 5536   Discomfort: 0
|1111111111111~~0000000000001111111***~~~~~~~11|  Savings: 5697   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 10.4%    Total discomfort degree minutes: 150

                    Minimum Isolation of Zones
 Average Percent Savings: 9.1%     Total discomfort degree minutes: 150

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   0   1  |
  |  1   1   2   3  |                       |  1   1   3   3  |
  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   2   1  |                       |  1   1   2   0  |

                          Final Classifications:

                           |  1   1   1   1  |
                           |  1   1   2   3  |
                           |  1   1   2   1  |
                           |  1   1   2   3  |
```

```
Present date:  5/9/1981
Pattern to be predicted:

|00000000000000000110000000000000000000000000110000|
|00000000000000000111111111111111111111101111101111|
|00000000000000000000000000000000000000001000000000|
|11111111111111111100000000000000000000000000000000|


Performance:

|0000000000~~~~~11110000000~~~~~~~~~~~~~~~1111~~~~|  Savings: 3564   Discomfort: 0
|000000000~~~~~~11111***1**1111111111111111111111|  Savings: 2225   Discomfort: 0
|000000000~~~~~~~~~~~000000~~~~~~~~~~111~~~~~~~~~|  Savings: 4002   Discomfort: 0
|1111111111111111~~00000000000~~~~~~000000~~~~11|  Savings: 5530   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 7.5%     Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 4.4%     Total discomfort degree minutes: 0

   Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   1   3   2  |                       |  1   0   3   2  |
  |  1   1   1   1  |                       |  1   2   1   1  |
  |  1   2   2   2  |                       |  1   2   2   2  |

                          Final Classifications:

                           |  2   2   1   2  |
                           |  1   2   3   2  |
                           |  1   2   1   2  |
                           |  1   2   2   2  |
```

Present date: 5/10/1981
Pattern to be predicted:

```
|00000000000000000000000000000000000000000000000110|
|00000000000000001011111111111111111111111111111000|
|00000000000000001111110000000000011100000000000000|
|11111111111111110011110000000000000000000000000001|
```

Performance:

```
|00000000000~~~~~00000000~~~~~~~~~~~~~~~00~~~1111~|  Savings: 4088   Discomfort: 0
|0000000000~~~~~11111111****11111111**1111111~~~~|  Savings: 2193   Discomfort: 0
|000000000~~~~~1111***1~~000~~~~~11111~~~~~00000000|  Savings: 3134   Discomfort: 0
|111111111111111111***1~~0000000~~~~~~000000~~~111|  Savings: 4807   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 7.0% Total discomfort degree minutes: 0

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	1
2	2	2	2
3	2	1	2
1	1	2	2

Last Classes Used:

2	2	1	1
2	2	1	2
3	0	1	2
1	0	2	2

Final Classifications:

2	2	1	2
2	2	3	2
3	2	1	2
1	2	2	2

```
Present date:  5/11/1981
Pattern to be predicted:

|0000000000011000000000000000000000G000000000001110|
|0000000000011000000000000000000011001111111111000|
|0000000000000110000000000000000011000000000000011|
|1111111111110000000000000000000111100000000000001|


Performance:

|000000000_111~~0000000000~~~~~~~~~~~~~~000~11111~|  Savings: 4841   Discomfort: 0
|0000000001111~~~~~~~~0000~~~~~11*1111111111111~~~|  Savings: 2780   Discomfort: 0
|000000000~~111100000000000~~~1111~~~~~~~~~~~~11**|  Savings: 6258   Discomfort: 0
|111111111111~~~~00000000~~0~~111111~000000~~~111|  Savings: 5323   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 9.4%     Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 5.4%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   2   1   1  |                       |  2   2   1   1  |
  |  2   2   3   3  |                       |  2   2   2   3  |
  |  3   2   1   1  |                       |  3   2   1   1  |
  |  1   1   2   1  |                       |  1   1   2   1  |

                          Final Classifications:

                           |  2   2   1   1  |
                           |  2   2   2   3  |
                           |  3   2   1   1  |
                           |  1   2   2   1  |
```

Present date: 5/12/1981
Pattern to be predicted:

```
|00000000000000000000000000000000000000001100000011|
|00000000000001000000000000000000001101101100111100|
|00000000000000110000000000000000001100001111100110|
|11111111111110000000000000000000011110011000000000|
```

Performance:

```
|0000000~~~~~~~~~000000000000~~~~0000__*1~~~~1111|  Savings: 6276   Discomfort: 38
|00000000~~~111~~~~~00000~~~000__*111*_*111****00|  Savings: 4058   Discomfort: 55
|000000000~~~1111~00000000000~~1111~~11111111111~|  Savings: 4645   Discomfort: 0
|1111111111111~~~~~0000000000~~111111__*1000000_1|  Savings: 5573   Discomfort: 13
```

Maximum Isolation of Zones
Average Percent Savings: 10.1% Total discomfort degree minutes: 105

Minimum Isolation of Zones
Average Percent Savings: 6.1% Total discomfort degree minutes: 0

Predicted Classes:

2	1	1	1
1	2	1	1
1	2	1	1
1	2	2	1

Last Classes Used:

2	1	1	2
1	2	2	2
1	2	1	1
1	1	2	3

Final Classifications:

1	1	1	1
1	2	2	3
1	2	1	1
1	2	2	3

```
Present date:  5/13/1981      school day
Pattern to be predicted:

|000000000000000000000000000000000000000000000000|
|000000000011000000000000000000000000000011000000|
|000000000011000000000000000000000000000011111111|
|111111111110000000000000000000000000001000000|


Performance:

|0000000000~~~~~000000000000000000000~~000~~~~~~~|  Savings: 7968   Discomfort: 0
|00000000__11~~~000000000~~~~~~~0000000_111~~~~~~|  Savings: 6332   Discomfort: 0
|000000000_111~~~00000000000000~~~~~~~~1111111111|  Savings: 5459   Discomfort: 0
|11111111111~~~~~~0000000~00000000000000__10~~~11|  Savings: 8651   Discomfort: 50

                    Maximum Isolation of Zones
 Average Percent Savings: 13.9%    Total discomfort degree minutes: 50

                    Minimum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   2   3   1  |                     |  2   2   1   1  |
  |  1   2   1   1  |                     |  1   2   1   1  |
  |  1   2   1   1  |                     |  1   1   1   3  |

                          Final Classifications:

                           |  1   1   1   1   |
                           |  1   2   1   1   |
                           |  1   2   1   1   |
                           |  1   1   1   1   |
```

Present date: 5/14/1981 school day
Pattern to be predicted:

```
|000000000000100000000000000000000000000000000111|
|000000000000101000000000000000000000000000011100|
|000000000000111000000000000000000000000000011100|
|111111111111000000000000000000000000000000011100|
```

Performance:

```
|00000000~~111~~~000000000000000000000~00000011111|  Savings: 8471   Discomfort: 0
|0000000~~~11111000000000~~~000000000000~~11111~~|  Savings: 7215   Discomfort: 0
|00000000~~11111~000000000000000000000~~~~~11111~~|  Savings: 6936   Discomfort: 0
|111111111111~~~000000000000000000000000001111111|  Savings: 8926   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 15.5% Total discomfort degree minutes: 0

Minimum Isolation of Zones
Average Percent Savings: 11.6% Total discomfort degree minutes: 0

Predicted Classes:				Last Classes Used:			
1	1	1	1	1	1	1	1
1	2	1	1	1	2	1	1
1	2	1	1	1	2	1	1
1	1	1	1	1	1	1	1

Final Classifications:

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Present date: 5/15/1981
Pattern to be predicted:

```
|0000000000000000000000000000000000000000000000000000|
|0000000000001100000000000000000011100011111111111111|
|0000000000000110000000000000000001100000000000000000|
|1111111111110000000000000000000011111100000000000000|
```

Performance:

```
|000000000~~~~~~~000000000000000000000~~~000~~~~~~|  Savings: 8129   Discomfort: 0
|00000000~~1111~~00000000000~~11111~11111111111**|  Savings: 5032   Discomfort: 0
|000000000~~1111~00000000000000__*1~~~~~~~~~~~~11|  Savings: 6120   Discomfort: 50
|111111111111~~~000000000000~~~111111**0~00000011|  Savings: 6178   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 12.5% Total discomfort degree minutes: 50

Minimum Isolation of Zones
Average Percent Savings: 9.5% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	2	3
1	1	1	1
1	1	2	2

Last Classes Used:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	3

Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	3

Present date: 5/16/1981
Pattern to be predicted:

```
|000000000000000000010000011100000000000000000111000|
|000000000000000001111111111111111111110011111000111|
|100000000000000011000000000000000000010000000000000|
|111111111111111111000000000000000000000000000000000|
```

Performance:

```
|0000000000~~~~~~11*000__*11~~~~~~~~~~00~~11111~~~|  Savings: 3658   Discomfort: 61
|000000000~~~~~111111**1*1111111111111111~11111|  Savings: 2218   Discomfort: 0
|*00000~~0000~~1111~~00000000~~~~~~11*0000000000~|  Savings: 5331   Discomfort: 0
|1111111111111111**~~000000000000~~~~0000000~~~11|  Savings: 6834   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 8.8% Total discomfort degree minutes: 61

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	3	2
1	1	1	1
1	1	2	2

Last Classes Used:

1	1	1	1
1	2	3	2
2	1	1	1
1	2	2	2

Final Classifications:

1	1	1	2
1	2	3	2
1	2	1	2
1	2	2	2

Present date: 5/17/1981
Pattern to be predicted:

```
|000000000000000010000011110000000000000000100000|
|000000000000000011111100011101111100011111111111|
|000000000000000011000000000010000010110000110000|
|111111111111111111000000000000000000000000100000|
```

Performance:

```
|00000000~~~~~~111~~~1_*111~~~~~~~~~~0000111~~~~~|  Savings: 2742   Discomfort: 20
|000000000~~~~~11111111~11111111111~1111111111111|  Savings: 2193   Discomfort: 0
|00000000~~~~~~1111~~000000_11~~~111111~0__**~~~~|  Savings: 5187   Discomfort: 16
|11111111111111111~~000000000000~~~~0000__*~~0__|  Savings: 6567   Discomfort: 38
```

Maximum Isolation of Zones
Average Percent Savings: 8.2% Total discomfort degree minutes: 74

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	1	1	2
2	2	3	2
3	2	1	2
1	2	2	2

Last Classes Used:

2	0	1	2
2	2	3	2
3	2	1	0
1	2	2	3

Final Classifications:

2	2	1	2
2	2	3	2
3	2	1	2
1	2	2	2

Present date: 5/18/1981
Pattern to be predicted:

```
|00000000000001000000000000000000000000000001100|
|00000000000010000000000000000000011001111111100111|
|00000000000001100000000000000000110000000011000|
|11111111111110000000000000000001111000000000000|
```

Performance:

```
|0000000000~111~~0000000000000000000000000_111~~|  Savings: 8459   Discomfort: 0
|0000000000111~~~~~~~0000~~~~~~1111111111111111*|  Savings: 2780   Discomfort: 0
|0000000000~~1111~000000000000~1111~~~~0~~1111~~~|  Savings: 5118   Discomfort: 0
|1111111111111~~~~000000000000~1111110000000~~~11|  Savings: 7349   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 11.6% Total discomfort degree minutes: 0

Minimum Isolation of Zones
Average Percent Savings: 5.4% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	1
2	2	3	3
3	2	1	1
1	2	2	1

Last Classes Used:

2	2	1	1
2	2	2	3
3	2	1	1
1	2	2	1

Final Classifications:

2	2	1	1
2	2	2	3
3	2	1	1
1	2	2	3

Present date: 5/19/1981
Pattern to be predicted:

```
|000000000000110000000000001100110000000000001100|
|000000000000100000000000000110111111111111110011|
|000000000000011000000000000110000000000000000000|
|111111111111100000000000000111100000000000000000|
```

Performance:

```
|00000000001111~~~~~00000__*11111~~~~00000~1111~~|  Savings: 5005   Discomfort: 150
|000000000~111~~~~~0000000__*11111**1**1111111_**|  Savings: 4101   Discomfort: 54
|000000000~~1111~000000000__*1~~~~~~~00000~~~~~00|  Savings: 5452   Discomfort: 38
|1111111111111~~~000000000__*111~~~~~0000~~~~~~11|  Savings: 5805   Discomfort: 38
```

Maximum Isolation of Zones
Average Percent Savings: 10.0% Total discomfort degree minutes: 279

Minimum Isolation of Zones
Average Percent Savings: 6.9% Total discomfort degree minutes: 150

Predicted Classes:

1	1	1	1
1	2	2	2
1	2	1	1
1	2	2	2

Last Classes Used:

1	1	0	1
1	2	1	2
1	2	0	1
1	2	0	2

Final Classifications:

1	1	1	1
1	2	3	3
1	2	1	1
1	2	2	3

```
Present date:  5/20/1981     school day
Pattern to be predicted:

|00000000000011000000000000000000000000000000000110|
|00000000000010000000000000000000000000000001111001|
|00000000000011000000000000000000000000000001111100|
|11111111111110000000000000000000000000000000110000|


Performance:

|000000000~1111~~~0000000~~~~~~000000000000~1111~|  Savings: 6232   Discomfort: 0
|00000000~~111~~~~~000000~~~~~0000000000011111111*|  Savings: 5729   Discomfort: 0
|00000000~~~111100000000000~~~~~~~~~~~~001111111~~|  Savings: 5764   Discomfort: 0
|1111111111111~~000000000~~~~~000000000000__11~~11|  Savings: 7404   Discomfort: 50

                    Maximum Isolation of Zones
 Average Percent Savings: 12.3%    Total discomfort degree minutes: 50

                    Minimum Isolation of Zones
 Average Percent Savings: 6.8%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   1   1  |                       |  1   1   1   2  |

                        Final Classifications:

                         |  2   2   1   1  |
                         |  2   2   1   1  |
                         |  3   2   1   1  |
                         |  1   2   2   1  |
```

```
Present date:  5/21/1981      school day
Pattern to be predicted:

|00000000000010000000000000000000000000000000000011|
|00000000000010000000000000000000000000000000000100|
|00000000000011100000000000000000000000000000000100|
|11111111111100000000000000000000000000000000000100|


Performance:

|000000000~111~~~~0000000000000000000000000~~1111|  Savings: 8468   Discomfort: 0
|00000000~~111~~000000000~~~000000000000~~~~111~~|  Savings: 7263   Discomfort: 0
|000000000~11111~0000000000000~~~~~00~000~~~~11111|  Savings: 6004   Discomfort: 0
|111111111111~~0000000000000000000000000~~111~~|  Savings: 9243   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 15.2%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 9.0%     Total discomfort degree minutes: 0

   Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |

                          Final Classifications:

                           |  1   1   1   1  |
                           |  1   1   1   1  |
                           |  1   1   1   1  |
                           |  1   1   1   1  |
```

Present date: 5/22/1981
Pattern to be predicted:

```
|00000000000001000000000000000000000011111011111|
|00000000000010000000000000000001101111111110000|
|11100000000011100000000000000010001111100000011|
|00111111111100000000000000000001111111100000000|
```

Performance:

```
|0000000000~111~~~000000000000000000_1111**111111|  Savings: 7262   Discomfort: 0
|000000000~111~~~~0000000000001111111111111111~~~0|  Savings: 5232   Discomfort: 0
|***00~~~~0__111~~000000000000__*~1111111~~~~111*|  Savings: 5889   Discomfort: 150
|111111111111~~~~00000000000~~~1111111111000000~~|  Savings: 5786   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 11.8% Total discomfort degree minutes: 150

Minimum Isolation of Zones
Average Percent Savings: 8.9% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	2	3
1	1	1	1
1	1	2	3

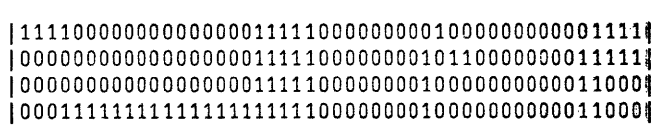
Last Classes Used:

1	1	1	1
1	1	2	3
2	1	2	1
1	1	2	3

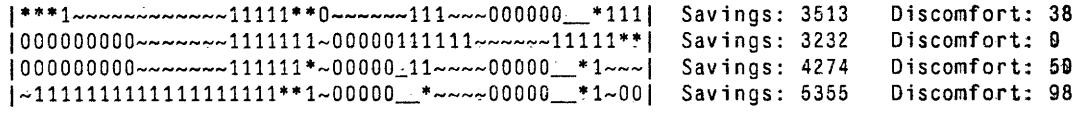
Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	3

Present date: 5/23/1981
Pattern to be predicted:



Performance:



Maximum Isolation of Zones
Average Percent Savings: 8.0% Total discomfort degree minutes: 186

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	3	2
1	1	1	1
1	2	2	2

Last Classes Used:

0	1	1	2
1	1	3	2
1	0	1	1
0	0	1	3

Final Classifications:

3	2	1	2
1	2	3	2
1	3	1	1
2	3	3	4

```
Present date:  5/24/1981
Pattern to be predicted:

|000000000000000000000011111111111111010000000110|
|110000000000000000000010111101111111011111111111|
|000000000000000000000011110000000001111000000110|
|001111111111111111111101100000000001110000000110|


Performance:

|000000000000~~~~~~~1_*1111111111111111~~0001111~|  Savings: 3429   Discomfort: 6
|1*000000000~~~~~~~~11111***11111111111111111111|  Savings: 2210   Discomfort: 0
|00000000000~~~~~~~~11111*00000~~111111~~~~~111*0|  Savings: 3496   Discomfort: 0
|_11111111111111111111111*00000000__***0~~~~11111|  Savings: 5400   Discomfort: 75

                    Maximum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 81

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   2   1   1  |                       |  2   1   1   1  |
  |  2   2   3   2  |                       |  2   2   1   2  |
  |  3   2   1   2  |                       |  3   2   1   2  |
  |  1   2   2   2  |                       |  0   2   2   3  |

                        Final Classifications:

                        |  2   2   1   2  |
                        |  2   2   3   2  |
                        |  3   2   1   2  |
                        |  1   2   4   2  |
```

Present date: 5/25/1981
Pattern to be predicted:

```
|00000000000000001100000000000000000111100000000|
|00000000000000001100000110000000010111101001111|
|00000000000000000011000000000000011111110110000|
|11111111111111111110000000000000000111100000000|
```

Performance:

```
|0000000000~~~~1111~~0000~~~~~~~~~__****~~~~~~~~~~|  Savings: 3147   Discomfort: 11
|0000000000~~001111~~~11**0~~~~~1_*11*11111111111|  Savings: 2715   Discomfort: 8
|00000000000~~~~~__*1~00000~~~~~11111***1111~~000|  Savings: 3677   Discomfort: 35
|*111111111111111111~~00000000~~~~111***~~00000__|  Savings: 4329   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 6.8% Total discomfort degree minutes: 54

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	1
2	2	3	2
3	2	1	1
1	2	2	2

Last Classes Used:

2	2	1	0
2	2	2	2
3	3	1	2
0	2	2	3

Final Classifications:

2	2	1	2
2	2	3	2
3	2	1	2
3	2	2	3

```
Present date:  5/26/1981
Pattern to be predicted:

|000000000000000000000000000000000000011100011000000|
|000000000000100000000000000000000110011100011111100|
|000000000000011000000000000000000001011111111000001|
|111111111111100000000000000000000011000001100000 11|
```

Performance:

```
|000000000000~~~~~~~00000000000000__*111111~~~~~~|  Savings: 6083   Discomfort: 50
|000000000011*~~~~~000000~~~~~1111111111111111~0|  Savings: 3325   Discomfort: 0
|~~~00000000_111~~0000000000000~11111111111~~~111|  Savings: 6181   Discomfort: 0
|1111111111111~~~~~~00000000~~~~1111~~~1_**~~1111|  Savings: 5281   Discomfort: 5

                    Maximum Isolation of Zones
 Average Percent Savings: 10.2%    Total discomfort degree minutes: 55

                    Minimum Isolation of Zones
 Average Percent Savings: 5.9%     Total discomfort degree minutes: 0
```

Predicted Classes:

1	1	1	1
2	2	3	3
1	2	1	1
1	2	2	3

Last Classes Used:

1	1	1	1
2	2	3	3
1	2	1	1
1	1	2	0

Final Classifications:

2	2	1	1
2	2	2	3
3	2	1	1
1	2	2	3

Present date: 5/27/1981
Pattern to be predicted:

```
|000000000000000000000000000000000000111000000000|
|000000000001110000000000000000000011001100001111|
|100000000000011000000000000000000010001111110000|
|111111111111000000000000000000000010000110010110|
```

Performance:

```
|000000000~~~~~~~~~~000000000000000__111~~~~~~~~~~|  Savings: 6146   Discomfort: 150
|00000000~11111~000000000~0000000__*1__*1~~11****|  Savings: 6999   Discomfort: 89
|*0000000~~~1111~000000000000~~~~111~11111111~~~~|  Savings: 4979   Discomfort: 0
|111111111111~~~~00000000000~~~~~111~0__**__*1111|  Savings: 5232   Discomfort: 54
```

Maximum Isolation of Zones
Average Percent Savings: 11.4% Total discomfort degree minutes: 294

Minimum Isolation of Zones
Average Percent Savings: 6.6% Total discomfort degree minutes: 0

Predicted Classes:

1	2	1	1
1	2	1	1
1	2	1	1
1	2	2	1

Last Classes Used:

1	2	1	1
1	2	2	2
1	2	1	1
1	2	2	2

Final Classifications:

1	1	1	1
1	2	2	3
1	2	1	1
1	2	2	3

```
Present date:  5/28/1981
Pattern to be predicted:

|00000000000011000000000000000000000000000000000000|
|00000000000110000000000000000000000000000000011011|
|00000000000011000000000000000000000000000000011011|
|11111111111100000000000000000000000000000000011111|


Performance:

|00000000~~1111~000000000000000000000~~~00~~~~~~~~|  Savings: 7665   Discomfort: 0
|00000000~1111~~000000000~00000000000000~~1111111|  Savings: 7737   Discomfort: 0
|00000000~~~11110000000000000~~~~~~~~~~~~1111111|  Savings: 5767   Discomfort: 0
|111111111111~0000000000000000000000000000~1111111|  Savings: 8963   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 14.8%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 10.6%    Total discomfort degree minutes: 0

   Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   1   1   1  |                     |  1   1   1   1  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   2   1   1   |
                         |  1   2   1   1   |
                         |  1   1   1   1   |
```

```
Present date:  5/29/1981
Pattern to be predicted:

|00000000000011000000000000000000011100000000011110|
|00000000000010000000000000000000111000011000000000|
|00000000000011110000000000000001100000000011100000|
|11111111111100000000000000000001100111100100000011|


Performance:

|000000000~1111~~00000000000000__**1~~~~~~111111~|  Savings: 6257   Discomfort: 75
|00000000~~111~~~00000000000011111~~1111~~~~~~~00|  Savings: 5317   Discomfort: 0
|000000000~1111110000000000001111~~~~~~1_*11~~~~0|  Savings: 5039   Discomfort: 10
|111111111111~~0000000000000~11111111**__*0001111|  Savings: 6193   Discomfort: 23

                    Maximum Isolation of Zones
 Average Percent Savings: 11.2%    Total discomfort degree minutes: 108

                    Minimum Isolation of Zones
 Average Percent Savings: 9.3%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   0   1  |
  |  1   1   2   3  |                     |  1   1   2   2  |
  |  1   1   2   1  |                     |  1   1   2   1  |
  |  1   1   2   1  |                     |  1   1   2   3  |

                        Final Classifications:

                         |  1   1   1   1  |
                         |  1   1   2   3  |
                         |  1   1   2   1  |
                         |  1   1   2   3  |
```

Present date: 5/30/1981
Pattern to be predicted:

```
|000000000000000000001111000011111100000000000000|
|0000000000000000000100111111000000000000000000011|
|0000000000000000000100000000000000011000000001100|
|1111111111111111111100000000000011000000000000000|
```

Performance:

```
|00000000~~~~~~~~~11111*0~11111111~~~00000~~~~~~~|  Savings: 2933   Discomfort: 0
|0000000~~~~~~~~~~111111*1111~~~~~~~~~~~~~~~~1111|  Savings: 2722   Discomfort: 0
|00000000~~~~~~~~0__*~~~~00~~~~~~1111000000__**~0|  Savings: 4306   Discomfort: 175
|11111111111111**111100000000~~1_11~~00000~~~~~11|  Savings: 4938   Discomfort: 19
```

Maximum Isolation of Zones
Average Percent Savings: 7.3% Total discomfort degree minutes: 194

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	2	3	2
1	2	1	1
1	1	2	2

Last Classes Used:

1	2	1	1
1	2	3	2
1	3	1	2
1	2	2	2

Final Classifications:

1	1	1	2
1	2	1	2
1	2	1	1
1	3	4	2

```
Present date:  5/31/1981
Pattern to be predicted:

|00000000000000000001000000000000000000011111111111|
|00000000000000001000000001000001111000000000000|
|00000000000000001110000000000000111101100000001|
|11111111111111110000000000000000111101000000000|

Performance:

|000000000000~~~~~1_*0000~~~~~~~~0~~11*111111111*|  Savings: 3473   Discomfort: 28
|0000000000~~~~111~~~~~00111~~~1_*111~~~~~~~~~~~0|  Savings: 2453   Discomfort: 6
|0000000000~~~~111110000000000~111111111~~~~~~111|  Savings: 5295   Discomfort: 0
|1111111111111111~~~00000~~~~00__*111_*0~~~~00~11|  Savings: 4227   Discomfort: 46

                    Maximum Isolation of Zones
 Average Percent Savings: 7.6%     Total discomfort degree minutes: 80

                    Minimum Isolation of Zones
 Average Percent Savings: 4.7%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   2   1   2  |                       |  2   2   1   2  |
  |  2   2   1   2  |                       |  2   2   3   2  |
  |  3   2   1   1  |                       |  3   2   1   1  |
  |  1   2   1   2  |                       |  1   2   2   3  |

                        Final Classifications:

                         |  2   2   1   1  |
                         |  2   2   2   2  |
                         |  2   2   2   1  |
                         |  1   1   2   3  |
```

```
Present date:  6/1/1981     school day
Pattern to be predicted:

|00000000000000000000000000000000000000000000000000|
|00000000000010000000000000000000000000000000001111|
|00000000000011100000000000000000000000000000001100|
|11111111111100000000000000000000000000000000001100|


Performance:

|00000000000~~~~~~~~000000000000000000000000~~~~~~~|  Savings: 7267   Discomfort: 0
|00000000~~111~000000000000~~~~~~~~~~000~~~111111|  Savings: 5648   Discomfort: 0
|~000000000011111~0000000000000~~~~~~~~~~~~~1111~~|  Savings: 5215   Discomfort: 0
|111111111111~~~~~~~~000000000000000000000~~111111|  Savings: 7137   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 12.4%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 6.9%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  1   2   1   1  |                     |  1   2   1   1  |
  |  2   2   3   1  |                     |  2   2   3   1  |
  |  1   2   1   2  |                     |  1   2   1   2  |
  |  1   3   1   1  |                     |  1   1   1   1  |

                        Final Classifications:

                         |  3   1   1   1  |
                         |  2   2   1   1  |
                         |  3   2   1   1  |
                         |  2   1   1   1  |
```

```
Present date:  6/2/1981
Pattern to be predicted:

|0000000000000000000000000000000000000000000000000000|
|0000000000001100000000000000010011010111111111111111|
|0000000000000111000000000000001001111000000011000|
|1111111111110000000000000000011111111100000000001|


Performance:

|0000000000~~~~~~0000000000000000000~~~~~~~~~~~~~~~~|  Savings: 7090   Discomfort: 0
|00000000~~1111~~00000000000_1111111111111111111**|  Savings: 5041   Discomfort: 0
|000000000~~1111100000000000__*111111~~~~~1111~~~~|  Savings: 5219   Discomfort: 50
|111111111111~~~~0000000000011111111111~000000111|  Savings: 5773   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 11.3%    Total discomfort degree minutes: 50

                    Minimum Isolation of Zones
 Average Percent Savings: 9.2%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   1   1   1  |
  |  1   2   2   3  |                       |  1   2   2   3  |
  |  1   2   1   1  |                       |  1   2   2   1  |
  |  1   2   2   3  |                       |  1   2   2   3  |

                        Final Classifications:

                         |  1   1   1   1  |
                         |  1   1   2   3  |
                         |  1   1   2   1  |
                         |  1   1   2   3  |
```

```
Present date:  6/3/1981     school day
Pattern to be predicted:

|00000000000000000000000000000000000000000000000000|
|00000000001100000000000000000000000000000000000111|
|00000000000111000000000000000000000000000000001100|
|11111111111000000000000000000000000000000000000000|


Performance:

|000000000~~~~~0000000000000~~~~~~000000000~~~~~|  Savings: 7285   Discomfort: 0
|00000000~1111~~~00000000~~~000000000000~~~~11111|  Savings: 6630   Discomfort: 0
|000000000~11111~000000000000~~~~~~~~00~~~~1111~~|  Savings: 5172   Discomfort: 0
|111111111111~~0000000000~~~~~0000000000000~~~~~11|  Savings: 8050   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 13.3%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 8.1%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   1   1  |                       |  1   1   1   1  |

                        Final Classifications:

                        |  1   1   1   1  |
                        |  1   2   1   1  |
                        |  1   2   1   1  |
                        |  1   2   1   1  |
```

Present date: 6/4/1981
Pattern to be predicted:

```
|00000000000000000000000000000000000000000000000000|
|00000000001100000000000000000011011111110011011|
|00000000000011000000000000000000100000000011000|
|11111111111000000000000000000001110000000011111|
```

Performance:

```
|000000000~~~~~~0000000000000000000000000000~~~~~|  Savings: 9336   Discomfort: 0
|0000000001111~~~00000000~~~~0__*1111***1111**_*1|  Savings: 4794   Discomfort: 117
|000000000~~1111~00000000000000_11~~~~00~~1111~~~|  Savings: 5983   Discomfort: 0
|111111111111~~~000000000000000__*11~00000_111111|  Savings: 7845   Discomfort: 50
```

Maximum Isolation of Zones
Average Percent Savings: 13.7% Total discomfort degree minutes: 167

Minimum Isolation of Zones
Average Percent Savings: 8.8% Total discomfort degree minutes: 109

Predicted Classes:

1	1	1	1
1	2	1	1
1	2	1	1
1	2	1	1

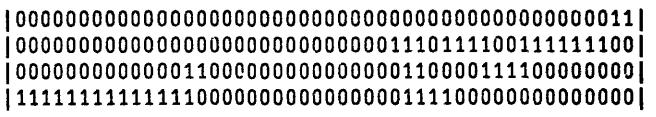
Last Classes Used:

1	1	1	1
1	2	2	3
1	2	1	1
1	2	2	1

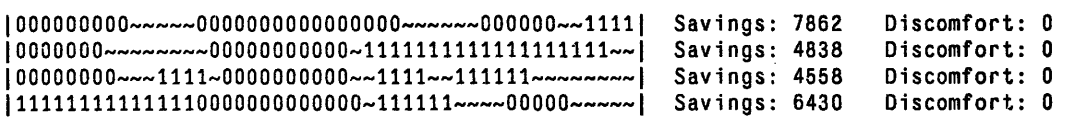
Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	3

Present date: 6/5/1981
Pattern to be predicted:



Performance:



```
|000000000~~~~~0000000000000000~~~~~~000000~~1111|  Savings: 7862   Discomfort: 0
|0000000~~~~~~~~00000000000~1111111111111111111~~|  Savings: 4838   Discomfort: 0
|00000000~~~1111~0000000000~~1111~~111111~~~~~~~~|  Savings: 4558   Discomfort: 0
|1111111111111100000000000000~111111~~~~00000~~~~~|  Savings: 6430   Discomfort: 0
```

Maximum Isolation of Zones
Average Percent Savings: 11.6% Total discomfort degree minutes: 0

Minimum Isolation of Zones
Average Percent Savings: 8.7% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	1	3	3
1	1	1	1
1	1	2	3

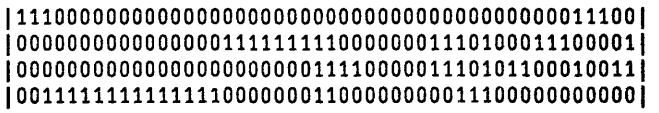
Last Classes Used:

1	1	1	1
1	1	3	3
1	1	1	1
1	1	2	3

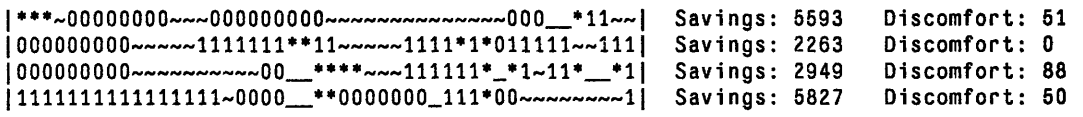
Final Classifications:

1	1	1	1
1	2	2	3
1	2	1	1
1	2	2	3

Present date: 6/6/1981
Pattern to be predicted:



Performance:



Maximum Isolation of Zones
Average Percent Savings: 8.1% Total discomfort degree minutes: 189

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	2	1	2
1	2	1	1
1	2	2	1

Last Classes Used:

3	1	1	2
1	0	2	2
1	2	0	0
1	2	2	3

Final Classifications:

3	2	1	2
1	2	3	2
1	2	1	2
1	2	2	2

Present date: 6/7/1981
Pattern to be predicted:

```
|00000000000000000000000000000000000000000001111110|
|11000000000000001001110011000000000000001001000011|
|00000000000000000000001101100000000000000111000000|
|01111111111111111111000011000000000000000001000001|
```

Performance:

```
|00000000000~0~~~~~~~~000~~~~~~~~~~~0~~~11111111~|  Savings: 3151   Discomfort: 0
|**0~~~~~~~~~~~1111111111**0~~~~~~~00111111~~1111|  Savings: 2430   Discomfort: 0
|~~0000000~~~~~~~~~~11*11**0~~~~~~~~~0__*11~~~~~~|  Savings: 2762   Discomfort: 58
|1111111111111111111~~11**0~~~~~~000000__*~~~111|  Savings: 3671   Discomfort: 38
```

Maximum Isolation of Zones
Average Percent Savings: 5.9% Total discomfort degree minutes: 96

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	1
2	2	3	2
3	2	1	1
1	2	2	2

Last Classes Used:

2	2	1	1
0	2	1	2
3	3	0	2
1	2	1	3

Final Classifications:

2	2	1	2
2	2	1	2
3	2	1	2
1	2	1	4

```
Present date:  6/8/1981     school day
Pattern to be predicted:

|00000000000000000000000000000000000000000000000000|
|00000000000110000000000000000000000000000000001101|
|00000000000111000000000000000000000000000000001110|
|11111111111100000000000000000000000000000000001100|


Performance:

|000000000000~~00000000000000000000000000~~~~~~~~|  Savings: 8914   Discomfort: 0
|000000000_111~~~00000000~~00000000000000~~111111|  Savings: 7138   Discomfort: 0
|000000000011111~0000000000000~~~~~~~~~~~~~111111|  Savings: 5260   Discomfort: 0
|111111111111~~~~~00000000000000000000000~~111111|  Savings: 7845   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 14.3%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 8.2%     Total discomfort degree minutes: 0

   Predicted Classes:                      Last Classes Used:

  |  2   2   1   1  |                     |  2   2   1   1  |
  |  2   2   1   1  |                     |  2   2   1   1  |
  |  3   2   1   1  |                     |  3   2   1   1  |
  |  1   1   1   1  |                     |  1   1   1   1  |

                          Final Classifications:

                           |  2   1   1   1   |
                           |  2   2   1   1   |
                           |  3   2   1   1   |
                           |  1   1   1   1   |
```

```
Present date:  6/9/1981
Pattern to be predicted:

|0000000000000000000000000000000000000000000000000000|
|0000000000011000000000000000001110011111111111111111|
|1000000000011111000000000000000010000000000000000000|
|1111111111110000000000000000000011100000000000000000|


Performance:

|000000000~~~~~000000000000000000000~~000~~~~~~~~|  Savings: 8608   Discomfort: 0
|00000000~1111~~000000000000011111111111111111111|  Savings: 5685   Discomfort: 0
|*0000~~~~111111~000000000000~~111~~~~~~~~~~~~~~~|  Savings: 4968   Discomfort: 0
|111111111111~~000000000000~~~~11111~~~~00000~~11|  Savings: 6026   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 12.4%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 8.7%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  | 2   1   1   1 |                         | 2   1   1   1 |
  | 1   2   2   3 |                         | 1   2   2   3 |
  | 3   2   2   1 |                         | 1   2   2   1 |
  | 1   1   2   3 |                         | 1   1   2   3 |

                          Final Classifications:

                           | 1   1   1   1 |
                           | 1   1   2   3 |
                           | 1   1   2   1 |
                           | 1   1   2   3 |
```

```
Present date:  6/10/1981      school day
Pattern to be predicted:

|000000000000000000000000000000000000000000000000000000|
|100000000000100000000000000000000000000000000001111|
|000000000000111000000000000000000000000000000001110|
|111111111111000000000000000000000000000000000001110|


Performance:

|000000000~~~~~000000000000000000000000000000000~|  Savings: 9776   Discomfort: 0
|*00~~~~~~~~111~~~00000000~~~0000000000000~~111111|  Savings: 6840   Discomfort: 0
|000000000~11111~000000000000~~~~~~~~~~00~~~11111~|  Savings: 5148   Discomfort: 0
|111111111111~~0000000000~~000000000000~000111111|  Savings: 8352   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 14.8%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 8.0%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   1  |
  |  1   2   1   1  |                       |  0   2   1   1  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   1   1   3  |                       |  1   1   1   3  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   2   1   1   |
                         |  1   2   1   1   |
                         |  1   2   2   1   |
```

```
Present date:  6/11/1981
Pattern to be predicted:

|0000000000000000000000000000000000000000000000000000|
|0000000000011100000000000000000000010110011111011|
|0000000000001110000000000000000000010111010000110|
|1111111111110000000000000000000000011001010000000|


Performance:

|000000000~~~~~000000000000000000000000000000~~~~|  Savings: 10754   Discomfort: 0
|00000000~11111~~00000000~~00~~~~1111111111111111|  Savings: 4706    Discomfort: 0
|00000~~~~~11111~000000000000~~~~111111111~~1111~|  Savings: 5017    Discomfort: 0
|111111111111~~000000000000000000__*1111_*000~~11|  Savings: 8010    Discomfort: 61

                    Maximum Isolation of Zones
 Average Percent Savings: 14.0%    Total discomfort degree minutes: 61

                    Minimum Isolation of Zones
 Average Percent Savings: 8.4%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   1   1   1  |
  |  1   2   2   3  |                       |  1   2   2   3  |
  |  3   2   2   1  |                       |  3   2   2   1  |
  |  1   1   1   3  |                       |  1   1   4   3  |

                        Final Classifications:

                          |  1   1   1   1  |
                          |  1   1   2   3  |
                          |  1   1   2   1  |
                          |  1   1   2   3  |
```

```
Present date:  6/12/1981
Pattern to be predicted:

|00000000000000000000000000000000000000000011111100|
|00000000000010000000000000000011001111010110000011|
|00000000000011100000000000000011001101111000000100|
|11111111111110000000000000000011111101111000000000|


Performance:

|000000000~~~~~000000000000000000000~~00__*11111~0|  Savings: 8826   Discomfort: 75
|00000000~~111~~~00000000~~~0__*1111111111~~~1111|  Savings: 4482   Discomfort: 41
|000000000~11111~~00000000000__1111111*11~~~111~0|  Savings: 5160   Discomfort: 50
|1111111111111~~~~~0000000000_1111111111*0000~~~~|  Savings: 5437   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 11.7%    Total discomfort degree minutes: 166

                    Minimum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  |  1   1   1   1  |                       |  1   1   1   2  |
  |  1   2   1   3  |                       |  1   2   3   3  |
  |  1   2   1   1  |                       |  1   2   1   1  |
  |  1   2   2   3  |                       |  1   1   2   3  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   2   2   3   |
                         |  1   2   1   1   |
                         |  1   2   2   3   |
```

```
Present date:  6/13/1981
Pattern to be predicted:

|00000000000000000011111000000000000000000000000000|
|11000000000000001000001111111111111111111101111111|
|00000000000000000000100000000000011000011000000000|
|00111111111111111111000000000100000000000000000000|


Performance:

|0000000000~~~~~~11111110~~~~~~~~~~~~~~~~~~~~~~~~~|  Savings: 2909   Discomfort: 0
|**0~~~~~~~~~~0~111~~~1111***11111111111111111111|  Savings: 2370   Discomfort: 0
|000000000~~~~~~~~~11*~~~0000~~~1111~0__*1~~~~~00|  Savings: 3374   Discomfort: 32
|11111111111111111~000000__*~~~~~~~0000000~~~~~|  Savings: 5185   Discomfort: 75

                    Maximum Isolation of Zones
 Average Percent Savings: 6.8%     Total discomfort degree minutes: 107

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  | 1  2  1  1 |                          | 1  2  1  1 |
  | 1  2  3  2 |                          | 0  2  1  2 |
  | 1  2  1  1 |                          | 1  3  1  2 |
  | 1  2  2  2 |                          | 1  2  1  2 |

                        Final Classifications:

                          | 3  2  1  2 |
                          | 3  2  3  2 |
                          | 1  3  1  2 |
                          | 2  3  3  2 |
```

Present date: 6/14/1981
Pattern to be predicted:

```
|00000000000000000000000010000000000000000000000|
|11000000000000001111100101101111111001111111111|
|00000000000000001100000111000000001111000010010|
|00111111111111111100000100110000000000000000000|
```

Performance:

```
|00000000000~00~~~~~~00__1~~~~~~~~~~~0~~~~~~~~~~~|  Savings: 3292   Discomfort: 54
|**0~00000000~~111111111*11111111111111111111111|  Savings: 2448   Discomfort: 0
|000000000000~~1111~~~__*11~~~~~~111111~~111~1_*0|  Savings: 3297   Discomfort: 25
|1111111111111111~~0__*1111~~~~~~~~00000000~~11|  Savings: 4223   Discomfort: 25
```

Maximum Isolation of Zones
Average Percent Savings: 6.5% Total discomfort degree minutes: 103

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	1
2	2	3	2
3	2	1	2
1	2	1	2

Last Classes Used:

2	2	1	1
3	2	3	2
3	2	1	2
1	2	1	2

Final Classifications:

2	2	1	2
2	2	3	2
3	2	1	2
3	2	1	2

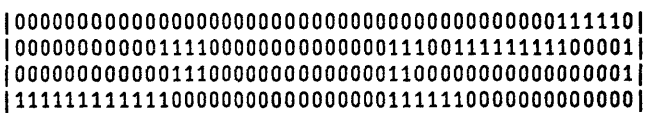
```
Present date:  6/15/1981     school day
Pattern to be predicted:

|000000000000000000000000000000000000000000000000000000|
|000000000000000000000000000000000000000000000000000111|
|000000000000011000000000000000000000000000000000000111|
|111111111111110000000000000000000000000000000000000110|


Performance:

|000000000~~~~~000000000000000000000000000000000000000|  Savings: 9362   Discomfort: 0
|000000000~~~~~~~~~~00000~~~~~~~~~~~000000~~11111|  Savings: 3860   Discomfort: 0
|0000000000~1111~~00000000000~~~~~~~~~~~~~~~~11111|  Savings: 4964   Discomfort: 0
|11111111111111~~~~000000~~~~0000000000~~~~~1111~|  Savings: 6117   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 11.9%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 5.9%     Total discomfort degree minutes: 0

   Predicted Classes:                        Last Classes Used:

  |  2   1   1   1  |                       |  2   1   1   1  |
  |  2   2   3   1  |                       |  2   2   1   1  |
  |  3   2   1   1  |                       |  3   2   1   1  |
  |  1   1   1   3  |                       |  1   1   1   3  |

                        Final Classifications:

                         |  2   2   1   1  |
                         |  2   2   1   1  |
                         |  3   2   1   1  |
                         |  3   2   2   1  |
```

```
Present date:  6/16/1981
Pattern to be predicted:

|0000000000000000000000000000000011000000000000000|
|0000000000000011111111111101111111111011110011|
|1100000000000001100010100000000100100000011000010|
|0011111111111110000000000111000000001111111001110|


Performance:

|0000000000~~~~~~~0000000~0000__*1~~~~~~~00000~~~|  Savings: 7484   Discomfort: 50
|0000000000~~111111****11**_111111111*_1111111111|  Savings: 2594   Discomfort: 16
|**0~~~~~~~~~~1111~1_*11~~0000__*111~0~__*1~~111~|  Savings: 3533   Discomfort: 71
|111111111111111~~000000__**1~~0000__**111111*111|  Savings: 5282   Discomfort: 150

                    Maximum Isolation of Zones
 Average Percent Savings: 9.3%     Total discomfort degree minutes: 287

                    Minimum Isolation of Zones
 Average Percent Savings: 4.3%     Total discomfort degree minutes: 0

  Predicted Classes:                    Last Classes Used:

  |  1   1   1   1  |                   |  1   1   0   1  |
  |  1   1   2   3  |                   |  1   0   1   3  |
  |  1   2   2   1  |                   |  0   0   1   2  |
  |  1   1   2   3  |                   |  1   1   1   3  |

                        Final Classifications:

                         |  1   1   1   2  |
                         |  1   1   3   2  |
                         |  2   1   2   2  |
                         |  1   1   2   2  |
```

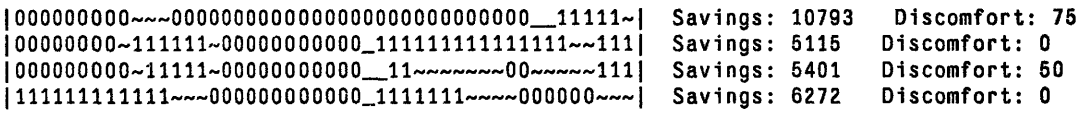
```
Present date:  6/17/1981      school day
Pattern to be predicted:

|00000000000000000000000000000000000000000000000000|
|00000000000010000000000000000000000000000000001111|
|00000000000011000000000000000000000000000000001110|
|11111111111100000000000000000000000000000000000000|


Performance:

|~00000000~~~000000000000000000000000000000000000000|  Savings: 10607   Discomfort: 0
|0000000000111~~~~~000000~~00000000000000~~111111|  Savings: 6462    Discomfort: 0
|~000000000~1111~00000000~~~~~~~~~~~~~~~~~~~111111|  Savings: 4042    Discomfort: 0
|1111111111111~~~00000000~0000000000000000~~~~~~1|  Savings: 7754    Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 14.1%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 6.5%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

  |  1   1   1   1  |                     |  1   1   1   1  |
  |  1   2   1   1  |                     |  1   2   1   1  |
  |  1   2   1   1  |                     |  1   2   1   1  |
  |  1   1   1   1  |                     |  1   1   1   1  |

                          Final Classifications:

                           |  2   2   1   1   |
                           |  2   2   1   1   |
                           |  3   2   1   1   |
                           |  3   2   1   3   |
```

```
Present date:  6/18/1981
Pattern to be predicted:

|00000000000000000000000000000000000000000000000011|
|00000000000010000000000000000000000000000000001101|
|10000000000011100000000000000000000000000000011110|
|01111111111100000000000000000000000000000000000000|


Performance:

|0000000000~~00000000000000000000000000000__*1|  Savings: 11872   Discomfort: 75
|000000000~111~~~~000000000000~~~~~~~~~~000~111111|  Savings: 5639   Discomfort: 0
|*000000000__111~~000000000000~~~~~~~~~~~~111111~|  Savings: 5198   Discomfort: 75
|111111111111~~~000000000000~~~~~~~~~~~~~~0000~~11|  Savings: 5906   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 14.0%    Total discomfort degree minutes: 150

                    Minimum Isolation of Zones
 Average Percent Savings: 8.6%     Total discomfort degree minutes: 0

  Predicted Classes:                      Last Classes Used:

 |  1   1   1   1  |                     |  1   1   1   2  |
 |  1   1   2   1  |                     |  1   1   2   1  |
 |  1   1   2   1  |                     |  2   1   2   1  |
 |  1   1   2   3  |                     |  1   1   2   3  |

                        Final Classifications:

                         |  1   1   1   1   |
                         |  1   1   1   1   |
                         |  1   1   2   1   |
                         |  1   1   1   1   |
```

Present date: 6/19/1981
Pattern to be predicted:



Performance:



Maximum Isolation of Zones
Average Percent Savings: 13.5% Total discomfort degree minutes: 125

Minimum Isolation of Zones
Average Percent Savings: 9.7% Total discomfort degree minutes: 0

Predicted Classes:

1	1	1	1
1	2	2	3
1	2	1	1
1	1	2	3

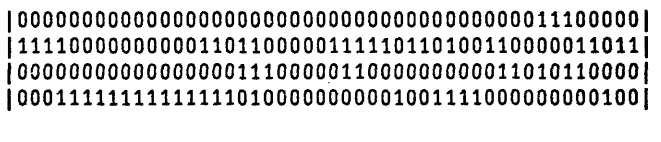
Last Classes Used:

1	1	1	1
1	2	2	3
1	2	1	1
1	1	2	3

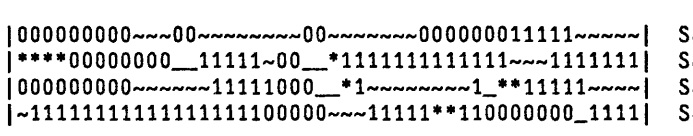
Final Classifications:

1	1	1	1
1	1	2	3
1	1	2	1
1	1	2	3

Present date: 6/20/1981
Pattern to be predicted:



Performance:



Savings: 4643	Discomfort: 0
Savings: 3673	Discomfort: 192
Savings: 3168	Discomfort: 67
Savings: 4836	Discomfort: 0

Maximum Isolation of Zones
Average Percent Savings: 8.0% Total discomfort degree minutes: 259

Minimum Isolation of Zones
Average Percent Savings: 4.7% Total discomfort degree minutes: 45

Predicted Classes:

1	2	1	1
1	2	3	2
1	2	1	1
1	3	1	2

Last Classes Used:

1	2	1	1
3	1	3	2
1	2	0	2
1	3	0	2

Final Classifications:

3	2	1	2
3	2	3	2
1	2	1	2
2	3	3	2

Present date: 6/21/1981
Pattern to be predicted:

```
|00000000000000000000000000000000000000000000000110|
|00000000000000001110010000011011011111111100001|
|00000000000000000000110000001001110000110001100|
|11111111111111110110010000001110000000000011000|
```

Performance:

```
|0000000000~~~~~~~~~~~000~~0000000~~~0~~~~~~11**~|  Savings: 4202   Discomfort: 0
|00000000~~~~~~11111111~0~1111111111111111~~111|  Savings: 2262   Discomfort: 0
|000000000~~~~~~~~~1_**~~00__111111~~~1111~1111~~|  Savings: 2836   Discomfort: 52
|1111111111111111111_*~~~~__*11~~0000000~1111~11|  Savings: 4241   Discomfort: 21
```

Maximum Isolation of Zones
Average Percent Savings: 6.6% Total discomfort degree minutes: 73

Minimum Isolation of Zones
Average Percent Savings: 4.3% Total discomfort degree minutes: 0

Predicted Classes:

2	2	1	2
2	2	3	2
3	2	1	2
1	2	2	2

Last Classes Used:

2	2	1	1
2	2	3	2
3	0	1	2
1	1	1	2

Final Classifications:

2	2	1	2
2	2	3	2
3	2	1	2
3	2	3	2

```
Present date:  6/22/1981     school day
Pattern to be predicted:

|00000000000000000000000000000000000000000000000000|
|00000000000011000000000000000000000000000000000111|
|00000000000011000000000000000000000000000000000111|
|11111111111100000000000000000000000000000000000111|


Performance:

|000000000~~~~~~0000000000000000000000000000~~~~~|  Savings: 8717   Discomfort: 0
|00000000~~1111~~~0000000~~~~~~~~~~000000~~~11111|  Savings: 4508   Discomfort: 0
|000000000~~1111~000000000~~~~~~~~~~~~~~~~~~11111|  Savings: 4182   Discomfort: 0
|111111111111~~~000000000~~0~~~~~~~~~00000~~11111|  Savings: 5298   Discomfort: 0

                    Maximum Isolation of Zones
 Average Percent Savings: 11.1%    Total discomfort degree minutes: 0

                    Minimum Isolation of Zones
 Average Percent Savings: 7.1%     Total discomfort degree minutes: 0

  Predicted Classes:                        Last Classes Used:

  | 2  2  1  1 |                            | 2  2  1  1 |
  | 2  2  3  1 |                            | 2  2  1  1 |
  | 3  2  1  1 |                            | 3  2  1  1 |
  | 1  1  2  1 |                            | 1  1  2  1 |

                        Final Classifications:

                         | 2  2  1  1 |
                         | 2  2  1  1 |
                         | 3  2  1  1 |
                         | 3  2  1  1 |
```

```
Present date:  6/23/1981
Pattern to be predicted:

|00000000000000000000000000000000000000000000011111|
|00000000000010100000000000000011011000000000000001|
|00000000000011100000000000000011000011110010000l|
|11111111111100000000000000000111011100111110001|


Performance:

|000000000~~~00000000000000000000000000001111111|  Savings: 10025   Discomfort: 0
|00000000~~11111~00000000000_111111~~~~~~~~~~~111|  Savings: 5186   Discomfort: 0
|000000~~~~11111~0000000000001111~~111111111~~111|  Savings: 4941   Discomfort: 0
|111111111111~~~00000000000~~111111111111****~111|  Savings: 4998   Discomfort: 0

                        Maximum Isolation of Zones
 Average Percent Savings: 12.3%    Total discomfort degree minutes: 0

                        Minimum Isolation of Zones
Average Percent Savings: 8.7%     Total discomfort degree minutes: 0
```

REFERENCES

Beckey, T., and L. Nelson, *Field Test of Energy Savings With Thermostat Setback*, ASHRAE Journal, Sept. 1978, pp. 67-70.

Bonne, U., J. E. Janssen, L. W. Nelson, and R. H. Torborg, *Control of Overall Thermal Efficiency of Combustion Heating Systems*, Proceedings of the Sixteenth International Symposium on Combustion at M.I.T., August 1976.

Drake, A. W., *Fundamentals of Applied Probability Theory*, McGraw–Hill, New York, 1967.

Edwards, Anthony Wlliam Fairbank, *Likelihood; An Account of the Statistical Concept of Likelihood and its Application to Scientific Inference*, Cambridge University Press, 1972.

Egan, M. David, *Concepts in Thermal Comfort*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975.

Lindley, D. V., *Bayesian Statistics, a Review*, University College, London, Siam: Philadelphia, PA., 1972.

Minsky, M. L., and S. Papert, *Perceptrons*, The MIT Press, Cambridge, Massachusetts, 1969.

Minsky, M. L., and O. G. Selfridge, *Learning in Random Nets*, Proceedings of the Fourth London Symposium on Information Theory, pp. 335-347, (edited by: C. Cherry), Academic Press: New York, 1961.

Nelson, L., and W. MacArthur, *Energy Savings Through Thermostat Setback*, ASHRAE Journal, Sept. 1978, pp. 49-54.

Papert, S., *Some Mathematical Models of Learning*, Proceedings of the Fourth London Symposium on Information Theory, pp. 353-363, (edited by: C. Cherry), Academic Press: New York, 1961.

Samuel, A. L., *Some Studies in Machine Learning Using the Game of Checkers*, IBM Journal of Research and Development, July 1959, pp. 211-229.

Saridis, G. N., *Toward the Realization of Intellegent Controls*, Proceedings of the IEEE, Vol. 67, No. 8, August 1979, pp. 1115-1133.

Shortliffe, E. H., and B. G. Bunchman, *A Model of Inexact Reasoning in Medicine*, Mathematical Biosciences, Vol. 23, pp. 351-379, 1975.

Tobias, J. R., *Energy Conservation With a Residential Zoning System*, Unpublished Internal White Paper, Honeywell, Inc. Residential Division, June 1975.