Process Control Decision Inference, Monitoring, and Execution

Robert Matania[1], Jean-Marie Forêt[2], Vicente Camarillo[3,] and Mark Walker[4]

[1,2,3,4]*D2K Technologies LLC, Oceanside, CA, 92056, USA*

robert.matania@d2ktech.com
jeanmarie.foret@d2ktech.com
vicente.camarillo@d2ktech.com
mark.walker@d2ktech.com

ABSTRACT

Approaches for inferring process control system decision trees from plant data have been heavily researched and demonstrated, but the utility of applying such decision tree inferences for autonomously monitoring, guiding, and executing control philosophies has been lacking. The authors have implemented an architecture leveraging model-based reasoning and graphical programming that investigates the utility of using control decision tree inferencing for validating, monitoring, and executing control strategies for both simple and complex process control problems. The techniques are potentially useful for assessing process control health, as well as extracting process control knowledge that may exist in daily operations but not recognized or well understood by analysts and management. Plant operators and managers can readily employ such insights for improving, augmenting, and extending process control system behavior, but can also potentially employ the refined and validated decision trees as a supervisory control layer on top of their existing control systems.

1. INTRODUCTION

Decision trees are a simple but powerful machine learning technique used for classification and regression based on a supervised learning algorithm (Quinlan, 1986). They predict values of a single target variable by applying decision rules to a set of input variables that influence the prediction. Like many other machine learning approaches, decision trees are reasonably good at reliably inferring decision boundaries from properly labeled data. Unlike other machine learning techniques such as artificial neural networks, decision trees capture and present the inferred decision logic in a form that is understandable by humans.

Once a decision tree has been formulated (e.g. inferred from data), the same decision tree can be used operationally to identify undesirable control decisions and/or, as the authors have been researching, automate validated control system actions based on monitored plant parameters and states. This makes decision trees an interesting and promising machine learning tool for applied Prognostics and Health Management (PHM) research.

As the name implies, a decision tree is a tree-like structure that displays the decision algorithm by showing the relationships between the target variable and the input variables.

Decision trees require a set of data examples in order to create the algorithm in a similar way to other machine learning techniques such as artificial neural networks, but they are easier to understand. A major advantage of decision trees over other machine learning techniques is the ability for a domain expert to view the relationships. It is also possible to generate executable code representing the decision tree. Due to their simplicity, decision trees are used in many different industries such as finance, e-commerce, law, insurance, manufacturing, batch processing, petroleum, chemical and pharmaceutical (Patel and Rana, 2014). In general, decision trees are used statically to predict an outcome based on a set of input values.

This paper introduces the basic principles of decision trees and describes the implementation of a decision tree algorithm within in a real-time simulation application for determining the appropriate actions to be taken to control the temperature of a room.

2. DECISION TREE PRINCIPLES

The following section outlines some basic decision tree principles and provides a background for the application of decision tree inferencing in industry.

2.1. Structure

Graphically, a decision tree is typically drawn upside down, consisting of a series of nodes starting from a single root node representing the initial decision to be taken as shown in Figure 1.

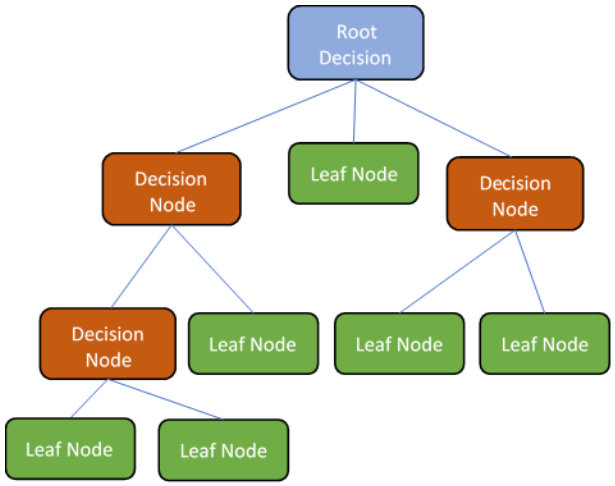


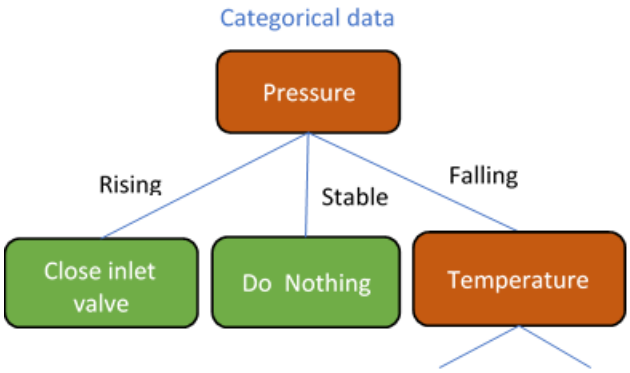
Figure 1. General Decision Tree Structure.

Branches downwards from the root node represent possible decisions. These branches are known as feature splits. The tree continues splitting on decisions until it ends with a terminal or leaf node where no more choices can be made. The leaf node represents the final decision or proposed action.

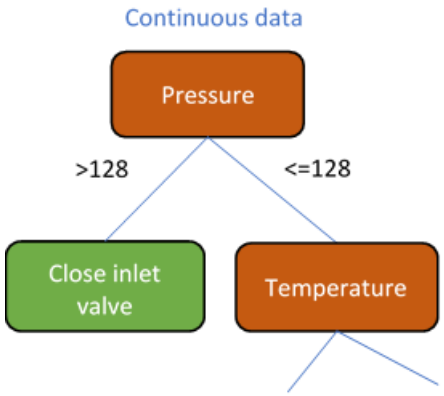
2.2. Decision Tree Types

Depending on the algorithm used to construct the decision tree, one or both of the types of data shown in Figure 2 can be handled:

- Categorical Data: Example: hot, cold, high, low...
- Continuous Data: parametrical or numerical values such as a temperature, cost…



a)



b)

Figure 2. Decision Tree Types.

Different decision tree algorithms have been developed in the past 35 years with improvements in performance and in the types of data that can be handled. Several of these are listed in Table 1 (Ezzikouri and Fakir, 2010). The first is the ID.3 (Interactive Dichotomizer 3) algorithm. Also heavily researched are the CART (Classification and Regression Tree) and C4.5/5.0 algorithms (extensions of the ID.3 algorithm). Differences in performance have been studied and documented (Mohan, 2013 and Cinaroglu, 2016), with additional distinctions associated with the support of both two-way and multi-way splitting, the ability to create general tress vs. binary-only trees, the ability to infer over symbolic (categorical) attributes, and the optimization or splitting function used to determine how to select optimal split points.

Table 1. Common Decision Tree Algorithms.

Algorithm	**Attributes**	**Splitting**	**Splitting formula**
ID3	Categorical	Multi-way	Gini
CART	Categorical and Continuous	2-way	Entropy
C4.5/C5.0	Categorical and Continuous	Multi-way	Entropy

The CART algorithm is based on the Random Forest approach, which generates multiple decision trees and selects the optimal tree based on majority vote. As in all search algorithms, finding an optimal solution can be time consuming – with limited certainty as to the goodness of any solution – and different algorithmic parameterizations and approaches can produce differences in the required processing time. The C4.5/5.0 algorithm has been chosen for by the authors for this published work due to its demonstrated

performance, ability to work well with many types of problems, handle both numeric and symbolic data data types, and take advantage of multi-way splitting (Yobero, 2018).

2.3. Decision Tree Creation

Building a decision tree requires a set of training data containing groups of input variables (features) and corresponding target variables. The use of symbolic categories/values for each input variable helps reduce the size of the tree and helps extend human understanding, but continuously valued numerical inputs are also possible. The approach is to use the values of the features to recursively split the data set into smaller and smaller subsets (Rokach and Maimon, 2010).

The decision tree algorithm starts with the entire training set and must determine which feature to initially split on. Each feature is tested to determine which one has the greatest effect on the target variable. The feature with the greatest effect is chosen for the initial split.

The data set is divided into subsets based on the chosen split. In the case of categorical data, a single subset would contain the same value for the feature. For example, if the first split is on the feature "comfort level", then there might be one subset with comfort level = 'too hot', one subset with comfort level = 'too cold' and a third subset with comfort level = 'ok'. In the case of continuous data, the algorithm would determine the best ranges for that feature. For example, if the split is on the feature "pressure", the subsets could be with pressure '<50' and pressure '>=50'.

The next level spits are determined using each of the subsets and are tested in the same way as the entire dataset. The algorithm then continues to split the data into smaller and smaller subsets, choosing the best candidate each time until either of the following:

- All features give the same target value.
- There are no more features to split among

3. C5.0 ALGORITHM

The C5.0 algorithm uses the concepts of entropy and information gain to determine how to split the data. For a target variable with *n* decision classification categories, the entropy is calculated using Eq. (1).

$$\text{Entropy} = S = \sum_{i=1}^{n} -p_i \, log_n(p_i) \qquad (1)$$

where p_i represents the probability that an input pattern fits into to the i^{th} split.

Entropy indicates the level of impurity of a set of data (a measure of difficulty in separating the classes). The higher the entropy, the higher the impurity. This is represented graphically in Figure 3.

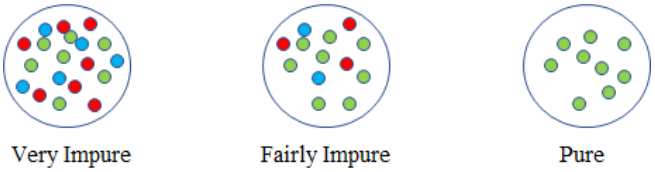


Figure 3. Decision Tree Data Impurity.

For example, in the context of control decisions, suppose we are given a target (control) variable 'heater' which can have symbolic values "turn on", turn off" or "do nothing". A subset of data which has 1000 rows with 200 corresponding to "turn on", 300 rows corresponding to "turn off", and 500 rows corresponding to "do nothing" would therefore possess entropy as in Eq. (2):

$$S = - 0.2*log_2(0.2) - 0.3*log_2(0.3) - 0.5*log_2(0.5) = \mathbf{1.4854753} \qquad (2)$$

The decision tree algorithm must still calculate which input variable (feature) to split on. To do this, the entropy formula is used again to calculate the information gain (*IG*) for each feature available for further splitting. Each feature's *IG* indicates how well the feature will create a pure group of values after a split on that input variable (*IG* = the expected reduction in entropy created by the split). The *IG* is calculated by the difference between the entropy of the parent node (S(Y)) and the conditional entropy based on the children produced by the split (S(Y|X)). This is captured in Eq. (3)

$$IG = [S\,(\text{Y}) - S\,(\text{Y}|\text{X})] \qquad (3)$$

where the conditional entropy S(Y|X) is determined from Eq. (4).

$$S(\text{Y}|\text{X}) = -\sum_{i=1}^{m}\sum_{\text{j}=1}^{n} p(y_i x_j)\, log_2\left(\frac{p(y_i x_j)}{p(x_j)}\right) \qquad (4)$$

Here, the subscripts are based on *m* branches from the parent node (initial feature) with *n* branches to consider for its children (subsequent feature). Suppose, for the above example of 1000 rows of data in the subset the subsequent feature 'window position' has values: 550 = "open" and 450 = "closed", as indicated in Table 2.

Table 2. Example Dataset Summary.

Window position	Turn Heater on	Turn Heater off	Do nothing
Open	150	200	200
Closed	50	100	300

Given the entropy calculated in Eq. 2, the information gain can be calculated for a possible split on 'Window Position' according to Eq. (5-8):

$$S(\text{'open'}) = -150/550*log_2(150/550) - 200/550*log_2(200/550) - 200/550*log_2(200/550) = \mathbf{1.5726} \quad (5)$$

$$S(\text{'closed'}) = -50/450*log_2(50/450) - 100/450*log_2(100/450) - 300/450*log_2(300/450) = \mathbf{1.2244} \quad (6)$$

$$S(Y|X) = (1.5726*550/1000) + (1.2244*450/1000) = \mathbf{1.4159} \quad (7)$$

$$IG(\text{window position}) = 1.48547 - 1.4159 = \mathbf{0.06955} \quad (8)$$

The algorithm repeats the same calculations for each feature. The decision node will split on the feature that gives the highest gain. The subsets obtained from the split will then be used to determine the subsequent splits and the process continues until one of the two criteria mentioned above are reached.

4. DECISION TREE INFERENCE – EXAMPLE APPLICATION

The authors are currently investigating a real-world operational application using decision trees to infer process control decisions being made by operators that may or may not be optimal for managing the process. Results from this applied research are briefly provided in section 5 but are restricted due to proprietary implications of the data. The algorithmic implementation associated with this application have been applied here as an academic example. Issues with using decision trees with real world plant data are numerous, but many can be investigated effectively through sophisticated simulations.

, issues are discussed to aid in both the understanding and validation of implemented algorithms, a simple (but well understood) hypothetical control system was modelled in software. The chosen application implements the decision logic necessary to monitor, control and manage the inside temperature of a greenhouse by controlling a heating/cooling unit and the position of a window (open/closed). A simple first principles physics model was implemented in order to simulate the thermal behavior of the room. The simulator and the decision tree logic that control the model have been built separately in order to allow for replacement of the simulation model by a data interface that receives data and commands from an external system.

A software object model has been created to simulate the heating and cooling behavior of a simple greenhouse, as depicted in Figure 4.

The simulation includes:

- A rectangular greenhouse structure. The walls are assumed to have a thermal conductance that determines the heat transfer between the external and internal temperatures.
- A window that, when open, modifies the heat transfer rate between the interior and the exterior.
- A reversible air conditioning unit that can either heat or cool the inside of the room. In this simulation, the A/C unit can be in one of three states: OFF, HEATING, or COOLING. With the current implementation, it is not possible to vary the degree of heating or cooling.

4.1. Simulation Model

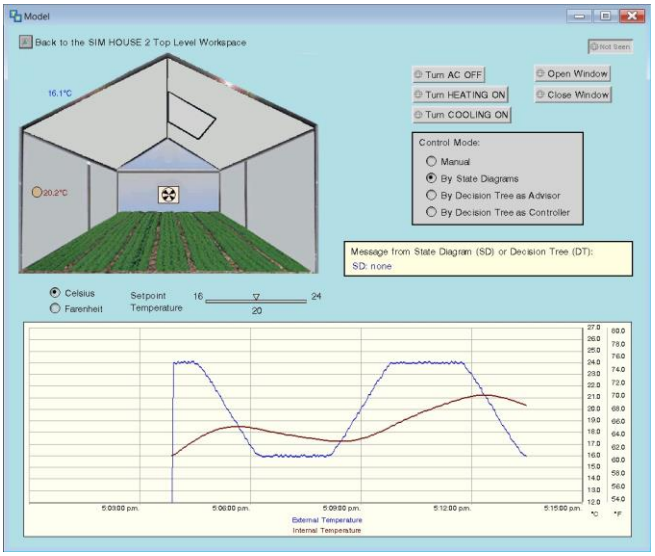


Figure 4. Temperature Control Simulation Model.

The simulator varies the outside temperature cyclically between 16°C and 24°C to simulate day and night as shown by the blue (trapezoidal) graph in Figure 5.

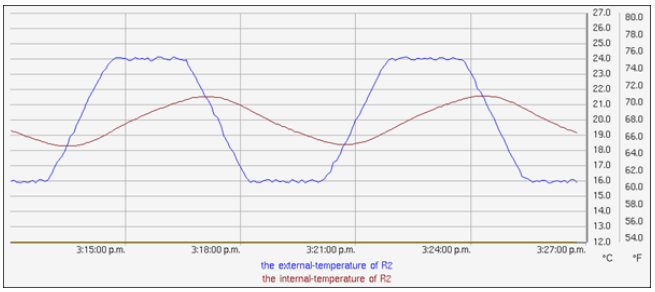


Figure 5. Simulator Temperature Cycles.

The inside temperature (shown by the red or sinusoidal graph) varies as the outside temperature changes because of heat transfer between the room interior and the exterior. The way the interior temperature varies will depend on several factors:

- The difference between the internal and external temperatures,
- Heat transfer through the thermal wall of the room,
- The position of the window (open or closed),
- The state of the A/C unit (off, heating, or cooling)

The model allows the value of a "Comfort Temperature" to be adjusted manually to set the desired inside temperature of the room. The goal of the example application is to infer/use a decision tree to help maintain the actual inside temperature of the room as close as possible to the desired comfort temperature by acting upon the window position and the A/C unit control. The "Comfort Level" value indicates whether the occupant of the room is Too Hot, Too Cold or Comfortable. If the inside temperature is within 0.5°C of the comfort temperature, the comfort level is "Comfortable", otherwise it is "Too Hot" or "Too Cold" depending whether the inside temperature is above or below the comfort temperature. This is summarized by Eq. (8).

$$\begin{aligned} &\text{-Too Hot} = T_{Inside} >= (T_{Comfort} + 0.5) \\ &\text{-Too Cold} = T_{Inside} <= (T_{Comfort} - 0.5) \\ &\text{Comfortable} = (T_{Comfort} - 0.5) <= T_{Inside} <= (T_{Comfort} + 0.5) \end{aligned} \quad (8)$$

4.2. Model Control

The simulation model can be controlled in one of four ways:

- **Manually:** The window is opened or closed, and the state of the A/C unit is set to {'heating', 'cooling', or 'off'}
- **By State Diagrams:** State diagrams control the position of the window and the state of the A/C unit to maintain the inside temperature close to the comfort level. The state diagrams are described in more detail in section 4.2.1 and are used to produce optimum control of the room temperature based on state equations.
- **By Decision Tree (as an advisor):** A decision tree will advise the user what action to take in order to maintain the inside temperature close to the comfort level.
- **By Decision Tree (as a controller):** A decision tree will control the position of the window and the state of the A/C unit to maintain the inside temperature close to the comfort level.

The control of the model by the state diagrams and decision trees, as opposed to the simulation part of the application, is built using an intelligent process software utility developed by the authors as part of a G2© based platform for building intelligent monitoring and control applications for process industries (Walker, 2010). Its principal component is a set of graphical programming (GP) blocks that can be created and configured to perform complex calculations and control functions on process data, leveraging an underlying comprehensive library of coded functions and algorithms.

The GP blocks provided by the utility each implement a specific functionality. Decision trees and state diagrams each have a specific set of such GP blocks. The blocks are associated with an Execution Controller that will decide how and when to execute the blocks.

4.2.1. State Diagram Control

A specific set of generic GP blocks enables the creation of state diagrams while another is dedicated to the creation of decision trees. While the focus of this paper is on decision tree creation, utility, and validation, a discussion of the state diagram functionality is also included since it provides the example application with optimal and verifiable control based on equations. In this case, the state diagrams have also been used to generate the training data set for the decision tree inference and allow the predictions of the tree to be compared with the logic of the optimal state diagram control. In a real-life situation, decision tree intelligence could be leveraged without the need for such mathematical models since the decision tree could be inferred directly from a set of historical data.

In our example, two distinct state diagrams have been implemented. One diagram controls the position of the window and the other controls the state of the A/C unit. Note: based on our implementation, the model cannot be controlled simultaneously by both the state diagram and decision tree.
The state diagram for the window defines two states: Open and Closed, as well as the transitions for moving from one state to the other. This graphically implemented state diagram is depicted in Figure 6, and is summarized below:

- When the window is closed, it will be opened only when the A/C unit is off and if opening the window will have a positive impact on the temperature inside, i.e. "if it is too warm inside, then open the window only if the outside temperature is colder", and "if it's too cold inside, then open the window only if the outside temperature is warmer".
- When the window is open, it will be closed when one of these conditions is true:
    - A/C is on (cooling or heating)
    - The output temperature has a negative impact on the temperature inside

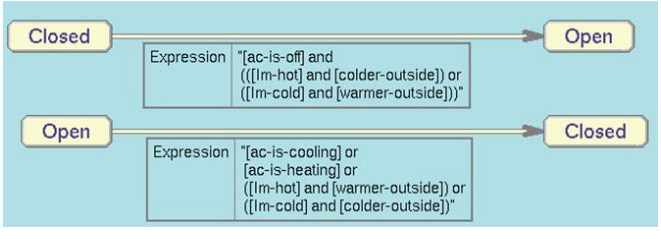


Figure 6. State Diagram Control for Window.

The state diagram for window position control is based on the state of the A/C unit, the selected comfort level, and the difference between the inside and outside temperatures. The expressions determine the new state and the action to be performed.

The state diagram for controlling the A/C unit defines 3 states: AC-Off, Cooling and Heating, as well as the transitions for moving from one state to the other (with the restriction that we do not allow the A/C to go directly from Cooling to Heating or conversely from Heating to Cooling). This graphically implemented state diagram is depicted in Figure 7, and is summarized below:

- When the A/C unit is off, it can be turned on either for cooling or heating, but in both cases it will check that the window is closed before doing so. To start cooling it will check that it is too warm inside and warmer outside, and to start heating it will conversely check that it is too cold inside and even cooler outside. This approach guarantees that the A/C unit will not be used in cases where opening or closing the window could help solve the temperature issue inside the room.
- When the A/C unit is cooling, it will be turned off it it is too cold inside or if the temperature inside is comfortable and the outside temperature is colder (i.e. it makes sense to open the window instead of using the A/C unit).
- When the A/C unit is heating, it will be turned off it is too hot inside or if the temperature inside is comfortable and the outside temperature is warmer (i.e. it makes sense to open the window instead of using the A/C unit).

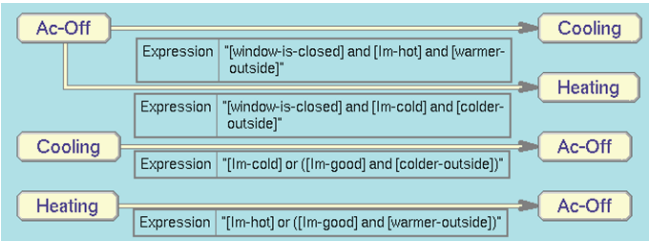


Figure 7. State Diagram Control for A/C.

With no control decisions (window closed, A/C unit off), the external and internal temperature cycles are plotted vs. time to produce the graph repeated in Figure 8.

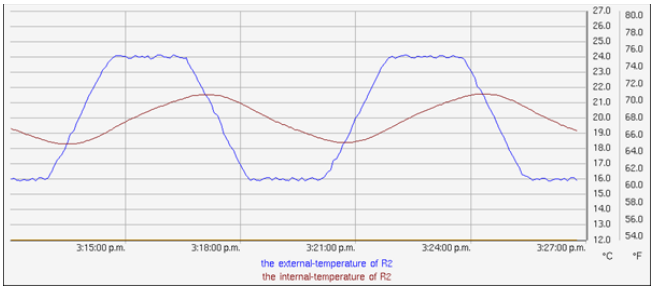


Figure 8. Temperature Cycles with No Control.

To reflect the behavior (and benefits) of automated control using the State Diagram option, another graph showing the external and internal temperature cycles that result from state diagram control is depicted in Figure 9.

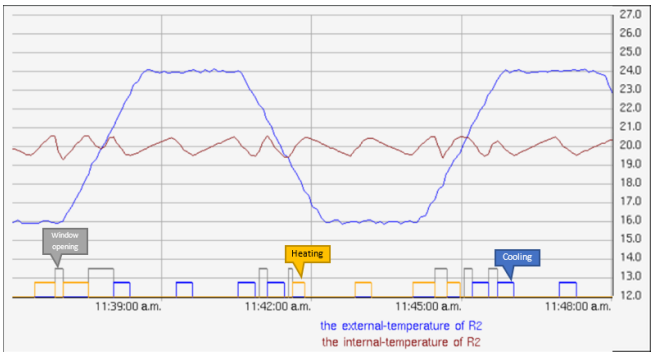


Figure 9. Temperature Cycles with State Diagram Control.

In this case, the comfort temperature was set to 20º C. As before, the trapezoidal graph shows the simulated external temperature, while the smaller graph shows the resulting internal temperature. The rectangles below the chart provide an indication of window position, A/C cooling, and A/C heating cycles that result from the state diagram control.

From this comparison it can be seen how the state diagrams attempt to maintain the inside temperature of the room at the comfort level. As mentioned previously, a deviation of +/- 0.5°C is allowed.

4.3. Decision Tree Inference

A decision tree was built (inferred) automatically using a set of historical data generated by the state diagrams. Historical data was simulated using the simulation model described in section 4.1 and was collected over an extended period. For the sake of this example, it is assumed that this data contains sufficient information for generating a complete decision tree (e.g. it contains sufficient behaviors to capture relevant control decisions).

The software utility has been implemented to include a set of generic GP blocks that aid in the configuration, training, execution, and validation of the decision tree inferencing. The approach is as follows:

1. Reading the historical data from a comma separated file.
2. Pre-processing the data for input to a decision tree creation block.
3. Executing the decision tree creation block.

A screenshot of the resulting graphical program is shown in Figure 10. The diagram shows both the data pre-processing block and the decision tree creation block, as well as the directed link that controls the processing flow.

Table 2 shows an example of the historical data produced from the state diagram control system. This data is considered raw data, archived according to timestamps and containing

Table 2. Example Historical Data.

Log_Time	T_Ext	T_Int	T_Comfort	Window	AC	Action
11/03/2019 11:04	16.065	19.501	20	CLOSED	AC-OFF	DO-NOTHING
11/03/2019 11:05	15.875	19.464	20	CLOSED	AC-OFF	START-HEATING
11/03/2019 11:06	15.955	19.579	20	CLOSED	HEATING	DO-NOTHING

control signals that have been formatted with mnemonic labels.

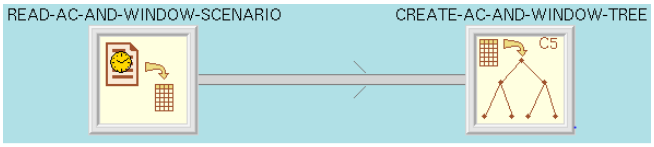


Figure 10. Graphical Program for Decision Tree Creation.

4.3.1. Data Pre-Processing

The raw simulation data as it exists is not well suited for training or creating a decision tree due to its temporal characteristics and missing symbolic information. These are issues that are further exacerbated in real-world plant data. To aid in the creation of decision branches that are understandable from a human decision perspective it is appropriate to pre-process the raw data and generate new columns and/or updated labels. Ideally, this process can be automated. In our example, the following pre-processing approach is applied to the temperature control raw data:

- The Log Time (timestamp) column is not actually required.
- The actual values of T_Ext and T_Int must be mapped to a new value that indicates whether it is hotter or colder outside:

  (if T_INT < T_EXT
  then 'HOTTER-OUTSIDE'
  else 'COLDER-OUTSIDE')

- T_Comfort must be mapped to a 'Comfort Level' based on the difference between the current inside temperature and the comfort level:

  IF T_INT>T_COMFORT + 0.5 then 'TOO-HOT' else if T_INT< T_COMFORT - 0.5 then 'TOO-COLD' else 'COMFORTABLE'

The resulting dataset after pre-processing is shown in Table 3. Notice the transformation of the data into a more symbolic representation, which will significantly aid in understanding and validating the inferred decision tree.

4.3.2. Decision Tree Creation

After the software utility has successfully pre-processed the data, the transformed data set is fed into a Decision Tree creation block that builds the decision tree using the C5 algorithm described in the section 3.

The graphical decision tree creation block creates:

- A graphical representation of the decision tree.
- A procedural representation of the decision tree in the form of a series of nested IF THEN statements.

For purposes of completeness, an example of the resulting decision tree generated from several days of temperature control system data is provided in Figure 11.

Note to the reader: due to font size limitations of this paper, the rendered decision tree is not legible in its complete form. It is presented to give the reader a feel for the number of decision branches that have been inferred from the control system logic described in section 4.1 and 4.2. Specific subsets of the graphical decision tree will be shared and discussed in the following sections.

Table 3. Pre-Processed Historical Data

T_Int	T_Comfort	Window	AC	Action
COLDER OUTSIDE	COMFORTABLE	CLOSED	AC-OFF	DO-NOTHING
COLDER OUTSIDE	TOO COLD	CLOSED	AC-OFF	START-HEATING
COLDER OUTSIDE	TOO HOT	CLOSED	HEATING	TURN-AC-OFF

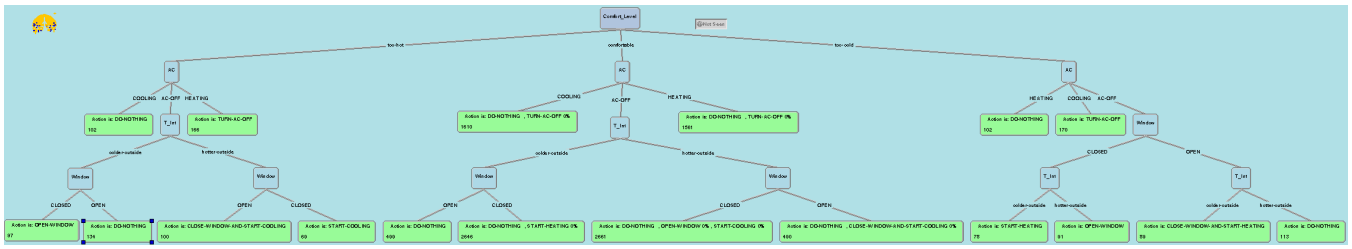


Figure 11. Complete Graphical Decision Tree from Pre-Processed Temperature Control Data.

Examining the leaf nodes of an inferred graphical decision tree reveals the final prediction or action recommendation for a particular decision path. In some cases, a leaf node might in fact result in multiple possible actions. Figure 12 shows an example of one of the leaf nodes that has multiple possible actions.

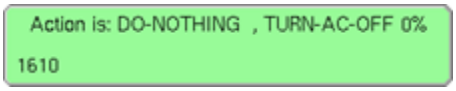


Figure 12. Example Leaf Node with Multiple Actions

The value at the lower left-hand corner of the node shown in Figure 12 indicates that 1610 data samples were in the data subset for that termination, however, the resulting action is DO-NOTHING or TURN-AC-OFF. The fact the TURN-AC-OFF equals 0%, indicates that less than 1% of the 1610 samples for this branch have this result. In fact, the actual split is 1608 DO-NOTHING and 2 TURN-AC-OFF

4.3.3. Decision Tree Procedural Representation

Another advantage of decision trees over other machine learning techniques is that they represent a simple series of tests and decisions. The result indicated by a leaf node can be represented by an equation. The equation for the decision path highlighted in Figure 13 is the equation of the action DO-NOTHING :

IF Comfort_Level = “COMFORTABLE”
AND AC =”AC-OFF”
AND T_Int = “COLDER-OUTSIDE”
AND Window = “OPEN”
THEN Action = “DO-NOTHING”

It is therefore relatively simple to generate a procedure representing the complete decision tree. In this way, it is possible to execute the decision tree procedure with a set of input values and obtain a result. This technique was used by the authors in the example temperature control application to either advise a user on the best action to take, or to control the elements of the room automatically by periodically executing the decision tree procedure with the current temperatures and states.

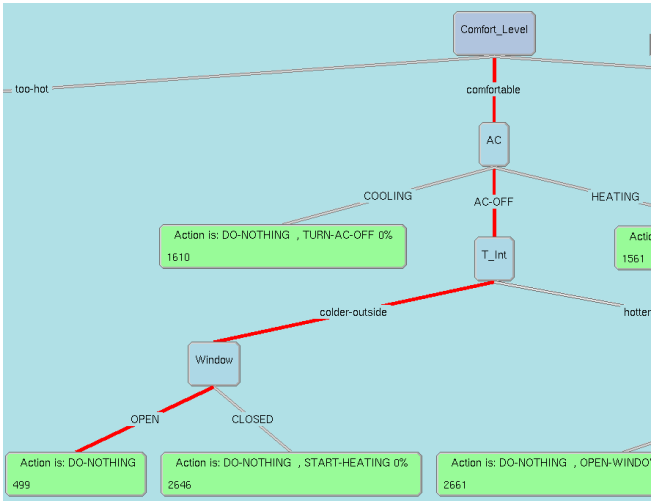


Figure 13. Example Graph-to-Procedure.

4.4. Decision Tree Driven Control

The two control modes “**By Decision Tree as an Advisor**” and **“By Decision Tree as a Controller”** described in section 4.2 allow the decision tree procedure to be executed on incoming previously unseen data. In both cases, the latest decision path determined by the decision tree when the procedure was executed can also be shown using a similar graphical representation. Figure 14 shows an example of an animated path and action determined by the decision tree procedure. The path and nodes can be animated each time the decision tree procedure is executed.

The charts in Figure 15 compare the results between the control performance of the example room temperature by state diagrams and decision tree. As one would expect, the decision tree performs as well as the state diagram since it was built from data produced by the state diagram and contains all possible situations. Real world operations involve control strategies that are not as well behaved as our example. As a result, a decision tree might be inferred from historical data that does not always result in a directly usable tree. This is dealt with in more detail in the following sections.

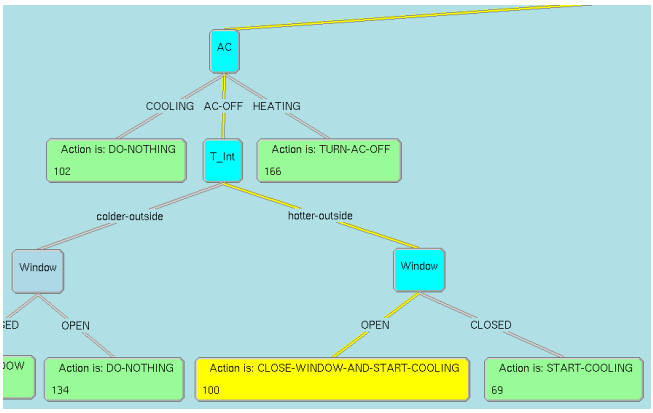


Figure 14. Graphical Representation of Decision Tree Execution.

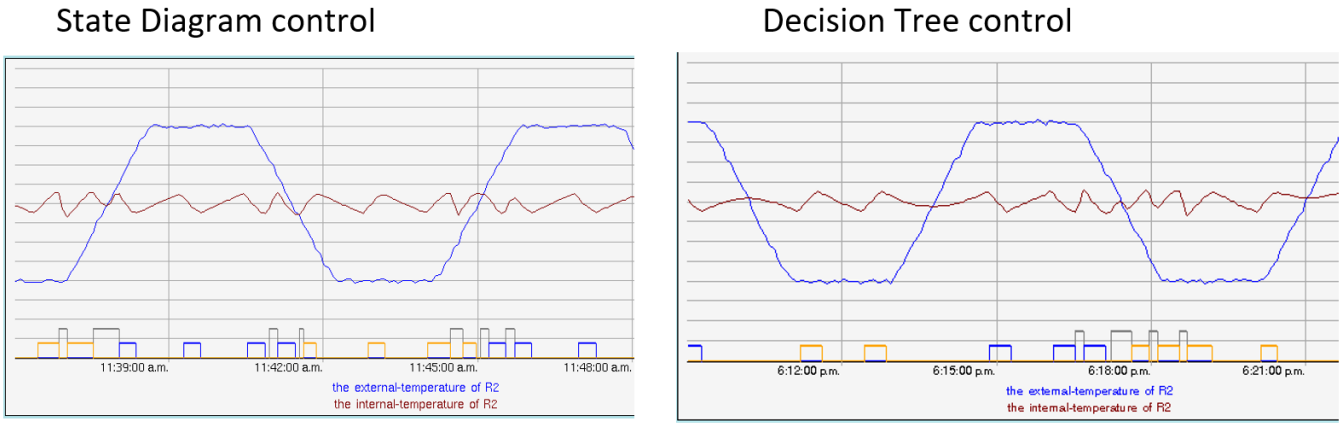


Figure 15. Control Performance Comparison Between State Diagram and Inferred Decision Tree.

4.5. Overfitting

When a decision tree is built from a training data set, it will continue splitting the data until no further split is possible. With a large dataset obtained from a process history log the data may contain "noisy" effects. "Noise" in the context of decision tree inferencing refers to situations which result in increased entropy of the dataset and therefore difficulty in resolving decision boundaries. Basically, these effects can greatly impact the accuracy of the generated decision tree. Examples of entropy increasing effects include the following situations:

- Effects caused by several examples of data having the same input variable values but different actions (see Figure 16).
- Effects caused by data that is incorrect because of inappropriate and/or conflicting decisions were actually made.
- Effects caused by data acquisition issues or sampling noise.
- Effects caused by features (input variables) that are not relevant to the decision process.

Since a decision tree will try to perfectly model the data, it will also model the data resulting from these effects. This could impact the ability of the tree to determine the correct result on data that it has not seen before. This is referred to as "Overfitting" and is the main disadvantage with using inferred decision trees for executing control. Overfitting can cause the tree to be over complex or difficult to read and reduces its capacity to "generalize". Note that too little training data will also cause the effect of overfitting.

There are several techniques to avoid overfitting such as defining some criteria to stop growing the tree, removing irrelevant attributes, or "pruning" the fully-grown tree (Johnson, 2014). In the software utility described in this work, the authors used a post-pruning method to remove unnecessary parts of the tree after creation. Post pruning can be done automatically or manually.

4.6. Pruning

Pruning a decision tree consists of replacing insignificant sub-trees with leaf nodes. A decision node that only has leaf nodes as children can effectively be pruned. An example is shown graphically in Figure 16.

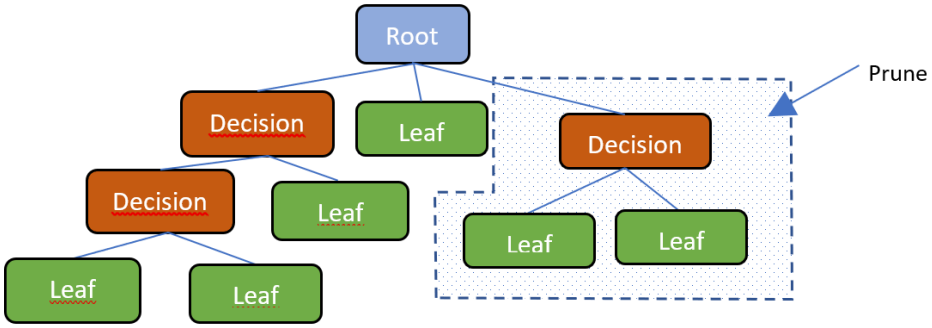


Figure 16. Pruning as a Decision Tree Optimization Technique.

Such groups of nodes (decision parents with only leaves as children) are known as "twigs". Pruning a twig is done by replacing the nodes with a single leaf node whose action is the majority action among the twig's leaves.

- A twig should only be pruned if the result does not increase the error-rate of the decision tree.

The software utility created by the authors (similar to that described in Walker, Figueroa, and Toro-Medina, 2017) also allows the error-rate of the decision tree to be determined programmatically by executing the tree with a test consisting of a separate set of data from the training set. The tree is executed for each row of the test data and the result provided by the tree is compared against the result in the test data. The error rate is calculated as the ratio of the number of wrong results for the complete test data set.

Automatic pruning uses a test data set. It executes the tree on the test data set and verifies whether the error rate increases after each twig is pruned. If the error does not increase after pruning a twig, the twig is replaced. The automatic pruning will then repeat the operation with a different twig. This continues until no more pruning is possible without increasing the error rate.

Pruning is an iterative process that can often be recursive. As twigs are pruned, new twigs are often created. This is summarized graphically in Figure 17 for the temperature control example. In this case, an entire branch ("Comfort_Level" = "COMFORTABLE") renders a pruned result of "DO-NOTHING". This has simplified the tree and enabled it to better generalize on new data. An expanded view of the before and after graphical decision trees is provided in the Appendix.

In this way it is possible to significantly reduce unwanted complexity from a decision tree inferred from data associated with an actual industrial process. The pruning process does come with some caution, however, as decision tree complexity can sometimes indicate other problems that need attention and should not be "filtered".

5. INDUSTRIAL DECISION TREE APPLICATIONS

The simple temperature control example presented in this paper for decision tree inference and execution has been implemented and studied in order to aid in understanding, qualifying, and validating the approach prior to (and in conjunction with) the application to decision tree inferencing in a real-world continuous process plant. Algorithm execution, decision tree creation, and pruning have been successfully validated using actual plant data for a number of control system decision cases and the underlying architecture is being applied to several real-world process applications (see Figueroa, Underwood, and Walker, 2019). Studied applications include those from batch processing, continuous processing, petroleum processing, chemical processing, pharmaceutical manufacturing, discrete manufacturing, and energy production (Walker, 2007). In these applications, there have been several significant successes and obstacles. Some of the obstacles are summarized in Table 4.

Table 4. Real-world Obstacles for Decision Trees

Issue	Discussion
Feature extraction and symbolic categorization of data	Plant data is typically parametric and continuously valued. Improved performance from decision tree inference follows from categorization of data channels using symbolic labelling. It is not always obvious what these should be. The authors continue to explore better approaches for feature extraction and data labelling but have achieved success in demonstrating its utility.
Time lags and delays, lack of time synchronization in data	Control decisions are generally made following awareness of trends, conditions, anomalies, or faults. Sometimes the decision is deferred (due to shift changes, delayed awareness, or even indifference). In order to capture

	correlations between trends and actions, the decision tree algorithm needs to support time windows. The authors are continuing to research effective mechanisms to augment historical data with such temporal implications.
Datasets with high entropy	Real world data is noisy (process noise, sampling noise) and messy making classification difficult due to high degrees of impurity. The authors have built utilities that help the practitioners see the correlations in data, check the spread of data, and identify approaches that help to lower the entropy. A screenshot of this utility is provided in Figure 17.
Conflict in decision data	In real world systems the data often includes cases where different decisions were made under similar or identical conditions. These cases create decision trees which cannot resolve the classification without error. Again, tools added to the decision tree algorithm can help identify and resolve these situations.
Required a priori knowledge	The presumption with using decision trees is that no knowledge about the process or decision logic is necessary in order to derive the decision tree. This is not always the case – especially when the data contains bad decisions. The authors are working with process engineers to help identify the appropriate amount of domain knowledge that should be applied during the decision tree inferencing.

Notwithstanding, the authors have successfully deployed decision tree inferencing, monitoring, and control for real world process plants where good correlations exist in the input dataset. The authors have also derived tools for aiding in the validation process – tools that provide metrics and information regarding the performance of the algorithms when subjected to validation data. An example dialog from the validation utilities is provided in Figure 18.

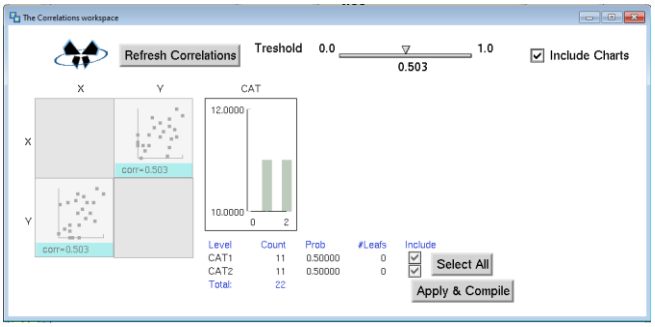


Figure 17. Correlation Utilities to aid in preprocessing.

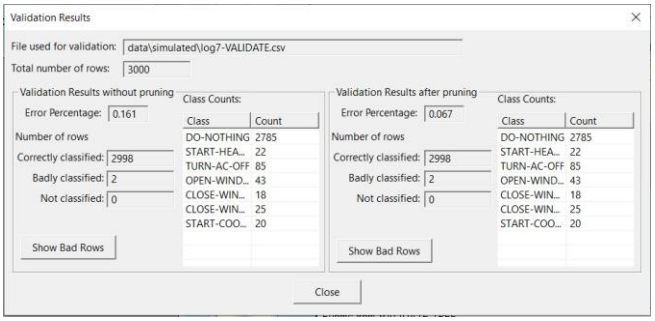


Figure 18. Validation Utilities.

With the aid of these correlation and validation utilities, the authors have been able to successfully infer decision trees that perform well on previously unseen data. Furthermore, when applying these validated trees for autonomous control operations, the decision trees demonstrate similar control performance to the original control strategy. However, this is clearly a work-in-progress. The authors will continue to apply and research decision tree techniques for deriving awareness regarding process health, product quality, and predicting future failures. The authors will also continue to advance the state of the art with regards to validating decision trees for supervisory control. The following list summarizes some of the successes, benefits, and suggestions for future research and desired objectives from the application of decision trees for monitoring and control:

- Improved understanding of underlying decision processes associated with operational control system logic.
- Identification of operational expertise not fully understood or assimilated within existing corporate policy.
- Identification of anomalous or erroneous control system logic and/or human expertise.
- Implementation of decision support and control system decision monitoring according to published concept of operations.
- Improved situational awareness
- Process health determination and reporting
- Abnormal Situation Management

- Process optimization
- Automated policy-driven process control and supervision
- Autonomous process operations

6. CONCLUSION

Decision trees are a popular machine learning tool and some of the principal reasons are as follows:

- Easy to understand and use
- Relatively easy to implement
- Efficient and fast to execute computationally
- Can operate on different types of data including mixture of continuous and categorical variables
- Data normalization is not necessary
- Can cover both regression and classification

A disadvantage is that they are prone to overfitting as described earlier.

The author's implementation of decision trees has proven to be very easy to pre-process data and generate decision trees. The resulting decision trees can either be exploited manually by using the graphical representation or automatically by feeding offline or online data to infer or execute decision tree logic in real-time. This approach has shown that decision trees can be created, tested, validated, and optimized using test data sets and automatic pruning.

The approach has demonstrated its utility for gaining multiple potential benefits for industrial applications, and further work is planned for the near future.

REFERENCES

Cinaroglu, S., (2016). Comparison of Performance of Decision Tre Algorithms and Random Forest: An Application on OECD Countries Health Expenditures. Intl. J. of Computer Applications. Vol. 138 – No. 1. March 2016, pp. 37-41.

Ezzikouri, H., & Fakir, M. (2010). Algorithmes de classification: ID3 & C4.5. Universite Sultan Moulay Slimane, Beni-Mellal, Maroc. https://www.academia.edu/33701469/Algorithmes_de_classification_ID3_and_C4.5.

Figueroa, F., Underwood, L., Walker, M., (2019). NASA Platform for Autonomous Systems (NPAS). Proceedings of the AIAA Scitech 2019 Forum. January 7-11, San Diego, CA.

Johnson, R., (2014). Data Mining. *CSE 40647 lectures*. https://www3.nd.edu/~rjons15/cse40647.sp14/www/content/lectures/24 - Decision Trees 3.pdf.

Mohan, V., (2013). Decision Trees: A comparison of various algorithms for building Decision Trees. https://pdfs.semanticscholar.org/3399/c175beca3ab4843d67f91bb28f564099d0bb.pdf.

Patel, B. R., & Rana, K. K., (2014). A Survey on Decision Tree Algorithm for Classification. *International Journal of Engineering Development and Research*, vol. 2, issue 1, pp. 1-5.

Quinlan, J. R., (1986). Induction of Decision Trees – Machine Learning (Theory). Boston: Kluwer Academic Publishers.

Rokach, L., & Maimon, O., (2010). Decision Trees. In Rokach, L., & Maimon, O. (Eds). *Data Mining and Knowledge Discovery Handbook* (pp. 165 – 192). New York: Springer Publishing.

Walker, M., (2007). Model-based Reasoning Applications for Remote Intelligent Systems Health Management. *Proceedings of ASNE Intelligent Ships Symposium*. May 9-10, Philadelphia, PA.

Walker, M., (2010). Next Generation Prognostics and Health Management for Unmanned Vehicles. *Proceedings of the IEEE Aerospace Conference*. March 6-11, Big Sky, MO.

Walker, M., Figueroa, F., Toro-Medina, J. (2017). PHM Enabled Autonomous Propellant Loading Operations. *Proceedings of the Aerospace Conference*, March 3-11, Big Sky MO.

Yobero, C., (2018). Determining Creditworthiness for Loan Applications Using C5.0 Decision Trees. *RPubs*. July 9, https://rpubs.com/cyobero/C50.

BIOGRAPHIES

Robert Matania is a computer and electronics engineer with over 30 years of experience in industrial computing and expert systems. He obtained his BS (Hons) degree in electronic engineering at Brighton Polytechnic in 1978. After working as a software engineer for the mainframe computer manufacturer ICL in Great Britain, he moved to France in 1981 where he began work designing and developing SCADA software. He spent 5 years with the Compagne Generale des Eaux (Veolia) as a project manager, senior software designer and product manager. He spent 15 years working for expert system manufacturer Gensym Corporation as a pre-sales engineer and then as a senior consultant specializing in artificial intelligence and expert systems. He also worked as a senior consultant for UReason applying advanced alarm management techniques to industrial processes. He is currently living in France and working as an independent consultant with D2K Technologies.

Jean-Marie Forêt is an experienced architect, designer, and developer of intelligent and distributed applications with more than 30 years of practical experience in industry. He received his Civil Engineer in IT degree at the Université Catholique de Louvain-la-Neuve in Belgium (1983). His experience in artificial intelligence began in 1983 when he implemented AI-based applications in VLSI design for Alcatel Bell, after which he founded AI Systems (1986-1991). He developed expert systems solutions for EASE Software Engineering (1991-1994) and was a Senior Consultant for expert system manufacturer Gensym Corporation (1995-2001). He also worked as a Senior Consultant and Knowledge Engineer for UReason B.V. (2005-2018). He has been working as a senior consultant for D2K Technologies since 8/2018, focusing on platform development and machine learning applications. He resides with his family in Brussels, Belgium.

Vicente Camarillo received his BSEE from the Instituto Tecnológico de la Laguna (1992) and his MSCS from the Instituto Tecnológico y de Estudios Superiores de Monterrey (2001), with a specialty in Artificial Intelligence. He has over 20 years of experience in expert systems development and deployment, mainly in the mining and metallurgical industry. His work experience includes time as Controls Engineer for Servicios Industriales Peñoles (1994-1995) and Met-Mex Peñoles (1995-2005), as Analyst for Peñoles Center for Energy Administration (2002-2006), a Senior Solutions Engineer for SGS de México (20016-2013) and Ignite Technologies (2013-2016). He has been working as a Senior Consulting Engineer for D2K Technologies since 12/2016. He resides with his family in Cd. Lerdo, Mexico.

Mark G. Walker received his BSEE from Cal Poly University, Pomona (1990), and his MSCompEng from the University of Southern California, Los Angeles, CA (1994), where he specialized in machine intelligence. From 1976-1983 he served in the U.S. Navy as a Nuclear Reactor Operator onboard the U.S.S. Long Beach CGN9. His experience in artificial intelligence began in 1989 as a DOE undergraduate fellow at the Center for Engineering and Science Advanced Research Lab at Oak Ridge National Laboratory where he developed image processing and perception software for autonomous robots. His work with HUMS and PHM began with BFGoodrich Aerospace, Vergennes, VT in 1996, and has co-authored four patents in the field. He also spent 6 years as Senior Consulting Engineer for expert system manufacturer Gensym Corporation and 10 years as Lead Engineer, Intelligent Systems for General Atomics, where he led GA in the development of reusable PHM systems applied to various industries. He founded D2K Technologies in 2014, a solution provider of intelligent model-based reasoning solutions for mission critical systems. He also serves as a PHM and Autonomous Operations SME for NASA, with active projects at SSC and JSC. He resides with his family in Oceanside, California.

APPENDIX

Pruning Example:

The iterative pruning process described in Section 4.6 continues searching upwards in the tree to find additional "twigs" that have been created by earlier pruning. This process allows for potentially complex but unnecessary branches to be reduced - thereby improving understandability and assessment of results.

In the simple temperature control example presented in Section 4.0, the first pass of pruning produces the elimination of one of the twigs associated with the Comfort_Level = "COMFORTABLE". This is shown graphically in Figure A as a result from the diagram shown in Figure 15 (reproduced here).

In summary, when the simulated historical data from the simple temperature control example is presented to the Decision Tree utility, the software produces the decision tree shown in Figure B. After the pruning process, the branches associated with the "COMFORTABLE" branch successively collapse, yielding the simplified tree shown in Figure C.

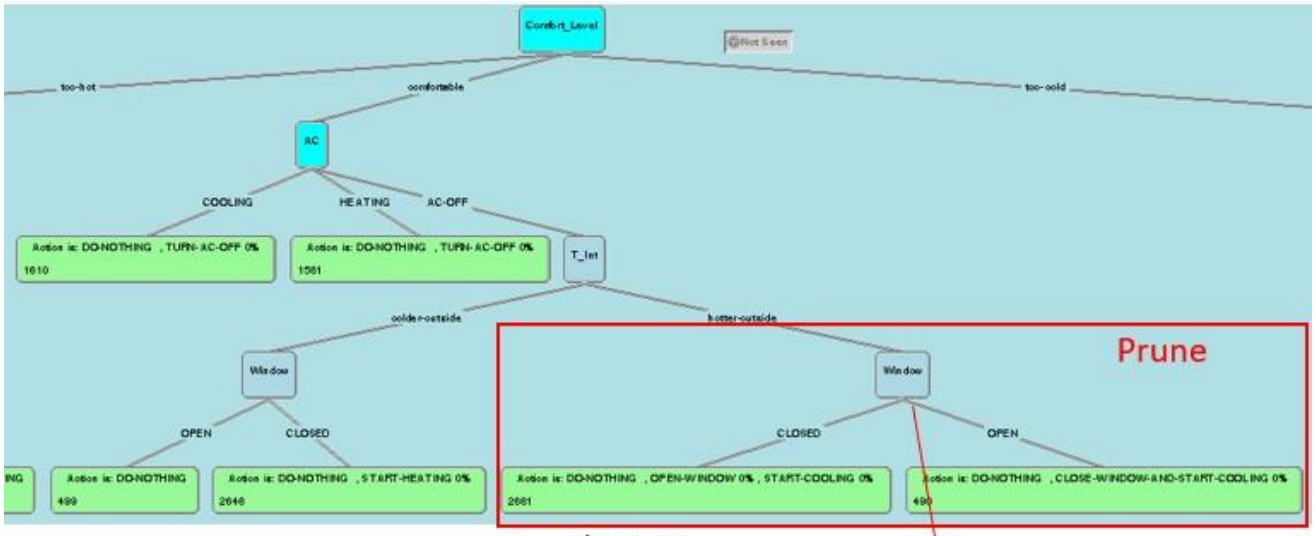


Figure 15

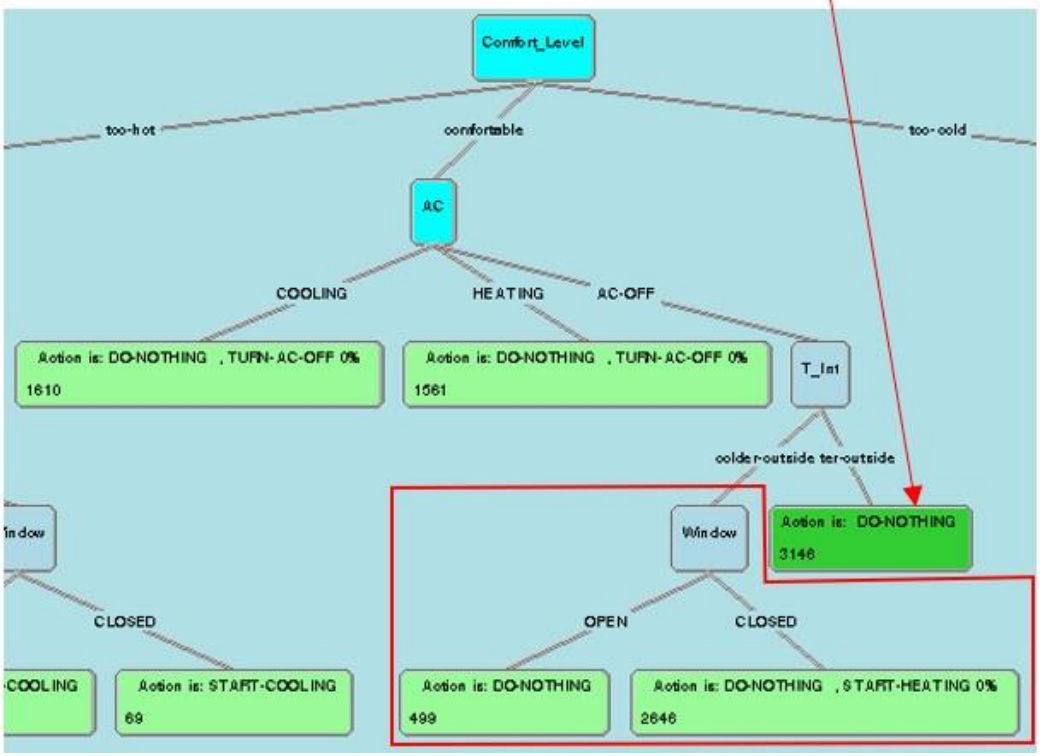


Figure A. Iterative Pruning Example from Temperature Control Application.

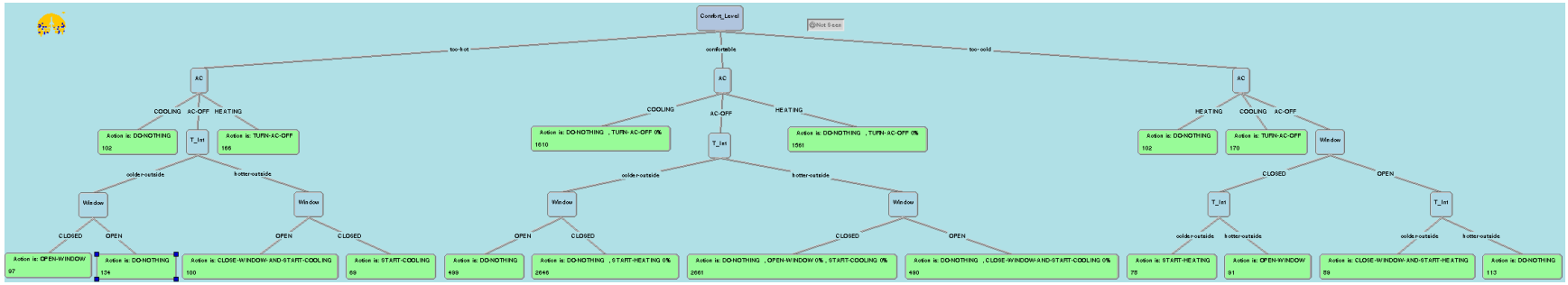


Figure B. Initial Decision Tree from Temperature Control Application.

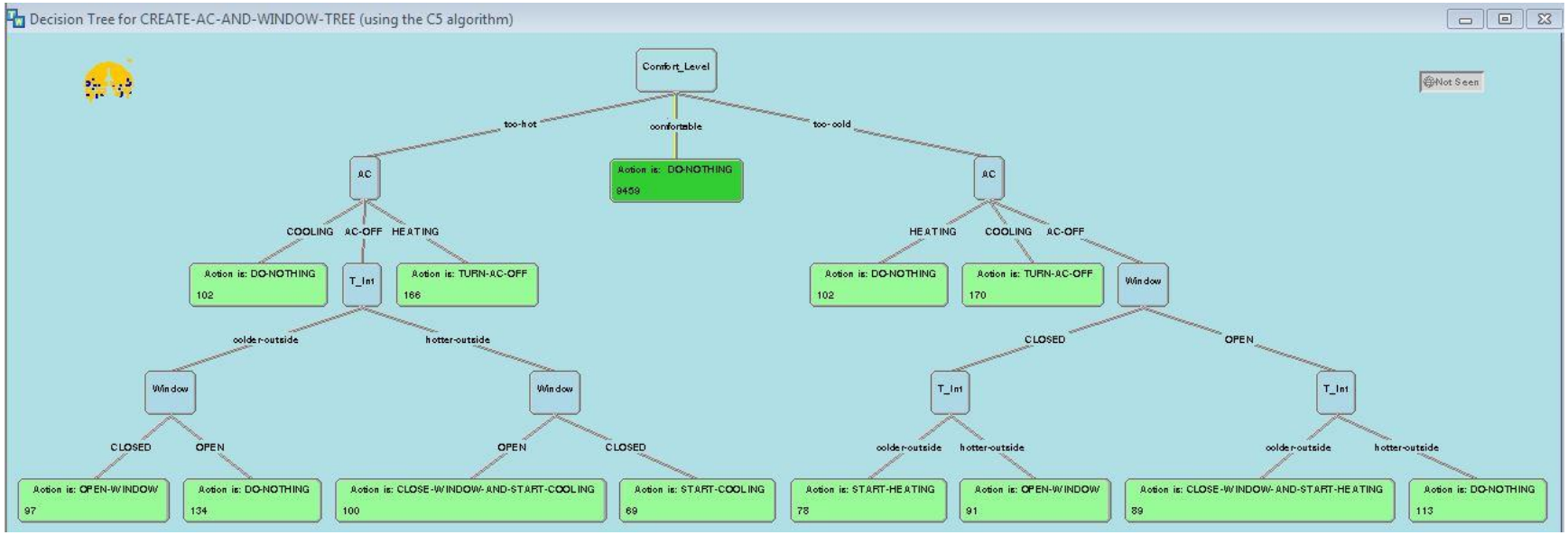


Figure C. Final Decision Tree from Temperature Control Application