Decision tree approach to power systems security assessment

L. Wehenkel* and M. Pavella
University of Liège – Institut Montefiore, Department of Electrical Engineering, Sart-Tilman B28, B-4000 Liège, Belgium

An overview of the general decision tree approach to power system security assessment is presented. The general decision tree methodology is outlined, modifications proposed in the context of transient stability assessment are embedded, and further refinements are considered. The approach is then suitably tailored to handle other specifics of power systems security, relating to both preventive and emergency voltage control, in addition to transient stability. Trees are accordingly built in these various application domains, and their salient features are explored, assessed, compared. Among attractive aspects of the trees, we mention their ability to uncover the intrinsic mechanism governing physical processes, and to provide a clear description in terms of tractable system parameters. Further, their dual, 'attribute space' representation is shown to have complementary possibilities for more refined analysis and, in addition, for sensitivity assessment and control. Overall, the approach is characterized by its great flexibility with respect to tree structure, types of physical parameters driving the phenomena, and classes of potential applications. To illustrate and support the developments, real-world examples are reported, relating to transient and voltage stability issues simulated on several power systems.

Keywords: decision trees, power system security assessment, real time control, artificial intelligence, transient stability, voltage security

I. Introduction

Power systems security is a multifacet problem encompassing static and dynamic stability aspects. The current trend to increase bulk power transfers over long distances on the one hand, and to operate systems closer to their limits on the other, makes security more intricate and at the same time more imperative to handle. Currently available conventional methods can hardly meet the imposed requirements. The need for seeking new and conceptually more appropriate aproaches is being increasingly felt.

The proposed decision tree method provides an interesting alternative. The general decision tree inductive inference method falls into the category of learning from examples[1]. The underlying principle consists of extracting off-line information about a given problem, and organizing it into decision trees which are both interpretable and appropriate for automatic on-line use; this enables one subsequently to infer knowledge about new, unseen cases, whenever they arise. The trees are automatically built on the basis of a 'learning set' of states, preclassified by means of 'system theory' methods and corresponding numerical programs.

The first attempt to apply decision trees to power system security was for assessing transient stability[2,3]. The principle of this specific approach may be stated as follows: using a set of preconstructed trees, assess the robustness of any new operating state by merely 'dropping' it at the top of the appropriate tree, and observing the position it finally reaches in the tree structure. Note that this is achieved in terms of precontingency state parameters; hence, it avoids all cumbersome transient stability computations with conventional methods, and allows transient stability assessment to be performed on-line. The results obtained with this decision tree transient stability method have been very encouraging[4]. Hence, research has been steadily pursued in order to improve the method, to apply it to real-world large systems[5], and at the same time to modify, tailor, and extend it to other security contexts[6–8].

This paper aims at giving a synthetic view of this recently proposed decision tree approach. Results – theoretical as well as practical – obtained so far are reported, stressing key issues, highlighting specific and common features, and attempting to suggest operating strategies. On the other hand, among the newly proposed refinements of the inductive inference method, we mention a measure which quantifies the 'quality' of a tree by combining the notions of precision and complexity. Also, backward and forward pruning techniques are considered, to optimize the tree quality.

Two salient features common to the devised approaches are highlighted, viz. effectiveness and flexibility; – effectiveness is combining quality of the results and real time computing requirements; – flexibility, in respect of the following: (i) the inductive inference method used to

Received 21 November 1991; Revised 7 July 1992

* Research Assistant, FNRS

build trees; (ii) system physical parameters participating in this building; (iii) practical potential uses offered by the trees themselves and their dual representation, the 'attribute space'[9,10].

Overall, of paramount importance for the success of the approach appear to be the suitability of the inductive inference method used for the automatic tree generation, and the representativeness of the learning set; or stated otherwise, human expertise and the close collaboration between engineers in charge of the power system operation and researchers in charge of the method.

The paper originates from ongoing research in the area of transient stability, and more recently of voltage security (both preventive and emergency types of voltage control). Experience gained via these investigations will help us to illustrate the methods by means of typical trees built for these security purposes with a variety of power systems ranging from simple, academic type to very large UHV real systems.

II. Thrust of the general decision tree methodology

Decision tree methodology in general is a nonparametric inductive learning technique, able to produce classifiers for a given problem which can assess new, unseen situations and/or uncover the mechanisms driving this problem[11–14]. The building of a decision tree relies on a learning set (**LS**), i.e. a set of preclassified states[a]: starting at the top node of the tree with the entire **LS**, one progresses by recursively creating successor nodes, i.e. by splitting the **LS** into subsets of increasing classification purity. The procedure is stopped when all the newly created nodes are 'terminal' ones, containing 'pure enough' learning subsets.

The ways of splitting the successive subsets, and even more of deciding when to stop splitting are essential. The general method we briefly describe hereafter is a modified version of ID3[15]; it was initially developed for the purpose of power systems transient stability assessment[3]. Its mathematical formalism may be found in References 4, 16 and 17. In what follows, we recall its fundamentals along with proposed improvements.

II.1 *General framework and notation*

We consider a domain specific set of *preclassified states*. Each state s_k is characterized by a certain number, say n, of *attributes*[b]; to simplify, we will assume for the time being these to be ordered numerical *attributes* a_i (the same number for each state); accordingly, s_k will be characterized by

$$s_k = [a_1 = v_{1k}] \cap [a_2 = v_{2k}] \cap \ldots \cap [a_n = v_{nk}] \qquad (1)$$

where v_{ik} is the value of the ith attribute of the state s_k; the $n\, v_{ik}$ components compose the vector

$$\boldsymbol{v}_k = (v_{1k}, v_{2k}, \ldots, v_{nk}) \qquad (2)$$

To ease the description we also assume that the states are classified into two classes only, $\{+, -\}$[c].

With the above notation, we define the *learning set* (**LS**), by the collection of a number, say N, of preclassified states

$$\mathbf{LS} \triangleq \{(\boldsymbol{v}_1, c_1), (\boldsymbol{v}_2, c_2), \ldots, (\boldsymbol{v}_N, c_N)\} \qquad (3)$$

where

$$c_k \in \{+, -\} \qquad (4)$$

Also, we define the *test set* (**TS**) by another number, say M, of preclassified states, obtained in a similar but independent way

$$\mathbf{TS} \triangleq \{(\boldsymbol{v}_{N+1}, c_{N+1}), (\boldsymbol{v}_{N+2}, c_{N+2}), \ldots, (\boldsymbol{v}_{N+M}, c_{N+M})\} \qquad (5)$$

In what follows, learning sets will be used to build decision trees, test sets to evaluate their ability to correctly classify unseen states.

Remarks

(1) The above sets are considered to be statistical samples drawn from the population of possible states. Their proper size is a question of great concern and essentially depends upon the particular application domain. This will be discussed in section III and IV, in the context of power system security.

(2) The generalization to more than 2-class classifications and to categorical attributes in addition to ordered ones may be easily obtained[4,16,17]. Both generalizations will be used in the security applications discussed in section III.

II.2 *Construction of decision trees*

II.2.1 *Automatic building procedure*

A *decision tree* (DT) is a tree structured upside down, comprising *test* and *terminal nodes*. Each test node is associated with a test *on the attribute values of the states*, to each possible outcome of which corresponds a *successor* node. The terminal nodes carry the information required to classify the states. Such a DT is portrayed in Figure 1 built for the purpose of transient stability assessment as outlined in section III.2.1 below. It is composed of five nodes; two test nodes, labelled 1, 2 and the remaining three terminal nodes. Note that here only numerical attributes are considered (whether continuous or discrete); the tests are therefore *dichotomic*, and the resulting DT is *binary*: each test node is split into *two* successors. In the sequel, only binary trees will be considered.

The building of a DT necessitates selecting *a priori* relevant attributes; we will henceforth refer to them as the *candidate attributes*. These are parameters of the system readily available and presumably carrying sound, essential information about the problem of concern.

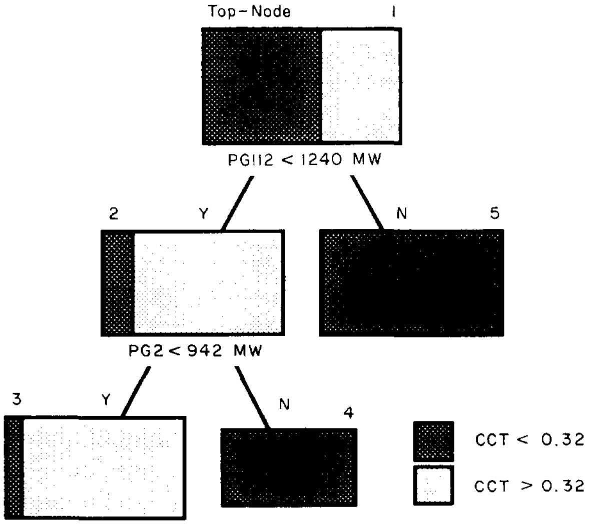


Figure 1. (From Reference 4)

Given a **LS** and a list of candidate attributes, the automatic building of a DT conforms to the procedure described below and illustrated in Figure 1[18].

- Starting at the *top-node* (*root*) of the tree, with the list of candidate attributes and with the entire learning set of preclassified states, analyse these states in order to select a test which achieves their 'optimal' splitting into two subsets, in the sense that these subsets provide a maximum increase in classification purity (or, equivalently, a maximum amount of information about the states' classification – e.g. see the Appendix). The selection proceeds in two steps:

  (i) for each attribute, say a_i, find its 'optimal' threshold value v_{i*} in the sense specified above, by scanning the values it assumes for all learning states; this defines a test

  $$a_i < v_{i*}\,?$$

  (ii) among the different candidate attributes, choose the best one, a_*, along with its optimal values, v_{**}, to split the node,[d] according to the test

  $$\mathrm{T}\colon\ a_* < v_{**}\,? \qquad (6)$$

  In short, test T consists of defining the optimal attribute along with its optimal threshold value: it uses the attribute along with its threshold value having the highest 'score' (see equation (A.5) in the Appendix).
- The selected test thus applied to the learning set of the node, splits it into two subsets, corresponding to the two successors of the node. Starting with the top node of the tree and the entire **LS**, the two subsets

  $$\mathbf{LS}_1 \triangleq \{v_k \in \mathbf{LS} \mid a_* < v_{**}\}$$
  $$\mathbf{LS}_2 \triangleq \{v_k \in \mathbf{LS} \mid a_* \geqslant v_{**}\}$$

  correspond to the two successors of the root.
- The successors are labelled terminal or nonterminal on the basis of the stop splitting criterion described below.

  For the nonterminal nodes, the overall procedure is called recursively, to build the corresponding subtrees. For the terminal nodes, the class probabilities p_+ and p_- are estimated on the basis of the corresponding subset of learning states there stored, and the class label of the majority class is attached, possibly along with a weighting factor expressing the class probability ratio.[e]

Obviously, the crux of the entire construction of a DT lies in the selection of the splits and the decision whether to declare a node terminal, or to continue splitting. These questions have been considered in References 4 and 16. Their thrust is recalled below.

II.2.2 *'Optimal' splitting and test attributes*

The above strategy consists of considering the best test to be the one which separates at most the states of the two classes in the local learning subset, i.e. which provides the purest *direct successors*. It is therefore *locally*, rather than *globally optimal*.

This locally optimal test selects at each node the *test attribute* along with its threshold value having the best '*score*', in the above, local sense. This score is assessed via the normalized information gain defined by equation (A.5) in the Appendix.

II.2.3 *The stop splitting criterion*

The decision tree construction algorithm is based on ID3[14]. An essential difference of the proposed methodology with respect to it resides in the stop splitting criterion. Indeed, ID3 stops splitting at a node only if the corresponding learning subset is completely class pure. According to our experience, however, with applications investigated so far, this strategy tends to build overly complex DTs with terminal nodes generally containing only a very small and unrepresentative sample of learning states; this contributes to increasing the complexity of the tree thus decreasing its interpretability of the phenomena, and also to weaken its ability to classify correctly unseen states (i.e. states not belonging to the learning set); in short, it decreases the tree effectiveness. To circumvent this difficulty, a more conservative criterion has been proposed in Reference 3, which stops splitting a node as soon as one of the following two conditions is met.

(1) The local subset of learning states is 'sufficiently' class pure; we henceforth refer to such a terminal node as a *leaf*. The degree of required class purity is fixed by means of a parameter of the algorithm specified by H_m, the maximum residual entropy (see in the Appendix). In practical investigations, we use the same constant value of H_m for all nodes (generally $H_m = 0.1$ or 0.01 bits). Such very low values would yield very detailed DTs; thus, most of the time, the stop splitting criterion actually relies on the second condition, described hereafter.

(2) There is no possibility of enhancing the tree accuracy in a statistically significant way by splitting the node further. Such a node is called a *deadend* terminal node. This condition for the stop splitting criterion can be formulated as a statistical hypothesis test:

    Given the states belonging to the node under consideration and the corresponding optimal split, can we accept the hypothesis that the apparent[f] increase in classification accuracy of the tree resulting from splitting the node further is a purely random effect?

    In quantitative terms:

    Under the hypothesis of no real increase in purity (i.e. the apparent increase of classification accuracy due to node-splitting is only a random effect) the statistic

    $$\chi^2 \triangleq 2 * \ln 2 * N_{\mathrm{node}} * I_C^T(\boldsymbol{S})$$

    is distributed according to a χ-square law[g].

    In the above, N_{node} is the number of states in the learning subset corresponding to the current node and $I_C^T(\boldsymbol{S})$ the *apparent* information provided by the optimal split, defined by equation (A.1) in the Appendix.

Hence, if we fix the α-risk of not detecting these situations, testing the value of χ^2 against the threshold χ^2_{cr} such that

$$\mathrm{Prob}\,(\chi^2 \geqslant \chi^2_{\mathrm{cr}}) = \alpha$$

allows one to detect with a probability of $(1-\alpha)$ the cases where the apparent increase in accuracy is a random effect. Figure 2 sketches such χ-square probability density functions.

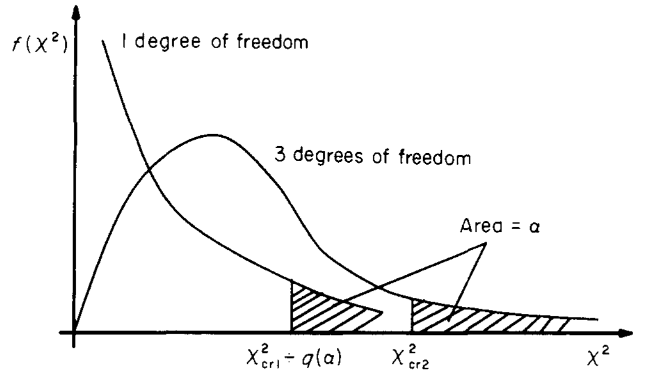


Figure 2. χ-square probability density functions of $2 * \ln 2 * N_{node} * I_C^T(\boldsymbol{S})$

Thus, the α-risk of the hypothesis test fixes the amount of evidence required at each node to split it; the outcome depends on the value of α, on the size of the local learning set, and on the amount of apparent information provided by the test. The question of 'how much evidence should be required to allow the splitting of a node', is related to the degree of representativity we impute to the learning set and the risk that this degree is overestimated. It is fixed by the user via the α value, which ranges from 1 (the criterion has no effect on the splitting procedure any more: the tree grows according to the above condition 1) to zero (no growth is allowed: the tree reduces to its root).

The value of α has indeed drastic effects on the resulting tree characteristics as will be illustrated in section III.

II.2.4 *On the right size of trees*

In general, the choice of α should be guided by the observation that too large a tree will yield a higher 'misclassification rate' and hide relevant relationships among more or less random details, whereas too small a tree will not make use of some of the information available in the **LS**. Indeed, the terminal nodes of too small a tree have not been expanded enough and this prevents getting purer subsets and the corresponding insight about the role that the attributes would have played in this expansion; too large a tree, on the other hand, results from the splitting of statistically unrepresentative subsets; therefore, it is likely to cause an increase in the misclassification rate when classifying states not belonging to the **LS**. Stated otherwise, a tradeoff appears between the two following sources of misclassification: *bias* (overlooking significant information in the **LS**) and *variance* (badly interpreting the randomness in the **LS**): too large a tree will suffer from variance whereas too small a tree will present bias[12].

III.2.5 *Tree effectiveness and its evaluation*

The reliability of a tree is its ability to classify properly unseen cases. Among the various criteria proposed to assess it, or equivalently its misclassification rate, we mention the resubstitution, the cross-validation, and the test sample estimates. Observe that all these are only estimates, since in real-world problems, even of deterministic type, scanning all the possible states is scarcely feasible.

The *resubstitution* estimate is defined as the proportion of correctly classified *learning states*. It is generally optimistically biased[12,20], in particular if the DTs fit too closely to the **LS**.

The *cross-validation* method provides a rather unbiased estimate (e.g. see Reference 20 for a description). It requires however quite large sample sizes in order to reduce its initially larger variance. Since it moreover requires the rebuilding of several (e.g. $\geqslant 10$) DTs, its computation may become cumbersome[12].

In the context of our investigations we have used the *test sample estimate*, which is obtained by considering all M states belonging to the *independent* test set, dropping them through the tree, and computing the proportion of misclassified states[h]. This is recognized to be the most reliable and unbiased estimate, provided M is sufficiently large (e.g. $M \geqslant 1000$)[12]. Another advantage is that its variance can be easily estimated; therefrom, confidence intervals may be computed. Hence, despite the drawback of requiring additional test states in sufficiently large number, we use it as the benchmark for accuracy estimation (the reciprocal of misclassification rate). *Accuracy* or *reliability*[i], together with *simplicity* (or its reciprocal, *complexity*[j]), determine the tree *effectiveness*.

II.2.6 *Note on the quantization of continuous valued attributes*

The automatic tree building procedure of section II.2.1 does not require any prior quantization of continuous valued attributes. Rather, the method determines at each test node of the tree the relevant threshold values in the following way

(1) the local learning set of the node is sorted by increasing order of the values of the considered attribute;

(2) let $v_{min} < v_2 < \ldots < v_{max}$ be the sorted sequence of values: a corresponding sequence of candidate threshold values $(v_2 + v_{min})/2 < (v_3 + v_2)/2 < \ldots$ are defined for test (6), and the one yielding the highest 'score' is selected as the optimal one;

(3) this procedure is repeated for each numerical attribute (continuous or discrete), at each node candidate for splitting.

Thus, if the learning set contains dense (resp. sparse) attribute values, a dense (resp. sparse) set of thresholds will be considered. In power system security applications, the number of different thresholds considered at a node is typically of the same order of magnitude as (and upper bounded by) the number of learning states (≈ 100 to 1000).

This technique allows one to exploit fully the information contained in the continuous attributes, at the expense of a lesser computational efficiency, in contrast to pre-quantization techniques which determine a restricted set of candidate threshold values prior to the DT building, repeatedly used at each test node[21–23]. However, the projection sort algorithm used in Reference 17 to find the optimal threshold for each numeric attribute reduces the computational complexity and succeeds in keeping efficient the optimal splitting procedure (see the method's computing requirements in section III.4 below).

II.3 *'Quality' of a tree*

A convenient means of quantifying the 'simplicity vs. reliability' compromise is provided by the quality measure proposed in Reference 16 and expressed by equation (A.8) of the Appendix. Indeed, this expression indicates quantitatively that, the larger the **LS** (and hence the higher the reliability likely to be expected of the tree), the more complex the trees that may be induced, provided they allow for enough explanation of the available data. Stated otherwise, the fewer the number of available learning instances, the less complex the tree one might reasonably infer from these data, without overfitting them. This is also in agreement with the well known fact that the more complex the relation that is to be learned, the larger the **LS** that is required to be to get a *precise* (of low residual uncertainty) and a *reliable* tree.

Quantitatively, equation (A.8) yields the following necessary condition in order to obtain a positive quality (i.e. a more credible tree than the trivial 'attributes vs. classes independence hypothesis')[k].

$$C(DT) < \frac{N * I_C^{DT}(\mathbf{LS})}{q}. \tag{7}$$

This indicates that the complexity of induced trees could increase at most linearly with N, the number of available learning samples. This behaviour was corroborated by many empirical observations, such as those described in section III.

The above quality measure has many interesting properties[16]. Theoretical type ones are pointed out in the Appendix. Among its nice practical features we mention the following.

Additivity. Given an arbitrary decomposition of the tree into subtrees, the total tree quality is equal to the sum of the qualities of its component trees. In particular, the tree quality is equal to the sum of the qualities of its test nodes. This property is exploited below, in the formulation of a recursive bottom up algorithm for the optimal tree pruning.

Generality. The measure Q expressed by (A.8) allows one to assess objectively and compare different trees. It may be used either during the tree growing stage, in order to select tests to split nodes and to decide when to stop splitting, or *a posteriori*, for example to simplify too complex trees by appropriately pruning them (see below).

Equivalence. It is shown that the elementary steps of the previously described inductive inference method are (almost) equivalent to choosing tests which improve as much as possible the quality (A.8) of the growing tree[16]. On the other hand, the deadends identified by the χ^2-based stop splitting criterion correspond to local maxima of quality. Moreover, there exists a correspondence between α and q (low values of α correspond to high values of q see Figure 2 and equation (A.9) of the Appendix).

A great number of other, more or less empirical, quality measures have been proposed in the literature for the evaluation of decision trees. Different measures are generally used at the tree growing stage[24] and at the subsequent tree pruning stage[25]. The theoretical properties of the proposed *global* quality measure, along with the above practically interesting features, however allow its use at both stages. For example, in the general framework proposed in Reference 16, node development and tree pruning are two particular search operators, which cooperate in order to find a tree of maximal quality; node development allows one to search in the direction of increasing complexity and pruning in the direction of increasing simplicity.

II.4 *Two pruning approaches*

We noted that a compromise should exist between simplicity and reliability. Two different approaches may be suggested to realize it.

The first consists of preventing unnecessary node developments *during* the tree growing[4,13]. In our method, this is achieved by using low values of α in the stop splitting criterion.

The second approach consists of controlling the tree size *after* the complete tree growing, by using the *pruning* approach[25].

More specifically a tree is selected in the following general fashion[12,16,25].

(1) First build a 'maximal' tree by setting $\alpha = 1$ (or equivalently $q = 0$) during the tree growing stage.
(2) Generate therefrom a sequence of simpler trees of decreasing complexity by using an appropriate pruning method, as described below.
(3) Select one of these trees as the final tree. This step generally relies on an independent test set, and consists of choosing the tree of minimal test set error rate.

In the context of the quality measure defined by equation (A.8) we propose the following two pruning methods.

Method 1. Forward pruning

For a given value of α, prune the DT by replacing each test node for which

$$N_{\text{node}} * I_C^T(S) \leqslant q(\alpha) \tag{8}$$

by a terminal node[l]. (This will produce exactly the same tree as when using the corresponding value of α in the stop splitting criterion during the tree growing.) For α decreasing from 1 to 0 this will yield a (finite) sequence of trees DT, DT_1, $DT_2, \ldots$ of decreasing complexity.

Method 2. Optimal (backward) pruning

For a given value of q, extract the pruned tree of *maximal* quality.

Due to the addivitivy of the quality measure, the optimal pruning of a DT can be achieved in a single bottom up pass as described by the following recursive algorithm:

- if the DT is trivial then no pruning could be achieved; the optimally 'pruned' tree is the DT itself;
- otherwise the subtrees corresponding to the direct successors of its root are first pruned optimally; and the quality of the DT is corrected by adding the corresponding quality improvements obtained for the subtrees (the latter are necessarily non-negative, since the subtrees are optimally pruned);
- if, after this step, the quality of the resulting DT is strictly positive, either because it was initially positive, or because the pruning of its subtrees has improved it enough, then the latter DT is the optimally pruned DT and is returned as the solution;
- otherwise, the resulting quality is non-positive and the DT is pruned by replacing its root by a leaf; the optimally pruned tree reduces to the trivial tree.

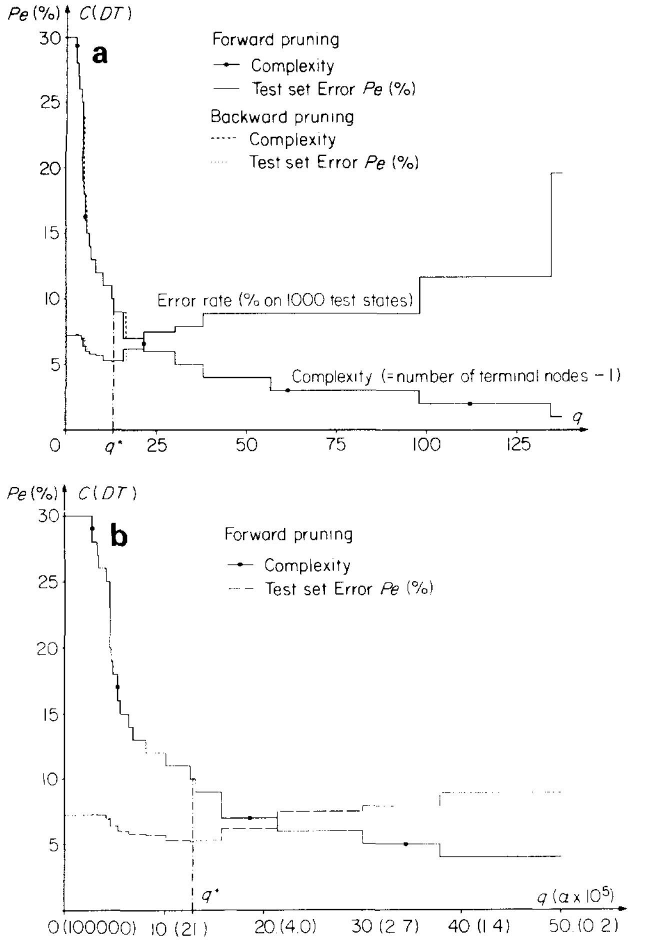


Figure 3. (a) Forward and backward pruning sequences for the DT of Figure 15. (b) Forward pruning sequence for the DT of Figure 15 (magnified view)

Thus the result of pruning a DT is either a trivial DT, of quality equal to zero, or a non-trivial DT of (strictly) positive quality and whose non-trivial subtrees are also of strictly positive quality. Consequently, the deepest test nodes of a pruned DT have to be of positive quality. One shows that for q increasing from 0 to $+\infty$ this produces also a sequence DT', DT_1', DT_2', ... of trees of decreasing complexity[26].

Although the two pruning methods are not necessarily equivalent, in practice only little difference is observed between the corresponding sequences of pruned trees. For example, in the context of trees that are grown according to the information theory criterion of ID3, and if minimum test set error rate is used as the criterion for selecting the pruned tree, the final trees produced by the two methods often appear to be identical.

This is also illustrated by the example of Figure 3, which shows the plot of the complexity $C(DT)$ and test set error rate P_e of the sequence of pruned trees obtained by varying $q(\alpha)$, for fixed ***LS*** and ***TS***. They correspond to the tree of Figure 15 of section III.3, built in the context of preventive voltage stability assessment. Figure 3a is drawn for both forward and backward pruning methods; observe that the differences between the corresponding curves of complexity and of test set error rate are negligible: both pruning procedures provide the same 'optimal' tree minimizing P_e. The characteristics of this tree are more easily depicted in Figure 3b which gives a magnified view of the curves corresponding to forward pruning in the range of the more interesting q values, near the minimum of P_e: the optimal tree (see Figure 14) is seen to correspond to $q \approx 13$ ($\alpha \approx 4.85 \times 10^{-5}$), $C(DT) = 10$, and $P_e = 5.2\%$.[m] Observe that, according to Figure 3, the pruning allows here a decrease in complexity of 70% and simultaneously in error probability of almost 30%. These figures are quite typical for the applications we have considered so far.

In the sequel we will consider forward pruning only, since it is equivalent to the stop splitting rule used so far.

II.5 *Tree and its dual representation, the attribute space*

A 'good' tree offers a clear description of the phenomena of concern. It also enables one to readily classify a state of *a priori* unknown classification on the basis of the known values of its *test attributes* (remember, these are the attributes appearing at the test nodes of the tree): beginning at the root of the tree, one merely has to sequentially apply the test at the test nodes and systematically move the state to the successor corresponding to the test's outcome, until a terminal node is eventually reached; the state is classified accordingly.

Another, very interesting classifier is provided by the dual representation of a DT in its 'attribute space'. The coordinates of this space (or hyperspace) are the test attributes of the tree[l]. This dual representation is an elegant geometric interpretation obtained by partitioning the space into regions corresponding to the classes. Each region is composed of the union of hyperboxes of the terminal nodes assigned to the class[9,10]. Figure 4 portrays the attribute space corresponding to the tree of Figure 1. (Admittedly, this tree and its corresponding attribute space are much simpler than those usually obtained; they were chosen for the purpose of demonstration.)

An interesting outcome of the attribute space is the possibility of defining *distances*. These may be used to quantify the degree of security of a state and to identify the 'locally relevant' parameters, as the more strongly affecting the state's distance to the security boundary. The latter sensitivity type information may further help identify possible on-line control strategies.

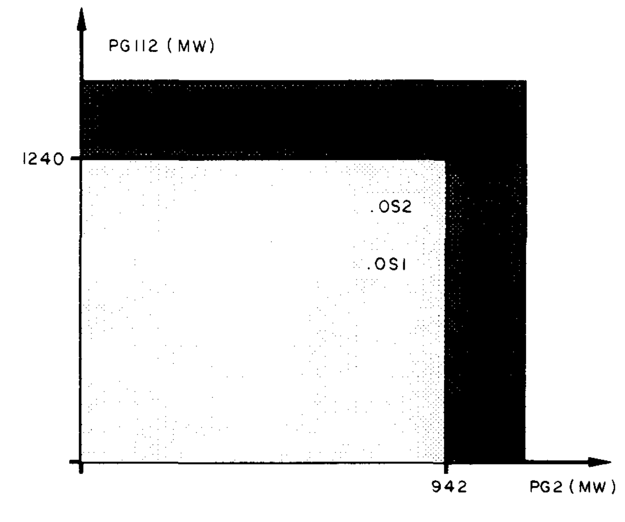


Figure 4. (From Reference 4)

III. Applications to power system security assessment

The general principle of the decision tree application to power system security is first stated (section III.1), then tailored to various types of dynamic security assessment (section III.2), finally illustrated on a sample of practical results (section III.3), which highlight general features (section III.4).

III.1 *Principle*

It follows the same general pattern: 'for the particular security problem at hand, build DTs on the basis of "relevant" information, to assess subsequently any new, unseen operating state in terms of the values of its test attributes'. Let us briefly comment on the terms of this general statement, in the context of security assessment.

The 'relevant' information concerns:

(1) 'relevant' operating states, preclassified with respect to a practical criterion characterizing the security problem of concern;
(2) a list of candidate attributes 'relevant' to this security problem, i.e. likely to govern the phenomena, and available in the on-line environment.

Depending on the particular application, information (1) may be obtained by:

- using past records of the system real life, and considering plausible scenarios of load-generation-topology schemes for which a load flow program is run to generate the corresponding operating states;[o]
- considering (a list of) contingencies relating to the security problem of concern; further, considering an appropriate method to assess security, i.e. to classify each operating state with respect to each such preassigned contingency. This sample of preclassified operating states constitutes the 'data base'. Subsequently, it will be decomposed into learning and test sets.

Obviously, information (1) implies the use of numerical programs, in particular those related to the 'system theory' security method[p].

Information (2) is concerned with parameters of the system likely to drive the phenomena; they must be easily available in the on-line environment (e.g. to the operator), so as to make the use of DTs independent of the 'system theory' security method. In addition, human expertise is fundamental for appropriate choice of contingencies, precontingency operating states and candidate attributes altogether; in short, it is essential to the success of the devised DT method.

Stated otherwise, a 'good' DT-based security method results from the combination of a good DT methodology, good 'system theory' security methods and numerical programs, good system modelling and sound human expertise. The expected advantages of this combination over the corresponding 'system theory' security methods are manifold. Anticipatively, we identify three areas, corresponding to three types of assessment:

- analysis type assessment: a DT identifies the more *relevant* parameters to the security problem of concern; once the values of the test attributes of a state are known, its classification by means of the DT is almost instantaneous;
- sensitivity type assessment: it can be efficiently obtained through distance computations in the attributes space; 'system theory' security methods are less suited to this kind of assessment;
- control type advices: it can also be obtained in the attribute space; in general, no such counterpart is available in 'system theory' security methods[q].

Recall that all this information may be assessed with any degree of system modelling sophistication and any 'system theory' security method, since these auxiliary tools are essentially used to build DTs, during the off-line phase.

The above advantages are obtained at the expense of a bulky off-line data base acquisition. A convenient data base management tool is of great help.

III.2 *Application to various types of security assessment*

The general principle stated above is now tailored to the specifics of various security issues.

III.2.1 *Conventional transient stability assessment: the decision tree transient stability (DTTS) method*

Transient stability is concerned with the ability of an electric power system to withstand severe disturbances. A conventional measure of this is the *critical clearing time (CCT)*, i.e. the maximum duration that the disturbance may remain without causing the irrevocable loss of machines' synchronism. Two broad families of 'system theory' methods are currently used to compute CCTs: the time-domain methods, which solve numerically the nonlinear differential equations governing the system motion, and the direct methods which rely on the Lyapunov criterion[27,28]. Admittedly, the CCT may not be the most appropriate information about transient stability: it merely provides a rather crude 'yes–or–no' answer, whereas what often matters in practice are 'stability margins' in terms of operating parameters and, where necessary, advice for remedial actions in terms of control variables. However, the CCT is a good benchmark, in particular for comparisons. Within the tree methodology it provides a handy means of classifying the learning states into classes of increasing stability with respect to a disturbance. The considerations of section III.1 suggested that from there on its is possible to get sensitivity and control tools.

In the basic DTTS method each tree is built for a single, preassigned contingency. Two main observations underlie this approach. First, transient stability is strongly dependent upon the contingency type and location; hence, the idea of building a tree per contingency. Second, transient stability is quite a localized phenomenon, driven by a few parameters of the system in its precontingency condition; (this assumes that there are indeed sound correlations between precontingency parameters of the system and its transient stability robustness); hence, the idea of selecting candidate attributes among the system steady state parameters. One may distinguish topological discrete attributes and electrical continuous ones, such as power flows, power injections, voltages.

The sample of precontingency operating states required for a tree building is obtained as indicated earlier, by considering a variety of scenarios and running for each one of them a load flow program to get the corresponding operating states. The classification of these states with respect to a contingency is performed by means of a stability method (time-domain is preferable to direct methods whenever better accuracy or modelling sophistication is sought).

The DTTS method has recently been extended to handle through a single tree several (related) contingencies (e.g. different faults located in the same substation). Among different explored ways of defining multicontingency trees let us succinctly describe the following. We consider each 'stability case' to be specified by a prefault operating state (characterized by prefault attributes, as above) and a fault, identified by a discrete attribute; it is accordingly classified as unstable if its operating state is unstable with respect to its fault. The learning and test sets are therefore composed of collections of such 'stability cases'. A DT built with such a learning set will be able to provide directly either of the following types of information.

- For a given fault (among those used to build the tree) and operating state, is the corresponding stability case likely to be unstable or not?
- For a given operating state, which are the faults corresponding to an unstable behaviour?
- Which conditions characterize the prefault attributes of stable operating states for a given set of possible faults?

Such a multicontingency approach is illustrated in section III.3 by the DT of Figure 13. It suggests that, although equivalent to the information provided by a set of single-contingency trees, the information provided by the corresponding multicontingency tree is presented in a more compact and easier to exploit fashion. This can be explained by the fact that similarities of different contingencies are exploited during the tree building so as to simplify the resulting tree. In particular, overlappings of unstable (resp. stable) regions are identified and imbedded in the tree: hence, combinatorial explosion, inherent in multicontingency control on the basis of single contingency trees, is avoided.

III.2.2 *Preventive voltage stability assessment*

The proposed method is a replica of the DTTS: it assesses the ability of a precontingency state to withstand a preassigned contingency in terms of the state parameters preselected by the tree, built for this contingency.

The precontingency states used for the tree building are obtained in a way similar to that of section III.2.1. The contingencies of concern here are generally outages of EHV transmission and/or generation equipment.

The classification of a precontingency state with respect to a given contingency may rely on various 'system theory' techniques[6]. One such technique is a load flow; the considered state will be classified 'stable' or 'unstable' according to whether the load flow converges or not towards an acceptable post-contingency operating state. Feasibility limits, such as upper and lower bounds on voltage magnitudes are evaluated. Admittedly, this type of classification is less refined than the critical clearing time used in transient stability; a more appropriate approach would be to consider adequate 'voltage collapse proximity indicators' to characterize the weakness of the post-contingency state (see e.g. References 29 and 30 and the references therein, for some possible such measures).

Finally, as in transient stability, the candidate attributes are parameters of the system in its precontingency state, relevant to voltage stability.

It is worth mentioning that References 22 and 23 propose a quite different tree approach, for the purpose of voltage optimization.

III.2.3 *Emergency voltage control*

The proposed approach and resulting procedure are quite different from the previous ones. The leading idea is that voltage instability following a contingency generally does not develop as fast as the transient one (typically voltage collapse takes several minutes whereas electromechanical loss of synchronism takes only a few seconds); this leaves time to detect the potentially critical states *after* the contingency occurrence and to take corrective actions[7].

A main difference of this method with respect to previous approaches is thus the type of considered system states. They result from various operating conditions, supposed to be subjected to a set of disturbances; they are determined after a short-term intermediate equilibrium has been reached, i.e. after the electromechanical transients have vanished (approximately 10–20 s after the disturbance inception). Such 'just after disturbance' (JAD) states along with their classification (non critical if the state ultimately reaches a new equilibrium, critical otherwise) are used to build a tree, which therefore is relative to a set of disturbances. Subsequently, the tree will be used on-line to decide whether, following a disturbance on the system, and in terms of attributes observed in the JAD state, the system state is critical or not. In the former case, the tree itself or its corresponding attribute space should be able to suggest some fast corrective actions.

Similarly, the candidate attributes proposed to the tree construction are parameters of the system in its JAD state.

Remarks

(1) One might *a priori* think that considering a set of contingencies, rather than a single one, would provide less accurate trees. This however is compensated by the rich information carried by the attributes about the disturbance undergone by the system. (This is corroborated by the simulations: during the tree construction, the method consistently selects attributes of the JAD type preferably to 'before disturbance' type.)

(2) To determine and classify JAD states many techniques may be used ranging from successive steady-state calculations up to refined dynamic simulations.

(3) Instead of the mere 'critical-noncritical' states classification, one might think of more refined ones; e.g., one could introduce the degree of criticality in various ways, for example in terms of how fast the system would evolve towards collapse.

(4) Another issue worth exploring is how many disturbances a single tree can handle effectively or, in other words how many trees should be built to cover properly all relevant disturbances located in a given geographic area. Appropriate electrical distances could be quite useful in this respect.

(5) Though the above emergency-wise approach is still in its infancy, the results of the academic-type example reported below show that the method has potential.

III.3 *Sample of illustrative examples*

We illustrate the decision tree security approach by means of the DTTS method which is the more thoroughly explored so far. We then provide a sample of trees obtained in the contexts of preventive and emergency voltage control.

III.3.1 *Decision tree transient stability (DTTS)*

Four different power systems of growing size have been used to test the DTTS method: a one-machine/infinite bus, academic type system; a 14-machine/92 bus, real system; a 31-machine/128 bus, synthetic system; and a 61-machine/561 bus, existing system. All in all, over 2000 decision trees have been built. The following range of parameters has been covered by the simulations:

- 36 different disturbances of several types (short-circuit located at generator (or load) buses, with (or without) single (or multiple) line (and/or machine) tripping);
- 2, 3, 4 stability classes;
- from a few to over 150 candidate attributes per tree;
- size of ***LS*** and ***TS*** ranging from 100 to 6000 states;
- multicontingency trees.

Below we illustrate the type of trees obtained on a detailed typical example, then we outline general salient features.

A detailed account of a typical decision tree

Figure 5 portrays a representative 2-class tree obtained on an earlier version of the French 400-kV 225-kV system comprising 561 nodes, 810 lines, 190 transformers and 61 machines; the tree was built for a 'busbar' fault, i.e. a 3-phase short circuit applied at a busbar and afterwards cleared by opening all lines terminating at the faulted busbar section[5]. The data base was generated from a base case modified according to predefined probability laws[r]. 3000 operating states were thus generated; they were classified with respect to the above contingency: stable (resp. unstable) if their critical clearing time (CCT) is higher (resp. lower) than a threshold CCT value[s]. (In Figure 5 this threshold is fixed at 155 ms, corresponding to the actual time required by the protection system to clear the fault.) Out of the 3000 states of the data base, 2000 are composing the learning set; the remaining 1000 the test set. The 57 candidate attributes used to build the tree comprise the following parameters (notice that the categories (1) to (4) are actually directly controllable attributes):

(1) active generation of each unit of each plant of the region;
(2) their EHV voltage magnitude;
(3) global regional active and reactive load;
(4) topology attributes, such as the status of lines and generation units and the number of nodes in the 400-kV substations around the faulty busbar;
(5) active and reactive power flows on the important lines in operation;
(6) '*ad hoc*' composite attributes, taking into account the prefault operating point and the postfault topology.

Before commenting on the information provided by the tree of Figure 5, let us explain the notation consistently used for the tree representation in all security applications. Each node of the DT is represented by a box. Above the box appears the node name, labelled T (test), D (deadend) or L (leaf) as appropriate, along with the number of its learning states. (Remember, a leaf is a node where the states of different stability degrees have been well enough separated; a deadend is a node where the method decides that the learning subset has shrunk too much to allow its further splitting in a reliable way[4].) The total number of different types of the tree nodes is indicated near the top-node. The node box itself is subdivided into upper and lower parts. Their relative height is proportional to the relative size of the learning and test sets at the node. The upper part is subdivided into differently shaded parts to indicate the ratio of learning states of the different stability classes. The lower part is subdivided into a white and black area, indicating the relative proportion of the test states correctly and incorrectly classified by the complete subtree corresponding to the node.

Coming back to the tree of Figure 5 we note that out of the 57 candidate attributes the method has selected only 8 to formulate the tree. Some of these are discrete and characterize the topology (e.g. 'P4-Post.Fault = isolée' is a logical attribute indicating whether a particular generation unit is isolated or not; 'P4 = en-serv' indicates whether unit P4 is in service or not); some others are real valued ones, describing the operating state (e.g. 'Q-P4' is the reactive generation of unit 'P4'; 'V-P4' denotes its EHV bus voltage magnitude).

The tree building is summarized in Table 1. For each test node of the tree this table lists the number of its unstable and stable states, and the three most discriminating attribute tests along with their 'score' G_C^T (see (A.5) in the Appendix). Before commenting on Table 1, let us derive in detail the 'score' of the test 'P4-Post-Fault = isolée' selected at the top-node, which splits the overall learning set into the two subsets corresponding to the nodes T2 and T3. It is obtained by using the formulae of the Appendix to successively calculate the following entropies and amount of information (in bits):

$$H_C(\boldsymbol{LS}) = -\left[\frac{152}{2000}\log_2\frac{152}{2000} + \frac{1848}{2000}\log_2\frac{1848}{2000}\right] = 0.388 \qquad \text{(Top-node)},$$

$$H_T(\boldsymbol{LS}) = -\left[\frac{94}{2000}\log_2\frac{94}{2000} + \frac{1906}{2000}\log_2\frac{1906}{2000}\right] = 0.274,$$

$$H_C(\boldsymbol{LS}_Y) = -\left[\frac{78}{94}\log_2\frac{78}{94} + \frac{16}{94}\log_2\frac{16}{94}\right] = 0.658 \quad \text{(T2)},$$

$$H_C(\boldsymbol{LS}_N) = -\left[\frac{74}{1906}\log_2\frac{74}{1906} + \frac{1832}{1906}\log_2\frac{1832}{1906}\right] = 0.237 \qquad \text{(T3)},$$

$$H_{C/T}(\boldsymbol{LS}) = \frac{94}{2000}H_C(\boldsymbol{LS}_Y) + \frac{1906}{2000}H_C(\boldsymbol{LS}_N) = 0.257,$$

$$I_C^T(\boldsymbol{LS}) = H_C(\boldsymbol{LS}) - H_{C/T}(\boldsymbol{LS}) = 0.131,$$

$$G_C^T(\boldsymbol{LS}) = \frac{2I_C^T(\boldsymbol{LS})}{H_C(\boldsymbol{LS}) + H_T(\boldsymbol{LS})} = 0.397.$$

Scores, as those given in Table 1, provide valuable information about the relative importance of the attributes. They allow one to readily identify the significant ones and to assess their correlations or complementarities. Attributes obtaining repeatedly similar scores at

Table 1. Scores of the attributes during the building of the DT of Figure 5

Node N_{unst} N_{st}	Attribute and test	G_C^T	Node N_{unst} N_{st}	Attribute and test	G_C^T	Node N_{unst} N_{st}	Attribute and test	G_C^T
Top	P4-Post-Fault = isolée	0.397	T2	P4 = en-serv	1.000	T3	Q-P4 < -100	0.134
152	P-Per-L <1853	0.309	16	P-PERT <137	1.000	74	P-Per-L <1232	0.129
1848	Nb-L <1	0.308	78	P-P4 <137	1.000	1832	V-P4 <396	0.112
T4	P-Per-L <1255	0.445	T5	Remise <2	0.603	T6	P-Decl <2356	0.130
109	Nb-L <3	0.372	39	Q-P4 < -167	0.173	35	Rou2-Q < -265	0.112
39	Barnabos <2	0.338	26	Q-P2 < -156	0.145	1723	P-Per-L <1630	0.105
T7	Rou2-Q < -265	0.126	T8	V-P4 <394	0.103	T9	P-Per-L <1230	0.284
31	V-P4 <395	0.111	26	Ter1-Q < -78	0.097	19	Ter2-P <1137	0.179
1722	V-P2 <395	0.099	1717	V-P2 <395	0.090	131	P-P4 <1197	0.129

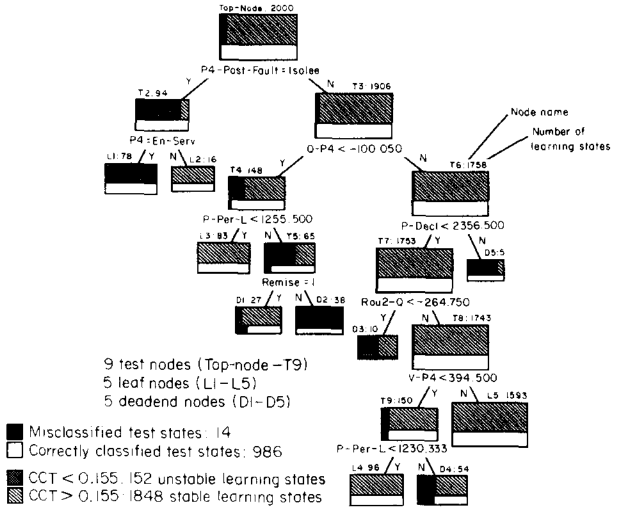


Figure 5. 61-machine system. Three-phase busbar fault. 2 classes. $N = 2000$, $M = 1000$, $\alpha = 1.0 \times 10^{-4}$, $P_e = 1.4\%$

several nodes of a tree are likely to be correlated, i.e. to provide similar information about the stability behaviour. Attributes obtaining high scores at different tree nodes reflect physically independent, complementary phenomena.

Considering node T2, one observes that the three first attributes obtain an identical score equal to 1, meaning that they classify the local learning set perfectly. In such a situation the method chooses the attribute appearing first in the list of candidate attributes, unless otherwise decided by the tree designer. Note that in the particular case of T2 the three tests are actually equivalent, from the physical point of view.

Among the more discriminating attributes (i.e. those providing the larger classification ability) is probably the qualitative test attribute of the tope node; indeed, inspection of the tree parameters shows that this attribute allows one to classify about 1800 learning states out of the 2000 ones. This is corroborated by the information quantity $N_{\text{node}} * I_C^T(S)$ provided by each one of the eight test attributes; we indeed find:

P4-Post-Fault: $2000*0.131 = 262.6$ (at the Top-node)
P-Per-L: $148*0.406$ (at T4) $+ 150*0.211$ (at T9) $= 91.5$
Q-P4: $1906*0.042 = 81.0$ (at T3)
P4: $94*0.655 = 61.6$ (at T2)
V-P4: $1743*0.027 = 47.9$ (at T8)
Remise: $65*0.58 = 38.0$ (at T5)
Rou2-Q: $1753*0.0113 = 19.9$ (at T7)
P-Decl.: $1758*0.0110 = 19.4$ (at T6).

The total information quantity provided by the tree is the sum of the above eight numbers and equals 621.9. Note that the total information quantity potentially contained in the ***LS*** is $2000*0.388 = 775.9$: thus, the tree classification has decreased the impurity (entropy) by about 80%.

General observations

- The tree building method shows itself to be an efficient attribute *selection* tool. (In the above example it selected 8 out of the 57 attributes to classify the states.)
- The obtained tree is easily interpretable thanks to the elementary nature of the candidate attributes, its very low complexity and its hierarchical structure (which highlights explicitly the more significant relationships between the driving variables and the stability of the system). This is admittedly of paramount importance in practice, since it is a prerequisite to allow the stability experts to cross-validate a DT with respect to their own knowledge of the problem, and vice versa. Moreover, since the DT is directly formulated in terms of the operating variables, it can be easily understood by operators and can therefore effectively guide decision making in real time operation.
- The method provides an effective data analysis tool, thanks to its quantitative information such as attribute scores and information quantities, together with the tree interpretability feature.
- Although quite simple, the tree turns out to be very efficient in discriminating stable from unstable states: only 14, out of the 1000 unseen test states, are misclassified; moreover, this is achieved in spite of the rather high disproportion of stable (1848/2000) and unstable (152/2000) learning states.

Let us focus on the misclassification rate $P_e = 1.4\%$. A close examination shows that the misclassified states are concentrated in the three deadends D1, D3 and D4 of the tree of Figure 5: according to the class majority, D1 and D4 are labelled stable, D3 unstable. Thus, the states misclassified by D1, D4 are assessed to be stable whereas they actually are unstable; those misclassified by D3 are assessed as unstable whereas they are stable. A closer analysis, obtained by considering the actual CCT values of the misclassified states, shows that among the 14 states, 10 fall into the ±10% error range (140–170 ms) around the threshold of 155 ms; we will consider them as 'normal' errors; 3 states fall into the range of (110–140) ms: they are the 'dangerous', 'abnormal' errors; the last state falls above 170 ms: it is an 'abnormal' false alarm.

Note that a more cautious strategy will consist of suitably weighting the diagnostic of a deadend by the class probability corresponding to its learning states, instead of simply considering its class majority. This will provide a more refined diagnostic, especially for the states falling in D3 and D4; this will be further discussed in section III.4. Another convenient and more accurate way of assessing stability, especially for borderline cases, could be provided by the attribute space representation of the tree by considering their distance to the boundary between stable and unstable regions of this space[9]. Further investigations are being pursued to make this attribute space approach fully efficient.

- Another issue of concern is the number of classes considered. Figure 6 illustrates a typical evolution of a tree with the number of classes: this 4-class tree is obtained under the same conditions as the 2-class of Figure 5. Observe the increase in complexity and in misclassification rate, as normally expected. This is further discussed in section III.4.

After general considerations illustrated on particular trees, let us now give a statistical, global assessment of their behaviour.

Effect of the parameter α

To explore the influence of α, we have considered the 31-machine system and have simulated disturbances consisting of 3-phase short-circuits, applied at machine nodes[4]. Figures 7 and 8 describe the effect of the parameter α on the size and the accuracy of trees, for 2, 3, and 4 classes. In each class the curves represent mean values for 12 different trees corresponding to 3 disturbances and different randomly selected learning sets of $N = 500$ states; each tree was tested by computing its error rate on the basis of an independent test set composed of $M = 1500$ states.

From these figures, the following observations can be made:

- the most important reduction of the tree complexity is obtained as soon as α enters the range 10^{-3} to 10^{-4}; quantitatively, this effect is more marked in the 3- and 4-class trees;
- although further lower values of α still reduce (sometimes notably) the tree size, the effect is generally less impressive;
- the trees corresponding to $\alpha = 5 \times 10^{-5}$ can be two to ten times smaller than those corresponding to $\alpha = 1.0$; besides
- when α decreases from 5×10^{-2} to 5×10^{-5}, the tree accuracy varies very slowly, and generally insignificantly (cf. the statistical uncertainty of the error estimates).

Similar results have been obtained in the context of other disturbances and other power systems. Table 2 provides some typical figures for the obtained tree parameters.

Finally, an important observation is that within a given application (here transient stability), the 'optimal' value of α is quite independent of the learning set size and of problem specifics like disturbance type and location, number of classes and power system. Consequently, the test set error based tree selection of the pruning approaches of section II.4 is required only during preliminary studies, in order to identify the optimal α value. Subsequent trees may use the same α and can thus

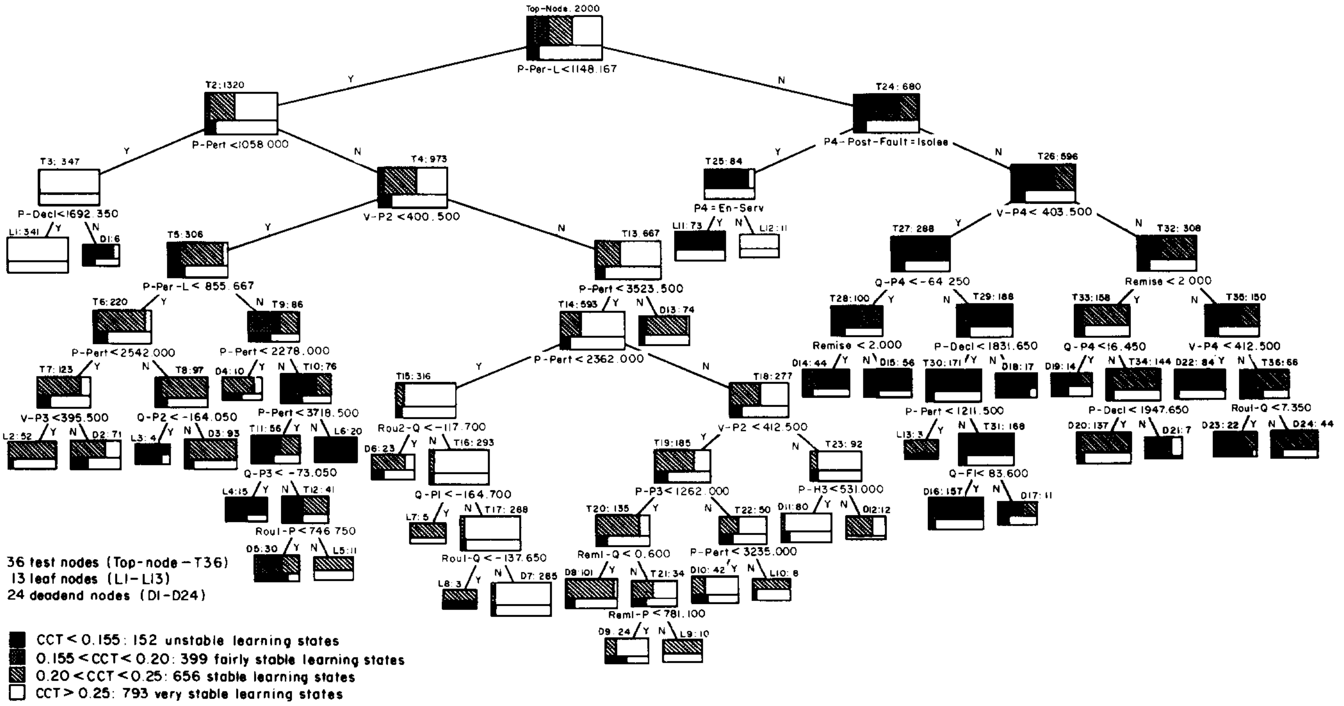


Figure 6. 61-machine system. Three-phase busbar fault. 4 classes. $N = 2000$, $M = 1000$, $\alpha = 1.0 \times 10^{-4}$, $P_e = 15.9\%$

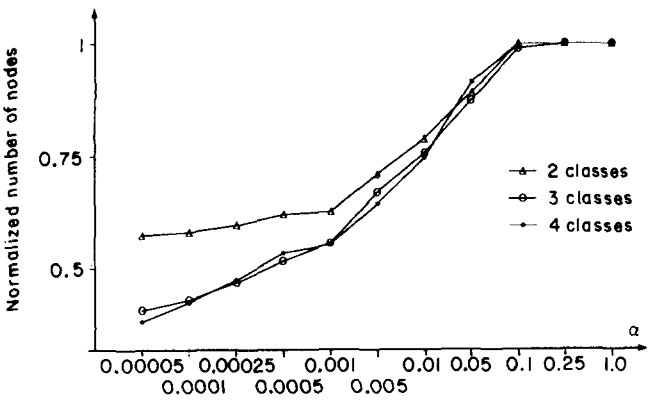


Figure 7. Influence of α on the normalized number of nodes (from Reference 4)

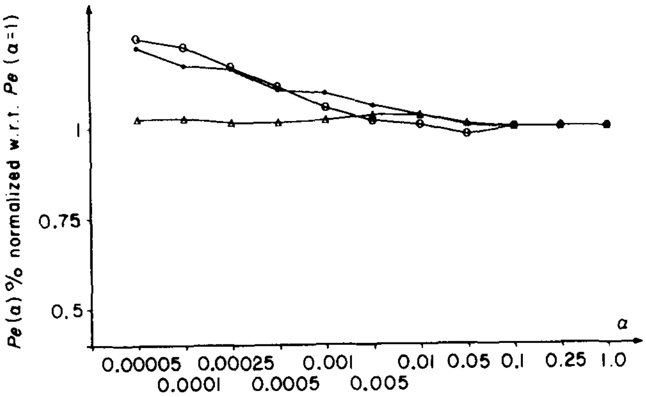


Figure 8. Influence of α on the normalized error estimate (from Reference 4)

Table 2. Effect of α on the tree size and error probability

Power system (Disturbance)	LS TS	No. of classes	Nb. nodes/P_e $\alpha=0.1$	Nb. nodes/P_e $\alpha=10^{-4}$
31-machine	500	2	13/ 1.9%	7/ 1.7%
(3-ph.sh.-circuit)	1500	4	55/ 5.8%	25/ 5.9%
61-machine	2000	2	63/ 1.5%	19/ 1.4%
(Busbar fault)	1000	4	341/16.0%	73/15.9%

be constructed on the basis of *all* available states, no test set being required to select the best tree at the pruning stage.

Influence of the learning set size

Figures 9 and 10 have been drawn on the basis of simulations performed on a 31-machine system. The trees are built for a 3-phase short-circuit with a 4-class classification. The figures indicate the effect of the learning set size N on the total number of tree nodes and on the error estimate. It is interesting to observe the slightly sublinear variation of the number of nodes with the learning set size; this confirms earlier analysis[16]. Observe also that the improvement of the trees error rate saturates slowly, for large values of N.

The above statistics are illustrated by Figures 11 and 12 obtained for two different values of N. The top parts of the two trees are very similar, although not identical.

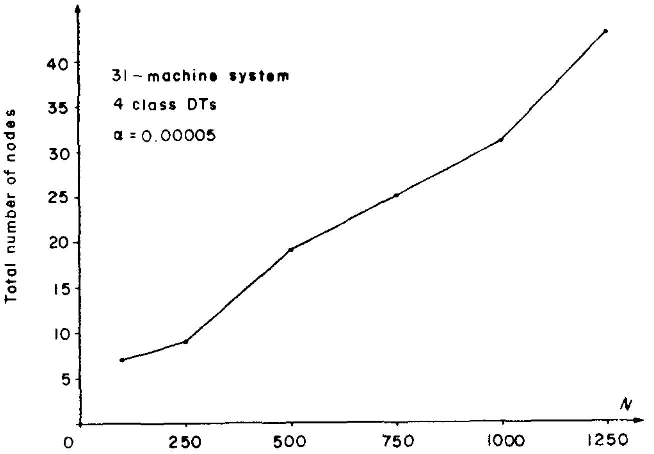


Figure 9. Influence of N on the total number of nodes (from Reference 4)

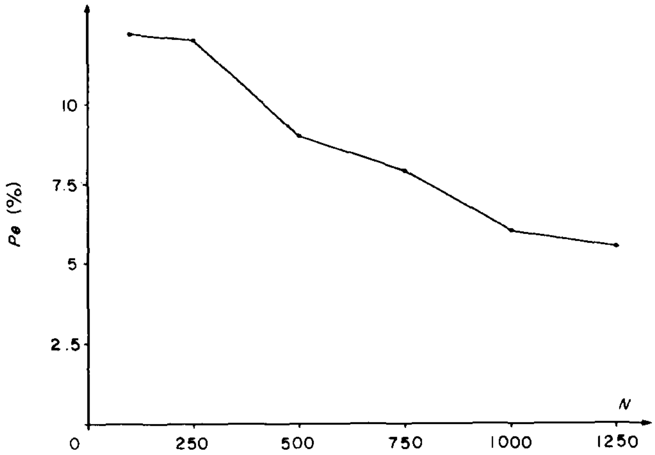
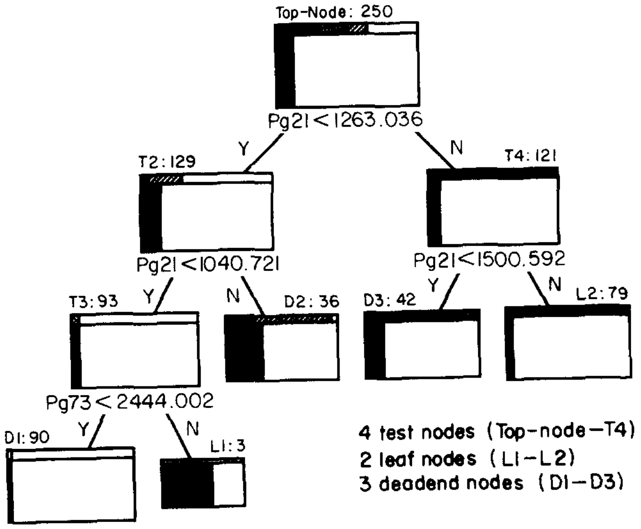


Figure 10. Influence of N on the error estimate (from Reference 4)



■ CCT < 0.27: 80 unstable learning states

■ 0.27 < CCT < 0.32: 47 fairly stable learning states

▨ 0.32 < CCT < 0.37: 34 stable learning states

□ 0.37 < CCT: 89 very stable learning states

Figure 11. 31-machine system. Three phase short-circuit at a generator bus (postfault = prefault). 4 classes; $N=250$, $M=1750$, $\alpha=5.0\times10^{-5}$, $P_e=12.1\%$ (from Reference 4)

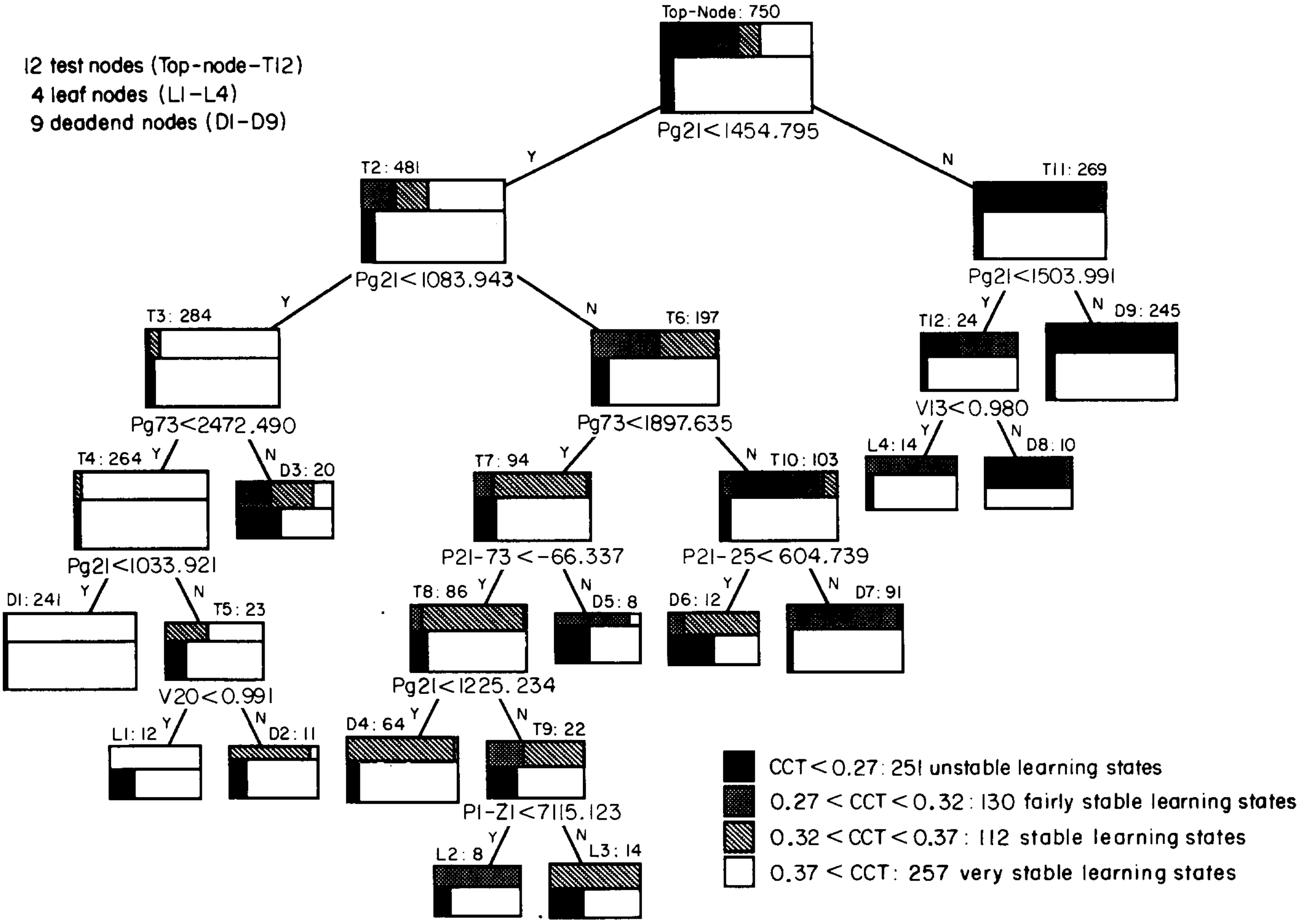


Figure 12. 31-machine system. Three-phase short-circuit at a generator bus (postfault=prefault). 4 classes. $N=750$, $M=1250$, $\alpha=5.0\times10^{-5}$, $P_e=7.9\%$ (from Reference 4)

The above observations hold valid for all the other tested power systems.

Effect of the number of classes

The simulations confirm the intuitive fact we mentioned earlier that the larger the number of classes, the more complex are the obtained trees. At the same time, the accuracy of the trees decreases[f], while the 'optimal' value of α slightly increases. Recall that using appropriate low α values allows a drastic reduction in tree complexity, which largely contributes to their interpretability, by suppressing their non-significant parts. For large number of classes, and for complex situations, when many parameters may affect the stability this feature becomes crucial.

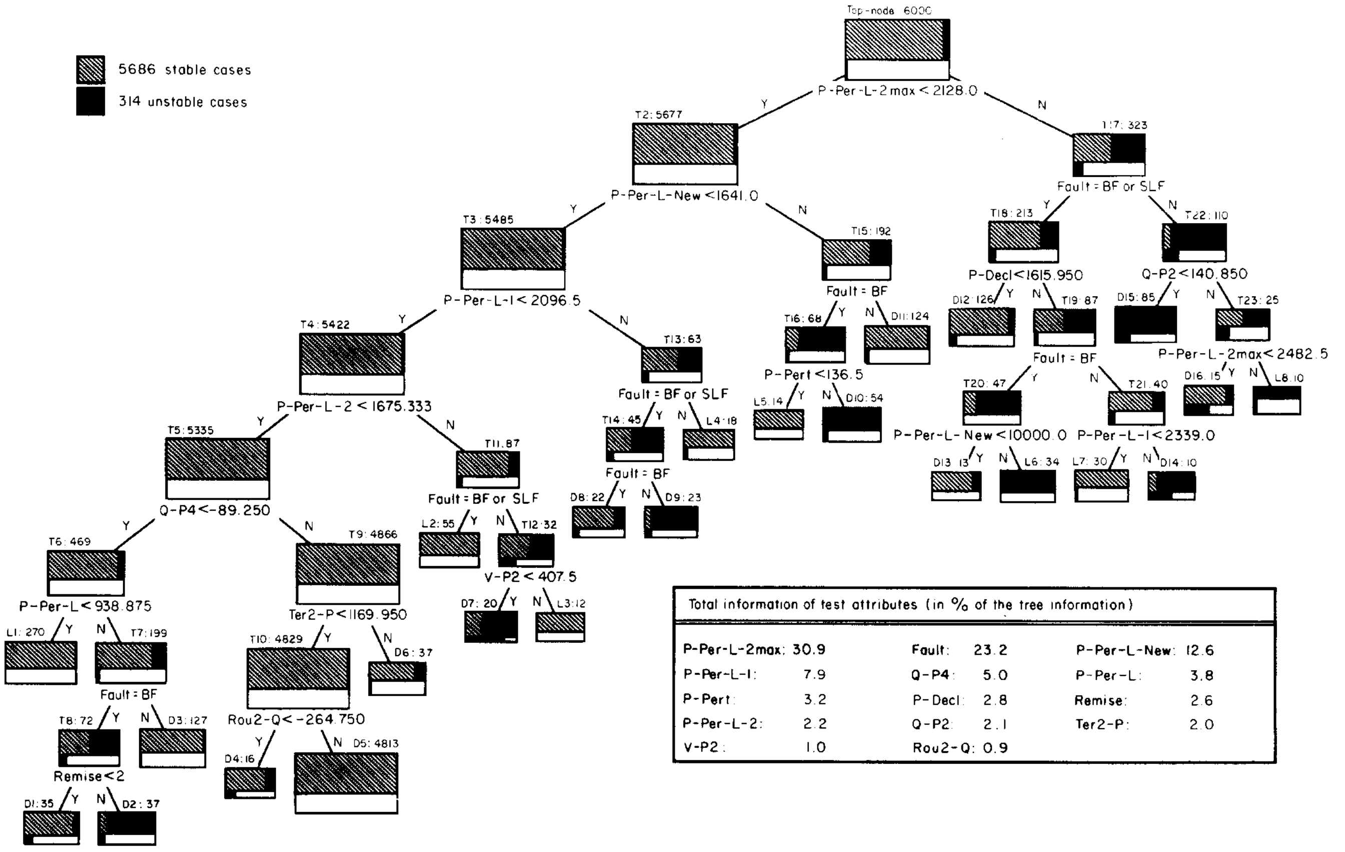
A multicontingency decision tree

Figure 13 represents a multicontingency tree obtained for the 61-machine system. Three different contingencies were used, located in the same substation: (i) the above mentioned busbar fault (denoted 'BF', in the tree), cleared after 155 ms; (ii) a single line fault (denoted 'SLF'), consisting of a short-circuit at one end of a line (close to the substation) cleared after 100 ms by tripping the line; (iii) a double line fault (denoted 'DLF'), consisting of a short-circuit on a double circuit line, cleared by tripping the two circuits after 100 ms.

The three contingencies together with the 3000 operating states of the data base yield a total number of 9000 stability cases: a random sample of 6000 are used as learning set, and the remaining 3000 are used to test the tree. Comparing the multicontingency tree with the corresponding single-contingency ones, we observe that the multicontingency tree has:

- a complexity of 47 nodes vs. 45, the total number of nodes of the three single contingency trees;
- an error rate of 1.6% vs. 1.7%, the mean error rate of the single contingency trees;
- 14 different test attributes (including the attribute 'Fault') vs. 18, the total number of different test attributes of the single contingency trees.

Thus, without loss of reliability, a multicontingency tree appears to be able to provide a more synthetic view of the stability relationship than a set of single contingency trees. Moreover, similarities among contingencies are identified and highlighted by the tree (e.g. the operating states corresponding to the DT node D5 are stable with respect to all three contingencies; states corresponding to node D15 are unstable only for the double line fault etc.). On the other hand, considering the information quantities of the test attributes, indicated in Figure 13, one can observe that the 'fault' attribute carries about 23% of the total information of the tree, although it appears generally at deep nodes. Thus the tree first extracts the information independent of the fault (e.g. carried by the 'P-Per-L ...' attributes, in Figure 13) and then only the more specific, fault dependent relations.



Total information of test attributes (in % of the tree information)					
P-Per-L-2max:	30.9	Fault:	23.2	P-Per-L-New:	12.6
P-Per-L-1:	7.9	Q-P4:	5.0	P-Per-L:	3.8
P-Pert:	3.2	P-Decl:	2.8	Remise:	2.6
P-Per-L-2:	2.2	Q-P2:	2.1	Ter2-P:	2.0
V-P2:	1.0	Rou2-Q:	0.9		

Figure 13. 61-machine system. Multicontingency tree. 3 contingencies: BF, SLF and DLF. $N = 6000$, $M = 3000$, $\alpha = 1.0 \times 10^{-4}$, $P_e = 1.6\%$

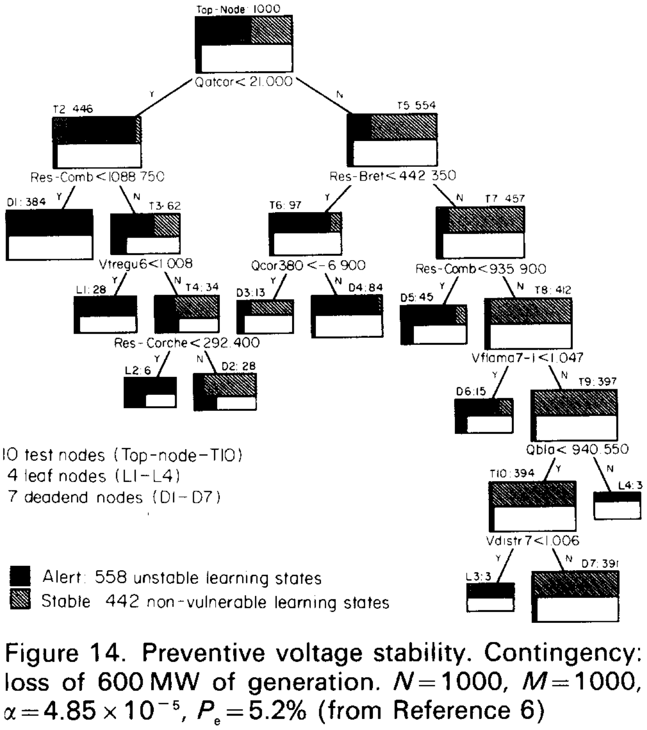


Figure 14. Preventive voltage stability. Contingency: loss of 600 MW of generation. $N = 1000$, $M = 1000$, $\alpha = 4.85 \times 10^{-5}$, $P_e = 5.2\%$ (from Reference 6)

III.3.2 *Preventive voltage stability assessment*

The trees built in this context concern the Britanny region of the EDF system which has in the past experienced voltage problems[31]. A data base composed of 2000 prefault operating states was generated, using a 320-bus, 55 generator, 614 branch model of EDF system, representative for the random modifications imposed in the Britanny region[6]. The above states were obtained by imposing random variations concerning (1) the active power generation schedule in a large enough region surrounding Britanny, (2) the local reactive resources (power plant configuration, voltage setpoints, HV and MV compensation, synchronous condenser), (3) the regional active and reactive load level, (4) single 400 kV 225 kV line or transformer outages.

Figure 14 gives a typical tree; the contingency considered is loss of 600 MW generation in the region. (Remember, its characteristics were fixed by the example of section II.4, Figure 3.) On the other hand, to illustrate the influence of parameter α, compare this tree with that of Figure 15, built under the same conditions (same contingency, list of candidate attributes and size of *LS*, apart from the α value ($\alpha = 4.85 \times 10^{-5}$, vs. $\alpha = 1$)). Observe that the upper parts of these trees are identical as is expected. Note also that the smaller tree of Figure 14 provides better reliability, in addition to interpretability, than the tree of Figure 15.

What happens if an important candidate attribute, initially selected as test attribute is afterwards discarded from the list of candidates? Before discussing reasons for such a 'masking', let us illustrate it in the particular case of almost equivalent candidate attributes, by means of the tree of Figure 16: this tree was grown under the same conditions as the tree of Figure 14, except that the top test attribute of this latter, Qatcor^u, was eliminated from the list of candidate attributes. This has led to the automatic selection at the top test of Figure 16 of the attribute Res-Comb^v. (In Figure 14 this latter attribute appears twice at two quite high level test nodes.) Observe that masking the top test attribute Qatcor hardly affects

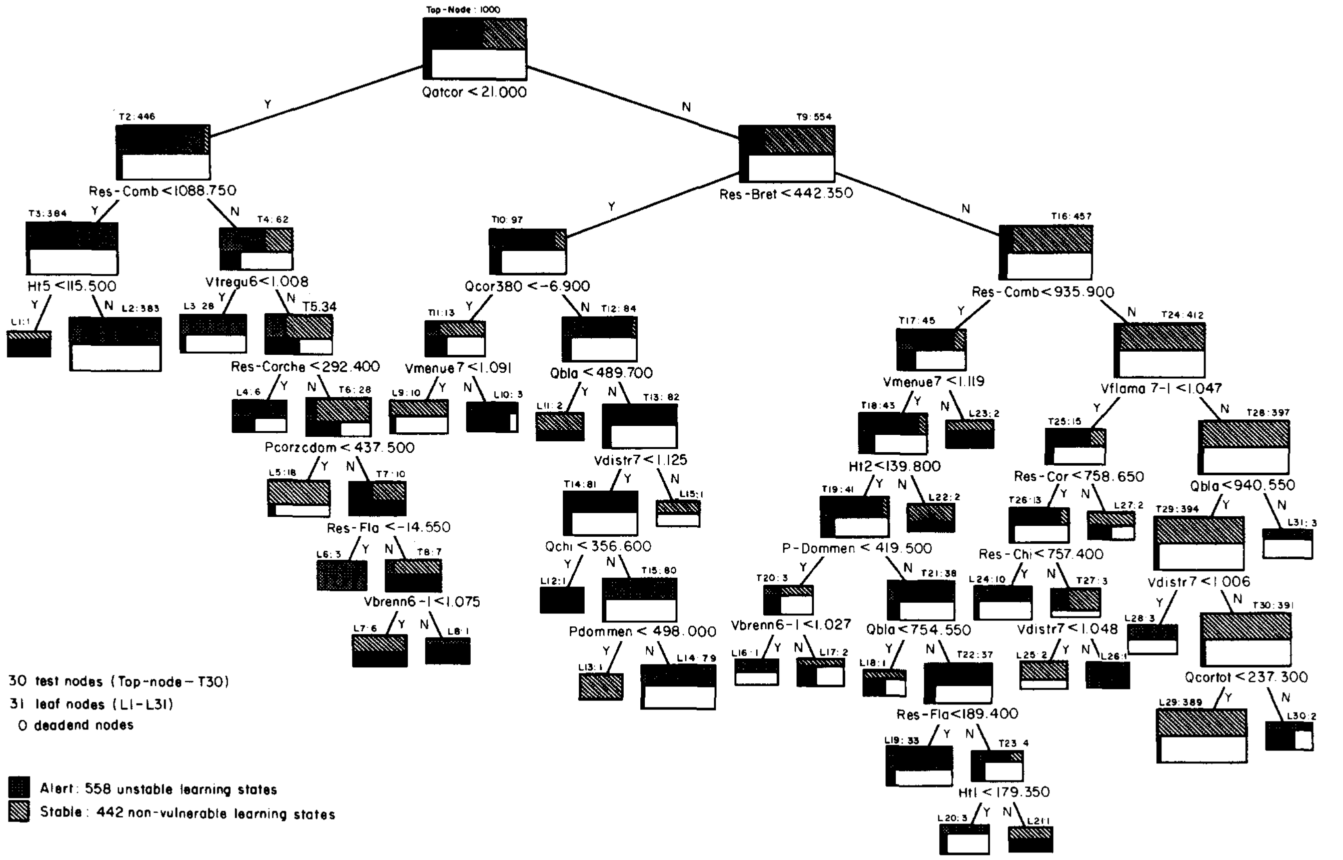


Figure 15. Preventive voltage stability. Contingency: loss of 600 MW of generation. $N=1000$, $M=1000$, $\alpha=1$, $P_e=7.2\%$ (from Reference 6)

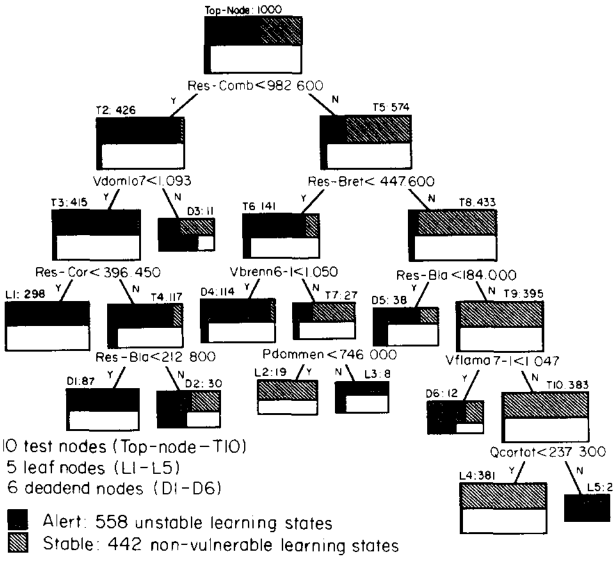


Figure 16. Preventive voltage stability. contingency: loss of 600 MW of generation. $N=1000$, $M=1000$, $\alpha=5.0\times10^{-5}$, $P_e=5.7\%$ (from Reference 6)

the overall tree accuracy. The explanation is given by the scores obtained by the test attributes at the root of the tree of Figures 14 and 16: Qatcor: 0.3856; Res-Comb: 0.3830.

Another interesting comparison concerns the information quantity provided by the higher (top) test nodes of each tree. Thus:

- for the tree of Figure 14: Qatcor: 382.1; Res-Comb: 154.6; Res-Bret: 134.3; total information quantity provided by the tree: 818.8;
- for the tree of Figure 16: Res-Comb: 378.1; Res-Bret: 188.0; Res-Bla: 99.1: total information quantity provided by the tree: 818.4.

Note that the total information quantity potentially contained in the tree is 990.3 bits.

The above masking suggests a flexible way of replacing some attributes by others (e.g. by more controllable ones). Another reason for masking attributes could be to uncover additional important ones.

Note, however, that masking the top test attribute often deteriorates the tree performances, apart from situations, as above, where a high redundancy allows one to replace it by another attribute of very similar score.

III.3.3 *Emergency voltage control*

The trees portrayed in Figures 17 and 18 have both been built for the purpose of emergency voltage control, following the procedure of section III.2.3. Figure 17 portrays an academic type example[7]: Figure 17a is the single-line diagram of the academic type system which, however, contains all realistic ingredients and system parameters; Figure 17b represents the tree built under the conditions briefly described hereafter. A set of 500 predisturbed states have been generated then subjected

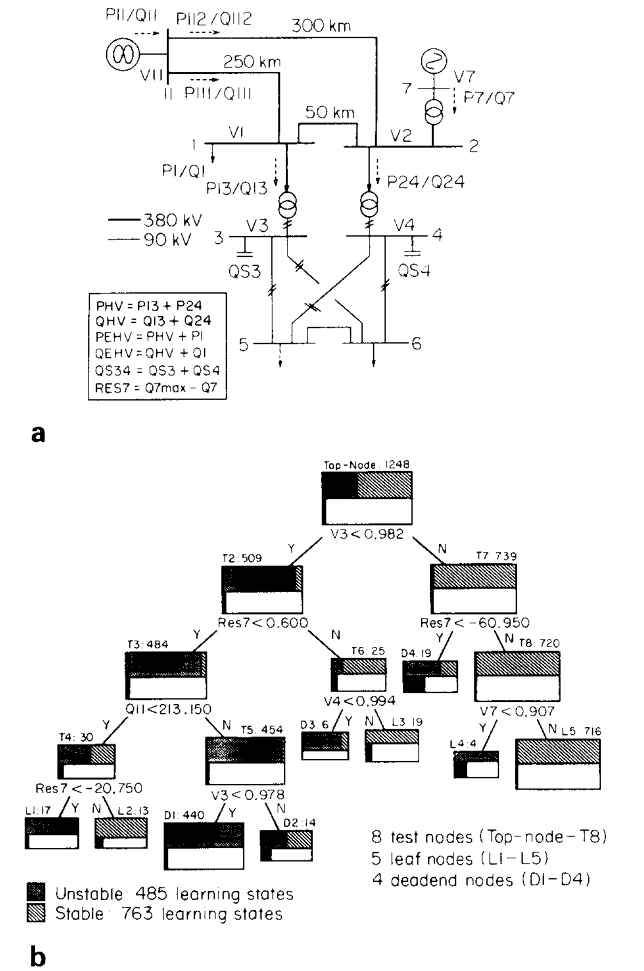


Figure 17. Emergency voltage control; 5 contingencies (from Reference 7). (a) One-line diagram of the academic system. (b) Decision tree. $N=1250$, $M=1250$, $\alpha=1.0\times10^{-4}$, $P_c=3.6\%$

to five contingencies, namely:

- tripping of the 400-kV line 11-1;
- tripping of the 400-kV line 11-2;
- loss of two units at bus 7;
- loss of three units at bus 7;
- loss of three units at bus 7 and simultaneous tripping of line 11-2.

The resulting 2500 JAD states have been assessed at 20 s after the disturbances inception. Of them, 1250 have been used as learning states, the other 1250 as test states. On the other hand, 28 candidate attributes have been proposed to the automatic tree building procedure; out of them, it has retained only five. Moreover, two of them, viz. voltage V3 (or V4) and reactive power reserve Res7 appear to carry the greatest deal of the tree information. From a physical viewpoint this result seems quite sensible: V3 quantifies how far the contingency has moved the system from its 'voltage stability' equilibrium point; Res7 evaluates the amount of available generation reserve to respond to the change in configuration and the subsequent on load tap changer dynamics (the latter will cause an increase in reactive demand and losses when pulling the MV and HV voltages back to a nominal value).

The tree portrayed in Figure 18 relates to the Britanny region of the EDF system[8,32]. The main difference between the corresponding two approaches essentially lies in the way of generating the sample of learning states: in the academic example, this generation conforms to the general pattern, seeking representativeness of the system behaviour; in the Britanny region on the other hand, a majority of 'borderline' states (close to the stability boundary) were generated on purpose by a deterministic approach[33]. This could explain the complexity of the tree of Figure 18. It also illustrates the robustness of the tree construction method: despite the stringent conditions imposed by the data base, it nevertheless yields a quite accurate tree.

III.4 *Discussion*

Structural parameters

The trees obtained so far within the various security contexts exhibit quite similar features.

- Curves of complexity vs. accuracy plotted for various αs show that the optimum α value lies around 10^{-4}. In this range, and for 2-class trees, the complexity amounts to about 10 to 40 nodes, involving generally less than 10 test attributes and yielding an accuracy of about 92 to over 99%.
- Complexity and accuracy increase rather monotonically with N, the number of learning states; for $N \sim 1250$ the curves, especially the accuracy one, begin to saturate, showing that a number between 2000 and 3000 for N should be quite sufficient.

Accuracy

We just mentioned that for 2-class trees the misclassification rate ranges in between 1 and 8%. For large number of classes, this error increases. To get a better insight of the real meaning of this global assessment, consider at first 'normal' errors, i.e. errors falling into the ±10% range around the threshold CCT. The decrease observed when the number of classes increases is apparent rather than real, in that the *gravity* of misclassification generally decreases. Indeed, this misclassification generally affects states of adjacent classes; hence, an erroneous diagnostic is less misleading for, say, 4-class than for 2-class trees. E.g., in transient stability, declaring stable a state which is actually unstable is less dangerous in a 4-class tree (classes labelled very unstable; unstable; stable; very stable), than in a 2-class tree (classes labelled unstable; stable). One could argue that the attribute space may anyhow provide a much more refined diagnostic than the crude 'stable–unstable' one provided by the tree itself; in this space one could even recover the actual critical clearing time (within a tolerance), by defining appropriate distances[9,10,17]. This is to the advantage of 2-class trees which provide more tractable attribute spaces; but anyhow the appropriate attribute space representation is still an open question.

Let us now concentrate on the 'abnormal' errors, or 'outliers', and more specifically on the dangerous ones. The existence of such outliers corresponds to an incomplete data base, where all salient cases have not been covered well enough. This difficulty intrinsically exists in all pattern recognition type of security approaches, especially if applied to well designed power systems since there is an imbalance between the information contained in a 'normally' generated data base (much more stable than unstable cases) and the information required to treat 'interesting' cases, i.e. unstable ones.

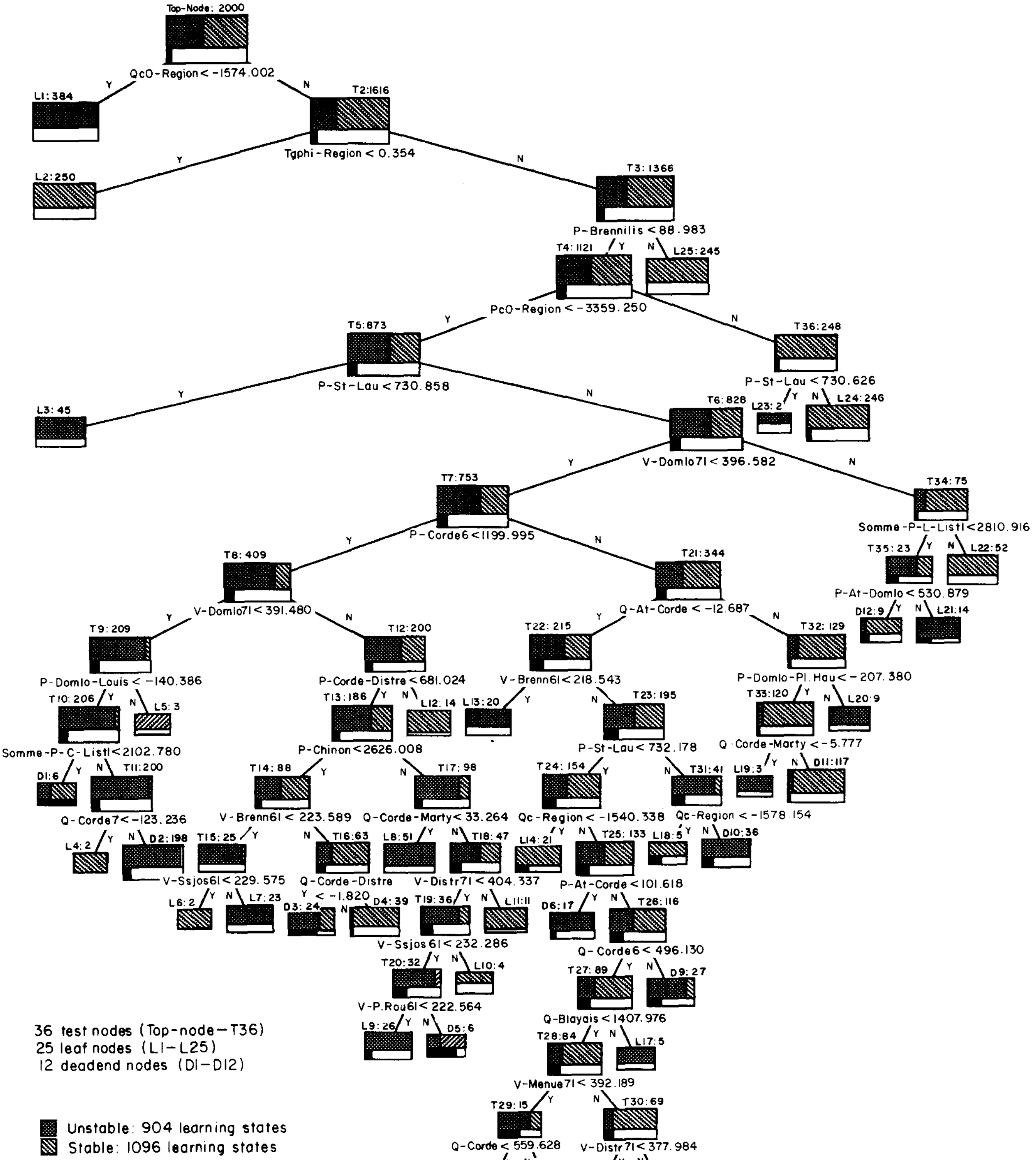


Figure 18. Emergency voltage control. French system. $N = 2000$, $M = 971$, $\alpha = 1.0 \times 10^{-3}$, $P_e = 6.2\%$ (from Reference 8)

One way to circumvent this difficulty is to call upon the expertise of system designers and operators in order to cover abnormal cases as well. During our investigations on transient stability, they helped efficiently to take properly into account complex topological changes which were initially overlooked and had created outliers. Another in-depth investigation has recently been carried out in order to improve the trees' ability to handle dangerous outliers. Among the various explored directions we mention the following.

- Develop 'pragmatic quality'^{w} measures to evaluate the tree proportion of false alarms and of dangerously optimistic diagnostics.
- Investigate and select methods to enhance the pragmatic quality both at the DT building stage and at the

DT interpretation stage, in order to avoid that actually quite unstable states are classified stable, while maintaining the number of false alarms as low as possible.

- Investigate the possibility of iteratively enhancing a tree, by incrementally developing its subtrees by:
  - tuning the DT to remove its dangerous (overly optimistic) diagnostics;
  - identifying its 'weak' points, i.e. its terminal nodes corresponding to high proportions of false alarms;
  - generating a learning subset for each weak node and building the corresponding subtree;
  - modifying the initial tree by replacing each of its weak nodes by the corresponding subtree.

  Combination of (some of) the above techniques allowed one to reduce the dangerous diagnostics quite efficiently, without increasing the number of false alarms too much[5].

Number of classes

- The advantage of 2-class trees is their simplicity, which in turn allows easier interpretability and provides more tractable attribute spaces.
- The advantage of having larger number of classes is the possibility for more refined assessment (for example by using a larger number of stability classes defined within narrower ranges), and/or richer exploration of multifacet phenomena (for example, an operating state found to be otherwise unstable, could be more finely assessed by distinguishing cases where appropriate actions can bring it back to security, from cases where these actions have insufficient influence).
- Once again, there is no universal optimal solution: various applications (or operation strategies) may call for different, complementary types of trees.

Candidate attributes

- Various types of candidate attributes may be thought of. In general, the more complex and sophisticated the candidate attributes, the less complex the resulting tree will be but also the less tractable the information it contains. (An extreme typical example would be to use in the DTTS method the critical clearing time itself as an attribute: this would provide a sole two-level 2-class tree, consisting of the top node and its direct successors.)

  Stated otherwise, the choice of candidate attributes may depend on the type of needs one wants a tree to meet. For example, attributes appropriate to interpret or describe a problem may be different from those appropriate to yield straightforward control actions.
- For a given list of candidate attributes, the selection of the test attributes relies on their 'scores', i.e. their ability to provide the purest possible successor nodes. Masking an attribute which has a large score, is often likely to uncover other good attributes which otherwise could never appear; the resulting tree could however be somewhat less efficient. This offers interesting tradeoffs for exploring various mechanisms of the physical phenomena, and various parameters of the system and their relationships. In short, it offers the possibility to build several trees, each relating to a particular aspect.
- Another piece of information is provided by the information quantity of the test attributes of a tree. Considering combinations of attributes carrying a good deal of information may also yield interesting trees.

Benefits of attribute space representation

A variety of remarkable outcomes may be thought of. In the field of analysis, the attribute space provides the possibility of recovering part of the information lost in the tree, especially for those states which are directed to deadends: their location in the attribute space may furnish much clearer and more accurate information. It may even be possible to recover or uncover the CCT value of all cases treated by a tree; this information, initially contained in the learning set, is subsequently 'degraded', since the tree merely assesses cases within a range of CCTs (e.g. 'stable' or 'unstable'). Using appropriate distances in the attribute space may again furnish CCTs. In the fields of sensitivity analysis and control, the information is even more necessary and valuable to system operators. A first exploration of some such potentials of the attribute space was reported in References 9, 10 and 17. Nevertheless, further investigations are needed to handle suitable multidimensional spaces. Moreover, appropriate ways of handling qualitative attributes in the distance formulation remain to be devised. For instance, to model the effect of topology, different distance parameter values could be used for each of the different topology classes identified by a tree. Admittedly, this may not be an easy task, but it is full of promise for interesting practical applications.

Computing requirements

The following tasks may be identified:

- the off-line bulk of simulations needed to generate the data base of preclassified states; depending upon the type of security considered, it amounts to days of CPU time; for example, the generation of the 3000 learning states for the French 61-machine system requires about 4 days CPU on a 28-MIPS SUN computer;
- the off-line building of DTs; with the same as above data, the computing time required to build a tree on the basis of 2000 states and 57 attributes is about 5 min;
- the on-line use of a DT is truly straightforward and requires quite reduced memory to store it; with the same as the data above, it requires about 0.1 ms.

IV. Overview of the tree-wise security approach

IV.1 *Properties of the tree method*

IV.1.1 *What category does it fall into?*

By its principle, the DT methodology belongs to the broad class of pattern recognition approaches ('learning from examples'). It proceeds, however, in a very particular way, where the features selection and the states classification are obtained in one shot, yielding the DT: the features are the test attributes selected at the test nodes, and the classifier is the tree itself.

By its formulation, the DT methodology belongs to artificial intelligence; together with information theory and statistical tests, they succeed in extracting information, then compressing and organizing it in a tree fashion.

By its sources of information, the tree methodology falls into the general category of expert systems. These sources are: quantitative expertise resulting from the use of 'system theory' methods and related computer programs on the one hand, and human expertise used to elaborate appropriate scenarios on the other. Their combination aims at providing a representative sample of preclassified instances necessary to the tree building. Also necessary to achieve this, is the selection of appropriate candidate attributes; operation and planning expertise is again very valuable.

IV.1.2 *Which kinds of structures does it provide?*

The methodology yields a dual structure: the tree itself and its attribute space. The tree provides a hierarchical account of the phenomena: the more relevant the features (attributes) to these phenomena, the closer, in general, they appear to the tree top. Stated otherwise, the closer to the tree top, the larger the amount of information to be processed. Such a hierarchical, complex construction is likely naturally to suit the description of complex phenomena – and those involved in power system security are complex, indeed. It provides both a powerful conditional class probability model[12,16] which may be exploited in many different ways[34], and also a simple deterministic classifier, easier to appraise and to use efficiently by human operators.

The attribute space is another way of describing the same phenomena; yet, it offers different types of possibilities, complementary to those of the trees.

In the pattern recognition terminology, the tree and its attribute space are two different, complementary although equivalent, classifiers. This dual classification is able to broaden significantly the field of applications.

IV.1.3 *What kind of security information is available?*

One could argue that the DT 'spoils' part of the learning information used to construct it. Thus, for example, in the case of transient stability, the tree allows one merely to classify a state within a range of CCT values, whereas each state in the learning set is known with its precise CCT value. However, this refined information may indeed be restored in the attribute space (within the accuracy limits of the constructed DT) as was shown in References 9 and 17, where distances to the stability boundaries in the attribute space were also defined.

In addition to refined diagnostic, the above distances in the attribute space also provide means to sensitivity analysis and control, and this may be crucial in real-time operation. Now, in order to yield a 'tractable' attribute space, a DT should be built on the basis of 'simple enough' attributes. This leads to the following question.

IV.1.4 *What types of attributes?*

Various types of candidate attributes may be considered. Recall that this yields a tradeoff between simple directly controllable and/or interpretable attributes and more complex ones, less practicable, yet capable of providing more efficient trees. This kind of flexibility may be properly exploited. We believe that different uses call for different choices of types of attributes, depending in particular on the strategies sought. The great *flexibility* of the overall methodology with respect to the diversity of practical constraints is a particularly interesting feature.

IV.1.5 *What is a 'good' data base, i.e. a 'good' sample of learning states?*

Generating a representative sample of cases is a necessary condition to the success of the tree methodology, as of any pattern recognition approach.

Yet, rules guaranteeing the appropriateness of a data base could not exist especially for such complex problems as those involved in power systems security. The following general guidelines may however be of help:

- to the maximum extent possible cover all foreseen plausible situations;
- however, avoid bias that would arise by considering too many odd situations;
- use a large number of simulations or of real life instances to test trees built on an initial learning set; if necessary adjust and augment this latter; this greatly helps to gain ones own experience;
- call upon the expertise of human operators.

However, our experience shows that generating a 'good' data base is not so difficult after all: thanks to its robustness, as is redefined below, the tree itself is able to detect anomalies and suggest how to correct them. A cut–and–try approach at the beginning of the investigations helps the designer get his own expertise quite soon.

IV.1.6 *When and how to refresh the data base?*

The answer to this question may greatly depend on the specifics of the particular security problem at hand, as well as on the operating/planning strategies of the considered power system utility. Since it would be difficult to provide a single definite answer, we will merely outline a plausible strategy for maintaining a set of decision trees up to date, when some major changes arise.

A distinction should be made between changes influencing the stability classification of scenarios (e.g. control law changes, revision of security criteria ...) and changes leading to new system operating states, not taken into account when building the initial data base (e.g. addition of new lines, power plants ...).

In the first situation, a simple adaptation would consist of reclassifying the operating states of the existing data base according to the new assumptions, and rebuilding the trees, at least those parts which turn out to be less reliable.

In the second case, we first note that the existing trees may still be used for situations where the new equipment is not in operation. On the other hand, a new data base should be easily obtained from the original one, to cover the situations with the new equipment in operation, e.g. by updating the base case file and the parameters used for the probability distributions, and making a new random sampling and classification of the data base. A set of updated trees may be generated therefrom. In a second step, one might prefer to merge the original and new data bases in order to build compound trees covering both kinds of situations. Admittedly, the overall data base should be kept to a manageable size and the information be excluded as soon as it becomes obsolete.

IV.2 *Features of the devised trees*

These features result from the properties discussed so far. Below we gather the more essential of them and identify three main aspects: flexibility and effectiveness from the viewpoint of the user; robustness from the viewpoint of both designer and user of the trees.

IV.2.1 *Flexibility*

A good deal of the method's attractiveness lies in the fact that the required information, both qualitative and quantitative, may be acquired off-line and, hence, embed any kind of desired refinements. Moreover, unlike other domain applications, this information is obtained by means of 'system theory' methods and relating numerical programs; there is therefore no difficulty in getting as much information as desired, provided that off-line CPU time is available.

Another salient feature is its great flexibility, with respect to number of classes, value of α fixing the degree of tree's development, size of learning sets, possibility of refreshing and/or augmenting it, types of proposed candidate attributes and related applications, possibility of using the sole trees, or their attribute space, or both. A corollary of this great flexibility is transparency and interpretability of the phenomena.

In respect of practical applications, the tree methodology is able to comply with all three typical domains in security studies, viz. planning, operation planning and real time operation, depending on the type of security assessment sought. Of them, the real time imposes particularly stringent requirements not yet satisfactorily met by existing methods, if at all. Admittedly, this application domain is particularly well suited to the tree method.

IV.2.2 *Effectiveness*

The effective use of DTs implies that they are able to provide accurate security assessment and limited computer requirements. Below we comment further on these two aspects, already discussed in section III.4.

IV.2.3 *Accuracy*

The trees portrayed in Figures 5 to 17 reflect quite well the global accuracy assessed in a very large number of trees built so far within the three security contexts. The fact that the accurcy, ranging from 92 to over 99% in the 2-class trees, deteriorates for trees with larger number of classes is more apparent than real as discussed previously. On the other hand, a closer examination of the misclassification errors detected in 2-class trees shows that they mainly occur at deadends. This is also corroborated by the observation that these errors are essentially confined around the stability boundary and are hardly misleading.

Generally speaking, as compared with other pattern recognition methods, in particular to artificial neural networks (ANN), the decision tree approach produces at least as reliable classifiers[35,36]. In the practical context of power system security, it presents moreover the following two very important advantages.

- As opposed to ANNs, where the structure and complexity of the classifier must be chosen *a priori*, i.e. 'by hand', the tree method determines the optimal structure and complexity for a given problem, during the tree building and on the objective basis of the learning set. It is thereby able to simplify the resulting structure to a very large extent (remember, in general less than 10% of the candidate attributes are actually selected), without overlooking significant effects.
- On the other hand, the explicit and hierarchical tree structure, together with its simplicity, enable its use for the purpose of control applications, which require to invert the attribute vs. security relationship. This intrinsically combinatorial problem could admittedly be difficult to tackle with 'black box' classifiers such as the ANN or the nearest-neighbour rule, to mention only the most popular ones. Bearing in mind the practical analysis and control requirements of most power system security problems, the decision tree approach thus appears to be particularly well suited for this kind of application.

More fundamentally, in the context of power system security problems, for a pattern recognition technique to be accepted it should be able to integrate smoothly in the existing operation planning environment, where the off-line security studies would actually be carried out. Thus, to be really effective, the method should produce its information in an interpretable fashion, in order to allow operation planning engineers to understand and validate it, and eventually reformulate it for its integration into the operation environment. From this point of view, the simplicity and interpretability of the decision trees make them particularly attractive to utility engineers.

IV.2.4 *Computing requirements*

Both memory and computing time requirements are extremely parsimonious. The storage of a large number of trees, their possible coordination, and their interaction with other functions of a complete software package would be quite straightforward, within a given security strategy. This leaves open possibilities of the design of such appropriate strategies. The off-line computing burden, i.e. essentially the time required to generate the data base – accounting for the fact that a large number of contingencies must in general be considered to obtain a sufficiently rich set of decision trees, for on-line analysis and control purposes – is certainly in the scope of currently available computer technology.

IV.2.5 *Robustness*

With respect to the data base, the trees are very robust indeed; they consistently uncover and properly describe the problem at hand, provided that 'sufficiently large' learning sets are used. Experience with the DTTS method shows that a reasonably representative learning set for a power system of the French system's size contains about 2000 to 3000 learning states. The trees are even able to detect anomalies in the data base, relating to the plausibility of either the scenarios used to generate the operating points or the contingencies. This is quite normal after all and follows the general pattern according to which anomalies in the behaviour of a physical system are often reflected in the method and its numerical treatment.

With respect to the list of candidate attributes the trees are also quite robust, i.e. able to handle correctly a security problem, provided that this list is comprehensive enough. To avoid missing salient attributes, it is advisable to use large lists, at least at the first stage of the tree design.

Finally, the trees are very robust too with respect to the choice of their optimal parameters. Figure 3 suggests that the range of $q(\alpha)$ parameter providing a good 'simplicity vs. reliability' compromise is quite large, indeed.

V. Conclusion

The objective of this paper was to give a global view of the tree methodology, shaped to handle power system security issues, and to highlight its essential facets.

Among the salient features are effectiveness and flexibility. Effectiveness is combining real-time possibilities, accuracy and interpretability of the results. Flexibility in respect of domain specifics, power system size and configurations, type of system parameters, modelling sophistication and 'system theory' methods, types of diagnostics provided and ways of their formulation, to mention only a few. Robustness is another important feature: the trees show to adjust quite nicely to learning data bases and further to suggest proper changes to fit better the problem at hand. This makes the representativeness of the data bases a much easier issue than one could anticipate.

Overall, it was shown that the method is by now mature enough to handle analysis aspects of security. Further research work is now needed, not so much in purely theoretical aspects of the tree methodology *per se*, but rather in the application domains of power system security. Such prominent progress would concern the tree attribute space (representations, definitions of distances, etc.) and the design of appropriate strategies for their use. This latter aspect points out once again the necessity of collaboration between designers of the method and its users, i.e. engineers in charge of power system operation. This will contribute to furnish original practical techniques missing today. The achievements accomplished so far are only a small sample of the potentials of this attractive, full of promise approach.

VI. References

1 **Quinlan, J R** 'Induction of decision trees', *Machine Learning* Vol 1 (1986) pp81–106

2 **Wehenkel, L, Van Cutsem, Th and Ribbens-Pavella, M** 'Artificial intelligence applied to on-line transient stability assessment of electric power systems' *Proc. 25th IEEE Conf. Decision and Control*, Athens, Greece (1986) pp649–650

3 **Wehenkel, L, Van Cutsem, Th and Ribbens-Pavella, M** 'An artificial intelligence framework for on-line transient stability assessment of electric power systems' *IEEE Trans. Power Systems* Vol PWRS-4 (1989) pp789–800

4 **Wehenkel, L and Pavella, M** 'Decision trees and transient stability of electric power systems' *Automatica* Vol 27 No 1 (1991) pp115–134

5 **Wehenkel, L, Pavella, M, Euxibie, E and Heilbronn, B** 'Decision tree based transient stability method – A case study' (to be presented at IEEE Winter Power Meeting (1993))

6 **Wehenkel, L, Van Cutsem, TH, Gilliard, M, Pavella, M, Heilbronn, B and Goubin, M** 'Decision trees for preventive voltage stability assessment' *Proc. Int. Workshop on Bulk Power System Voltage Phenomena – Voltage Stability and Security* Deep Creek Lake, McHenry, Maryland (1991) pp217–228

7 **Van Cutsem, Th, Wehenkel, L, Pavella, M, Heilbronn, B and Goubin, M** 'Decision trees for detecting emergency voltage conditions' *Proc. Int. Workshop on Bulk Power System Voltage Phenomena – Voltage Stability and Security* Deep Creek Lake, McHenry, Maryland (1991) pp229–240

8 **Wehenkel, L** *'Etude de la Stabilité du Plan de Tension au Niveau d'une Région. Exploitation des Ensembles d'Apprentissage Fournis par le LAIH de Valenciennes'* Internal Report, University of Liège (in French) (1991)

9 **Wehenkel, L, Van Cutsem, Th and Ribbens-Pavella, M** 'Decision trees applied to on-line transient stability assessment of electric power systems' *Procs. IEEE Int. Symp. on Circuits and Systems* Vol 2 Helsinki, Finland (1988) pp1887–1890

10 **Wehenkel, L** 'Artificial intelligence methods for on-line transient stability assessment of electric power systems' *Proc. Symp. on Expert Systems Application to Power Systems* Stockholm-Helsinki (1988) pp5.1–5.8

11 **Morgan, J N and Sonquist, J A** 'Problems in the analysis of survey data, and a proposal' *J. Amer. Statist. Assoc.* No 58 (1963) pp415–434

12 **Breiman, L, Friedman, J H, Olshen, R A and Stone, C J** *Classification and Regression Trees,* California: Wadsworth International (1984)

13 **Kononenko, I, Bratko, I and Roskar, E** *'Experiments in Automatic Learning of Medical Diagnosis Rules'*, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia, 1984

14 **Quinlan, J R** 'Learning efficient classification procedures and their application to chess endgames', in *Machine Learning: An Artificial Intelligence Approach,* **R S Michalski, J G Carbonell and T M Mitchel** (Editors) Springer, Berlin (1984) pp463–482

15 **Wehenkel, L, Van Cutsem, Th and Ribbens-Pavella, M** 'Artificial intelligence applied to on-line transient stability assessment of electric power systems' *Proc. 10th IFAC World Congress,* Munich, F.R.G. (1987) pp308–313

16 **Wehenkel, L** 'A probabilistic framework for the induction of decision trees' (Submitted for publication, 1991)

17 **Wehenkel, L** Une approche de l'intelligence artificielle appliquée à l'evaluation de la stabilité transitoire des réseaux electriques PhD Thesis (in French), University of Liège, Belgium (1990)

18 **Wehenkel, L, Van Cutsem, Th and Ribbens-Pavella, M** 'Inductive inference applied to on-line transient stability assessment of electric power systems' *Automatica* Vol 25 (1989) pp 445–451

19 **Kvålseth, T O** 'Entropy and correlation: some comments' *IEEE Trans. Syst. Man. Cybern.* Vol SMC-17 (1987) pp517–519

20 **Toussaint, G T** 'Bibliography on estimation of misclassification' *IEEE Trans. Inf. Theory* Vol. IT-120 (1974) pp472–479

21 **Zhou, X J and Dillon, T S** 'Multi-branching decision trees for induction', *Proc. 11th Int. Conf. on Expert Systems and their Applications* Vol 1 Avignon, France (1991) pp191–203

22 **Liu, C C, Wang, Shing-Ming, Wong, L, Marathe, H Y and Lauby, M G** 'A self learning expert system for voltage control of power systems' *Proc. 2nd Symp. on Expert Systems Application to Power Systems,* Seattle, Wa (1989) pp462–468

23 **Liu, C C and Wang, Shing-Ming** 'Development of expert systems and their learning capacility for power system applications' *Academic Press Series on Advances in Control and Dynamic Systems* (1991)

24 **Mingers, J** 'An empirical comparison of selection measures for decision-tree induction' *Machine Learning* Vol 3 (1989) pp319–342

25 **Mingers, J** 'An empirical comparison of pruning methods for decision tree induction' *Machine Learning* Vol 4 (1989) pp227–243

26 **Wehenkel, L** 'An information quality based decision tree pruning method' *Proc. of the 1992 IPMU Congress on Information Processing and Management of Uncertainty in Knowledge Based Systems* Palma de Mallorca, Spain (July 6–10 1992)

27 **Bergen, A R** *Power System Analysis* Prentice Hall (1986)

28 **Ribbens-Pavella, M and Evans, F J** 'Direct methods for studying dynamics of large scale electric power systems – A survey *Automatica* Vol 21 No 1 (1985) pp1–21

29 **Van Cutsem, T** 'A method to compute reactive power margins with respect to voltage collapse' *IEEE Trans. on Power Syst.* Vol 6 No 2 (1991) pp145–156

30 **Lemaître, C, Paul, J-P, Tesseron, J-M, Harmand, Y and Zhao, Y S** 'An indicator of the risk of voltage profile instability for real-time control applications' *IEEE Trans. on Power Syst.* Vol 5 No 1 (1990) pp154–161

31 IEEE System Dynamic Performance Subcommittee, 'Voltage stability of power systems: concepts, analytical tools, and industry experience' *Report 90TH0358-2-PWR* (1990)

32 **Goubin, M** 'Cadre d'une étude pour évaluer la possibilité d'utiliser des arbres de décisions pour la détection des états critiques en tension dans une région' Internal document – *EDF, HR-46/833* (1989)

33 **Zhao, Y S** *'Conception d'un Système Expert Destiné à la Caractérisation des Etats en Tension des Réseaux Electriques'* Final report of contract EDF/LAIH No. R46L08/1E7184, (1990)

34 **Pearl, J** *Probabilistic Reasoning in Intelligent Systems – Networks of Plausible Inference* Morgan-Kaufman, Ca (1988)

35 **Weiss, S M and Kapouleas, I** 'An empirical comparison of pattern recognition, neural nets, and machine learning classi-

fication methods', *Proc. of 11th Int. Joint Conf. on Artificial Intelligence,* Detroit, MI (1989) pp 781–787

36 **Atlas, L, Cole, R C, Muthusamy, Y, Lippman, A, Connor, J, Park, D, El-Sharkawi, M and Marks II, R J,** 'A performance comparison of trained multilayer perceptrons and trained classification trees' *Proc. of IEEE* Vol 78 No 10 (1990) pp 1614–1619

37 **Zhou, X J and Dillon, T S** 'A statistical-heuristic feature selection criterion for decision tree induction' *IEEE Trans. Pattern Analysis and Machine Intelligence* Vol PAMI-13 (1991) pp 834–841

38 **Zhou, X J and Dillon, T S** 'Learning multi-branching decision trees in noisy domains', *Proc. TC-7 IFIP Int. Conf. Communications, Automation and Information Systems* Rome, Italy (1990) pp 205–212

39 **Goodman, L A and Kruskal, W H** 'Measures of association for cross-classifications', *J. Amer. Stat. Assoc.* Vol 49 (1954) pp 732–764

40 **Daróczy, Z** 'Generalized information functions', *Information and Control* Vol 16 (1970) pp 36–51

41 **Devijver, P A** 'Entropie quadratique et reconnaissance de formes' *NATO ASI Series, Computer Oriented Learning Processes,* **J C Simon** (Editor) Nordhoff, Leyden (1976) pp 257–278

Appendix – Useful expressions

Note: A sample of useful expressions are collected below. Although, for the sake of simplicity, they are given in the context of two-class classifications and binary trees, they are easily generalized to an arbitrary number of classes and test node successors. Their mathematical derivation may be found in one of the References 4, 16, 17.

A.1 *Gain of information*

Consider a tree node and let $\boldsymbol{S}$ be the subset of learning states at it. Moreover denote by $p(+/\boldsymbol{S})$ (resp. $p(-/\boldsymbol{S})$) the probability of a state of $\boldsymbol{S}$ to belong to the class $+$ (resp. $-$).[x]

According to the information theory, the *information gain* (or entropy decrease) in $\boldsymbol{S}$ provided by the test T expressed by equation (6) may be quantified by

$$I_C^T(\boldsymbol{S}) \triangleq H_C(\boldsymbol{S}) - H_{C/T}(\boldsymbol{S}) \qquad \text{(A.1)}$$

Indeed, in equation (A.1), $H_C(\boldsymbol{S})$ is the *prior classification entropy* of $\boldsymbol{S}$, measuring the impurity of $\boldsymbol{S}$; $H_{C/T}(\boldsymbol{S})$ is the *mean posterior classification entropy* of $\boldsymbol{S}$ given the outcome of T, measuring the residual impurity should $\boldsymbol{S}$ be split into $\boldsymbol{S}_{\text{YES}}$ and $\boldsymbol{S}_{\text{NO}}$.

$H_C(\boldsymbol{S})$ is expressed by

$$H_C(\boldsymbol{S}) \triangleq -[p(+/\boldsymbol{S})\log_2 p(+/\boldsymbol{S}) + p(-/\boldsymbol{S})\log_2 p(-/\boldsymbol{S})] \qquad \text{(A.2)}$$

Similarly, $H_{C/T}(\boldsymbol{S})$ is expressed by

$$H_{C/T}(\boldsymbol{S}) \triangleq p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S})H_C(\boldsymbol{S}_{\text{YES}}) + p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S})H_C(\boldsymbol{S}_{\text{NO}}) \qquad \text{(A.3)}$$

$H_C(\boldsymbol{S})$ ranges in between 0 and 1: $H_C(\boldsymbol{S}) = 0$ corresponds to a perfectly pure subset (since then $p(+/\boldsymbol{S}) = 1$ or 0 and correspondingly $p(-\boldsymbol{S}) = 0$ or 1), while $H_C(\boldsymbol{S}) = 1$ corresponds to a perfectly impure subset (since then $p(+/\boldsymbol{S}) = p(-/\boldsymbol{S}) = 1/2$).

Similarly, the *information gain provided by the entire DT* readily derives from (A.1) (replace sample $\boldsymbol{S}$ by $\boldsymbol{LS}$ and test T by DT):

$$I_C^{DT}(\boldsymbol{LS}) = H_C(\boldsymbol{LS}) - H_{C/DT}(\boldsymbol{LS}) \qquad \text{(A.4)}$$

Another way of quantifying the ability of test T to produce purified successors is the *normalized information gain* or 'score' proposed in Reference 3

$$G_C^T(\boldsymbol{S}) \triangleq \frac{2I_C^T(\boldsymbol{S})}{H_C(\boldsymbol{S}) + H_T(\boldsymbol{S})} \qquad \text{(A.5)}$$

where $H_T(\boldsymbol{S})$ measures the uncertainty of the outcome of T in $\boldsymbol{S}$

$$H_T(\boldsymbol{S}) \triangleq -[p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S})\log_2 p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S}) + p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S})\log_2 p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S})] \qquad \text{(A.6)}$$

Expression (A.5) is more efficient than (A.1) thanks to its normalized properties; in particular, it is dimensionless and varies in between 0 (no gain of information is provided by the test T) and 1 (the test provides the maximum of information and yields perfectly pure subsets $\boldsymbol{S}_{\text{YES}}$ and $\boldsymbol{S}_{\text{NO}}$). (At the same time, $I_C^T(\boldsymbol{S})$ ranges from 0 to $H_C(\boldsymbol{S})$ respectively.)

Note: In the context of variable branching factors, another important advantage of the normalized measure is that its sample estimate is relatively unbiased[19], and thus allows one to compare in a fair fashion tests with variable numbers of outcomes. This is in strong contrast with the behaviour of the unnormalized information (A.1) used by ID3, whose positive bias, proportional to the number of successors, favours many-valued splits and often leads to more complex and less reliable trees[1,3,17].

In References 37 and 38 a similar argument is used to derive the so-called symmetrical τ measure from Goodman–Kruskal's asymmetrical τ[39]. The resulting measure is actually very similar to our normalized correlation measure: instead of the logarithmic entropy ($H = -\sum_i p_i \log p_i$) used in (A.5) it uses the so-called 'Gini' diversity index[12] or quadratic entropy ($H^2 = -\sum_i p_i(p_i - 1)$)[40,41].

Indeed, replacing H_C in (A.2) by

$$H_C^2(\boldsymbol{S}) \triangleq -[p(+/\boldsymbol{S})(p(+/\boldsymbol{S}) - 1) + p(-/\boldsymbol{S})(p(-/\boldsymbol{S}) - 1)]$$

H_T in (A.6)

$$H_T^2(\boldsymbol{S}) \triangleq -[p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S})(p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S}) - 1) + p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S})(p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S}) - 1)]$$

$H_{C/T}$ in (A.3) by

$$H_{C/T}^2(\boldsymbol{S}) \triangleq p(\boldsymbol{S}_{\text{YES}}/\boldsymbol{S})H_C^2(\boldsymbol{S}_{\text{YES}}) + p(\boldsymbol{S}_{\text{NO}}/\boldsymbol{S})H_C^2(\boldsymbol{S}_{\text{NO}})$$

I_C^T in (A.1) by

$$I_C^{2T}(\boldsymbol{S}) \triangleq H_C^2(\boldsymbol{S}) - H_{C/T}^2(\boldsymbol{S})$$

and defining by

$$H_{T/C}^2(\boldsymbol{S}) \triangleq p(\boldsymbol{S}_+/\boldsymbol{S})H_T^2(\boldsymbol{S}_+) + p(\boldsymbol{S}_-/\boldsymbol{S})H_T^2(\boldsymbol{S}_-)$$

and by

$$I_T^{2C}(\boldsymbol{S}) \triangleq H_T^2(\boldsymbol{S}) - H_{T/C}^2(\boldsymbol{S})$$

the analogue of formula (A.5) yields the symmetrical τ measure

$$\tau(\boldsymbol{S}) \triangleq \frac{I_C^{2T}(\boldsymbol{S}) + I_T^{2C}(\boldsymbol{S})}{H_C^2(\boldsymbol{S}) + H_T^2(\boldsymbol{S})}$$

A.2 *Information quantity*

The *information quantity* provided by a test attribute at a tree test node derives from (A.1) and equals

$$I_Q^T = N_{\text{node}} * I_C^T(\boldsymbol{S}) \qquad \text{(A.7)}$$

where N_{node} denotes the number of learning states contained in $\boldsymbol{S}$. Thanks to the additivity property of the above information quantity, the overall information quantity provided by a test attribute selected at different test nodes (with different threshold values) amounts to the sum of partial information quantities provided by this attribute at the different test nodes. Similarly, the overall *information quantity provided by a tree amounts to the sum of the information quantities provided at all its test nodes.*

A.3 *'Quality' measure of a tree*

Reference 16 proposes to quantify the quality of a tree by means of the quality measure

$$Q(DT;\ \boldsymbol{LS}) \triangleq N * I_C^{DT}(\boldsymbol{LS}) - q * C(DT) \tag{A.8}$$

where

- $I_C^{DT}(\boldsymbol{LS})$ is the information gain in $\boldsymbol{LS}$ provided by DT expressed by (A.4)
- $N * I_C^{DT}(\boldsymbol{LS})$ is the *total information* provided by the tree on the learning states classification
- $C(DT)$ is its complexity: $C(DT)$ = number of terminal nodes -1[y]
- q is an appropriate weighting factor (generally in the range of $[0 \ldots 20]$).

Section II.7 has pointed out interesting practical properties of the above measure. Below we mention some others, more theoretically oriented.

Interpretation. The above measure may be interpreted as the difference of the information quantity (number of bits) required to represent the classification of the $\boldsymbol{LS}$ explicitly, without any knowledge on the attribute–class relationship, and the number of bits required to represent the same information by using the DT (i.e. the sum of representation length of the tree itself, and of the residual classification information of the $\boldsymbol{LS}$ not provided by the tree). Thus, the higher the tree quality, the higher the amount of 'information compression' that is achieved.

Justification. From a theoretical point of view, the measure may be derived from Bayesian decision theory[16]. One may show that maximizing the quality is equivalent to maximizing the *a posteriori probability* of a tree, given the $\boldsymbol{LS}$, and assuming *a prior probability* of the tree decreasing exponentially with its complexity. Under this assumption, the weighting factor q would be inversely proportional to the prior complexity expectation.

It is interesting to note that q is related to α used in the χ^2-based stop splitting criterion of section II.4 by Reference 16

$$q(\alpha) = \frac{\chi^2_{cr}(\alpha)}{2 \ln 2} \tag{A.9}$$

Note also that an expression similar to (A.8) may be used to assess the quality of part of a tree. In particular, the quality increment due to a node's development into the two successors provided by the test T takes on the form

$$\Delta Q(T; \boldsymbol{S}) = N_{\text{node}} * I_C^T(\boldsymbol{S}) - q \tag{A.10}$$

since the corresponding complexity increase equals 1

A.4 *Usefulness and uses*

Let us specify the context within which the above various measures show to be particularly convenient.

(1) The information quantity expressed by (A.7) allows one to quantify and compare the overall purification ability of the various test attributes *at the various tree test nodes.* (Also, remember that the overall information quantity of a given test attribute selected at more than one test node equals the sum of its partial information quantities.)

On the other hand, the overall information quantity provided by a tree (totalling the partial ones provided by all the tree test nodes), as compared with that potentially contained in the learning set, allows one to assess conveniently the classification purity provided by this tree.

(2) The normalized information gain or score (see equation (A.5)) allows one to compare objectively the ability of the various candidate attributes to produce purified successors at *a given test node* of the growing tree, and hence to select the one furnishing the local optimum test.

(3) The quality measure Q expressed by (A.8) is particularly useful to identify objectively the range of 'optimal tradeoff' between information quantity provided by a tree and its complexity. Together with the corresponding test set error estimates, they allow one to fix an 'optimum' tree, for example during the backward pruning procedure described in section II.8.

Notes

[a] To fix ideas, in the context of transient stability the learning states are operating states; their classification is considered with respect to a preassigned contingency, as for example a three-phase short circuit; a handy means to classify a state with respect to this contingency is by assessing whether its corresponding critical clearing time lies above or below a threshold value, suggested by the current practice.

[b] In our transient stability example, a state is an operating state of the system in its normal, precontingency configuration; its attributes are parameters of this state, presumed to drive transient stability phenomena, as for example, voltage magnitudes, active–reactive powers, etc. These 'attributes' are proposed by the tree designer and automatically treated during the tree building. Later on, we will refer to these as the 'candidate attributes'.

[c] In our transient stability example, these could be the stable and unstable classes.

[d] For example, in Figure 1 attribute 'PG112' and its threshold value 1240 MW are found to be optimal at the top-node. The test PG112<1240 MW? contributes to increase the classification purity: the top node is composed of 41% stable and 59% unstable learning cases; its splitting according to the selected test yields 83% stable and 17% unstable learning states at its LH successor and 100% unstable learning states at its RH successor.

[e] In the example of Figure 1 the LH terminal node is a (stable) deadend (91% of the cases are stable), the remaining two terminal nodes are unstable leaves.

[f] Apparent as opposed to real increase in accuracy, refers to that measured in the **LS** as opposed to that measured in the **TS**, i.e. for states not used during the tree building.

[g] With one degree of freedom in the particular case considered here of a 2-class classification and dichotomic DTs. More generally, the number of degrees of freedom equals $(n_c - 1)(n_t - 1)$, where n_c is the number of classes and n_t the number of possible issues of the test (6)[19].

[h] In the sequel the terms *'test set error rate'*, *'misclassification rate'*, or *'error probability'* are used interchangeably.

[i] These two terms will be used interchangeably.

[j] Intuitively, complexity of a tree refers to the number of its nodes. In the context of the formulation used in this paper, it is more specifically defined to be equal to the number of the tree terminal

[k] I_C^{DT}, the information provided by the DT, is upper-bounded by the prior entropy of the learning set, which is independent of N (see equation (A.4) in the Appendix); $C(DT)$ represents the tree complexity and q an appropriate weighting factor (Appendix).

[l] Inequality (8) derives directly from equation (A.10).

[m] According to the definition of the complexity, $C(DT) = 10$ corresponds to a total node number of 21, since, by construction, the total number of nodes of a binary tree equals twice the number of terminal nodes minus one, i.e. twice its complexity plus one.

[n] Hence, the dimension of the attribute space equals the number of test attributes.

[o] In addition to the normal (presumably secure) situations gathered from past records of the system real life, a sufficient number of hypothetical weakened situations should be considered, in order to provide a data base representative of both secure and insecure situations.

[p] For example, in transient stability studies, one may use time-domain methods, integrating numerically the non-linear differential equations of motion to assess the system dynamics, or direct, Lyapunov-type methods. In the sequel, we will conventionally refer to them as 'system theory' methods, as opposed to heuristic ones, keeping in mind that generally these methods require bulky numerical computations incompatible with real time constraints, at least for real life power systems.

[q] Figure 4 readily illustrates how sensitivity can be assessed and control actions can be suggested using the attribute space. For example, operating state OS1 is obviously more stable than OS2; to reinforce stability of this latter, one should descrease the power level at generator 112.

[r] The description of the base case as well as the preselected probability laws used to generate the overall data base are available upon request. For more details the reader is kindly requested to refer to Reference 5.

[s] More generally, t threshold values determine $t+1$ stability classes.

[t] This – more apparent than real decrease – is discussed in section III.4.

[u] Total reactive power flow through the 400/225 kV transformer at a given substation.

[v] Total combined reactive power reserve of the power plants of Britanny.

[w] Pragmatic quality, as opposed to the theoretic quality of equation (A.8), takes into account the fact that in power system practice overly optimistic errors are more dangerous than overly pessimistic ones; it accounts also for the 'gravity' of the errors, e.g. in terms of the difference of the CCT of the misclassified states, with respect to the threshold value used to define the classes.

[x] Note that these are *apparent* probabilities, in that they are assessed on the basis of the sample of learning subsets, as opposed to real probabilities on all the possible states which however are generally unknown.

[y] Intuitively, the complexity of a tree generally refers to the number of its nodes. In the context of expression (A.8), it takes on the above more specific definition.