Sparse: An Intelligent Alarm Processor and Operator Assistant

Zita A. Vale and A. Machado e Moura, University of Porto
M. Fernanda Fernandes, Albino Marques, and Couto Rosado, REN/EDP
(National Transmission Network/Electricity of Portugal Group)
Carlos Ramos, Polytechnic Institute of Porto

Power system control centers operate and control medium and large electrical networks that usually cover a large geographic area. Important developments in computers and in techniques for transmitting information have enabled electrical utilities to accomplish these tasks. Supervisory control and data acquisition (SCADA) systems let these centers acquire information about the power system and its transmission to control centers in real time. Control center operators can also send orders to power plants and substations, performing remote control and maneuvers, which support unattended power-plant and substation operation.

Modern control centers use several computer applications, including load-flow analysis, state estimation, contingency analysis, and short-circuit analysis. However, human operators still must supervise the power system and make critical decisions. These tasks can be very complex, especially during emergency situations, when control center operators receive a huge flow of real-time information and must make decisions under great stress—sometimes, in the absence of the most experienced operators.

Under these conditions, operators might miss important information mixed in with a large amount of useless data. Important errors might arise, perhaps causing avoidable blackouts. Thus, a new generation of alarm processors were needed to provide more intelligent information to control center operators.

The symbolic nature of the reasoning involved in alarm processing made AI techniques a promising solution. Several years ago, large electrical utilities around the world began to develop knowledge-based applications to incorporate in their control centers. These applications handle the information available in the control center and help operators in decision making. However, such applications are very demanding, because, to support them, computers must handle enormous volumes of information in real time.

In 1986, the first results related to alarm handling and fault diagnosis using AI techniques appeared. Nowadays, an increasing number of power utilities are developing expert systems for alarm handling.

In this article, we share our experience in developing the Sparse (Expert System for Incident Analysis and Power Restoration Assistance) intelligent alarm processor for Portuguese substations control centers (SCCs). Such an undertaking can be very demanding, requiring much cooperation among utility experts, but can also be very rewarding because it leads to useful, efficient applications. The sidebar, “The Portuguese Transmission Network,” indicates the scope of the problems involved.

The purpose of Sparse

Sparse’s main goal is to help SCC operators during incidents to rapidly interpret the enormous quantities of alarm messages during incidents (for example, short circuits in transmission lines). Operators have been using Sparse in the field, with positive results.

The Sparse intelligent alarm processor helps operators at power system control centers to rapidly interpret enormous quantities of alarm messages during incidents (for example, short circuits in transmission lines). Operators have been using Sparse in the field, with positive results.
enormous quantity of alarm messages that Scada systems can generate, and to minimize the amount of nonsupplied power (the total amount of power not delivered to clients because of incidents).

Sparse also provides other operator-assistance features, such as advising operators about the actions they should or should not take (for example, in case of power restoration), and alerting them when abnormal events occur (for instance, when a breaker receives a tripping order and does not open). Moreover, Sparse supports:

- Supports knowledge about the network topology—namely, each line’s state (in service, out of service) and each substation’s operation type (manual, automatic).
- Informs operators about the beginning of an incident (such as a line opening because of protections tripping or voltage dropping in a substation, leading to automatic opening of breakers).
- Presents chronological conclusions about the incident.
- Shows a graphical image of the network, with important information about the elements and plants involved in the incident.
- Gives advice about required actions to restore service—namely, what lines will be handled by automatic operators, what actions operators should take, and what actions should not be continued.
- Reports actions taken and the corresponding agents (automatic operators, control center operators, operators in the substations).
- Informs operators of the end of the incident.
- Alerts them to abnormal situations that require further consideration.

Sparse performs all these tasks in real time, helping SCC operators control the power system. (See the sidebar, “An example of Sparse operation.”)

The hardware

Sparse runs on a DECstation 5000/240, under the Ultrix environment (see Figure 1). An Ethernet LAN of duplicate configuration connects this machine with the two µVax II machines that are in charge of control and supervision functions in the SCC. One µVax is always online, supporting Scada and control functions, and the other is in hot standby.

We have installed Sparse in a computer independent of the machines that support Scada functions in SCCs. This minimizes interference in control functions during installation, test stages, and normal operation. SCC machines have been in service since 1989 and are quite overloaded. However, new hardware and software for Portuguese SCCs have been ordered. We expect that this new system will be installed during 1998. We will integrate Sparse in this new system.

The software

To be able to send Sparse the real-time information that is required for alarm processing and operator assistance, we developed the Ttlogw (time-tag log) software application, and installed it in the µVax II machines. This application sends the relevant information to Sparse and receives some information (for example, a message to report the end of an incident) from it.

This application, which we developed in Fortran, includes the following features: When the power system is in a normal state, Ttlogw sends Sparse only those messages that address changes in the state of breakers or disconnectors or in the state of the type of plant operation. When Ttlogw detects the beginning of an incident, it sends to Sparse all the messages received (alarms and state changes) since two minutes before the beginning of the incident, until it receives information about the end of the incident (from Sparse or from SCC operators).

To give Sparse knowledge about the power system state, Ttlogw sends a state protocol when Sparse starts to run and when the µVax boots. This state protocol includes information about the state of all the breakers and disconnectors and of the operating system in all the substations.

Our system’s architecture includes a software-interface module, a preprocessing module, a knowledge base, an inference engine, an explanation module, and a user interface.

Figure 2 shows the software organization. Our system is mainly written in Prolog. The software interface and the preprocessing module are written in C language.

The knowledge base

Developing Sparse required several kinds of knowledge, including knowledge about power system elements and topology (power plants, substations, and feeders), the alarms that are generated at each moment (alarm lists), the information-transmission system (how data is transmitted and what is transmitted), and alarm interpretation. The first two kinds of knowledge are included in the fact base, under the form of Prolog facts.

The messages are handled by a preprocessing module written in C language. This module converts them into Prolog facts that are transmitted to Sparse, which processes them online. This preprocessor standardizes the information contained in each field of the message and converts the time and the date of the message into a numeric element, thereby simplifying the treatment of temporal problems. The preprocessor can handle more than 500 messages per second. Let us consider the following message:

```plaintext
µVax

Ttlogw

DECstation

Sparse software interface (SSI)

Preprocessing module

Rule base

Inference engine

Fact base

Explanations

User interface

User

µVax

Routing

Ethernet

Figure 1. Sparse’s hardware configuration.

Figure 2. Software organization.
```
The Portuguese Transmission Network

The Portuguese Transmission Network involves transmission lines and substations operated at 150, 220, and 400 kV, and some 60 kV lines. Figure A shows this transmission network. It includes about 6,000 km of lines and 43 substations, and its annual peak demand in 1995 was 5,250 MW.

The preprocessor identifies the following fields in this message:

- date (15-MAR-1996)
- time (16:05:09:962)
- plant (SVM) from where the message came
- code (145) of the feeder that originated the message
- name (ERMESINDE 1) of the feeder that originated the message
- event (reception of a TRIP ORDER)
- state (Active - 01)

The preprocessing module converts this...
Figure B presents the structure of the network’s control system and includes the following levels:

1. This level includes the National Dispatching Center responsible for energy-management functions.
2. This level includes
   - two substations control centers (Vermoin and Sacavém),
   - four hydraulic-power-plant control centers (Régua 1, Régua 2, Caniçada, and Castelo do Bode), and
   - thermal power plants that directly depend on the National Dispatching Center.
3. This level includes
   - substations operated through substations control centers, and
   - hydraulic power plants operated through hydraulic-power-plant control centers.

The two substations control centers have a Scada system that receives information from about 46 remote terminal units, corresponding to 600 bays (high voltage lines, transformers, and capacitor banks).

About 75% of the network’s substations include automatic operators that supervise the substation, detect incidents, and act as ideal operators, operating the required breakers to restore service or simplify the work of control center operators.7

When the power system is in a normal state, the information received at the SCC allows efficient real-time control. However, during an incident, especially when it involves several plants, the information presented to operators might include several hundreds of messages per minute. Human operators must interpret all the available information to understand the situation and to take measures to contain the incident, reach a wider area, and restore service as soon as possible.

To assist operators in these tasks, we have developed the Sparse (Expert System for Incident Analysis and Power Restoration Assistance) expert system for Portuguese SCCs.1–6

References
Figure C gives a set of messages that arrived at the Sacavém control center.

These messages correspond to an incident in the Ferreira do Alentejo-Palmela (SFA-SPM) line, involving only one phase that implied the tripping of both ends. Automatic reclosure equipment reclosed the line successfully in Palmela substation but unsuccessfully in Ferreira do Alentejo substation. For this incident, Sparse provided the conclusions given in Figure D.

The second message informed operators that a single-phase fault had occurred in the Palmela-Ferreira do Alentejo line. The automatic reclosure was successful in Palmela but unsuccessful in Ferreira do Alentejo. Sparse can enormously reduce the number of messages presented to operators. The original set of messages included almost 50 messages for a period of 130 ms. Sparse reduced the number of messages presented to operators during the same period to 5. These messages included not only an interpretation of the incident (the second and fourth messages) but also advice for operators (the third message). For this purpose, the system takes into account knowledge about OPA characteristics and about typical service-restoration procedures.

The conclusions to be made and the actions to be taken (A1 to A2, in this example) can be one of the following types:

L1. A fact whose truth must be proved.
R1. Generation of facts (conclusions to be asserted to the knowledge base).
R2. Elimination of facts (conclusions to be deleted from the knowledge base).
R3. Interaction with the user interface.

Figure 3 is a simplified version of a rule concerning a three-phase tripping with automatic reclosure by the protection system.

The inference engine

We have developed our system’s inference engine completely in Prolog, with this application specifically in mind. It allows efficient processing of alarm messages in real time, treating more than 1,200 alarm messages per minute.

The main issues that led us to develop our own inference engine were that

- The amount of the involved information is very large.
- The processing must be done in real time, according to the arrived information.
• The analysis involves complex temporal reasoning.
• The power system is a dynamically changing environment requiring nonmonotonic reasoning.

The inference engine uses a forward-chaining strategy of reasoning—triggering the appropriate rules when a new fact, external (alarm) or internal (conclusion), arrives, in order to derive new conclusions. It uses metaknowledge to select the appropriate rules and guide the reasoning process, improving system efficiency (see Figure 4).

For each arriving fact (alarm or conclusion), the rule selector selects the rules to be triggered. The rule selector generates facts, such as the following:

\[
\text{trigger}(NF, NR, T1, T2)
\]

That means that rule number NR should be triggered, until it is successful, between instant T1 and T2, because of the arrived fact number NF.

The metaknowledge used by this module specifies, in each case, the time that the system should wait before triggering the selected rules. This allows the introduction of a delay in the triggering of the rules. This delay lets the system wait, for some moments, for the arrival of facts that are premises of the rule to be triggered. In this case, two situations are possible: the delay corresponds to a tolerance allowed in message arrival (to consider delays in the transmission of the information), or the delay corresponds to the time required for the arrival of other messages involved in the same rule (for instance, the maximum time interval between a trip order message and a breaker open message). The delay also lets the system wait for a certain period of time, for the arrival of facts that are, in the negative form, premises of the rules to be triggered. (If one of a line’s extremities opens because of tripping, and the other extremity does not open within a defined time interval, the system informs the operator.)

An intelligent alarm processor’s inference engine must handle complex temporal reasoning—mainly because the sequence of events is especially important for the interpretation of alarm lists, or the meaning of a message depends on what happened before and, in certain cases, on what might happen next.

The rule-trigger module selects, for the present moment, among the selected rules, the next one to be triggered. For each triggered rule, the system checks the veracity of the premises. If the veracity is proved, the rule is successful and the system accepts the actions and conclusions of its right-hand side. When all the rules selected to be triggered until the present moment have been considered, the rule selector considers the facts that have arrived at the system and that have not yet been considered.

Because power systems are dynamic environments, the corresponding information also changes dynamically. This requires dynamically changing knowledge bases for intelligent alarm processors and inference engines supporting nonmonotonic reasoning. In fact, alarm messages reflect changes in the power system, and the arrival of a message can render false some conclusions previously achieved. The inference engine must be able to test the veracity of previous conclusions according to the most recent information.

In our system, “facts” that are found to be false are not simply erased from the knowledge base. They are recorded as “old facts,” allowing their later use for explanation facilities. The following fact

\[
\text{old_fact}(\text{-940}, \text{breaker}(\text{15-MAR-96}, \text{'16:00:29'}, \text{‘145’}, \text{protocol, closed}), \text{‘8162900’, ‘8191024’}).
\]

means that a fact reporting that the breaker of feeder 145 of the Vermoin substation (SVM) is closed has become true at instant 8162900 and has become false at instant 8191024. In this case, the fact that is retracted has a negative number (–940), which means it is a consequence of the state protocol received when the system boots.

**The user interface**

In developing our system, we paid special attention to the operator interface. The graphic interface is based on the X Window System, presenting the required information to the user in an easy, flexible, and well-structured way. Our system’s human interface is based on a set of widgets, including windows, scroll windows, buttons, menus, and dialogue fields (see Figure 5).

The command window has six buttons, which (along with providing other functions) let the user zoom in on a chosen area of the network and obtain explanations for the pre-
sented conclusions. In the same window, a menu bar lets the operator access complementary information about the electrical network's components (for example, functions of the automatic operator of a specified plant) and consult the rule base. The message window presents the expert-system conclusions. The network window presents a simplified image of the network, dynamically showing the changing information.

Although the user interface provides good performance, it presents an important drawback. Because Prolog predicates let us develop this interface with X Windows, it is very dependent on the machine and on the version of Prolog used. Therefore, we have developed, completely in C language, a new user interface for Sparse, based on X Windows and Motif. We are designing this new interface to be more portable and to provide adaptive and intelligent behavior. It is presently under test, and we plan to integrate it soon with Sparse.

The explanation module

One of the most interesting features of expert systems is their ability to explain their reasoning. Explanation facilities are very important during these systems' development, because they provide a trace of the undertaken reasoning. Explanations are also used for validating the expert system, because they allow comparisons between the reasoning of the expert systems and of human experts.

Our system includes an explanation module that allows its use as a tutor for novice operators. This possibility is very interesting because, fortunately, serious disturbances rarely occur in power systems, which makes it difficult to train operators without a simulator. Our system can replay real situations in a training program.

Explanation facilities are also very important for the success of Sparse. They help control center operators by presenting the reasoning used to derive conclusions.

Sparse provides a very friendly way to get explanations. The user can use the mouse to select conclusions from those presented by the system and to get explanations for those selected conclusions.

Important points in developing Sparse

The decision to develop an inference engine, with this application specifically in mind, allows real-time performance and consideration of temporal and nonmonotonic reasoning. This decision was very important to the success of the intelligent system, because, for very demanding applications, especially when flexible and complex temporal reasoning must be performed, using commercial shells has significant limitations.

Another important point for the success of Sparse was the participation of utility experts...
and operators from the beginning of the development phase. This made it possible to take into account their real needs and to involve them more deeply in the project goals. The most important and difficult part of the development of Sparse was acquiring the knowledge related to alarm interpretation. This phase took about one year and required the active involvement of very experienced operators, who are always very busy. Nevertheless, during this phase, these operators felt very enthusiastic about Sparse and made knowledge acquisition rather easy. Acquisition took place during several meetings, during which we analyzed message files corresponding to past incidents and we evaluated Sparse performance.

It is very important to ensure the use of an intelligent application after the test phase. In fact, some applications pass this phase but afterward are abandoned. The main reason for this is the difficulty of knowledge maintenance. For Sparse, we are developing tools to make this task as easy as possible.

The results already obtained with Sparse show the adequacy of AI techniques for alarm processing and operator assistance in power systems. Sparse is presently installed in the Veromoin SCC, running online, and the results obtained thus far are positive. We are undertaking some new research work, to be integrated with Sparse in the near future. This work involves the formal verification of the knowledge base, the Intelligent Tutoring System for operator training, and the automatic updating of the knowledge base via machine-learning techniques. Hence, intelligent applications can be developed and used in real time in power system control centers and plants, providing high-quality assistance to human operators and contributing to better quality in electrical-utilities service.

References


Zita A. Vale is a professor of power systems at the Faculty of Engineering of the University of Porto, Portugal. Her main research interests concern AI applications of power system operation and control, and user-interface and human-factors issues. She received her diploma and her PhD in electrical engineering, both from the University of Porto. She is a member of the IEEE (Power Engineering Society; Computer Society; Systems, Man, and Cybernetics Society; and Industry Applications Society), the ACM, and the Association for Logic Programming. Readers can contact Vale at the Dept. of Electrical and Computer Eng., Univ. of Porto, Faculty of Engineering, Rua dos Bragas, 4099 Porto Codex, Portugal; zav@fe.up.pt; http://www.cim.isep.ipp.pt.

A. Machado e Moura is a full professor of electrical power systems at the Faculty of Engineering of the University of Porto, Portugal. His main research interests concern power systems design and operation. He received his PhD in electrical engineering from the University of Porto. He is a member of the IEEE and the CIGRE (Conference Internationale des Grands Réseaux Électriques). Readers can contact Machado e Moura at the Dept. of Electrical and Computer Eng., Univ. of Porto, Faculty of Engineering, Rua dos Bragas, 4099 Porto Codex, Portugal; mmoura@fe.up.pt.

M. Fernanda Fernandes is head of the Control and Protection Systems Department at REN/EDP (National Transmission Network/Electricity of Portugal Group). Her main research interests concern power systems operation and control. She received a diploma in electrical engineering from Technical University of Lisbon (IST) and a diploma in computer engineering from New University of Lisbon. Readers can contact Fernandes at REN/EDP, Rua Cidade de Goa, 4, 2686 Sacavém Codex, Portugal; mff@edinfor.pt.

Albino Marques is head of the Department of Operation of the North Operation Group at REN (EDP Group). His main research interests concern power systems operation and control. He received a diploma in electrical engineering and an MS from the Faculty of Engineering at the University of Porto. Readers can contact Marques at REN/EDP, Apartado 3, 4471 Maia Codex, Portugal.

Couto Rosado is head of the Department of Control Centers at REN (EDP Group). His main research interests concern power systems operation and control. He received a diploma in electrical engineering from Technical University of Lisbon (IST). Readers can contact Couto Rosado at REN/EDP, Rua Cidade de Goa, 4, 2686 Sacavém Codex, Portugal.

Carlos Ramos is a coordinator professor in the Department of Computer Engineering of the Institute of Engineering at the Polytechnic Institute of Porto (ISEP/IPP), Portugal. His main research interests are AI applications and computer-integrated manufacturing. He received his diploma and his PhD in electrical engineering, both from the University of Porto. Readers can contact Ramos at the ISEP/IPP, Rua de S. Tomé, 4200 Porto, Portugal; csr@dei.isep.ipp.pt.