1992

A-teams for real-time operations

Sarosh Talukdar
Carnegie Mellon University

V. C. Ramesh

Carnegie Mellon University.Engineering Design Research Center.

A-Teams for Real-Time Operations

S.N. Talukdar, V.C. Ramesh

EDRC 18-31-92

A-TEAMS FOR REAL-TIME OPERATIONS

S. N. Talukdar **V. C. Ramesh**
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA., 15213
(e-mail: snt@edrc.cmu.edu)

Submitted to the International Journal of Electrical Power & Energy Systems, Butterworth Heinemann Ltd., Nov 12, 1991.

A-TEAMS FOR REAL-TIME OPERATIONS

Sarosh Talukdar V.C.Ramesh
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213, USA

ABSTRACT

Rapid changes in the operating environments of electric utilities call for modifications in their operating strategies. This paper proposes two ways for making modifications. The first is the adoption of an organizational scheme, called an A-team (Asynchronous Team), for EMS (Energy Management System) software. A-teams are modular and extremely flexible. They have the potential for very high performance. The second proposal is for the use of an event tree that would extend several contingencies from the current operating state into the future. This tree would be continually updated by a set of predictive and prescriptive programs, called control specialists, that would be distributed over a network of computers.

Keywords: Distributed Problem Solving, Real-Time Control, Organizations, Decision Support, Power System Security

INTRODUCTION

In its operations, every electric utility must deal with exogenous phenomena which it can neither control nor predict with certainty. These phenomena include customer demands, transactions with neighboring utilities and sudden disturbances or failures (called *contingencies*). To counteract the ill-effects of these phenomena, utilities use a mix of proactive (feedforward) and reactive (feedback) strategies for the real-time control of their networks. These strategies work very well and have allowed U.S. utilities to provide their customers with service of exceedingly high quality. The question is: When, if ever, will the existing strategies require a major renovation? We feel that the time for these renovations is here, because the operating environments of U.S. utilities are in the beginning of a period of rapid change, as indicated by the following observations:

- Customer demand is growing faster than system capacity. As a result, equipment usage is increasing and energy is being shipped over greater distances.
- The unbundling of services being required by deregulation will cause profound changes in operating practice. For instance, the familiar notions of costs and losses will have to be replaced by the unfamiliar and more complex notions of revenues and profits as operating objectives.
- NUGs (Non-Utility Generators), load management technologies, and other artifacts of deregulation are proliferating.
- Environmental concerns are increasing in importance and will play a larger role in operations.

We feel that the best way to adapt to the growing complexity being produced by the above changes, is to restructure the computer-based Decision Support System used in Energy Management Systems. The succeeding material makes some suggestions for how this restructuring should be done.

ORGANIZATIONS

We can, without any loss of generality, think of the decision support system for the real-time operations of any power network as consisting of an organization a computing environment (Fig.1). The organization consists of agents (interfaces and other programs), stores (databases) for the data the agents need and produce, and mechanisms for the agents and stores to interact. The computing environment consists of hardware and software needed to build and use the organization.

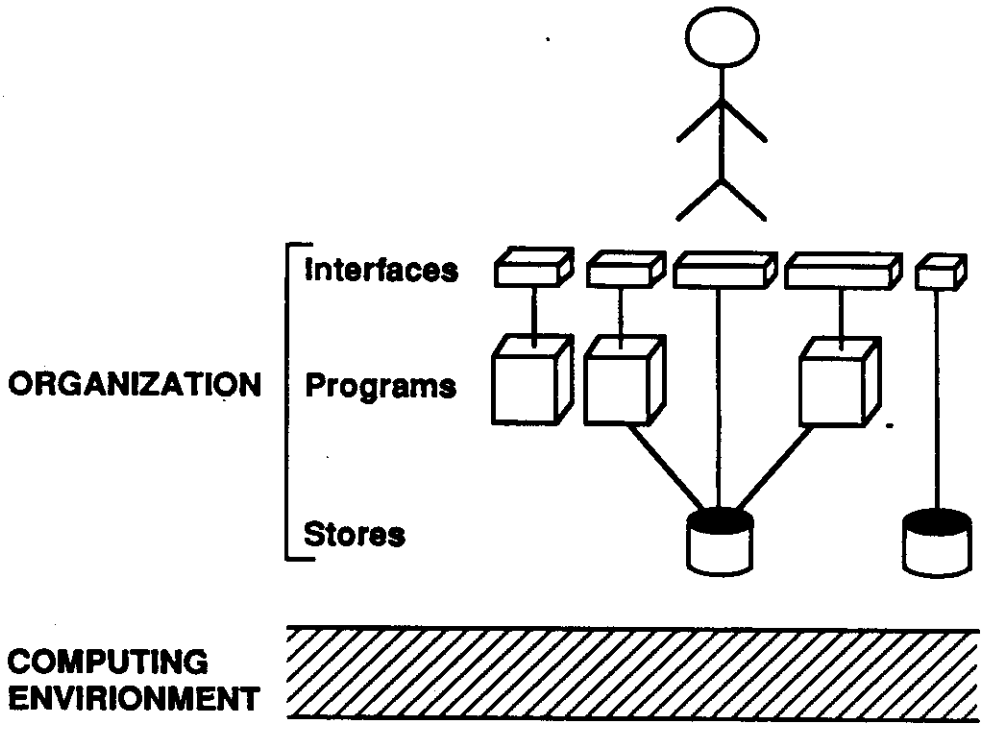


Fig. 1 A decision support system.

Computing environments are growing rapidly in power and capability. Organizations, however, have hardly changed since energy management systems were first built and constitute a bottleneck for the improvement of their decision support systems. Some of the weaknesses of these organizations are:

- severely limited rationality--the agents tend to be small in number, numeric rather than knowledge-based in nature, and predictive rather than prescriptive in function. As a result, existing tool-kits leave many parts of the decision-making process uncovered.
- inflexibility--it is very difficult to add new agents or change the interconnections of existing agents.
- poor interfaces--the agents tend to generate masses of numeric data that is not always easy for humans to assimilate and can lead to information overloading in emergencies.
- no capacity for automatic learning--the only learning that is done is by the human.

TERMINOLOGY AND MODELS

This section develops organizational models in preparation for a discussion of ways to eliminate some of the previously mentioned weaknesses. The models provide three viewpoints of an organization: topology, operating policy and openness (potential for growth).

TOPOLOGY

Harel has developed a formalism called a Higraph [Harel 88] that is convenient for visualizing most features of organizational topology. In the succeeding material we will modify Higraphs by the addition of a few additional features to obtain what we call a Tao graph. We will also develop an algebraic equivalent of a Tao graph that is useful in analysis.

Aspects and Stores

Complex problem-solving processes invariably involve large amounts of data and many different representations. In recognition of this fact, we will use a coarse unit of data called an *aspect*. Conceptually, an aspect is a view, model or partial description of some object of interest. For example, circuit diagrams, lists of materials, operating manuals and behavioral specifications are some of the many aspects of an electric motor. More formally, an aspect x is a double: x = (R, V), where R is the representational scheme used by the aspect and V is a collection of values that instantiates this representation.

We define a store as a set of aspects. A store is said to be *homogeneous* if all its aspects use the same representational scheme, otherwise, it is *heterogeneous*. Stores and their Cartesian products are denoted by closed figures (Fig. 2).

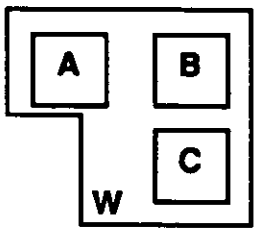


Fig. 2 Four stores such that: W = A x B x C

Agents and Tasks

We define an agent as an operator capable of mapping the elements of one store into those of another. This mapping may be one-to-one, one-to-many or many-to-many, it may be deterministic or stochastic, and it may be into or onto. Pictorially, we will denote all these types of mappings by directed arcs as shown in Fig. 3. Symbolically, an agent is denoted by the relation:

$$y = f^{k}_{ij}(x),\ x \in S_i \text{ and } y \in S_j \qquad \text{(E1)}$$

where k is the agent's name, S_i is the agent's input store and S_j is its the output store.

Fig. 3 An agent that maps from S_i to S_j.

Any computational task can be thought of as a transformation of a given aspect into a goal aspect. The task of designing a motor, for example, is equivalent to transforming an aspect that specifies the motor's behavior into an aspect that describes its structure. Thus, a task is defined by two aspects; the first is given; the second is to be calculated. If the semantic gap between these aspects is too large to conveniently bridged with a single agent, then intermediate stores may be placed to serve as stepping stones along the way. The connecting sequence of agents and intermediate stores is called a *computational path* or *string*.

Tao Graphs, Data Flows and Authority Flows

The purpose of a Tao graph is to help visualize the topology of an organization--the relative locations of its agents and stores, the tasks the organization can perform, the different paths it may take in performing these tasks, and the supervisory relations among its agents.

A Tao graph has two components: a data flow and an authority flow.

An *authority flow* is a set of broken arrows that represent the supervisory relations among agents. As such, an authority flow is equivalent to the"organization chart" that is traditionally used to depict who supervises whom in human organizations. The four principal types of authority flows are: a simple hierarchy (Fig. 4), a compound hierarchy (one with more than two levels), a matrix (at least one agent has more than one supervisor), and a null flow (there are no supervisors and all the agents must be completely autonomous).

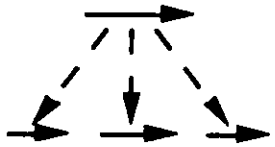


Fig. 4 A simple hierarchy.

A *data flow* is a directed graph whose nodes represent stores and whose arcs represent agents. We say that a data flow is *functional redundant* if at least one of its stores can be reached by more than one path. Cycles in a data flow (Fig. 5) make feedback and iteration possible, which in turn, make possible the reactive improvement and correction of computed results.

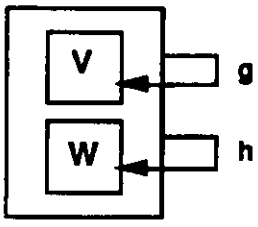


Fig. 5 Cycles in a data flow make iteration possible.

Note that all the information in a data flow can be expressed by a set of equations obtained by writing (E1) for every agent in the organization, as shown below:

$$y = f^{k}_{ij}(x), \text{ for } x \in S_i,\ y \in S_j, \text{ and } \forall\, k \in K \qquad \text{(E2)}$$

where K is the set of all agents. Though visually less appealing than a data flow, this set of equations has uses that we will get to shortly.

OPERATING POLICY

The operating policy of an organization is a collection of rules and regulations that govern its temporal activities. Two of the main issues are (c.f. (E2)):

- *coordination*: how is the input, x, for each agent to be selected from among the many entries that could accumulate in its input store? Does this selection require global information or can it be made locally?
- *coupling*: do the interactions among agents impose an order on their invocation? Can some or all of the agents work in parallel?

The coordination policy has a large effect on the quality, convergence and stability of an organization's computations. In addition, it influences the coupling among the agents. To illustrate, consider the following coordination policies for iteration with the cyclic data flow of Fig. 5:

(OP1): $v_{n+1} = g(v_n, w_n)$

$w_{n+1} = h(v_{n+1}, w_n)$ $\qquad n = 0,1,2,\text{-----}$

(OP2): $v_{n+1} = g(v_n, w_n)$

$w_{n+1} = h(v_n, w_n)$ $\qquad n = 0,1,2,\text{-----}$

(OP3): $v_{n+1} = g(v_n, w_*)$

$w_{n+1} = h(v_*, w_n)$ $\qquad n = 0,1,2,\text{-----}$

where "*" means "latest available," n is the iteration count, and $v \in V$, $w \in W$.

The choice of inputs dictated by the coordination policy of (OP1) requires the agents h and g to be operated serially in alternating order: g, h, g, h,---. The coordination policy of (OP2) is less restrictive. It allows the agents to be operated in parallel, provided the faster agent waits for the slower one between iterations. Thus, it allows the sequence: g//h, g//h,----. The coordination policy of (OP3) places no restrictions on the order in which the agents may be invoked; both agents may proceed in parallel and at different speeds.

When no agent has to wait for any other, we will say that the agents are *asynchronously* coupled. The advantage of asynchronous coupling is that it allows all the agents to work in parallel all the time. The disadvantage is that it is more prone to divergence and instability.

GROWTH

Two attributes that are useful in characterizing an organization's potential for growth are:

- *unit-of-growth*: the quantum of expansion. Typical quanta are agents and stores.
- *cost-of-growth:* the cost of any modifications that must be made to the organization to make it capable of accepting and using a new unit. Note: this cost does not include any expenses incurred in assembling and packaging the unit-of-growth.

CURRENT PRACTICE

Energy management systems have gone through two generations of evolution over the last thirty years. However, this evolution has largely bypassed their organizations which now, as was the case thirty years ago, are characterized by acyclic data flows, little functional redundancy and synchronously coupled agents (Fig. 6). The organizational weaknesses listed earlier are a result of these features. What types of organizations should be considered as replacements? The artificial intelligence community as well as a number of engineering domains have made widespread use of an organization called a blackboard [Nii 86a, b] (Fig. 7). This organization has the advantages of being much more open than that used by energy management systems and also, allows its agents to react opportunistically rather than confining them to predetermined computational sequences. However, any blackboard has three major weaknesses.First, it does not allow its agents to work in parallel. Second, the single, centralized store is difficult to expand. Third, the system is overly dependent on its supervisor; errors made by the supervisor critically affect performance; also the supervisor must be modified every time a new type of agent is added. The first two weaknesses are easily remedied. One could, for example, assemble a number of blackboards, distribute them over a network of computers, and assign some agents from each blackboard to handle interactions with other blackboards [Leao 88]. One approach to eliminating the third weakness would be to seek more dependable and self-modifying supervisors. Instead, we have chosen to eliminate the need for supervision, resulting in a class of organizations we call A-Teams.

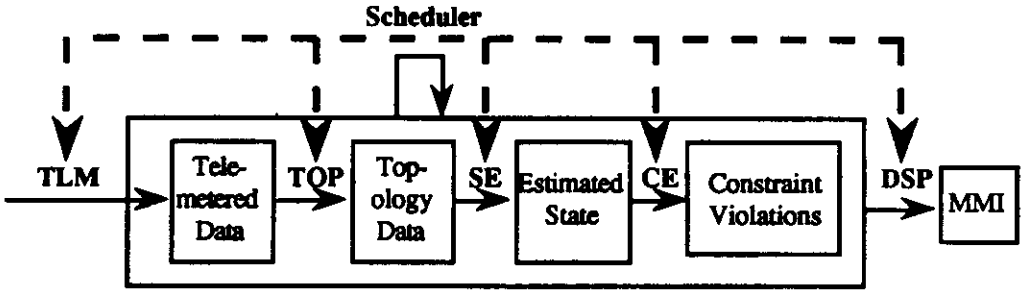


Fig 6. A partial TAO graph of security assessment in the first generation EMS. The Telemetry agent (TLM) acquires data from the real world and stores it. Certain data changes cause the Topology agent (TOP) to process network topology and store it. The State Estimation agent (SE) runs periodically to translate whatever telemetered and topology data is present into a state estimate of the power system. The new state estimate then triggers the Contingency Evaluation (CE) agent, which identifies critical contingencies, simulates them and determines the violation they cause. The Display agent (DSP) periodically translates data from any of a number of stores into a display screen store (MMI) that a human can examine and use to make decisions.

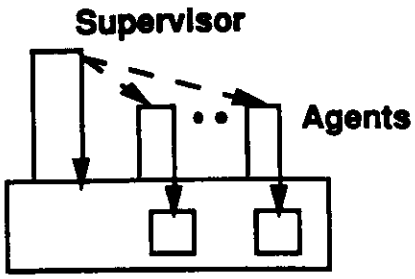


Fig 7. The topology of a traditional blackboard. The agents work serially in an order determined by the supervisor.

A-TEAMS

We define an A-Team as any organization whose authority flow is null (there are no supervisors), whose data flow is cyclic, and whose agents are asynchronously coupled (so all the agents can work in parallel virtually all the time).

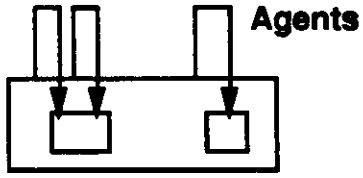


Fig. 8 The general from of an A-Team's data flow.

Attribute	A-Team	Traditional Blackboard
Topology		
Authority flow	none	simple hierarchy
Data flow	strongly cyclic	weakly cyclic
Functional redundancy	usually high	low to moderate
Operating Policy		
Coordination	local, often randomized	local
Coupling	asynchronous	synchronous
Growth		
Unit-of-growth	string	agent
Cost-of-growth	none	moderate to high

Table 1. Signatures (the values of some important organizational attributes) of an A-Team and a traditional blackboard.

A general form of the topology of an A-Team (Fig. 8) is very like that of its ancestor, the traditional blackboard (Fig. 7). However, there are differences in their operation and growth. Since A-Teams are often confused with blackboards, it is worthwhile to point out some of these differences, as is done in Table 1. Clearly, a traditional blackboard is not an A-Team. However, one can make a strong case for including both scientific communities, as described in [Kornfeld 81], and insect societies [Wilson 71] in the class of A-Teams. The arguments are given in [Talukdar 91]; we do not have the space to reproduce them here. We merely note that any organization which can serve the needs of agents as diverse as scientists and insects must have many strengths. These include easy growth (new agents and stores can be added without making any

modifications to the system), high dependability (high functional redundancy allows some agents or computational paths to fail without compromising overall performance), and high performance (large numbers of agents working in parallel allow wide spaces to be searched in relatively short times).

DESIGNING A-TEAMS

There are two principal steps to designing an A-Team:

- select a data flow; and
- devise a coordination policy for its agents. This policy must be locally implementable and allow the agents to be asynchronously coupled. (There are no supervisors to enforce coordination policy. Instead, each agent must make its own decisions with information contained in its input store.)

A CONJECTURE

We conjecture that there are a number of problem domains for which appropriate coordination policies (that is, policies that are local and asynchronous) can be devised that are simple (requiring no more than a few rules to be added to each agent), and effective (resulting in teams with high performance)

Some support for this conjecture is provided by the following observations.

- Consider a cyclic data flow with no functional redundancy (Fig. 5, for example), and the coordination policy of (OP3), namely: select the latest aspect placed in the input store. This policy is simple and local. Moreover, sufficient conditions for its convergence are only slightly more restrictive than the sufficient conditions for any synchronous policy, such as (OP1) [Talukdar 83].
- Many biological organizations with autonomous agents and a great deal of functional redundancy, flocks of birds and schools of fish, for example, appear to use simple, local and asynchronous coordination policies to produce collective behaviors that are quite complex [Ermentrout 91, Reynolds 87, Heppner 90].
- In two problem domains: solving sets of nonlinear algebraic equations and the travelling salesman problem, we have been able to demonstrate that simple and effective coordination policies exist [Talukdar 91].

APPLICATIONS

We feel that A-Teams will be useful, not as replacements to existing organizations in energy management systems, but rather as "add-ons" for new performing new tasks. One such task is described below.

EVENT TREES

In this section we will develop representations, called event trees, which can be useful in operations, and describe how such trees can be calculated by A-teams.

An event tree is a directed graph whose nodes represent states of a power system and whose arcs represent events that cause transitions from one state to another. The root node represents the existing state of the power system. Nodes that are n arcs from the root node are said to be in level-n.

The event tree has two basic forms. In the first and simpler form, called a C1-tree, the only events considered are unscheduled equipment outages (contingencies). In other

words, nodes in level-n represent states that would be reached after n equipment outages, provided no corrective action is taken after any outage.

The second form of event tree, called a C2-tree, allows for both outages and corrective actions. The result is similar to the trees obtained for a two person game such as chess. Odd levels contain states produced by outages (moves made by what might be considered as the power system operators' opponent). Even levels contain states resulting from corrective actions (moves made by the operators to counter their opponent).

As yet, C1- and C2-trees are fairly raw ideas and we can only speculate about their uses. We envision C1-trees being useful in evaluating dynamic security. To explain how, suppose that an outage occurs. A flurry of dynamic activity usually follows. Large excursions of voltages, power flows or generator speeds can result, causing further outages. Dynamic security studies are undertaken to predict these outages and thereby, the configurations in which the dynamics will leave the system. If these configurations and their steady state behaviors are acceptable then the system is deemed to be dynamically secure. A traditional approach making this determination, is to simulate the dynamics in excruciating detail. An alternate approach would be to use heuristics or system-specific knowledge to establish N, an upper bound on the number of outages that the dynamics in question could cause and then search the C1-tree up to level N+1 for its worst nodes. If these nodes were acceptable, then the system would be secure.

We envision C2-trees being useful in looking several contingencies into the future and in developing advance plans for corrective actions so any delay between the occurrence of a contingency and the institution of the best corrective action can be eliminated.

ASYNCHRONOUS MAINTENANCE OF EVENT TREES

Event trees are very large. If they are to be useful, they need to be pruned so only small and interesting portions are calculated. Also, the states and other operating conditions of the power systems they represent are always changing. Therefore, event trees need to be updated continually. We propose that both the pruning and updating be done by A-teams of control specialists. By "control specialist" we mean any predictive or prescriptive agent. As yet, we have developed only one such agent, a hybrid package called CQR, which consists of a mix of knowledge-based and numeric programs [Christie 89a,c]. CQR is a static security assessment agent; given the state of a power system, it will select the worst single-contingencies, evaluate the effects of these contingencies using an AC load flow, and summarize the effects in a brief report. As such, CQR is a agent for expanding nodes in C1-trees. Copies of CQR running in a distributed network of computers have been formed into an A-team for calculating and updating C1-trees. The data flow of this team is shown in Fig. 9. The coordination policy for the CQR-type agents is selection of the oldest unexpanded nodes from their input stores. The S-type agents serve the purpose of eliminating all obsolete entries from these stores. The result is an exceedingly simple A-Team. Nevertheless it can produce useful results of the sort shown in Fig. 10.

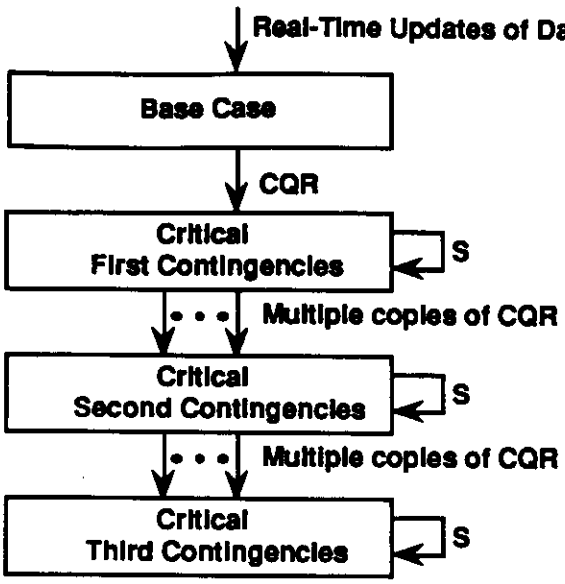
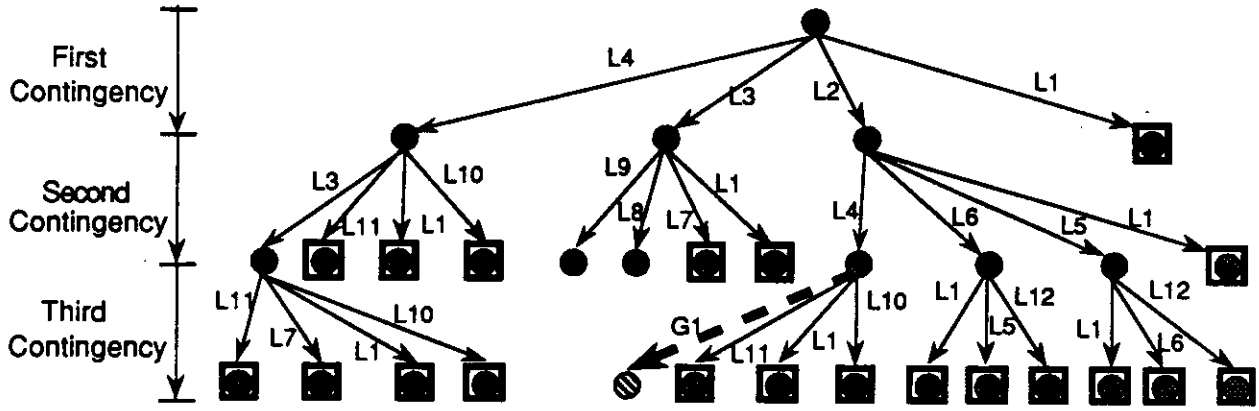


Fig 9. The data flow of an A-Team for maintaining an Event Tree of the C1 variety.



LEGEND

Loss of Lines →
L1: Muskngm138-Smmrfld138
L2: Olive345-Sorensn345
L3: Olive345-Olive138
L4: Muskngm345-Sporn345
L5: Olive138-SthBend138
L6: NCarlil138-Olive138
L7: Lima345-Lima138
L8: TnnrsCk345-TnnrsCk138
L9: Sorensn345-Sorensn138
L10: Philo138-Muskngm138
L11: Sporn345-Sporn138
L12: Olive138-Kankake138

System State

UNCORRECTABLE
(Violations are so severe that there is no time for corrective action)

CORRECTABLE
(Violations exist but there is enough time to correct them)

NO ACTION REQUIRED
(No Violations)

Loss of Generators - - - - →
G1: Muskngm345

Base Case
IEEE-AEP 118-bus test system
Outaged Lines: (Transfer Trip)
Hammer345-Tidd345
Tidd345-Tidd138
Muskngm345-Hammer345

Fig. 10 A snapshot of a C1-tree for the IEEE-AEP 118 bus test system.

Nodes in the tree are time-stamped. When a member of the A-team becomes available, it searches for the oldest node and updates it.

CONCLUSIONS

This paper has suggested two ideas. The first is the use of A-Teams for the organization of computer-based agents. The second is the use of event trees to help in predicting and planning power system operations. The ideas are still preliminary and a great deal needs to be done to flesh them out. Our immediate plans are to develop a number of control specialists capable of suggesting corrective actions and to incorporate these specialists into the A-team we have formed with copies of CQR.

ACKNOWLEDGEMENT

We are grateful to Anjan Bose for his ideas on using C1-trees for dynamic security assessment; to Richard Christie for his help with CQR; to Jim Nixon for his ideas on energy management; and to Richard Quadrel for his thoughts on insect societies.

REFERENCES

[Aho 83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, "Data Structures and Algorithms", Addison-Wesley, Reading, MA, 1983.

[Andrews 83] G. R. Andrews and F. B. Schneider, "Concepts and Notations for Concurrent Programming", Computing Surveys, Vol.15,No.1, March 1983.

[Ben-Ari 82] M. Ben-Ari, "Principles of Concurrent Programming", Prentice-Hall, Englewood Cliffs, N.J.,1982.

[Cardozo 87] E. Cardozo, "DPSK: A Kernel for Distributed Problem Solving", PhD Thesis, Carnegie Mellon University, January 1987.

[Christie 89a] R. D. Christie, "An expert system for on-line power system security assessment", PhD Thesis, Carnegie Mellon University, July 1989.

[Christie 89b] R. D. Christie et al, "Discrete Approximations and Means-Ends Analysis for Static Security Assessment", In Second Symposium on Expert Systems Applications to Power Systems, 1989.

[Christie 89c] R. D. Christie, S. N. Talukdar and J. C. Nixon, "CQR: A Hybrid Expert System," PICA-89, to appear in IEEE Transactions on PAS.

[Crowder 80] H. Crowder and M.W. Padberg. Solving Large-Scale Symmetric Travelling Salesman Problems to Optimality. *Management Science*, Vol. 26, No. 5, May 1980.

[Decker 87] Keith S. Decker. Distributed Problem-Solving Techniques: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics.* September/October, 1987.

[de Souza 91] P.S. de Souza and S.N. Talukdar. Genetic Algorithms in Asynchronous Teams. *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann, Los Altos, CA, July 1991.

[Erman 80] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys.* ACM. Vol. 12, pp. 213-253, June 1980.

[Ermentrout 91] B. Ermentrout, "An Adaptive Model for Synchrony in the Firefly Pteroptyx Malaccae", *Journal of Mathematical Biology*, Springer-Verlag, Vol 29, pg. 571-585.

[Findler 88] N. V. Findler et al, "An Examination of Distributed Planning in the World of Air Traffic Control", In A.H.Bond & L.Gasser (editors), Readings in Distributed Artificial Intelligence, pp. 617-627, Morgan Kaufmann Publishers Inc., 1988.

[Fink 78] L. H. Fink et al, "Operating under stress and strain", IEEE Spectrum, March 1978.

[Fletcher 87] R. Fletcher. "Practical methods of Optimization", Chirchester, UK: John Wiley & Sons, 1987.

[Fox 88] Mark Fox. An Organizational View of Distributed Systems. A.H. Bond & L. Gasser (eds.), *Readings in Distributed Artificial Intelligence*, pages 140-150. Morgan Kaufmann Publishers, Inc., 1988.

[Garey 79] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to NP-completeness", W.H. Freeman, San Francisco, CA, 1979.

[Goldbergh 89] D.E. Goldebergh. "Genetic Algorithms in Search, Optimization & Machine Learning", Reading, MA: Addison-Wesley, 1989.

[Goldbergh 87] D.E. Goldebergh and J. Richardson. "Genetic Algorithms with Sharing for Multimodal Function Optimization", Proceedings of the Second International Conference on Genetic Algorithms, Massachusetts Institute of Technology, Cambridge, MA, 1987.

[Harel 88] D. Harel. On Visual Formalisms. *Communications of the ACM.* Vol. 31, No. 5, May 1988.

[Heppner 90] Frank Heppner and U. Grenander, "A Stochastic Nonlinear Model for Coordinated Bird Flocks", from The Ubiquity of Chaos, ed. Saul Krasner, Washington: AAAS Publ. 1990.

[Hewitt 88] C. Hewitt. Officer are Open Systems. *Readings in Distributed Artificial Intelligence.* A.H. Bond and L. Gasser (eds.), Morgan Kaufmann, 1988.

[Irisarri 81] G. D. Irisarri et al, "An automatic contingency selection method for on-line security analysis", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-100, No.4, April 1981, pp.1838-1844.

[Johnson 90] D.S. Johnson. Local Optimization and the Travelling Salesman Problem. *Lectures Notes in Computer Science*, G. Goos and J. Hartmanis (eds.), Automata, Languages and Programming, 17th Int. Coll., Worwick University, Eng., July 90, Springer-Verlag, Vol. 443.

[Kim 85] J. Kim et al, "Contingency Ranking and Simulation for On-Line use", IEEE Transactions on PAS, Vol.PAS-104, No.9, September 1985, pp.2401-2407.

[Kornfeld 81] W.A. Kornfeld and C.E. Hewitt. The Scientific Community Metaphor. IEEE Trans. Sys., Man, Cybern., Vol. SMC-11, pp. 24-33, Jan. 1981.

[Lamps 81] B. W. Lampson et al, "Distributed Systems - Architecture and Implementation", Springer-Verlag, Berlin, 1981.

[Leao 88] L.V. Leao and S. Talukdar, "COPS: A System for Constructing Multiple Diagnosis", Expert Systems in Engineering, IFS Publications Ltd. England, D.T. Pham (ed.), 1988, pp. 241-252.

[Lin 73] S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, Vol. 21 (1973), pp. 498-516.

[Mullender 89] S. Mullender. Distributed Systems. *Addison-Wesley.* 1989.

[Nii 86a] H. Penny Nii. Blackboard Systems: The Blackboard Model of Problem Solving

[Nii 86b] H. Penny Nii. Blackboard Systems: The Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective, *The AI Magazine.* August 1986.

[Padberg 87] M. Padberg and G. Rinald. Optimization of a 532-city Symmetric Travelling Salesman Problem by Branch and Cut. *Operations Res. Lett.*, Vol. 6 (1987), pp. 1-7.

[Quadrel 91] Richard Quadrel. Asynchronous Design Environments: Architecture & Behavior. PhD Thesis. Department of Architecture. Carnegie Mellon University. September 1991.

[Reynolds 87] Craig W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Computer Graphics*, Vol. 21, No. 4, p. 25-33, July 1987.

[Rychener 88] M.D. Rychener. Expert Systems for Engineering Design. *Academic Press.* 1988.

[Simon 76] Herbert Simon. The Design of Large Computing Systems as an Organizational Problem. In P. Verberg et al, Organisatiewetenschap en praktijk, 1976.

[Steeb 88] R. Steeb et al, "Architectures for Distributed Air-Traffic Control", In A.H.Bond & L.Gasser (editors),Readings in Distributed Artificial Intelligence, pp.90-98, Morgan Kaufmann Publishers Inc., 1988.

[Talukdar 83] S.N. Talukdar, S.S. Pyo and R. Mehrotra, "Distributed Processors for Numerically Intense Problems", Final Report for EPRI Project, RP 1764-3, March 1983.

[Talukdar 86a] S. N. Talukdar et al, "A System for Distributed Problem Solving", Coupling Symbolic and Numerical Computing in Expert Systems, Elsiever Science Publishers B.V., New York, 1986, pp.59-68.

[Talukdar 86b] S. N. Talukdar and E. Cardozo, "Artificial Intelligence Technologies for Power System Operations", Report EL-4323, Electric Power Research Institute, Jan. 1986.

[Talukdar 88] S. N. Talukdar and E. Cardozo, "Building Large-Scale Software Organizations" in Expert systems for Engineering Design, edited by M. Rychener, Academic Press, 1988.

[Talukdar 89] S.N. Talukdar and S. Fenves. Towards a Framework for Concurrent Design. *Proceedings of the MIT-JSME Workshop on Cooperative Product Development.* D. Sriram. R. Logcher, and S. Fukuda (eds.), Cambridge, MA: MIT, Nov. 1989.

[Talukdar 90] S. N. Talukdar and P. S. deSouza, "Asynchronous Teams" Second SIAM Conference on Linear Algebra: Signals, Systems and Control, San Francisco, CA. Nov. 1990.

[Talukdar 91] S. Talukdar, P.S. de Souza, R. Quadrel, and V.C. Ramesh, "A-Teams: Multi-Agent Organizations for Distributed Iteration," Submitted to the 11th International Workshop on Distributed Artificial Intelligence; Glen Arbor, Michigan; Feb 26-28, 1992.

[Wilson 71] E.O. Wilson. *The Insect Societies.* Belknap/Harvard Press, 1971.

[Wollenberg 87] B. F. Wollenberg and T. Sakaguchi, "Artificial Intelligence in Power System Operations", Proceedings of the IEEE, Vol. 75, No.12, December 1987, pp. 1678-1685.