Carnegie Mellon University
Research Showcase @ CMU

Department of Electrical and Computer Engineering

Carnegie Institute of Technology

1992

Contingency constrained operations

Sarosh Talukdar
Carnegie Mellon University

V. C. Ramesh

Carnegie Mellon University.Engineering Design Research Center.

Follow this and additional works at: http://repository.cmu.edu/ece

This Technical Report is brought to you for free and open access by the Carnegie Institute of Technology at Research Showcase @ CMU. It has been accepted for inclusion in Department of Electrical and Computer Engineering by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Contingency Constrained Operations

S.N. Talukdar, V.C. Ramesh

EDRC 18-36-92

CONTINGENCY CONSTRAINED 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 Power Industry Computer Applications (PICA) '93 Conference, Phoenix, Arizona, May 4-7, 1993.

CONTINGENCY CONSTRAINED OPERATIONS

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

ABSTRACT
This paper develops a new approach to an old and difficult problem: how to make and update plans for correcting a given set of contingencies. The approach is to decompose the problem into a set of loosely coupled, much smaller problems that can be solved by a team of cooperating agents. The team is expandable and its agents are autonomous, work in parallel, and communicate asynchronously. Most of the current cadre of agents use algorithms that are either well known or fairly obvious. One type, however, uses a new algorithm for finding starting points for nonlinear programming codes. This algorithm is described and the performance of the team illustrated with small examples.

1. INTRODUCTION

1.1 Terminology

Think of a power system as a network containing m switches, each of which can be either open or closed. Thus, the system can adopt $M = 2^m$ different configurations, denoted by $C_0, C_1, \ldots C_M$, where C_0 is the current configuration. Let X_n be a vector whose elements are the bus voltages and bus power injections of C_n. Though X_n contains both state and control variables, we will, for the purposes of brevity, call it a state vector.

The concerns in operating a power system can be divided into two broad categories: cost and quality. Cost is usually represented by a function: $f(X_n)$. Quality concerns are usually expressed as a set of nonlinear relations (sometimes, called load and operating constraints) that are configuration-specific, and have the general form:

$$G_n(X_n, D(t)) = 0$$
$$H_n(X_n, D(t)) \leq 0$$

where t is time and D is a vector of exogenous, time-varying quantities, such as customer demands for electric energy.

X_n is said to be a normal state if it satisfies these constraints. S_n, the set of all normal states for configuration C_n, is called the normal set of C_n. Configurations for which S_n is empty are said to be uncorrectable; all other configurations are said to be correctable.

Two sorts of events can cause a system state to become abnormal: gradual changes in the exogenous variables, D, and sudden disturbances that result in random configuration changes. The latter can cause far larger excursions, and hence, are much more dangerous.

Let $\tau_n(X_{na}, X_{nb})$ be the least time required to change the state of C_n from X_{na} to X_{nb} through a sequence of control actions. We will call τ_n a transition delay. Note that τ_n is non-zero because many control actions are rate limited. The output of a typical generator can, for instance, be increased at most by a few megawatts per minute.

1.2 Focus

This paper deals with control actions to counter the ill effects of sudden disturbances. These actions can be discrete (switching operations) or continuous (changes in the independently controllable components of the state vector). The paper concentrates on the latter.

2. PROBLEM FORMULATION

2.1 Optimum Power Flows (OPFs)

One of the simplest operating philosophies is to minimize instantaneous costs while keeping the state normal. In other words:

$$\text{(OPF):} \quad \text{Min } f(X_0)$$
$$\text{s.t. } G_0(X_0, D) = 0$$
$$H_0(X_0, D) \leq 0$$

Since the dimensions of X_0, G_0, and H_0 are often of the order of 1000, this is a large problem; available techniques are barely able to solve it fast enough for the results to be useful in real-time operations [1, 2, 3].

2.2 Adding Contingency Constraints

How can one limit the ill effects of the random configurational changes that result from sudden disturbances? By far the most common practice involves two steps [4, 5, 6]. First, a set of critical configurational changes (called contingencies) is identified. Second, plans are made to reestablish a normal state within some short period after the occurrence of each contingency.

The identification of critical contingencies requires system-specific knowledge, much of which can be encoded in expert systems [7]. In other words, much if not all of the identification process can be

automated with existing techniques. The same is not true for planning responses to these contingencies. To understand why, suppose that the n-th contingency would cause the system's state to change from X_0 to X_{nc}. If X_{nc} is abnormal, the planning problem is to find a normal state, X_n, that can be achieved within an acceptably short time, say T_n. There are two different ways to formulate this problem: the first treats correction times as hard constraints; the second treats them in a softer way, specifically, as terms of an objective function. The modifications that result to (OPF) from these two treatments are indicated below:

$$
\text{(CCP1):} \quad \begin{array}{lll}
\text{Min} & f(X_0) & \\
\text{s.t.} & G_0(X_0, D) = 0 & \\
 & H_0(X_0, D) \le 0 & \\
 & G_n(X_n, D) = 0 & \\
 & H_n(X_n, D) \le 0 & n = 1, 2, \ldots, N \\
 & \tau_n(X_n - X_0) \le T_n &
\end{array}
$$

$$
\text{(CCP2):} \quad \text{Min } f(X_0) + \sum_{i=1}^{N} w_n \tau_n(X_n - X_0)
$$

$$
\begin{array}{lll}
\text{s.t.} & G_n(X_n, D) = 0 & \\
 & & n = 0, 1, \ldots, N \\
 & H_n(X_n, D) \le 0 &
\end{array}
$$

where: N is the number of contingencies to be considered; T_n is the time allowed for the n-th contingency to be corrected, w_n is a weight assigned to the n-th contingency; and it has been assumed that $\tau_n(X_n - X_{nc})$ can be approximated by $\tau_n(X_n - X_0)$.

Both these formulations are very large--at least N+1 times as large as (OPF). As such, both are beyond existing capabilities for fast, reliable and repeated solution. In addition, each requires the user to select some parameters: $\{T_n\}$ in the case of (CCP-1) and $\{w_n\}$ in the case of (CCP-2). It happens that the selection of $\{T_n\}$ is much more difficult. The explanation is as follows. Let $Z = [X_0, X_1, \ldots, X_N]$ be a vector called a super-state. Let S1 be the feasible set of (CCP1), that is, the set of all values of Z that satisfy the constraints of (CCP1). Let S2 be the feasible set of (CCP2). Then S1 is small and sensitive to the values selected for $\{T_n\}$ while S2 is much bigger and insensitive to the values of $\{w_n\}$. Another way of putting it is that the constraints of (CCP1) require all the contingencies be correctable and also, all the corrections be completed within time limits, $\{T_n\}$, that must be selected apriori. In contrast, the constraints of (CCP2) require only that all the contingencies be correctable. In selecting $\{T_n\}$ there is a considerable risk of making S1 empty, in which case little useful information is likely to result from attempts to solve (CCP1), even though these attempts be long and painful. In selecting $\{w_n\}$, however, the user is merely expressing an opinion on the relative importance of the contingencies and can adjust this opinion interactively.

Because of (CCP1)'s profound disadvantages relative to (CCP2), we will henceforth consider only (CCP2). Also, recall that the vector of exogenous variables, D, is time varying, and therefore, the solution of (CCP2) is time varying.

2.3 A Decomposition

Notice that the constraints of (CCP2) consist of N+1 independent blocks. As a result, (CCP2) can be decomposed into a set. $\{(IP_n)\}$, of N+1 subproblems each having the form:

$$
\text{(IPn):} \quad \begin{array}{lll}
\underset{X_n}{\text{Min}} & f_n(X_n, Z_n) & \\
\text{s.t:} & G_n(X_n, D) = 0 & \\
 & & n = 1, 2, \ldots, N \\
 & H_n(X_n, D) \le 0 &
\end{array}
$$

where:

$Z_n = Z \backslash X_n$, that is, the super-state Z with the elements of X_n removed

$$
f_0(X_0, Z_0) = f(X_0) + \sum_{n=1}^{N} w_n \tau_n(X_n - X_0)
$$

$$
f_n(X_n, Z_n) = \tau_n(X_n - X_0) \qquad \text{for } n = 1, 2, \ldots, N
$$

Let $\{(IP_n)\}$ be the set of all the (IPn) and Z^* be a simultaneous solution of this set. Then, it can easily be shown that Z^* is also a solution of (CCP2); hence, $\{(IP_n)\}$ and (CCP2) are equivalent [8].

2.4 A Skewed Approximation

Can the couplings among the members of $\{(IP_n)\}$ be loosened so their parallel solution becomes easier?

Note that the exact solution of $\{(IP_n)\}$ is unobtainable because the exact value of the exogenous vector, D, is unknown. The elements of D are time varying and are measured by sensors that can be hundreds of miles apart. There is always some delay and time skew in making and collecting these measurements. What if delays and time skews were allowed for the values of Z_n? More specifically, suppose that each (IP_n) is treated as a separate problem that is solved iteratively for X_n, while Z_n and D are treated as exogenous variables whose values are updated as new estimates of them become available. Then we have a set, $\{(IP_n')\}$, of more loosely coupled problems, each of the form:

$$
\text{(IPn'):} \quad \begin{array}{lll}
\underset{X_n}{\text{Min}} & f_n(X_n, Z'_n) & \\
\text{s.t:} & G_n(X_n, D') = 0 & \\
 & H_n(X_n, D') \le 0 &
\end{array}
$$

where Z'_n and D' are the latest available values of Z_n and D.

Consider the case where (IPn) has multiple solutions. Intuitively, one would expect each solution of (IP_n') to track the corresponding solution of (IPn) as it varies in time with an error that increases

smoothly with the skew in the values of Z'_n and D'. That this is actually the case is easily proved [8, 9].

3. MULTI-AGENT SOLUTION PROCESSES

3.1 Asynchronous Teams (A-Teams)

The preceding sections have decomposed the contingency constrained problem, (CCP2), into a set, {(IPn')}, of N+1 smaller problems, each of the form and size of an optimum power flow. The smaller problems are very loosely coupled and can be solved by a team of agents working in parallel, provided the team is properly organized.

The organization we will use is called an A-Team and is described in [10]. Its main features are:

- Agents with a multitude of skills are combined so they complement and help one another
- All the agents are autonomous. Most agents use only locally available data. All the agents work in parallel and communicate asynchronously (that is, no agent has to wait for results from another).
- The organization is very open. The addition of a new agent may require some modifications to that agent, but none to the rest of the organization. As a result, the number of agents tends to grow continually.
- The agents develop and maintain populations of solutions to the overall problem and its components.

The structural features of A-Teams make them well suited to distributed implementations in networks of computers. The key question is: can an A-Team be made to do anything useful? After all, autonomous agents, each deciding for itself what it is going to do and when, if ever, it will communicate with its team mates, can act at cross purposes. Surprisingly, there are simple strategies to keep this from happening. One of them is to balance agents that create solutions with agents that destroy them. In difficult integer programming problems (travelling salesman and robot design) this strategy has been shown to produce scale efficient behavior (as agents are added better solutions are obtained more quickly and speed of the team increases) and even synergistic cooperation (the capabilities of the team appear to be greater than the sum of the capabilities of its agents) [10].

To visualize how this strategy works, think of an A-Team as a distributed collection of memories (Fig. 1). One of these memories contains a population of solutions to the overall problem being considered. The others contain populations of solutions to sub-problems. Each population is continually transformed by agents working in parallel. Some, called creators, add members to the population, others, called destroyers, cull members from it. Suppose that for each population there are several criteria by which the goodness of members can be measured. Think of the population as a set of points in the space whose axes represent these criteria. We want the creators and destroyers to act so that together, they herd the population into a desirable part of the space. It has been demonstrated that this happens even with very narrow creators and destroyers, each able to take only a single criterion into account in making its decisions [11]. In essence, each creator works to produce solutions that are better in terms of its criterion; each destroyer tests solutions with respect to its criterion and eliminates those that fail.

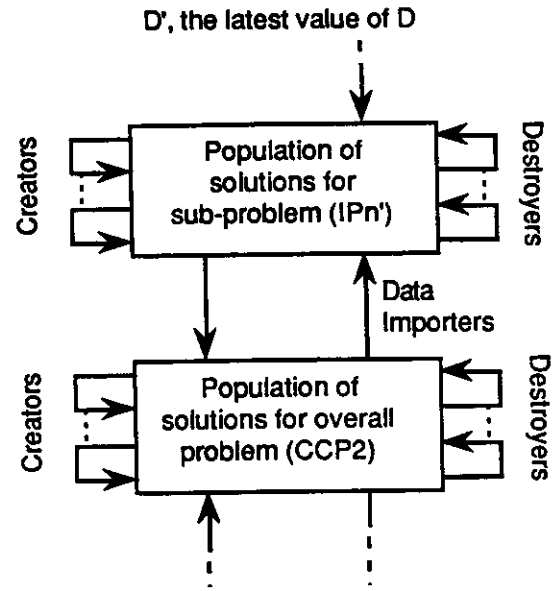

Fig 1: The structure of an A-Team. Memories are represented by rectangles; agents, by arrows.

3.2 Agents

Like rules in an expert system, the number of agents in an A-Team tends to grow continually. We do not have the space to describe all the agents now in the team for solving (CCP2). Instead, we will list the agents involved in maintaining a population of solutions to its principal sub-problem, (IPn'), and describe one of them in some detail. These agents are:

- Data importers: to collect the latest values of Z'_n and D'.
- Probes: to perform fine searches of neighborhoods, that is, to find local minima of (IPn') from given starting points. This capability is provided by conventional nonlinear programming codes [12].
- Voyagers: to provide starting points for the probes. The voyagers do this by conducting coarse searches that locate neighborhoods in which good solutions of (IPn') are likely to lie.
- Inhibitors: to place "fences" (implemented in the form of constraints) around neighborhoods that have been investigated, and thereby, to keep voyagers from wasting effort on revisiting these neighborhoods. (The fences are similar in concept to the discrete restrictions in tabu search [13], and can be viewed as their extension to continuous domains.)
- Destroyers: to perform two functions. First, to eliminate obsolete solutions and fences (Recall

that the solutions of (IPn') are time varying. Therefore, solutions that are valid at time t are less valid at time t+Δt.) Second, to eliminate results from voyagers that have become trapped in some unproductive part of their search space and force these voyagers to jump to a new location.

Except for the voyagers, these agents use either well known methods or fairly obvious heuristics. The voyagers, however, use a new trajectory-based heuristic that is described below.

4. THE VOYAGER ALGORITHM

4.1 Coarse Search by a Relaxed Interior Point Method

Consider an optimization problem with multiple solutions, some better than others. The purpose of a voyager is to find points close enough to the better solutions to serve as starting points for conventional nonlinear programming codes (these codes invariably employ greedy algorithms that head for the nearest local optimum, regardless of its quality). Fig. 2 illustrates how the voyagers and probes cooperate to find solutions.

The two main ideas behind the voyager-algorithm are:

- Replace the objective and constraints of the optimization problem to be solved ((IPn') in our case) by a binary vector field $\eta'(Xn)$. The magnitude of this field is 1 everywhere except at the solutions of the optimization problem, where it is 0. Every field line passing through an infeasible point leads to a feasible region; every field line that passing through a feasible point leads to a minimum.
- Each voyager behaves as a Newtonian particle under the influence of two forces. The first force acts to align the particle's velocity with the vector field ; the second, to keep the magnitude of the particle's velocity constant. As a result, particles are attracted by feasible regions and minima but circulate at near-constant speed, never stopping.

Once a voyager enters a feasible region it can only escape if the region is small. The constraints that delineate the boundaries of the region behave as elastic membranes, forcing the voyager back into the interior whenever it violates them. Thus, the voyager can be thought of as using a relaxed interior point method. (Traditional interior point methods [14, 15, 16] treat constraints as rigid rather than elastic boundaries. This requires a great deal of computational effort. Since there are few if any benefits, the relaxed approach seems preferable.)

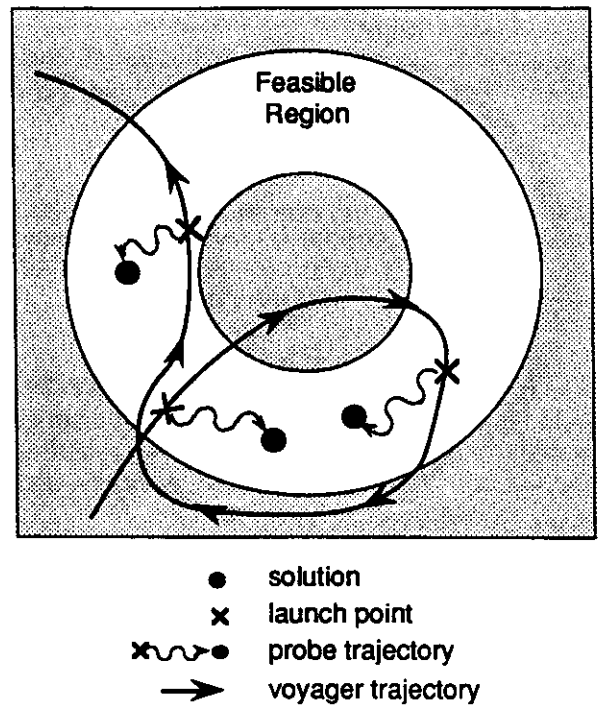

Fig 2: A voyager conducts a coarse search which takes it close to solutions of problem BH-2 (described in the appendix). Its purpose is to find starting (launch) points from which conventional nonlinear programming algorithms (called probes) can begin fine searches for the solutions.

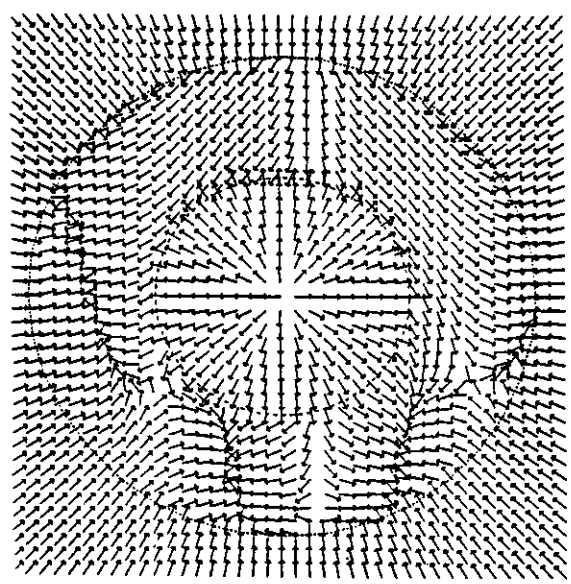

Fig 3: The vector field for the problem in Fig. 2

4.2 The Vector Field

The process for replacing (IPn') by a vector field is as follows. First, the equality constraints: $G_n(X_n, D')=0$ are approximated by inequalities:

$$G_n(X_n, D') \le a$$
$$G_n(X_n, D') \ge -a$$

where a is a non-negative, user-selected parameter. Thus (IPn') is replaced by a problem with only inequality constraints which, for the purposes of notational brevity, we will write as follows:

$$\text{(VP):} \quad \text{Min } f(X) \quad \text{s.t. } c_j(X) \le 0, \quad j=1, \ldots, M$$

where M is the total number of inequality constraints. Let: $\eta'(X)$ be the field vector at point X; I be the set of constraints that are violated at X, that is, $I = \{ j \mid c_j(X) > 0\}$; $\emptyset$ be the null set; ∇ be the gradient operator; and $\|\cdot\|$ be the Euclidean norm. Then:

$$\eta'(X) = \begin{cases} \dfrac{-\sum c_i(X)\,\nabla c_i(X)}{\|\sum c_i(X)\,\nabla c_i(X)\|} & i \in I, \quad \text{if } I \ne \emptyset \\[2ex] \dfrac{-\nabla f(X)}{\|\nabla f(X)\|} & \text{if } I = \emptyset \text{ and } \|\nabla f(X)\| \ne 0 \\[2ex] 0 & \text{if } I = \emptyset \text{ and } \|\nabla f(X)\| = 0 \end{cases}$$

In words, the field η' is of unit magnitude everywhere except at solutions to (VP). Also, at infeasible points the field is directed towards feasible regions; at feasible points, it is directed towards solutions of (VP).

4.3 Voyager Dynamics

The dynamics of the voyager are described by the following differential equations:

$$\frac{dV(X)}{dt} = \frac{1}{\tau}(B_1(X) + B_2(X))$$

$$\frac{dX}{dt} = V(X)$$

where:

$$B_1(X) = \|\eta(X) - V(X)\| N(X)$$

$$B_2(X) = W(E - \|V(X)\|)\frac{V(X)}{\|V(X)\|}$$

$\eta(X) = \eta'(X)\,\|V(X)\|$; τ is the mass of the voyager; N(X) is a unit vector that is normal to V(X) and in the plane of V(X) and $\eta'(X)$; E is the speed to be maintained; and W is a constant.

In words, the force B_1 is normal to V in direction and proportional to the angle between V and η' in magnitude. B_1 acts to align V with η'. In contrast, the force B_2 is co-linear with V and proportional to the difference between $\|V\|$ and E in magnitude. B_2 acts to make $\|V\|$ equal to the desired speed, E.

4.4 Discussion

By using only the direction, not the magnitude, of the vector field as a guide, the voyager is able to travel at near constant speed and thereby, keep from being trapped by minima. This feature distinguishes voyagers from other trajectory based algorithms [17, 18].

Besides voyagers, how can one pick starting points for nonlinear programming codes? Perhaps the most widely applicable technique is to pick the points randomly. Table1 compares the results obtained from starting points generated by a voyager to starting points generated randomly (random multi-start or RMS). The results were calculated by a nonlinear programming code (SQP routine VF13 from the Harwell Library) for several small but difficult optimization problems that are described in the Appendix. Calculations were continued until the global optimum was found. The results shown are the averages over three trials. The "function evaluation" counts include both function and gradient evaluations. "Good solutions" are those within 30% of the global optimum.

Table 1: Voyagers vs. random selection (RMS) for generating starting points for optimization problems. The problems increase in size from He-6 to Gr-20.

Problem	Starting point generation	No. of function evaluations	No. of solutions found	No. of good solutions found
He-6	Voyager	36,587	13	7
	RMS	44,030	10	3
Gr-10	Voyager	217,963	49	47
	RMS	659,853	48	38
Gr-15	Voyager	467,653	49	49
	RMS	1,850,368	45	38
Gr-20	Voyager	1,454,397	86	85
	RMS	5,916,456	85	71

Notice that the voyager picks better starting points than RMS and its advantage increases with problem size and dimension.

5. A SMALL POWER SYSTEM EXAMPLE

Some results from tests on a system with 6 buses, 11 lines and 3 generators [19] are reported here. Nine contingencies were considered, each involving the outage of a single line. Operating cost was approximated by a weighted sum of generations. Correction times were those required to make the generation shifts necessitated by each contingency.

An A-Team containing 10 voyagers (one for the base-case and one for each contingency), 10 probes and sundry other agents (c.f. Section 3), distributed over a net of 7 workstations was used to produce the results given in Table 2. The four cases vary in the weights assigned to the terms of the objective function of (CCP2). Specifically, the weights were chosen so only cost was minimized in Case-1; only the correction time for contingency-2 was minimized in Case-2; cost and all correction times were equally weighted in Case-3; and contingency-2 was given a greater weight, than cost or any other contingency, in Case-4. Thus, by varying the weights it is possible to explore the tradeoffs among cost and contingency correction times. In an EMS an operator could conduct such an exploration periodically and from the results, select the tradeoff he/she liked most.

Table 2
Results for a 6 bus, 11 line, 3 generator system

		Case-1	Case-2	Case-3	Case-4
Relative operating Cost		19.32	20.59	20.03	18.65
Contingency correction - time (min).	C1	4.61	29.67	5.13	0.24
	C2	31.76	4.67	28.07	13.03
	C3	20.41	9.75	14.32	3.78
	C4	4.52	30.32	0.76	21.77
	C5	8.18	32.68	8.03	24.30
	C6	11.90	29.41	5.97	21.20
	C7	11.08	35.70	11.78	27.07
	C8	28.75	30.72	6.62	22.29
	C9	3.24	30.42	0.82	22.02

6. CONCLUSIONS

This paper has developed a process for solving contingency constrained optimum power flows. The process has two main components. First, the overall problem is decomposed into a set of N+1 loosely coupled, smaller problems, each of the form and size of an optimum power flow, where N is the number of contingencies to be considered. Second, the smaller problems are solved by an asynchronous team of agents working in parallel. This team is open (so new agents can be easily added), distributable (so it can be readily implemented in a network of computers), and effective (so it finds good solutions quickly). The openness and distributability result from using autonomous agents that communicate asynchronously. The effectiveness is a result of two factors. First, the skills of several agents (particularly, voyagers and probes) are combined so they can find solutions that none of them could find alone. Second, the team maintains populations of solutions and so has greater coverage than an approach confined to working with a single solution. The populations are herded in profitable directions by the combined actions of agents that create solutions and agents that destroy them. We believe that in dealing with difficult problems, it is at least as important to be able to recognize and destroy bad solutions as to create good ones.

The solution process has, as yet, only been tested on small problems. Many implementation questions must be answered before the process can be applied to full-sized power systems and real time operations. Chief among these is: will the process scale-up to full-sized power systems? We believe that the answer is "yes." As a first approximation, the computational complexity of our A-Team is the complexity of its most complex agent. This agent is the nonlinear programming code used to find solutions to optimum power flows. Such codes have been in existence for some time and are known to perform fairly well on full-sized systems. Therefore, it is reasonable to expect the A-Team to perform at least as well.

7. REFERENCES

[1] B. Stott, O. Alsac and J.L. Marinho, "The Optimal power flow problem", *Electric Power Problems: The Mathematical Challenge*, Philadelphia, PA: SIAM Publ. 1980, pp. 327-351.

[2] J. Carpentier, "Optimal power flows", *International Journal of Electrical Power Energy Systems*, Vol. 1, pp. 3-15, Apr 1979.

[3] W.F. Tinney and D.I. Sun, "Optimal power flow: Research and code development," *EPRI Project 1724-1*, Final Report, Feb. 1987.

[4] B. Stott, O. Alsac, and A.J. Monticelli, "Security analysis and optimization", *Proceedings of the IEEE*, Vol. 75, No. 12, Dec. 1987, pp 1623-1644.

[5] H.J.C.P. Pinto, M.V.F. Pereira, and M.J. Teixeira, "New parallel algorithms for the security constrained dispatch with post-contingency corrective actions", *Proceedings of the 10th Power Systems Computation Conference*, Austria, August 1990, Butterworths, pp. 848-854.

[6] T.C. Giras, "A variable-metric method and parallel processing for contingency-constrained power flow optimization", *Ph.D. Thesis*, Carnegie Mellon University, Pittsburgh, Jan. 1984.

[7] Christie, R.D., Talukdar, S.N. and Nixon, J.C., "CQR: A hybrid expert system for security assessment", *IEEE Transactions on Power Systems*, Vol. 5, No. 4, November 1990, pp. 1503-1509.

[8] V.C. Ramesh, "A parallel global optimization approach and its application to power systems security", *Ph.D. Thesis*, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, to be submitted.

[9] Talukdar, S.T., Pyo, S.S. and Mehrotra, R., "Designing Algorithms & Assignments for Distributed Processing", *Final Report for EPRI Research Project 1764-3*, Nov., 1983.

[10] Talukdar, S.N., "Asynchronous Teams", *Fourth Symposium on Expert Systems Application to Power Systems*, Jan 4-8 1993, Melbourne, Australia.

[11] Murthy, S., "Synergy in Cooperating Agents: Designing Manipulators from Task Specifications,", *Ph.D. Dissertation*, Carnegie Mellon University, September 1992.

[12] Harwell Subroutine Library Specification (routines VF13, VF04), 1987.

[13] F. Glover, "Tabu search, part II", *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 4-32.

[14] Karmarker, N., "A new polynomial-time algorithm for linear programming", *Combinatorica 4*, 1984, pp. 373-395.

[15] K.A. Clements, P.W. Davis and K.D. Frey, "An interior point algorithm for weighted least absolute value power system state estimation", *IEEE Paper* 91 WM 235-2 PWRS.

[16] C.N. Lu and M.R. Unum, "Network constrained security control using an interior point algorithm", *IEEE paper* 92 SM 584-3 PWRS.

[17] A.H.G. Rinnooykan and G.T. Timer, "Global Optimization", *Handbooks in OR & MS*, Vol. 1, G.L. Nemhauser et al, Eds., Elsevier Science Publishers B.V. (North-Holland) 1989.

[18] P.M. Pardalos and J.B. Rosen, "Constrained Global Optimization: Algorithms and Applications", *Lecture Notes in Computer Science*, No. 268, Springer-Verlag, 1987.

[19] Wood, A.J. and Wollenberg, B.F., *Power Generation Operation and Control*, John Wiley and Sons, 1984, pg 74, Fig. 4.1.

[20] Torn, A. and Zilinskas, A., "Global Optimization", *Lecture Notes in Computer Science*, No. 350, Springer-Verlag, 1989.

APPENDIX

Some small but difficult optimization problems, adapted from [20], are listed below.

BH-2

$$\text{Min } \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$

$$\text{s.t.} \quad (x_1 - 5)^2 + (x_2 - 10)^2 \geq 5^2$$
$$(x_1 - 5)^2 + (x_2 - 10)^2 \leq 10^2$$

The problem has 3 minima at (-3.142, 12.275), (3.142, 2.275), (9.425, 2.475) with objective 0.398.

He-6

$$\text{Min} \quad -(25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2 + (x_6 - 4)^2$$

$$\text{s.t.} \quad x_1, x_2 \geq 0,\ 1 \leq x_3 \leq 5,\ 0 \leq x_4 \leq 6,\ 1 \leq x_5 \leq 5,$$
$$0 \leq x_6 \leq 10,\ 2 \leq x_1 + x_2 \leq 6,\ -x_1 + x_2 \leq 2,$$
$$x_1 - 3x_2 \leq 2,\ 4 \leq (x_3 - 3)^2 + x_4,\ 4 \leq (x_5 - 3)^2 + x_6$$

There is a global minimum at $x^* = (5, 1, 5, 0, 5, 10)$ with value -310

Gr-10, Gr-15, Gr-20

$$\text{Min} \quad \sum_{i=1}^{N} x_i^2 / d - \prod_{i=1}^{N} \cos(x_i/\sqrt{i}) + 1$$

$$\text{s.t.} \quad -600 \leq x_i \leq 600,\ i = 1, 2, \ldots N$$
$$\sum_{i=1}^{N} x_i^2 \leq (1800)^2$$

For Gr-10,	d = 4000,	N = 10
For Gr-15,	d = 80,000,	N = 15
For Gr-20,	d = 800,000,	N = 20

All three of these problems have a global minimum at the origin with value zero. There are several thousand local minima.