Coarse-Grained Distributed Optimal Power Flow

Balho H. Kim and Ross Baldick
The University of Texas at Austin, TX 78712

ABSTRACT

We present an approach to parallelizing optimal power flow (OPF) that is suitable for coarse-grained distributed implementation and is applicable to very large interconnected power systems. We demonstrate the approach on several medium size systems, including IEEE Test Systems and parts of the ERCOT system. Our simulations demonstrate the feasibility of distributed implementation of OPF. Rough estimates are made of parallel efficiencies and speed-ups.

1 INTRODUCTION

In this paper, we present an approach to parallelizing optimal power flow (OPF) that is suitable for distributed implementation and is applicable to very large inter-connected power systems. We demonstrate the approach on several medium size systems, including IEEE Test Systems and parts of the Electric Reliability Council of Texas (ERCOT) system. The approach could be used by utilities to optimize economy interchange without disclosing details of their operating costs to competitors.

Unlike traditional approaches to parallel non-contingency constrained (NCC) OPF (see [6, 8, 9] and the discussion in [11], for example) that concentrate on parallelizing individual steps such as Jacobian factorization, we propose a decomposition of the overall OPF problem into regions. We solve optimal power flows for each region and coordinate the multiple OPFs through an iterative update on constraint Lagrange multipliers. For all but very small systems, the process converges very rapidly to a solution of the overall OPF.

The iterative updates require the exchange of a very modest amount of data between adjacent regions. Although our current *implementation* is centralized, our approach is suitable for distributed implementation because of the very small amount of data that must be transferred between processors. To the best of our knowledge, our implementation is the first demonstration of the viability of large-scale distributed OPF.

96 SM 564-5 PWRS A paper recomemended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the 1996 IEEE/PES Summer Meeting, July 28 - August 1, 1996, in Denver, Colorado. Manuscript submitted July 21, 1995; made available for printing June 25, 1996.

We propose a scenario where each individual utility solves a modified OPF that includes its own service area and the borders it shares with other utilities. The modified OPF is similar to a standard OPF except that "dummy generators" are modeled at the border buses. The dummy generators mimic the effects of the external part of the system through a cost for supply of real and reactive power, voltage support, etc.

Naturally, the OPFs solved in each region can be implemented with the fastest available algorithms. However, it is also possible for each utility to have a *different* OPF implementation for its area. Our paper therefore concentrates on the issue of coordinating the regional OPFs. We discuss the regional OPFs themselves only briefly and note that through our use of OPF as a *building block* we can incorporate essentially any OPF algorithm into our approach.

The overall algorithm involves alternating solution of individual OPFs and updates of Lagrange multipliers. It converges, in principle, to a solution of the overall multi-utility OPF, yielding appropriate generation levels in each utility to minimize overall production costs. The Lagrange multipliers on the constraints could be used to set prices for exchange of real and reactive power. However, alternative ways to distribute savings, such as the split savings rule, can also be used.

There are two dummy generators for each transmission tie-line between a pair of regions. One of the dummy generators is placed in each region. To minimize the coupling between the regions and therefore maximize the solution speed, it is reasonable to divide the overall system in a way that minimizes the number of transmission lines in the cutsets of lines defining the regions. Fortunately, this will usually be consistent with dividing a multi-utility system into individual utilities. This is because typical utilities tend to have a relatively complex mesh transmission system internally, but relatively few, often radial, connections externally. Our two largest test systems, for example, are portions of the ERCOT system divided into their constituent companies.

In summary, the most effective implementation of our distributed scheme corresponds well with the most likely institutional implementation: division into regions along utility boundaries. In our scheme, there is no need for a uniform implementation of OPF across all utilities, nor any need for all the utilities in the system to run full OPFs, so long as each region can represent dummy generators in its OPF or economic dispatch (ED). This means that the

distributed OPF can be implemented by individual utilities across a multi-utility system without major disruption to existing OPF or ED investments in the utilities' energy management systems.

We mention that our computational results point to only modest speed-ups and efficiencies, even in ideal situations; however, we believe that the features of the distributed algorithm make it the *only* feasible way to achieve OPF in large-scale interconnected systems. Furthermore, its features make it ideal for use as a *trading tool* to enhance inter-utility trading of electricity.

In contrast, a centralized implementation of large-scale OPF (whether using a serial or parallel algorithm) suffers from several technical and institutional drawbacks, including:

- Communication bottlenecks in pooling information at a single control center.
- Anti-trust prohibitions against pooling of multi-utility data. (We note that the United States Department of Justice has already begun anti-trust investigations into Texas utilities concerning electricity trade [5].)
- Reliability issues in having a single center dispatch a large geographical area.

Our distributed algorithm avoids these drawbacks and presents a practical approach to optimizing multi-utility systems in a competitive environment. The approach can also be used to solve state estimation or solve the power flow equations for a multi-utility region without explicit external network models and without the exchange of large amounts of line, load, and generator data.

In the next section, we review the basic theory necessary to perform the regional decomposition that allows parallel and distributed solution of OPF. In section 3 we present results based on our prototype implementation. We conclude in section 4, with technical details relegated to the Appendix.

2 Regional Decomposition

2.1 Illustration

To illustrate the regional decomposition, we will consider dividing a power system into two overlapping regions. In practice, we envisage a multi-utility system being divided into its constituent utilities; however, the main issues can be illustrated in the two region case.

2.1.1 Variables. Because of our emphasis on the decomposition rather than the OPF itself, we will follow [14] in not distinguishing the controls from the dependent variables in our formulation. Instead, we will distinguish the variables by their *geographical* relationship to the regional decomposition.

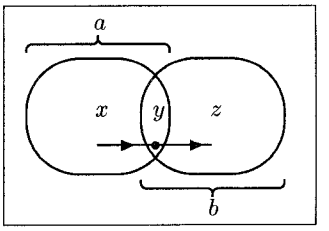

Fig. 1: Decomposition of a power system into two overlapping regions, a and b. The border variables in the overlap region are denoted y, while the core variables in regions a and b are denoted x and z, respectively.

Consider figure 1, which shows the case of a single tie-line joining regions a and b. Between and common to the two regions there is an overlap region, with a vector of variables denoted by y. The entries in y are defined as follows. For each tie-line we must include a bus in the border region. If there is no bus already there, we create a "dummy bus." Associated with each dummy bus are the real and reactive power flows through the bus and the voltage and angle at the bus. That is, the vector y has four entries for each tie-line.

In addition, in figure 1 we have shown vectors of variables x and z. The vector x consists of all the OPF variables that are relevant to region a but not already included in y. Similarly, z includes the region b variables not included in y. In summary, region a has state vector (x, y), while region b has state vector (y, z). The y variables are the overlap or border variables, while x and z can be thought of as core variables for regions a and b, respectively. In typical systems, the vector y will be much smaller than the vectors x and z and we will make use of this observation in analysis of the decomposition.

2.1.2 Objective. For the purposes of exposition, we will adopt minimum cost of production as our objective, but recognize that other considerations, such as minimizing the number of controls to be rescheduled, must also be considered. To analyze the decomposed system, we assume that the production costs for the whole system can be written as $c_a(x) + c_b(z)$. That is, we are assuming that there are no generators included in the border between the two regions.

2.1.3 Constraints. We assume that the constraints on the system involve x and y or y and z, but not x and z nor x, y, and z. That is, we assume that the constraints in each region involve only the core variables and the border variables for that region.

This assumption is reasonable for the power flow equations, since the bus admittance matrix couples only those variables pertaining to buses that are connected directly by a line. For example, a tie-line limit would be represented as a constraint on the flows to and from the border buses. If some of the other constraints are functions of elements of both x and z, then this can be handled by

moving more of the state vector into the border vector y. That is, our assumption on the dependence of constraints on core and border variables can always be satisfied, but it may require us to increase the dimension of y by enlarging the border region.

With this assumption, we can write the power flow constraints for region a in the form $F_a(x,y) = \mathbf{0}$ and for region b in the form $F_b(y,z) = \mathbf{0}$. Similarly, we can write the inequality constraints for region a in the form $G_a(x,y) \leq \mathbf{0}$ and for region b in the form $G_b(y,z) \leq \mathbf{0}$. The functions G_a and G_b represent the line flow, voltage, and contingency constraints in the individual regions.

Define the two sets: $A = \{(x,y) : F_a(x,y) = \mathbf{0}, G_a(x,y) \leq \mathbf{0}\}$ and $B = \{(y,z) : F_b(y,z) = \mathbf{0}, G_b(y,z) \leq \mathbf{0}\}$. Then a feasible power flow solution is a point (x,y,z) that satisfies $(x,y) \in A$ and $(y,z) \in B$.

2.1.4 OPF formulation. The OPF problem can be written formally as:

$$\min_{\substack{(x,y)\in A\\(y,z)\in B}} \{c_a(x) + c_b(z)\}. \tag{1}$$

We make the usual assumptions that c_a and c_b are convex approximations to the actual cost functions and that there is a unique solution to (1). These assumptions are discussed in [10, §III.B].

2.1.5 Serial OPF algorithm. We assume that there is an OPF algorithm available. Throughout this paper we will refer to such an algorithm as a "serial OPF algorithm." We will use the serial OPF algorithm as a building block in our parallel scheme by solving OPFs for region a and for region b separately.

2.2 Auxiliary problem principle

We are going to decompose the problem into regions by duplicating the border variables and imposing coupling constraints between the two variables. Our approach is inspired by the work of Batut and Renaud who were the first to apply duplication and decomposition to power systems problems in their unit commitment formulation [1]. The approach is superficially like diakoptics [7], but differs fundamentally in the details of the decomposition. As discussed in [1], standard Lagrangian approaches to relaxing the coupling constraints can be expected to converge slowly to the solution. Therefore, we follow Batut and Renaud in using a linearized augmented Lagrangian approach to improve convergence.

First, define the copies of y to be y_a and y_b, assigned to the regions a and b, respectively. Then, for $\gamma \geq 0$, (1) is equivalent to:

$$\min_{\substack{(x,y_a)\in A\\(y_b,z)\in B}} \{c_a(x) + c_b(z) + \tfrac{\gamma}{2}\|y_a - y_b\|^2 : y_a - y_b = \mathbf{0}\}. \tag{2}$$

So far we have complicated the problem without changing its solution. The quadratic term added to the objective does not affect the solution since the constraint $y_a - y_b = \mathbf{0}$ will make the quadratic term equal to zero at any solution; however, when we decompose the problem, this term will significantly aid in convergence [1].

Next we apply the "auxiliary problem principle" [3]. Under certain conditions, we can solve (2) by solving a sequence of problems of the form:

$$(x^{k+1}, y_a^{k+1}, y_b^{k+1}, z^{k+1}) = \operatorname*{argmin}_{\substack{(x,y_a)\in A\\(y_b,z)\in B}} \left\{ \begin{array}{l} c_a(x) + c_b(z) + \\ \frac{\beta}{2}\left\|y_a - y_a^k\right\|^2 + \frac{\beta}{2}\left\|y_b - y_b^k\right\|^2 + \\ \gamma(y_a - y_b)^\dagger(y_a^k - y_b^k) + \\ {\lambda^k}^\dagger(y_a - y_b) \end{array} \right\}, \tag{3}$$

$$\lambda^{k+1} = \lambda^k + \alpha(y_a^{k+1} - y_b^{k+1}), \tag{4}$$

where the superscript k is the iteration index, α and β are positive constants, and the superscript $\dagger$ denotes transpose. Some sufficient conditions for this iterative scheme to converge to a solution of (2) are presented in the Appendix. While (4) appears to be simply a sub-gradient update, the "proximal point" terms in (3) enhance the convergence of the overall algorithm.

The initial conditions $x^0, y_a^0, y_b^0, z^0, \lambda^0$ can be any convenient starting point such as a previous solution or flat start. The value of the Lagrange multiplier λ_i at iteration k is an estimate of the cost to maintain the constraint $y_{ai} - y_{bi} = 0$. If y_i represents, for example, power flow from region a to b along a particular line, then λ_i is the "shadow-cost" [12] on the interchange of power along that line. If some regions must import power to satisfy local demand, then the initial conditions for the border flows can be set to reflect the generation deficiency; however, this is not necessary for convergence since the dummy generators can be arranged to supply the imports necessary for a feasible initial solution.

For the purposes of distributing the computations, the important thing to note is that problem (3) separates into smaller problems for regions a and b, respectively. For example, the problem for region a is:

$$(x^{k+1}, y_a^{k+1}) = \operatorname*{argmin}_{(x,y_a)\in A} \left\{ \begin{array}{l} c_a(x) + \frac{\beta}{2}\left\|y_a - y_a^k\right\|^2 + \\ \gamma {y_a}^\dagger(y_a^k - y_b^k) + {\lambda^k}^\dagger y_a \end{array} \right\}, \tag{5}$$

which is essentially an OPF problem for region a including its border. The second through fourth terms in the objective of (5) constitute the cost function of the dummy generators in region a. The costs are quadratic and depend on the values of the Lagrange multipliers as well as on previous values of the iterates.

In figure 1, we illustrated a single tie-line between regions a and b. At the border bus, there is a real and

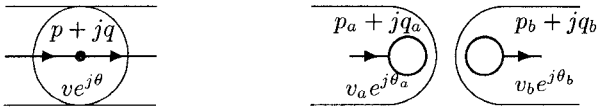

Fig. 2: (Left panel) A tie-line joining two regions with a dummy bus defined at the border. (Right panel) The dummy bus on the tie-line is duplicated into two dummy generators, with variables y_a and y_b, respectively.

reactive power flow, measured for example in the direction from a to b, and a voltage magnitude and phase. The dummy generators are created by duplicating this bus. The border variables are then $y_a = (\ p_a \quad q_a \quad v_a \quad \theta_a\ )^\dagger$ and $y_b = (\ p_b \quad q_b \quad v_b \quad \theta_b\ )^\dagger$, respectively, the real and reactive flow and the voltage magnitude and phase at the copies of the dummy bus in regions a and b. This situation is illustrated in figure 2. The iterative process drives the values of y_a and y_b together.

To enhance convergence, we make a modification to the basic scheme. In each regional OPF, the reference bus angle is arbitrary. At each iteration, we choose this angle so that the average of the border angles in region a equals the average of the border angles in region b. In the case of three or more regions, we use an *ad hoc* approach to approximately match the average of the border angles at each iteration.

2.3 *Distributed implementation, communication, and synchronization issues*

By locating each processor physically in its assigned region, the typical length of the communication path for telemetering data for the regional optimization would be on the order of the radius of the region. This contrasts with a centralized implementation where the typical length for the data path would be on the order of the radius of the whole system.

Since y is a much smaller vector than x or z, the communication overhead necessary to perform the Lagrange multiplier update (4) for each iteration of the algorithm is small compared to the total amount of information to perform OPF. The update only requires communication between adjacent regions. Therefore, if the number of iterations needed for (3)–(4) to converge is relatively small, then the overall communication overhead will be small. (This is in stark contrast to the communication bottlenecks that are likely with any on-line implementation of a parallel OPF scheme that requires multi-utility data to be pooled at a single control center.)

A natural implementation of the distributed algorithm is shown in figure 3. The Telemeter and Dispatch steps require intra-regional communication of data and control signals. The loop termination criterion requires global communication, while the Exchange step only requires communication between adjacent regions. In the case of multiple regions, each region will solve an OPF for its core and border variables.

Initialize $x^0, y_a^0, y_b^0, z^0, \lambda^0$;
$k = -1$;
Telemeter load and topology data from each region to its processor;
Repeat{
  Increment k;
  In parallel, solve the regional OPF for region a (using (5)) and for region b;
  Exchange y_a^k and y_b^k between regional processors;
  Update λ^{k+1} using (4);
  } Until y_a^k and y_b^k converge to within tolerance;
Dispatch generators according to OPF solution.

Fig. 3: Distributed implementation of parallel OPF.

3 Results

3.1 *Prototype Implementation*

3.1.1 Serial OPF. We implemented a *serial* NCC AC OPF using GAMS 2.25 with the MINOS package [2]. The serial OPF was used to directly solve the case study systems described in section 3.2. These solutions provided data to validate the distributed algorithm.

3.1.2 Parallel OPF. We then used GAMS to centrally simulate the parallel computations involved in the Repeat loop of the algorithm in figure 3. Non-contingency constrained AC OPFs were performed for all cases with real and reactive generator limits and line and voltage constraints imposed. All GAMS computations were performed on a Sun Sparc-20 workstation.

The GAMS system provides a convenient tool to prototype and test algorithms and has allowed us to relatively quickly develop the distributed algorithm. However, it does not yield realistic estimates of the growth of cputime versus the number of buses. In particular, the cputime versus number of buses relationship for solving the OPF with GAMS grows faster than for a state-of-the-art implementation such as described in [14]. For this reason, we will present both raw cputime results from GAMS and also results calibrated on the basis of cputime data presented in [14]. In summary, our approach is two-fold:

1. Use GAMS to demonstrate the basic convergence properties of the overall algorithm, such as the number of iterations to satisfy a stopping criterion, and,

2. Use the cputimes from a state-of-the-art implementation to estimate the speed-ups possible with the algorithm.

Buses	Regions	Core Buses	Ties	Lines	Load
50	2	24,24	2	80	50
78	3	24,24,24	6	126	74
108	4	24,24,24,24	12	186	100
238	2	118,118	2	376	76
360	3	118,118,118	6	570	126
376	2	271,105	3	574	157
753	4	271,105,128,237	12	1100	209

Table 1: Case study systems.

Naturally, these estimates should not be taken too literally, but serve only to demonstrate that the decomposition is viable.

3.2 Case Study Systems

Data from two IEEE Reliability Test Systems and four Texas utilities were used to demonstrate the performance of the algorithm. Table 1 summarizes the test systems. The first column shows the total number of buses in each system, while the second and third columns show the number of regions and the number of core buses in each region. The fourth column shows the number of tie-lines that interconnect the regions, while the fifth column shows the total number of lines in each complete system. The last column shows the total per unit loads in the systems. The five smaller systems consist of two, three, or four copies of two IEEE Test Systems, while the two Texas systems use line data from, respectively, two and four Texas utilities. The line data is available from the authors on request.

The objective to be minimized is the production cost for active and reactive power. The cost of reactive power is assumed to be 10^{-6} of the active power cost for each generator, while real power costs were adapted from [13, §3 and 4]. The cost data is available from the authors on request.

In order to see how the algorithm responds to small changes in system status, we solved a base-case and several change-cases for each system. Each base-case was solved from a flat start with initially no interchange on any tie-line, while the change-cases were solved using the solution of the base-case as a starting point. The change-cases were as follows:

1. increase in demand of 5% at all demand buses;
2. increase in demand of 10% at all demand buses;
3. an outage of a single generator with capacity equal to approximately 2–3% of the total system demand.

The change-cases demonstrate the tracking behavior of the algorithm for an on-line application.

3.3 Test Results

3.3.1 Reliability of convergence and selection of parameters. In all cases tested, the iterates produced by the distributed algorithm converged towards the solution obtained by the serial GAMS implementation; however, the rate of convergence is dependent on the system and on the parameters α, β, and γ. These parameters were tuned for each system to minimize the number of iterations required to satisfy the stopping criterion specified in subsection 3.3.2. A brief discussion of the choice of parameters is contained in the Appendix.

3.3.2 Stopping criterion. We chose the maximum mismatch between the border variables as the stopping criterion. To select the tolerance on the maximum mismatch, we experimented with the performance of the algorithm. We found that the choice 0.03 per unit maximum mismatch yielded a solution with total costs that were within 0.1% of the optimal production costs from the serial algorithm. Typically, the mismatches on most buses were much smaller than 0.03 per unit.

For some small systems, including the 50 bus system described in Table 1, this criterion required over 20 iterations. However, for larger systems, including the two ERCOT systems, the criterion required no more than five iterations to satisfy the stopping criterion.

The mismatch tolerance of 0.03 per unit may seem large; however, the sensitivities of total dispatch costs to tie-line flows are relatively small when the system is close to being optimally dispatched. Furthermore, the mismatch in net interchange was usually much smaller than 0.03 per unit when summed across all lines from one region to another. Therefore, in practice, any simple strategy such as splitting the difference will usually be adequate in scheduling near-to-optimal interchange levels.

For each test case, we calculated the total costs by dispatching the generators at the levels suggested by the last iteration of the algorithm and then performing a power flow for the whole system. That is, the reported costs are based on a power flow and represent *bona fide* costs for a feasible dispatch of the whole system.

3.3.3 Cputimes. Because our prototypical GAMS implementation is not efficient, the cputime to perform the calculations is not reflective of performance in a production environment. Nevertheless, the data is presented for completeness and also because it provides some qualitative information that is useful in judging the performance of an efficient implementation.

Figure 4 shows the cputime versus the number of buses for the base- and change-cases for solving the OPF serially with GAMS. (All GAMS cputimes include overhead and set-up times that would be largely eliminated in production code.)

As expected, the trend is generally for cputime to in-

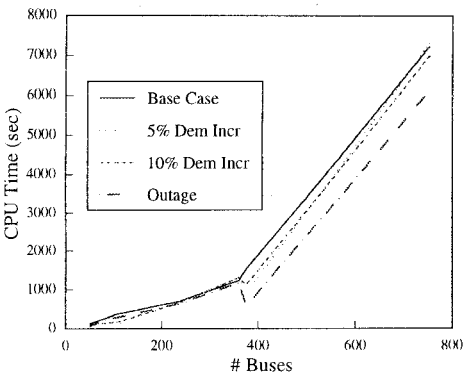

Fig. 4: Cputime to solve OPF serially. (Sun Sparc-20 cpu seconds.)

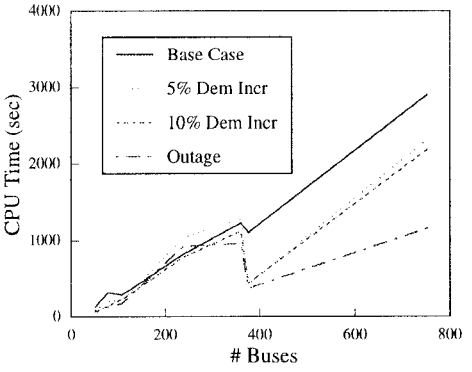

Fig. 5: Total cputime to solve OPF using parallel algorithm. (Sun Sparc-20 cpu seconds.)

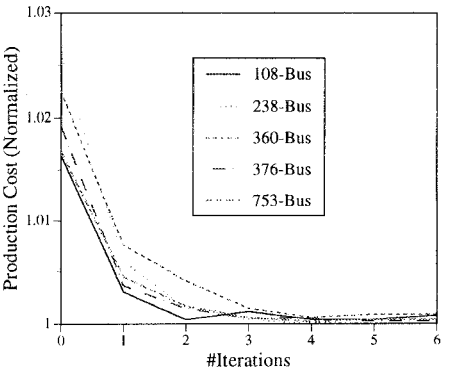

Fig. 6: Production costs for parallel algorithm versus number of iterations for base-case.

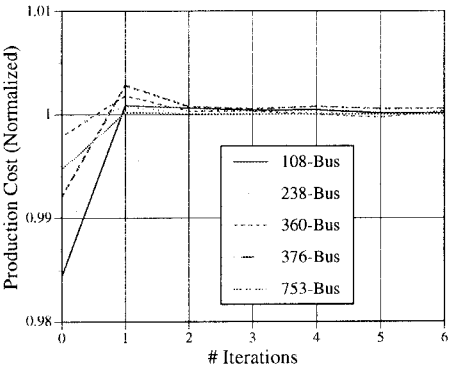

Fig. 7: Production costs for parallel algorithm versus number of iterations for generator outage case.

crease with the number of buses. While the growth is not monotonic across all seven systems, it is monotonic for the IEEE systems as a group (the five smallest systems) and for the Texas systems as a pair. Clearly, solution difficulty does not only depend on the number of buses but also on other parameters such as the number of lines. The change-cases generally take about the same or less time to solve than the base-cases.

Figure 5 shows the cputime versus the number of buses for the base- and change-cases for solving the OPF using the parallel algorithm in a centralized implementation. Each regional OPF was solved with GAMS and, as described above, the calculations were performed by simulating the parallel computations on a single workstation. The data in figure 5 show *total* cputime, where the iterations are terminated when the absolute value of real and reactive power mismatches on the tie-lines are all less than a tolerance of 0.03 per unit.

3.3.4 Progress towards optimum. Figures 6 and 7 show, respectively, the convergence of the parallel algorithm to:

1. the optimal value of the base-case and,
2. the optimal value of the generator outage change-case.

The vertical axis of each graph shows the production costs normalized by the optimal production costs for the corresponding case and system.

In figure 6, the value of the vertical axis at the 0-th iteration shows the normalized total production costs if each region is optimized separately assuming no interchange of power. The 0-th iteration data in figure 6 therefore represents the increase in costs of self-dispatch over optimal dispatch for each case and system. In figure 7, the value of vertical axis at the 0-th iteration shows the normalized production costs before the generator outage occurred.

In both figures, the values at subsequent iterations show the total costs *if* the generations obtained at this iteration were actually used to dispatch the system. As indicated above, to calculate the costs for each iteration, we took the generations suggested by the algorithm and solved the power flow equations for the whole system using these generations. (We chose as swing bus a generator near a load center.) We then calculated the costs resulting from this dispatch. That is, mismatches were reduced to zero to calculate *bona fide* total system costs.

The most significant feature of these graphs is that the solution converges within 3 or 4 iterations. That is, most of the reduction in production costs are obtained in the first few iterations.

The relationship between cputime and the number of iterations is shown in figure 8. The second and subsequent iterations of the parallel algorithm each take between 5% and 30% of the time required for the first iteration. This is because the second and subsequent iterations involve OPFs that have the same constraints as the first iteration, start with a feasible solution (the solution from the

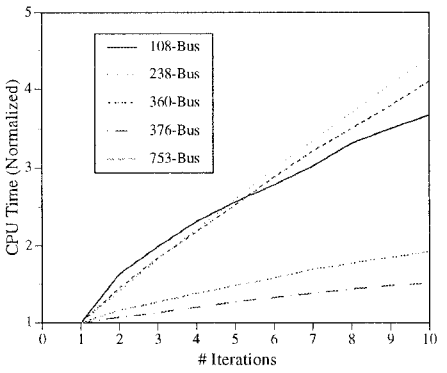

Fig. 8: Cputime for parallel algorithm versus number of iterations for base case. (Sun Sparc-20 cpu seconds. (Normalized.))

previous iteration), and involve only small changes to the objective.

Our GAMS implementation only makes rudimentary use of the information from previous iterations. An efficient implementation could exploit the repeated solution of similar problems to avoid refactorization of the Jacobian, further reducing the time for the subsequent iterations. This observation indicates that it is not very critical to choose a stopping criterion for the parallel algorithm, since most of the improvement in the costs is obtained in the first few iterations, while an extra iteration or two does not take very much more time. As we have noted, the cputime used by GAMS is much longer than for an efficient implementation; however, with an efficient OPF we would still expect to see a similar reduction in cputime per iteration after the first iteration.

3.4 Speed-up and Efficiency

3.4.1 GAMS results. The raw cputime data in figures 4 and 5 suggest parallel "efficiencies" (based on the ratio of serial to parallel cputime) greater than one. However, this is merely an artifact of the inefficiency of the GAMS implementation of the OPF. That is, the cputime using GAMS grows so fast with the number of buses that the parallel scheme takes less cputime in total than with the serial scheme. In fact, as we will show in the following subsections, with an efficient OPF implementation the true efficiencies would be less than one.

3.4.2 Estimated cputimes based on interpolation. We used the cputime data from [14, Tables 3 and 7] to estimate a cputime versus number of buses characteristic for an efficient serial OPF implementation. A least squares fit to this data yielded:

$$T_{\text{cpu}} = 0.0344 N^{1.1112}, \tag{6}$$

where T_{cpu} is the cputime and N is the number of buses. We then used this relationship to estimate the cputime, speed-up, and efficiency for an efficient implementation of our parallel algorithm.

Table 2 shows the estimated cputime for the base-case for the serial and parallel implementations of the OPF algorithm, using the relationship in (6). Naturally, these estimates are very rough since, as we observed, there are other issues besides the number of buses that affect cputime. To estimate the parallel cputime we made the additional assumptions that:

- there was one processor per region,
- the first iteration of the parallel algorithm takes time predicted by (6), and,
- the subsequent iterations take about 20% of the time for the first iteration.

The estimates of the parallel cputime are optimistic for distributed implementation in that they ignore overheads of an actual distributed implementation, such as idle time and communication between regions. The estimates should therefore be only taken as indicative of possible performance; however, as we argued in the introduction, the speed-up and efficiency may be not as significant as the other advantages of a distributed implementation.

3.4.3 Estimated efficiencies. The estimated efficiencies for the larger systems are between about 60 and 75%, based on the 0.03 per unit tie-line mismatch criterion. As shown in figures 7 and 8, almost all of the *potential* production cost savings are achieved within 3 or 4 iterations. If we terminate after 3 or 4 iterations, then the efficiency improves to 70 or 80%, with production costs still within 0.1% of optimal. Several of our test systems are of modest size and the ratio of the number of border to core variables is large. We expect better performance for larger systems with lower ratios of border to core variables.

4 Conclusion and Future Study

We have demonstrated an effective parallel algorithm for the OPF problem that is suitable for distributed implementation. We believe that our implementation is the first demonstration of the viability of large-scale distributed OPF and we have argued that a distributed approach is the only practical way to achieve large-scale on-line OPF. Our next step is to incorporate a state-of-the-art serial OPF algorithm such as described in [14] into a distributed environment to obtain more definitive speed-up and efficiency estimates.

We will also explore ways to improve convergence of the basic algorithm: one possibility is to use a dummy generator and a dummy load to represent each border bus, instead of two dummy generators. Finally, to be of use to the industry, we must incorporate contingency constraints into the implementation. We plan to begin by considering outages of tie-lines.

System	50-Bus	78-Bus	108-Bus	238-Bus	360-Bus	376-Bus	753-Bus
Cputime (Serial)	2.7	4.4	6.3	15.1	23.9	25.1	54.3
Cputime (Parallel)	6.6	6.3	8.5	22.1	37.3	42.2	83.9
Speed-Up	0.8	2.1	3.0	1.4	1.9	1.2	2.6
Efficiency (%)	41.0	69.3	74.4	68.4	64.1	59.5	64.7

Table 2: Estimated cputime, speed-up, and efficiency for parallel OPF.

Appendix: Sufficient Conditions for Convergence of Distributed Algorithm

We will describe some sufficient conditions for convergence of the iterative scheme (3)–(4). Assume that the sets $\{y_a : \exists x \text{ such that } (x, y_a) \in A\}$ and $\{y_b : \exists z, \text{ such that } (y_b, z) \in B\}$ are each closed and convex. Consider the following functions:

$$J_a(y_a) = \min_x \{c_a(x) : (x, y_a) \in A\}, \quad (7)$$

$$J_b(y_b) = \min_z \{c_b(z) : (y_a, z) \in B\}. \quad (8)$$

Assume that J_a and J_b are convex and differentiable. Suppose that $\alpha < 2\gamma < \beta$. Then, by Theorem 15 of [4] (see comment (iii) for Algorithm 14), the iteration (3)–(4) converges to the solution of (2). These conditions on the functions J_a and J_b are too strict to be directly applicable to the OPF problem, since we cannot in general prove that the cost of the OPF solution is a convex function of the border variables. Nevertheless, we found empirically that convergence was reliable with the choice:

$$\alpha = \frac{1}{2}\beta = \gamma. \quad (9)$$

For most test cases, we tuned the parameters by adjusting α and defining the other parameters according to (9).

References

[1] J. Batut and A. Renaud. Daily generation scheduling optimization with transmission constraints: A new class of algorithms. *IEEE Transactions on Power Apparatus and Systems*, 7(3):982–989, August 1992.

[2] Anthony Brooke, David Kendrick, and Alexander Meeraus. *GAMS User's Guide*. The Scientific Press, Redwood City, CA, 1990.

[3] Guy Cohen. Auxiliary problem principle and decomposition of optimization problems. *Journal of Optimization Theory and Applications*, 32(3):277–305, November 1980.

[4] Guy Cohen and Dao Li Zhu. Decomposition coordination methods in large scale optimization problems: The nondifferentiable case and the use of augmented Lagrangians. In J. B. Cruz, editor, *Advances in Large Scale Systems, Volume 1*, pages 203–266. JAI Press Inc., 1984.

[5] Bruce Hight. 2 electric utilities are investigated. *The Austin-American Statesman*, page D1 and D8, November 4, 1995.

[6] G. Huang and W. Ongsakul. Speedup and synchronisation overhead analysis of Gauss-Seidel type algorithms on a Sequent balance machine. *IEE Proceedings, Part C*, 141(5):437–444, September 1994.

[7] Gabriel Kron. *Diakoptics: The Piecewise Solution of Large-Scale Systems*. Macdonald, London, 1963.

[8] Shyan-Lung Lin and J. E. Van Ness. Parallel solution of sparse algebraic equations. *IEEE Transactions on Power Systems*, 9(2):743–749, May 1994.

[9] Tsutomu Oyama, Tatsuya Kitahara, and Yasuo Serizawa. Parallel processing for power system analysis using band matrix. *IEEE Transactions on Power Systems*, 5(3):1010–1016, August 1990.

[10] B. Stott, O. Alsac, and A. J. Monticelli. Security analysis and optimization. *Proceedings of the IEEE*, 75(12):1623–1644, December 1987.

[11] Task Force of the Computer and Analytical Methods Subcommittee of the Power Systems engineering Committee. Parallel processing in power systems computation. *IEEE Transactions on Power Systems*, 7(2):629–637, May 1992.

[12] Hal R. Varian. *Microeconomic Analysis*. W. W. Norton and Company, New York, 3rd edition, 1991.

[13] Allen J. Wood and Bruce F. Wollenberg. *Power Generation, Operation, and Control*. Wiley, New York, second edition, 1996.

[14] Yu-Chi Wu, Atif S. Debs, and Roy E. Marsten. A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows. *IEEE Transactions on Power Systems*, 9(2):876–883, May 1994.

Balho H. Kim was born in South Korea. He received his BSEE from Seoul National University, and his MSEE from the University of Texas at Austin. Currently, he is a Ph.D candidate in the Department of Electrical and Computer Engineering at the University of Texas at Austin.

Ross Baldick received his B.Sc. and B.E. from the University of Sydney, Australia and his M.S. and Ph.D. from the University of California, Berkeley. He is currently an assistant professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin.

Acknowledgment This work was funded in part by the National Science Foundation under grant ECS-9496193. We would like to thank Dr. Craig Chase of the Department of Electrical and Computer Engineering at the University of Texas at Austin for comments during the course of this work. We would also like to thank Dr. Alex Papalexopoulos and his colleagues at Pacific Gas and Electric for their support and encouragement during this work.