一种快速求解大规模安全约束最优潮流的多核并行方法

傅志生，白晓清，李佩杰，韦 化

（广西电力系统最优化与节能技术重点实验室(广西大学)，广西 南宁 530004）

摘要：针对传统安全约束最优潮流计算方法占用内存大、计算时间长的问题，提出了一种多核并行安全约束最优潮流算法。由正常运行状态检验预想故障状态，选取少量严重故障逐步修正正常运行状态，直到所有状态都得到满足。多核并行技术用于实现算法从高到低层次的高效率整体并行，以提高计算速度。其中，采用有向无环图并行数值分解算法实现正常运行状态修正过程的细粒度并行，并根据算法特性，直接实现预想故障状态检验过程的粗粒度并行。某省-3301 节点系统(预设 693 个故障)等 3 个系统的计算结果表明：随着系统规模的扩大，所提方法占用内存不足传统方法的 1%，且计算速度可快于传统方法三个数量级以上。

关键词：安全约束最优潮流；快速计算；多核并行；故障过滤；*N*−1 分析

A high-speed multi-core parallel method for solving large-scale security constrained OPF

FU Zhisheng, BAI Xiaoqing, LI Peijie, WEI Hua

(Guangxi Key Laboratory of Power System Optimization and Energy Technology (Guangxi University), Nanning 530004, China)

Abstract: The traditional Security Constrained Optimal Power Flow (SCOPF) methods are limited for the realistic systems due to the high memory consumption and low computation efficiency. A high-speed multi-core parallel algorithm is proposed to overcome these disadvantages. The contingencies are checked using the pre-contingency configuration at first. Afterward, a few serious contingencies are extracted to correct gradually the normal operation status until all conditions are satisfied. The multi-core parallel technology is applied to the algorithm from both high and low levels for obtaining high efficient parallel computation. The high-level parallels are used during the contingencies checking procedure, and fine-grained parallels are used during the pre-contingency configuration correction process based on the directed acyclic graph (DAG) parallel numerical decomposition algorithm. The proposed method has been successfully tested on three systems, which include a realistic provincial 3301-bus system with 693 contingencies. The numerical results indicate that the proposed method is suitable for solving large SCOPF problems because of fast and steady computation time. Furthermore, the memory consumptions of the proposed method are less 1% than that of traditional method, and the computation speeds are three orders of magnitude faster than that of the traditional ones as the system scale enlarges.

This work is supported by National Program on Key Basic Research Project (973 Program) (No. 2013CB228205) and National Natural Science Foundation of China (No. 51367004 and No. 51407036).

Key words: SCOPF; high-speed; multi-core parallel technology; contingency filtering; *N*−1 analysis

中图分类号： TM74 文献标识码：A 文章编号： 1674-3415(2015)03-0029-09

0 引言

以风电、光伏发电为代表的新能源具有间歇性、随机性的特点，其大量接入为电力系统稳定运行带来了新的挑战[1-3]。2011 年以来，西北甘肃风电基地、华北张北风电基地相继发生大规模风电脱网事故，研究电力系统故障预防及故障后应急处理技术，可为避免类似事件提供理论支撑。

安全约束最优潮流[4-5](Security-Constrained Optimal Power Plow，SCOPF)同时考虑正常状态和预想故障状态的约束条件，理论上能够有效保障电网安全。然而 SCOPF 通常考虑的预想故障数目众多，其求解过程存在内存消耗大和计算时间长的缺陷，造成其实际应用的限制。

基金项目：国家重点基础研究发展计划项目(973 项目)(2013CB228205)；国家自然科学基金项目(51367004, 51407036)

考虑 n 个预想故障的 SCOPF 问题，采用传统的联立方法[6]求解时，变量数相当于 OPF 问题的 $(n+1)$ 倍，相对应的雅可比和海森矩阵规模为 $(n+1)^2$ 倍。当 n 很大时，SCOPF 问题规模巨大，难以求解。目前主要有三种求解思路。

1) Benders 分解法[7-8]：该方法将 SCOPF 问题分解成为一个主问题和若干个子问题并交替迭代求解，直到主问题和子问题都得到可行解。主问题和子问题规模较小，且子问题可并行[7]处理，这很大程度上降低了原问题的求解难度。相对于传统联立方法，Benders 分解法大大提高了计算速度，扩大了求解规模。然而，该方法理论上要求可行域为凸，但交流模型的 SCOPF 难以保证可行域为凸[9]。

2) 网络压缩方法[10-11]：该方法考虑故障发生后，系统只有局部受到影响，设立一个受影响与否的判断标准，将故障状态下的网络节点拆分为受影响节点和不受影响节点两部分，计算过程保持后者的参数不变，只考虑受影响节点，从而达到缩减故障后变量数的目的。然而，该方法判定准则存在近似性，在计算中固定一些参数，或多或少会对计算结果产生一定的不利影响，这通常会导致计算结果出现轻微越界[12]。

3) 有效故障筛选类方法[9,11-14]：实际电力系统的 SCOPF 问题，预想故障集中可能存在一些故障对于系统的运行状态是没有影响的。基于这一特征，提出了有效故障筛选方法。在 SCOPF 计算过程中只考虑有效故障，从而缩小 SCOPF 问题的求解规模。然而目前尚无明确的故障筛选准则[8]。本文方法能够保证所有的预想故障状态都得到满足，为有效故障筛选类方法。

在文献[9]的基础上，本文提出了一种快速求解大规模 SCOPF 问题的多核并行方法。利用正常运行状态检验预想故障状态，然后选取少量严重故障逐步修正正常运行状态，并引入 *N*-1 分析、故障过滤和多核并行技术，提出三种改进方法。以上两个过程交替迭代，直到所有状态都得到满足。IEEE-118、IEEE-300 及某省-3301 节点系统多故障状态的计算结果表明：所提方法有效地降低了 SCOPF 问题的求解难度，能够在不增加计算机硬件投入的前提下快速求解大规模 SCOPF 问题，具有广泛的应用前景。

1 SCOPF 问题描述

在 OPF 模型基础上扩展预想故障约束条件构成如下 SCOPF 模型：

$$\begin{aligned} \min\ & f(\boldsymbol{x}_0) \\ \text{s.t.}\ & \boldsymbol{h}_k(\boldsymbol{x}_k)=0 \\ & \underline{\boldsymbol{g}}_k \le \boldsymbol{g}_k(\boldsymbol{x}_k) \le \overline{\boldsymbol{g}}_k \\ & |u_k-u_0| \le \Delta\overline{u}_k \\ & k \in C=\{0,1,2,\cdots,n\} \end{aligned} \tag{1}$$

式中：$\boldsymbol{C}$ 为所考虑状态的集合；$k=0$ 表示正常状态，$k\neq 0$ 表示预想故障状态；$\boldsymbol{x}_k \in R^n$ 为计算变量；$f(\boldsymbol{x}_0)$ 为目标函数；$\boldsymbol{h}_k(\boldsymbol{x}_k)$ 为正常状态和故障状态下的潮流方程；$\boldsymbol{g}_k(\boldsymbol{x}_k)$ 为正常状态和故障状态下的运行约束；$\overline{\boldsymbol{g}}_k$、$\underline{\boldsymbol{g}}_k$ 分别表示 $\boldsymbol{g}_k$ 的上、下限；$\boldsymbol{u}_k \in \boldsymbol{x}_k$，$\boldsymbol{u}_k \in R^m$ 表示控制变量(发电机出力，无功投入、变比等)。

$\Delta\overline{\boldsymbol{u}}_k$ 表示当第 k 个故障发生后，控制变量允许的最大调整量，由发电机等设备的特性以及故障发生后允许的最大调整时间共同决定。假设发电机等设备的爬坡率为 $\mathrm{d}\boldsymbol{u}/\mathrm{d}t$，事故 k 发生后允许的最大调整时间为 t，则 $\Delta\overline{\boldsymbol{u}}_k = t\times \mathrm{d}\boldsymbol{u}/\mathrm{d}t$。

1.1 目标函数

不失一般性，本文采用正常运行状态发电费用最少作为目标函数，表述为

$$f(\boldsymbol{x}_0)=\sum_{i\in S_G}\left(a_{2i}P_{Gi(0)}^2+a_{1i}P_{Gi(0)}+a_{0i}\right) \tag{2}$$

式中：$P_{Gi(0)}$ 为第 i 台发电机正常状态下的有功出力；S_G 为所有发电机集合；a_{2i}、a_{1i}、a_{0i} 为发电机费用特性曲线参数。

1.2 等式约束

等式约束为正常状态和故障状态功率平衡方程，如式(3)。

$$\boldsymbol{h}_k(\boldsymbol{x}_k)=\begin{cases} \boldsymbol{P}_{g(k)}-\boldsymbol{P}_d-\boldsymbol{P}(\boldsymbol{v},\boldsymbol{\theta})=0 \\ \boldsymbol{Q}_{g(k)}-\boldsymbol{Q}_d-\boldsymbol{Q}(\boldsymbol{v},\boldsymbol{\theta})=0 \end{cases} \tag{3}$$

式中：k 与式(1)相同；$\boldsymbol{P}_g$、$\boldsymbol{Q}_g$ 分别为节点注入有功、无功矢量；$\boldsymbol{P}_d$、$\boldsymbol{Q}_d$ 分别表示节点输出有功、无功矢量；$\boldsymbol{v}$、$\boldsymbol{\theta}$ 分别表示电压幅值、相角矢量。

1.3 不等式约束

不等式约束包含正常状态和故障状态设备的物理特性约束以及电能质量约束。

$$\boldsymbol{g}_k(\boldsymbol{x}_k)=\begin{bmatrix} \boldsymbol{P}_{G(k)} \\ \boldsymbol{Q}_{R(k)} \\ \boldsymbol{v}_{(k)} \\ \boldsymbol{S}_{ij(k)} \end{bmatrix};\ \underline{\boldsymbol{g}}_k=\begin{bmatrix} \underline{\boldsymbol{P}}_G \\ \underline{\boldsymbol{Q}}_R \\ \underline{\boldsymbol{v}} \\ \underline{\boldsymbol{S}}_{ij} \end{bmatrix};\ \overline{\boldsymbol{g}}_k=\begin{bmatrix} \overline{\boldsymbol{P}}_G \\ \overline{\boldsymbol{Q}}_R \\ \overline{\boldsymbol{v}} \\ \overline{\boldsymbol{S}}_{ij} \end{bmatrix} \tag{4}$$

式中：k 与式(1)相同；$\boldsymbol{P}_{\mathrm{G}}$ 为发电机有功出力矢量；$\boldsymbol{Q}_{\mathrm{R}}$ 为无功电源点注入的无功矢量；$\boldsymbol{v}$ 为节点电压幅值矢量；$\boldsymbol{S}_{ij}$ 为支路视在功率矢量。

2　求解方法分析

2.1 基本思想

假设系统在正常运行状态 $\boldsymbol{x}_0$ 下，发生单个故障 k，若能够通过校正控制变量 $\boldsymbol{u}_k$ 使新的运行状态 $\boldsymbol{x}_k$ 满足式(1)中的等式和不等式约束，则故障 k 为 $\boldsymbol{x}_0$ 下的可控故障，反之为不可控故障，相应的集合称为可控故障集和不可控故障集。

大规模 SCOPF 问题中可能只需要考虑不可控的故障，就能得到问题式(1)的最优解。而实际的多预想故障系统中不可控故障往往只是预想故障集 $\boldsymbol{C}$ 的一小部分。也就是说可能只需要考虑 $\boldsymbol{C}$ 中的一小部分故障，便能够得到大规模 SCOPF 问题的最优解。

对预想故障状态逐个求解故障后 OPF(Post Contingency Optimal Power Flow，PCOPF)以判断其是否可控。PCOPF 模型描述如式(5)。

$$\begin{aligned} \min \quad & f = \boldsymbol{e}^{\mathrm{T}} \times \boldsymbol{l}_k \\ \text{s.t.} \quad & \boldsymbol{h}_k(\boldsymbol{x}_k) = 0 \\ & \underline{\boldsymbol{g}}_k \le \boldsymbol{g}_k(\boldsymbol{x}_k) \le \overline{\boldsymbol{g}}_k \\ & |\boldsymbol{u}_k - \boldsymbol{u}_0| \le \Delta\overline{\boldsymbol{u}}_k + \boldsymbol{l}_k \end{aligned} \tag{5}$$

式中：$\boldsymbol{e} = [1, \cdots, 1]^{\mathrm{T}}$；$\boldsymbol{u}_0$ 为正常状态下的控制量，是已知量；k 是一个要检查的故障；$\boldsymbol{l}_k \ge 0$ 为新引入的松弛变量。若 $f = 0$，则存在 $\boldsymbol{x}_k$ 满足式(1)中的等式和不等式约束，故障 k 为状态 $\boldsymbol{x}_0$ 下的可控故障；若 $f > 0$，则故障 k 为状态 $\boldsymbol{x}_0$ 下的不可控故障。

2.2 基本方法

基于 2.1 节所述求解思想，建立基本方法，其迭代过程如下。

1) 计算含故障集 $\boldsymbol{C}_{\mathrm{S}}$ (初始设置 $\boldsymbol{C}_{\mathrm{S}}$ 为空集 ϕ)的 SCOPF 问题，得到系统最优正常运行状态 $\boldsymbol{x}_0$。

2) 在 $\boldsymbol{x}_0$ 处对故障集 $\boldsymbol{C}_{\mathrm{NS}} = \{k \in \boldsymbol{C},\ k \notin \boldsymbol{C}_{\mathrm{S}}\}$ 中故障逐个进行 PCOPF 计算，得到不可控故障集合 $\boldsymbol{C}_{\mathrm{E}}$。若 $\boldsymbol{C}_{\mathrm{E}}$ 为 ϕ，计算结束，$\boldsymbol{x}_0$ 为最优解；否则 $\boldsymbol{C}_{\mathrm{S}} \leftarrow \boldsymbol{C}_{\mathrm{S}} \bigcup \boldsymbol{C}_{\mathrm{E}}$ 转 1)。

基本方法的流程如图 1 所示。1)、2)可用内点法求解。首次迭代 $\boldsymbol{C}_{\mathrm{S}}$ 为 ϕ，此时 1) 相当于 OPF 计算。2) 中单次 PCOPF 计算的工作量与 OPF 计算基本相当。

方法收敛的最坏情况为每次交替只找到一个不可控故障，其交替次数为最优解处不可控故障数。通过对大量实际电力系统的计算，发现不可控故障只占预想故障的一小部分，且首次交替能够找出大部分不可控故障，故方法能在 3~5 次交替迭代后收敛。

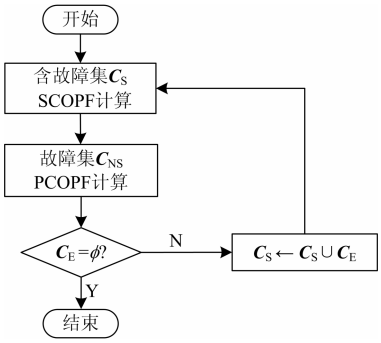

图 1 基本方法流程图

Fig. 1 Flowchart of basic method

2.3 改进方法 1：N−1 分析

基本方法的步骤 2)，计算量依然很大，对于 n 个预想故障的 SCOPF 问题，每次交替过程其计算量大约相当于计算 n 次 OPF。减少 PCOPF 的计算次数，能够进一步提高计算速度。因此，在运行状态 $\boldsymbol{x}_0$ 处，预先使用 PF 计算对 $\boldsymbol{C}_{\mathrm{NS}}$ 进行 N−1 分析，对于 N−1 不通过的故障再利用 PCOPF 进行检查，以减少 PCOPF 的计算次数，从而提升计算速度。

引入 N−1 分析改进基本方法，形成改进方法 1，其流程如图 2 所示。

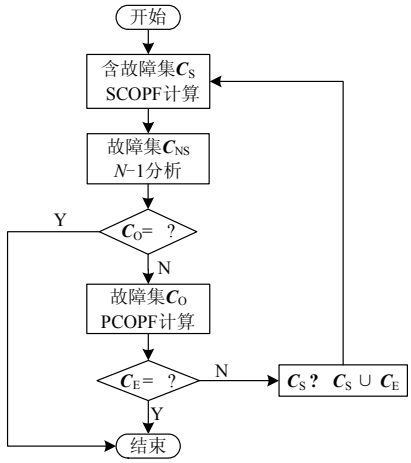

图 2 改进方法 1 流程图

Fig. 2 Flowchart of improvement method 1

改进方法 1 步骤如下：

1) 计算含 $\boldsymbol{C}_{\mathrm{S}}$ 的 SCOPF 问题，得到最优运行状态 $\boldsymbol{x}_0$。

2) 在 $\boldsymbol{x}_0$ 处对故障 $k \in \boldsymbol{C}_{\mathrm{NS}}$ 逐个进行 N−1 分析，得到越界故障集 $\boldsymbol{C}_{\mathrm{O}}$。若 $\boldsymbol{C}_{\mathrm{O}}$ 为 ϕ，计算结束，$\boldsymbol{x}_0$ 为最优解。

3) 在 $\boldsymbol{x}_0$ 处对故障 $k \in \boldsymbol{C}_{\mathrm{O}}$ 逐个进行 PCOPF 计算，得到不可控故障集 $\boldsymbol{C}_{\mathrm{E}}$。若 $\boldsymbol{C}_{\mathrm{E}}$ 为 ϕ，计算结束，$\boldsymbol{x}_0$ 为最优解；否则 $\boldsymbol{C}_{\mathrm{S}} \leftarrow \boldsymbol{C}_{\mathrm{S}} \bigcup \boldsymbol{C}_{\mathrm{E}}$ 转 1)。

这一步改进基于以下两点：

① PF 的计算量远小于 PCOPF，利用 PF 在 PCOPF 前进行一次预筛选，可能提升计算速度。

② 通过 *N*-1 分析的故障在计算 PCOPF 模型时其目标值一定为零。因为当 $\boldsymbol{u}_k - \boldsymbol{u}_0 = 0$ 时，PF 计算得到的解一定满足(5)中等式约束，若 *N*-1 检查通过，则表示不等式约束也得到满足，存在 $\boldsymbol{x}_k$ 使得式(5)中目标函数值为零，即通过 *N*-1 分析的故障为可控故障。

2.4 改进方法 2：故障过滤

在实际系统中的越界故障之间存在严重程度的差别。往往更严重的故障对于其他故障存在一种“伞”[15]关系，即只需要满足更严重的故障便能够保证系统的安全稳定，而更严重的故障只是越界故障的一小部分。为了提高计算速度，在改进方法 1 的基础上引入故障过滤技术对越界故障进行摘选识别，缩减进行 PCOPF 计算的越界故障数。

通过对预想故障 *N*-1 分析得出其满足网络基尔霍夫定律的运行参数，然后用所得参数推导其越界量的大小，以越界量大小作为越界性能指标，根据这一指标提出了一种主导故障选择方法。对于越界故障状态的同一个控制变量，认为越界程度越大其重新调整到功率平衡状态的难度越大，选取越界程度最大的故障作为主导故障。

具体做法如下：

1) 定义 *N*-1 分析中无解故障为主导故障的一部分。

2) 对于 *N*-1 分析过程有解的越界故障，选取越界性能指标最大的做为主导故障一部分。

由 1)和 2)做描述所选择出的故障为主导故障集合。单个故障 *k* 的越界性能指标定义为

$$\boldsymbol{M}\boldsymbol{g}_k^r = \begin{cases} 0 & \underline{\boldsymbol{g}}^r \le \boldsymbol{g}_k^r \le \overline{\boldsymbol{g}}^r \\ \underline{\boldsymbol{g}}^r - \boldsymbol{g}_k^r & \boldsymbol{g}_k^r < \underline{\boldsymbol{g}}^r \\ \boldsymbol{g}_k^r - \overline{\boldsymbol{g}}^r & \boldsymbol{g}_k^r > \overline{\boldsymbol{g}}^r \end{cases} \tag{6}$$

式中：$\boldsymbol{g}_k$ 同式(4)中所描述；*r* 表示不等式约束位置。根据预想故障的越界性能指标，从中选择出值最大的作为主导故障。存在故障状态 *j*，使得任意故障状态 *k* 满足：$\boldsymbol{M}\boldsymbol{g}_j^r \ge \boldsymbol{M}\boldsymbol{g}_k^r$。于是选择 *j* 作为主导故障。

改进方法 1 中引入故障过滤技术，形成改进方法 2，其流程如图 3 所示，步骤如下。

1) 计算含 $\boldsymbol{C}_{\mathrm{S}}$ 的 SCOPF 问题，得到最优运行状态 $\boldsymbol{x}_0$。

2) 在 $\boldsymbol{x}_0$ 处对故障 $k \in \boldsymbol{C}_{\mathrm{NS}}$ 逐个进行 *N*-1 分析，得到越界故障集 $\boldsymbol{C}_{\mathrm{O}}$。若 $\boldsymbol{C}_{\mathrm{O}}$ 为 ϕ，则 $\boldsymbol{x}_0$ 为最优解，计算结束。

3) 对 $\boldsymbol{C}_{\mathrm{O}}$ 进行故障过滤，得到主导故障集 $\boldsymbol{C}_{\mathrm{D}}$ 和被主导故障集 $\boldsymbol{C}_{\mathrm{ND}}$。

4) 在 $\boldsymbol{x}_0$ 处对故障 $k \in \boldsymbol{C}_{\mathrm{D}}$ 逐个进行 PCOPF 计算，得到不可控故障集 $\boldsymbol{C}_{\mathrm{E1}}$。若 $\boldsymbol{C}_{\mathrm{E1}}$ 不为 ϕ，则 $\boldsymbol{C}_{\mathrm{S}} \leftarrow \boldsymbol{C}_{\mathrm{S}} \bigcup \boldsymbol{C}_{\mathrm{E1}}$ 转 1)。

5) 若 $\boldsymbol{C}_{\mathrm{ND}}$ 为 ϕ，则 $\boldsymbol{x}_0$ 为最优解，计算结束。

6) 在 $\boldsymbol{x}_0$ 处对故障 $k \in \boldsymbol{C}_{\mathrm{ND}}$ 逐个进行 PCOPF 计算，得到不可控故障集 $\boldsymbol{C}_{\mathrm{E2}}$。若 $\boldsymbol{C}_{\mathrm{E2}}$ 为 ϕ，则 $\boldsymbol{x}_0$ 为最优解，计算终止；否则 $\boldsymbol{C}_{\mathrm{S}} \leftarrow \boldsymbol{C}_{\mathrm{S}} \bigcup \boldsymbol{C}_{\mathrm{E2}}$，转 1)。

图 3 虚线框保证了所得结果能够满足所有故障状态。当 $\boldsymbol{C}_{\mathrm{E1}}$ 为 ϕ 时，对 $\boldsymbol{C}_{\mathrm{ND}}$ 进行 PCOPF 计算得到 $\boldsymbol{C}_{\mathrm{E2}}$，$\boldsymbol{C}_{\mathrm{E1}}$ 不为 ϕ 时无需对 $\boldsymbol{C}_{\mathrm{ND}}$ 进行相应的计算，这样既可以减少参与 PCOPF 计算的越界故障数，又能保证整体故障得到满足。

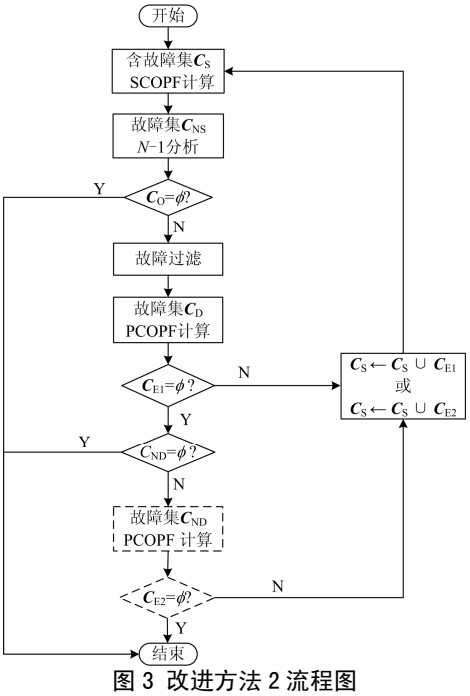

图 3 改进方法 2 流程图

Fig. 3 Flowchart of improvement method 2

2.5 改进方法 3：多核并行计算

多核并行利用多核处理器，能在不增加硬件投入的情况下，提高计算速度。但如果仍采用串行算法，计算速度并不会因为采用多核处理器而自动大幅提高。

图 3 改进方法 2 是一个串行的计算流程，方法

计算量集中于步骤 1)、2)、4)、6)。在多核计算机上，通过多线程编程，实现步骤 1)、2)、4)、6)的并行处理，可充分利用多核处理器的计算能力，提高方法整体的计算速度。

近年来多核处理器技术得到快速发展，核间通信带宽大幅增加，延时明显减少，为细粒度并行创造了有利条件。本文采用有向无环图[16](Directed Acyclic Graph, DAG)并行算法对步骤 1)中计算耗时最大的部分——Cholesky 分解进行细粒度解耦并行。

使用对称正定矩阵 $\boldsymbol{A}$ 的变换为例以说明 Cholesky 分解过程。$\boldsymbol{A}$ 的 Cholesky 分解描述：存在一个主对角元全为正数的下三角矩阵 $\boldsymbol{L}$，使得 $\mathbf{A}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$。

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1n}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \cdots & \boldsymbol{A}_{2n}\\ \vdots & \vdots & & \vdots\\ \boldsymbol{A}_{n1} & \boldsymbol{A}_{n2} & \cdots & \boldsymbol{A}_{nn}\end{bmatrix} \tag{8}$$

由于 $\boldsymbol{A}$ 为对称矩阵，可以仅对其下三角部分进行讨论，如式(8)中三角框内部分所示，A_{ij} 为方阵结构，可以推导出

$$\begin{cases}\boldsymbol{A}_{i1}=\boldsymbol{L}_{i1}\boldsymbol{L}_{11}^{\mathrm{T}}\\ \boldsymbol{A}_{ir}=\sum_{k=1}^{r}\boldsymbol{L}_{ik}\boldsymbol{L}_{rk}^{\mathrm{T}}\end{cases}\Rightarrow\begin{cases}\boldsymbol{L}_{i1}=\boldsymbol{A}_{i1}(\boldsymbol{L}_{11}^{\mathrm{T}})^{-1}\\ \boldsymbol{L}_{ir}=(\boldsymbol{A}_{ir}-\sum_{k=1}^{r-1}\boldsymbol{L}_{ik}\boldsymbol{L}_{rk}^{\mathrm{T}})(\boldsymbol{L}_{rr}^{\mathrm{T}})^{-1}\end{cases} \tag{9}$$

矩阵 $\boldsymbol{A}$ 块状结构排列的 Cholesky 分解流程为

For　j=1:n−1

Step1：$\boldsymbol{L}_{jj}=\mathrm{Cholesky}(\boldsymbol{A}_{jj})$

Step2：For　i=j+1:n

$\boldsymbol{L}_{ij}=\boldsymbol{A}_{ij}(\boldsymbol{L}_{jj}^{\mathrm{T}})^{-1}$

End

Step3：For k=j+1:n

For i=k:n

$\boldsymbol{A}_{ik}\leftarrow\boldsymbol{A}_{ik}-\boldsymbol{L}_{ij}\boldsymbol{L}_{kj}^{\mathrm{T}}$

End

End

End

Step4：$\boldsymbol{L}_{nn}=\mathrm{Cholesky}(\boldsymbol{A}_{nn})$

图 4 为 4×4 分块矩阵 Cholesky 分解过程。以此为例，说明进行一次 Step1~Step4 运算过程，图中 $\boldsymbol{A}_{ik}^{j}$ 上标表示 $\boldsymbol{A}_{ik}$ 第 j 次修正后的值。

在 Cholesky 分解流程中 Step2 和 Step3 循环内部计算之间相互独立，可以进行并行运算。其并行计算过程如图 5 所示(由左往右进行分解)，图中方框表示计算任务。这一并行结构有一个很明显的缺陷：每一个步骤的计算都依赖前一步的完成，且各个步骤之间可并行度不一致，这导致多核并行过程中CPU核利用率不高，采用DAG能够有效的克服这一缺陷。

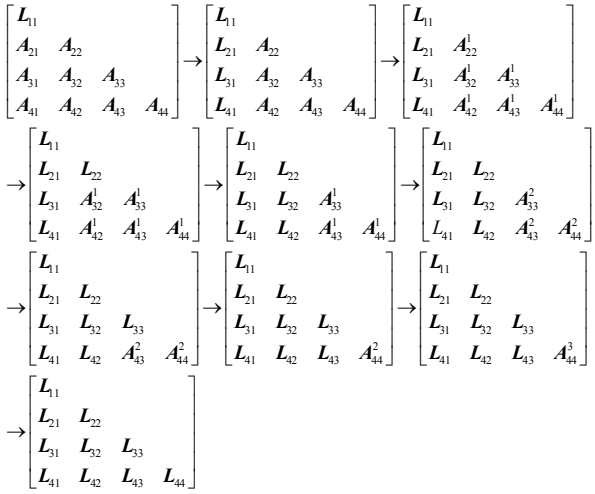

图 4 4×4 分块矩阵 Cholesky 分解

Fig. 4 4×4 block matrices Cholesky factorization

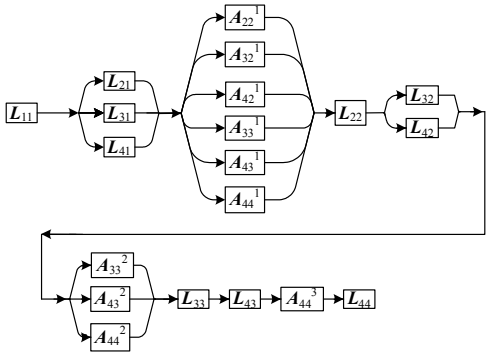

图 5 4×4 分块矩阵 Cholesky 分解的传统并行结构

Fig. 5 Cholesky parallel structure of 4×4 block matrices

图 6 为相应的 DAG 并行结构。方框表示计算任务，箭头表示任务间依赖关系，箭头指向表示要求先完成前者才能计算后者。各任务只要不违背任务间的依赖关系，可乱序执行。设计算法时，标识任务的依赖条件是否完成，不停地主动选取满足条件的任务进行并行计算，并更新任务标记，直至完成全部任务的计算。这样可改善多核负载不平衡，提高并行过程中 CPU 核的利用率。

步骤 2)、4)、6)中故障状态之间不存在耦合关系，可直接进行并行计算。

于是，改进方法 2 中引入多核并行计算技术，形成改进方法 3，流程如图 7 所示。其中：

① $\boldsymbol{C}_{\mathrm{NS1}}\cup\boldsymbol{C}_{\mathrm{NS2}}\cup\cdots\cup\boldsymbol{C}_{\mathrm{NS}n}=\boldsymbol{C}_{\mathrm{NS}}$

② $\boldsymbol{C}_{\mathrm{D1}}\cup\boldsymbol{C}_{\mathrm{D2}}\cup\cdots\cup\boldsymbol{C}_{\mathrm{D}n}=\boldsymbol{C}_{\mathrm{D}}$

③ $\boldsymbol{C}_{\mathrm{ND1}} \cup \boldsymbol{C}_{\mathrm{ND2}} \cup \cdots \cup \boldsymbol{C}_{\mathrm{NDn}} = \boldsymbol{C}_{\mathrm{ND}}$

④ $\mathrm{CPU}_1, \mathrm{CPU}_2, \cdots \mathrm{CPU}_n$ (参与并行的核)。

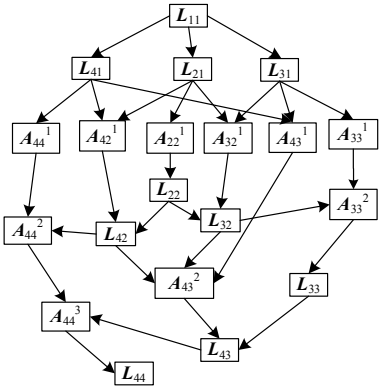

图 6 4×4 分块矩阵的 DAG 并行结构

Fig. 6 4×4 block matrices DAG parallel structure

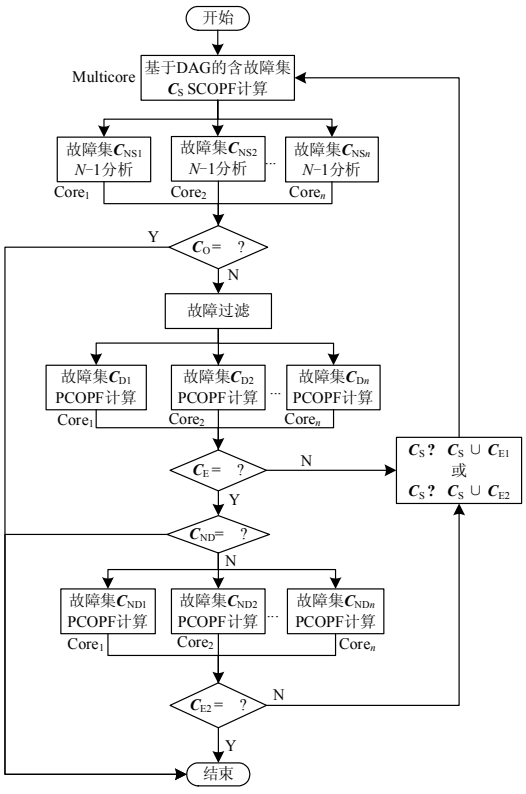

图 7 改进方法 3 流程图

Fig. 7 Flowchart of improvement method 3

3 数值结果

3.1 计算环境

对 IEEE-118，IEEE-300 以及某省-3301节点系统进行测试，系统参数如表 1、表 2 所示。为了便于比较计算过程的占用内存，某省-3301 系统仅设置了 693 个预想故障。

表 1 测试系统规模

Table 1 Scale of test system

测试系统	节点数	支路数	线路数	变压器数
IEEE-118	118	186	177	9
IEEE-300	300	409	302	107
某省-3301	3 301	3 740	2 032	1 708

表 2 设置故障数

Table 2 Set the number of contingencies

测试系统	IEEE-118	IEEE-300	某省-3301
线路故障数	163	253	335
变压器故障数	9	43	358
$\boldsymbol{C}$ 故障数	172	296	693

服务器参数。CPU 型号：AMD 皓龙 6166HE; CPU 主频：1.8 GHz；CPU 核数：12 核；内存：128 G。在此服务器上使用 Matlab2012b 编写程序实现所提方法。

3.2 计算结果

表 3 为改进方法 2 整体迭代情况。由表可知所提方法交替迭代次数不多(3~4 次)，且首次交替迭代一般能找出大部分不可控故障。

表 3 改进方法 2 迭代情况

Table 3 Iteration situation of improvement method 2

测试系统	交替迭代次数	C_S故障数	C_O故障数	C_D故障数	C_{E1}故障数	C_{E2}故障数
IEEE-118	1	0	172	33	1	-
	2	1	171	33	1	-
	3	2	170	32	0	0
IEEE-300	1	0	296	72	7	-
	2	7	289	67	1	-
	3	8	288	65	1	-
	4	9	287	67	0	0
某省-3301	1	0	134	41	5	-
	2	5	160	38	1	-
	3	6	242	38	0	0

比较越界故障集 $\boldsymbol{C}_{\mathrm{O}}$、主导故障集 $\boldsymbol{C}_{\mathrm{D}}$ 和不可控故障集 $\boldsymbol{C}_{\mathrm{E}}$ ($\boldsymbol{C}_{\mathrm{E1}}$、$\boldsymbol{C}_{\mathrm{E2}}$)，可知：主导故障数要远小于越界故障数，即故障过滤技术有效的提升了方法的计算速度；不可控故障数远小于主导故障数，即方法能够有效降低 SCOPF 问题的计算难度。

在 IEEE-118、IEEE-300 测试系统测试迭代过程中，故障数 $\boldsymbol{C}_{\mathrm{O}}$ (表 4)非常接近设置的整体故障 $\boldsymbol{C}$(表 2)，此时 N-1 分析并不能起到减少 PCOPF 计算次数的目的，反而导致计算过程中增加了 N-1 分析的耗时，故 IEEE-118 和 IEEE-300 系统改进方法 1 计算时间要略微长于基本方法；而某省-3301节点系统中，越界故障数 $\boldsymbol{C}_{\mathrm{O}}$ 远小于 $\boldsymbol{C}$，故其改进方法 1 计算速度要远快于基本方法。

表 4 给出了本文所提基本方法、改进方法 1、

改进方法 2、改进方法 3-12 核并行和传统方法的计算性能测试情况。表中 P_{Gi}、P_{Gi}^0 分别为所提方法和传统方法计算所得的发电机出力。由于传统方法中某省-3301 节点系统计算规模过于庞大，仅对其单次内点法迭代进行了测试，耗时大约为 12.5h，故表 5 中传统方法的目标函数值、表 6 中发电机出力比较值未列出。假设某省-3301 节点系统计算需要迭代 60 次，则需要近 1 个月的时间方能完成计算。

由目标函数和 $\max\dfrac{|(P_{Gi}-P_{Gi}^0)|}{P_{Gi}^0}$ 值可知，所提方法计算所得结果与传统方法相当，即其能够得到正确的计算结果。

表 4 计算性能比较

Table 4 Calculated performance comparison

测试方法	测试量	测试系统		
		IEEE-118	IEEE-300	某省-3301
传统求解	目标函数	196.6	128.1	--
	计算时间/s	101.7	654.2	约 27×10^5
	占用内存/M	635	1 235	125 639
基本方法	目标函数	196.6	128.1	243.3
	计算时间/s	108.5	569.5	14 636
	占用内存/M	379	420	594
	$\max\dfrac{\lvert(P_{Gi}-P_{Gi}^0)\rvert}{P_{Gi}^0}$	5.7×10^{-8}	1.3×10^{-4}	--
改进方法 1	计算时间/s	113.2	596.6	4 484
	占用内存/M	379	420	594
改进方法 2	计算时间/s	43.8	267.5	2 215
	占用内存/M	379	420	594
改进方法 3 -12 核	计算时间/s	9.69	48.3	308.5
	占用内存/M	382	425	604

分析占用内存值：随着系统规模的扩大，所提方法计算过程使用内存会低于传统方法的 1%。其中某省-3301 节点系统占用内存情况如图 8 所示，12 核并行情况下所提方法占用内存仅为传统方法的 0.481%(604/125639= 0.481%)。由此可知，所提方法可以降低 SCOPF 计算对于计算机硬件环境的依赖。

分析计算时间值：随着系统规模的扩大，所提方法计算速度可快于传统方法两个数量级以上。其中某省-3301 节点系统计算时间情况如图 9 所示，在 12 核并行的情况下，所提方法要比传统方法快超过 10^3 倍。由此可知，所提方法可以快速求解大规模 SCOPF 问题。

综合分析表 4 中各项值，可知所提方法能够在保证正确的计算结果的前提下，降低 SCOPF 计算对于硬件环境的依赖，并提升计算速度，易于工程应用。

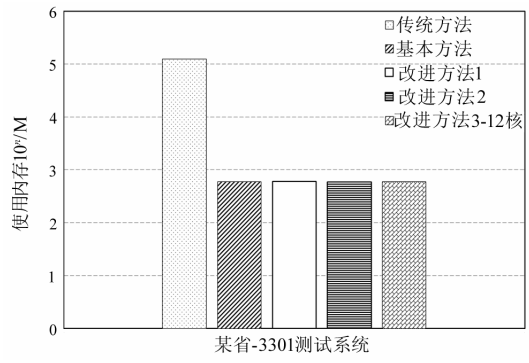

图 8 使用内存比较图(纵坐标取对数)

Fig. 8 Comparison of the memory consumption

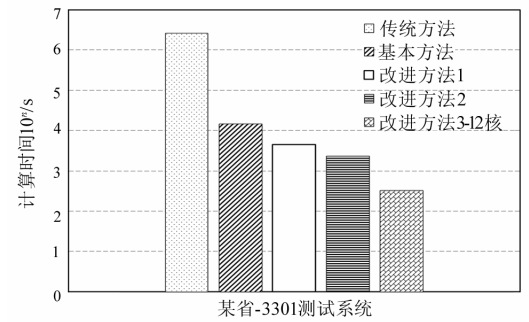

图 9 计算时间比较图(纵坐标取对数)

Fig. 9 Comparison of the computation time

3.3 并行计算结果

改进方法 3 完整的交替过程中，有四部分参与并行：步骤 1)、2)、4)、6)，即含 $\boldsymbol{C}_{\mathrm{S}}$ 的 SCOPF 的 DAG 并行、故障集 $\boldsymbol{C}_{\mathrm{NS}}$ 的 N-1 分析、故障集 $\boldsymbol{C}_{\mathrm{D}}$ 的 PCOPF 计算、故障集 $\boldsymbol{C}_{\mathrm{ND}}$ 的 PCOPF 计算。由于 $\boldsymbol{C}_{\mathrm{S}}$ 规模远小于整体故障集 $\boldsymbol{C}$，故讨论 N-1 分析计算时间可以用 $\boldsymbol{C}$ 近似 $\boldsymbol{C}_{\mathrm{NS}}$。故障集 $\boldsymbol{C}_{\mathrm{ND}}$ 的规模要大于 $\boldsymbol{C}_{\mathrm{D}}$，但 $\boldsymbol{C}_{\mathrm{ND}}$ 的计算次数要小于 $\boldsymbol{C}_{\mathrm{D}}$，两者对加速效果的影响都不可忽略。

为方便讨论并行效果，称含 $\boldsymbol{C}_{\mathrm{S}}$ 的 SCOPF 的计算为 SCOPF 模块，称故障集 $\boldsymbol{C}_{\mathrm{NS}}$ 的 N-1 分析为 N-1 模块，称故障集 $\boldsymbol{C}_{\mathrm{D}}$ 和 $\boldsymbol{C}_{\mathrm{ND}}$ 的 PCOPF 计算为 PCOPF 模块。

步骤 1)应用 DAG 并行数值分解算法进行并行处理，步骤 2)、4)、5)调用 Matlab 中的 parallel computing toolbox 工具箱中的 parfor 函数进行并行处理。

图 10 给出了 SCOPF 模块 DAG 加速比曲线，其中加速比定义为 Matlab 中的左除算法计算时间与基于 DAG 算法的并行计算时间的比值。加速比越大表明多核并行下的计算耗时越短。

图 11 给出了 N-1 分析模块的多核加速比曲线，图 12 给出了 PCOPF 模块的多核加速比曲线，图 13

为改进方法 3 整体加速情况曲线，其中加速比定义为串行计算时间与 parfor 并行计算时间的比值。

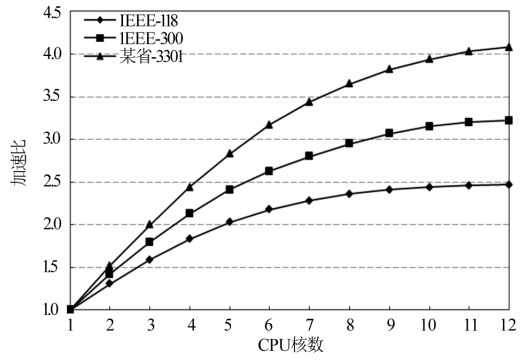

图 10 SCOPF 模块 DAG 加速比曲线

Fig. 10 DAG speedup of SCOPF module

SCOPF 模块 DAG 并行过程由于仅对计算量最大的“左除”部分进行了细粒度解耦并行，在核数较少时(1~6 核)多核并行效果非常明显，加速比曲线呈现近似线性关系。随着并行计算的核数的增加，串行部分占整体计算时间比重增加，加速比曲线逐渐平缓。

N-1 分析和 PCOPF 计算模块中，故障数目众多；并行处理以后，其并行颗粒大且通信量小。其加速比和并行核数呈现线性关系(图 11、图 12)。可以预测，若继续增加并行核数，该部分的计算速度将进一步提升。此外，PCOPF 计算模块要比 *N*-1 分析模块并行颗粒大，在相同核数的情况下前者加速比要明显大于后者。

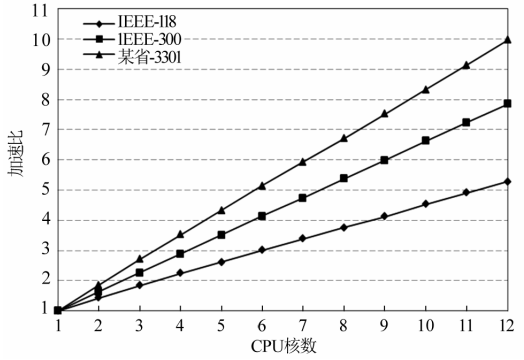

图 11 *N*-1 模块加速比曲线

Fig. 11 Speedup of *N*-1 module

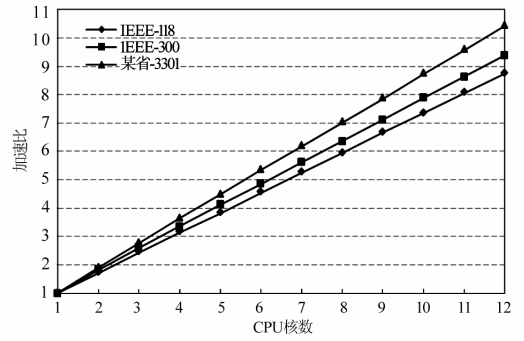

图 12 PCOPF 模块加速比曲线

Fig. 12 Speedup of PCOPF module

整体计算时间可以分为两部分：①SCOPF 模块、*N*-1 模块和 PCOPF 模块的计算时间；②串行处理的故障过滤和空集判断部分的计算时间。由于②计算量非常少，计算时间基本可以忽略，故整体加速性能，主要受 SCOPF 模块、*N*-1 模块和 PCOPF 模块加速性能的影响，其中 *N*-1 模块以及 PCOPF 模块占用及整体计算时间的绝大部分(图 14 为某省-3301 系统多核并行过程中各模块计算时间占整体计算时间比例情况)，由图 11 和图 12 可知 *N*-1 模块和 PCOPF 模块都具有良好的线性加速性能，因此整体加速比与 CPU 核数呈现出近似线性加速关系(图 13)，随着参与计算的 CPU 核数的增加，本文方法依然可以提升计算速度。

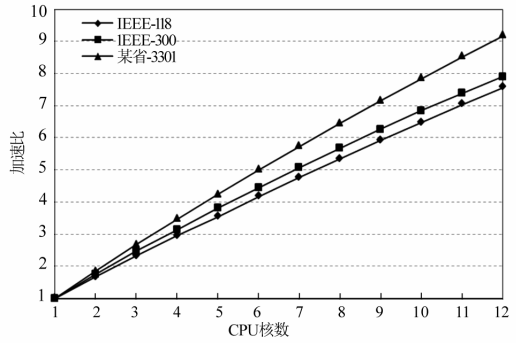

图 13 方法整体加速比曲线

Fig. 13 Speedup of the strategy

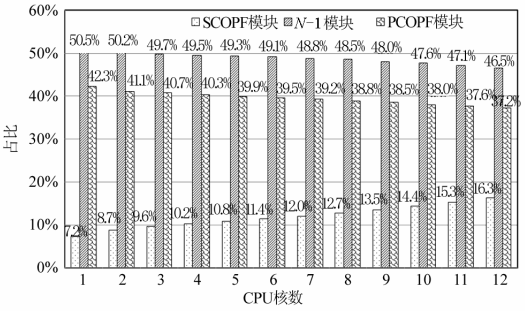

图 14 某省-3301 系统各模块计算时间占比

Fig. 14 Each module calculation time in the Province-3301

4 结论

本文结合 *N*-1 分析、故障过滤和多核并行计算，提出了一种快速求解大规模 SCOPF 问题的计算方法。IEEE-118、IEEE-300 及某省-3301 节点系统多故障状态的计算结果表明：所提方法在保证正确的计算结果的前提下，能够显著提升 SCOPF 计算速度并降低计算对于硬件环境的依赖，具有广泛的应用前景。

参考文献

[1] 蒋程, 刘文霞, 张建华, 等. 含风电接入的发输电系统风险评估[J]. 电工技术学报, 2014, 29(2): 260-270.
JIANG Cheng, LIU Wenxia, ZHANG Jianhua, et al. Risk assessment of generation and transmission systems considering wind power penetration[J]. Transactions of China Electrotechnical Society, 2014, 29(2): 260-270.

[2] 陈昌松, 段善旭, 蔡涛, 等. 基于模糊识别的光伏发电短期预测系统[J]. 电工技术学报, 2011, 26(7): 83-89.
CHEN Changsong, DUAN Shanxu, CAI Tao, et al. Short-term photovoltaic generation forecasting system based on fuzzy recognition[J]. Transactions of China Electrotechnical Society, 2011, 26(7): 83-89.

[3] 何世恩, 郑伟, 智勇, 等. 大规模集群风电接入电网电能质量问题探讨[J]. 电力系统保护与控制, 2013, 41(2): 39-44.
HE Shien, ZHENG Wei, ZHI Yong, et al. Power quality issues of large-scale cluster wind power integration[J]. Power System Protection and Control, 2013, 41(2): 39-44.

[4] 许丹, 赵鸿图, 丁强, 等. 基于实用化安全约束经济调度扩展建模策略[J]. 电力系统保护与控制, 2013, 41(24): 76-81.
XU Dan, ZHAO Hongtu, DING Qiang, et al. Modeling strategy based on utility of security constrained economic dispatch[J]. Power System Protection and Control, 2013, 41(24): 76-81.

[5] MONTICELLI A J, PEREIRA M V P, GRANVILLE S. Security-constrained optimal power flow with post-contingency corrective rescheduling[J]. IEEE Transactions on Power Systems, 1987, 2(1): 175-182.

[6] CAPITANESCU F, WEHENKEL L. Improving the statement of the corrective security constrained optimal power flow problem[J]. IEEE Transactions on Power Systems, 2007, 22(2): 887-889.

[7] YUAN Li, MCCALLEY J D. Decomposed SCOPF for improving efficiency[J]. IEEE Transactions on Power Systems, 2009, 24(1): 494-495.

[8] 钟世民, 韩学山, 刘道伟, 等. 计及校正控制的安全约束最优潮流的奔德斯分解算法[J]. 中国电机工程学报, 2011, 31(1): 65-71.
ZHONG Shimin, HAN Xueshan, LIU Daowei, et al. Benders decomposition algorithm for corrective security constrained optimal power flow[J]. Proceedings of the CSEE, 2011, 31(1): 65-71.

[9] CAPITANESCU F, WEHENKEL L. A new iterative approach to the corrective security constrained optimal power flow problem[J]. IEEE Transactions on Power Systems, 2008, 23(4): 1533-1541.

[10] KAROUI K, CRISCIU H, SZEKUT A, et al. Large scale security constrained optimal power flow[C] // The 16th Power Systems Computation Conference (PSCC), Glasgow, Scotland, 2008.

[11] PLATBROOD L, CAPITANESCU F, MERCKX C, et al. A generic approach for solving nonlinear discrete security constrained optimal power flow problems in large-scale systems[J]. IEEE Transactions on Power Systems, 2014, 29(3): 1194-1203.

[12] PLATBROOD L, CRISCIU H, CAPITANESCU F. Solving very large-scale security-constrained optimal power flow problems by combining iterative contingency and network compression[C] // The 17th Power Systems Computation Conference (PSCC), Stockholm, Sweden, 2011.

[13] 张小兵, 吴政球, 李连伟, 等. 基于拟合方法的 *N*-1 静态电压稳定裕度计算[J]. 电力系统保护与控制, 2010, 38(14): 23-27.
ZHANG Xiaobing, WU Zhengqiu, LI Lianwei, et al. *N*-1 steady-state voltage margin calculation based on fitting computation method[J]. Power System Protection and Control, 2010, 38(14): 23-27.

[14] 张勇军, 蔡广林, 邱文锋. 基于最优乘子潮流估计的故障筛选与排序[J]. 电工技术学报, 2010, 25(1): 123-128.
ZHANG Yongjun, CAI Guanglin, QIU Wenfeng. Contingency screening and ranking based on optimal multiplier power flow evaluation[J]. Transactions of China Electrotechnical Society, 2010, 25(1): 123-128.

[15] BOUFFARD F, GALIANA F D, ARROYO J M. Umbrella contingencies in security-constrained optimal power flow[C] // The 15th Power Systems Computation Conference (PSCC), Belgium, Liège, 2005.

[16] 李佩杰, 韦化, 李滨, 等. 最优潮流中有向无环图的并行数值分解算法[J]. 电力系统自动化, 2012, 36(19): 66-72.
LI Peijie, WEI Hua, LI Bin, et al. A parallel numerical factorization algorithm based on directed acyclic graph in optimal power flow[J]. Automation of Electric Power Systems, 2012, 36(19): 66-72.

收稿日期：2014-05-05

作者简介：

傅志生(1987-)，男，硕士研究生，研究方向为电力系统最优化；

白晓清(1969-)，女，博士，副教授，研究方向为电力系统最优化；

李佩杰(1984-)，男，通信作者，博士，副教授，研究方向为最优化理论在电力系统小干扰稳定中的应用，电力系统稀疏、并行计算。E-mail: lipeijie@gxu.edu.cn