面向图形处理器的高效交流最优潮流分析

´Kasia Swirydowicz先进计算、数学与数据部 太平洋西北国家实验室美国华盛顿州里奇兰 kasia.swirydowicz@pnnl.gov

Nicholson Koukpaizan国家计算科学中心 橡树岭国家实验室美国田纳西州橡树岭 koukpaizannk@ornl.gov

Shrirang Abhyankar电力基础设施与建筑部 太平洋西北国家实验室美国华盛顿州里奇兰 shri@pnnl.gov

Slaven Peleˇs计算科学与工程部 橡树岭国家实验室美国田纳西州橡树岭 peless@ornl.gov

arXiv:2302.08656v2 [cs.CE] 6 May 2023

摘要—**我们介绍了在图形处理器（GPU）上进行稀疏交流最优潮流（ACOPF）分析的一种实现方案。具体而言，我们讨论了性能瓶颈，并详细介绍了加速线性求解器所做的工作。线性求解器是交流最优潮流的核心组件，并占据了主要的计算时间。对美国东北部（25,000 条母线）和东部（70,000 条母线）两个大型系统的合成电网进行的交流最优潮流分析表明， [1]，在图形处理器上的运行速度相较于使用最先进求解器在中央处理器（CPU）上的分析有望实现加速。据我们所知，这是首个展示在图形处理器上显著加速稀疏交流最优潮流的结果。**

索引词—**交流最优潮流，图形处理器，异构计算，优化，稀疏求解器，经济调度**

I. 引言

随着新技术、新流程和数字化的涌现，电力电网正经历着巨大的转型，旨在为每个人提供更清洁、更智能、更实惠且更公平的电力。然而，与此同时，电网正面临气候变化和网络攻击带来的前所未有的威胁。近期，加利福尼亚州因热浪导致的停电，以及佛罗里达州和波多黎各因飓风导致的电力中断，都是影响数百万人生活的大规模电力中断案例。在此类“暗空”事件期间运行电网，以及通常在可再生能源预测不确定的“间歇天空”日运行电网，仍是一项日益严峻的挑战。精准、快速的广域分析工具为电网运营商和规划人员提供了关键支持，帮助他们作出决策。然而，更高的保真度也意味着更高的计算复杂度。

本研究得到了百亿亿次计算项目（17-SC- 20-SC）的支持。该项目由美国能源部科学办公室和国家核安全管理局合作开展。本研究使用了橡树岭领导力计算设施的资源，该设施由美国能源部科学办公室根据合同编号 DE- AC05-00OR22725 提供支持。

异构计算，例如将中央处理器与硬件加速器相结合，正成为计算领域的主导范式。在电子游戏产业、人工智能和计算机视觉应用需求的推动下，图形处理器如今已成为最普及的硬件加速器。图形处理器通过大规模细粒度并行，以较低成本提供强大的计算能力。

使用图形处理器面临的挑战源于其单指令多数据（SIMD）架构，该架构对建模和求解器施加了约束。SIMD架构适合移植基于CPU的传统稠密向量和矩阵运算。传统稀疏运算（当前最先进的交流最优潮流分析所采用的运算） [2]若不经过额外的数学考量，就无法改写为SIMD形式。

在我们先前的工作中， [3],我们讨论了如何通过压缩电力系统中自然产生的稀疏形式，使用稠密求解器在图形处理器上求解交流最优潮流 [2],[4] 。此前，文献 [5]提出了一种使用完全稠密线性代数和 Cholesky 求解器求解凸交流最优潮流问题的方法。然而，稠密线性求解器的计算复杂度为 $O(N^3)$，内存复杂度为 $O(N^2)$，其中 N为线性方程的数量。对于更大型的电网模型，图形处理器带来的所有性能提升都会被计算成本的立方增长和内存的平方增长抵消。

本文介绍我们在图形处理器上求解稀疏交流最优潮流的研究结果。具体而言，我们详细介绍了加速线性求解器的工作；线性求解器是交流最优潮流的核心组成部分，也是计算成本的主要来源 [6]。这项探索的动机来自我们近期在交流最优潮流问题的稀疏线性求解器方面的工作[7], [8], ，以及近期关于图形处理器上动态相量仿真 [9], [10], 潮流 [11], [12],和批量潮流仿真 [13], [14] 的研究。

本文的主要贡献如下：

•首次报告了在图形处理器上实现有意义的交流最优潮流加速，使用我们开发的稀疏线性求解器，并在大型互联系统级电网模型上进行了演示；该结果可通过开源™ 软件包 ExaGO 复现（第 III 节）。

• 交流最优潮流在 CPU（第 II 节）和图形处理器（第 III 节）上的性能分析，并指出了主要瓶颈。
• 预测当前技术的性能及其局限，并确定为在图形处理器上实现更大幅度加速所需的研究方向（第 IV 节和第 V 节）。

II. 交流最优潮流分析的先进方法

交流最优潮流属于非线性非凸优化问题。其紧凑的数学形式可表述为:

$$\min_{x} \; F(x) \tag{1}$$
$$\text{s.t.} \;\; g(x) = 0 \tag{2}$$
$$h^- \le h(x) \le h^+ \tag{3}$$
$$x^- \le x \le x^+ \tag{4}$$

其中， $F(x)$ 是定义优化目标的标量函数，通常为发电总成本； x 是优化变量向量，例如发电机有功功率输出；向量 x^-， x^+ 定义变量界限，例如资源容量限制；向量函数 $g(x)$ 定义等式约束，例如功率平衡；向量函数 $h(x)$ 定义安全约束，其安全限值由向量 h^-， h^+ 给出。有关交流最优潮流公式的详细说明，请参阅 [2], [15] 。

优化问题 (1)-(4) 通常采用内点法进行迭代求解 [16]。为使表述简洁且不失一般性，我们将 (1)-(4) 改写为更紧凑的形式

$$\min_{y} \; f(y) \tag{5}$$
$$\text{s.t.} \;\; c(y) = 0 \tag{6}$$
$$y \ge 0 \tag{7}$$

在此，我们通过添加松弛变量将 (3) 转换为等式约束 s', $s'' \ge 0$ 满足 $h(x) - h^- - s'$ 且 $h(x) - h^+ + s''$。通过引入变量 x'， $x'' \ge 0$ 使其满足 $x - x^- = x'$ 和 $x^+ - x = x''$，将边界条件 (4) 表示为形式 (7)。在紧凑形式中，优化原始变量向量为 $y = (x',\ x'',\ s',\ s'') \in \mathbb{R}^n$ 且 $f: \mathbb{R}^n \to \mathbb{R}$ 是标量目标函数 $F(x)$，它在原始变量空间中表示为y。所有约束均表示为 c: $\mathbb{R}^n \to \mathbb{R}^m$，其中 m是等式约束和不等式约束的数量。

为强制满足变量界限，将目标函数 (5) 加入障碍函数，得到

$$\phi(y) = f(y) + \mu \sum_{i=1}^{n} \ln y_i. \tag{8}$$

然后，通过求解非线性方程即可得到障碍子问题的解

$$\nabla f(y) + \nabla c(y)\lambda - z = 0, \tag{9}$$
$$c(y) = 0, \tag{10}$$
$$Yz - \mu e = 0 \tag{11}$$

其中， $J \equiv \nabla c(y)$ 为 $m \times n$ 约束雅可比矩阵， $\lambda \in \mathbb{R}^m$ 和 $z \in \mathbb{R}^n$ 分别为等式约束 (10) 和边界约束 (11) 的拉格朗日乘子向量， $Y \equiv \text{diag}(y)$，而 $e \in \mathbb{R}^n$ 为全 1 向量。

障碍子问题 (9)-(11) 使用牛顿法求解，并通过设置 $\mu \to 0$的延续算法得到原始优化问题的解。在搜索最优解的过程中，内点法会求解一系列 Karush-Kuhn-Tucker (KKT) 类型的线性化系统 $K_k \Delta x_k = r_k$,$k = 1, 2, \ldots, M$。内点法的典型实现会通过分块高斯消元，从 (9)-(11) 的线性化形式中消去z ，以得到更小的对称系统

$$\overbrace{\begin{bmatrix} H + D_y & J \\ J^T & 0 \end{bmatrix}}^{K_k} \overbrace{\begin{bmatrix} \Delta y \\ \Delta \lambda \end{bmatrix}}^{\Delta x_k} = \overbrace{\begin{bmatrix} r_y \\ r_\lambda \end{bmatrix}}^{r_k}, \tag{12}$$

其中，索引 k 表示优化求解器的迭代次数（包括 μ 中的延续步骤和牛顿迭代）， K_k为系统 (KKT) 矩阵，向量 Δx_k 为原始变量和对偶变量的搜索方向，而 r_k 由在原始变量和对偶变量当前取值处计算的 (9)-(11) 的残差向量导出。海森矩阵

$$H \equiv \nabla^2 f(y) + \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(y),$$

是一个稀疏对称 $n \times n$ 矩阵， $D_y \equiv \mu Y^{-2}$ 是一个对角 $n \times n$ 矩阵。关于内点法实现的更多细节，请参阅 [16]。

(12) 中的矩阵 K_k 是稀疏对称不定矩阵。所有K_k 都具有相同的稀疏模式，求解器可以利用这一特性。交流最优潮流的特点是，分析过程中生成的 K_k非常稀疏，且具有不规则的非零模式，不存在稠密块 [17]。图 1 展示了一个这样的例子。

当障碍参数 $\mu = 0$时，矩阵 (12) 是奇异的。因此，驱动 $\mu \to 0$ 的延续算法需要在增广系统的解足够接近原系统的解时退出，但必须早于 (12) 变得过于病态。KKT 矩阵通常具有极高的条件数（$> 10^{16}$），这意味着以简单方式求解这些系统（例如，不使用主元选取和均衡化）会导致解的误差超过计算机浮点双精度的精度。

只有少数线性求解器能够有效处理这类问题。这些求解器通常是基于 LDL^{T} 或 LU 分解 [18]的稀疏直接求解器。当前最先进的方法通常采用 KLU（用于非对称矩阵）[10], [19]或 MA57 [20]。

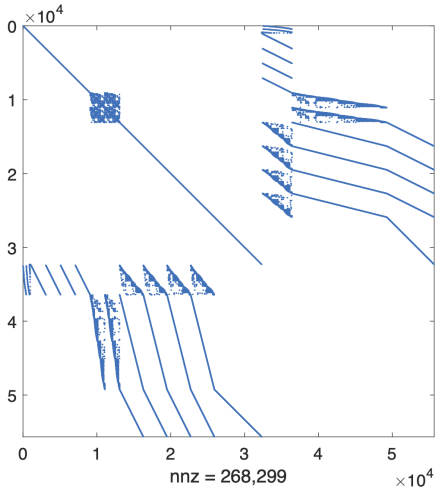
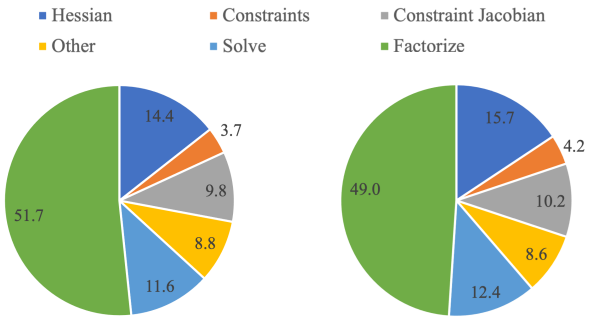


图 1：对 2000 母线德克萨斯电网模型进行交流最优潮流分析得到的 KKT矩阵稀疏模式。

稀疏直接求解器通常分三个阶段计算线性系统的解：(*i*)分析阶段，在此阶段对系统进行均衡，设置列置换和行置换，确定主元顺序，并选定因子的非零模式；(*ii*)数值分解阶段，在此阶段计算矩阵因子；以及(*iii*)求解阶段，在此阶段使用前一阶段计算出的因子进行前向和后向三角求解。一些求解器会在第三阶段之后进行迭代改进 [21]。第一阶段通常比其他阶段开销更大，但如果不同系统的矩阵非零模式保持不变（如式 (12) 所示），分析阶段便可只执行一次，并将结果用于所有 K_k [10]、 [12]。数值分解阶段通常需要进行主元选取，以确保计算保持稳定。主元选取会改变 L和 U 因子的稀疏模式，在某些情况下甚至可能需要重新分配内存。因此，虽然主元序列的变化对 CPU 而言或许并不构成问题，但正如我们在第 III 节所述，这会降低图形处理器上的性能。

对于交流最优潮流问题，总计算成本通常有一半以上用于求解 KKT线性系统。其中，矩阵分解（符号分解和数值分解）的成本最高，其次是三角方程求解（后向-前向代换）。其他重要项目包括导数矩阵的计算——约束雅可比矩阵 J 和海森矩阵 H。图 2 展示了这一成本分解，分别对应美国东北部电网的 25,000 母线模型和美国东部电网的 70,000 母线模型。表 I 描述了所评估的这两个合成电网。

分析使用 ExaGO™软件包 [22]，并采用 HiOp 优化引擎 [23] 和 MA57线性求解器完成。数值矩阵分解约占交流最优潮流总计算时间的一半，而三角求解则占 12%。模型评估



(a) 美国东北部电网　　(b) 美国东部电网

图 2：交流最优潮流的计算成本按求解器功能细分。线性求解器功能（矩阵分解和三角求解）占交流最优潮流总计算时间的 60%。

表 I：性能分析中使用的合成电网模型 [1]说明。各模型的线性系统 (12)具体信息以矩阵K_k 大小 (N) 和非零元数量 (nnz) 表示。

Grid	母线	发电机	支路	N	nnz
美国东北部	25,000	4,834	32,230	107,558	1,191,152
美国东部	70,000	10,390	88,270	296,214	3,228,456

操作约占总计算时间的 30%。交流最优潮流中的所有其他操作（目标函数、目标函数梯度、不可行性范数等的计算）耗时均不到总时间的 10%。能否实现加速比，很大程度上取决于能否在图形处理器上高效求解 KKT线性系统。

III. 使用图形处理器进行计算

科学界一直投入大量精力，致力于利用 GPU加速提升稀疏直接线性求解器的速度，现有软件通常也至少具备一定的 GPU 功能。然而，在图形处理器上求解本文所述的线性系统尤其具有挑战性：(*i*) 由于电力系统具有不规则且极度稀疏的特性，其底层线性问题不具备传统多前沿法 [24] 和超节点法 [25] 可用于构造稠密块的结构，因此无法在这些稠密块上执行 SIMD 运算。(*ii*) 底层线性问题条件不良，为确保数值分解稳定性，必须进行主元选取。主元选取会因条件分支和大量数据移动而降低图形处理器的性能 [13]。

在 [7],中，我们识别并测试了几种经图形处理器加速的稀疏直接求解器，并得出结论：它们的速度均不及 MA57，尤其是在最大的测试用例中。这些求解器只将部分计算（例如矩阵-矩阵乘法或三角方程求解）卸载到图形处理器上，其余计算则在 CPU 上进行。这会因分配和释放小块 GPU内存，以及在设备与主机之间复制数据而产生显著开销。对于我们的测试用例，我们还观察到，在图形处理器上进行分解的成本尤其高。

我们采用重分解方法在图形处理器上求解（12）（见算法 1， M 为待求解系统的总数，该数量取决于优化求解器的设置，通常为数百个）。类似方法已成功用于动态相量 [9]，[10]和潮流 [12]仿真，但尚未用于交流最优潮流。就我们所知，我们的方法首次使用图形处理器加速交流最优潮流分析中的稀疏线性求解器，并且性能超过了当前最先进的 CPU 求解器。我们将软件栈中的 MA57 求解器替换为一个使用 KLU 和 NVIDIA 的 cuSolver 库实现的求解器。我们首先使用 KLU 在 CPU 上对第一个系统进行分解，设置 L 和U的模式，并计算置换向量。我们还评估了 cuSolver 用于设置 L 和 U模式的方法，但发现 KLU 的性能更好。完成计算后，在其余计算过程中，我们保持因子的稀疏模式不变。随后，将因子的非零结构和置换向量复制到图形处理器，并传递给 cuSolver。重分解完成设置后，后续每个系统均使用 GPU 重分解求解，无需进行主元选取，因为在图形处理器上进行主元选取的成本高得令人难以接受。

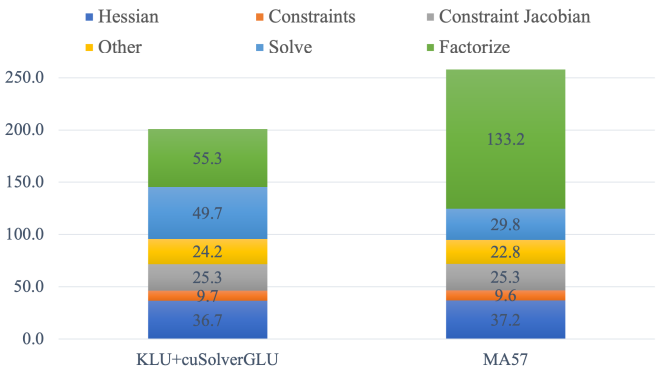
使用这种方法，所有数据移动和映射都只在 CPU 上执行一次，而后续计算则通过图形处理器上的 SIMD 运算完成。其权衡之处在于，重分解方法得到的解质量较低，因此求解阶段之后可能需要进行迭代改进，以恢复优化方法所需的精度。

我们测试了两种重分解接口：cuSolverGLU 和 cuSolverRf。cuSolverGLU 是 cuSolver 库中一个未公开文档（但可公开使用）的部分。它通常比 cuSolverRf 更快，数值稳定性也更好，并且内置了迭代改进功能。本文展示的所有结果均使用 cuSolverGLU 得出。cuSolverGLU 要求以 CSR 格式组合存储这些因子，因此我们在使用 KLU（其操作对象为 CSC 格式的矩阵）后，需要进行格式转换。

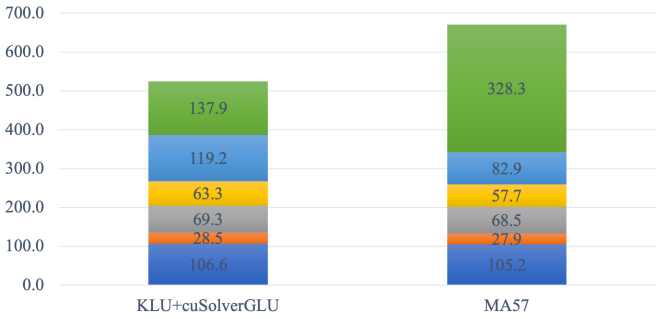
cuSolver 重分解的一大优势是，用户可以提供自己的非零模式、因子以及置换向量；这些因子可以来自任意 LU 求解器（只要数据格式正确），而 cuSolver 会在图形处理器上继续为序列中的后续系统执行计算。

由于优化求解器通常会调用线性求解器数百次，因此，在 CPU 上执行一次线性求解的成本可以分摊到后续迭代中。在图 3 和表 II 中，我们展示了采用此方法获得的计算加速比。图中显示了每次优化求解器迭代的平均成本，其中计入了使用 KLU 进行一次性分解的分摊成本。数值实验使用 IBM Power 9 CPU 和 NVIDIA V100 GPU 进行；该 GPU 配备 16 GB 高带宽内存，双精度峰值性能约为 7 TF。

整体加速比完全来自数值分解的加速。计入初始 KLU



(a) 美国东北部电网



(b) 美国东部电网

图 3：CPU 与图形处理器的计算成本对比（每次迭代的平均时间，单位为毫秒），并按最耗时的操作进行细分。两个测试案例中的模型评估时间相同，因为只有线性求解器由图形处理器加速。

表 II：交流最优潮流总运行时间对比：在图形处理器上使用稀疏线性求解器可实现 30% 的加速比

美国东北部电网			
使用的线性求解器	KLU+cuSolver	MA57	加速比
总时间（秒）	116.0	152.1	1.3
迭代次数	547	529	-
美国东部电网			
所使用的线性求解器	KLU+cuSolver	MA57	加速比
总时间（秒）	146.7	196.4	1.3
迭代次数	263	263	-

分解后，与 CPU 上的 MA57 基准相比，分解的平均加速比为 $2.4\times$ ，整体加速比为 $1.3\times$ 。有些出乎意料的是，图形处理器上的三角求解速度慢了 30–40%。这可能是由于 cuSolverGLU 三角求解器函数中嵌入的迭代改进实现不够理想。我们还注意到，使用图形处理器时，“其他”计算成本略有增加。这主要是 CUDA API 的开销所致，包括启动图形处理器内核的成本。计算的其他部分未通过图形处理器加速，因此在两种测试配置中的成本大致相同。求解过程中 CPU 与图形处理器之间的数据移动耗时不到图形处理器运行时间的 2%。

该求解器在图形处理器上仅需少量内存（1.1 GB，并且

算法 1 重因式分解求解策略：KLU +cuSolverGLU

输入： 线性系统序列 $K_k x_k = b_k,\ k = 1, 2, \ldots, M$
输出： 解向量序列 x_k

1：使用 KLU 求解 $Ax_1 = b_1$
2：提取符号分解和置换向量。 ◁使用 klu_extract，以 CSC 格式返回 L、U
3：将 CSC 转换为合并的 L+U CSR对象。 ◁到此为止，均在 CPU 上计算
4: 设置 cuSolverGLU ◁图形处理器开始计算
5: **对于** $k = 2, \ldots, M$ **执行** ◁ M 是线性系统的总数
6: 使用来自 K_k的新数值更新分解对象。 ◁使用 cuSolverGLU 函数
7: 重新分解.
8: 对 b_k 执行三角求解。 ◁始终随后进行内置迭代改进
9: **结束循环**

对于美国东北部电网的 25,000 母线模型和美国东部电网的 70,000 母线模型，分别需要 2.9 GB）。这与 [3], 中的实现形成鲜明对比，后者在美国西部电网规模较小的 10,000 母线模型上需要 21 GB 的 GPU内存。

IV. 利用现有技术实现的预期性能提升

交流最优潮流的全部性能提升都归功于在图形处理器上加速矩阵分解（图 3）。 [3]中的结果表明，模型评估也可以在图形处理器上加速。因此，我们可以对进一步的性能提升进行预测。我们保守估计，在图形处理器上进行稀疏海森矩阵评估可实现 $4\times$ 的加速比，约束、约束雅可比矩阵及其他模型组件可实现 $3\times$ 的加速比。在 [3], 中，对于所评估的最大电网（含 2,000 个母线的德克萨斯电网模型），向量内核实现了 2 至 $3\times$ 的加速比，稠密矩阵内核的加速比则超过 $15\times$ 。尽管评估稠密矩阵元素时内存访问效率更高，我们仍认为，对于稀疏矩阵评估，至少可实现 $4\times$ 的加速比，尤其是在我们考虑的大型应用场景中。图 4 展示了若在图形处理器上执行模型评估，美国东部电网的预期性能（在图形处理器上的加速比为 $2\times$ ）。

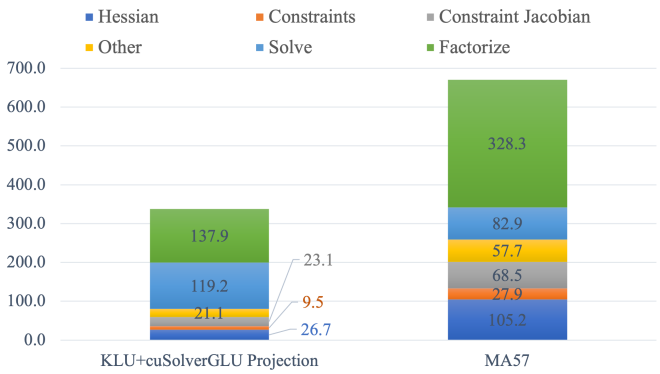


图 4：基于 ExaGO™ 和 HiOp 库中已有的技术，对交流最优潮流分析的加速比进行保守估算，结果显示美国东部电网的加速比为 $2\times$ 。

V. 结论与后续工作

通过将稀疏矩阵分解速度提升 $2.4\times$ ，相较于 CPU 上以 MA57 为基准的方案，我们消除了在图形处理器上高效开展交流最优潮流分析的主要瓶颈。仅加速线性求解器，就使交流最优潮流分析的整体速度提升了 30%。我们还证明，利用现有技术，可以将整个分析过程至少加速 $2\times$ 。

GPU 性能分析结果清楚地表明，整个线性系统求解流程的下一个瓶颈如今是 GPU 三角求解。我们认为，通过采用比 cuSolverGLU 库内置方法更高效、可配置性更强的迭代改进方法，有望实现显著提升。三角方程求解也可以通过雅可比迭代或高斯-赛德尔迭代进行近似，且性能结果颇具前景 [26], ，但我们对这一主题的研究仍处于初期阶段。

交流最优潮流矩阵具有文献中所称的鞍点结构。文献 [8]中开发了一种专用的、适用于 GPU 的求解器，可利用这一结构。该求解器对规模较小的矩阵执行三角求解，初步（独立）结果显示，其性能加速比显著优于 MA57。未来，我们计划将其集成到软件栈中，并评估其性能。

致谢

我们感谢 NVIDIA 的 Lungsheng Chien 和 Doris Pan 在使用 cuSolver 库中未公开的 cuSolverGLU 模块方面提供的帮助。我们还感谢 Cosmin Petra 和 Nai-Yuan Chiang 在使用 HiOp 优化求解器时给予的指导。衷心感谢 Phil Roth 和 Christopher Oehmen 对本工作的支持，也感谢 Shaked Regev 对稿件的认真审阅并提出宝贵意见。最后，感谢三位匿名审稿人提出反馈意见，使这篇稿件成为更好的出版物。

参考文献

[1] A. B. Birchfield、T. Xu、K. M. Gegner、K. S. Shetye 和 T. J. Over- bye，“以电网结构特征作为合成网络的验证标准”，*IEEE* 电力系统汇刊，第 32 卷，第 4 期，第 3258–3265 页，2017年。

[2] R. P. O’Neill、A. Castillo 和 M. B. Cain，“交流最优潮流问题的 IV 公式与线性近似（OPF 论文 2）”，FERC 工作人员技术报告，第 12 期，第 1–18 页，2012年。 [在线]。网址：http://www.ferc.gov/industries/electric/indus-act/market-planning/opf-papers/acopf-2-iv-linearization.pdf [3] S. Abhyankar、S. Peles、R. Rutherford 和 A. Mancinelli，“在图形处理器上评估交流最优潮流”，载于 2021 年 IEEE 电力与能源协会全会（PESGM），2021年，第 01–05 页。[4] S. Frank、S. Rebennack 等，“最优潮流入门：理论、公式与实践示例”，科罗拉多矿业学院，技术报告 14，2012年。[5] L. Rakai 和 W. Rosehart，“最优潮流问题的 GPU 加速求解”，载于 2014 年第 47 届夏威夷国际系统科学会议，2014年，第 2511–2516 页。[6] X. Su、C. He、T. Liu 和 L. Wu，“全并行潮流求解：用于牛顿-拉夫逊实现的基于GPU-CPU的向量化并行化与稀疏技术”，IEEE 智能电网汇刊，第 11 卷，第 3 期，第 1833–1844 页，2019年。[7] K. ´Swirydowicz、E. Darve、W. Jones、J. Maack、S. Regev、M. A. Saunders、S. J. Thomas 和 S. Pele˘s，“电网优化问题的线性求解器：GPU加速线性求解器综述”，《并行计算》，第 111 卷，文章编号 102870，2022年。[8] S. Regev、N.-Y. Chiang、E. Darve、C. G. Petra、M. A. Saunders、K. ´Swirydowicz 和 S. Pele˘s，“HyKKT：一种求解 KKT 线性系统的混合直接-迭代方法”，《优化方法与软件》，第 1–24 页，2022年。[9] J. Dinkelbach、L. Schumacher、L. Razik、A. Benigni 和 A. Monti，“用于实时电力系统仿真高性能 LU 分解的因子分解路径重新因子分解方法”，《能源》，第 14 卷，第 23 期，文章编号 7989，2021年。[10] L. Razik、L. Schumacher、A. Monti、A. Guironnet 和 G. Bureau，“电力系统仿真中 LU 分解方法的比较分析”，载于 2019 年 IEEE 米兰 PowerTech。IEEE，2019年，第 1–6 页。[11] D. Ablakovic、I. Dzafic 和 S. Kecici，“使用 GPU 并行化求解辐射状三相配电潮流”，载于 2012 年第 3 届 IEEE PES 欧洲创新智能电网技术会议（ISGT Europe），2012年，第 1–7 页。[12] M. D’orto、S. Sj¨oblom、L. S. Chien、L. Axner 和 J. Gong，“比较使用牛顿-拉夫逊方法求解大规模潮流问题的不同方法”，《IEEE Access》，第 9 卷，56 615，2021年 56 604页。[13] G. Zhou、R. Bo、L. Chien、X. Zhang、F. Shi、C. Xu 和 Y. Feng，“用于海量潮流并发分析的 GPU 批量 LU 因子分解求解器”，IEEE 电力系统汇刊，第 32 卷，第 6 期，第 4975–4977 页，2017年。[14] G. Zhou、R. Bo、L. Chien、X. Zhang、S. Yang 和 D. Su，“在线概率潮流的 GPU 加速算法”，IEEE 电力系统汇刊，第 33 卷，第 1 期，第 1132–1135 页，2018年。[15] R. D. Zimmerman、C. E. Murillo-S´anchez 和 R. J. Thomas，“MAT-POWER：面向电力系统研究与教育的稳态运行、规划和分析工具”，IEEE 电力系统汇刊，第 26 卷，第 1 期，第 12–19 页，2011年。[16] A. W¨achter 和 L. T. Biegler，“大规模非线性规划中内点滤波线搜索算法的实现”，《数学规划》，第 106 卷，第 1 期，第 25–57 页，2006年。[17] L. R. de Araujo、D. R. R. Penido、J. L. R. Pereira 和 S. Carneiro，“用于电力系统求解算法的稀疏线性系统实现”，《控制、自动化与电气系统杂志》，第 24 卷，第 4 期，第 504–512 页，2013年 8 月。[18] G. H. Golub 和 C. F. Van Loan，《矩阵计算》。JHU 出版社，2013年。[19] T. A. Davis 和 E. Palamadai Natarajan，“算法 907：KLU，一种用于电路仿真的直接稀疏求解器”，ACM数学软件汇刊（TOMS），第 37 卷，第 3 期，第 1–17 页，2010年。[20] I. S. Duff，“MA57——用于求解稀疏对称定性与不定性系统的代码”，ACM数学软件汇刊（TOMS），第 30 卷，第 2 期，第 118–144 页，2004年。[21] R. D. Skeel，“迭代改进意味着高斯消元具有数值稳定性”，《计算数学》，第 35 卷，第 151 期，第 817–832 页，1980年。[22] S. Abhyankar、S. Peles、A. Mancinelli、R. Rutherford 和 B. Palmer，《百亿亿次级电网优化工具包》，2020年。 [在线]。网址：https://https://gitlab.pnnl.gov/exasgd/frameworks/exago

[23] C. G. Petra、N. Chiang 和 J. Wang，《HiOp——用户指南》，应用科学计算中心，劳伦斯利弗莫尔国家实验室，技术报告 LLNL-SM-743591，2018年。[24] I. S. Duff 和 J. K. Reid，《不定稀疏对称线性方程组的多前沿求解》，*ACM*数学软件汇刊（*TOMS*），第 9 卷，第 3 期，第 302–325 页，1983年。[25] J. W. Demmel、S. C. Eisenstat、J. R. Gilbert、X. S. Li 和 J. W. Liu，《稀疏部分主元选取的超节点方法》，*SIAM*矩阵分析与应用杂志，第 20 卷，第 3 期，第 720–755 页，1999年。[26] H. Anzt、E. Chow 和 J. Dongarra，《用于预条件的稀疏三角方程组迭代求解》，载于欧洲并行处理会议。施普林格，2015年，第 650–661 页。