Vol. 46 (2026)
No. 4

数 学 杂 志
J. of Math. (PRC)

基于 DFS 的 GaBP 算法优化研究

邱德志, 陈思远
(南京航空航天大学数学学院, 南京 210016)

摘 要: 本文研究了 Gauss 信念传播算法在具有多个子系统的线性系统中的加速问题. 利用深度优先遍历算法来减少 Gauss 信念传播算法在这一类线性系统中的计算复杂度, 得到了 DFS-GaBP 算法. 同时我们获得了 DFS-GaBP 算法在子系统上的收敛结果. 我们的计算结果表明, 对比传统算法, DFS-GaBP 算法在具有多个子系统的线性系统中表现出更优的性能.

关键词: Gauss 信念传播算法, 深度优先遍历, 子系统, 弱连通分量
MR(2010) **主题分类号**: 15A06; 65F10 **中图分类号**: O241.4
文献标识码: A **文章编号**: 0255-7797(2026)04-0187-11

1 简介

线性数值代数的基本问题是求解一个线性方程组 $Ax = b$, 其中 A 为可逆矩阵, b 是一个给定的向量. 最初用来解决这类问题的基本方法是 LU 分解法 (等价于 Gauss 消元法) 和 Cholesky 分解法 [1]. 这些方法通过将矩阵 A 分解为下三角矩阵 L 和上三角矩阵 U(或对称正定矩阵的 Cholesky 分解), 从而将原问题转化为两个易于求解的三角方程组. 尽管直接法在理论上能够精确求解线性方程组, 但当矩阵规模十分庞大且稀疏时, 这些方法的计算复杂度和存储需求会显著增加, 导致其在实际应用中效率低下.

为了应对大规模稀疏线性方程组的求解问题, 迭代算法应运而生. 与直接法不同, 迭代算法通过逐步逼近解的方式, 能够在占用较少的计算资源的情况下获得近似解. 经典的迭代算法包括 Jacobi 迭代法、Gauss Seidel 迭代法 [2] 以及共轭梯度法 (Conjugate Gradient, CG)[3] 等. 这些方法通过利用矩阵的稀疏性, 显著降低了计算复杂度, 特别适用于大规模问题的求解. 然而, 传统的迭代算法在某些情况下仍然存在收敛速度慢、对矩阵条件数敏感等问题, 这促使研究者们不断探索更高效的算法.

在本篇论文中, 我们将探讨一种高效的分布式迭代算法－－ Gauss 信念传播算法 (Gaussian Belief Propagation, GaBP), 用于求解线性方程组. GaBP 算法最初来源于概率图模型中的信念传播 (Belief Propagation, BP)[4] 方法, 其核心思想是将线性方程组的求解问题转化为特殊定义下的图上的概率推理问题.

关于 GaBP 算法的基本属性, 例如收敛性和计算复杂度, 已有学者进行了深入研究. 研究表明, 原始的 GaBP 算法在矩阵 A 为对称正定且图结构为树图时能够稳定收敛 [5]. 然而, 这些条件在实际应用中往往过于严格, 限制了算法的适用范围. 例如, 对称正定矩阵的要求意

*收稿日期: 2025-04-10 接收日期: 2025-06-23
作者简介: 邱德志 (2004–), 男, 广西玉林, 本科生, 研究方向: 信息与计算科学. E-mail:082210219@nuaa.edu.cn.
通讯作者: 陈思远 (2004–), 男, 本科生, 研究方向: 信息与计算科学. E-mail: misyun@nuaa.edu.cn

味着 GaBP 算法只能用于特定类型的线性方程组, 而树图结构的限制则进一步缩小了其应用场景.

在最近的研究中, 有学者将算法进行了改进, 将收敛条件进行了扩大, 使得 GaBP 算法适用于非对称的线性系统 [6]. 本论文将在此基础上进行讨论, 运用数据结构中的相关知识, 对一些具有特殊性质的矩阵在 GaBP 算法求解的运行过程下进行改良优化.

深度优先搜索 (Depth-First Search, DFS)[7] 是一种经典的图遍历算法, 广泛应用于图论、人工智能、网络分析以及组合优化等领域. 其核心思想是从一个起始节点出发, 沿着一条路径尽可能深入地访问节点, 直到无法继续深入为止, 然后回溯到上一个节点, 继续探索其他路径. DFS 通过递归或显式使用栈的方式实现, 能够系统地探索图或树的所有节点, 适用于解决路径查找、连通性检测、拓扑排序等问题.

在实际应用中, DFS 被广泛用于多种场景. 例如, 在拓扑排序中, DFS 可以有效地对有向无环图 (DAG) 进行排序; 在迷宫生成和求解中, DFS 可以生成随机的迷宫并找到从起点到终点的路径; 在图的连通性检测中, DFS 可以判断图中是否存在从起点到终点的路径, 或者找出所有的连通分量. 此外, DFS 的变种如迭代加深 DFS(IDDFS) 和双向 DFS 进一步扩展了其应用范围. IDDFS 通过限制搜索深度来避免 DFS 的深度过大问题, 同时保留了 DFS 的空间效率; 双向 DFS 则从起点和终点同时进行搜索, 显著减少了搜索空间, 适用于已知起点和终点的场景.

本研究提出一种基于 DFS 的 GaBP 算法优化方法, 通过系统性地整合 DFS 的图遍历特性与 GaBP 的消息传递机制, 设计了一种改进的迭代流程, 显著提升了算法在弱连通分量系统中的计算效率.

在本文中, I_n 为 n 阶的单位矩阵, A^T 表示 A 的转置, $|A|$ 表示 A 的绝对值矩阵, $\rho(A)$ 表示 A 的谱半径, $\mathbf{diag}(A)$ 表示由方阵 A 的对角线元素构成的对角矩阵. 假设 D 是对角矩阵, 且对角元素非零, 则 $D^{-1/2}$ 表示将矩阵 D 的所有对角元素 $d_{ii}, (i=1,\cdots,n)$ 取为 $1/\sqrt{d_{ii}}$ 得到的对角矩阵.

2 基于 DFS 的 GaBP

2.1 Gauss 信念传播算法

信念传播算法最开始用于求解概率图模型中的边缘概率密度, 后来 Weiss 等人经过研究发现其在 Gauss 图中的传播具有良好的特性, Bickson 在 2008 年提出用于求解对称正定的线性系统的 Gauss 信念传播算法, 2022 年 Fanaskov 提出了用于非对称矩阵的 GaBP 算法. [5, 6, 8]

信念传播算法可以看作是概率图中每个顶点向周围发送信息的过程, 给定每个节点的初始信息, 我们要求每轮迭代中每个节点向邻接节点发送信息, 这就是信念传播这一名称的由来. 在树状图中, 只要消息持续迭代下去, 我们就一定能够得到每个节点的边缘概率密度, 并且迭代次数等于树的直径. 值得注意的是, 在 Gauss 图中, 每个节点的边缘概率密度可以由两个参数简单地进行表示: 均值 μ 和方差 σ^2, 这就意味着在 Gauss 图中使用信念传播算法, 我们可以不用显式地去写出每个节点的概率密度, 而只需要掌握每个节点的均值及方差即可,

Gauss 信念传播算法的迭代格式如下:

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k\in N(i)\backslash j} m_{ki}(x_i)dx_i, \tag{2.1}$$

$m_{ij}(x_j)$ 表示节点 i 到节点 j 发送的消息, 而 $N(i)$ 表示节点 i 的所有邻接点的集合, 其中自势函数 ϕ_i 和边势函数 ψ_{ij} 的显式定义分别为 $\phi_i(x_i) \triangleq \exp\left(-\frac{1}{2}A_{ii}x_i^2 + b_i x_i\right)$, $\psi \triangleq \exp\left(-\frac{1}{2}x_i A_{ij} x_j\right)$. 最终每个节点通过整合得到的消息, 即可得到自身的边缘概率密度:

$$p(x_i) = \alpha\phi_i(x_i) \prod_{k\in N(i)} m_{ji}(x_i), \tag{2.2}$$

这里的 α 为归一化参数. 可以证明, 在 Gauss 图中, 所有传播的信息 $m_{ij}(x_j)$ 以及自势函数 $\phi_i(x_i)$ 都服从 Gauss 分布, 因此只需保存每条信息的均值以及方差即可. 为了更清楚地说明这一点, 我们介绍下面的定理.

定理 2.1 [5] 假设 X_1 与 X_2 分别为服从高斯分布的随机变量, $f_1(x)$ 与 $f_2(x)$ 分别是 X_1, X_2 的概率密度函数, 且 $X_1 \sim N(\mu_1, P_1^{-1}), X_2 \sim N(\mu_2, P_2^{-1})$, 那么 $f(x) = f_1(x)f_2(x)$ 是分布为 $N(\mu, P^{-1})$ 的随机变量的概率密度函数. 其中

$$\begin{aligned} \mu &= P^{-1}(P_1\mu_1 + P_2\mu_2), \\ P^{-1} &= (P_1 + P_2)^{-1}. \end{aligned}$$

根据定理 2.1, 我们很容易就得到迭代格式 (2.1) 的均值以及方差形式:

$$P_{ij}^{(n+1)} = \overbrace{P_{ii}^{(n)}}^{\phi_i(x_i)} + \sum_{k\in N(i)\backslash j} \overbrace{P_{ki}^{(n)}}^{m_{ki}(x_i)}, \mu_{ij}^{(n+1)} = \left(P_{ij}^{(n)}\right)^{-1} (\overbrace{P_{ii}^{(n)}\mu_{ii}^{(n)}}^{\phi_i(x_i)} + \sum_{k\in N(i)\backslash j} \overbrace{P_{ki}^{(n)}\mu_{ki}^{(n)}}^{m_{ki}(x_i)}). \tag{2.3}$$

对于 (2.2) 式同样可以写成均值以及方差的形式:

$$P_i^{(n)} = \overbrace{P_{ii}^{(n)}}^{\phi_i(x_i)} + \sum_{k\in N(i)} \overbrace{P_{ki}^{(n)}}^{m_{ki}(x_i)}, \mu_i^{(n)} = \left(P_i^{(n)}\right)^{-1} (\overbrace{P_{ii}^{(n)}\mu_{ii}^{(n)}}^{\phi_i(x_i)} + \sum_{k\in N(i)} \overbrace{P_{ki}^{(n)}\mu_{ki}^{(n)}}^{m_{ki}(x_i)}). \tag{2.4}$$

(2.4) 式中的 $\mu_i^{(n)}, P_i^{(n)}$ 分别为第 n 次迭代得到的第 i 个边缘概率密度的均值与方差. 上面的迭代算法被称为并行 GaBP, 其每一步的迭代都是使用上一轮迭代的结果. 事实上 GaBP 算法可以使用本轮迭代的数据来进行迭代, 这种算法被称为串行 GaBP. 一般而言, 串行 GaBP 的迭代速度要比并行 GaBP 的迭代速度更快, 两者的关系类似于 Jacobi 迭代法和 Gauss-Seidel 迭代法.

同时, GaBP 算法也可以应对非对称线性系统, 这时迭代格式变为:

$$m_{ji}^{(n+1)} = P_{ji}^{(n)} \left(b_j + \sum_{k\in N_{in}(j)\backslash i} m_{kj}^{(n)}\right), P_{ji}^{(n)} = -\frac{A_{ij}}{A_{jj} + \sum_{k\in N_{in}(j)\backslash i} P_{kj}^{(n)} A_{kj}}, \tag{2.5}$$

$$\mu_i^{(n)} = \frac{b_i + \sum_{j\in N_{in}(i)} m_{ji}^{(n)}}{A_{ii} + \sum_{j\in N_{in}(i)} P_{ji}^{(n)} A_{ji}}, P_i^{(n)} = A_{ii} + \sum_{j\in N_{in}(i)} P_{ji}^{(n)} A_{ji}. \tag{2.6}$$

其中 $\mu_i^{(n)}, P_i^{(n)}$ 即为计算得到的边缘概率分布的均值与方差, $N_{in}(i) \triangleq \{j|A(i,j) \neq 0\}$, 同样地, 我们可以定义 $N_{out}(i) \triangleq \{j|A(j,i) \neq 0\}$, 上面的迭代格式也同样可以改为串行格式.

需要注意的是, (2.5) 式与 (2.6) 式中的 m_{ji}, P_{ij} 均为人为定义的消息, 这是为了防止求解非对称线性系统的过程中出现的分母为 0 的情况. 这点很容易理解, 考察 (2.3) 式可以发现, 由于 P 是协方差矩阵, 因此是对称正定的, 如果 $P_{ij}^{(n)}$ 不为 0, 那么 $P_{ji}^{(n)}$ 也同样不为零, 但是在非对称线性系统中, 该条件无法保证, 因此要引入人为消息来避免这种情况发生 (当然, 在对称矩阵中, 两类迭代格式是等价的). 以上的 GaBP 算法都遵循定理 3. 1 以及定理 3. 3 的结论.

2.2 线性系统的子系统

在介绍线性系统的子系统之前, 我们首先给出图的基本术语.

- **子图**:设有两个图 $G = (V, E)$ 和 $G' = (V', E')$, 若 V' 是 V 的子集, 且 E' 也是 E 的子集, 则称 G' 是 G 的子图.
- **连通图和连通分量**: 在无向图 G 中, 若两个顶点之间存在路径, 则认为这两个顶点是连通的. 如果在无向图 G 中, 任意两个顶点都是连通的, 则称 G 是连通图. 无向图中的极大连通子图被称为它的连通分量.
- **基图**: 将有向图的所有的有向边替换为无向边, 所得到的图称为原图的基图.
- **弱连通图和弱连通分量**: 假设有向图 G 的基图是连通图, 则称 G 是弱连通图. 有向图中的极大弱连通子图被称为它的弱连通分量.

由于 GaBP 可以视作在概率图中进行信息传播, 因此我们可以从图的角度来进行思考: 假设一个无向图可以被分为若干个连通分量, 由于图中的每一个点都是由相邻点传播消息来得到该点的边缘概率密度, 那么对其中的每一个点来说, 处于不同连通分量的其它点不会对该点得到的信息产生影响, 那么由此我们可以分别对不同的连通分量分别运行 GaBP. 为了说明这一点, 我们考虑图 1 无向图以及对应的邻接矩阵[1]:

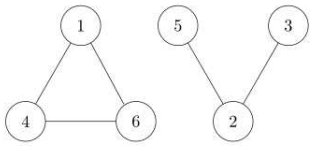
图 1 线性系统可以分成两个子系统, 其中一个子系统由矩阵的第 $1, 4, 6$ 行和列构成, 记作 $A\{1, 4, 6\}$, 另外一个子系统由矩阵的第 $2, 3, 5$ 行和列组成, 记作 $A\{2, 3, 5\}$. 因此在求解线性系统时, 我们可以将线性系统划分为若干个子系统分别求解, 然后将得到的结果按顺序排列即可. 该方法具有以下优点:

- 可以减少 GaBP 的计算规模, 我们从原先考虑的整体系统拆解为若干个子系统, 子系统的规模要小于整体系统, 从而减少计算量.
- 避免了计算冗余, 在之前的算法中, 我们通过遍历矩阵中的元素来考虑不同点之间是否会传播信息, 在我们对系统进行拆解后, 我们知道不同系统之间一定不会传播信息, 这样就避免了不必要的计算.

[1]这里的矩阵取自 [5]

- 不同子系统的性质不同, 对于简单的子系统, 我们能够使用更少的计算量来求解, 从而将更多的计算资源给到颇具难度的系统中.

对于图 1 中的例子来说, 由于 $A\{2,3,5\}$ 所对应的无向图是直径为 3 的树图, 因此对于任给的三维向量 b, 我们使用 GaBP 来求解线性系统 $A\{2,3,5\}x=b$ 只需迭代 3 次. 然而对于另外一个子系统 $A\{1,4,6\}$, 求解该系统所需的计算量要远大于 $A\{2,3,5\}$, 而 $A\{2,3,5\}$ 在第三次迭代时就已经完成了收敛, 所以在后续的计算中 GaBP 会产生计算冗余. 如果将系统分开求解, 我们就可以专注于还未收敛的系统, 以此来减少计算量.



$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 1 \\ 0 & 1 & -2 & 0 & 3 & 0 \\ 0 & -2 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 1 \\ 0 & 3 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

图 1 一个具有两个连通分量的图以及对应的邻接矩阵

2.3 深度优先遍历

由于拆分子系统可以提高 GaBP 的计算效率, 因此我们可以使用 DFS 来寻找矩阵对应图中的所有子系统的结构. DFS 可以利用邻接表或邻接矩阵来得到其对应无向图的所有连通分量, 并输出连通分量上所有的点. 该算法通过递归或显式使用栈的方式实现, DFS 的基本执行过程可以分为以下几个步骤: 首先, 选择一个起始节点并标记为已访问; 然后, 递归地访问其未访问的邻接节点. 对于每个邻接节点, 重复上述过程, 直到当前路径上的所有节点都被访问完毕. 当所有邻接节点都被访问后, 算法回溯到上一个节点, 继续探索其他未被访问的路径, 当递归结束或栈为空时, 表明当前的连通分量已经被访问过了, 记录所有被访问的点, 即为该连通分量上的所有点. 此时寻找未被访问的点, 重复上面的过程, 直到所有的点都被访问. 这样就能得到图中所有的连通分量以及其中的点.

对于邻接表, DFS 的计算复杂度为 $O(V+E)$, 对于邻接矩阵, 两种算法的计算复杂度均为 $O(V^2)$, 而 GaBP 算法每次迭代的计算复杂度为 $O(V^2)$, 显然使用 DFS 寻找连通分量的代价是可接受的. 因此我们考虑将 DFS 与 GaBP 相结合来求解线性系统.

2.4 DFS-GaBP

考察非对称 GaBP 算法, 我们注意到算法可以在弱连通分量中运行, 因此在实际应用中, 我们应该将图中所有的有向边变为无向边再应用 DFS 寻找连通分量, 如果在矩阵中执行, 可以做矩阵加法 $B=|A|+|A^T|$, 再对矩阵 B 执行 DFS 算法. 通过对矩阵 B 执行 DFS 算法, 我们能够得到矩阵 A 的所有子系统.

对于 n 阶方阵 A, 假设 A 中所有子系统构成的集合为 $\mathcal{S}(A)=\{A_1,A_2,\cdots,A_N\}$, N 为弱连通分量的个数, 通过 DFS 算法我们可以找到所有 $A_i(i=1,\cdots,N)$ 的构造. 每一次迭代分别在不同的 A_i 中执行 GaBP 算法, 最终得到解向量 $x=(x_1,x_2,\cdots,x_N)^T$, 其中 x_i 是子系统 $A_ix=b_i$ 的解.

如果要求和原始 GaBP 算法的精确程度相当, 那么 DFS-GaBP 算法的收敛阈值应不大

于原始的 GaBP, 具体的收敛阈值需要根据实际情况具体选择. 基于上面的考虑, 我们正式定义了算法 1.

Algorithm 1: 基于 DFS 的 GaBP 算法

输入: 系数矩阵 A, 向量 b

输出: 解向量 x

```
计算 B = |A| + |A^T|, 在 B 中执行 DFS, 得到子系统集合 S(A).
所有消息 m_ij, P_ij 初始化为 0, 设置收敛阈值 ε.
while S(A) 不为空 do
    for A_k ∈ S(A) do
        for 第 k 个连通分量 A_k 的第 i 节点 k_i, 令 m = b_{k_i}, P = A_{k_i k_i}. do
            for j ∈ N_in(k_i) do
                m = m + Σ_{j∈N_in(k_i)} m_{jk_i}, P = P + Σ_{j∈N_in(k_i)} P_{jk_i} A_{jk_i}
            end
            for j ∈ N_out(k_i) do
                P_{k_i j} = −A_{jk_i}/(P − P_{jk_i} A_{jk_i}), m_{k_i j} = P_{k_i j}(m − m_{jk_i})
            end
            x_{k_i} = m/P
        end
        if 解在 A_k 中收敛 then
            S(A) = S(A) − {A_k}
        end
    end
end
```

3 DFS-GaBP 的性质

3.1 收敛性分析

关于 GaBP 的收敛性有来自 Malioutov 等人 [9] 的结果, 我们可以利用其结果来分析 DFS-GaBP 算法. 对于线性系统 $Ax = b$, 假设矩阵可以分为若干个子系统, 则存在排列方阵 P, 使得

$$P^T AP = \begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_k \end{pmatrix} \triangleq \Lambda \tag{3.1}$$

显然 Λ 是准对角矩阵, 这时线性问题转化为 $\Lambda Px = Pb$. 由于 P 是排列矩阵, 因此它仅仅只重新排列矩阵的行和列, 而不改变矩阵元素的数值, 所以我们有如下结论.

定理 3.1 矩阵 A 对角占优当且仅当矩阵 Λ 对角占优.

首先 [8] 中提出当矩阵可逆且对角占优时, GaBP 算法收敛. 而根据定理 3.1, 我们知道矩阵 A 收敛时所有的子矩阵 A_i 都是对角占优的. 实际上 DFS-GaBP 的过程是在每一个 A_i 上分别执行 GaBP 算法, 由于 DFS 算法可以寻找每个弱连通分量具有哪些节点, 因此我们可

以省去寻找排列矩阵 P 的过程. 而所有的 A_i 都是对角占优的, 因此 GaBP 在所有的 A_i 上收敛, 基于上面的分析我们可以得到 DFS-GaBP 收敛的充分条件.

定理 3.2 矩阵 A 可逆且对角占优时, DFS-GaBP 收敛.

上面的定理在实际上比较实用, 因为矩阵的对角性质较为容易判断. 而在理论上, 关于 GaBP 的收敛性有更加一般的结论.

定理 3.3 [9] 假设矩阵 A 满足条件

$$\rho(|I_n - D^{-1/2}AD^{-1/2}|) < 1, \tag{3.2}$$

则 GaBP 收敛, 其中 $D = \mathbf{diag}(A)$.

但当矩阵 A 满足定理 3.3 的条件时, Λ 的对角块矩阵是否满足相同的条件? 事实上, 答案是肯定的, 并且这是一个充要条件. 因此当矩阵 A 满足上述的条件时, 所有的 A_i 满足一样的条件, 于是 DFS-GaBP 在每个 A_i 上收敛.

定理 3.4 当矩阵 A 满足条件

$$\rho(|I_n - D^{-1/2}AD^{-1/2}|) < 1, \tag{3.3}$$

则 GaBP 在每一个 A_i 上收敛, 即 DFS-GaBP 收敛, 其中 $D = \mathbf{diag}(A)$.

证 事实上, 由于每个 A_i 都是 Λ 上的对角块矩阵, 因此我们只需要证明

$$\rho(|I_n - D^{-1/2}AD^{-1/2}|) < 1 \Leftrightarrow \rho(|I_n - S^{-1/2}\Lambda S^{-1/2}|) < 1, \tag{3.4}$$

即可, 其中 $S = diag(\Lambda)$. 根据 (3.1) 式, 其中的 P 是排列矩阵, 因此也同时是正交矩阵, 得到

$$\begin{aligned} D^{-1/2}AD^{-1/2} &= D^{-1/2}P\Lambda P^T D^{-1/2} \\ &= (PP^T)D^{-1/2}P\Lambda P^T D^{-1/2}(PP^T) \\ &= P(P^T D^{-1/2}P)\Lambda(P^T D^{-1/2}P)P^T. \end{aligned}$$

因为 $(P^T D^{-1/2}P)\Lambda(P^T D^{-1/2}P) = P^T D^{-1/2}AD^{-1/2}P$, 而 $D^{-1/2}AD^{-1/2}$ 对角线上元素均为 1,P 为排列矩阵, 因此矩阵 $(P^T D^{-1/2}P)\Lambda(P^T D^{-1/2}P)$ 的对角线元素均为 1, 容易验证 $P^T D^{-1/2}P$ 是对角矩阵, 因此 $P^T D^{-1/2}P = S^{-1/2}$. 则有 $D^{-1/2}AD^{-1/2} = PS^{-1/2}\Lambda S^{-1/2}P^T$. 于是

$$\begin{aligned} I_n - D^{-1/2}AD^{-1/2} &= I_n - PS^{-1/2}\Lambda S^{-1/2}P^T \\ &= PP^T - PS^{-1/2}\Lambda S^{-1/2}P^T \\ &= P(I_n - S^{-1/2}\Lambda S^{-1/2})P^T. \end{aligned}$$

由于 P 是排列矩阵, 而对于矩阵 A, 取绝对值和行列交换两种操作是可交换的. 因此有 $|PAP^T| = P|A|P^T$, 所以根据上面的式子, 我们有 $|I_n - D^{-1/2}AD^{-1/2}| = P|I_n - S^{-1/2}\Lambda S^{-1/2}|P^T$, 于是 $|I_n - D^{-1/2}AD^{-1/2}|$ 和 $|I_n - S^{-1/2}\Lambda S^{-1/2}|$ 等价, 这样就证明了 $\rho(|I_n - D^{-1/2}AD^{-1/2}|) < 1 \Leftrightarrow \rho(|I_n - S^{-1/2}\Lambda S^{-1/2}|) < 1$.

我们还可以证明定理 3.4 的条件是定理 3.2 的条件的充分条件, 因此定理 3.4 的条件更加一般. 而在实际应用中我们更多地使用定理 3.2.

表 1 算法时间复杂度比较

算法	时间复杂度
GaBP	$k \times O(n^2)$
DFS-GaBP	$O(n^2) + \sum_{A_i \in \mathcal{S}(A)} k_i \times O(n_i^2)$

3.2 时间复杂度分析

假设矩阵 A 的阶数为 n, A 的弱连通分量的集合为 $\mathcal{S}(A) = \{A_1, A_2, \cdots, A_k\}$, 分别记弱连通分量的节点个数为 $n_i (i = 1, \cdots, k)$. 根据 (3.1) 式, 我们知道 DFS-GaBP 每次迭代的时间复杂度为:$\sum_{i=1}^{k} O(n_i^2)$, 由于 $\sum_{i=1}^{k} n_i = n$, 因此一定有 $\sum_{i=1}^{k} n_i^2 \leqslant n^2$, 而 GaBP 每次迭代的时间复杂度为 $O(n^2)$, 因此在理论上 DFS-GaBP 每次迭代的计算量要少于原始 GaBP. 表 1 比较了两者的时间复杂度.

在实际应用中, 可仅对未收敛的弱连通分量执行 GaBP. 如果 GaBP 算法在其中某个子系统 A_k 中收敛了, 那么接下来的计算中我们不需要在 A_k 中进行迭代, 这时 $\mathcal{S}(A) = \mathcal{S}(A) - \{A_k\}$, 因此实际上每次迭代时间复杂度应该为 $\sum\limits_{A_i \in \mathcal{S}(A)} O(n_i^2)$. 由于执行 DFS 的计算复杂度为 $O(n^2)$, 这与原始 GaBP 算法每次迭代的计算量相当, 所以在有多个子系统的线性系统中, 我们可以使用 DFS-GaBP 算法来求解线性方程组.

4 实验及其分析

4.1 基于黑油模型的 IMPES 模拟矩阵

图 2 是一个油藏模拟的 5005 阶稀疏矩阵 [10] 的零元素比例分块热图以及连通分量热图, 右图中不同颜色代表着不同的连通分量, 我们使用 DFS 算法计算得到该矩阵具有 2111 个连通分量.

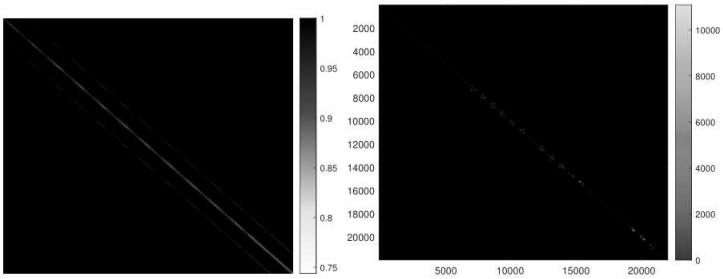


图 2 矩阵结构及其连通分量热图

我们以该矩阵为基础, 以 10^{-5} 为收敛阈值使用了 Jacobi 迭代、SOR 迭代、GMRES 子空间法 (最大迭代步长为 20000)、原始 GaBP 算法以及我们改进之后的 DFS-GaBP 算法进行了求解和比较, 并记录了它的迭代情况和各算法计算时间情况如图 3.

在这一算法中, 我们首先看到 Jacobi 迭代的速度十分缓慢, 迭代曲线十分平缓, 迭代了 57304 次才成功收敛. 另外 GMRES 子空间法无法适用于这个问题, 算法无法收敛. 而 SOR 迭代法则表现较好, GaBP 算法与 DFS-GaBP 算法也稳步收敛.

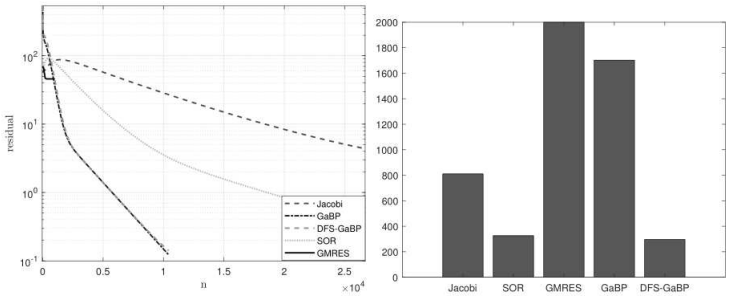


图 3 迭代次数 (纵坐标为 $\|\mathbf{x}^{(n)}-\mathbf{x}^{(n-1)}\|_2$) 以及算法计算时间比较

再通过计算时间的比较我们可以更好的发现 DFS-GaBP 的优势所在 (此处因 GMRES 子空间法无法收敛所以取了一个较大值). 在所有算法中, 只有 SOR 迭代法和 DFS-GaBP 算法对这个问题有十分优秀的计算效率, 而 DFS-GaBP 算法是计算时间最少, 也就是计算效率最高的算法. 通过比较, 我们也发现 DFS-GaBP 算法计算该问题的时间, 比原始 GaBP 算法快了将近八倍, 取得了较好的成果.

4.2 化学过程模拟矩阵

我们继续在一个 21982 阶的化学模拟矩阵 [11] 上进行实验, 该矩阵具有 11091 个连通分量, 且是对角占优矩阵, 但该矩阵不可逆, 因此我们只增加矩阵对角线元素的值, 不改变其结构的情况下使其可逆. 由于子空间法在该矩阵上很难收敛, 因此我们只使用 GaBP 算法与迭代法来进行求解. 结果如图 4 所示.

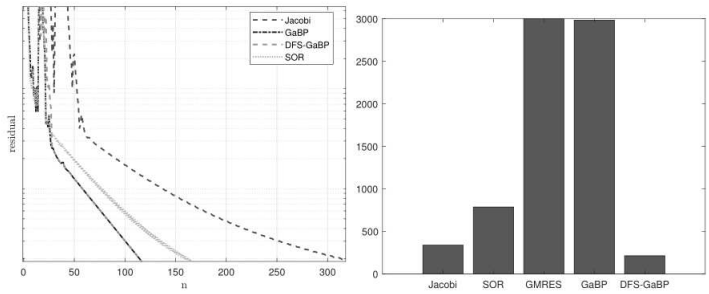
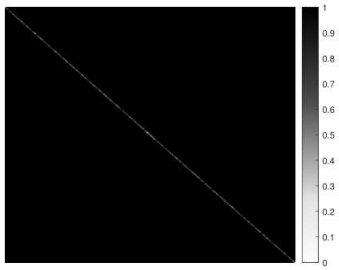


图 4 迭代次数 (纵坐标为 $\|\mathbf{x}^{(n)}-\mathbf{x}^{(n-1)}\|_2$) 以及算法计算时间比较

在这个矩阵中, 通过收敛曲线我们可以发现 GaBP 算法的迭代速度较快, 但计算时间长, 但是对于 DFS-GaBP 算法而言, 其收敛曲线与 GaBP 算法相当, 但是计算时间比其他所有算法都要少, 因此在这一线性系统中, 我们的算法仍然是最优的.

4.3 随机构造的修正矩阵

我们运用程序自带的随机生成指令, 构造出一个 1000 阶且连通分量为 562 个的稀疏矩阵, 其零元素比例分块热图如下:



将这个 1000 阶的矩阵同样以 10^{-5} 为收敛阈值使用 Jacobi 迭代、SOR 迭代、GMRES 子空间法 (最大迭代步长为 20000)、原始 GaBP 算法以及我们改进之后的 DFS-GaBP 算法来进行求解比较, 所得的计算结果如下:

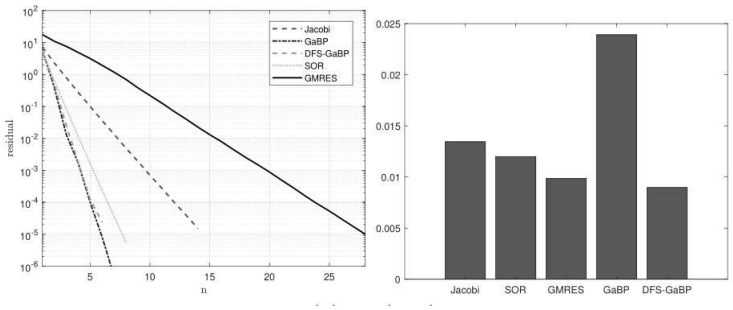


图 5 迭代次数 (纵坐标为 $\|\mathbf{x}^{(n)}-\mathbf{x}^{(n-1)}\|_2$) 以及算法计算时间比较

我们发现 DFS-GaBP 在这个矩阵中的表现情况无疑也是非常好的, 与 GaBP 一样, 经过较少的迭代次数就很快地接近我们的收敛阈值. 而且计算时间同样也是在这些算法中最少的.

通过上述几个实验我们发现, 在线性系统具有多个子系统的情况下, DFS-GaBP 算法的计算效率是优于原始 GaBP 算法的. 所以在处理大规模矩阵中, 比起 GaBP 算法, DFS-GaBP 算法的表现更好.

5 总结

GaBP 算法在求解大型稀疏矩阵上要优于传统的迭代法, 而 DFS-GaBP 算法在处理具有多个子系统的线性系统上具有一定的优势, 同时执行 DFS 所需要的计算量是可以被接受的. 我们证明了当整体系统满足收敛条件时, 所有的子系统也同样满足收敛条件, 因此可以在所有的子系统上应用 GaBP 算法. 并且我们选取了几个较大规模的矩阵来验证我们算法的性质, 可以看到 DFS-GaBP 算法和 GaBP 算法在这些系统中的收敛性质相当, 并且我们优化后的算法每次迭代所需的计算量要少于原始的 GaBP 算法.

参 考 文 献

[1] 徐树方, 高立, 张平文. 数值线性代数 (第 2 版)[M]. 北京: 北京大学出版社, 2013.

[2] 郝艳花. Jacobi 迭代法与 Gauss-Seidel 迭代法 [J]. 山西大同大学学报 (自然科学版), 2017, 33(05): 3–5.

[3] 何国良, 张文星, 赵熙乐. 共轭梯度法的直观解释 [J]. 大学数学, 2022, 38(01): 73–82.

[4] 郑汉垣. 大规模稀疏线性方程组求解的并行 GaBP 算法研究 [D]. 上海: 上海大学, 2014.

[5] Bickson D. Gaussian belief propagation: Theory and application[D]. Jerusalem: The Hebrew University of Jerusalem, 2009.

[6] Fanaskov V. Gaussian belief propagation solvers for nonsymmetric systems of linear equations[J]. SIAM Journal on Scientific Computing, 2022, 44(1): A77–A102.

[7] 孙涵, 黄元元, 高航, 等. 数据结构: 抽象建模、实现与应用 [M]. 北京: 机械工业出版社. 2020.

[8] Weiss Y, Freeman W T. Correctness of belief propagation in Gaussian graphical models of arbitrary topology[J]. Neural Computation, 2001, 13(10): 2173–2200.

[9] Malioutov D M, Johnson J K, Willsky A S. Walk-sums and belief propagation in Gaussian graphical models[J]. Journal of Machine Learning Research, 2006, 7: 2031–2064.

[10] Sherman A. Oil reservoir simulation challenge matrices IMPES simulation of a black oil model[DB/OL]. https://math.nist.gov/MatrixMarket/data/HarwellBoeing/sherman/sherman3.html, 1984.

[11] Velzen N V. Chemical process simulation[DB/OL]. https://sparse.tamu.edu/VanVelzen/std1_Jac3_db, 2006.

RESEARCH ON OPTIMIZATION OF GABP ALGORITHM BASED ON DFS

QIU De-zhi, CHEN Si-yuan

(*School of Mathematics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*)

Abstract: This paper investigates the acceleration of the Gaussian belief propagation (GaBP) algorithm in linear systems with multiple subsystems. By utilizing a depth-first search (DFS) algorithm to reduce the computational complexity of the GaBP algorithm in such linear systems, we obtain the DFS-GaBP algorithm. Meanwhile, we derive convergence results for the DFS-GaBP algorithm on the subsystems. Our computational results demonstrate that, compared to traditional algorithms, the DFS-GaBP algorithm exhibits superior performance in linear systems with multiple subsystems.

Keywords: gaussian belief propagation; depth-first search; subsystem; weak connected components

2010 MR Subject Classification: 15A06; 65F10