2024 年 8 月 计 算 数 学 第 46 卷第 3 期
Aug., 2024 MATHEMATICA NUMERICA SINICA Vol.46, No.3

doi: 10.12286/jssx.j2023-1128

基于分裂迭代算法求解多重线性 PageRank 问题*1)

唐舒婷 邓秀勤 刘冬冬2)
(广东工业大学, 数学与统计学院, 广州 510006)

摘 要

本文针对多重线性 PageRank 问题, 结合松弛技术, 提出了新的张量分裂算法, 并给出了相应的收敛性分析. 数值实验表明, 在适当选择松弛参数的情况下, 新算法具有较好的数值效果.

关键词: 多重线性 PageRank 问题; 张量分裂; 松弛算法.

MR (2010) 主题分类: 65F10, 65H10.

1. 引 言

PageRank 算法最初作为互联网网页重要度的计算方法. 1996 年由谷歌创始人 Page 和斯坦福大学的 Brin 提出 [1], 并用于谷歌搜索引擎的网页节点排序. 我们称 $(\alpha, P, \mathbf{v})$ 是一个 PageRank 问题, 它被定义为

$$\mathbf{x} = \alpha P\mathbf{x} + (1-\alpha)\mathbf{v}.$$

其中 $P = (p_{ij})$ 是一个列随机矩阵 (即: $\sum_{i=1}^{n} p_{ij} = 1, p_{ij} \geq 0, i, j \in \langle n\rangle$), $\langle n\rangle = \{1, 2, \cdots, n\}$. 概率 $\alpha \in (0,1)$, $\mathbf{v}$ 是随机向量 (即: $\sum_{i=1}^{n} v_i = 1, v_i \geq 0,\ i \in \langle n\rangle$), $\mathbf{x}$ 是特求的 PageRank 向量. 2015 年, Gleich 等 [2] 针对随机游走等实际应用问题以及黎等 [3] 关于高阶 Markov 链的工作, 提出了多重线性 PageRank 问题.

令 $\mathcal{P}$ 是一个 m 阶 n 维随机张量, 满足

$$p_{i_1 i_2 \cdots i_m} \geq 0, \sum_{i_1=1}^{n} p_{i_1 i_2 \cdots i_m} = 1,$$

概率 $\alpha \in (0,1)$, $\mathbf{v}$ 是一个随机向量, 则多重线性 PageRank 问题定义如下:

$$\mathbf{x} = \alpha\mathcal{P}\mathbf{x}^{m-1} + (1-\alpha)\mathbf{v}. \tag{1.1}$$

其中 $\mathbf{x}$ 是所求的随机向量 (被称为多重线性 PageRank 向量),

$$\mathcal{P}\mathbf{x}^{m-1} \equiv \sum_{i_2, i_3, \cdots, i_m = 1}^{n} p_{i i_2 \cdots i_m} x_{i_2} \cdots x_{i_m}.$$

* 2023 年 7 月 18 日收到.

1) 基金项目: 国家自然科学基金 (12101136), 广东省自然科学基金 (2023A1515011633), 广州市科技计划 '续航' 项目 (2024A04J2056), 广东省研究生教育创新计划项目 (2021SFKC030) 资助.

2) 通信作者: 刘冬冬, Email: ddliu@gdut.edu.cn.

值得注意的是, 多重线性 PageRank 问题同样可以转化为高阶 Markov 链模型

$$\mathbf{x} = \mathcal{P}_{\mathbf{v}}\mathbf{x}^{m-1},\ \mathcal{P}_{\mathbf{v}} = \alpha\mathcal{P} + (1-\alpha)\mathcal{V},\ (\mathcal{V})_{i_1 i_2 \cdots i_m} = v_{i_1}. \tag{1.2}$$

近年来, 许多学者深入研究了多重线性 PageRank 问题, 并且提出了许多相关理论 [4–6] 和算法. 目前针对多重线性 PageRank 问题的算法大致可以分为两类: 基于梯度的算法与非梯度算法. 对于基于梯度类算法, 例如 Gleich 等 [2] 首次将牛顿法应用到求解多重线性 PageRank 问题. 随后 Meini 等 [7] 以及郭等 [8] 分别提出了 Perron 牛顿迭代算法和多步牛顿法. 另外可以将多重线性 PageRank 问题转化为模型 (1.2), 利用相关牛顿法 [9,10] 进行求解. 涉及梯度计算的相关算法有着非常好的性能, 但是求解非对称情况时, 由于梯度计算困难, 会导致算法计算昂贵. 为此目前文献中大都采用张量半对称化或对称化的技术. 这种技术的应用尽管便于理论分析, 但在计算非对称张量模型的梯度时, 实际计算代价仍然很大. 因此设计非梯度类算法可以很好地解决这一困难. 例如: Gleich 等 [2] 提出的幂法和带位移幂法. 刘等 [11] 提出了求解模型 (1.2) 的松弛方法. 进一步, 学者们提出了一系列幂法加速算法: 二次外推法 [12], 带重启的外推法 [13], 移位定点法 [13]以及 Anderson 加速不动点迭代 [14] 等.

分裂算法是一类重要的非梯度算法, 在张量计算中有着非常重要的应用. 刘等 [15] 给出了张量的分裂算法并应用求解张量方程组和高阶 Markov 链模型. 本文将探索一种新的张量分裂算法, 对多重线性 PageRank 问题进行求解.

本文的主要贡献如下:

- 本文提出了新的求解多重线性 PageRank 问题的分裂方法, 并进一步给出四种特殊的分裂方法: 内外迭代法、Jacobi 迭代法、Gauss-Seidel 迭代法和逐次超松弛迭代法.
- 我们对所提出的算法给出了收敛性分析, 并通过数值实验说明所提出算法是有效性的.

本文的组织如下: 在第 2 章中, 介绍了相关的定义和符号等预备知识. 在第 3 章中, 我们给出了求解多重线性 PageRank 问题新的分裂算法. 第 4 章给出了算法的收敛性分析. 进一步, 在第 5 章, 我们设计了一些数值例子来说明所提算法的有效性. 最后在第 6 章中进行了总结.

2. 预备知识

首先我们给出本文中用到的符号和基本知识. 设 $\mathbb{R}$ 是实域, $\mathbb{R}^{[m,n]}$ 表示实域中所有 m 阶 n 维张量的集合, 特别地, $\mathbb{R}^{[2,n]}$ 和 $\mathbb{R}^n$ 分别表示 $n\times n$ 维实矩阵集和 n 维实向量集. $\|\cdot\|_1$ 表示 1− 范数. $\mathbf{e}$ 表示元素全为 1 的列向量 (即 $\mathbf{e} = [1,1,\cdots,1]^T$). 对于任意两个矩阵 $A = (a_{ij})_{n\times n}, B = (b_{ij})_{n\times n} \in \mathbb{R}^{[2,n]}$, $A \geq B$ 意味着 $a_{ij} \geq b_{ij}, i,j \in \langle n\rangle$. 一个有 n^m 个元素的 m 阶 n 维张量 $\mathcal{A}$ 定义为

$$\mathcal{A} = (a_{i_1\cdots i_m}), a_{i_1\cdots i_m} \in \mathbb{R}, i_j \in \langle n\rangle, \quad j = 1,\cdots,m.$$

若 $\mathcal{A}$ 满足

$$a_{i_1 i_2\cdots i_m} = \begin{cases} 1, & i_1 = \cdots = i_m, \\ 0, & \text{其他}, \end{cases}$$

则称 $\mathcal{A}$ 为单位张量.

定义 2.1. [5] 设 $\mathcal{A}\in\mathbb{R}^{[m,n]}$,$\mathbf{x}^{(l)}\in\mathbb{R}^n$ $(l=1,\cdots,m-1)$, 其中 $x_i^{(l)}$ 表示 $\mathbf{x}^{(l)}$ 的第 i 个元素. 定义 $\mathcal{A}\mathbf{x}^{(1)}\mathbf{x}^{(2)}\cdots\mathbf{x}^{(m-1)}$ 是 n 维向量, 它的第 i 个元素由以下公式给出

$$(\mathcal{A}\mathbf{x}^{(1)}\mathbf{x}^{(2)}\cdots\mathbf{x}^{(m-1)})_i=\sum_{i_2,i_3,\cdots,i_m\in\langle n\rangle}a_{ii_2\cdots i_m}x_{i_2}^{(1)}\cdots x_{i_m}^{(m-1)}.$$

如果 $\mathbf{x}^{(1)}=\mathbf{x}^{(2)}=\cdots=\mathbf{x}^{(k)}=\mathbf{y}(k\leq m-1)$, $\mathcal{A}\mathbf{x}^{(1)}\mathbf{x}^{(2)}\cdots\mathbf{x}^{(m-1)}$ 可改写为 $\mathcal{A}\mathbf{y}^k\mathbf{x}^{(k+1)}\cdots\mathbf{x}^{(m-1)}$. 特别地, 如果 $\mathbf{x}^{(1)}=\mathbf{x}^{(2)}=\cdots=\mathbf{x}^{(m-1)}=\mathbf{x}$, $\mathcal{A}\mathbf{x}^{(1)}\mathbf{x}^{(2)}\cdots\mathbf{x}^{(m-1)}$ 可改写为 $\mathcal{A}\mathbf{x}^{m-1}$.

对于张量 $\mathcal{A}\in\mathbb{R}^{[m,n]}$ 及向量 $\mathbf{x}\in\mathbb{R}^n$, 张量与向量乘积 $\mathcal{A}\mathbf{x}^{m-2}$ 定义为

$$(\mathcal{A}\mathbf{x}^{m-2})_{ij}=\sum_{i_3,\cdots,i_m=1}^{n}a_{iji_3\cdots i_m}x_{i_3}\cdots x_{i_m},\quad i,j=1,2,\cdots,n.$$

易知 $\mathcal{A}\mathbf{x}^{m-1}=\mathcal{A}\mathbf{x}^{m-2}\mathbf{x}$. 文 [11] 定义了

$$\mathcal{P}(\mathbf{x}^{m-1}-\mathbf{y}^{m-1})\equiv\mathcal{P}\mathbf{x}^{m-1}-\mathcal{P}\mathbf{y}^{m-1},\ \mathcal{P}(\mathbf{x}^{m-2}-\mathbf{y}^{m-2})\equiv\mathcal{P}\mathbf{x}^{m-2}-\mathcal{P}\mathbf{y}^{m-2}.$$

注意到式 (1.1) 可改写为

$$(\mathcal{I}-\alpha\mathcal{P}\mathbf{x}^{m-2})\mathbf{x}=(1-\alpha)\mathbf{v}.$$

若一个实方阵 $A=sI-B$, 实数 $s>0$ 且 $B\geq 0$ (即 B 为非负矩阵), 则 A 称为 Z 矩阵. 进一步, 满足 $s>\rho(B)$, 则 A 称为非奇异 M 矩阵, 其中 $\rho(B)\equiv\max\limits_{\lambda\in\delta(B)}\{|\lambda|\}$, $\delta(B)$ 表示 B 的所有特征值的集合.

定义 2.2. [16] 令 $O\in\mathbb{R}^{[2,n]}$ 是零矩阵, $A,M\in\mathbb{R}^{[2,n]}$ 是实矩阵, 若 M 是非奇异矩阵. 称 $A=M-N$ 为 A 的分裂, 进一步可称为 (i) 正则分裂, 若 $M^{-1}\geq O,\ N\geq O$; (ii) 弱正则分裂, 若 $M^{-1}\geq O,\ M^{-1}N\geq O$; (iii) 收敛的分裂, 若 $\rho(M^{-1}N)<1$.

多重线性 PageRank 问题解的存在性和唯一性理论是研究算法收敛性分析的重要基础. Gleich 等证明了多线性 PageRank 问题的解一定存在, 且当 $\alpha<\frac{1}{m-1}$ 时, 该问题有唯一解. 上述条件与 $\mathcal{P}$ 无关且易于检验, 但遗憾的是, 该条件所给出的 α 范围比较小, 而实际应用中 α 往往逼近于 1 . 因此, 黎等 [4,5,19] 基于插入参量的技术改进了唯一性条件. Fasino D 等 [6] 给出 3 阶张量的遍历系数, 基于此给出了 3 阶情况下的多线性 PageRank 问题解的唯一性条件. 值得注意的是, 理论上无法比较这些条件的好坏, 数值表现上这些条件也没有最优 [4]. 本文基于 Fasino D 等给出的唯一性条件, 对提出的算法给出收敛性分析. 令 $\mathcal{P}$ 是 3 阶 n 维随机张量, $\mathbb{S}_1=\{\mathbf{x}|\mathbf{x}\in\mathbb{R}^{[2,n]},\ \mathbf{x}\geq 0,\ \|\mathbf{x}\|_1=1\}$ 和 $\mathbb{Z}_1=\{\mathbf{x}|\mathbf{x}\in\mathbb{R}^{[2,n]},\ \mathbf{e}^T\mathbf{x}=0,\ \|\mathbf{x}\|_1=1\}$, 则高阶遍历系数定义如下:

$$\mathcal{T}(\mathcal{P})=\sup_{\mathbf{x},\mathbf{y}\in\mathbb{S}_1}\frac{\|\mathcal{P}\mathbf{x}\mathbf{x}-\mathcal{P}\mathbf{y}\mathbf{y}\|_1}{\|\mathbf{x}-\mathbf{y}\|_1},$$

或等价为

$$\mathcal{T}(\mathcal{P})=\max_{\mathbf{x}\in\mathbb{S}_1}\max_{\mathbf{y}\in\mathbb{Z}_1}\|\mathcal{P}\mathbf{x}\mathbf{y}+\mathcal{P}\mathbf{y}\mathbf{x}\|_1.$$

同样可以定义左遍历系数和右遍历系数如下:

$$\mathcal{T}_L(\mathcal{P})=\max_{\mathbf{x}\in\mathbb{S}_1}\max_{\mathbf{y}\in\mathbb{Z}_1}\|\mathcal{P}\mathbf{x}\mathbf{y}\|_1,$$

$$\mathcal{T}_R(\mathcal{P})=\max_{\mathbf{x}\in\mathbb{S}_1}\max_{\mathbf{y}\in\mathbb{Z}_1}\|\mathcal{P}\mathbf{y}\mathbf{x}\|_1.$$

Fasino D 等 [6] 给出了上述系数的具体表达式

$$\mathcal{T}_L(\mathcal{P}) = \frac{1}{2}\max_{jk_1k_2}\sum_i |p_{ijk_1} - p_{ijk_2}|,$$

$$\mathcal{T}_R(\mathcal{P}) = \frac{1}{2}\max_{j_1j_2k}\sum_i |p_{ij_1k} - p_{ij_2k}|,$$

$$\mathcal{T}(\mathcal{P}) = \frac{1}{2}\max_{jk_1k_2}\sum_i |p_{ijk_1} - p_{ijk_2} + p_{ik_1j} - p_{ik_2j}|.$$

进一步, 可知

$$0 \le \mathcal{T}_L(\mathcal{P}), \mathcal{T}_R(\mathcal{P}) \le 1, \quad 0 \le \mathcal{T}(\mathcal{P}) \le \mathcal{T}_L(\mathcal{P}) + \mathcal{T}_R(\mathcal{P}) \le 2.$$

3. 提出的算法

令 $\mathbf{x} \in \mathbb{R}^n$ 且 $\mathbf{x}_+ = \max(\mathbf{x}, 0)$ 为非零向量, 这里 $\max(\cdot, 0)$ 是 MATLAB 函数. 我们定义投影算子如下:

$$\mathrm{proj}(\mathbf{x}) = \frac{\mathbf{x}_+}{\|\mathbf{x}_+\|_1}.$$

易证 $\mathrm{proj}(\mathbf{x})$ 是随机向量. 令 $I - \alpha\mathcal{P}\mathbf{x}^{m-2} = M_{\mathbf{x}} - N_{\mathbf{x}}$ 是 (弱) 正则分裂, 则式 (1.1) 可改写为

$$M_{\mathbf{x}}\mathbf{x} = N_{\mathbf{x}}\mathbf{x} + \mathbf{b}, \ \mathbf{b} = (1-\alpha)\mathbf{v}. \tag{3.1}$$

基于式 (3.1), 结合松弛技术, 我们设计如下新的分裂迭代算法.

算法 1 张量分裂迭代 (TARS)

1: 选择初始向量 $\mathbf{x}_0 \in \mathbb{S}_1$, 令 $k = 0$;
2: 更新序列 $\mathbf{y}_{k+1} = M_{\mathbf{x}_k}^{-1}N_{\mathbf{x}_k}\mathbf{x}_k + M_{\mathbf{x}_k}^{-1}\mathbf{b}$, $\hat{\mathbf{x}}_{k+1} = \gamma\mathbf{y}_{k+1} + (1-\gamma)\mathbf{x}_k$, $\mathbf{x}_{k+1} = \mathrm{proj}(\hat{\mathbf{x}}_{k+1})$;
3: 若 $\mathbf{x}_{k+1}$ 满足终止条件, 则终止, 否则令 $k = k+1$, 返回步骤 2.

当式 (3.1) 取不同分裂时, 我们可以得到不同的分裂算法. 本文将给出四种数值效果较好的分裂迭代算法: 内外迭代、Jacobi 迭代、Gauss-Seidel 迭代和逐次超松弛迭代. 当 $M_{\mathbf{x}_{k-1}} = (I - \beta\mathcal{P}\mathbf{x}_{k-1}^{m-2}), N_{\mathbf{x}_{k-1}} = (\alpha-\beta)\mathcal{P}\mathbf{x}_{k-1}^{m-2}$, 其中 $\beta \in (0, \alpha)$. 因此我们得到算法 2 (内外迭代).

算法 2 内外迭代 (TARS-IO)

1: 给定随机张量 $\mathcal{P}$, α, $\beta \in (0, \alpha)$, 最大迭代步数 $k_{\max}$, 停机误差 ε, 松弛因子 $\gamma > 0$ 以及初始向量 $\mathbf{x}_0 \in \mathbb{S}_1$, 令 $k = 1$;
2: 当 $k < k_{\max}$;
3: $(I - \beta\mathcal{P}\mathbf{x}_{k-1}^{m-2})\mathbf{y}_k = (\alpha-\beta)\mathcal{P}\mathbf{x}_{k-1}^{m-2}\mathbf{x}_{k-1} + \mathbf{b}, \hat{\mathbf{x}}_k = \gamma\mathbf{y}_k + (1-\gamma)\mathbf{x}_{k-1}, \mathbf{x}_k = \mathrm{proj}(\hat{\mathbf{x}}_k)$;
4: 若 $\left\|\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x}_k^{m-1} - \mathbf{b}\right\|_1 < \varepsilon$, 则终止并且输出 $\mathbf{x}_k$;
5: $k = k+1$, 返回步骤 2.

令 $U_{\mathbf{x}}$ 是矩阵 $\mathcal{P}\mathbf{x}^{m-2}$ 的严格上三角矩阵, $D_{\mathbf{x}}$, $L_{\mathbf{x}}$ 分别是对角矩阵和下三角矩阵, 使得 $\mathcal{P}\mathbf{x}^{m-2} = D_{\mathbf{x}} + U_{\mathbf{x}} + L_{\mathbf{x}}$. 令 $\tilde{M}_{\mathbf{x}} = I - \alpha D_{\mathbf{x}}$, $\tilde{N}_{\mathbf{x}} = \alpha(L_{\mathbf{x}} + U_{\mathbf{x}})$, 那么可得到算法 3 (Jacobi 迭代).

算法 3 Jacobi 迭代 (TARS-JCB)

1: 给定随机张量 $\mathcal{P}$, α, 最大迭代步数 $k_{\max}$, 停机误差 ε, 松弛因子 $\gamma > 0$ 以及初始向量 $\mathbf{x}_0 \in \mathbb{S}_1$, 令 $k = 1$;
2: 当 $k < k_{\max}$;
3: $\mathbf{y}_k = \alpha(I - \alpha D_{\mathbf{x}_{k-1}})^{-1}(L_{\mathbf{x}_{k-1}} + U_{\mathbf{x}_{k-1}})\mathbf{x}_{k-1} + (I - \alpha D_{\mathbf{x}_{k-1}})^{-1}\mathbf{b}$, $\hat{\mathbf{x}}_k = \gamma\mathbf{y}_k + (1-\gamma)\mathbf{x}_{k-1}$, $\mathbf{x}_k = \mathrm{proj}(\hat{\mathbf{x}}_k)$;
4: 若 $\left\|\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x}_k^{m-1} - \mathbf{b}\right\|_1 < \varepsilon$, 则终止并且输出 $\mathbf{x}_k$;
5: $k = k + 1$, 返回步骤 2.

令 $\hat{M}_{\mathbf{x}} = I - \alpha D_{\mathbf{x}} - \alpha U_{\mathbf{x}}$, $\hat{N}_{\mathbf{x}} = \alpha L_{\mathbf{x}}$, 那么可得到算法 4 (Gauss-Seidel 迭代).

算法 4 Gauss-Seidel 迭代 (TARS-GS)

1: 给定随机张量 $\mathcal{P}$, α, 最大迭代步数 $k_{\max}$, 停机误差 ε, 松弛因子 $\gamma > 0$ 以及初始向量 $\mathbf{x}_0 \in \mathbb{S}_1$, 令 $k = 1$;
2: 当 $k < k_{\max}$;
3: $\mathbf{y}_k = \alpha(I - \alpha D_{\mathbf{x}_{k-1}} - \alpha U_{\mathbf{x}_{k-1}})^{-1}L_{\mathbf{x}_{k-1}}\mathbf{x}_{k-1} + (I - \alpha D_{\mathbf{x}_{k-1}} - \alpha U_{\mathbf{x}_{k-1}})^{-1}\mathbf{b}$, $\hat{\mathbf{x}}_k = \gamma\mathbf{y}_k + (1-\gamma)\mathbf{x}_{k-1}$, $\mathbf{x}_k = \mathrm{proj}(\hat{\mathbf{x}}_k)$;
4: 若 $\left\|\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x}_k^{m-1} - \mathbf{b}\right\|_1 < \varepsilon$, 则终止并且输出 $\mathbf{x}_k$;
5: $k = k + 1$, 返回步骤 2.

令 $\bar{M}_{\mathbf{x}} = \frac{1}{\omega}(I - \alpha D_{\mathbf{x}} - \omega\alpha U_{\mathbf{x}})$, $\bar{N}_{\mathbf{x}} = \frac{1}{\omega}(\omega\alpha L_{\mathbf{x}} + (1-\omega)(I - \alpha D_{\mathbf{x}}))$, 那么可得算法 5 (逐次超松弛迭代).

算法 5 逐次超松弛迭代 (TARS-SOR)

1: 给定随机张量 $\mathcal{P}$, α, ω, 最大迭代步数 $k_{\max}$, 停机误差 ε, 松弛因子 $\gamma > 0$ 以及初始向量 $\mathbf{x}_0 \in \mathbb{S}_1$, 令 $k = 1$;
2: 当 $k < k_{\max}$;
3: $\mathbf{y}_k = (I - \alpha D_{\mathbf{x}_{k-1}} - \omega\alpha U_{\mathbf{x}_{k-1}})^{-1}(\omega\alpha L_{\mathbf{x}_{k-1}} + (1-\omega)(I - \alpha D_{\mathbf{x}_{k-1}}))\mathbf{x}_{k-1} + \omega(I - \alpha D_{\mathbf{x}_{k-1}} - \omega\alpha U_{\mathbf{x}_{k-1}})^{-1}\mathbf{b}$, $\hat{\mathbf{x}}_k = \gamma\mathbf{y}_k + (1-\gamma)\mathbf{x}_{k-1}$, $\mathbf{x}_k = \mathrm{proj}(\hat{\mathbf{x}}_k)$;
4: 若 $\left\|\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x}_k^{m-1} - \mathbf{b}\right\|_1 < \varepsilon$, 则终止并且输出 $\mathbf{x}_k$;
5: $k = k + 1$, 返回步骤 2.

注 3.1. 若 $\mathbf{x}_k \in \mathbb{S}_1$, 则 $\mathcal{P}\mathbf{x}_k^{m-2}$ 是列随机矩阵, 且当 $0 < \alpha < 1$ 时, $I - \alpha\mathcal{P}\mathbf{x}_k^{m-2}$ 是非奇异 M 矩阵.

注 3.2. 事实上, 当 $m = 2$, 算法 2 可以退化到由文 [17] 所提出的内外迭代法, 算法 3、4、5 可以退化到由文 [18] 所提出的 Jacobi 迭代法、Gauss-Seidel 迭代法和逐次超松弛迭代法.

4. 收敛性分析

本文将基于 Fasino D 等给出的 3 阶多重线性 PageRank 问题解的唯一性条件, 对提出的算法给出收敛性分析. 首先介绍下面几个引理.

引理 4.1. (见 [6], 推论 8.1) 如果 $\alpha\mathcal{T}(\mathcal{P})<1$, 则多重线性 PageRank 问题 (1.1) 有唯一的解.

根据遍历系数的定义, 我们很容易得到以下引理

引理 4.2. 令 $\mathcal{P}\in\mathbb{R}^{[3,n]}$ 为随机张量, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ 是 n 维随机向量, 则下列不等式成立

$$\left\|\mathcal{P}(\mathbf{x}^2-\mathbf{y}^2)\right\|_1\le\mathcal{T}(\mathcal{P})\left\|\mathbf{x}-\mathbf{y}\right\|_1,$$
$$\left\|\mathcal{P}\mathbf{z}(\mathbf{x}-\mathbf{y})\right\|_1\le\mathcal{T}_L(\mathcal{P})\left\|\mathbf{x}-\mathbf{y}\right\|_1,$$
$$\left\|\mathcal{P}(\mathbf{x}-\mathbf{y})\mathbf{z}\right\|_1\le\mathcal{T}_R(\mathcal{P})\left\|\mathbf{x}-\mathbf{y}\right\|_1.$$

引理 4.3. [11] 令 $\hat{\mathbf{c}}=(\hat{c}_i)\in\mathbb{R}^n$, 且 $\sum_{i=1}^n\hat{c}_i=1,\mathbf{y}=(y_i)$, 其中 $\hat{\mathbf{c}}$ 和 $\mathbf{y}$ 都是随机向量. 如果 $\mathbf{c}=\mathrm{proj}(\hat{\mathbf{c}})$, 则 $\left\|\mathbf{c}-\mathbf{y}\right\|_1\le\left\|\hat{\mathbf{c}}-\mathbf{y}\right\|_1$.

引理 4.4. 令 $\hat{\mathbf{c}}=(\hat{c}_i)\in\mathbb{R}^n$, $\mathbf{y}=(y_i)$ 是随机向量, 如果 $\mathbf{c}=\mathrm{proj}(\hat{\mathbf{c}})$, 则 $\left\|\mathbf{c}-\mathbf{y}\right\|_1\le 2\left\|\hat{\mathbf{c}}-\mathbf{y}\right\|_1$.

证明. 因为 $\mathbf{y}$ 是随机向量, 故 $\left\|\mathbf{y}\right\|_1=1$, 由 $\mathbf{c}=\mathrm{proj}(\hat{\mathbf{c}})=\frac{\hat{\mathbf{c}}_+}{\left\|\hat{\mathbf{c}}_+\right\|_1}$, 得

$$\begin{aligned}
\left\|\mathbf{c}-\mathbf{y}\right\|_1=\left\|\mathrm{proj}(\hat{\mathbf{c}})-\mathbf{y}\right\|_1&=\left\|\frac{\hat{\mathbf{c}}_+}{\left\|\hat{\mathbf{c}}_+\right\|_1}-\mathbf{y}\right\|_1\\
&=\frac{\left\|(1-\left\|\hat{\mathbf{c}}_+\right\|_1)\hat{\mathbf{c}}_++\left\|\hat{\mathbf{c}}_+\right\|_1(\hat{\mathbf{c}}_+-\mathbf{y})\right\|_1}{\left\|\hat{\mathbf{c}}_+\right\|_1}\\
&\le\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1+|1-\left\|\hat{\mathbf{c}}_+\right\|_1|\\
&=2\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1+|1-\left\|\hat{\mathbf{c}}_+\right\|_1|-\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1\\
&\le 2\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1+|1-\left\|\hat{\mathbf{c}}_+\right\|_1|-|\left\|\hat{\mathbf{c}}_+\right\|_1-\left\|\mathbf{y}\right\|_1|\\
&=2\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1+|1-\left\|\hat{\mathbf{c}}_+\right\|_1|-|\left\|\hat{\mathbf{c}}_+\right\|_1-1|\\
&=2\left\|\hat{\mathbf{c}}_+-\mathbf{y}\right\|_1\\
&\le 2\left\|\hat{\mathbf{c}}-\mathbf{y}\right\|_1.
\end{aligned}$$

□

引理 4.5. 令 $I-\alpha\mathcal{P}\mathbf{x}^{m-2}=M_{\mathbf{x}}-N_{\mathbf{x}}$ 是 (弱) 正则分裂, 则 $\left\|M_{\mathbf{x}}^{-1}\right\|_1\le\frac{1}{1-\alpha}$.

证明. 由 $I-\alpha\mathcal{P}\mathbf{x}^{m-2}=M_{\mathbf{x}}-N_{\mathbf{x}}$, 得 $M_{\mathbf{x}}^{-1}+M_{\mathbf{x}}^{-1}N_{\mathbf{x}}=I+\alpha M_{\mathbf{x}}^{-1}\mathcal{P}\mathbf{x}^{m-2}$, $M_{\mathbf{x}}^{-1}\ge 0$, $M_{\mathbf{x}}^{-1}N_{\mathbf{x}}\ge 0$, 进一步, 我们得到

$$\left\|M_{\mathbf{x}}^{-1}\right\|_1\le\left\|I+\alpha M_{\mathbf{x}}^{-1}\mathcal{P}\mathbf{x}^{m-2}\right\|_1\le 1+\alpha\left\|M_{\mathbf{x}}^{-1}\right\|_1.$$

因此

$$\left\|M_{\mathbf{x}}^{-1}\right\|_1\le\frac{1}{1-\alpha}.$$

定理 4.1. 令 $\mathcal{P}\in\mathbb{R}^{[3,n]}$ 是随机张量, 满足 $\alpha\mathcal{T}(\mathcal{P})<1$, 则对任意初始向量 $\mathbf{x}_0\in\mathbb{S}_1$, 当 $\eta_{\mathbf{x}_k}<\frac{1}{2}$ 且 $\frac{1}{2(1-\eta_{\mathbf{x}_k})}<\gamma<\frac{3}{2(\eta_{\mathbf{x}_k}+1)}$ 时, 由算法 1 生成的序列 $\{\mathbf{x}_k\}$ 收敛, 且

$$\left\|\mathbf{x}_k-\mathbf{x}\right\|_1\le\kappa_{\mathbf{x}_{k-1}}\cdots\kappa_{\mathbf{x}_0}\left\|\mathbf{x}_0-\mathbf{x}\right\|_1,$$

其中 $\mathbf{x}$ 是式 (1.1) 的解, $\eta_{\mathbf{x}_k}=\left\|M_{\mathbf{x}_k}^{-1}N_{\mathbf{x}_k}\right\|_1+\alpha\mathcal{T}_L(\mathcal{P})\left\|M_{\mathbf{x}_k}^{-1}\right\|_1$, $\kappa_{\mathbf{x}_k}=2(\gamma\eta_{\mathbf{x}_k}+|1-\gamma|)$.

证明. 由算法 1 的迭代格式可知

$$\begin{cases} M_{\mathbf{x}_k}\mathbf{y}_{k+1} = N_{\mathbf{x}_k}\mathbf{x}_k + \mathbf{b}, \\ M_{\mathbf{x}}\mathbf{x} = N_{\mathbf{x}}\mathbf{x} + \mathbf{b}. \end{cases}$$

进一步得

$$M_{\mathbf{x}_k}(\mathbf{y}_{k+1} - \mathbf{x}) + (M_{\mathbf{x}_k} - M_{\mathbf{x}})\mathbf{x} = N_{\mathbf{x}_k}(\mathbf{x}_k - \mathbf{x}) + (N_{\mathbf{x}_k} - N_{\mathbf{x}})\mathbf{x},$$

将 $\hat{\mathbf{x}}_{k+1} = \gamma\mathbf{y}_{k+1} + (1-\gamma)\mathbf{x}_k$ 代入上式, 令 $\mathbf{e}_{k+1} = \mathbf{x}_{k+1} - \mathbf{x}, \hat{\mathbf{e}}_{k+1} = \hat{\mathbf{x}}_{k+1} - \mathbf{x}$, 得

$$M_{\mathbf{x}_k}(\hat{\mathbf{e}}_{k+1} - \mathbf{e}_k + \gamma\mathbf{e}_k) + \gamma(M_{\mathbf{x}_k} - M_{\mathbf{x}})\mathbf{x} = \gamma N_{\mathbf{x}_k}\mathbf{e}_k + \gamma(N_{\mathbf{x}_k} - N_{\mathbf{x}})\mathbf{x}.$$

即

$$\begin{aligned} M_{\mathbf{x}_k}(\hat{\mathbf{e}}_{k+1} - \mathbf{e}_k + \gamma\mathbf{e}_k) &= \gamma N_{\mathbf{x}_k}\mathbf{e}_k + \gamma(N_{\mathbf{x}_k} - N_{\mathbf{x}})\mathbf{x} - \gamma(M_{\mathbf{x}_k} - M_{\mathbf{x}})\mathbf{x} \\ &= \gamma N_{\mathbf{x}_k}\mathbf{e}_k + \gamma(M_{\mathbf{x}_k} - M_{\mathbf{x}} + \alpha\mathcal{P}\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x})\mathbf{x} - \gamma(M_{\mathbf{x}_k} - M_{\mathbf{x}})\mathbf{x} \\ &= \gamma N_{\mathbf{x}_k}\mathbf{e}_k + \gamma\alpha\mathcal{P}\mathbf{x}\mathbf{e}_k. \end{aligned}$$

因此

$$\hat{\mathbf{e}}_{k+1} = \gamma M_{\mathbf{x}_k}^{-1}N_{\mathbf{x}_k}\mathbf{e}_k + \gamma\alpha M_{\mathbf{x}_k}^{-1}\mathcal{P}\mathbf{x}\mathbf{e}_k + \mathbf{e}_k - \gamma\mathbf{e}_k. \tag{4.1}$$

结合引理 4.2 得

$$\|\hat{\mathbf{e}}_{k+1}\|_1 \le \gamma \left\|M_{\mathbf{x}_k}^{-1}N_{\mathbf{x}_k}\right\|_1 \|\mathbf{e}_k\|_1 + \gamma\alpha\mathcal{T}_L(\mathcal{P}) \left\|M_{\mathbf{x}_k}^{-1}\right\|_1 \|\mathbf{e}_k\|_1 + |1-\gamma| \|\mathbf{e}_k\|_1. \tag{4.2}$$

易知 $\kappa_{\mathbf{x}_k} < 1$. 由式 (4.2) 和引理 4.4, 得

$$\|\mathbf{e}_{k+1}\|_1 \le \kappa_{\mathbf{x}_k} \|\mathbf{e}_k\|_1 \le \kappa_{\mathbf{x}_k}\kappa_{\mathbf{x}_{k-1}} \cdots \kappa_{\mathbf{x}_0} \|\mathbf{e}_0\|_1.$$

故序列 $\{\mathbf{x}_k\}$ 收敛到式 (1.1) 的唯一解, 证毕. □

注 4.1. 由于定理 4.1 中, $\kappa_{\mathbf{x}_k}$ 与 $\mathbf{x}_k$ 有关, 在实际问题中很难验证, 因此针对具体的算法 2 – 5, 我们给出可验证的收敛性分析.

定理 4.2. 令 $\mathcal{P}$ 是 3 阶 n 维随机张量, 满足 $\beta t < 1$, 则对任意初始向量 $\mathbf{x}_0$, 当 $\alpha \in (0, \frac{1}{t}), \gamma \in (\frac{2\beta t}{1+\alpha t}, \frac{2}{1+\alpha t})$ 时, 由算法 2 生成的序列 $\{\mathbf{x}_k\}$ 收敛, 且

$$\|\mathbf{x}_k - \mathbf{x}\|_1 \le \varsigma^k \|\mathbf{x}_0 - \mathbf{x}\|_1,$$

其中 $\mathbf{x}$ 是式 (1.1) 的解, $t = \mathcal{T}_L(\mathcal{P}) + \mathcal{T}_R(\mathcal{P})$, $\varsigma = \frac{\beta\mathcal{T}_R(\mathcal{P}) + |\gamma\alpha - \beta|\mathcal{T}(\mathcal{P}) + |1-\gamma|}{1-\beta\mathcal{T}_L(\mathcal{P})}$.

证明. 由 $\beta t < 1$ 可推出 $\alpha\mathcal{T}(\mathcal{P}) < 1$, 故式 (1.1) 有唯一的解 $\mathbf{x}$. 由算法 1 知算法 2 可以写成形式: $M_{\mathbf{x}_k}\mathbf{y}_k = N_{\mathbf{x}_{k-1}}\mathbf{x}_{k-1} + \mathbf{b}$, 其中 $I - \alpha\mathcal{P}\mathbf{x}_{k-1} = M_{\mathbf{x}_{k-1}} - N_{\mathbf{x}_{k-1}}$, $M_{\mathbf{x}_{k-1}} = (I - \beta\mathcal{P}\mathbf{x}_{k-1})$, $N_{\mathbf{x}_{k-1}} = (\alpha - \beta)\mathcal{P}\mathbf{x}_{k-1}$, 即

$$(I - \beta\mathcal{P}\mathbf{x}_{k-1})(\hat{\mathbf{x}}_k - (1-\gamma)\mathbf{x}_{k-1}) = \gamma(\alpha - \beta)\mathcal{P}\mathbf{x}_{k-1}^2 + \gamma\mathbf{b}.$$

整理得:

$$\hat{\mathbf{x}}_k = \beta\mathcal{P}\mathbf{x}_{k-1}\hat{\mathbf{x}}_k + (\gamma\alpha - \beta)\mathcal{P}\mathbf{x}_{k-1}^2 + (1-\gamma)\mathbf{x}_{k-1} + \gamma\mathbf{b}. \tag{4.3}$$

由于 $\mathbf{x}$ 是式 (1.1) 的解, 则

$$\mathbf{x} = \beta\mathcal{P}\mathbf{x}^2 + (\gamma\alpha - \beta)\mathcal{P}\mathbf{x}^2 + (1-\gamma)\mathbf{x} + \gamma\mathbf{b}. \tag{4.4}$$

令 $\hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}, \mathbf{e}_k = \mathbf{x}_k - \mathbf{x}$, 式 (4.3) 减式 (4.4) 得

$$\hat{\mathbf{e}}_k = \beta\mathcal{P}\mathbf{x}_{k-1}\hat{\mathbf{e}}_k + \beta\mathcal{P}\mathbf{e}_{k-1}\mathbf{x} + (\gamma\alpha - \beta)(\mathcal{P}\mathbf{x}_{k-1}^2 - \mathcal{P}\mathbf{x}^2) + (1-\gamma)\mathbf{e}_{k-1}. \tag{4.5}$$

由 (4.3) 得

$$\begin{aligned}\sum_{i=1}^{n}((I - \beta\mathcal{P}\mathbf{x}_{k-1})\hat{\mathbf{x}}_k)_i &= (\gamma\alpha - \beta)\sum_{i=1}^{n}\sum_{i_2,i_3=1}^{n}\mathcal{P}_{ii_2i_3}x_{k-1,i_2}x_{k-1,i_3} + (1-\gamma)\sum_{i=1}^{n}x_{k-1,i} + \gamma\sum_{i=1}^{n}b_i\\ &= \gamma\alpha - \beta + 1 - \gamma + \gamma(1-\alpha)\\ &= 1 - \beta,\end{aligned}$$

因此

$$\begin{aligned}1 - \beta &= \sum_{i=1}^{n}\hat{x}_{k,i} - \beta\sum_{i=1}^{n}\sum_{i_2,i_3=1}\mathcal{P}_{ii_2i_3}\hat{x}_{k,i_2}x_{k-1,i_3}\\ &= \sum_{i=1}^{n}\hat{x}_{k,i} - \beta\sum_{i=1}^{n}\hat{x}_{k,i},\end{aligned} \tag{4.6}$$

故

$$\sum_{i=1}^{n}\hat{x}_{k,i} = 1, \sum_{i=1}^{n}\hat{e}_{k,i} = 0.$$

结合引理 4.2 和式 (4.5), 可得

$$\|\hat{\mathbf{e}}_k\|_1 \le \beta\mathcal{T}_L(\mathcal{P})\|\hat{\mathbf{e}}_k\|_1 + \beta\mathcal{T}_R(\mathcal{P})\|\mathbf{e}_{k-1}\|_1 + |\gamma\alpha - \beta|\,\mathcal{T}(\mathcal{P})\|\mathbf{e}_{k-1}\|_1 + |1-\gamma|\,\|\mathbf{e}_{k-1}\|_1.$$

则

$$\|\hat{\mathbf{e}}_k\|_1 \le \frac{\beta\mathcal{T}_R(\mathcal{P}) + |\gamma\alpha - \beta|\,\mathcal{T}(\mathcal{P}) + |1-\gamma|}{1 - \beta\mathcal{T}_L(\mathcal{P})}\|\mathbf{e}_{k-1}\|_1,$$

结合引理 4.3 及式 (4.6), 得

$$\|\mathbf{e}_k\|_1 \le \varsigma\|\mathbf{e}_{k-1}\|_1 \le \varsigma^k\|\mathbf{e}_0\|_1.$$

由已知条件, 可得 $\varsigma < 1$, 证毕. □

定理 4.3. 令 $\mathcal{P}$ 是 3 阶 n 维随机张量, $0 < \alpha < \frac{1}{2\mathcal{T}(\mathcal{P})+1}$, 则对任意初始向量 $\mathbf{x}_0$, 当 $\frac{1-\alpha}{2(1-\alpha-\alpha\mathcal{T}(\mathcal{P}))} < \gamma < \frac{3(1-\alpha)}{2(1-\alpha+\alpha\mathcal{T}(\mathcal{P}))}$ 时, 由算法 3 生成的序列 $\{\mathbf{x}_k\}$ 收敛, 且

$$\|\mathbf{x}_k - \mathbf{x}\|_1 \le \delta^k\|\mathbf{x}_0 - \mathbf{x}\|_1,$$

其中 $\mathbf{x}$ 是式 (1.1) 的解, $\delta = 2(\gamma\frac{\alpha+\alpha\mathcal{T}(\mathcal{P})}{1-\alpha} + |1-\gamma|)$.

证明. 由 $\alpha \in (0, \frac{1}{2\mathcal{T}(\mathcal{P})+1})$, 可知 $\alpha\mathcal{T}(\mathcal{P}) < 1$, 故 $\mathbf{x}$ 是式 (1.1) 的唯一解. 由 $\tilde{M}_{\mathbf{x}_k} = I - \alpha D_{\mathbf{x}_k}, \tilde{N}_{\mathbf{x}_k} = \alpha(L_{\mathbf{x}_k} + U_{\mathbf{x}_k})$, 结合式 (4.1), 得

$$\begin{aligned}\hat{\mathbf{e}}_{k+1} &= \gamma\tilde{M}_{\mathbf{x}_k}^{-1}\tilde{N}_{\mathbf{x}_k}\mathbf{e}_k + \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}\mathcal{P}\mathbf{x}\mathbf{e}_k + \mathbf{e}_k - \gamma\mathbf{e}_k\\ &= \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}(L_{\mathbf{x}_k} + U_{\mathbf{x}_k})\mathbf{e}_k + \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}\mathcal{P}\mathbf{x}\mathbf{e}_k + (1-\gamma)\mathbf{e}_k\\ &\le \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}\mathcal{P}(\mathbf{e}_k\mathbf{x}_k + \mathbf{x}\mathbf{e}_k) + (1-\gamma)\mathbf{e}_k\\ &= \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}\mathcal{P}(\mathbf{x}_k^2 - \mathbf{x}^2) + (1-\gamma)\mathbf{e}_k.\end{aligned} \tag{4.7}$$

结合引理 4.2 和引理 4.5 , 得

$$\|\hat{\mathbf{e}}_{k+1}\|_1 \leq \frac{\gamma\alpha\mathcal{T}(\mathcal{P})}{1-\alpha}\|\mathbf{e}_k\|_1 + |1-\gamma|\,\|\mathbf{e}_k\|_1,$$

结合式 (4.7) 和引理 4.4, 得

$$\|\mathbf{e}_{k+1}\|_1 \leq \delta\,\|\mathbf{e}_k\|_1 \leq \delta^{k+1}\,\|\mathbf{e}_1\|_1.$$

由已知条件易证 $\delta<1$, 故 $\{\mathbf{x}_k\}$ 收敛于 $\mathbf{x}$, 证毕. □

定理 4.4. 令 $\mathcal{P}$ 是 3 阶 n 维随机张量, $0<\alpha<\frac{1}{2\mathcal{T}(\mathcal{P})+1}$, $\mathbf{x}$ 是式 (1.1) 的解, 则对任意初始向量 $\mathbf{x}_0$, 当 $\frac{1-\alpha}{2(1-\alpha-\alpha\mathcal{T}(\mathcal{P}))}<\gamma<\frac{3(1-\alpha)}{2(1-\alpha+\alpha\mathcal{T}(\mathcal{P}))}$ 时, 由算法 4 生成的序列 $\{\mathbf{x}_k\}$ 收敛, 且

$$\|\mathbf{x}_k-\mathbf{x}\|_1 \leq \delta^k\,\|\mathbf{x}_0-\mathbf{x}\|_1.$$

证明. 证明过程类似定理 4.3. □

定理 4.5. 令 $\mathcal{P}$ 是 3 阶 n 维随机张量, $0<\alpha<\frac{1}{2\mathcal{T}(\mathcal{P})+1}$, 则对任意初始向量 $\mathbf{x}_0$, 当 $\frac{2}{3-\alpha(1+2\mathcal{T}(\mathcal{P}))}\leq\omega<1$ 且 $\frac{1}{2(1-\bar{\eta})}<\gamma<\frac{3}{2(\bar{\eta}+1)}$ 时, 由算法 5 生成的序列 $\{\mathbf{x}_k\}$ 收敛, 且

$$\|\mathbf{x}_k-\mathbf{x}\|_1 \leq \vartheta^k\,\|\mathbf{x}_0-\mathbf{x}\|_1.$$

其中 $\mathbf{x}$ 是式 (1.1) 的解, $\bar{\eta}=\frac{\frac{1-\omega}{\omega}+\alpha\mathcal{T}(\mathcal{P})}{1-\alpha}$,$\vartheta=2(\gamma\tilde{\eta}+|1-\gamma|)$.

证明. 由 $0<\alpha<\frac{1}{2\mathcal{T}(\mathcal{P})+1}$, 可知 $\alpha\mathcal{T}(\mathcal{P})<1$, 故 $\mathbf{x}$ 是式 (1.1) 的唯一解. 由 $\bar{M}_{\mathbf{x}_k}=\frac{1}{\omega}(I-\alpha D_{\mathbf{x}_k}-\omega\alpha U_{\mathbf{x}_k})$, $\bar{N}_{\mathbf{x}_k}=\frac{1}{\omega}(\omega\alpha L_{\mathbf{x}_k}+(1-\omega)(I-\alpha D_{\mathbf{x}_k}))$, 可得 $\|\bar{N}_{\mathbf{x}_k}\|_1\leq\frac{1-\omega}{\omega}+\alpha$, 结合式 (4.1), 得

$$\begin{aligned}
\hat{\mathbf{e}}_{k+1} &= \gamma\bar{M}_{\mathbf{x}_k}^{-1}\bar{N}_{\mathbf{x}_k}\mathbf{e}_k+\gamma\alpha\bar{M}_{\mathbf{x}_k}^{-1}\mathcal{P}\mathbf{x}\mathbf{e}_k+\mathbf{e}_k-\gamma\mathbf{e}_k\\
&= \frac{\gamma}{\omega}\bar{M}_{\mathbf{x}_k}^{-1}(\omega\alpha L_{\mathbf{x}_k}+(1-\omega)(I-\alpha D_{\mathbf{x}_k}))\mathbf{e}_k+\gamma\alpha\bar{M}_{\mathbf{x}_k}^{-1}\mathcal{P}\mathbf{x}\mathbf{e}_k+(1-\gamma)\mathbf{e}_k\\
&\leq \gamma\alpha\bar{M}_{\mathbf{x}_k}^{-1}\mathcal{P}(\mathbf{e}_k\mathbf{x}_k+\mathbf{x}\mathbf{e}_k)+\frac{\gamma(1-\omega)}{\omega}\bar{M}_{\mathbf{x}_k}^{-1}\mathbf{e}_k+(1-\gamma)\mathbf{e}_k\\
&= \gamma\alpha\tilde{M}_{\mathbf{x}_k}^{-1}\mathcal{P}(\mathbf{x}_k^2-\mathbf{x}^2)+\frac{\gamma(1-\omega)}{\omega}\bar{M}_{\mathbf{x}_k}^{-1}\mathbf{e}_k+(1-\gamma)\mathbf{e}_k.
\end{aligned}\tag{4.8}$$

结合引理 4.2 和引理 4.5 , 得

$$\begin{aligned}
\|\hat{\mathbf{e}}_{k+1}\|_1 &\leq \gamma\alpha\mathcal{T}(\mathcal{P})\left\|\bar{M}_{\mathbf{x}_k}^{-1}\right\|_1\|\mathbf{e}_k\|_1+\frac{\gamma(1-\omega)}{\omega}\left\|\bar{M}_{\mathbf{x}_k}^{-1}\right\|_1\|\mathbf{e}_k\|_1+|1-\gamma|\,\|\mathbf{e}_k\|_1\\
&\leq \gamma\frac{\frac{1-\omega}{\omega}+\alpha\mathcal{T}(\mathcal{P})}{1-\alpha}\|\mathbf{e}_k\|_1+|1-\gamma|\,\|\mathbf{e}_k\|_1\\
&= (\gamma\bar{\eta}+|1-\gamma|)\,\|\mathbf{e}_k\|_1.
\end{aligned}$$

由式 (4.8) 和引理 4.4, 得

$$\|\mathbf{e}_{k+1}\|_1 \leq \vartheta\,\|\mathbf{e}_k\|_1 \leq \vartheta^{k+1}\,\|\mathbf{e}_0\|_1.$$

由已知条件易证 $\vartheta<1$, 故 $\{\mathbf{x}_k\}$ 收敛于 $\mathbf{x}$, 证毕. □

注 4.2. 本文基于 Fasino D 等给出的多线性 PageRank 问题解的唯一性条件, 对所提出的算法进行收敛性分析. 值得注意的是, 基于其他文献中的唯一性条件 (可参考 [4,5]), 我们同样可以给出相应的收敛性分析, 在此不再赘述.

5. 数值实验和结果

本节中, 我们将通过数值实验来验证所提出算法的有效性. 本文将从迭代步数 (IT), 以秒为单位的迭代时间 (CPU), 误差 (err) 这三个方面来检验所提算法的效率. 其中, 误差 err 的定义如下:

$$\mathrm{err} = \left\|\mathbf{x}_k - \alpha\mathcal{P}\mathbf{x}_k^{m-1} - \mathbf{b}\right\|_1.$$

所有测试例子均在 MATLAB R2016b 中完成, 笔记本电脑: Hasee 神州 Z6-KP5GT, Intel(R) Core(TM) i4-29210M CPU @ 2.50 GHz: 操作系统是 Windows 10. 初始值 $\mathbf{x}_0 = \mathbf{v} = \frac{1}{n}\mathbf{e}$, 最大迭代数设置为 1000, 停机误差 $\varepsilon = 10^{-10}$. 选取 $\alpha = \{0.90, 0.95, 0.98, 0.99, 0.999, 1\}$ 来测试算法. $\beta \in (0, \alpha)$ 选取范围是 0.1 到 1, 间隔为 0.1. 松弛因子 γ 选取范围是 0.1 到 3, 间隔为 0.1, 以此获得最好的数值效果. 设置

$$D_{\mathbf{x}} = \frac{1}{2}\mathrm{diag}(\mathcal{P}\mathbf{x}^{m-2}),\ L_{\mathbf{x}} = \mathrm{tril}(\mathcal{P}\mathbf{x}^{m-2}, -1) + D_{\mathbf{x}},\ U_{\mathbf{x}} = -\mathrm{triu}(\mathcal{P}\mathbf{x}^{m-2}, 1).$$

对于式 (1.1) 中的随机张量 $\mathcal{P}$, 我们通过以下几种方式获取.

例 5.1. [4] 运用 MATLAB 中的 rand 函数, 生成若干个 3 阶 n 维随机张量 $\mathcal{P}$.

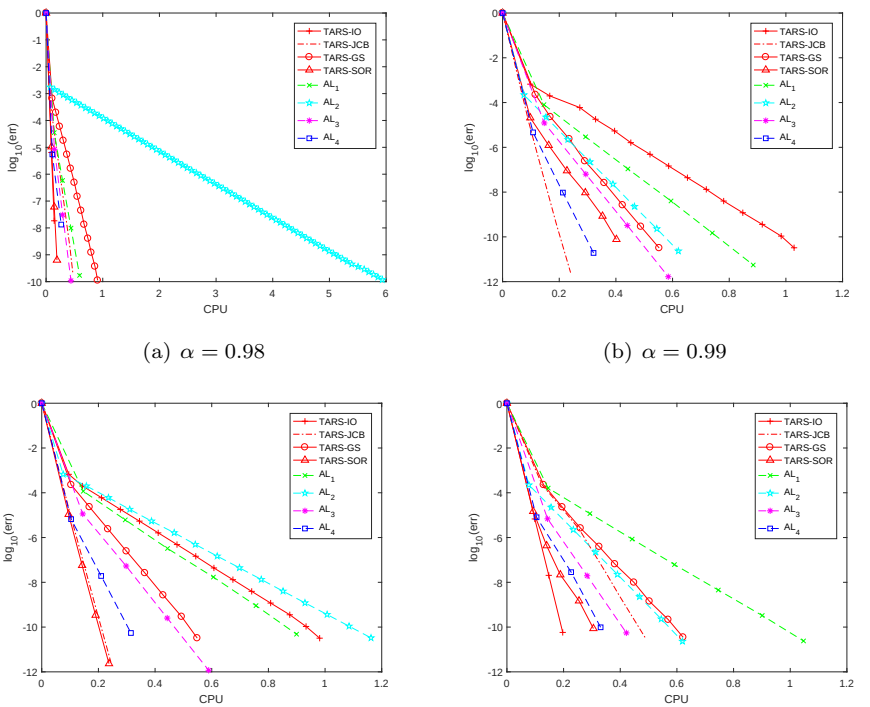
首先, 我们将本文提出的算法与文 [11] 中的算法 1–4 (分别记为 AL_1, AL_2, AL_3, AL_4) 进行比较. 在例 5.1 中, 我们给出了当 $n = 100$, 150 和 200 这三种情况, 数值实验结果如表 1 所示.

表 1 例 5.1 对比文 [11] 的数值结果

			TARS-IO		TARS-JCB		TARS-GS			TARS-SOR		AL_1		AL_2		AL_3		AL_4	
n	α	β	CPU	IT	CPU	IT	CPU	IT	ω	CPU	IT	CPU	IT	CPU	IT	CPU	IT	CPU	IT
100	0.9	1	0.014514	4	**0.014083**	20	0.014930	35	0.12	0.014157	4	0.024818	10	0.014131	8	0.024362	4	0.057224	77
	0.95	1.3	0.014990	15	0.014894	11	0.014937	35	0.45	0.014397	6	0.024286	4	0.014363	35	0.024654	4	0.056602	11
	0.98	1.5	0.014326	26	**0.014300**	20	0.014400	37	0.53	0.014448	7	0.023801	5	0.014589	35	0.024477	4	0.064786	19
	0.99	1.5	0.014444	26	0.015634	15	0.014624	15	0.56	**0.014035**	7	0.024034	7	0.014199	26	0.024390	4	0.065078	49
	0.999	1	0.014707	4	0.014566	15	0.014652	16	0.43	**0.014221**	6	0.024274	7	0.014252	83	0.025112	4	0.065734	11
	1	1	0.014738	4	0.015090	20	0.014619	15	0.43	**0.014203**	6	0.023581	5	0.014316	12	0.024409	4	0.057192	8
150	0.9	1.6	0.057544	34	0.057122	48	0.057224	77	0.51	**0.055820**	6	0.088626	5	0.061979	15	0.086806	4	0.218509	3
	0.95	1	0.056330	4	0.056249	8	0.056602	11	0.11	**0.056165**	4	0.087041	4	0.062323	19	0.085393	4	0.224913	4
	0.98	0.8	0.056925	11	0.057244	11	0.064786	19	1	**0.056445**	7	0.089598	5	0.062427	8	0.086063	4	0.224695	3
	0.99	1	0.057436	4	0.057043	4	0.065078	49	0.16	**0.055590**	4	0.089707	10	0.062117	80	0.087821	4	0.224714	3
	0.999	1	0.057312	4	0.058218	35	0.065734	11	0.31	**0.056367**	5	0.088513	6	0.062212	25	0.087365	4	0.224418	3
	1	0.9	**0.056284**	8	0.061711	11	0.057192	8	0.7	0.056352	7	0.086170	5	0.061815	11	0.087017	4	0.224174	3
200	0.9	0.7	0.142392	19	**0.139977**	19	0.170990	9	0.82	0.146257	7	0.292860	9	0.151884	19	0.289260	4	0.746584	3
	0.95	0.4	0.144240	19	0.162351	8	0.166620	7	0.11	**0.142541**	4	0.293014	10	0.150669	4	0.289826	3	0.767117	2
	0.98	0.9	0.143006	3	0.159464	8	0.169215	9	0.15	**0.142071**	4	0.290502	5	0.150898	76	0.294886	4	0.760298	3
	0.99	0.2	**0.143242**	15	0.144695	4	0.180945	8	0.63	0.146929	6	0.290584	6	0.151497	8	0.291844	4	0.761288	3
	0.999	0.7	0.143378	15	0.148364	4	0.158925	8	0.1	**0.142227**	4	0.289899	6	0.153134	15	0.291031	4	0.754382	3
	1	0.8	0.147666	3	0.171693	8	0.182178	9	0.4	**0.143634**	5	0.291357	7	0.152547	8	0.282004	3	0.754345	3

从表 1 中可以看出, 所提的算法所需的 CPU 时间比 $AL_1 - AL_4$ 算法要少. 在大多数情况下算法 TARS-SOR 所需的 CPU 时间较其他算法更少. 图 1 给出了当 $n = 200$ 时所测试算法在不同 α 下的 CPU 和 err 的关系曲线. 其中, err 的值取以 10 为底的对数, 节点代表迭代步

数. 图中表明相比于文 [11] 中的算法, 本文所提算法具有更小的误差. 其中, 当 $\alpha = 0.98, 0.999$ 和 1 时, 算法 TARS-SOR 的 CPU 和最小, 算法 TARS-JCB 在 $\alpha = 0.99$ 时表现最好.



(a) $\alpha = 0.98$ (b) $\alpha = 0.99$

(c) $\alpha = 0.999$ (d) $\alpha = 1$

图 1 例 5.1 中各算法在不同 α 下的 CPU 与 err 关系

例 5.2. [5, 11, 12] 令 Γ 表示具有节点集 $\mathbb{V} = \langle n \rangle = \mathbb{D} \cup \mathbb{P}$ 的有向图, 其中 $\mathbb{D}$ 和 $\mathbb{P}$ 分别表示悬挂节点和两两连通节点的集合. n_p 表示子集 $\mathbb{P}$ 中的节点数, 即 $\mathbb{P} = \{1, 2, \cdots, n_p\}$. 那么我们可以构造非负张量 $\mathcal{A}$

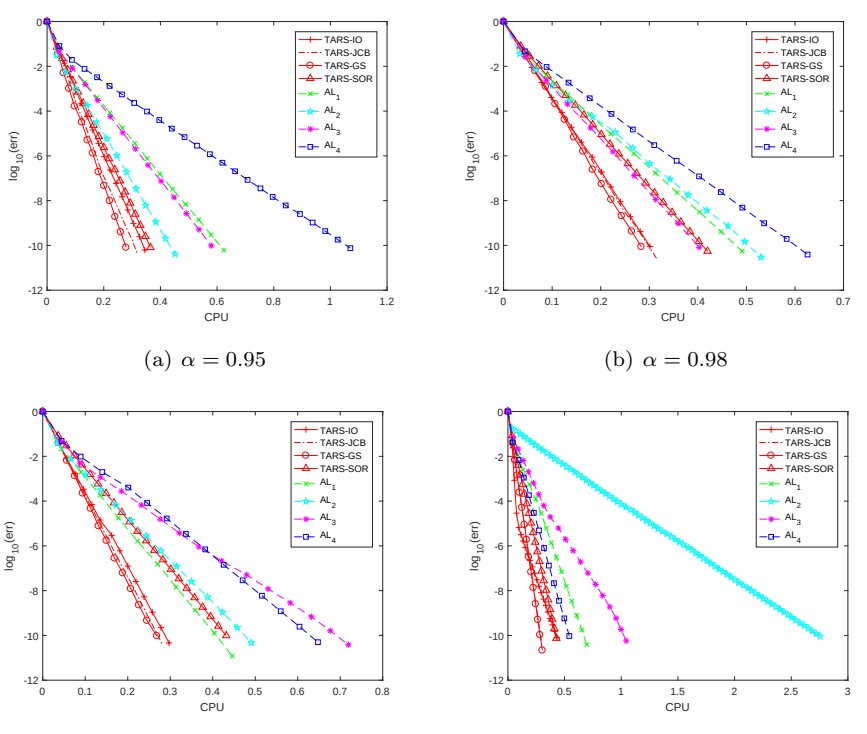
$$a_{i_1 i_2 \ldots i_l} = \begin{cases} a_{i_1 i_2 \ldots i_l}, & i_k \neq i_{k+1}, i_1, i_k, i_l \in \mathbb{P},\ k = 1, \ldots, l-1, \\ 0, & i_1 = i_2 (\text{或 } i_1 \in \mathbb{D}),\ i_k \neq i_{k+1}, i_k \in \mathbb{P},\ k = 2, \ldots, l-1, \\ \dfrac{1}{n}, & \text{其他}, \end{cases}$$

其中 $a_{i_1 i_2 \ldots i_l}$ 从 (0,1) 之间随机选取, 规范化 $p_{i_1 i_2 \ldots i_l} = \dfrac{a_{i_1 i_2 \ldots i_l}}{\sum_{i_1=1}^{n} a_{i_1 i_2 \ldots i_l}}$ 得到随机张量 $\mathcal{P} = p_{i_1 i_2 \ldots i_l}$.

在例 5.2 中, 我们给出了当 $n = 150$, $n_p = 60$, 90 和 120 这三种情况下的数值结果, 结果如表 2 所示. 当 $n_p = 60$ 时,TARS-IO 花费的 CPU 时间较少, 而当 $n_p = 90$ 或 120 时, 尽管 TARS-SOR 的迭代步数比其他算法多, 但所需 CPU 时间更少. 图 2 给出了当 $n_p = 120$ 时各算法在不同 α 下的 CPU 和 err 的关系曲线. 由图 2, 新算法具有更小的误差和 CPU 时间, 收敛速度更快, 这也说明了通过选取合适的参数后, 新算法比现有算法更有优势.

表 2　例 5.2 对比文 [11] 的数值结果

			TARS-IO		TARS-JCB		TARS-GS			TARS-SOR		AL_1		AL_2		AL_3		AL_4	
n_p	α	β	CPU	IT	CPU	IT	CPU	IT	ω	CPU	IT	CPU	IT	CPU	IT	CPU	IT	CPU	IT
60	0.9	0.1	0.0534028	41	**0.0530607**	42	0.0620506	25	0.94	0.0562526	36	0.0834071	27	0.0639469	66	0.0892419	34	0.21137815	33
	0.95	0.6	**0.0535916**	45	0.0607397	34	0.0610323	30	0.63	0.0565794	47	0.0862453	25	0.0638477	81	0.0886419	36	0.21235625	27
	0.98	0.2	**0.0526973**	129	0.0675064	42	0.0697749	31	0.28	0.0604276	60	0.0886126	34	0.0645803	42	0.0905563	17	0.20973255	59
	0.99	0.5	**0.0539799**	43	0.0558535	43	0.0685121	37	0.48	0.0604446	62	0.0898273	32	0.0648539	104	0.0829809	37	0.21405745	31
	0.999	0.4	**0.0531169**	45	0.0538151	43	0.0533077	37	0.98	0.0543651	62	0.0845698	43	0.0632201	96	0.0788035	68	0.21087425	62
	1	0.6	**0.0526258**	85	0.0526843	43	0.0736683	37	0.19	0.0532515	72	0.0878069	32	0.0633036	75	0.1070365	66	0.20762955	49
90	0.9	0.1	0.0692163	79	0.0780934	25	0.0717065	58	0.58	**0.0592819**	40	0.0814981	23	0.0657232	69	0.080474	48	0.253816555	48
	0.95	0.8	0.0704547	85	0.1183176	33	0.0850696	60	0.32	**0.065063**	53	0.0808947	26	0.0661521	56	0.0841014	24	0.250386455	13
	0.98	0.5	0.0677803	35	0.0755355	55	0.0729206	39	0.51	0.0673235	61	0.0828618	26	0.0655838	55	0.0838675	45	0.257287755	21
	0.99	0.9	0.0758312	28	0.0705211	54	0.0768998	41	0.81	**0.0633692**	61	0.0811415	41	0.0660505	414	0.0848214	64	0.250462855	11
	0.999	0.6	**0.0518806**	90	0.0656324	44	0.0599667	41	0.54	0.0591266	71	0.0807065	40	0.0648731	294	0.091268	73	0.248931955	73
	1	0.5	**0.052589**	38	0.0806161	46	0.0730648	41	0.34	0.0605891	75	0.081165	47	0.067416	66	0.0840651	67	0.247725855	64
120	0.9	0.5	0.0570465	28	0.060147	14	0.0579597	15	0.97	**0.0528763**	18	0.0875251	15	0.0643747	67	0.123128	9	0.19484245	7
	0.95	0.8	0.0580147	16	0.0671589	13	0.0575532	13	0.81	**0.0533129**	19	0.0881585	14	0.0666801	13	0.0868601	13	0.19593505	24
	0.9856	0.3	0.0578389	14	0.061243	14	0.0654914	13	0.27	**0.0534952**	22	0.0870482	11	0.0665237	14	0.0877997	9	0.19493395	14
	0.99	0.4	0.0565954	14	0.0559973	14	0.0611193	13	0.19	**0.0548693**	22	0.0864128	10	0.0669176	14	0.0866258	15	0.19582235	14
	0.999	0.6	0.0561521	21	0.0590136	14	0.0612947	14	0.34	**0.0547978**	22	0.0872785	15	0.0688795	78	0.0895359	19	0.19519225	12
	1	0.4	**0.0567623**	55	0.0895154	14	0.0851147	14	0.48	0.0571899	22	0.0873431	11	0.0683851	42	0.0877086	13	0.19567825	14



(a) $\alpha = 0.95$　　(b) $\alpha = 0.98$

(c) $\alpha = 0.99$　　(d) $\alpha = 0.999$

图 2　例 5.2 中各算法在不同 α 下的 CPU 与 err 关系

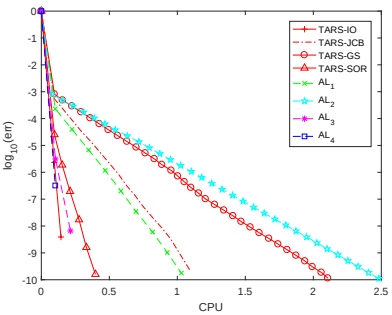
例 5.3. [4] 设 t 表示张量的零元素密度 (如果 $t=0$ 或 $t=1$, 则 $\mathcal{P}$ 分别是正张量或零张量). 运用 MATLAB 中的 randsample 函数生成具有不同密度的稀疏张量 $\mathcal{P}$.

在例 5.3 中, 我们给出了当 $n=200$ 时, t 从 $0.1-0.9$, 以 0.1 为间隔取值下的数值情况. 数值实验结果如表 3 所示.

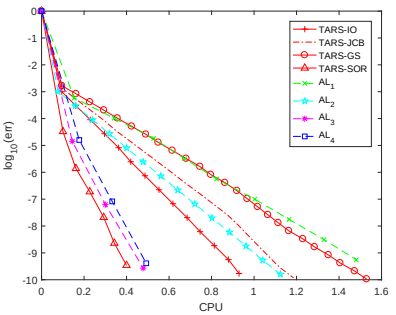
表 3 例 5.3 对比文 [11] 的数值结果

			TARS-IO		TARS-JCB		TARS-GS			TARS-SOR		AL_1		AL_2		AL_3		AL_4	
t	α	β	CPU	IT	CPU	IT	CPU	IT	ω	CPU	IT	CPU	IT	CPU	IT	CPU	IT	CPU	IT
0.1	0.9	0.5	**0.146777**	18	0.165779	3	0.168160	75	0.67	0.148283	7	0.206211	10	0.151103	8	0.213690	3	0.475844	3
	0.95	0.3	**0.145544**	18	0.146873	18	0.145606	18	0.77	0.152824	7	0.208550	10	0.148402	24	0.215883	3	0.478567	3
	0.98	0.4	**0.145750**	33	0.152518	24	0.145918	33	0.25	0.148580	5	0.204652	11	0.149509	46	0.213015	3	0.486762	3
	0.99	0.9	**0.144840**	18	0.153376	14	0.151802	15	0.27	0.148906	5	0.204049	9	0.149120	47	0.210171	3	0.476848	2
	0.999	0.7	0.144978	3	**0.144859**	18	0.151955	34	0.63	0.149911	7	0.204283	10	0.154436	33	0.215170	3	0.476537	2
	1	0.8	0.145841	18	0.152768	74	**0.145034**	34	0.67	0.151856	7	0.205287	10	0.154148	14	0.207767	3	0.487173	3
0.2	0.9	0.3	0.159379	3	0.150339	34	**0.147400**	33	0.81	0.154318	7	0.325359	9	0.150295	76	0.355355	3	0.903584	3
	0.95	0.1	0.163345	8	0.153461	76	0.158340	25	0.97	**0.145903**	8	0.312809	7	0.148760	48	0.359990	4	0.892122	3
	0.98	0.2	0.171305	15	0.147404	25	**0.144616**	34	0.85	0.151523	8	0.319196	7	0.148867	15	0.346560	4	0.837500	3
	0.99	0.9	0.172014	8	0.160388	8	0.164933	77	0.9	0.151169	8	0.334396	9	0.150371	34	0.347934	4	0.843848	3
	0.999	0.1	0.159318	15	**0.145256**	25	0.159622	77	0.79	0.147393	7	0.328374	8	0.149914	8	0.335786	4	0.852520	3
	1	0.4	0.164419	15	**0.145220**	48	0.148432	34	0.61	0.153482	7	0.324193	5	0.152580	19	0.348429	4	0.870277	3
0.3	0.9	0.3	**0.142600**	19	0.144539	15	0.153892	34	0.97	0.159766	8	0.334066	9	0.149308	25	0.217159	3	0.883122	3
	0.95	0.9	0.142765	34	**0.142489**	19	0.153133	19	0.65	0.146318	7	0.314832	4	0.148376	48	0.213685	4	0.875347	3
	0.98	0.3	**0.143558**	19	0.145346	19	0.146115	19	0.25	0.148482	5	0.328269	7	0.148506	11	0.215017	4	0.860289	4
	0.99	0.3	0.142134	15	0.162559	25	0.142020	20	0.17	**0.141713**	5	0.322408	8	0.149504	77	0.211984	4	0.867009	4
	0.999	0.6	**0.141861**	15	0.142534	34	0.162575	35	0.14	0.151878	4	0.334525	9	0.149356	11	0.223877	4	0.874072	4
	1	0.1	0.146725	19	**0.144193**	48	0.144267	35	0.69	0.151748	7	0.329559	7	0.149047	34	0.241075	3	0.865447	4
0.4	0.9	0.3	**0.142199**	19	0.153133	34	0.151053	8	0.15	0.161222	4	0.335682	6	0.156237	79	0.363650	4	0.621589	3
	0.95	0.2	**0.141998**	19	0.158135	15	0.148752	26	0.55	0.157939	6	0.327831	9	0.155661	19	0.344781	4	0.592048	3
	0.98	0.6	**0.140167**	4	0.147191	34	0.142740	49	0.46	0.163011	6	0.334523	9	0.155617	19	0.300279	4	0.634632	3
	0.99	0.1	**0.144299**	15	0.152137	20	0.163437	26	0.94	0.149916	7	0.303487	10	0.157289	15	0.294773	4	0.619984	4
	0.999	0.3	**0.144343**	15	0.155077	8	0.150162	49	0.69	0.147760	6	0.323468	4	0.155560	8	0.308816	4	0.597355	3
	1	0.3	**0.143393**	35	0.144734	19	0.157015	8	0.96	0.148456	7	0.343166	6	0.153126	78	0.295256	4	0.563547	3
0.5	0.9	0.5	0.142358	19	**0.140534**	35	0.140705	35	0.24	0.143553	5	0.212890	7	0.168977	11	0.238318	4	0.497446	3
	0.95	0.7	**0.143920**	15	0.149025	15	0.156628	26	0.99	0.147994	8	0.216705	11	0.160769	4	0.239481	4	0.490125	3
	0.98	0.6	0.145035	35	0.149968	4	0.158837	26	0.13	**0.142119**	4	0.211648	4	0.196804	79	0.238942	4	0.494959	3
	0.99	0.7	0.145826	35	**0.143241**	36	0.147828	9	0.45	0.151717	6	0.219455	4	0.209377	184	0.237917	4	0.497683	4
	0.999	0.1	0.145390	20	**0.143303**	15	0.159349	15	0.82	0.150966	8	0.218901	4	0.167752	82	0.242420	4	0.494775	3
	1	0.2	**0.141488**	35	0.145278	20	0.162038	79	0.69	0.146379	7	0.222696	10	0.198602	50	0.240682	4	0.495852	3
0.6	0.9	0.1	**0.145037**	8	0.160134	8	0.167399	26	0.48	0.152523	6	0.342299	7	0.145526	83	0.321753	4	0.938670	4
	0.95	0.4	0.156485	35	0.164918	15	0.156265	51	0.11	**0.149620**	4	0.328923	4	0.153206	8	0.303716	4	0.944672	4
	0.98	0.6	**0.150152**	35	0.153840	80	0.163630	20	0.83	0.152143	8	0.336925	7	0.150198	80	0.348360	4	0.925356	3
	0.99	0.8	0.159081	20	0.156203	4	0.157193	9	0.93	**0.144924**	8	0.337842	11	0.151430	20	0.351543	4	0.925329	4
	0.999	0.9	0.151008	15	0.160433	27	0.160551	12	0.23	**0.144598**	5	0.309147	11	0.156182	4	0.359714	5	0.930469	4
	1	0.4	0.155867	4	0.155846	36	0.158069	51	0.3	0.156573	6	0.306762	9	0.148686	35	0.362035	4	0.927923	3
0.7	0.9	0.3	**0.141359**	20	0.148714	53	0.143627	52	0.94	0.153453	8	0.228381	9	0.152223	20	0.213414	4	0.497008	4
	0.95	0.3	**0.142492**	36	0.157271	81	0.143169	20	0.42	0.147495	6	0.240932	8	0.152537	8	0.215522	4	0.499670	4
	0.98	0.7	**0.141049**	36	0.142635	37	0.145576	51	0.38	0.149760	6	0.236688	6	0.151943	8	0.215984	4	0.495512	3
	0.99	0.6	**0.142294**	20	0.143875	51	0.160302	51	0.92	0.147563	8	0.236347	7	0.152392	8	0.214469	4	0.497418	3
	0.999	0.8	**0.135204**	20	0.145076	36	0.144168	36	0.72	0.153537	7	0.239126	6	0.152355	8	0.212712	5	0.500333	3
	1	0.1	**0.143522**	36	0.140859	36	0.146583	36	0.65	0.149488	7	0.239529	10	0.150503	207	0.214635	4	0.499206	4
0.8	0.9	0.8	0.157429	4	0.149263	16	**0.149079**	16	0.17	0.164547	5	0.301657	11	0.164321	16	0.205829	5	0.552921	4
	0.95	0.4	0.166843	27	**0.150536**	51	0.162340	53	0.14	0.153840	5	0.304573	6	0.160758	21	0.219263	5	0.547053	3
	0.98	0.9	0.173244	20	**0.148825**	16	0.151308	16	0.57	0.153654	7	0.300688	5	0.160406	229	0.209991	5	0.550819	5
	0.99	0.5	0.179040	16	**0.154489**	20	0.179563	37	0.4	0.154440	7	0.223923	12	0.156971	20	0.211355	5	0.558891	3
	0.999	0.6	0.181755	21	0.172377	36	0.170994	28	0.51	0.161413	7	0.209892	11	0.154773	9	0.211375	5	0.555490	4
	1	0.5	0.154220	16	0.182621	5	0.183924	9	0.55	**0.153823**	7	0.218503	7	0.154488	16	0.213109	5	0.551223	3
0.9	0.9	0.2	0.180261	39	**0.156503**	13	0.167918	28	0.87	0.159771	8	0.231828	5	0.158908	85	0.244897	5	0.646826	5
	0.95	0.4	0.166106	39	**0.146921**	17	0.155379	17	0.15	0.150551	5	0.259903	6	0.167980	9	0.245647	6	0.644649	5
	0.98	0.7	0.163864	12	0.155653	13	0.165836	58	1	**0.153885**	9	0.262686	5	0.163116	21	0.244985	5	0.647015	4
	0.99	0.3	0.163023	29	**0.152050**	21	0.168040	38	0.14	0.156741	5	0.261178	9	0.163800	100	0.245903	5	0.650362	5
	0.999	0.3	0.157806	16	**0.150616**	53	0.151770	22	0.66	0.157226	8	0.244404	12	0.166089	21	0.246423	6	0.646966	4
	1	0.2	0.161862	12	0.173856	22	0.169644	54	0.9	**0.158706**	9	0.242357	12	0.164088	85	0.240420	5	0.644960	4

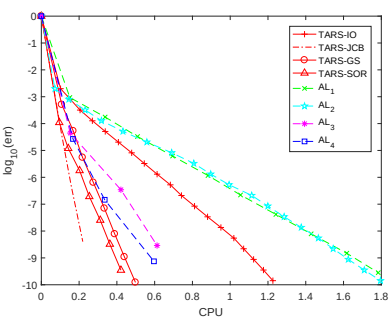
表 3 中数据表明, 在选取合适参数的情况下, 本文所提算法总是比现有算法花费更小的 CPU 时间. 其中算法 TARS-IO 表现出了明显的优势. 图 3 给出了各算法在不同 α 和 t 下的 CPU 与 err 的关系曲线. 尽管算法 TARS-GS 具有更多的迭代步数, 但是具有更小的误差, 且多数情况下, 迭代步数, CPU 和误差总是比 AL_2 要小.



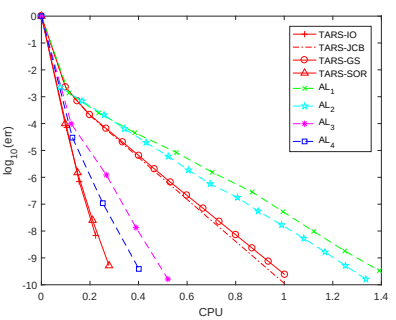
(a) $t = 0.1, \alpha = 0.999$



(b) $t = 0.4, \alpha = 0.99$

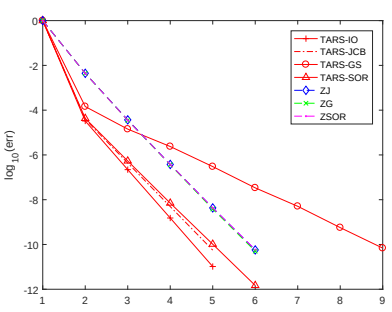


(c) $t = 0.6, \alpha = 0.99$

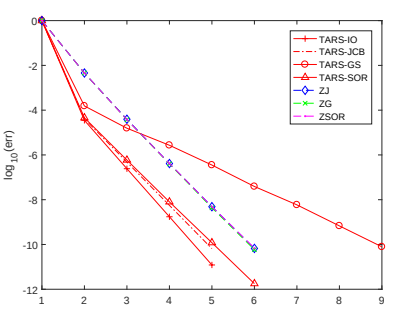


(d) $t = 0.8, \alpha = 0.90$

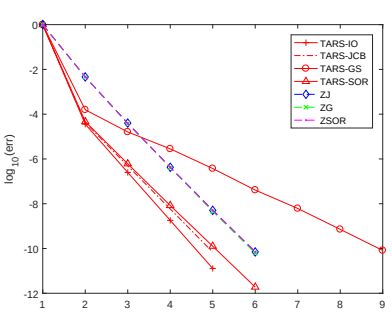
图 3　例 5.3 中各算法在不同的 t 和 α 下 CPU 与 err 的关系



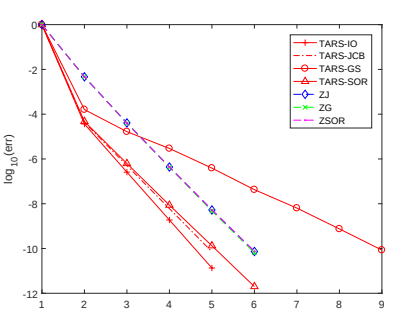
(a) $\alpha = 0.95$



(b) $\alpha = 0.98$

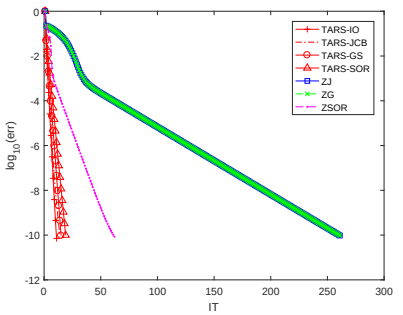


(c) $\alpha = 0.99$

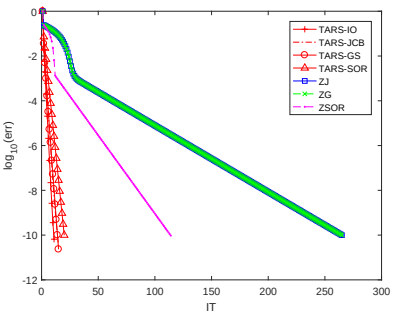


(d) $\alpha = 0.999$

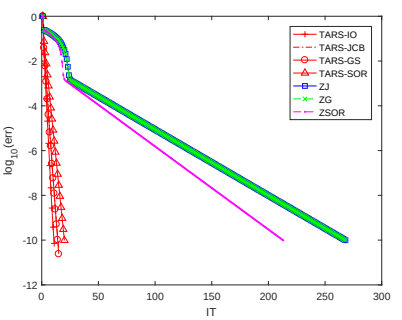
图 4　例 5.1 中各算法在不同 α 下的 IT 与 err 关系



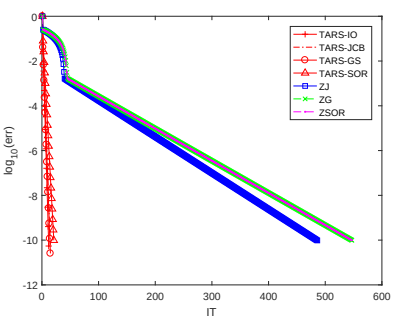
(a) $\alpha = 0.90$



(b) $\alpha = 0.95$

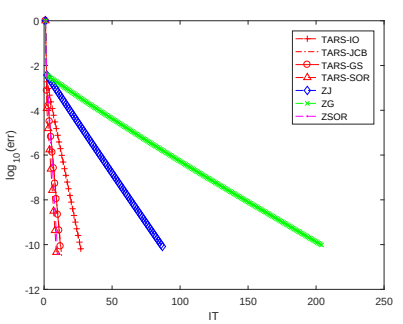


(c) $\alpha = 0.98$

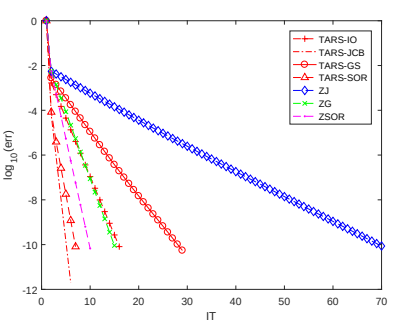


(d) $\alpha = 0.999$

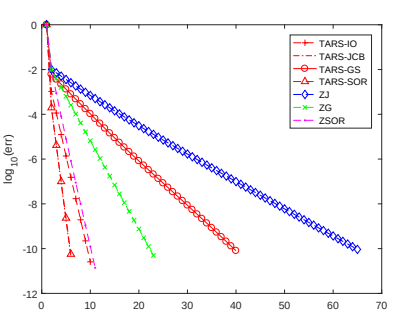
图 5 例 5.2 中各算法在不同 α 下的 IT 与 err 关系



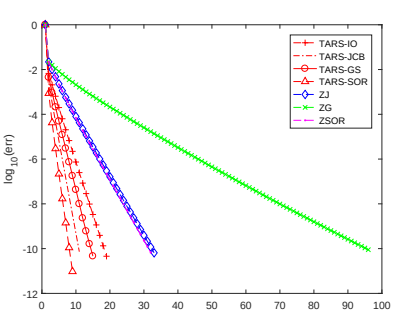
(a) $t = 0.2$



(b) $t = 0.3$



(c) $t = 0.6$



(d) $t = 0.9$

图 6 例 5.3 中各算法在不同 α 下的 IT 与 err 关系

进一步, 本文与文 [15] 中的算法 II 中的 Jacobi 型 (记为 ZJ), G-S 型 (记为 ZG) 和 SOR 型 (记为 ZSOR) 进行比较. 我们通过比较各算法的迭代步数和误差之间的关系来验证算法的有效性.

在例 5.1 中, 我们测试了 3 阶 100 维的随机张量, 结果如图 4 所示. 在例 5.2 中, 我们测试了 $n = 100$, $n_p = 80$ 的情况, 结果如图 5 所示. 在例 5.3 中, 我们测试了 $n = 100$, $t = 0.2$, 0.3, 0.6, 0.9 的情况, 结果如图 6 所示. 图 4 – 图 6 表明, 当选取了合适的参数时, 新算法花费更少的 CPU 时间, 更小的误差和更少的迭代步数. 这样便进一步验证了新算法的有效性.

此外, 我们考虑一个实际应用的例子. 考虑仿真一种多人竞技的纸牌游戏, 具体游戏规则可参考维基百科 (https://en.wikipedia.org/wiki/Winner_(card_game)). 传统的排名是由玩家赢得的游戏数量决定的. 本文中我们考虑通过建立和求解多重线性 PageRank 模型对竞技者进行排名. 我们考虑玩家的个数为 n, 赢得游戏的能力的得分为 $\mathbf{w}$. 在此游戏的一轮中, 每三个玩家玩一场游戏, 即每一轮有 $\binom{n}{3}$ 场游戏. 对于每场游戏, 三个玩家 i, j 和 k 用一副纸牌. 定义玩家 i, j 和 k 分别赢得游戏的规则如下:

$$\begin{cases} i \text{ 赢}, & \text{如果 } \varphi \in \left[0, \dfrac{w_i}{w_i + w_j + w_k}\right], \\ j \text{ 赢}, & \text{如果 } \varphi \in \left(\dfrac{w_i}{w_i + w_j + w_k}, \dfrac{w_i + w_j}{w_i + w_j + w_k}\right], \\ k \text{ 赢}, & \text{如果 } \varphi \in \left(\dfrac{w_i + w_j}{w_i + w_j + w_k}, 1\right]. \end{cases}$$

其中随机数 φ 是由 MATLAB 的 rand 函数生成. 我们按照以下步骤生成的随机张量 $\mathcal{P}$. 其中 diag, zeros 和 reshape 均是 MATLAB 函数.

步骤 1 输入回合数 n_c, 玩家数量 n, 分数向量 $\mathbf{w}$, 参数 α 及随机向量 $\mathbf{v}$.

步骤 2 初始一个张量 $\mathcal{A} = \text{zeros}(n, n, n)$.

for 所有的 n_c 轮 **do**

for 所有任意 3 个玩家 i, j, k **do**

$$\begin{cases} a_{jki} = a_{jki} + 1, a_{kji} = a_{kji} + 1, & \text{如果 } i \text{ 赢}, \\ a_{ikj} = a_{ikj} + 1, a_{kij} = a_{kij} + 1, & \text{如果 } j \text{ 赢}, \\ a_{ijk} = a_{ijk} + 1, a_{jik} = a_{jik} + 1, & \text{如果 } k \text{ 赢}. \end{cases}$$

end for

end for

步骤 3 令 $\mathbf{d} = \mathbf{e}^T \mathcal{A}_{(1)}$, $D = \text{diag}(\mathbf{d})$ 及 $P = \mathcal{A}_{(1)} D^+$, 其中 D^+ 表示 D 的广义逆矩阵.

步骤 4 令 $\bar{\mathcal{P}} = \text{reshape}(P, n, n, n)$, 生成的随机张量 $\mathcal{P}$ 如下

for $j, k = 1, 2, \cdots, n$ **do**

$$p_{ijk} = \begin{cases} \frac{1}{n}, & \text{对任意 } i \in \langle n \rangle, \bar{p}_{ijk} = 0, \\ \bar{p}_{ijk}, & \text{其他}. \end{cases}$$

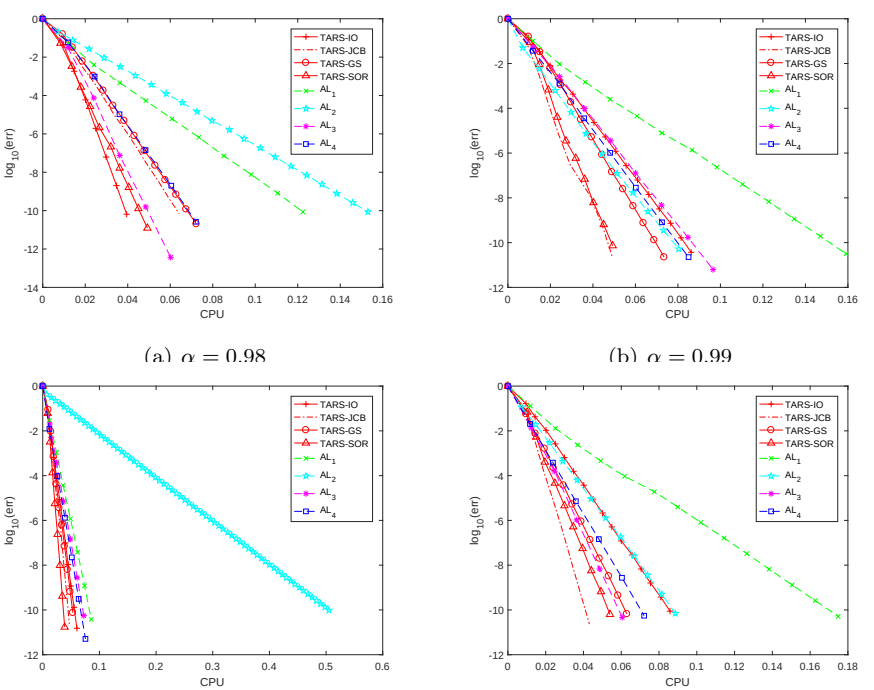
end for

我们给出了当 $n=100$, $n_c=10$, $\mathbf{w}=(n,n-1,\cdots,2,1)^T$ 的情况, 数值实验结果如表 4 所示.

表 4 纸牌游戏例子对比文 [11] 的数值结果

		TARS-IO		TARS-JCB		TARS-GS			TARS-SOR		AL_1		AL_2		AL_3		AL_4	
α	β	CPU	IT	CPU	IT	CPU	IT	ω	CPU	IT	CPU	IT	CPU	IT	CPU	IT	CPU	IT
0.9	0.7	0.013708	7	0.013888	21	0.014771	15	0.5	**0.012773**	8	0.023323	9	0.014426	68	0.023068	3	0.066269	6
0.95	0.7	0.014446	21	**0.013918**	36	0.014772	23	0.1	0.015372	8	0.023922	11	0.014330	15	0.024267	6	0.066877	7
0.98	0.5	0.014546	7	0.014799	12	0.014178	14	0.7	**0.012543**	10	0.024079	10	0.014256	21	0.023545	5	0.066113	6
0.99	0.7	0.014345	16	**0.013841**	8	0.014433	14	1	0.014609	10	0.023858	13	0.014160	11	0.023662	8	0.066896	7
0.999	0.9	0.014454	11	0.014548	8	0.014391	10	0.1	**0.012821**	8	0.023856	7	0.014178	68	0.024154	6	0.066344	6
1	0.1	0.014472	16	**0.014282**	8	0.014281	12	0.9	0.014981	10	0.023931	14	0.014249	12	0.024305	5	0.066113	6

表 4 表明, 通过选取合适的参数, 所提的算法比现有算法所花费的 CPU 时间更少. 图 7 给出了所测试的算法在不同 α 下的 CPU 和 err 的关系曲线. 从图 7 中可以看出, 本文所提算法具有更小的 err 和 CPU. 当 $\alpha=0.999$ 时, 所提算法所需 CPU 总时间都比算法 AL_1-AL_4 要少.

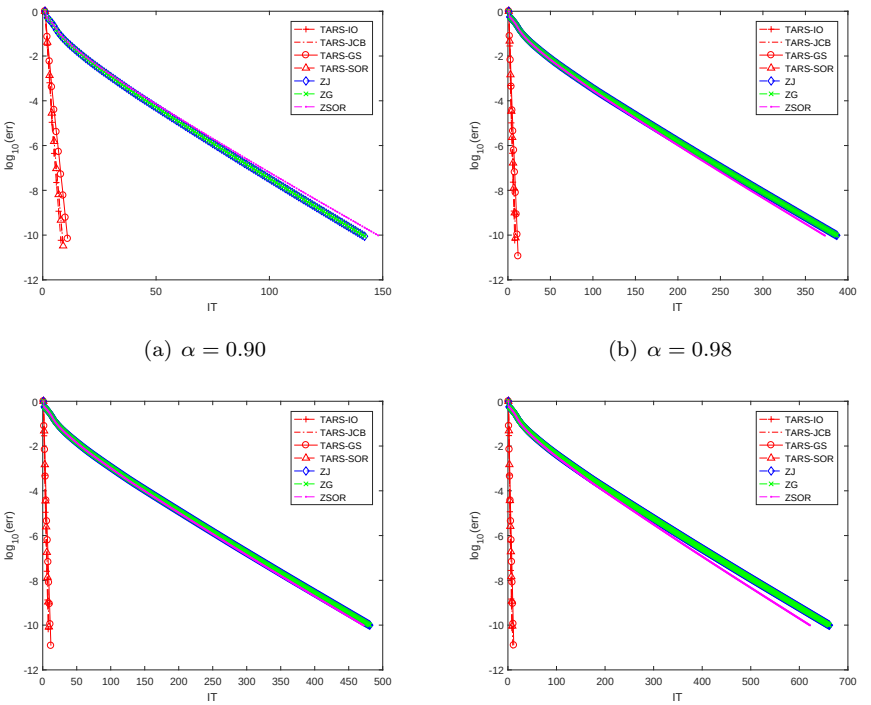


(a) $\alpha=0.98$ (b) $\alpha=0.99$

(c) $\alpha=0.999$ (d) $\alpha=1$

图 7 纸牌游戏中各算法在不同 α 下的 CPU 与 err 关系

接着, 我们测试了 $n=50$, $n_c=15$ 的情况来比较各算法的迭代步数和误差之间的关系, 并与 ZJ, ZG 和 ZSOR 进行比较. 结果如图 8 所示. 从图中可以看出, 本文所提算法所需的 CPU 时间和迭代步数更少, 同时具有更小的误差, 其中 TARS-GS 算法表现最明显.



(a) $\alpha = 0.90$　　(b) $\alpha = 0.98$

(c) $\alpha = 0.99$　　(d) $\alpha = 0.999$

图 8　纸牌游戏中各算法在不同 α 下的 IT 与 err 关系

6. 总　　结

我们基于文 [11] 提出的求解高阶马尔可夫链和多线性 PageRank 问题的松弛算法基础上, 结合传统的正则分裂方法, 提出了新的求解多线性 PageRank 的分裂算法, 并进一步给出了内外迭代、Jacobi 迭代、Gauss-Seidel 迭代和逐次超松弛迭代法, 并对新算法进行了收敛性分析. 数值实验结果证实, 通过选择合适的参数, 新算法对于解决多线性 PageRank 问题是有效的.

参 考 文 献

[1] Page L, Brin S, Motwani R, T Winograd. The PageRank citation ranking: Bringing order to the web[R]. Technical report, Stanford University, Stanford, CA, 1999: 1999–66.

[2] Gleich D F, Lim L H, Yu Y. Multilinear PageRank[J]. SIAM Journal on Matrix Analysis and Applications, 2015, 36(4): 1507–1541.

[3] Li W, Ng M K. On the limiting probability distribution of a transition probability tensor[J]. Linear and Multilinear Algebra, 2014, 62(3): 362–385.

[4] Li W, Liu D, Vong S W, Xiao M. Multilinear PageRank: Uniqueness, error bound and perturbation analysis[J]. Applied Numerical Mathematics, 2020, 156: 584–607.

[5] Li W, Liu D, Ng M K, Vong S W. The uniqueness of multilinear PageRank vectors[J]. Numerical Linear Algebra with Applications, 2017, 24(6): e2107.

[6] Fasino D, Tudisco F. Ergodicity coefficients for higher-order stochastic processes[J]. SIAM Journal on Mathematics of Data Science, 2020, 2(3): 740–769.
[7] Meini B, Poloni F. Perron-based algorithms for the multilinear PageRank[J]. Numerical Linear Algebra with Applications, 2018, 25(6): e2177
[8] Guo P C, Gao S C, Guo X X. A modified Newton method for multilinear PageRank[J]. Taiwanese Journal of Mathematics, 2018, 22(5): 1161-1171.
[9] Benson A R, Gleich D F. Computing tensor Z-eigenvectors with dynamical systems[J]. SIAM Journal on Matrix Analysis and Applications, 2019, 40(4): 1311–1324.
[10] Zhang X, Ni Q, Ge Z. A convergent Newton algorithm for computing Z-eigenvalues of an almost nonnegative irreducible tensor[J]. Optimization Methods and Software, 2020, 35(2): 377–393.
[11] Liu D, Li W, Vong S W. Relaxation methods for solving the tensor equation arising from the higher-order Markov chains[J]. Numerical Linear Algebra with Applications, 2019, 26(5): e2260.
[12] Yu G, Zhou Y, Lv L. Accelerating power methods for higher-order Markov chains[J]. arXiv preprint arXiv:2003.00686, 2020.
[13] Cipolla S, Redivo-Zaglia M, Tudisco F. Extrapolation methods for fixed-point multilinear PageRank computations[J]. Numerical Linear Algebra with Applications, 2020, 27(2): e2280.
[14] Lai F, Li W, Peng X, Chen Y. Anderson accelerated fixed-point iteration for multilinear PageRank[J]. Numerical Linear Algebra with Applications, 2023: e2499.
[15] Liu D, Li W, Vong S W. The tensor splitting with application to solve multi-linear systems[J]. Journal of Computational and Applied Mathematics, 2018, 330: 75–94.
[16] Varga R S. Iterative analysis[M]. Berlin: Springer, 1962.
[17] Gleich D F, Gray A P, Greif C, Lau T. An inner-outer iteration for computing PageRank[J]. SIAM Journal on Scientific Computing, 2010, 32(1): 349–371.
[18] Huang N, Ma C F. Parallel multisplitting iteration methods based on M-splitting for the PageRank problem[J]. Applied Mathematics and Computation, 2015, 271: 337–343.
[19] Liu D, Vong S W, Shen L. Improved uniqueness conditions of solution for multilinear PageRank and its application[J]. Linear and Multilinear Algebra, 2022: 1–31.

SPLITING ITERATION METHOD FOR SOLVING MULTILINEAR PAGERANK PROBLEM

Tang Shuting Deng Xiuqin Liu Dongdong
(*School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou 510006, China*)

Abstract

In this paper, combined with the relaxation algorithm, we present new tensor splitting methods for solving multilinear PageRank problem. The convergence analysis of the proposed algorithms is also shown. It is shown that the proposed algorithms perform well from some numerical experiments when relaxation parameters are properly selected.

Keywords: Multilinear PageRank problem; Tensor splitting; Relaxation algorithm.
2010 Mathematics Subject Classification: 65F10, 65H10.