基于无冲突并行随机梯度下降的图布局求解方法

王智[1)], 薛明亮[2)], 王一凡[1)], 钟发海[1)], 汪云海[1,3)*]

[1)] (山东大学计算机科学与技术学院 青岛 266237)
[2)] (山东大学软件学院 济南 250101)
[3)] (中国人民大学信息学院 北京 100872)
(wang.yh@sdu.edu.cn)

摘 要: 应力模型是计算节点连接图布局时最常用的方法之一. 随机梯度下降法由于具有很好的收敛性, 常被用于求解应力模型, 但该方法难以实现有效并行. 虽然无锁随机梯度下降方法能大幅提高并行效率, 但其求解过程中常存在线程冲突, 导致结果准确性低. 为了提高并行图布局的效率和准确性, 提出一种无冲突的随机梯度下降的并行求解方法. 首先提出一种面向应力模型的线程分配算法, 将与节点 j 相同的点对分配到同一线程内计算, 保证基于随机梯度下降方法的图布局无冲突化求解; 然后仅对线程内的样本随机洗牌并减少次数, 进一步提升并行效率. 在 16 个不同规模的真实数据集上进行实验, 并将所提方法应用在稀疏化应力模型的求解上, 实验结果显示所提方法在求解精度上无损失且求解速度提高 10 倍以上, 从布局质量和运行效率 2 个方面证明了该方法的高效性和可用性.

关键词: 图布局; 随机梯度下降; 并行计算; 图可视化
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2023-00627

Graph Drawing by Conflict-Free Parallel Stochastic Gradient Descent

Wang Zhi[1)], Xue Mingliang[2)], Wang Yifan[1)], Zhong Fahai[1)], and Wang Yunhai[1,3)*]

[1)] (*School of Computer Science and Technology, Shandong University, Qingdao* 266237)
[2)] (*School of Software and Technology, Shandong University, Jinan* 250101)
[3)] (*School of Information, Renmin University of China, Beijing* 100872)

Abstract: The stress model is one of the most commonly used methods for graph layout. The stochastic gradient descent (SGD) algorithm is often used to solve the stress model due to its ease of convergence, but it is difficult to implement effective parallelization. Although the lock-free SGD method can greatly improve parallel efficiency, there are often thread conflicts during the solution process, leading to low accuracy of results. To improve the efficiency and accuracy of parallel graph layout, this paper proposes a conflict-free parallel SGD solution method. Firstly, a thread allocation algorithm for the stress model is designed, which assigns pairs of nodes with the same j to the same thread for computation, thus ensuring a conflict-free graph layout solution based on SGD. Secondly, the parallel efficiency is further improved by shuffling only the samples within each thread and reducing the number of iterations. To test the efficiency of this method, comparative experiments were conducted on 16 real datasets of different scales, and the proposed method was applied to solve the sparse stress model. The experimental results show that the proposed method achieves no loss in solution accuracy and increases the solution speed by more than tenfold, demonstrating the efficiency and usability of the proposed method in terms of layout quality and runtime efficiency.

收稿日期: 2023-09-20; 修回日期: 2024-05-06. **基金项目**: 科技创新 2030—“新一代人工智能”重大项目(2022ZD0160805); 国家自然科学基金(62132017, 62141217); 山东省自然科学基金(ZQ2022JQ32). **王智**(1999—), 男, 硕士研究生, 主要研究方向为图可视化; **薛明亮**(1995—), 男, 博士, 主要研究方向为图可视化; **王一凡**(2001—), 男, 硕士研究生, 主要研究方向为图可视化; **钟发海**(1998—), 男, 硕士, 主要研究方向为图可视化、高维数据可视化; **汪云海**(1984—), 男, 博士, 教授, 博士生导师, CCF 会员, 论文通信作者, 主要研究方向为数据可视化与可视分析.

Key words: graph layout; stochastic gradient descent; parallel computing; graph visualization

网络数据又称图数据，是一种常见的数据类型，在金融、生物、医疗等领域有广泛的应用. 在信息可视化领域，最常用的图可视化方法是节点连接图，其使用节点表示数据中的实体，同时在节点间连接一条边表示实体间存在的关系. 为了增强节点连接图的视觉表达能力，研究者针对不同的数据类型和可视化任务提出许多图布局方法. 图布局方法指在节点连接图中对节点位置进行合理布局，以准确地表达节点间的邻居关系的方法，该方法可以帮助用户准确地理解网络数据中的关系信息，从而更好地分析和理解网络数据.

经典图布局方法分为弹簧-电荷模型[1-2]和应力模型[3-4]这 2 类. 弹簧-电荷模型的设计理念是在布局结果中有边相连的节点对应该比不相连的节点布局得更近；但这种模型仅考虑了相邻节点间的连接关系，不能很好地表达网络数据中节点间的整体关系. 应力模型则将弹簧-电荷模型的设计理念拓展为所有节点之间的布局距离应该与它们之间的图论距离接近，该模型反映了节点间的整体连接关系，在数据的全局结构表达上效果更好；然而，应力模型的求解复杂度较高，求解速度远不如弹簧-电荷模型. 目前比较流行的应力模型求解方法是基于随机梯度下降(stochastic gradient descent, SGD)的方法[5]，其所需的迭代次数最少，同时求解精度也比较高，但求解效率仍旧不能满足用户的实时分析需求. 因此，本文通过研究并行 SGD 的方法，以提升应力模型的求解效率.

SGD 的并行研究在机器学习领域中已经非常常见[6-8]. 该方法的并行化面临的主要问题是：每次迭代都需要用小批量样本计算部分梯度并更新参数，而这些样本中可能涉及相同的参数，因此在并行中如果直接使用异步方案会出现参数冲突和覆盖的问题. 研究者通过使用加锁的方案[9]来避免并行 SGD 方法中出现冲突的情况；然而加锁的方案会严重影响SGD的计算效率，不能解决其求解效率低的问题. Niu 等[10]提出直接使用无锁并行的方法求解 SGD，并证明在稀疏问题上冲突的数量是有上限的，可以忽略其对求解结果的影响. 由于在求解基于应力模型的图布局问题上，无锁并行方法造成的冲突会影响布局结果，不能被忽略，因此在求解图布局模型时，需要设计更合理的 SGD 并行方案.

本文提出一种无锁的并行 SGD 方法，通过 GPU 并行加速应力模型求解. 本文分析了在无锁并行过程中消除冲突的必要性，并提出一种无冲突的并行 SGD 方法，可以高效、高质量地求解应力模型；另外，通过对 SGD 求解应力模型过程中用到的随机化算法进行分析，提出在并行计算环境下高效率的随机化策略，提升本文方法的效率.

1 相关工作

1.1 应力模型

应力模型[3-4]是图布局中的一类重要方法，其核心是使用有长度的弹簧建模节点之间的相互作用关系. 应力模型在所有节点对之间施加一个弹簧力，其中，弹簧的长度为两点之间的图论距离. 弹簧力的存在使得布局距离小于弹簧长度的节点对互相排除，大于弹簧长度的节点对互相吸引. 通过这种建模方式，应力模型使布局结果中节点对之间的距离尽可能地接近它们之间的图论距离，获得全局结构较好的布局结果.

由于所有节点对之间都需要计算弹簧力，因此应力模型的计算复杂度为 $O(V^2)$；其中，V 表示图中节点个数，远高于 Barnes-Hut 加速[11]的弹簧-电荷模型[12]的计算复杂度 $O(V\times \mathrm{lb}V)$. 为了减少应力模型的计算时间，Brandes 等[13]和 Khoury 等[14]通过近似计算图中所有节点对的最短路径距离减少计算时间，Ortmann 等[15]则通过减少应力项的项数加速应力模型求解：这些方法都是通过稀疏化策略降低应力模型的计算复杂度.

除了上述优化计算复杂度的近似方法，应力模型还有很多用于提升布局质量的求解方法. Kamada 等[3]最初采用梯度下降法求解应力模型，但这种方法在求解中容易陷入局部极小值，布局质量较差；Gansner 等[4]提出应力最优化方法，能确保求解过程中的应力误差(stress error, SE)单调下降，比最初的梯度下降法在布局质量上更有优势. 还有一些方法对应力模型的表达形式进行改进，以获得更好的布局效果. Wang 等[16]以边向量的形式对应力模型进行重新建模，进一步提高了应力模型的表现力.

Xue 等[17]提出统一力模型 Taurus、弹簧引力和斥力更加均衡的均衡应力模型，以及高效的 ASGD(augment stochastic gradient descent, ASGD)求解器，使得应力模型能够在更多的图数据类型上得到较好的布局效果和较高的求解效率；Ahmed

等[18]提出一种通用的方法 SGD²，支持优化多种目标函数，它通过直接优化应力函数求解应力模型，获得具有较高可理解性的布局结果；Zheng 等[5]提出使用 SGD 求解应力模型的方法，通常只需要十几轮的迭代就能收敛，同时求解出的布局结果 SE 更小，是目前最流行的应力模型求解方法，且 SGD 不依赖良好的初始化布局，预处理开销较小. 因此，本文在文献[5]的基础上进行改进，提出无冲突并行的 SGD 求解方法，很大程度上缓解了应力模型求解效率低的问题.

1.2　SGD

SGD 是一种求解无约束优化问题的优化求解方法[6-8]，在文本分类和自然语言处理等领域得到了广泛应用，可用于解决这些领域中经常遇到的大规模稀疏机器学习问题[19]. SGD 的核心是在每轮迭代中随机选择一个小批量的样本，计算损失函数的梯度并用此更新参数；这种随机性使得算法更具鲁棒性，能够避免陷入局部极小值，并且训练速度也会更快[20-21]. 由于 SGD 每次迭代都用小批量样本计算得到梯度的更新参数，因此在并行中会出现参数冲突和覆盖问题，不易使用并行计算加速. 为了解决冲突的问题，Zinkevich 等[9]提出有锁并行 SGD 方法，通过加锁的方式保证涉及相同变量的随机梯度不会在并行计算中出现冲突；但有锁并行 SGD 方法运行效率较低，计算速度与串行 SGD 方法相比提升不大，在实际应用中对 SGD 的求解效率提升非常有限. 为了提升 SGD 并行求解效率，Niu 等[10]提出一种无锁并行 SGD 方法“HOGWILD!”，证明了在稀疏的优化问题中，大多数的梯度更新造成的冲突是有上限的，只影响小部分参数，因此，无锁的 HOGWILD![10]方法的计算速度比任何内存锁定的并行化方法都要快；然而，作为无锁并行 SGD 方法，HOGWILD![10]并没有处理冲突的问题，导致其布局结果与串行的SGD实现得到的布局结果在质量上存在一定的差距，会严重降低图布局的求解质量，在基于应力模型的图布局方法求解中是不可忽视的，因此不能直接用无锁并行SGD方法求解图布局. 本文提出了一种无冲突的无锁并行SGD方法，在满足无锁的前提下消除并行冲突，可以同时保证求解质量和求解速度.

2　本文方法

为了解决 SGD 在应力模型求解中效率低的问题，本文提出一种无冲突并行的 SGD 求解方法，既能显著地提高计算效率，又能保证求解精度.

2.1　SGD 求解应力模型

应力模型的目标是最小化图中节点之间理想距离欧几里得距离的差距，因此定义一个能量方程量化图布局中的误差[3-5]为

$$L(X)=\sum_{i<j} w_{ij}\left(\left\|X_i-X_j\right\|-d_{ij}\right)^2 .$$

其中，X 表示图中节点的坐标；d_{ij} 表示图中节点 i 与 j 之间的最短路径长度；w_{ij} 表示权重因子，在应力模型中一般设置为 d_{ij}^{-2}.

SGD 求解应力模型是通过最小 $L(X)$ 能量方程来实现的，每次仅考虑一个节点对 (i,j)，移动这对节点，最小化此对节点之间的能量，公式为

$$U_{ij}=\left(\left\|X_i-X_j\right\|-d_{ij}\right)^2 \Big/ d_{ij}^2 .$$

SGD 通过迭代的方式求解应力模型，每次迭代中，以随机的顺序取出每个节点对 (i,j)，并给这对节点施加一个作用力 $\boldsymbol{f}_{ij}$，让它们顶点在相反的方向移动. $\boldsymbol{f}_{ij}$ 定义为

$$\boldsymbol{f}_{ij}=\frac{1}{2}\frac{\left\|X_i-X_j\right\|-d_{ij}}{d_{ij}^2}\times\boldsymbol{e}_{ij} .$$

其中，$\boldsymbol{e}_{ij}$ 表示力的方向，定义为 2 个节点之间的单位方向向量. SGD 在每次迭代中依次移动每对节点，重复迭代固定次数或者满足收敛条件后求得最终的布局. 这种求解方法具有迭代次数少、不依赖于初始布局、快速收敛的优点；但是每次迭代都需要计算任意对节点之间的作用力并移动，时间复杂度是 $O(V^2)$. 对于大规模图数据，这样的时间开销依然是用户无法接受的. 为了加速应力模型求解，本文提出 SGD 求解应力模型的并行化方法.

2.2　无锁并行方法

针对并行计算过程中的资源冲突问题，解决方法分为有锁和无锁 2 种. 有锁方法是给共享资源的访问加锁，保证互斥资源的顺序访问，从而避免了资源的冲突；但该方法具有加锁和释放锁的时间开销，且线程会存在阻塞，访问同一共享资源的线程越多，其他线程的等待时间就越长. 因此，有锁方法的并行性能可能会受到较大的影响. 相反，在无锁方法中，每个线程都可以独立地访问共享资源，不受其他线程操作的阻塞，可以极大地提升该方法的并发性能.

在并行 SGD 求解应力模型时，样本是每个待

计算的节点对，梯度是它们之间的作用力，共享变量是图 G 的所有节点坐标. 为了得到更好的加速效果，本文使用无锁并行方法加速计算. 无锁并行方法的核心是，在优化过程中每个线程处理待计算的点对 (i, j) 时，使用局部梯度直接更新节点 i 和 j 的坐标，且在此过程中没有任何同步互斥的锁. 无锁并行整个过程中，内存是由全部线程共享的，而且没有任何加锁操作的时间开销，因此无锁并行方法拥有非常快的速度；但由于多个线程同时更新同一个节点的坐标，后更新的线程会覆盖前一个线程更新的结果，导致一部分梯度丢失，带来误差.

本文将冲突定义为：2 个或多个线程同时更新同一个节点的坐标，本质上是多线程环境下的写冲突问题. 将理论冲突数定义为：第 n 个线程运行时(即系统中有 n 个线程在同时计算)，线程之间的冲突数的数学期望. 使用无锁并行方法求解图布局时，线程的理论冲突数取决于线程数和样本数(即节点数)，下面将给出简单推导. 假设一共需要计算 m 个样本，此时已经有 t 个计算线程在运行并更新线程对应节点的坐标，线程之间相互独立且等概率地取每一个样本，若发生冲突，意味着第 $t+1$ 个线程取到了与前 t 个线程相同的样本. 假设已经取到的非冲突样本数为 b_t，即有 b_t 个线程是第 1 个取到某节点，若同一时刻没有其他线程在更新这些节点，则这 b_t 个节点的更新是正确无误的；已经取到的冲突的样本数为 c_t，即有 c_t 个节点取到了和前面 c_t 个节点相同的样本；则有

$$t = c_t + b_t \tag{1}$$

即由于一共有 m 个样本，此时样本中发生冲突的概率为

$$p_t = c_t / m \tag{2}$$

第 $t+1$ 个线程在运行时，取样本只会发生 2 种可能的情况：与前 t 个线程发生冲突时，其概率为 p_t，冲突数为 c_t+1；不与前 t 个线程发生冲突时，其概率为 $1-p_t$，冲突数不变. 因此，第 $t+1$ 个线程取样本时，冲突数的递推公式为

$$c_{t+1} = p_t \cdot (c_t + 1) + (1 - p_t) \cdot c_t \tag{3}$$

将式(1)(2)代入式(3)，可得到

$$c_{t+1} = \left(1 - \frac{1}{m}\right) c_t + \frac{t}{m} \tag{4}$$

第 1 个线程执行时不会发生冲突，有 $c_1 = 0$，代入式(4)，化简得到冲突数量为

$$c_{t+1} = m\left(1 - \frac{1}{m}\right)^t - m + t .$$

可见，线程数越多，样本量越少，理论冲突数就越多. 对于 m 个样本，计算线程数为 n 的冲突比例，即 n 个线程之间发生冲突的概率为

$$r_{n,m} = \left(1 - \frac{1}{m}\right)^n - 1 + \frac{n}{m} .$$

图 1 所示为线程数从 256~2 048 的冲突比例随样本量变化的趋势. 可以看出，随着计算样本量的增加，理论冲突比例会逐渐降低；但随着线程数的增加，冲突比例有显著的提升.

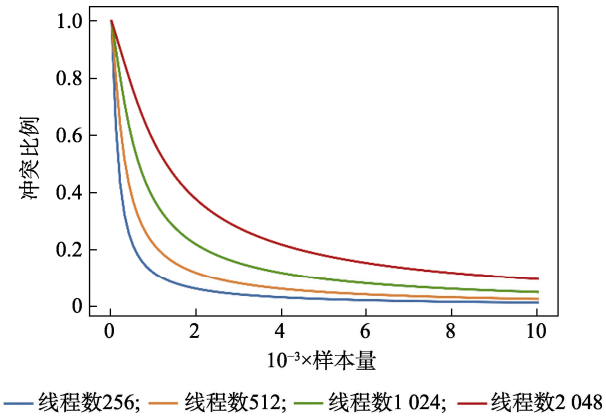


图 1 冲突比例随线程数增加的变化趋势图

在并行计算的过程中，线程数通常决定了计算速度，而线程数的增加会使冲突比例增加，导致布局质量降低，这将严重影响并行算法的性能上限. 同时，对于常见的图布局任务，图数据的节点数往往不超过 1 万，因此冲突造成的影响会非常明显. 综上所述，解决 SGD 并行计算中的冲突问题是很有必要的.

2.3 无冲突并行方法

为了解决直接使用无锁并行方法带来的冲突问题，本文在无锁并行的基础上进行改进，提出无冲突并行方法. 该方法也是一种无锁方法，在共享变量的更新过程中没有任何加锁操作，而是在构造计算任务时使用线程分配算法给样本预先分配执行的线程，避免发生更新冲突.

分配规则是将与节点 j 相同的点对 (i, j) 分配到同一个线程中，同时修改节点坐标的更新过程；线程更新节点坐标时，只用计算得到的力更新节点 j 的坐标，而不更新节点 i 的坐标. 这种方案可以保证并行执行过程中不存在任何更新冲突. 对于不同线程，梯度的计算和更新是同时的，但是由于不同线程中更新的节点 j 不重复，不存在更新

同一个点的梯度的情况，因此也就不会发生写冲突；而一个线程内的任务是同步执行的，只有上一个对节点的更新操作完成以后，才会进行下一个对节点的更新操作，因此也不存在更新节点坐标上的冲突；同时，线程内部不需要加锁和释放锁的操作，保证了线程内部执行的高效性.

虽然该方法在计算过程中仍存在线程之间的读/写冲突，但这种冲突仅仅相当于改变了线程之间点对的执行顺序. 由于 SGD 对节点对的执行顺序并无要求，因此读/写冲突并不会对求解精度造成影响，这一点将在本文实验中得到验证. 此外，在计算过程中，如果没有对同一个节点的 x 和 y 坐标进行原子更新，可能会出现一个线程读取到另一个线程已更新节点的 x 坐标和未更新的 y 坐标，从而引发数据冲突. 但通过实验比较分析，此种冲突并未导致计算误差；相反，原子更新同一节点 x 和 y 坐标的策略使计算速度降低了一半以上. 因此，出于加速效率的考量，本文方法未原子更新节点的 x 和 y 坐标.

对于 V 个节点的图 G，根据线程分配算法，与节点 j 相同的点对 (i,j) 会分配到相同线程执行，则每个线程被分配的计算任务都是 $V-1$ 个，因此线程分配算法是负载均衡的. 此外，将计算点对按列优先方式存储，以提高程序运行时的空间局部性.

本文方法步骤如下：

输入. 图 $G=(V,E)$，最大迭代次数 n_{iter}.

输出. 图中节点坐标矩阵 $\boldsymbol{X}$.

Step1. 计算图中任意一对节点之间的最短路径.

Step2. 给图中每个节点分配随机的坐标.

Step3. 构造计算任务并分配线程:

Step3.1. 对于所有点对 (i,j)，根据其对应的参数和最短路径距离生成参数元组;

Step3.2. 建立元组集合并添加所有参数元组，根据元组集合中的每个元组对应的点对 (i,j)，使用线程分配算法，按 j 为每个元组分配计算线程 t_{id};

Step4. 开始 n_{iter} 次迭代计算，对于每个线程，每次迭代中循环执行下列操作:

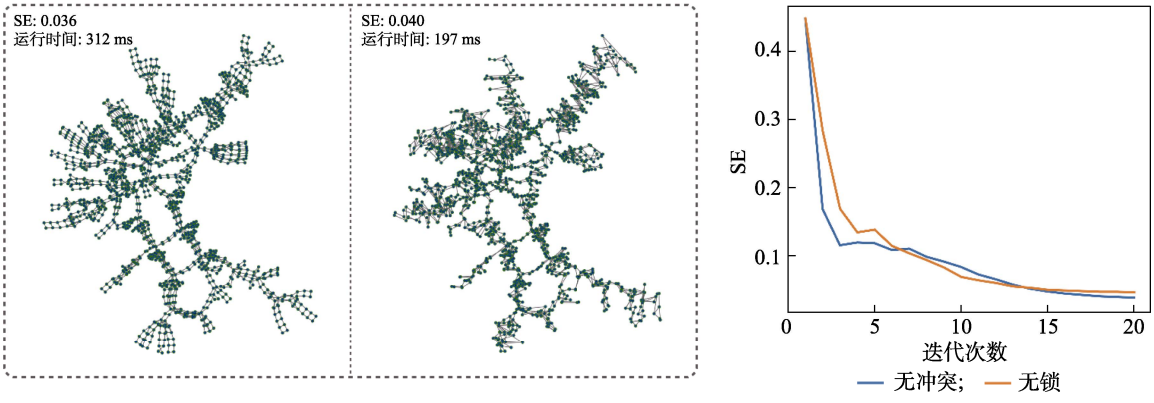
Step4.1. 线程随机选择一个分配到的参数元组;

Step4.2. 根据参数元组中的参数和最短路径距离，计算 (i,j) 之间的作用力 $\boldsymbol{f}$;

Step4.3. 使用 $\boldsymbol{f}$ 更新节点 j 的坐标;

Step4.4. 当线程内所有参数元组都计算完成，结束当前线程.

图 2 所示为无锁并行方法和无冲突并行方法在 QH1484 这个电力系统网络数据上的求解结果. 可以看出，求解过程中，2 种方法收敛速度相差不大，且 20 次迭代后都能收敛到全局最小值附近；但最终结果上，无冲突并行方法的损失函数值比无锁并行方法低 10%，而且从视觉效果上看，无冲突并行方法比无锁并行方法在局部结构上表现要好很多，无锁并行方法的布局结果中出现了大量视觉杂乱，这是因为其存在较大比例的更新冲突，局部结构保持能力差，无法展示数据的细节信息.



a. 无冲突方法　　b. 无锁方法　　c. 损失函数变化图

图 2　无冲突并行和无锁并行效果比较

从运行时间上比较，无冲突并行方法略慢于无锁并行方法，这是由于无冲突并行预先分配线程带来的时间开销.

2.4　随机化策略

使用梯度下降方法最小化能量方程[22]时，需要先计算所有节点对之间的梯度，然后将梯度累加到每个点的梯度上，最后更新每个点的位置. 而使用SGD最小化能量方程时，在计算节点对之间的梯度后，直接将梯度更新到对应的一个节点上. 另外，与传统梯度下降不同，SGD 每次计算节点对之

间的力后会立即更新节点位置, 因此需要使用随机化策略将节点对顺序打乱, 防止固定顺序的更新导致结果陷入局部最小值. 常用的随机化策略是使用洗牌算法将样本随机打乱, 使得每个样本出现在任意位置的概率相等, 有效地减小计算样本之间参数更新互相抵消的问题[23-24]. Zheng 等[5]在 SGD 中的实验结果, 证明了使用洗牌算法 Random Reshuffling 的 SGD 的求解效果最好, 可以有效地避免结果陷入局部最小值[25]. 但是, 对于应力模型, 要计算的节点对的数量是 V^2, 并且每次迭代前都需要一次完整 Random Reshuffling 打乱, 对于 n_{iter} 轮迭代, 时间开销为 $O\left(n_{\text{iter}} \times V^2\right)$, 这也是一个不可忽视的时间开销. 在 GPU 并行环境下, 本文提出了更适合并行计算的样本顺序打乱策略, 进一步减小了时间开销.

串行 SGD 方法求解时, 需要将所有样本严格按照 Random Reshuffling 完全打乱, 而 GPU 执行线程的样本顺序本身具有一定的随机性, 所以并不需要完全打乱. 本文改进的随机化策略只在每个线程内使用 Random Reshuffling 打乱节点对顺序. 另外, 在实验中发现, 可以只在第 1 轮迭代开始前进行一次随机打乱, 随后的过程中即使不再进行打乱也能收敛到较低的损失函数值. 实验结果证明, 对于在 GPU 上运行的无冲突并行方法, 仅打乱一次样本顺序并不影响 SGD 求解的收敛性. 因此, 本文的并行实验均基于单次线程内打乱样本顺序的新策略.

2.5 稀疏化应力模型的并行方法

对于具有 V 个顶点和 E 条边的图 G, 如果使用应力模型求解图布局, 为了最小化图中任意 2 个顶点之间的图论距离和欧几里得距离, 需要计算全对顶点之间的最短路径(all pairs shortest path, APSP). 计算 APSP 的朴素算法需要 $O(VE \times \text{lb}V)$ 的时间[26], 且 APSP 矩阵需要 $O(V^2)$ 的空间来存储, 这对大规模图是不可能实现的; 如果使用其他不需要 APSP 的布局方法, 如弹簧-电荷模型, 又很难实现布局结果中节点对间的图论距离保持.

为了布局结果的良好距离保持效果和扩展性, 本文在大规模图数据上使用 Ortmann 等[15]的稀疏化应力模型, 由于该模型仅考虑 E 对有边相连的节点以及 p 个枢轴点和全部 V 个节点之间的作用力, 因此也只需要存储 $pV+E$ 对节点之间的最短路径, 空间复杂度和每轮迭代的时间复杂度都为 $O(pV+E)$, 远小于全应力模型的 $O(V^2)$. 通常, 枢轴点的数量 p 在 100~500[6], 对于大规模图数据, $p \ll V$, 因此, 稀疏化应力模型的提出使得将应力模型应用于大规模图数据成为可能.

为了将无冲突并行方法应用到大规模图数据的应力模型求解上, 本文对无冲突并行方法进行一些改进, 使得这种方法能适用于稀疏化应力模型. 由于 SGD 求解不依赖于一个好的初始化节点布局, 因此本文方法不需要采用 PivotMDS 初始化节点坐标[13], 可以直接使用随机初始化节点坐标以节省预处理时间; 同时, 在实验中发现, GPU 上运行的 SGD 可以随机选择枢轴, 相比使用 “max/min random sp”方法[6]选择枢轴更加灵活且计算复杂度更低. 鉴于这 2 种方法的实际效果相差不大, 本文选择计算复杂度更低的随机选择枢轴方法. 本文参考 Zheng 等[5]的策略选取枢轴的数量 p=200, 保证了求解速度和求解质量之间的均衡.

3 实验及结果分析

从 2 个方面评估本文提出的并行 SGD 的图布局求解方法的有效性. (1) 在速度和布局质量上, 比较串行 SGD 方法, Taurus 方法[17]和 SGD^2 方法[18], 以及无冲突并行 SGD 方法和无锁并行 SGD 方法在求解应力模型上的表现; (2) 在 6 个大规模的图数据上对稀疏化并行方法进行实验, 给出运行时间和可视化结果. 所有图的初始布局都采用相同的随机初始化. 所有实验都在一台配置为 Intel Xeon Silver 4110 处理器, NVIDIA RTX 2080Ti 显卡, 64 GB 内存的 Linux 服务器上完成.

3.1 无冲突并行

首先定量比较串行 SGD 方法、Taurus 方法[17]和 SGD^2 方法[18]、与无冲突并行 SGD 方法和无锁并行 SGD 方法在求解应力模型上的运算效率和布局质量; 然后从视觉效果上, 比较上述 5 种方法的布局质量. 其中, 串行 SGD 方法、Taurus 方法[17]和 SGD^2 方法[18]在 CPU 上运行, 无冲突并行和无锁并行方法在 GPU 上运行, GPU 上共使用 32 个 grid 和 256 个 block.

由于并行计算过程存在一定的随机性, 每次运行结果存在差异, 因此将每种方法各运行 5 次, 取每次运行结果测量指标的平均值作为实验结果.

3.1.1 应力模型数据集

选择 10 个经典图数据组成测试数据集进行实验, 其中, 节点数为 1 138~11 434, 边数为 1 458~32 818.

3.1.2　应力模型求解质量比较

本文使用 SE 评估布局结果的全局距离结构保持度，测量图中任意 2 个节点 i 和 j 之间的最短路径距离 d_{ij} 和它们的欧几里得距离的平方差并取平均，公式为

$$L_{\mathrm{SE}}=\frac{2}{V(V-1)}\sum_{i<j}\frac{\left(\left\|X_i-X_j\right\|-d_{ij}\right)^2}{d_{ij}^2}.$$

其中，V 表示图中节点数；X_i 和 X_j 分别表示节点 i 和 j 的坐标位置；d_{ij} 表示节点 i 和 j 之间的最短路径长度. SE 可以很好地衡量应力模型的优化结果，其值越小，布局结果中整体的距离保持度越好.

表 1 所示为 5 种方法求解结果的 SE 值对比. 其中，串行 SGD 方法、Taurus 方法[17]、无冲突并行 SGD 方法和无锁并行 SGD 方法的收敛条件均为迭代次数达到 30 次; SGD^2 方法[18]的收敛条件设置为 SE 值的变化小于10^{-7}. 可以看出，在所有数据上，串行方法和 Taurus 方法[17]的表现比较接近，都好于 SGD^2 方法[18]; 无冲突并行方法的 SE 值都等于或好于串行方法和 Taurus 方法[17]，都远好于 SGD^2 方法[18]，而无锁并行方法的 SE 值几乎都比串行方法和 Taurus 方法[17]差.

表 1　5 种方法求解结果的 SE 值对比

数据	方法				
	串行	Taurus[17]	SGD^2[18]	无冲突并行	无锁并行
Bus1138[4,27]	**0.061**	**0.061**	0.105	**0.061**	0.075
QH1484[4]	**0.036**	0.047	0.193	**0.036**	0.041
Sierpinski3D[4,27]	**0.062**	**0.062**	0.207	**0.062**	0.075
data[12,28]	**0.009**	0.013	0.151	**0.009**	0.010
3elt[13,29-30]	0.041	**0.037**	0.299	**0.037**	0.043
USpowerGrid[12-13,30]	0.058	0.058	0.233	**0.056**	0.061
add32[12-13,28]	0.088	0.079	0.117	**0.078**	0.089
Whitaker3[12-13,28]	**0.001**	**0.001**	0.648	**0.001**	**0.001**
crack [13]	**0.003**	**0.003**	0.532	**0.003**	**0.003**
FE_4elt2[13]	**0.035**	0.042	0.649	**0.035**	0.036

注. 粗体表示最优值.

实验结果表明，无冲突并行方法求解的精度明显高于无锁并行方法，且相比串行方法的求解精度没有损失，同时比 Taurus 方法[17]和 SGD^2 方法[18]的求解精度高. 此外，定量评估的结果表明，无冲突并行方法在某些数据上的距离保持度比串行方法更高，这是由于 GPU 中线程之间的执行顺序存在不确定性，相比 CPU 上 Random Reshuffling 方法有更高的随机性，更容易收敛到全局最小值，可以很好地避免陷入局部极小值造成的全局结构损失. 总体上，无冲突并行方法在运行效率和布局质量上都有较好的表现.

3.1.3　应力模型求解速度比较

图 3 所示为 5 种方法的运行时间对比. 可以看出，2 种并行方法的运行速度远快于串行方法、Taurus 方法[17]和 SGD^2 方法[18]; 当节点数达到 5 000 时，与串行方法相比，无冲突并行方法可以达到 15 倍加速比，且随着数据规模的增大，加速比也越来越大，说明使用并行方法可以显著地加快 SGD 的求解速度. 由于线程预先分配需要额外的时间开销，当数据规模增加时，无冲突并行的速度比无锁并行稍慢. 但是整体上看，无冲突并行的速度与无锁并行是近似的.

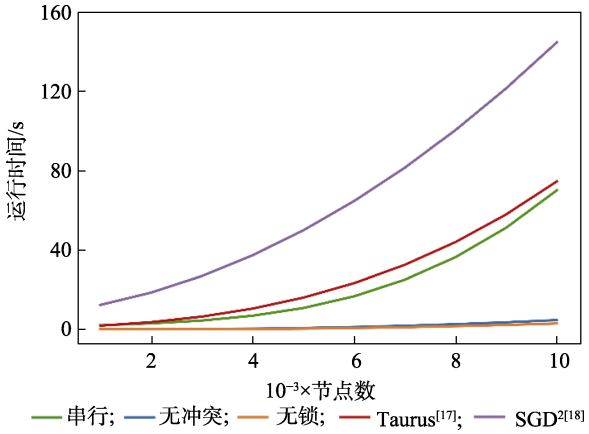
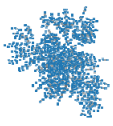
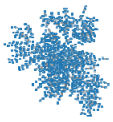
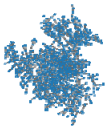
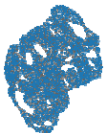
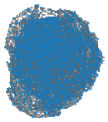
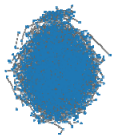
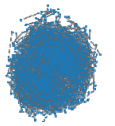
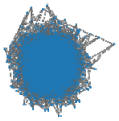
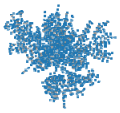
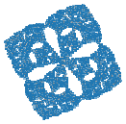
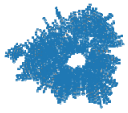
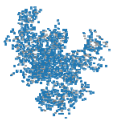
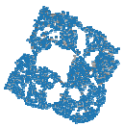
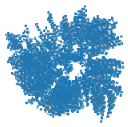


图 3　5 种方法的运行时间对比

3.1.4　应力模型视觉效果展示

表 2 所示为使用 5 种方法求解应力模型在测试数据上得到的部分布局结果. 可以看出: (1) 串行 SGD 方法和 Taurus 方法[17]的视觉效果基本一致，而无冲突并行方法的视觉效果与这 2 种串行方法差距不大，仅有细节上的偏差，表明无冲突并行方法在计算过程中基本没有丢失求解精度. (2) 由于 SGD^2 方法[18]仅适合较小的数据集，对于节点数大于 2 000 的数据，无冲突并行方法的结果远好于 SGD^2 方法[18]的结果. (3) 无锁并行方法的视觉效果虽然与无冲突并行方法在全局结构上相似，但是在局部细节差距很大，如在 add32 数据上，无锁并行方法的结果出现了比较严重的扭曲; 在 Sierpinski3D 数据上，4 部分相同的结构也并没有展示出相同的视觉效果，与另外 2 种方法具有较大差异. 证明了无锁并行方法中的更新冲突使其求解质量比无冲突并行的方法要低很多，对布局结果的可理解性产

表 2 5 种方法的视觉效果图

方法	Bus1138[4,27]	Sierpinski3D[4,27]	3elt[13,29-30]	USpowerGrid[12-13,30]	add32[12-13,28]	FE_4elt2[13]
串行						
Taurus[17]						
SGD^2[18]						
无冲突并行						
无锁并行						

生了非常大的影响.

3.1.5 无冲突并行方法求解应力模型小结

本节实验从数值指标和视觉效果 2 个方面比较了串行 SGD 方法、Taurus 方法[17]和 SGD^2 方法[18], 以及无冲突并行 SGD 方法和无锁并行 SGD 方法这 5 种方法在求解应力模型上的表现. 实验结果充分证明, 本文提出的无冲突并行方法在使用 SGD 求解应力模型的问题上优于无锁并行方法, 比串行 SGD 方法、Taurus 方法[17]和 SGD^2 方法[18]的求解效率更高.

3.2 大规模图数据的稀疏化并行

为了验证无冲突并行方法在大规模图数据上的性能, 在 6 个大规模图数据上对无冲突并行方法和串行方法进行实验, 并在时间和视觉效果上进行比较与分析. 无锁并行方法不适合这组实验, 原因是图数据稀疏化后计算样本量大幅减少, 相同线程数下, 无锁并行方法冲突的概率显著提高, 导致无锁并行方法在很多图数据上无法收敛.

3.2.1 稀疏化应力模型数据集

选择 6 个大规模真实图数据组成测试数据集, 其中, 节点数为 11 738~114 599, 边数为 67 828~3 260 967.

3.2.2 稀疏化应力模型求解质量比较

使用 SE 评估一个布局结果的全局距离结构保持度. 表 3 所示为在 6 个数据集上, 串行方法和无冲突并行方法在求解稀疏化应力模型时, 布局结果的 SE 值. 对比数值结果可以看出, 本文提出的稀疏化并行方法在 Pesa 数据上的表现稍差于串行方法, 但在其他数据上与串行方法结果相似甚至更好; 尤其在 CA-CondMat 数据上, 稀疏化并行方法求解的结果显著好于串行方法. 定量评估的结果表明, 与串行方法相比, 本文改进的稀疏化并行方法并没有引入计算误差.

表 3 串行和并行 SE 值对比

数据	方法	
	串行	并行
Pesa[31]	**0.133**	0.134
CA-CondMat[31]	0.421	**0.246**
Finance256[31]	0.058	**0.052**
Venkat50[31]	0.213	**0.198**
SC-pkustk13[31]	**0.292**	**0.292**
Luxebourg_osm[27]	0.082	**0.072**

注. 粗体表示最优值.

3.2.3 稀疏化应力模型求解速度比较

表 4 所示为 6 个数据上稀疏化应力模型的串行求解和无冲突并行求解的运行时间对比, 其中, 收敛条件均为迭代次数达到 200 次. 可以看出, 与串行方法相比, 无冲突并行方法的求解速度非常快, 所有图的加速比都超过 20, 其中, 在 Finance256 数据集上, 加速比达到 48; 即使在包含 114 599 个节

点的图数据 Luxebourg_osm 上，求解时间也只需 0.602 s，且仅使用 355 MB 的内存空间. 实验结果表明，无冲突并行求解方法显著提高了稀疏化应力模型的运算效率.

表 4　串行和并行运行时间及加速比

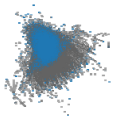
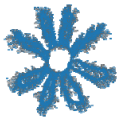
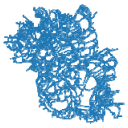
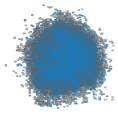
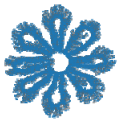
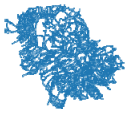
数据	运行时间/s		加速比
	串行方法	并行方法	
Pesa[31]	3.265	**0.113**	28.90
CA-CondMat[31]	6.270	**0.202**	31.04
Finance256[31]	13.034	**0.271**	48.10
Venkat50[31]	8.912	**0.412**	21.63
SC-pkustk13[31]	15.049	**0.744**	20.23
Luxebourg_osm[27]	15.789	**0.602**	26.24

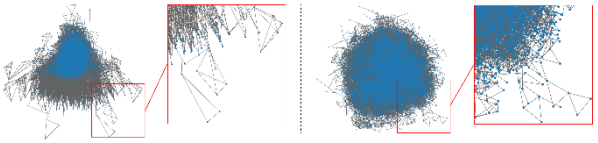
注. 粗体表示最优值.

3.2.4　稀疏化应力模型视觉效果展示

表 5 所示为稀疏应力模型的布局结果. 可以看出，无冲突并行方法的结果与串行方法的结果近似，并没有损失全局结构；无冲突并行方法在 CA-CondMat 和 Finance256 这 2 个数据上有着更好的表现，对于 CA-CondMat 数据，无冲突并行方法结果的节点分布更加均匀，参考图 4 中的局部放大效果，无冲突并行方法的边缘节点组成的形状接近等边三角形，且三角形大小比较近似，而串行方法的边缘节点组成的三角形边长不统一；对于 Finance256 数据，无冲突并行方法结果的视觉效果更加对称，全局结构保持更好. 参考表 3 的 SE 数值，同样可以证明无冲突并行方法的求解精度更加准确. 本文推测缘由如下：枢轴元素只有 p 个，但在计算中需要受到 V 个节点的作用力 $(p << V)$，需要移动 V 次，因此容易导致枢轴元素的坐标震荡；而无冲突并行方法只移动 (i, j) 点对中的节点 j，并且线程分配时会优先将枢轴元素放置到点对中节点 i 的位置上，使得枢轴元素移动次数减少到接近 $O(p)$；并行化的 SGD 方法更进一步打乱了计算顺序，使得梯度下降的方向发生偏离的概率更大；而无冲突并行 SGD 更容易避免陷入局部最优和更容易接近全局最优解：以上结果进一步证明了本文方法的有效性.

表 5　2 种方法的视觉效果图

方法	Pesa[31]	CA-CondMat[31]	Finance256[31]	Venkat50[31]	SC-pkustk13[31]	Luxebourg_osm[27]
串行						
无冲突并行						



a. 串行方法　　b. 无冲突并行方法

图 4　2 种方法在 CA-CondMat 数据上局部视觉效果

3.2.5　本文方法求解稀疏化应力模型小结

本节实验结果表明，本文的无冲突并行方法不仅显著地加速了稀疏化应力模型的求解过程，在布局结果的视觉效果上也表现出色，提高了稀疏化应力模型的求解效率和扩展性.

4　结　语

为了快速求解应力模型，本文提出一种无冲突并行 SGD 方法，包括无冲突的线程分配算法和改进的随机化方法. 将该方法应用于求解稀疏化的应力模型，实验结果表明，本文方法在大规模图数据布局问题中具有高效性和可靠性，为图数据分析领域中的图布局问题提供了一种有效的解决方案. 然而，本文仅证明了无冲突并行 SGD 方法对求解应力模型与稀疏化应力模型的良好加速效果，后续研究将扩展至弹簧-电荷模型和统一力模型 Taurus 的求解.

参考文献(References):

[1] Eades P A. A heuristic for graph drawing[J]. Congressus Numerantium, 1984, 42: 149-160

[2] Fruchterman T M J, Reingold E M. Graph drawing by force-directed placement[J]. Software: Practice and Experience, 1991, 21(11): 1129-1164

[3] Kamada T, Kawai S. An algorithm for drawing general undirected graphs[J]. Information Processing Letters, 1989, 31(1): 7-15

[4] Gansner E R, Koren Y, North S. Graph drawing by stress majorization[C] //Proceedings of the 12th International Symposium on Graph Drawing. Heidelberg: Springer, 2004: 239-250

[5] Zheng J X, Pawar S, Goodman D F M. Graph drawing by stochastic gradient descent[J]. IEEE Transactions on Visualization and Computer Graphics, 2019, 25(9): 2738-2748

[6] Robbins H, Monro S. A stochastic approximation method[J]. The Annals of Mathematical Statistics,1951, 22(3): 400-407

[7] Nemirovski A, Yudin D B. On Cezari's convergence of the steepest descent method for approximating saddle point of convex-concave functions[J]. Soviet Mathematics. Doklady, 1978, 19: 258-269

[8] Shalev-Shwartz S, Singer Y, Srebro N, *et al*. Pegasos: primal estimated sub-GrAdient SOlver for SVM[C] //Proceedings of the 24th International Conference on Machine Learning. New York: ACM Press, 2007: 807-814

[9] Zinkevich M A, Weimer M, Smola M, *et al*. Parallelized stochastic gradient descent[C] //Proceedings of the 23rd International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc, 2010: 2595-2603

[10] Niu F, Recht B, Re C, *et al*. HOGWILD!: a lock-free approach to parallelizing stochastic gradient descent[C] //Proceedings of the 24th International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc, 2011: 693-701

[11] Barnes J, Hut P. A hierarchical O(*N* log *N*) force-calculation algorithm[J]. Nature, 1986, 324(6096): 446-449

[12] Hu Y. Efficient high quality force-directed graph drawing[J]. The Mathematica Journal, 2005, 10(1): 37-71

[13] Brandes U, Pich C. Eigensolver methods for progressive multidimensional scaling of large data[C] //Proceedings of the 14th International Symposium on Graph Drawing. Heidelberg: Springer, 2006: 42-53

[14] Khoury M, Hu Y F, Krishnan S, *et al*. Drawing large graphs by low-rank stress majorization[J]. Computer Graphics Forum, 2012, 31(3pt1): 975-984

[15] Ortmann M, Klimenta M, Brandes U. A sparse stress model[C] //Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization. Heidelberg: Springer, 2016: 18-32

[16] Wang Y H, Wang Y Y, Sun Y Q, *et al*. Revisiting stress majorization as a unified framework for interactive constrained graph visualization[J]. IEEE Transactions on Visualization and Computer Graphics, 2018, 24(1): 489-499

[17] Xue M L, Wang Z, Zhong F H, *et al*. Taurus: towards a unified force representation and universal solver for graph layout[J]. IEEE Transactions on Visualization and Computer Graphics, 2023, 29(1): 886-895

[18] Ahmed R, De Luca F, Devkota S, *et al*. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$[J]. IEEE Transactions on Visualization and Computer Graphics, 2022, 28(6): 2388-2399

[19] Shi Jiarong, Wang Dan, Shang Fanhua, *et al*. Research advances on stochastic gradient descent algorithms[J]. Acta Automatica Sinica, 2021, 47(9): 2103-2119(in Chinese)
(史加荣, 王丹, 尚凡华, 等. 随机梯度下降算法研究进展[J]. 自动化学报, 2021, 47(9): 2103-2119)

[20] Nagaraj D, Jain P, Netrapalli P. SGD without replacement: sharper rates for general smooth convex functions[C] //Proceedings of the 36th International Conference on Machine Learning. New York: PMLR, 2019: 4703-4711

[21] Gürbüzbalaban M, Ozdaglar A, Parrilo P A. Convergence rate of incremental gradient and incremental Newton methods[J]. SIAM Journal on Optimization, 2019, 29(4): 2542-2565

[22] Li Xingyi, Yue Yang. Survey of gradient descent algorithm[J]. Software Engineering, 2020, 23(2): 1-4(in Chinese)
(李兴怡, 岳洋. 梯度下降算法研究综述[J]. 软件工程, 2020, 23(2): 1-4)

[23] Boyd S P, Vandenberghe L. Convex optimization[M]. Cambridge: Cambridge University Press, 2004

[24] Safran I, Shamir O. How good is SGD with random shuffling?[C] //Proceedings of the Conference on Learning Theory. New York: PMLR, 2020: 3250-3284

[25] Gürbüzbalaban M, Ozdaglar A, Parrilo P A. Why random reshuffling beats stochastic gradient descent[J]. Mathematical Programming, 2021, 186(1-2): 49-84

[26] Zhang Zhong. Study of shortest path problem on large-scale graph[D]. Hefei: University of Science and Technology of China, 2014(in Chinese)
(张钟. 大规模图上的最短路径问题研究[D]. 合肥: 中国科学技术大学, 2014)

[27] Gansner E R, Hu Y, North S. A Maxent-stress model for graph layout[J]. IEEE Transactions on Visualization and Computer Graphics, 2013, 19(6): 927-940

[28] Walshaw C. A multilevel algorithm for force-directed graph drawing[C] //Proceedings of the 8th International Symposium on Graph Drawing. Heidelberg: Springer, 2001: 171-182

[29] Kruiger J F, Rauber P E, Martins R M, *et al*. Graph layouts by *t*-SNE[J]. Computer Graphics Forum, 2017, 36(3): 283-294

[30] Zhu M F, Chen W, Hu Y Z, *et al*. DRGraph: an efficient graph layout algorithm for large-scale graphs by dimensionality reduction[J]. IEEE Transactions on Visualization and Computer Graphics, 2021, 27(2): 1666-1676

[31] Kolodziej S P, Aznaveh M, Bullock M, *et al*. The suiteSparse matrix collection website interface[J]. Journal of Open Source Software, 2019, 4(35): Article No.1244