Report of Parallel Label Propagation

GuangCheng-Li 李光程
16098537-II20-0016
sky9475@live.com
Faculty of Information Technology, Macau University of Science and Technology
https://sites.google.com/view/guangcheng/

---

1. 实现方法

我们的Label Propagation基于KNN图，完全按参考中的方法进行实现。串行与并行算法均使用Matlab 2016a编写。
算法中共定义以下几个函数：

- main：用于程序启动的主函数；
- loadDataFromTxt：从txt中加载原始数据，并随机地将部分标签去除；
- compute_knn：寻找节点K个邻居的KNN算法函数；
- buildGraph：根据KNN建立概率转移矩阵；
- labelPropagation：执行标签传播，将无label的节点的label补全；
- show：将结果以图形的形式展示。

特别地，程序中有几处参数和语句可根据实际情况调节，特列举如下：

- main函数中max_iter变量：设置最大迭代次数；
- main函数中knn_num_neighbors：设置KNN算法保留的邻居个数；
- loadDataFromTxt函数中第8行 select_num_rowsfloor(rows * s)：s代表labeled的节点占比，可通过调节此参数（范围0-1）来控制labeled和unlabeled的节点比例；
- labelPropagation函数中tol变量：用于控制收敛精度，该参数越小，则结果收敛的越精确，但耗时越长；
- labelPropagation函数中第28与29行：此处有两个可选的迭代结束条件，在正常使用中，建议使用第28行的终止条件，即 while (iter < max_iter && changed > tol)，该结束条件使得算法在迭代次数大于等于max_iter或达到预期收敛精度时终止。而第29行的终止条件 while (iter < 100000) 是在测试并行算法优越性时，为拉大并行算法与串行算法的时间差距而使用的。

在并行算法中，我们使用matlab的GPU并行优化来提升算法的速度。特别地，在matlab 2016a与2016b中，对于Pascal architecture的显示卡支持不佳，官方已提供解决方案，可参考 https://www.mathworks.com/support/bugreports/1439741进行修复。

2. 实验与结果

2.1 实验环境

实验所用PC硬体配置如下：
MainBoard: Gigabyte G1.Sniper B7 (Intel Sunrise Point B150)
CPU: Intel Core i5-6500, 3300 MHz
RAM: 16 GB
GPU: MSI GTX 1060 Gaming X 6GB @1784 MHz

实验所用软体环境如下：
System: Microsoft Windows 10 Pro 10.0.15063.296
Matlab: 2016a

2.2 功能测试

为测试算法是否可正确运行，我们采用Aggregation数据集，并设置labeled的数据占比为0.3，迭代终止条件为while (iter < max_iter && changed > tol)，其中max_iter = 1000，tol = 1e-10，以验证算法正确性。由于串行与并行算法之逻辑相同，因此我们使用并行算法进行实验。实验结果如图1所示。从图中不难看出，我们的程序对数据的label正确性较高。

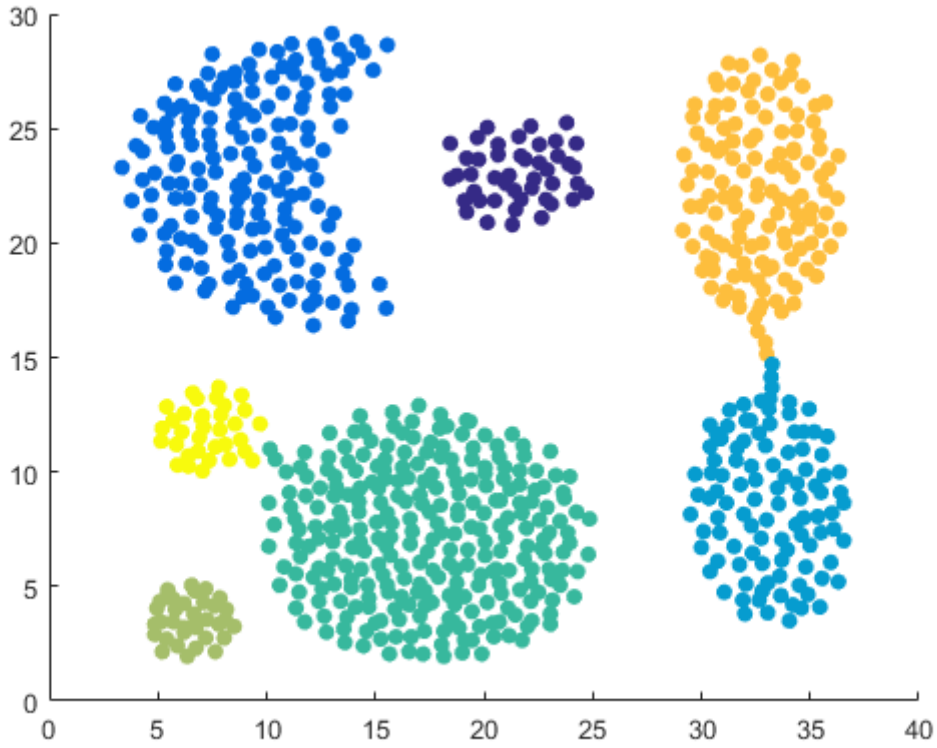

FIG. 1 算法运行结果

2.3 算法准确性

在该实验中，我们使用GPU并行算法，并使用Aggregation数据集。将labeled的数据占比分别设置为0.1与0.3，并将迭代终止条件设置为 while changed > 0 ，分别观察数据结果与迭代次数。

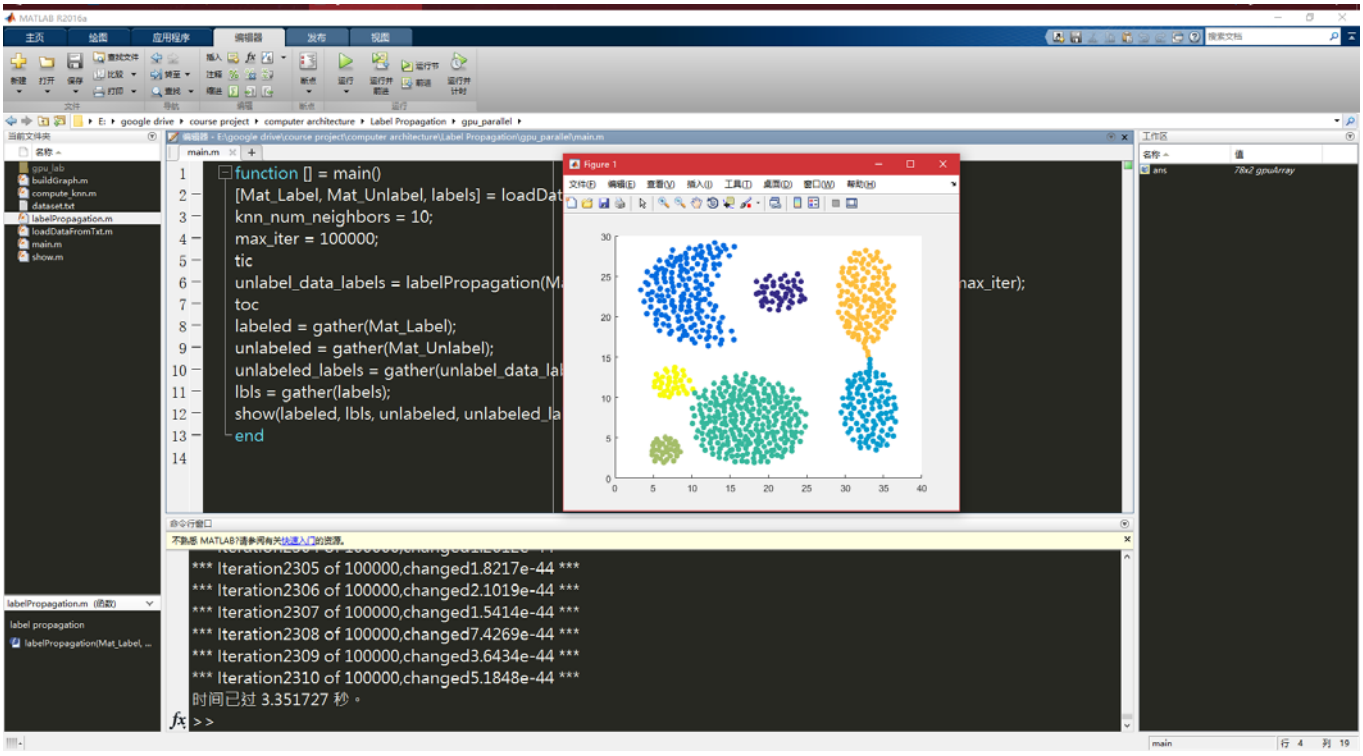

FIG. 2 labeled占比0.1时算法运行至完全收敛之结果

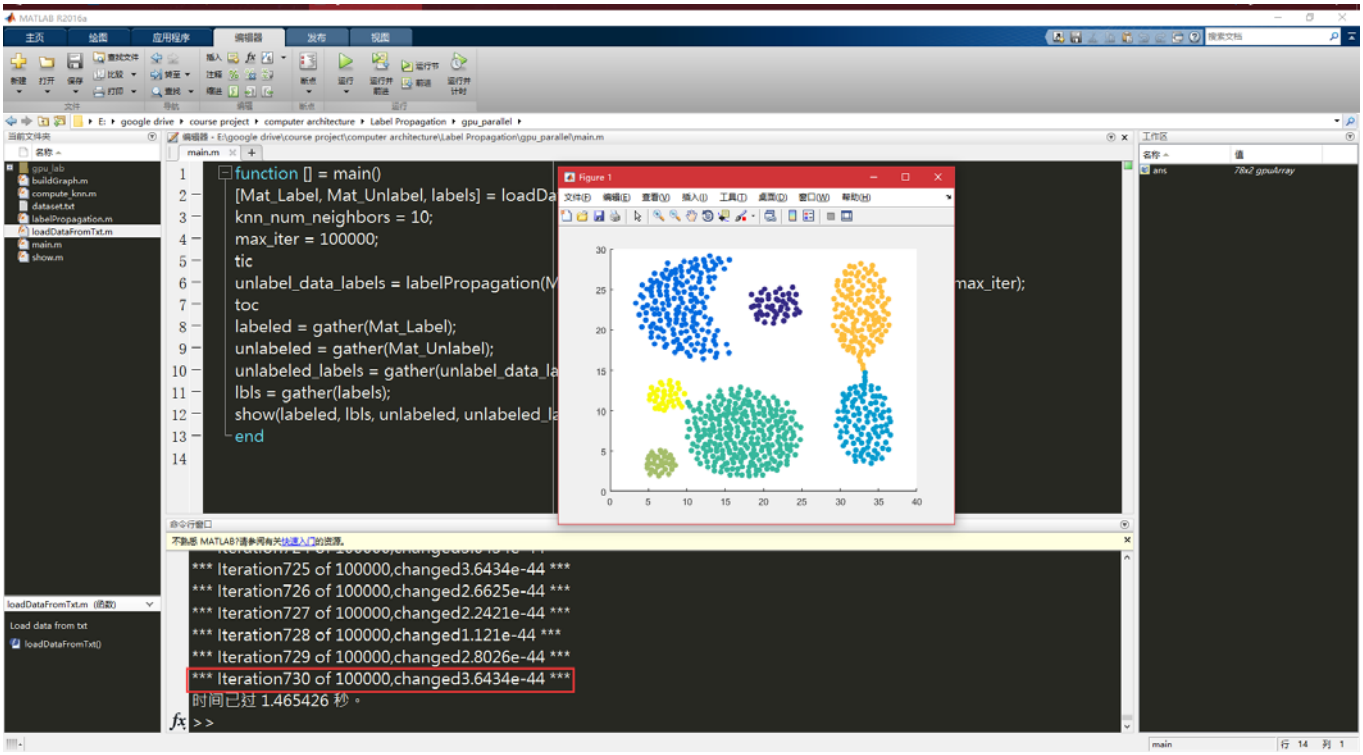

FIG. 3 labeled占比0.3时算法运行至完全收敛之结果

FIG. 2 与 FIG. 3分别展示了labeled数据占比分别设置为0.1与0.3的实验结果。从图中不难看出，在两次实验中，数据均被较好的标记，但当labeled数据占比较低时，所需的迭代次数较多。

2.4 并行算法加速比

在本实验中，我们使用Aggregation数据集。我们分别控制串行算法与并行算法的labeled数据占比为0.1，并将迭代终止条件设置为while (iter < 100000)，分别计算串行算法与并行算法执行所需时间，最终求得加速比。

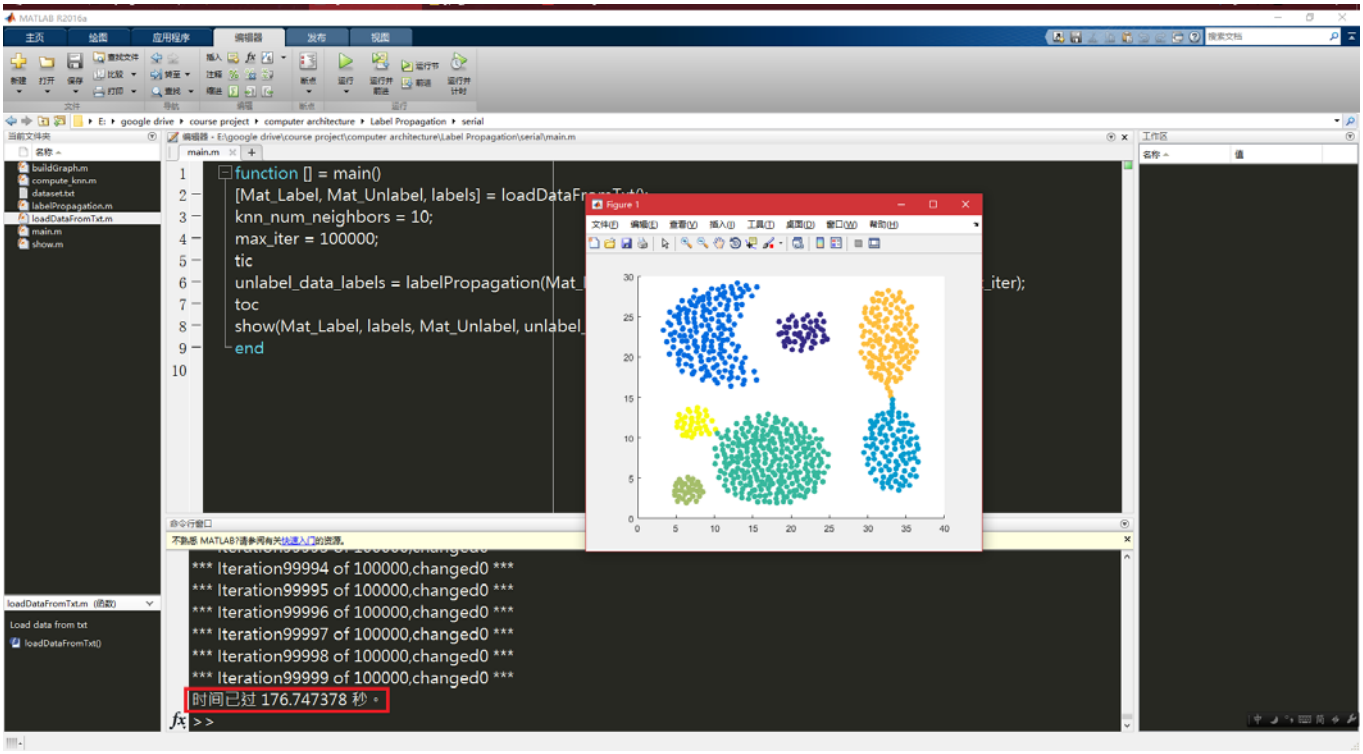

FIG. 4 迭代100000次串行算法所用时间

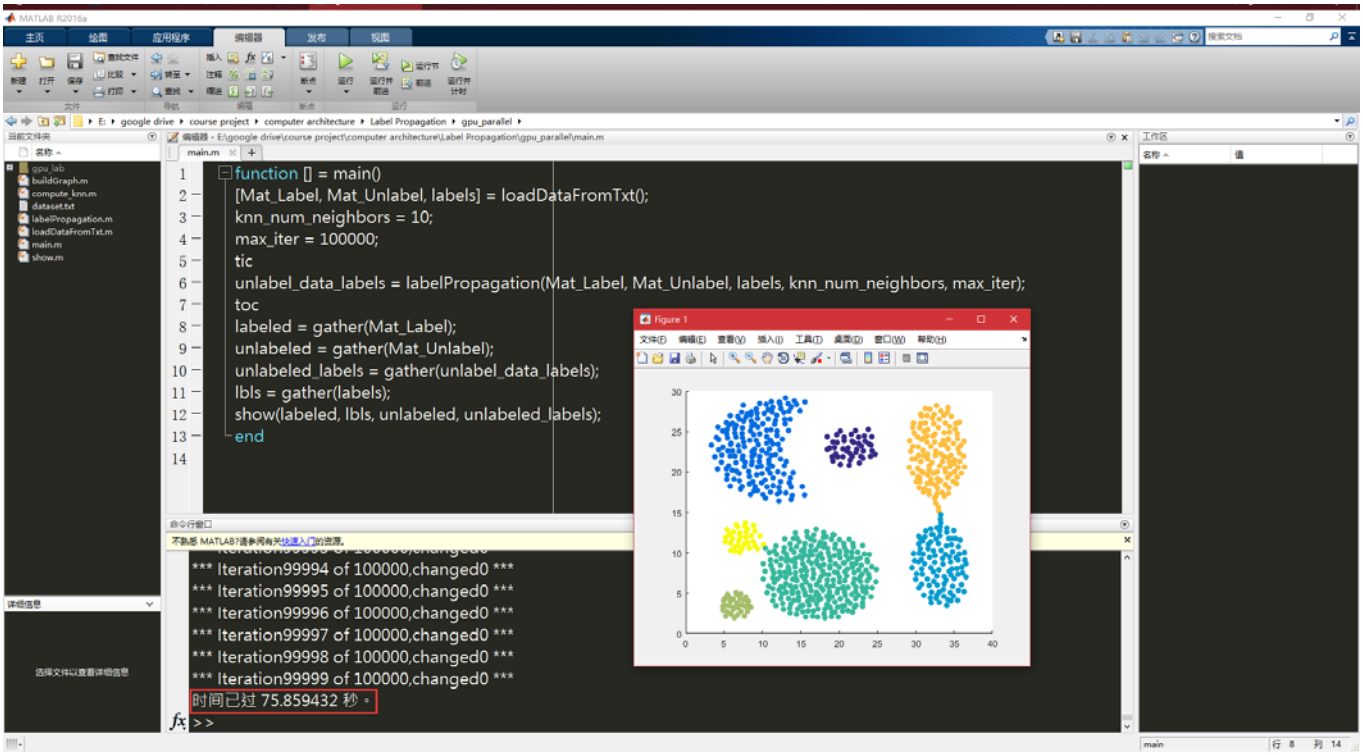

FIG. 5 迭代100000次并行算法所用时间

FIG. 4 与FIG. 5 分别展示了算法迭代100000次串行算法和并行算法分别所花费的时间。其中，串行算法耗时176.75s，并行算法耗时75.86s。因此，并行算法之加速比为R = T0/Tp = 176.75/75.86 = 2.323。

3. 参考与开源

3.1 参考

在构建算法的过程中，我们参考了以下文档及其相关算法。

Y. Zhang and Z.-H. Zhou. Non-metric label propagation. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09), Pasadena, CA, 2009, pp.1357-1362.
http://lamda.nju.edu.cn/code_NMLP.ashx

3.2 开源

该算法在 Apache License 2.0 协议下开源至GitHub：https://github.com/Li-GuangCheng/Label-Propagation_knn。