基于 consR 的并行图匹配方法

田豪爽[1],戴东波[1,2],张惠然[1,2],谢 江[1,2]
(1. 上海大学 计算机工程与科学学院,上海 200444;
2. 上海大学 高性能计算中心,上海 200444)

摘 要:随着社交网络、生物网络规模的迅速扩大,能够快速、高效地实现对这些网络的匹配、查询等工作已经成为许多应用领域的迫切需求。给定两个网络图,图匹配的过程即为对图 G_1 中的每个节点在图 G_2 中找到唯一一个相对应的最为相似的节点,使得给定的两个图的匹配边的数量最多。文中基于大图匹配方法 consR,进行了两方面的优化:当图的节点数目较少时,优化了图 G_1、G_2 的相似性矩阵计算策略,从而使得图匹配的计算更加快捷;当图的节点数目较大时,针对匹配过程中最为耗时的步骤进行并行优化处理。实验结果表明,在与 consR 方法计算出的匹配结果保持一致的情况下,一定程度上缩短了图匹配计算时间。

关键词:图匹配;consR 算法;归并排序;并行化
中图分类号:TP391;TP311 **文献标识码**:A **文章编号**:1673-629X(2015)07-0020-07
doi:10.3969/j.issn.1673-629X.2015.07.005

Parallel Graph Matching Method Based on consR Algorithm

TIAN Hao-shuang[1], DAI Dong-bo[1,2], ZHANG Hui-ran[1,2], XIE Jiang[1,2]
(1. School of Computer Engineering and Science, Shanghai University, Shanghai 200044, China;
2. High Performance Computing Center, Shanghai University, Shanghai 200044, China)

Abstract: With the rapid enlargement of the size of social network and biology network, how to implement the network matching and querying fast and efficiently has become an urgent need in many application research. Given two graphs G_1 and G_2, the task of graph matching is to find, for each node in G_1, the most similar node in G_2 to maximize the number of matched edges. In this paper, based on the effective large graph matching method consR, improve its efficiency from two aspects. When the number of nodes in the graph is relatively small, the computation cost for similarity matrix of G_1 and G_2 is reduced by a simplified yet effective computing strategy. When the number of nodes is large, adopt parallel strategy to speed up the matching procedure for large graphs. The experimental results show that the proposed method reduces the computation time of image matching at a certain extent while gets as good matching results as consR does.

Key words: graph matching; consR algorithm; merge sort; parallelism

0 引 言

图匹配在许多应用领域都发挥着越来越重要的作用。在生物信息领域,随着人类基因组测序工作的完成,生物信息数据呈爆炸式增长,蛋白质分子相互作用网络、基因调控网络等都可以用图的形式来表示。对这些复杂生物网络用图匹配的方法进行生物网络的比对,就能够找出这些网络之间的异同点,从而帮助发现理解不同物种的蛋白质、基因之间的相互作用的共同和不同之处,揭示物种进化原理[1-2]。在生物化学领域,给定同一个受体相互作用的多个配体组成的集合,对于这多个配体进行结构的比对,找出共同的结构特征,即为研究者要找寻的药效团(药效结构)[3]。在 2D&3D 图像分析领域,形状匹配是众多学者研究的一个热点,给定一个对象,通过合理的方法将其与已经存储的范本进行比对,从而实现对象识别[4]。在视频分析领域,可以借助于动态图匹配的方法来实现多目标

收稿日期:2014-07-05 **修回日期**:2014-10-07 **网络出版时间**:2015-06-23
基金项目:国家自然科学基金重大研究计划项目(91330116);上海市科委重点项目(11510500300);高等学校博士学科点专项科研基金(20113108120022)
作者简介:田豪爽(1991-),男,硕士研究生,CCF 会员,研究方向为生物信息、高性能计算;戴东波,博士,讲师,CCF 会员,研究方向为生物信息、高性能计算;张惠然,博士,讲师,CCF 会员,研究方向为生物信息、高性能计算;谢 江,博士,副教授,CCF 会员,研究方向为生物信息、高性能计算。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150623.1031.024.html

跟踪[5]。在这些不同的领域中,图的规模是在不断地增长扩大的,而图匹配是 NP-完全问题。因此,如何能够快速、高效地实现图匹配成为了一个十分关键的问题。

图匹配问题可以分为两类:精确图匹配问题和不精确图匹配问题[6]。精确图匹配为给定两个图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$,其中 $|V_1|\leqslant|V_2|$,找出一个一对一映射 $f:V_1\rightarrow V_2$ 使得每一条 G_1 的边 $(u,v)\in E_1$ 均有 $(f(u),f(v))\in E_2$ 与其相对应。Marlon 等人就通过使用邻接矩阵的方法来实现快速精确图匹配[7]。不精确图匹配相对于精确图匹配而言,放松了匹配的条件,允许 E_1 中存在未匹配的边,即存在 $(u,v)\in E_1$,并且 $(f(u),f(v))\notin E_2$ [8]。SAGA[9] 就是一种不精确的图匹配方法,该方法允许有出现未被匹配的节点、错误匹配的节点以及匹配的结构差异性,放松了匹配的条件,主要用于查询生物学保守路径。Closure-Tree[10] 则是通过使用启发式的非精确图匹配方法计算查询图和待匹配图之间的距离来衡量图的相似性。SIGMA[11] 是采用基于特征集的高效过滤算法,将问题转化为用重叠多重集来覆盖图的缺失特征问题,用于实现非精确图匹配。由此可见,非精确图匹配问题已经成为了近年来诸多学者关注的焦点。而文中将要优化的算法 consR[12]—高效率、高质量的大图匹配方法,同样用于解决非精确图匹配问题。

1　图匹配问题陈述

图匹配即为给定两个图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$,假定图 G_1 的节点个数小于等于图 G_2 的节点个数,即 $|V_1|\leqslant|V_2|$,则匹配 M 为在图 G_1 的节点集 V_1 和图 G_2 的节点集 V_2 之间的一种单射关系,即 $M=\{(u,v)\mid u\in V_1,v\in V_2\}$,$M$ 满足 $\forall(u_1,v_1)\in M$,$\forall(u_2,v_2)\in M$,如果 $u_1\neq u_2$,则 $v_1\neq v_2$。由此可见,给定两个图 G_1 和 G_2,两者之间存在着 $|v_2|!/(|v_2|-|v_1|)!$ 种不同的图匹配方式。对于一个匹配 M,对于 $(u_1,v_1)\in M$,$(u_2,v_2)\in M$,如果 u_1 和 u_2 之间存在边 $e_{u_1u_2}$,v_1 和 v_2 之间存在边 $e_{v_1v_2}$,则称 $e_{u_1u_2}$ 和 $e_{v_1v_2}$ 为匹配边。从 $|v_2|!/(|v_2|-|v_1|)!$ 个不同匹配中选出匹配边数量最大的匹配作为要寻求的最优图匹配结果。即最优匹配 M 使得匹配边的数量 Num(M)[12] 最大。

$$\mathrm{Num}(M)=\frac{\sum_{(u_1,v_1)\in M}\sum_{(u_2,v_2)\in M}e_{u_1u_2}e_{v_1v_2}}{2}\tag{1}$$

式中,节点 u_1 和 u_2 之间若有边直接相连,则 $e_{u_1u_2}=1$,否则,$e_{u_1u_2}=0$;同理,节点 v_1 和 v_2 之间若有边直接相连,则 $e_{v_1v_2}=1$,否则,$e_{v_1v_2}=0$。

图 1 中的 G_1 和 G_2 之间的最优匹配 $M=\{(v_1,u_1),(v_2,u_2),(v_3,u_3),(v_4,u_4)\}$,使得 Num($M$)达到最大值 4,即最大匹配的边的数量为 4。而匹配 $M'=\{(v_1,u_1),(v_2,u_5),(v_3,u_3),(v_4,u_4)\}$ 则不是最优匹配,因为 Num(M')的值为 3,小于 $\mathrm{Max}_{\mathrm{Num}(M)}$ 的值 4。

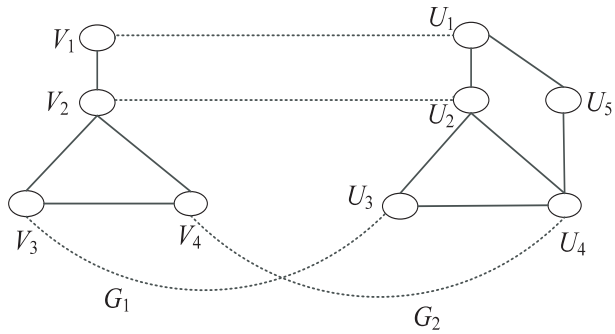

图 1　图 G_1 和图 G_2 的匹配 M

图匹配的过程主要分为两个步骤:

(1)计算图 G_1 和 G_2 的相似性矩阵(见表 1);

(2)根据表 1 中的相似性矩阵,力求从每一行找出相似性值最大的点对(黑色标记的点对)加入匹配 M。详细的计算过程将在后面介绍。

表 1　图 G_1 和 G_2 的相似性矩阵

节点	u_1	u_2	u_3	u_4	u_5
v_1	**0.51**	0.22	0.16	0.22	0.62
v_2	0.36	**0.62**	0.24	0.52	0.36
v_3	0.62	0.59	**0.76**	0.59	0.62
v_4	0.62	0.59	0.76	**0.79**	0.62

2　consR 图匹配方法的优化及其并行化

consR 图匹配方法是由朱等人提出的非精确图匹配方法。该图匹配方法主要由两部分组成:匹配构建(matching construction)和匹配优化(matching refinement)。文中将针对此方法的某些部分进行算法优化,对于耗时的部分进行并行优化。

2.1　图匹配构建和优化方法(consR)

consR 方法主要包括四个步骤,首先是计算图 G_1 和 G_2 之间的相似性矩阵,然后根据相似性矩阵找出关键点对[13],再次基于关键点对优先匹配与关键点对关联的其他点对,最后再进行匹配优化。其中,前三个步骤属于图匹配构建的过程。

2.1.1　计算相似性矩阵

在 consR 方法中,相似性矩阵的定义见公式(2)。

$$S[u,v]=S_g[u,v]*S_l[u,v]\tag{2}$$

其中,$S_g[u,v]$ 是衡量节点 u 和 v 在图 G_3 和 G_4 中的全局相似性值;$S_l[u,v]$ 是衡量节点 u 和 v 分别在其邻居节点组成的子图中的局部相似性值。

$S_g[u,v]$ 的计算是由 UMEYAMA 提出的使用特征分解的方法来计算全局相似性矩阵[14]。以图 2 所示

G_3 和 G_4 为例,来分别说明计算 $S_g[u,v]$ 和 $S_1[u,v]$ 的过程。

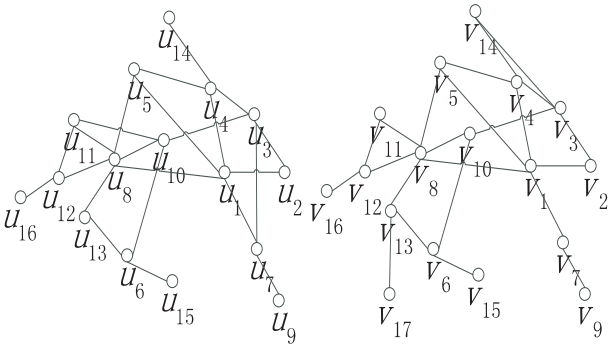

图 2 图 G_3 和 G_4

在计算全局相似性矩阵时,对于图 G_3 和 G_4,首先计算它们的拉普拉斯矩阵 $\boldsymbol{L}$。

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{3}$$

其中,$\boldsymbol{D}$ 为对角度矩阵,即 $\boldsymbol{D}$ 主对角线上的元素分别为各个节点的度数,其他元素值为零;$\boldsymbol{A}$ 为邻接矩阵。

之后对矩阵 $\boldsymbol{L}$ 进行特征值分解。

$$\boldsymbol{L}_3 = \boldsymbol{U}_3\boldsymbol{\Lambda}_3\boldsymbol{U}_3^{\mathrm{T}}, \boldsymbol{L}_4 = \boldsymbol{U}_4\boldsymbol{\Lambda}_4\boldsymbol{U}_4^{\mathrm{T}} \tag{4}$$

其中,$\boldsymbol{\Lambda}_3$ 和 $\boldsymbol{\Lambda}_4$ 为对角特征值矩阵。

$$\boldsymbol{\Lambda}_3 = \mathrm{diag}(\alpha_i), \boldsymbol{\Lambda}_4 = \mathrm{diag}(\beta_i) \tag{5}$$

$\boldsymbol{L}_3$ 和 $\boldsymbol{L}_4$ 的特征值分别为 $\alpha_1 \geqslant \alpha_2 \geqslant \cdots \geqslant \alpha_n$ 和 $\beta_1 \geqslant \beta_2 \geqslant \cdots \geqslant \beta_n$,$\boldsymbol{U}_3$ 和 $\boldsymbol{U}_4$ 由 $\boldsymbol{L}_3$ 和 $\boldsymbol{L}_4$ 进行特征分解得到。$\bar{\boldsymbol{U}}_3$ 和 $\bar{\boldsymbol{U}}_4$ 为对矩阵 $\boldsymbol{U}_3$ 和 $\boldsymbol{U}_4$ 中的每个元素取绝对值。

$$\bar{\boldsymbol{U}}_3 = |\boldsymbol{U}_3|, \bar{\boldsymbol{U}}_4 = |\boldsymbol{U}_4| \tag{6}$$

在公式(7)中,$\bar{\boldsymbol{U}}_3'$ 和 $\bar{\boldsymbol{U}}_4'$ 分别取 $\bar{\boldsymbol{U}}_3$ 和 $\bar{\boldsymbol{U}}_4$ 的前 c 列,参数 c 的定义见公式(8)。图 G_3 和 G_4 的全局相似性矩阵定义为矩阵 $\bar{\boldsymbol{U}}_3'$ 乘上矩阵 $\bar{\boldsymbol{U}}_4'$ 的转置(见公式(7))。

$$S_g = \bar{\boldsymbol{U}}_3'\bar{\boldsymbol{U}}_4'^{\mathrm{T}} \tag{7}$$

$$c = \min\{|V(G_3)|, |V(G_4)|\} \tag{8}$$

如果仅根据全局相似性矩阵计算出图 G_3 和 G_4 之间的匹配 M,Num(M)的值为 10,远远小于 G_3 和 G_4 的最大匹配边数量 21。因此朱等人又提出了通过计算局部相似性矩阵 $S_l[u,v]$ 来优化 UMEYAMA 特征分解方法计算出的全局相似性矩阵。局部相似性矩阵中的每个元素计算定义如下:

$$S_l[u,v] = \frac{(n_{\min} + 1 + D(u,v))^2}{(|V(G_u^k)| + |E(G_u^k)|)(|V(G_v^k)| + |E(G_v^k)|)} \tag{9}$$

$$n_{\min} = \min\{N_k(u), N_k(v)\} \tag{10}$$

$$D(u,v) = \frac{\min\{d(u), d(v)\} + \sum_{i=1}^{n_{\min}} \min\{d_{1,i}, d_{2,i}\}}{2} \tag{11}$$

其中,$N_k(u)$ 是指节点 u 在图 G_3 中的 k -邻居节点集,其定义为由节点 u 的一阶邻居节点(与节点 u 的最短路径距离为 1 的节点)和节点 u 的二阶邻居节点(与节点 u 的最短路径长度距离为 2 的节点),一直到节点 u 的 k 阶邻居节点组成的集合。同理,$N_k(v)$ 是指节点 v 在图 G_4 的 k -邻居节点集。G_u^k 是指 $N_k(u)$ 和节点 u 的并集在 G_3 中的诱导子图,G_v^k 是指 $N_k(v)$ 和节点 v 的并集在 G_4 中的诱导子图,$n_{\min}$ 取节点 u 的 k -邻居节点数目和节点 v 的 k -邻居节点数目较小的一个值。$d(u)$ 和 $d(v)$ 指节点 u 和 v 各自在图 G_3 和 G_4 中的度数。序列 $d_{1,1}, d_{1,2}, \cdots$ 是 $N_k(u)$ 节点集中的节点在诱导子图 G_u^k 中的度数按照非增序排列的节点度数序列,$d_{2,1}, d_{2,2}, \cdots$ 是 $N_k(v)$ 节点集中的节点在诱导子图 G_v^k 中的度数按照非增序排列的节点度数序列。

根据 $S_l[u,v]$ 的定义,$D(u,v)$ 表示的含义为将 u 在图 G_3 中的 k -邻居节点诱导子图与节点 v 在图 G_4 中的 k -邻居节点诱导子图匹配能够得到的理想匹配边的数目。$n_{\min} + 1$ 表示节点 u 的 k -邻居节点诱导子图节点数目和节点 v 的 k -邻居节点诱导子图节点数目较小的一个值。因此,如果节点 u 在图 G_3 中的 k -邻居节点诱导子图和节点 v 在图 G_4 中的 k -邻居节点诱导子图完全同构的话,将有 $n_{\min} + 1 = |V(G_u^k)| = |V(G_v^k)|, D(u,v) = |E(G_u^k)| = |E(G_v^k)|$,则 $S_l[u,v]$ 的值将为 1;如果两者结构差异性越大,$S_l[u,v]$ 的值则越小。

在计算全局相似性矩阵和局部相似性矩阵的过程中,较为耗时的部分为计算局部相似性矩阵的过程。假定 N_3 为图 G_3 的节目数目,N_4 为图 G_4 的节点数目。如果图 G_3 和 G_4 都是完全子图的话,那么在计算图 G_3 中的所有节点的 k -阶邻居子图的相关数据时,时间复杂度将达到 $O(N_3^3)$。同理,图 G_4 的最大时间复杂度将达到 $O(N_4^3)$。因此,将对此过程进行并行化处理。

2.1.2 基于相似性矩阵寻找关键点对

在一个网络中的关键点是指在网络度数较高、与其他节点关联性较强的节点,换言之,关键点相当于网络中的 Hub 节点[15]。在 consR 方法中,关键点对 (u, v) 的定义如下:

$$\min\{d(u), d(v)\} \geqslant \max\left\{\frac{2 * |E(G_3)|}{|V(G_3)|}, \frac{2 * |E(G_4)|}{|V(G_4)|}\right\} \tag{12}$$

$$S[u,v] \geqslant \tau \tag{13}$$

公式(12)的含义为关键点对中两个节点的度数

均大于图 G_3 平均节点度数和 G_4 平均节点度数之间的较大值。公式(13)的含义则是节点 u,v 在前面计算出的相似性矩阵中对应的值不小于阈值 τ 。consR 方法中给出了找寻关键点对的步骤:

给定:图 G_3 和图 G_4,关键点对集 A 。

1:计算相似性矩阵 $\boldsymbol{S}$;

2: $A \leftarrow \varnothing$; $S_1 \leftarrow \varnothing$; $S_2 \leftarrow \varnothing$;

3:for all $u \in V(G_3)$, $v \in V(G_4)$ 将它们对应在相似性矩阵 $S[u,v]$ 中的值按照降序进行排列

if $S[u,v] \geqslant \tau$ and

$\min\{d(u),d(v)\} \geqslant$

$$\max\left\{\frac{2 * |E(G_3)|}{|V(G_3)|}, \frac{2 * |E(G_4)|}{|V(G_4)|}\right\}$$

and $u \notin S_1$ and $v \notin S_2$

then

$A \leftarrow A \cup \{(u,v)\}$; $S_1 \leftarrow S_1 \cup \{(u)\}$; $S_2 \leftarrow S_2 \cup \{(v)\}$;

return A ;

不难看出,在关键点对选择过程中,需要将 $|V(G_3)| * |V(G_4)|$ 大小的相似性矩阵 $\boldsymbol{S}[u,v]$ 中的值按照降序排列,当图 G_3 和 G_4 的规模扩大后,如果仍使用串行排序的话,排序时间将会很慢。因此,在此处使用并行归并排序[16]来代替串行排序。

2.1.3　基于关键点对匹配其他点对和匹配优化

基于关键点对匹配其他点对的过程在 consR 中被称作关键点对-膨胀。这一过程即为优先选择与关键点对 (u,v) 相邻的节点对 $N(u) \times N(v)$ 加入到匹配集合 M 中,其中 $N(u)$ 表示节点 u 在图 G_3 中的邻居节点, $N(v)$ 表示节点 v 在图 G_4 中的邻居节点。

图匹配优化(Matching Refinement)则是基于图节点覆盖集[11]对图匹配创建(Matching Construction)得到的初始图匹配结果进行优化。

在 consR 方法中,基于关键点对-膨胀(anchor-expansion)的时间约为关键点对-选择(anchor-selection)时间的 1/38,而图匹配优化的时间约为关键点对-选择时间的 1/6。因此,对于这两部分,不再进行详细介绍和并行优化。

2.2　consR 图匹配方法的简化

对 consR 图匹配方法的简化主要是针对匹配图的节点数较少时进行的。以图 G_3 和 G_4 为例,利用特征分解的办法计算它们的全局相似性矩阵 $\boldsymbol{S}_g$ 见图 3。

$$\begin{pmatrix}
\boxed{0.86} & 0.47 & 0.72 & 0.70 & 0.65 & 0.52 & 0.42 & 0.75 & 0.32 & 0.57 & 0.49 & 0.51 & 0.47 & 0.39 & 0.34 & 0.38 & 0.31 \\
0.52 & 0.76 & 0.39 & 0.44 & 0.69 & 0.48 & 0.71 & 0.38 & 0.65 & 0.52 & 0.67 & 0.39 & 0.41 & 0.67 & 0.47 & 0.51 & 0.40 \\
0.63 & 0.51 & 0.66 & 0.64 & 0.66 & 0.62 & 0.63 & 0.49 & 0.49 & 0.47 & 0.56 & 0.63 & 0.44 & 0.48 & 0.37 & 0.44 & 0.29 \\
0.52 & 0.47 & 0.69 & 0.74 & 0.61 & 0.74 & 0.40 & 0.58 & 0.34 & 0.57 & 0.41 & 0.61 & 0.56 & 0.49 & 0.54 & 0.38 & 0.46 \\
0.54 & 0.54 & 0.64 & 0.62 & 0.81 & 0.59 & 0.47 & 0.47 & 0.42 & 0.67 & 0.53 & 0.40 & 0.54 & 0.49 & 0.51 & 0.37 & 0.48 \\
0.39 & 0.48 & 0.57 & 0.60 & 0.57 & 0.64 & 0.52 & 0.38 & 0.54 & 0.63 & 0.55 & 0.78 & 0.59 & 0.57 & 0.49 & 0.52 & 0.46 \\
0.38 & 0.37 & 0.44 & 0.36 & 0.45 & 0.62 & 0.60 & 0.35 & 0.49 & 0.59 & 0.36 & 0.49 & 0.74 & 0.38 & 0.53 & 0.38 & 0.58 \\
0.75 & 0.43 & 0.48 & 0.60 & 0.54 & 0.40 & 0.42 & \boxed{0.91} & 0.35 & 0.52 & 0.36 & 0.34 & 0.41 & 0.40 & 0.28 & 0.28 & 0.26 \\
0.22 & 0.37 & 0.25 & 0.23 & 0.32 & 0.48 & 0.43 & 0.25 & 0.49 & 0.36 & 0.51 & 0.42 & 0.53 & 0.46 & 0.79 & 0.69 & 0.84 \\
0.70 & 0.56 & 0.77 & 0.80 & 0.80 & 0.58 & 0.64 & 0.58 & 0.49 & 0.78 & 0.43 & 0.40 & 0.56 & 0.58 & 0.39 & 0.33 & 0.37 \\
0.51 & 0.60 & 0.72 & 0.72 & 0.78 & 0.78 & 0.48 & 0.45 & 0.50 & 0.70 & 0.49 & 0.53 & 0.52 & 0.65 & 0.56 & 0.44 & 0.42 \\
0.34 & 0.49 & 0.57 & 0.46 & 0.54 & 0.86 & 0.53 & 0.41 & 0.57 & 0.63 & 0.37 & 0.57 & 0.78 & 0.54 & 0.56 & 0.41 & 0.53 \\
0.46 & 0.63 & 0.38 & 0.43 & 0.53 & 0.53 & 0.52 & 0.32 & 0.57 & 0.67 & 0.54 & 0.49 & 0.49 & 0.60 & 0.52 & 0.47 & 0.43 \\
0.26 & 0.40 & 0.31 & 0.32 & 0.40 & 0.50 & 0.31 & 0.35 & 0.38 & 0.33 & 0.64 & 0.45 & 0.44 & 0.48 & 0.78 & 0.71 & 0.79 \\
0.33 & 0.64 & 0.44 & 0.45 & 0.47 & 0.47 & 0.50 & 0.26 & 0.69 & 0.45 & 0.65 & 0.56 & 0.44 & 0.74 & 0.57 & 0.79 & 0.57 \\
0.32 & 0.60 & 0.49 & 0.44 & 0.50 & 0.55 & 0.52 & 0.25 & 0.73 & 0.49 & 0.54 & 0.45 & 0.52 & 0.71 & 0.63 & 0.69 & 0.65
\end{pmatrix}$$

图 3　图 G_3 和 G_4 的全局相似性矩阵 $\boldsymbol{S}_g$

$$\boldsymbol{S}_g[u,v] \geqslant \tau \tag{14}$$

不再计算局部相似性矩阵,而是直接根据全局相似性矩阵 $\boldsymbol{S}_g$ 来找关键点对。依据公式(12)、(14),设定阈值 $\tau = 0.85$,这样就得到关键点对 (u_1,v_1),(u_8,v_8)(相似性矩阵 $\boldsymbol{S}_g$ 中用方框标记)。接下来对于一个关键点对 (u,v) ,再次用特征分解的方法计算 $u \cup N_k(u)$ 的诱导子图 G_u^k 和 $v \cup N_k(u)$ 的诱导子图 G_v^k 之间的相似性矩阵 $\boldsymbol{S}_g'$ 。这一过程中,排除了图 G_3 中 $\{u' \mid u' \neq u, u' \notin N_k(u)\}$ 和图 G_4 中 $\{v' \mid v' \neq v, v' \notin N_k(v)\}$ 即非节点 u、v 的 k -邻居节点对 $N_k(u)$ 和 $N_k(v)$ 的结构相似性的干扰。设定 $k=2$, u_1、v_1 的 k -邻居子图之间的相似性矩阵 $\boldsymbol{S}_g'$ 如图 4 所示。

基于相似性矩阵 $\boldsymbol{S}_g'$,使用匈牙利算法[17]找出图 G_3 和 G_4 之间的匹配结果为 $M' = \{(u_1,v_1),(u_2,v_2),(u_3,v_3),(u_4,v_4),(u_5,v_{10}),(u_7,v_7),(u_8,v_8),(u_9,v_9),(u_{10},v_5),(u_{11},v_{11}),(u_{12},v_{12}),(u_{13},v_{13}),(u_{14},v_{14})\}$,由于图 $G_{u_1}^2$ 中没有节点 u_6,u_{15},u_{16} ,因此 $\boldsymbol{S}_g'$ 中对应的第 6、15、16 行的值均为 0。$G_{u_1}^2$ 没有节点 v_6,v_{15},v_{16},v_{17} ,故 6、15、16、17 列的值为 0。在已经得到的匹配 M' 基础上,改进了文献[18]中的公式得到公式(15),在节点对 $\{u_6,u_{15},u_{16}\} \times \{v_6,v_{15},v_{16},v_{17}\}$ 找出剩余节点间的匹配。

$$S(u_n,v_n) = 0.5 * \frac{\min(d(u_n),d(v_n))}{\max(d(u_n),d(v_n))} + 0.5 * \frac{n_M(u_n,v_n)}{\max(|\Gamma(u_n)|,|\Gamma(v_n)|)} \tag{15}$$

其中,第一部分表示两个节点之间的度数差异;第二部分中, $n_M(u_n,v_n)$ 表示在前面得到的匹配 M' 中与

节点 u_n,v_n 分别连接的已匹配点对的数量,$\Gamma(u_n)$ 则表示与节点 u_n 直接相连的节点数量。根据公式(15),得到剩余节点之间的匹配结果 $M' = \{(u_6,v_6),(u_{15},v_{15}),(u_{16},v_{16})\}$。对于图 G_3 和 G_4,得到的图匹配结果 $M' \cup M''$ 与最优匹配结果相比,只有 (u_5,v_{10}) 点对匹配错误,正确率达到 90% 以上,并且与 consR 方法比省去了匹配优化的步骤。

$$\begin{pmatrix}
0.87 & 0.41 & 0.64 & 0.67 & 0.60 & 0.00 & 0.34 & 0.67 & 0.21 & 0.52 & 0.38 & 0.38 & 0.24 & 0.32 & 0.00 & 0.00 & 0.00\\
0.51 & 0.74 & 0.37 & 0.46 & 0.66 & 0.00 & 0.56 & 0.33 & 0.51 & 0.67 & 0.38 & 0.38 & 0.40 & 0.68 & 0.00 & 0.00 & 0.00\\
0.69 & 0.51 & 0.87 & 0.69 & 0.70 & 0.00 & 0.52 & 0.43 & 0.36 & 0.49 & 0.54 & 0.54 & 0.29 & 0.44 & 0.00 & 0.00 & 0.00\\
0.62 & 0.41 & 0.82 & 0.85 & 0.58 & 0.00 & 0.30 & 0.53 & 0.24 & 0.53 & 0.50 & 0.50 & 0.25 & 0.54 & 0.00 & 0.00 & 0.00\\
0.55 & 0.30 & 0.62 & 0.67 & 0.66 & 0.00 & 0.42 & 0.32 & 0.31 & 0.79 & 0.20 & 0.20 & 0.20 & 0.50 & 0.00 & 0.00 & 0.00\\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00\\
0.36 & 0.33 & 0.36 & 0.44 & 0.54 & 0.00 & 0.79 & 0.39 & 0.65 & 0.27 & 0.51 & 0.51 & 0.24 & 0.37 & 0.00 & 0.00 & 0.00\\
0.72 & 0.30 & 0.42 & 0.55 & 0.39 & 0.00 & 0.35 & 0.90 & 0.27 & 0.40 & 0.31 & 0.31 & 0.25 & 0.24 & 0.00 & 0.00 & 0.00\\
0.24 & 0.46 & 0.30 & 0.31 & 0.35 & 0.00 & 0.68 & 0.26 & 0.82 & 0.27 & 0.57 & 0.57 & 0.56 & 0.54 & 0.00 & 0.00 & 0.00\\
0.52 & 0.60 & 0.64 & 0.70 & 0.78 & 0.00 & 0.47 & 0.46 & 0.41 & 0.56 & 0.48 & 0.48 & 0.42 & 0.53 & 0.00 & 0.00 & 0.00\\
0.32 & 0.47 & 0.52 & 0.58 & 0.51 & 0.00 & 0.43 & 0.33 & 0.45 & 0.37 & 0.68 & 0.68 & 0.42 & 0.61 & 0.00 & 0.00 & 0.00\\
0.39 & 0.71 & 0.34 & 0.39 & 0.60 & 0.00 & 0.59 & 0.35 & 0.62 & 0.37 & 0.62 & 0.62 & 0.50 & 0.60 & 0.00 & 0.00 & 0.00\\
0.23 & 0.27 & 0.21 & 0.23 & 0.19 & 0.00 & 0.33 & 0.26 & 0.45 & 0.26 & 0.57 & 0.57 & 0.87 & 0.31 & 0.00 & 0.00 & 0.00\\
0.33 & 0.56 & 0.50 & 0.49 & 0.33 & 0.00 & 0.29 & 0.27 & 0.43 & 0.43 & 0.59 & 0.59 & 0.61 & 0.71 & 0.00 & 0.00 & 0.00\\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00\\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{pmatrix}$$

图 4 图 $G^2_{u_1}$ 和 $G^2_{v_1}$ 的相似性矩阵 S'_g

2.3 consR 图匹配方法的并行化

2.3.1 计算局部相似性矩阵的并行化

在前面已经提到,计算两个图的局部相似性矩阵过程中,在计算每个节点的 k-邻居节点诱导子图的边数目 $|E(G^k_u)|$、节点数目 $|V(G^k_u)|$ 以及将 $N_k(u)$ 节点集中的节点在诱导子图 G^k_u 中的度数按照非增序排列时,当两个图都为完全图时,计算复杂性达到最高,最大时间复杂度为 $O(N^3)$,即时间复杂度随着图的平均节点度数的增加而增大。N 取两个图节点数目较大的一个。如果使用串行程序进行计算,要依次找到每个节点的 $1-k$ 阶邻居节点,计算这些节点在 k-邻居节点诱导子图中的度数,并进行排序。在计算大图间的局部相似性矩阵时,计算时间将会很慢。因此,对此部分进行 MPI 并行化处理[19]。在并行化时,有以下两个特点:

(1)可以同时计算图 G_1 和 G_2 中的每个节点的 k-邻居节点诱导子图的相关信息。假定一共有 N 个进程,则可以分配 N_1 个进程来计算 G_1 中每个节点的 k-邻居节点诱导子图的相关信息;同理,分配 N_2 个进程计算 G_2 中每个节点的 k-邻居节点诱导子图的相关信息。

$$N_1 = \frac{|V(G_1)|}{|V(G_1)| + |V(G_2)|} * N \tag{16}$$

$$N_2 = \frac{|V(G_2)|}{|V(G_1)| + |V(G_2)|} * N \tag{17}$$

(2)每个节点的 k-邻居节点诱导子图的计算是相互独立的。因此只需把图 G_1 的邻接矩阵分别广播到 $1-N_1$ 进程即可。则 $1-N_1$ 每个进程计算 $\frac{|V(G_1)|}{N_1}$ 个节点的 k-邻居节点诱导子图的相关信息。如若 $|V(G_1)|$ 不能被 N_1 整除,则前 $|V(G_1)|\%N_1$ 个进程多计算一个节点的 k-邻居节点诱导子图。同理,将图 G_1 的邻接矩阵分别广播到 $(N_1+1)-(N_1+N_2)$ 进程,按照同样的方法计算 G_2 中每个节点的 k-邻居节点诱导子图的相关信息。另外,在并行计算过程中,每个进程在计算过程中不需要通信,计算结束后都将计算结果以数组的形式发送回主进程。

不难看出,上述并行过程使用了 MPI 主从模式进行并行程序的设计。将任务平均分配到多个进程进行计算,有效地优化了算法的执行时间,并行加速比将在实验部分给出。

2.3.2 并行归并排序相似性矩阵内的元素

两个图 G_1 和 G_2 的相似性矩阵内一共有 $|V(G_1)|\times|V(G_2)|$ 个元素,假定 N 等于 $|V(G_1)|\times|V(G_2)|$,在所有串行排序中效率最高的快速排序最大时间复杂度为 $O(N^2)$,平均时间复杂度为 $O(N*\log(N))$。当图 G_1 和 G_2 的规模较大时,排序时间将会十分缓慢,因此使用并行归并排序[16]来优化排序时间。

归并排序是采用分治法的一种典型应用,它是十分稳定的排序算法。首先将待排序的序列分为若干个子序列,然后将每个子序列排序,最后再将排好序的子序列进行合并。由此可以看出,归并排序是十分适合并行化的,在 MPI 并行程序中,可以由多个进程来分别将子序列进行排序,同样可以由多个进程进行子序列的合并。可以使用完全二叉树来表示并行归并排序的过程。

在图 5 上排序树中,叶节点(Node 3-6)首先对每

个子序列使用快速排序进行排序后,节点 3、4 将其排序结果发给节点 1,节点 1 对两个子序列进行合并。同理,节点 2 合并节点 5 和 6 排序过的两个子序列。最后,节点 0 合并节点 1、2 处理好的子序列。但是在这样的排序树中,当 3、4、5、6 节点进行排序时,节点 0、1、2 是闲置的,为了充分利用排序树中的每个节点,根据文献[16],使用优化后的排序树(图 5 下)进行归并排序,每个父节点都将自己序列一半的数据发往子节点,另一半数据留在自己节点,这样父节点与子节点同时进行子序列的排序,不仅提高了每个节点的利用率,而且还降低了通信量。

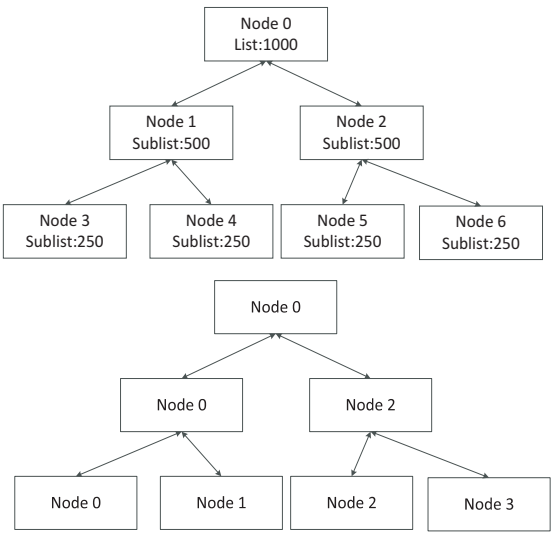

图 5　并行归并排序树(上)和优化后的并行归并排序树(下)

使用优化后的并行排序树结构对相似性矩阵内的元素进行降序排序,排序时间与串行快速排序时间相比,得到了很好的优化。

3　实验结果

当节点数目较小的图进行匹配时,在 2.2 小节,简化了 consR 的步骤,因此计算时间要比 consR 方法小。而匹配边正确率(实际匹配边的数量/最优匹配边的数量)与 consR 方法基本保持一致(见图 6)。实验数据来自于文献[12]提到的美国西部电网数据[20]。在电网中,每个节点为发电机、变压器、变电站等,边则是这些节点间的高压输电线路。使用的网络 g_1 为有 39 个节点,85 条边;网络 g_2 为有 49 个节点,108 条边;网络 g_3 为有 118 个节点,297 条边。用于和 g_1、g_2、g_3 对比的图 g_1'、g_2'、g_3' 分别为随机删除 g_1、g_2、g_3 一部分节点和边而得到。

简化后的 consR 方法在对图 g_2 和 g_2'、g_3 和 g_3' 进行匹配后,匹配边正确率略低于 consR 方法,但是基本上仍保持一致,并且省去了 consR 方法中匹配优化的步骤。图 6 中纵轴表示 consR 方法计算时间/简化后的 consR 计算时间,纵轴值越大,说明简化后节省的计算时间越明显。随着图节点数目的增大,纵轴值在不断减少,即简化后的 consR 方法的计算时间随着节点数目增加在逐步接近 consR 方法的计算时间。因此,简化后的 consR 方法适用于节点数目较小的图匹配,而当图的节点数目较大时,使用后面的并行程序来优化计算时间。

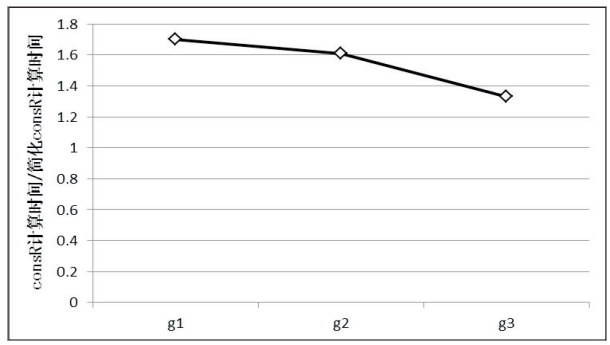

图 6　consR 方法和简化后 consR 方法计算时间比率图

本并行程序在上海大学自强 4000 高性能集群运行,集群共有 140 台 IBM PureFlexx240 刀片节点。每个节点的硬件配置为 2 颗 intel E5 - 2690 CPU (2.9GHz/8 - core),内存为 64 G RDIMM DDR3 1600 MHz,计算网络的带宽为 5.6 GB。操作系统为 CentOS Linux 5.7,程序均是使用 C++编写,编译器为 MPICH2 并行环境下的 g++ 4.4.6。

并行程序实验数据同样是美国西部电网数据。在计算局部相似性矩阵和并行归并排序时,使用的电网规模 G_1 为 5 300 个节点,13 571 条边,平均节点度数为 5。用于同电网 G_1 进行对比的网络 G_2 为随机地删除 G_1 中一部分节点和边而得到。另外,在计算局部相似性矩阵时,为了对比计算不同平均节点度数的图的局部相似性矩阵的加速比,使用 Pajek 得到 5 300 个节点,53 000 条边,平均节点度数为 20 的随机生成图 G_3,与之进行对比的图 G_4 为随机地删除 G_3 中一部分节点和边而得到。

在图 7 和图 8 中的加速比即为同一任务在单处理系统和并行集群系统中运行消耗时间的比率。从图 7 可以看出,在计算美国西部电网 G_1 和 G_2 的局部相似性矩阵时,在核数为 15 时,加速比 $Speedup_1$ 达到最大。而在计算 Pajek 生成图 G_3 和 G_4 的局部相似性矩阵时,在核数为 19 时,加速比 $Speedup_2$ 达到最大,并略高于 $Speedup_1$。由此可以看出,图的平均节点度数越高,并行效果越好。

在图 8 中,并行归并排序相似性矩阵内元素时,在核数为 7 时加速比达到最大。由此可见,本并行程序很好地优化了 consR 方法中耗时部分的计算时间,提

高了运算效率。

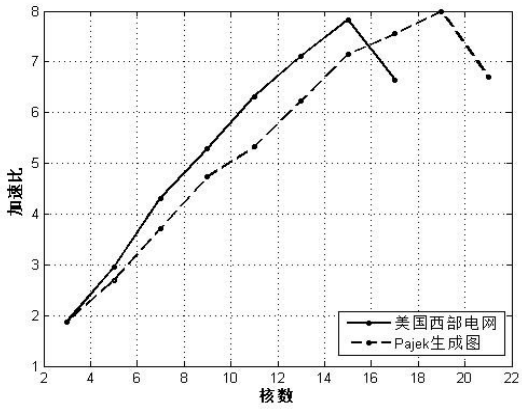

图 7 计算局部相似性矩阵并行加速比

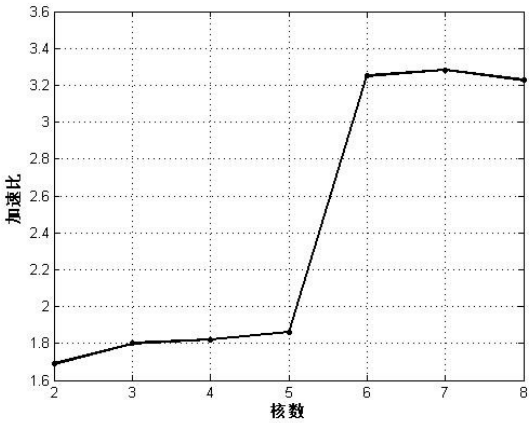

图 8 归并排序相似性矩阵内元素并行加速比

4 结束语

文中在朱等人提出的 consR 大图匹配方法的基础上进行了改进和优化。在图节点数目较小时,使用文中优化的方法可以省去图匹配优化的步骤;在图节点数目较大时,分析了 consR 方法中最为耗时的两部分:计算相似性矩阵和关键点对的选择,并分别对这两部分进行了并行优化。并行实验结果表明,MPI 并行程序对原有的串行方法在执行时间上有了很好的优化。在今后的工作中,将使用 GPU 并行方法来进行大图之间的匹配,并将并行效果与现在的 MPI 方法进行对比。

参考文献:

[1] Li Zhenping, Zhang Shihua, Wang Yong, et al. Alignment of molecular networks by integer quadratic programming[J]. Bioinformatics, 2007, 23(13): 1631-1639.

[2] 马 进,谢 江,戴东波,等. 用于生物分子网络比对的自适应匈牙利贪心混合算法的并行化[J]. 计算机应用, 2013, 33(12): 3321-3325.

[3] Finn P W, Kavraki L E, Latombe Jean-Claude, et al. RAPID: randomized pharmacophore identification for drug design[J]. Computational Geometry, 1998, 10(4): 263-272.

[4] Berg A C, Berg T L, Malik J. Shape matching and object recognition using low distortion correspondences[C]//Proc of IEEE computer society conference on computer vision and pattern recognition. [s. l.]: IEEE, 2005: 26-33.

[5] Chen Hwann-Tzong, Lin Horng-Horng, Liu Tyng-Luh. Multi-object tracking using dynamical graph matching[C]//Proc of IEEE computer society conference on computer vision and pattern recognition. [s. l.]: IEEE, 2001: 210-217.

[6] Livi L, Rizzi A. The graph matching problem[J]. Pattern Anal Appl, 2013, 16(3): 253-283.

[7] Etheredge M. Fast exact graph matching using adjacency matrices[C]//Proceedings of the third workshop on procedural content generation. New York, NY, USA: ACM, 2012.

[8] Riesen K, Jiang Xiaoyi, Bunke H. Exact and inexact graph matching: methodology and applications[J]. Managing and Mining Graph Data Advances in Database Systems, 2010, 40: 217-247.

[9] Tian Yuanyuan, McEachin R C, Santos C, et al. SAGA: a subgraph matching tool for biological graphs[J]. Bioinformatics, 2007, 23(2): 232-239.

[10] He Huahai, Singh A K. Closure-tree: an index structure for graph queries[C]//Proceedings of the 22nd international conference on data engineering. [s. l.]: [s. n.], 2006.

[11] Mongiovì M, Natale R D, Giugno R, et al. Sigma: a set-cover-based inexact graph matching algorithm[J]. Journal of Bioinformatics and Computational Biology, 2010, 8(2): 199-218.

[12] Zhu Yuanyuan, Qin Lu, Yu J X, et al. High efficiency and quality: large graphs matching[J]. International Journal on Very Large Data Bases, 2013, 22(3): 345-368.

[13] Hu Nan, Rustamov R M, Guibas L J. Graph matching with anchor nodes: a learning approach[C]//Proc of 26th IEEE conference on computer vision and pattern recognition. [s. l.]: IEEE, 2013: 2906-2913.

[14] Umeyama S. An eigendecomposition approach to weighted graph matching problems[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1988, 10(5): 695-703.

[15] Ilyas M U, Shafiq M Z, Liu A X, et al. a distributed algorithm for identifying information hubs in social networks[J]. IEEE Journal on Selected Areas in Communications, 2013, 31(9): 629-640.

[16] Rolfe T J. A specimen of parallel programming: parallel merge sort implementation[J]. ACM Inroads, 2010, 1(4): 72-79.

[17] Kuhn H W. The Hungarian method for the assignment problem[M]. [s. l.]: [s. n.], 2010.

[18] Du Fang, Xuan Qi, Wu Tiejun. One-to-many node matching between complex networks[J]. Advances in Complex Systems, 2010, 13(6): 725-739.

[19] 都志辉. 高性能计算之并行技术—MPI 并行程序设计[M]. 北京:清华大学出版社, 2001.