YUSUR 中科驭数 ...free the power of data

驭数道场

有向无环图的划分算法

@中科驭数软件组

2019/7/4

ADAPT Architecture for Data Analytics and Processing Technology
计算机体系结构国家重点实验室 State Key Laboratory of Computer Architecture, ICT, CAS
中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Basic Concept

$G = (V = \{0, \ldots, n-1\}, E, c, \omega)$

edge weights $\omega : E \rightarrow \mathbb{R}_{>0}$

node weights $c : V \rightarrow \mathbb{R}_{\geq 0}$

partition V, i.e., $V_1 \cup \cdots \cup V_k = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$

$c(V') := \sum_{v \in V'} c(v)$ and $\omega(E') := \sum_{e \in E'} \omega(e)$

underloaded [*overloaded*] if $c(V_i) < L_{\max}$ [if $c(V_i) > L_{\max}$]

DAG
n : 节点数
m：边数
权重都大于等于0
L_{max}平衡约束，限制分区大小

Problem Definition

- 平衡约束 $\forall i \in \{1..k\} : c(V_i) \leq \ L_{\max} \ := \ (1+\epsilon)\lceil \frac{c(V)}{k} \rceil$
- 无环约束
- 目标函数 $\sum_{i,j} w(E_{ij}) \text{ where } E_{ij} \ := \ \{(u,v) \in E \ : \ u \in V_i, v \in V_j\}$

Epsilon 不平衡参数 设置成一个比较小的数，k是分区的数量
C(V):一个分区内部节点权重和，
无环约束：分区之间的依赖关系不能成环
定义为找到一个分割，在满足平衡约束和无环约束的条件下，使目标函数最小。

K不是给定的常数，对于这个问题不可能存在一个有限因子近似算法

图划分算法

- ➢启发式算法(SM, AM, GM, FM) (单级算法)
- ➢多级算法(KaFFPa)
- ➢进化算法

图划分属于NP问题，目前没有一个算法可以在多项式时间内算出最优解，所以我们通常用启发式算法，就是通过经验给出一些可行解，再从中得到一个接近最优解的解。

启发式：局部搜索邻域范围不同

多级算法：对多个级别的图进行优化

进化算法：把现有的解作为父代，生成更有子代，类似进化的过程，繁衍多代，最后得到结果。

高级的算法会用低级的算法作为基础。

启发式算法 Heuristic Algorithms

➢Construction Algorithm (Kahn's Algorithm)

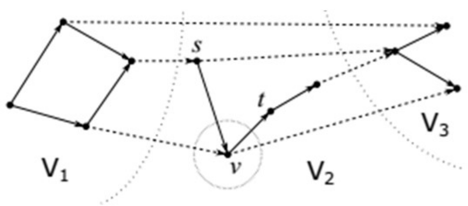
- list S: zero-indegree nodes
- empty list T
- a topological ordering of all nodes $\longrightarrow$ blocks of size $\lfloor \frac{c(V)}{k} \rfloor$ or $\lceil \frac{c(V)}{k} \rceil$

➢Local Search Algorithms (SM, AM, GM, FM)

- gain : reduction of the edge cut

$$C_{in}(v,i) := \omega(\{(u,v) \in E : u \in V_i\})$$

$$C_{out}(v,i) := \omega(\{(v,u) \in E : u \in V_i\})$$



基于拓朴排序得到初始划分

得到大致划分后，在分区之间移动节点，减少边割
能否移动一个节点要看它减少的边割，这里定义为增益gain
C：C_{in}：从分区Vi指向节点v的边的权重和
C_{out} ：从节点v指向分区Vi的边的权重和

SM, AM, GM, FM区别在于搜索邻域范围

Simple Moves (SM)

- 对于$v \in V_i$，仅考虑将其移动到相邻分区V_{i-1}和V_{i+1}
- Gain:

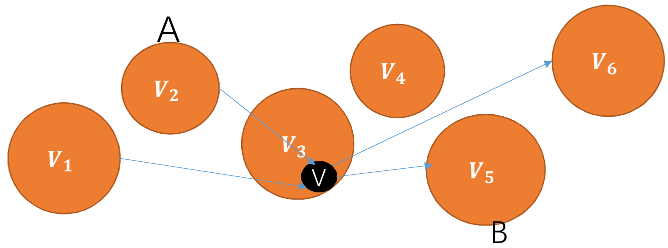
$$\begin{cases} C_{in}(v, i-1) - C_{out}(v, i) & \text{when moving } v \text{ to } V_{i-1} \\ C_{out}(v, i+1) - C_{in}(v, i) & \text{when moving } v \text{ to } V_{i+1}. \end{cases}$$

- 时间复杂度：O(m)

只要移动不创造环，且不超载分区，就是合格的。
增益必须为正，或者为0但能改善平衡，那么就把节点移动过去。
两个方向的移动，都合格，且增益一样，随机选择一个
没有节点能带来正的增益或者改善平衡，算法终止，找到局部极小值
每条边被考虑两次，边数m，O(m)

Advanced Moves (AM)

- 对于$v \in V_i$，检查所有入边以找到节点$u \in V_A$，其中A是最大的。还检查v所有出边以找到节点$w \in V_B$ ，其中B是最小的。
- 考虑将其移动到分区$V_A, \dots, V_{i-1}, V_{i+1}, \dots, V_B$
- Gain： $C_{in}(v,j) - C_{out}(v,i) + C_{out}(v,j) - C_{in}(v,i)$
- 时间复杂度： O(m)



因为初始划分是来自拓扑序的，所以按照这样的移动，同样维持拓扑序。这是充分条件不是必要条件，所以可能存在一些允许的移动被忽略

每次迭代，每条边被考虑两次来计算增益，O(m)
J<i时，后面两项必须为0，否则会产生环；反之同理

Global Moves (GM)

- 时间复杂度$O(m(m_Q+k))$，稀疏图$O(m+nlog\frac{n}{k})$
- m_Q：商图的边的数量

之前说到AM的移动范围是保证无环约束的充分条件,有一些合适的的移动可能会被忽略，现在把移动范围扩展到全局，但是在最坏情况下，每此移动都需要检查可行性，也就是是否满足两个约束。
商图，就是把各个分区看作一个节点

这几种算法都存在一个问题，就是很容易陷在局部极小值出不来。

FM Moves

- A combination of:
  - AM
  - adapted Fiduccia-Mattheyses Algorithm（FM）
- 在一对分区A, B之间交换边界节点（假定A在B之前）
  - enabled nodes:
    - 属于A，且没有出边指向B之前的分区
    - 属于B，且没有入边来自A之后的分区
  - priority queue：gain & node identifier

AM的邻域搜索范围保证了无环
将候选移动加入优先级队列，保存了移动的增益和节点标识，按照增益从大到最小排序，如果增益相同，则利用随机数进行排序（比较器需要自己自己实现）
FM让算法可以爬出局部极小值
A，B是随机选择的，但是要保证至少其中有一个分区是活跃的，即有节点是enabled的
每次从队列中提取第一个移动，如果它可用，并且不会打破平衡约束，则提交该移动
每次移动都会改变当前可用的移动，所以队列中的移动不一定都是可用的，在取出后先要检查可用性。

FM Moves

- Moves：
  - If节点w从A移动到B，锁定w
    - 禁用B中所有满足$(\mathbf{w}, v) \in \mathbf{E}$的节点
    - 启用 **A中拥有指向w的出边&移动w后没有其他出边指向B之前的分区** 的节点
- Inner pass stops when the priority queue is depleted
- Iteration limit：$\frac{2n}{k}$
- Outter pass stops when no blocks are active
- 时间复杂度:$O(m + m_Q \frac{n}{k} \log \frac{n}{k})$，稀疏图$O(m + nlog \frac{n}{k})$

如果将节点w从A移动到B，那么就要禁用B中所有w为起点的边的终点，因为它们的移动会打破无环约束
注意移动的增益不需要重新计算，因为移动后节点w被blocked锁定，所有与w相连的v节点，在本轮传递里不在会被启用。
另一方面，要启用A中拥有到w的出边的节点（这时变成了边界节点），并且移动w后没有其他出边指向B之前的分区，启用节点：计算移动增益，并加入优先级队列
算法停止，即当所有节点都被禁用，或者被锁定。

移动的重新插入，导致难以估计迭代次数，可以人为设置上限2n/k
记录传递中取得的最佳结果

Experimental Evaluation

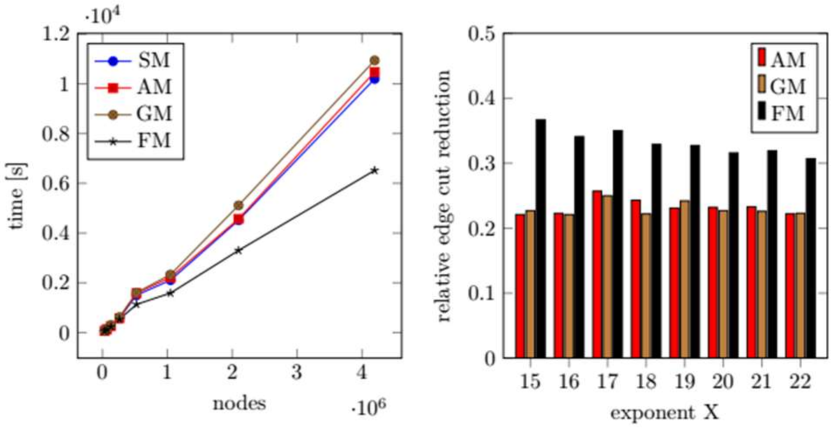
➢Comparison with Optimal Solutions

k	ϵ	SM	AM	GM	FM
2	20 %	3.41 %	3.41 %	3.41 %	0.26 %
	30 %	11.94 %	11.91 %	11.90 %	0.33 %
	40 %	14.71 %	14.78 %	14.58 %	1.29 %
	50 %	23.32 %	23.36 %	23.04 %	1.21 %
4	20 %	1.89 %	1.27 %	1.33 %	0.74 %
	30 %	4.03 %	3.22 %	3.25 %	0.67 %
	40 %	5.09 %	3.65 %	3.69 %	0.44 %
	50 %	6.50 %	4.04 %	4.19 %	0.31 %

每组参数下有25个接近实际应用的随机图
节点数在10-20范围
比最优解增加的代价
最优解是通过穷举法算出来的，穷举搜索算法运行时间在几秒和几天不等（取决于图大小）

Experimental Evaluation

➢Random Geometric Graphs



(a) execution time averaged over 100 passes **(b)** edge cut relative to Simple Moves

随机几何图
a.运行时间与节点数的关系
b.与SM相比，边割减少的相对值

多级算法 Multi-level Approach

➢KaFFPa多级算法：

- 粗化 graph coarsening
- 初始划分 initial partitioning
- 精化 partition refinement

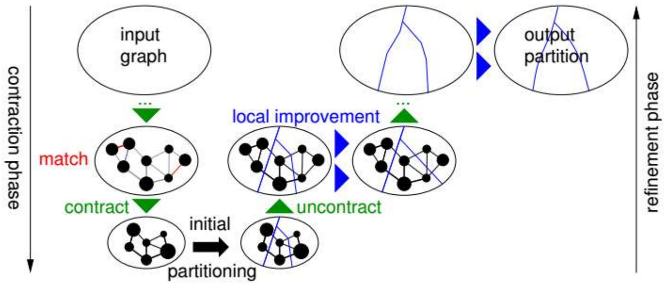


Fig. 1. Multilevel graph partitioning.

KaFFPa 本来是是无向图划分的一种算法，将其中的方法加以修改用到DAG中
在使用的时候，算法会把所有边都当做无向边

KaFFPa多级算法

➢修改后：

- 初始划分
- 粗化
- 精化

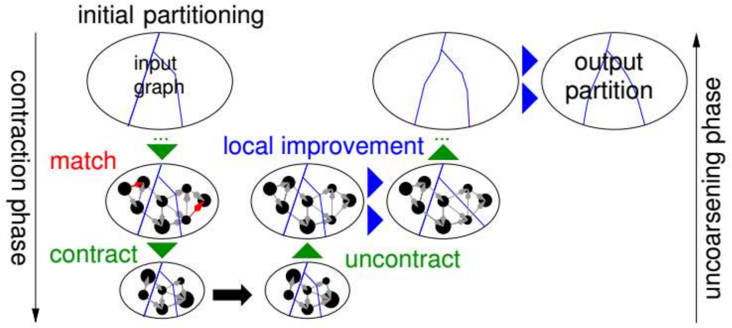


Fig. 2. The multi-level approach to graph partitioning.

因为先进行收缩匹配可能创建包含环的更粗糙的图，(收缩的时候把所有边都当作无向边来考虑)，因此可能无法在最粗图的级别上找到的解。在对图进行初始分区之后，我们继续对图进行粗化，直到它没有剩余的匹配边为止。

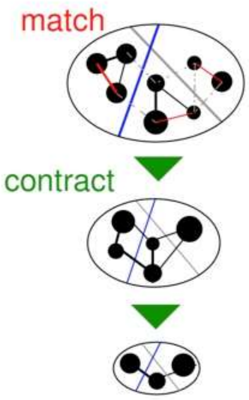
初始划分 Initial Partition

- 构造算法：基于拓扑排序的初始划分
- 局部搜索算法：启发式（SM，AM，GM，FM）

粗化 Coarsening

➢收缩 contracting

- 收缩一条边{u, v}，用新节点x代替节点u和v。
- 新节点权重 $c(x) = c(u) + c(v)$
- 新边权重 $\omega(\{x, w\}) = \omega(\{u, w\}) + \omega(\{v, w\})$



➢停止

- Stop when graph is “small enough”
- 当节点总数 < $\max(60k, n/(60k))$

初始划分后的割边不会进行收缩，不然初始划分就没有意义了。
粗化可以让整个图变小，有利于更快的划分。
这个限制是，paper里给的，但我们在用的时候肯定要改，只要保证节点数量少到足够我们用算法进行图划分就可以了。

粗化 Coarsening

➢评分函数：基于局部信息，衡量收缩该边的意义有多大

$$\text{rating function expansion}^{*2}(\{u,v\}) := \frac{\omega(\{u,v\})^2}{c(u)c(v)}$$

➢匹配算法：基于全局结构，最大化收缩边的总分

- GPA(Global Paths Algorithm)
  - integrating the greedy algorithm & Path Growing Algorithm.

比较简单的评分函数，可以就选边的权重，但是这样考虑会过于片面，可以考虑上节点。
收缩权重较大的边更好，因为减少了割的size，同样也避免生成拥有很多出边的分区
并且倾向于收缩权重更小的节点，为了让各级收缩后的节点的权重比较相近。

再对各边进行评分之后，并不是所有边都可以收缩。
直观的想法，肯定是先收缩权重大的边

匹配算法 Matching Algorithm

➢Greedy

- $O(m + sort(m))$

```
GRDY(G = (V, E), w: E → IR^{≥0})
M := ∅
while E ≠ ∅ do
    let e be the edge with biggest weight in E
    add e to M
    remove e and all edges adjacent to its endpoints from E
return M
```

Fig. 1. The greedy algorithm for approximate weighted matchings.

计算每条边的评分，时间正比于边的数量， Sort(m)是排序所花时间
按边的评分降序，依次收缩

匹配算法 Matching Algorithm

➢Path Growing Algorithm

- $O(m)$

```
PGA'(G = (V, E), w: E → ℝ^{≥0})
M := ∅
while E ≠ ∅ do
  P := ⟨⟩
  arbitrarily choose v ∈ V with deg(v) > 0
  while deg(v) > 0 do
    let e = (v, u) be the heaviest edge adjacent to v
    append e to P
    remove v and its adjacent edges from G
    v := u
  M := M ∪ MaxWeightMatching(P)
extend M to a maximal matching
return M
```

Fig. 2. The improved Path Growing Algorithm PGA′.

随机选择一个度大于零的节点
选择评分最高的边，加入P，现在再以边另一端的节点为中心，寻找分数最高的边，直到节点没有其他边，小循环停止
大循环是每次取M和P的最大权重匹配，这个maxWeightMatching也是一个函数，之后会讲。当没有边可以匹配是，大循环结束
最后M集合里的边就是匹配成功的边。

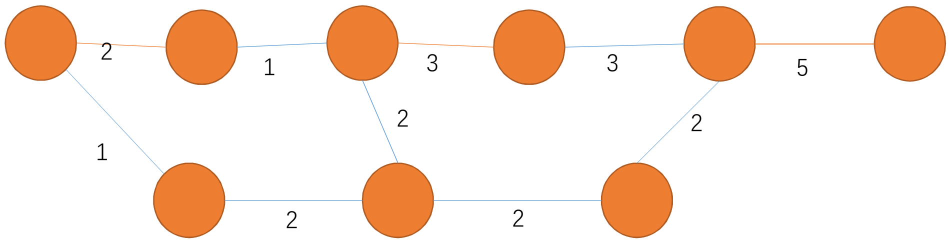
Path Growing Algorithm

```
MaxWeightMatching(P = ⟨e_1, ..., e_k⟩)
W[0] := 0;   W[1] := w(e_1)
M[0] := ∅;   M[1] := {e_1}
for i := 2 to k do
   if w(e_i) + W[i − 2] > W[i − 1] then
      W[i] := w(e_i) + W[i − 2]
      M[i] := M[i − 2] ∪ {e_i}
   else
      W[i] := W[i − 1]
      M[i] := M[i − 1]
return M[k]
```

Fig. 7. Obtaining a maximum weight matching for a path by dynamic programming.

刚才提到最大权重匹配，实质上是动态规划

Path Growing Algorithm



按照算法，匹配的边是黄色的

匹配算法 Matching Algorithm

➢Global Path Algorithm

- $O(Sort(m)+m) \sim O(mlogn)$

```
GPA(G = (V, E), w: E → ℝ≥0)
M := ∅
E' := ∅
for each edge e ∈ E in descending order of their weight do
   if e is applicable then add e to E'
for each path or cycle P in E' do
   M' := MaxWeightMatching(P)
   M := M ∪ M'
return M
```

Fig. 3. The Global Paths Algorithm GPA.

GPA算法结合了Greedy和PGA。
然后，通过按weight（评分）降序依次添加适用边来扩展集合。

如果边连接不同路径的两个端点 或者 奇数长度路径的两个端点，则称为“适用”。
如果边是路径形成奇数长度的环，或者它与连接现有路径的内部两个节点，则边“不适用”。

扫描所有边缘后，将为每个路径和循环计算最大权重匹配。

精化 Refinement

➢Local improvement:

- 解除粗化阶段的匹配
- 启发式算法

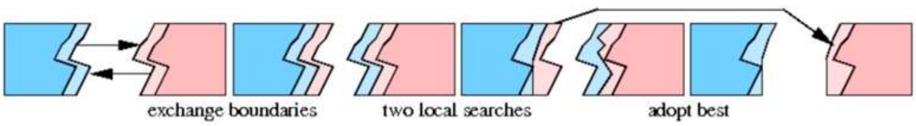


Figure 2: Refinement between two blocks using boundary exchange.

解除粗化阶段的收缩的边，把合并的节点和边还原，再进行细致的优化
可能有人会问，刚刚把那么多边收缩了，现在有把他们都解除，有什么意义呢？
其实精化不是一次性完成的，先解除一部分收缩，然后应用启发式进行优化，然后再解除一部分，最后完成整个精化过程。
由于收缩的定义，节点和边的权重会相应改变，所以不同级别的图在相同划分下，目标函数和平衡是一致的。

多级算法的思路有点像显微镜，先粗调，再微调

进化算法 Evolutionary algorithm

- 交叉重组 Cross Recombine
- 突变 Mutation
- 适应函数 Fitness Function
- 杂集 Miscellanea

进化算法，是把每种划分看作个体，多个个体构成一个种群，种群会一代一代进化，每一代都比之前一代的适应性更强。
当进化到符合我们设定的一个标准时，算法停止。

进化算法 Evolutionary algorithm

Algorithm 1 A classic general steady-state evolutionary algorithm.

procedure *steady-state-EA*
  create initial population P
  while stopping criterion not fulfilled
    select parents p_1, p_2 from P
    combine p_1 with p_2 to create offspring o
    mutate offspring o
    evict individual in population using o
  return the fittest individual that occurred

这里设定的是，每代只产生一个后代，称为steady-state

种群的初始化：是利用多级算法用不同随机种子创建一定数量的个体

从种群中选择两个个体做为父代，产生一个后代（创造后代的过程中可能存在突变基因）

把后代放入种群中，再驱逐出一个个体（驱逐策略）

交叉重组 Cross Recombine

- ➢P1&P2➔O
- ➢P1: 2-way tournament selection algorithm
- ➢P2: KaFFPa (multi-level algorithm)
  - $k' \in [\frac{k}{4}, 4k]$
  - $\epsilon' \in [\frac{k}{4}, 4\epsilon]$
  - Balance constraint: $\max c(V_i) \leq (1+\epsilon')\frac{c(V)}{k'}$

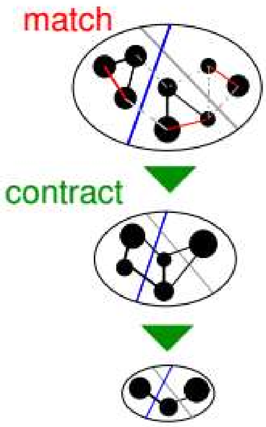
重组分为三种:
普通重组：两个父代都是从种群中选择的
交叉重组/转导：一个父代是从种群中选的，另一个是算法在这个一选择的个体的基础上生成的
另一种，Natural-cuts，自然分割，它的个体生成算法就是独立的，不依赖种群中的个体

我们这个主要讨论交叉重组，其中一个父代使用锦标赛算法从种群中选择的。2-way，表示从种群中随机选择两个体提进行锦标赛，其实这里就相当于直接比较选出最佳。

另一个父代，我们用前面提到的多级算法生成，在设置参数的时候，参考第一个父代，第一个个体的分区数量为k，不平衡参数为epsilon，我们将在这个范围随机选择第二个父代的参数，在用多级算法算出一个满足约束的划分结果。

交叉重组 Cross Recombine

- P1&P2➔O
- KaFFPa
  - 粗化
  - 初始划分
  - 精化



现在有了两个父代，怎么让他们身边产生子代呢？用多级算法。（初始划分，粗化，精化）
先进行粗化，两个父代的割边都不会收缩，剩下的边按照多级算法进行收缩。
将两个父代中较好的划分作为初始化分，再进行精化。
如果两个父代一样好，随机选一个。

突变 Mutation

- 两种：都是用来自当前种群的随机个体P1
  - 使用多级算法创造k-partition P2
  - P2=P1

对之前的后代进行突变
主要思想是用不同的种子迭代粗化和精化得到随机的划分
第一种，目的是引入新的划分
第二种，保证突变后的质量不会更差

最后两种突变选择较好的一个。

适应函数 Fitness Function

- Cut
- Time

之前经常需要对两个划分进行比较。
这里我们评价一个个体的适应度，不仅仅是考虑它的目标函数，还考虑执行时间，分区的执行时间最好是比较接近。

杂集 Miscellanea

Algorithm 3 All PEs perform basically the same operations using different random seeds.

procedure *locallyEvolve*
  estimate population size S
  while time left
    if elapsed time $< t_{\text{total}}/f$ **then** create individual and insert into local population
    else
      flip coin c with corresponding probabilities
      if c shows head **then**
        perform a mutation operation
      else
        perform a combine operation
      insert offspring into population if possible
    communicate according to communication protocol

为了加快得到最终结果
可以同时存多个PE（processing element），一个PE拥有一个种群，各个种群都是用多级算法生成的。
每个PE并行执行，并且两个PE之间会传播种群内部的最优个体，也就是最佳划分。