半径步进的并行最短路径算法

Parallel Shortest Paths Using Radius Stepping

汇报人：乔志鹏

Algorithm 1: The RADIUS-STEPPING Algorithm.

Input: A graph $G = (V, E, w)$, vertex radii $r(\cdot)$, and a source vertex s.

Output: The graph distances $\delta(\cdot)$ from s.

```
1  δ(·) ← +∞, δ(s) ← 0
2  foreach v ∈ N(s) do δ(v) ← w(s, v)
3  S_0 ← {s}, i ← 1
4  while |S_{i−1}| < |V| do
5      d_i ← min_{v∈V\S_{i−1}} {δ(v) + r(v)}
6      repeat
7          foreach u ∈ V\S_{i−1} s.t. δ(u) ⩽ d_i do
8              foreach v ∈ N(u)\S_{i−1} do
9                  δ(v) ← min{δ(v), δ(u) + w(u, v)}
10     until no δ(v) ⩽ d_i was updated
11     S_i = {v | δ(v) ⩽ d_i}
12     i = i + 1
13 return δ(·)
```

循环次数

（1）外循环次数（while）

环内结点数量 P

$\Delta d_i = d_i - d_{i-1}$

（2）内循环次数（repeat）

环内路径最大边数 K

定义一：$\hat{d}(u, v)$ （Hop Distance）（跳跃步数）：

对图上任意两点u，v，$\hat{d}(u, v)$ 表示 u 到 v 的最短路径中的最少边数。

定义二：$\bar{r}_k(u)$ （k-Radius）：

对图上一点 u，$\bar{r}_k(u) = \min_{v \in V, \hat{d}(u,v) > k} d(u, v)$

即和 u 的跳跃步数大于 k 的点到k的最短路径和

也就是说，
对任意一点u，若 $\hat{d}(u, v)$ > k，则 d（u，v）> $\bar{r}_k(u)$ 。

对这个结论取逆反命题可以得到，
对任意一点u，若 d（u，v）< $\bar{r}_k(u)$ ， $\hat{d}(u, v)$ <= k。

因此可以看到，我们可以通过限制 r（v）<= $\bar{r}_k(u)$ ，
来限制环内任意两点的跳跃步数 K <= k+1 ，
从而限制内循环（repeat）的次数

因此加上最后多循环一次确定无节点被更新，
总共需要循环至多 k+2 = O（k）次。

定义三：$r_\rho(v)$ （ρ -Nearest Distance）

对图上一点 v，$r_\rho(v)$ 为与 v 距离第 ρ 近的点离 v 的距离。

因此，对任意一点 u ρ 若 d（u，v）>= $r_\rho(v)$ ，则比 u 更接近 v 的点有至少 ρ 个。

同样的，为得 P 在一定范围内尽可能大，我们设 r（v）>= $r_\rho(v)$ 。

首先我们发现r（v）在 $[r_\rho(v), r_{\rho+1}(v))$ 等价的，不影响结果，因此我们可以直接设置r（v）= $r_\rho(v)$ 。

其次，与 K 不同的是，我们能确保在环内有至少 ρ 个结点吗?

$r(v) < \bar{r}_k(u)$ 是对于（v，r）圆内每条路径共有的性质，而 $r(v) >= r_\rho(v)$ 是对圆内所有节点的和数量的性质。

又因为该圆并非完全属于（i-1，i）环，因此我们并不能保证某个环内的结点数大于等于 ρ 。

然而如果我们反向思考，我们不能保证结点数足够大，但我们或许能保证它不能一直小。

因此，对于连续若干个环，我们依然可以找到一些可以保证 P 的数量的性质。

我们可以通过考虑最坏情况来保证结点数量。
我们假设最多步进 t 次可以包含ρ 个结点，
即每连续 t 个环内（（i，i+t）环）必然包含ρ 个结点。
那么若（i，j ）环已包含 ρ 个结点，（i<j<i+t)
则假设成立，
若没有，因为 $r(v) >= r_\rho(v)$ ，则 $d(s,v_j) - d_i < r_\rho(v_j) \le r(v_j)$ ，
否则，（i，j）环包含了（v，r）圆必含有ρ 个结点，
所以

$$\begin{aligned} d_j - d_i &= \delta(v_j) + r(v_j) - d_i \\ &\geqslant d(s, v_j) + \left(d(s, v_j) - d_i\right) - d_i \\ &= 2(d(s, v_j) - d_i) \geqslant 2(d_{j-1} - d_i). \end{aligned}$$

因此，若结点数一直很小，则d的增长速率将以指数级增长，

那还得了！

于是我们算出在最坏情况下，$d_j - d_i > 2^{j-i-1}$ ，
而这当然不能永远扩大下去，
我们设图中的最大权重为L，即 $L = \max_e w(e)$ ，
则在距离 $d = \rho L$ 中必然包含了 ρ 个结点，
因此我们有在最坏情况下 $\rho L \geq d_{i+t} - d_i > 2^{t-1}$ ，
算出 $t = \log_2(\rho L) + 1$ ，
平摊一下，我们可以保证每个环包含至少 $\frac{\rho}{t}$ 个结点，
因此外循环（while）被限制至多循环 $\frac{nt}{\rho} = \frac{n}{\rho}(1 + \log_2(\rho L)) = O(\frac{n}{\rho} \lg \rho L)$ 次。

Algorithm 2: The Shortest-path Algorithm.

Input: A graph $G = (V, E, w)$, vertex radii $r(\cdot)$ and a source node s.

Output: The graph distances $\delta(\cdot)$ from s.

```
1  δ(·) ← +∞, δ(s) ← 0
2  i ← 1
3  Q = {w(s,u) | u ∈ N(s)}
4  R = {w(s,u) + r(u) | u ∈ N(s)}
5  while |Q| > 0 do
6      d_i ← R.EXTRACT-MIN()
7      {A_i, Q} = Q.SPLIT(d_i)
8      foreach u ∈ A_i do R.REMOVE(u)
9      repeat
10         foreach u ∈ A_i, v ∈ N(u) do
11             if δ(v) > δ(u) + w(u,v) then
12                 if δ(v) > d_i and δ(u) + w(u,v) ⩽ d_i then
13                     R.REMOVE(v)
14                     Q.REMOVE(v)
15                     A_i.INSERT(v)
16                 δ(v) = δ(u) + w(u,v)
17                 if δ(v) > d_i then
18                     Q.DECREASE-KEY(v, δ(v))
19                     R.DECREASE-KEY(v, δ(v) + r(v))
20     until no δ(v) that v ∈ A_i is updated
21     i = i + 1
22 return δ(·)
```

现在我们得出了内外循环的次数，只要得到内部松弛程序的复杂度就可以分析这整个算法的复杂度了。

我们用两个集合来分别储存w（s，u）与w（s，u）+r（u）两组元素，

于是内部程序被展开如图。

若我们选择使用二叉树的数据结构，这时我们发现这些操作都可以在O（lgn）内完成。

将三者相乘，最终我们得到整个算法的时间复杂度为：

$$T = O\left(\frac{kn}{\rho} \lg\lg \rho L\right)$$

等等，各位，我们今天的讨论还没有结束
我们还存在着最后一个问题，也是一个致命的问题，是我们整个讨论的根基

$r_\rho(v) = r(v) < \bar{r}_k(u)$ 一定成立吗？

定义四：$((k,\rho)\text{-BALL AND } (k,\rho)\text{-GRAPH})$.
若 $r_\rho(v) \leqslant \bar{r}_k(v)$ 则对于 v存在一(k,ρ) 球，
若该图所有结点都存在(k,ρ) 球，
则该图是一个(k,ρ) 图。

由此我们可以看出我们之前讨论的一切只对(k,ρ) 图成立。
那是否所有的图都是(k,ρ) 图呢？

显然并不是。。。
但或许又可以是!

事实上，我们只需要对原图进行一些简单的预处理就可以使它成为 (k,ρ) 图。

对任意不相邻两点u，v，构造一条边（u，v），
且让w（u，v）=d（u，v）
于是δ（u）与δ（v） 不变但 $\hat{d}(u,v)$ 减少
且 $\hat{d}$（s，u）与 $\hat{d}$（s，v）可能减少

以 k=1 为例，
为了确保图为(k,ρ)图，
我们需要将每个结点 v 的最近的ρ 个结点与 v 构造一条边，

于是我们对 n 个结点并行地执行dijkstra算法 次即可，
复杂度为T=O（ρ lgn）

当 k>1 时同理，但为了找到最优解需要用到动态规划或贪心算法
T 仍为 O（ρ lgn）
故T（总）=$O(\frac{kn}{\rho}\lg n\lg\rho L)$ $O(\rho lgn)$

参考资料：

1. “Parallel Shortest Paths Using Radius Stepping”. Blelloch, Guy E.; Gu, Yan; Sun, Yihan; Tangwongsan, Kanat (2016). Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures - SPAA ’ 16. New York, New York, USA: ACM Press: 443–454.
2.无了（全网唯一资料来源，wiki是这份文献的综述）

感谢倾听

感谢马老师的指导

汇报人：乔志鹏