基于GraphLite的SimRank算法实现

组长：余学辉

组员：耿洪娜

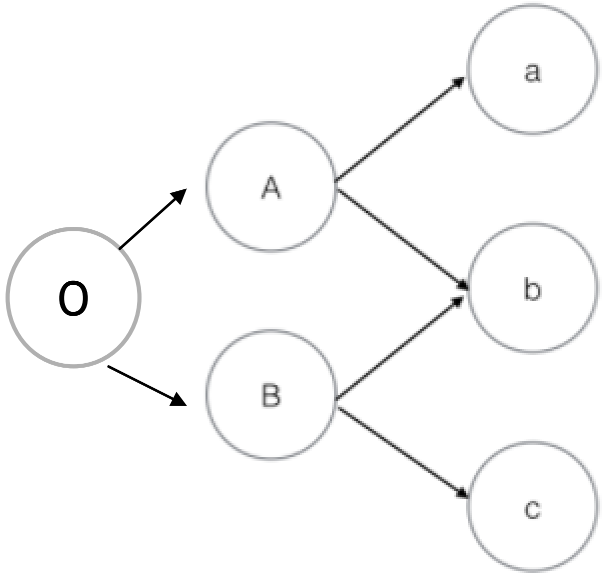
- 算法调研
- 算法实现
- 性能测试

算法调研 - naïve simrank

$$R_0(a,b) = \begin{cases} 0 & if\ a \neq b \\ 1 & if\ a = b \end{cases}$$

$$R_{k+1}(a,b) = \begin{cases} \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) & if\ a \neq b \\ 1 & if\ a = b \end{cases}$$

被同一个顶点引用地两个顶点是相似的；
被相似地顶点引用的两个顶点是相似的；



算法调研 - naïve simrank

$$\begin{cases} S^{(0)} = (1-c) \cdot I_n \\ S^{(k+1)} = c \cdot Q^T \cdot S^{(k)} \cdot Q + (1-c) \cdot I_n \end{cases}$$

$$\|S^{(k)} - S\|_{max} \leq c^{k+1} \quad (\vee k = 0, 1, 2 \ldots)$$

算法调研- 平方缓存法 simrank

$$
\begin{cases}
S_{\langle 2 \rangle}^{(0)} = (1-c) \cdot I_n \\
S_{\langle 2 \rangle}^{(k+1)} = S_{\langle 2 \rangle}^{(k)} + c^{2^k} \cdot (Q^{2^k})^T \cdot S_{\langle 2 \rangle}^{(k)} \cdot Q^{2^k}
\end{cases}
$$

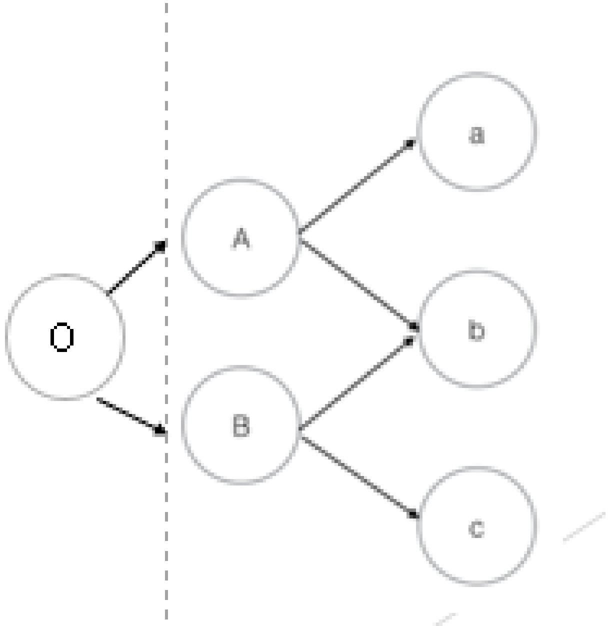
$$
\| S_{\langle 2 \rangle}^{(k)} - S \|_{max} \le c^{2^k} \quad (\vee k = 0, 1, 2 \ldots)
$$

还有一些基于对Q/S矩阵进行降维分解来简化计算的方法

算法调研 - Monte Carlo

$$R_0(a,b) = \begin{cases} 0 & if\ a \neq b \\ 1 & if\ a = b \end{cases}$$

$$R_{k+1}(a,b) = \begin{cases} \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) & if\ a \neq b \\ 1 & if\ a = b \end{cases}$$



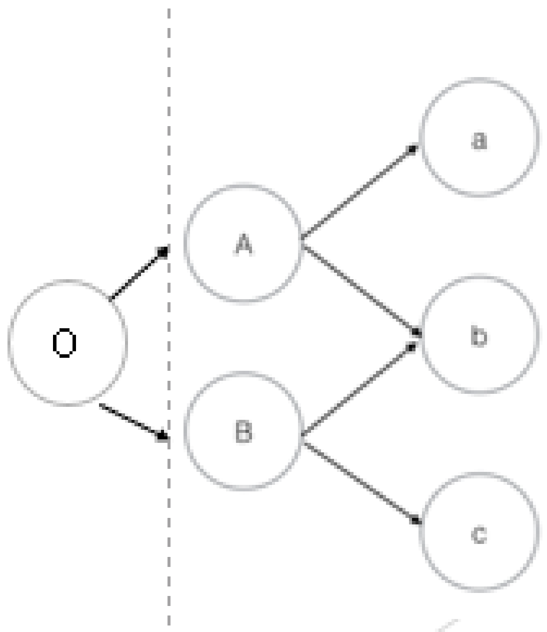
R(a, b) = k1*R(0, 0) + k2*R(A, A)

K1 = C^2 * p1;
K2 = C^1 * p2;

算法调研- Monte Carlo

$$s(a,b) = \sum_{t(a,b)\to(x,x)} P[t]C^{l(t)}$$

$$s^{k+1}(a,b) = \sum_{t(a,b)\to(x,x);l(t)\leq k+1} P[t]C^{l(t)}$$



1. 两个walker a, b从顶点a,b出发，随机游走；
2. t(a, b)表示地是a, b两个walker在t时刻到达的顶点；
3. P[t]表示walker a, b游走到t(a,b)的整个出现地概率；
4. C是衰减因子，l(t)表示游走地路径长度。

算法实现

Monte Carlo AL

```
input: K Monte Carlo loop times, L max length of random walking path, G the test Graph.
output: S similar matrix.

for k in 1:K,
    1. all vertex generate a walker.
    2. walker random walking in Graph follow reversed path.
    3. if two walker meet in same vertex and they have not meet before, calculate they similar, updat
e S.
    4. if a walker have no path to choose, then remove it out.
    5. when random walking path length >= L,  all walker disapear.
```

Get/Set similar matrix

```
for(uint64_t  i=0; i+1<walkers.size(); i++) {
    for (uint64_t j = i + 1; j < walkers.size(); j++) {
        auto id_i = walkers.at(i).source_id;
        auto id_j = walkers.at(j).source_id;
        if(sim[id_i][id_j] <= EPS){
            sim[id_i][id_j] = C_K;
        }
    }
}
```

Get

Set

全局访问matrix的三种方法

- Static
- Aggregator
- Nest in Message

全局访问matrix的三种方法:

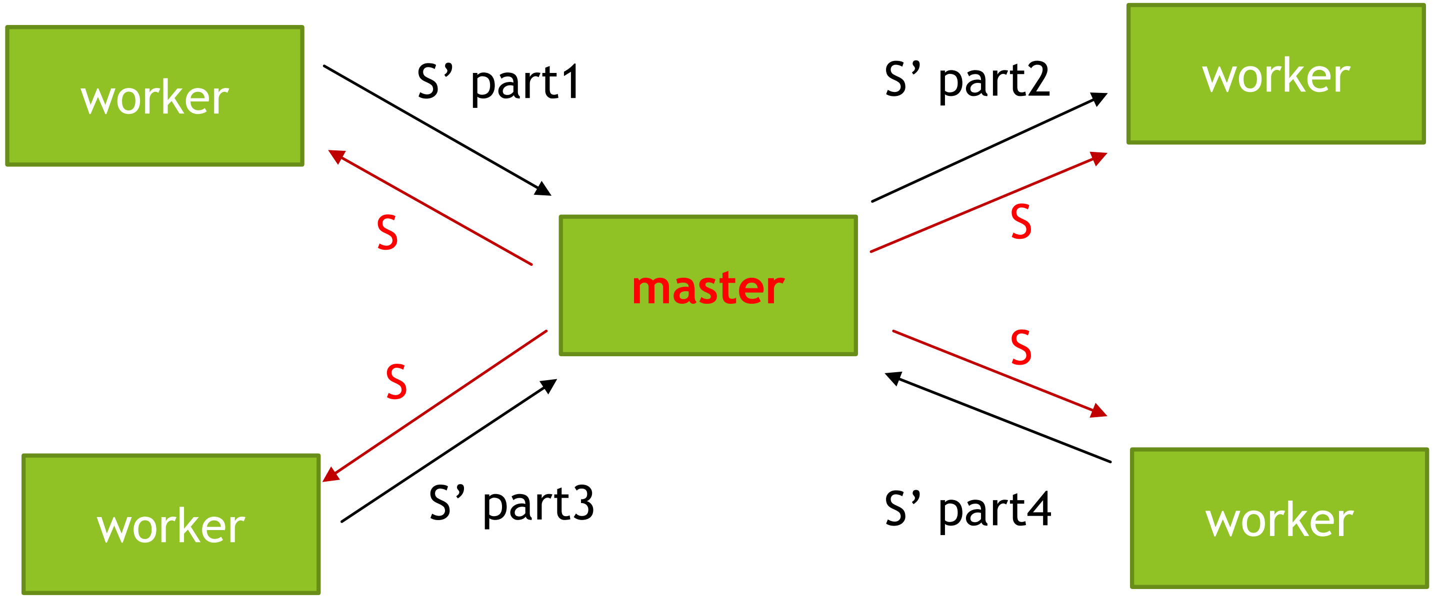
- static
  - static 变量对于单个worker里的定点是共享的，但不同worker间的顶点是不共享的。

全局访问matrix的三种方法：

- static
  - static 变量对于单个worker里的定点是共享的，但不同worker间的顶点是不共享的。

全局访问matrix的三种方法

- Aggregator



每个超步开始，给每个worker发送S矩阵；
每个超步结束，每个worker把S发回master;
通讯传输数据大小为|workers| * sizeof(S).
可能卡在数据通信上。

全局访问matrix的三种方法

- Nest in Message

只需要S矩阵的第id_i行或id_j行

```
for(uint64_t  i=0; i+1<walkers.size(); i++) {
    for (uint64_t j = i + 1; j < walkers.size(); j++) {
        auto id_i = walkers.at(i).source_id;
        auto id_j = walkers.at(j).source_id;
        if(sim[id_i][id_j] <= EPS){
            sim[id_i][id_j] = C_K;
        }
    }
}
```

全局访问matrix的三种方法

- Nest in Message
  - Carry row of S in Message
  - Use sparse matrix
  - Store half matrix

```
struct Msg{
    uint64_t source_id;
}
```

```
struct Msg{
    uint64_t source_id;
    map<uint64_t, double> sim;
}
```

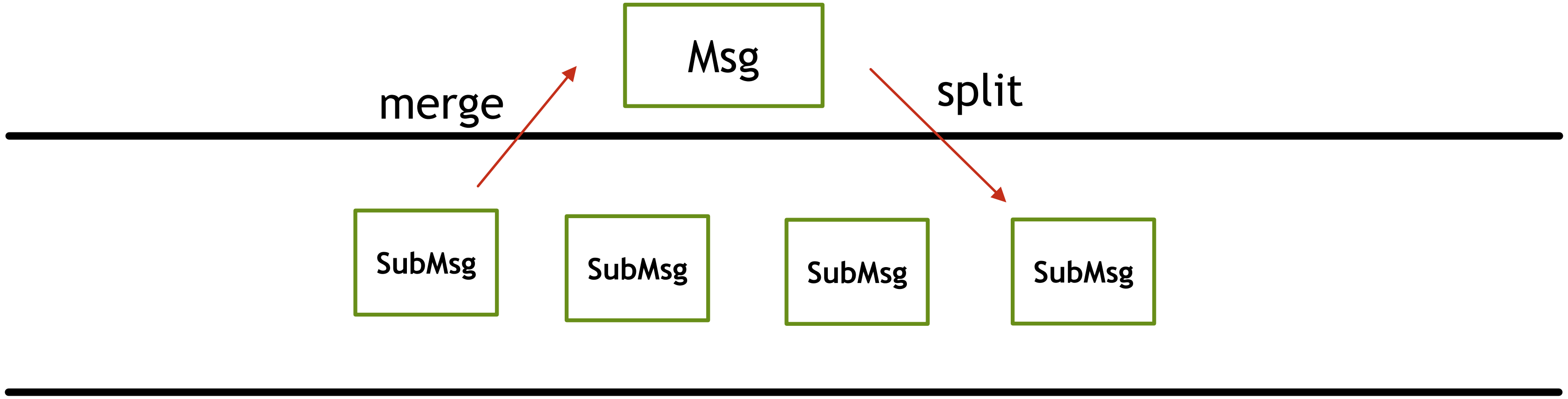
整个图中存在的walker（Message）的数量不会超过图的顶点数；
每个walker携带稀疏矩阵地一行；
整个图中传递地数据量不会超过sizeof(S);

全局访问matrix的三种方法

- Merge and split Message

```
struct Msg{
    uint64_t source_id;
    map<uint64_t, double> sim;
}
```

```
struct SubMsg{
    uint64_t i; // source_id
    uint64_t j; // simj = sim[j]
    double simj;
}
```



全局访问matrix的三种方法

- Collect result

```
struct Msg{
    uint64_t source_id;
    map<uint64_t, double> sim;
}
```

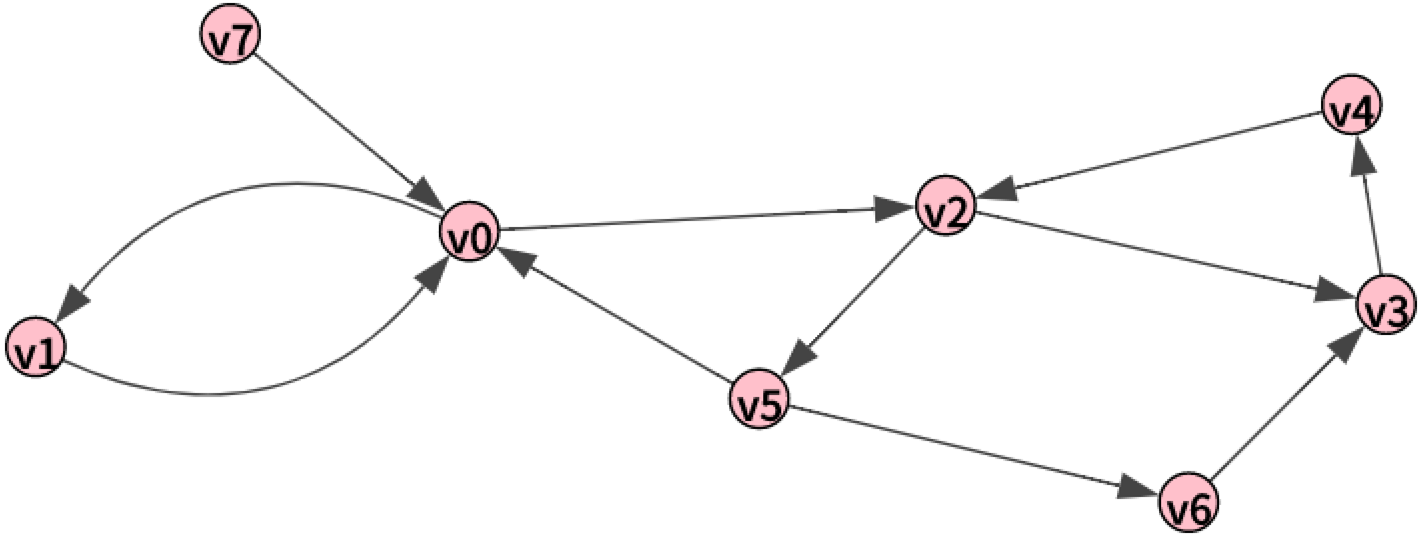
⇨

```
struct Msg{
    uint64_t source_id;
    map<uint64_t, double> sim;
    MSGFLAG flag; // DEAD or ALIVE
}
```

1. 每个vertex value中存储S的一行S[vid]，最后写到结果文件里去；
2. 一个walker随机游走时flag是ALIVE;
3. 当一个walker(Message)消失或者达到最大游走长度时，将flag设置为DEAD，将它发回到vid=source_id的顶点；
4. 每个vertex计算收到ALIVE的Msg对应source_id的相似性，使用DEAD的MSg更新自身地value值。

性能测试：testgraph

$$error = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |S(i,j) - S^{true}(i,j)|}{max(|S(i,j)|_0, |S^{true}|_0)}$$

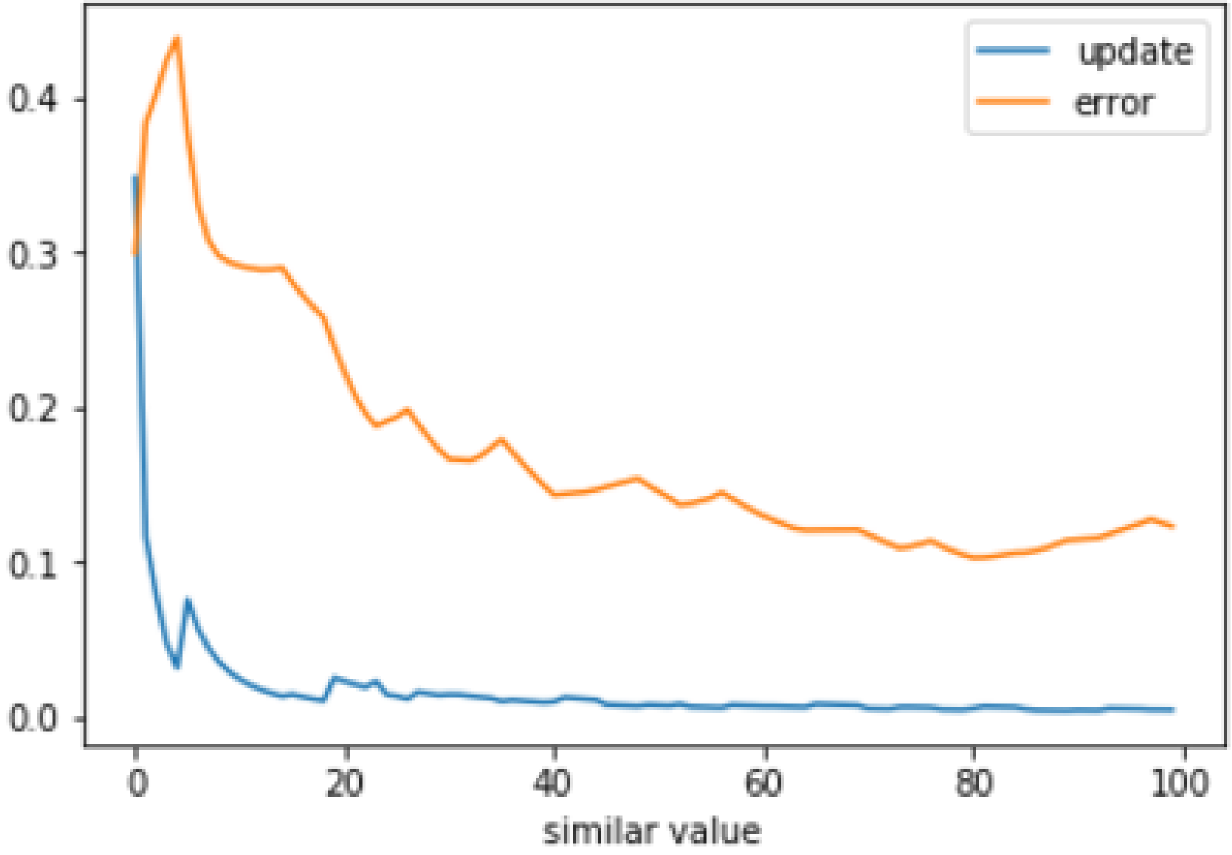


性能测试：testgraph

同步sim

异步sim

Monte simrank hLite)

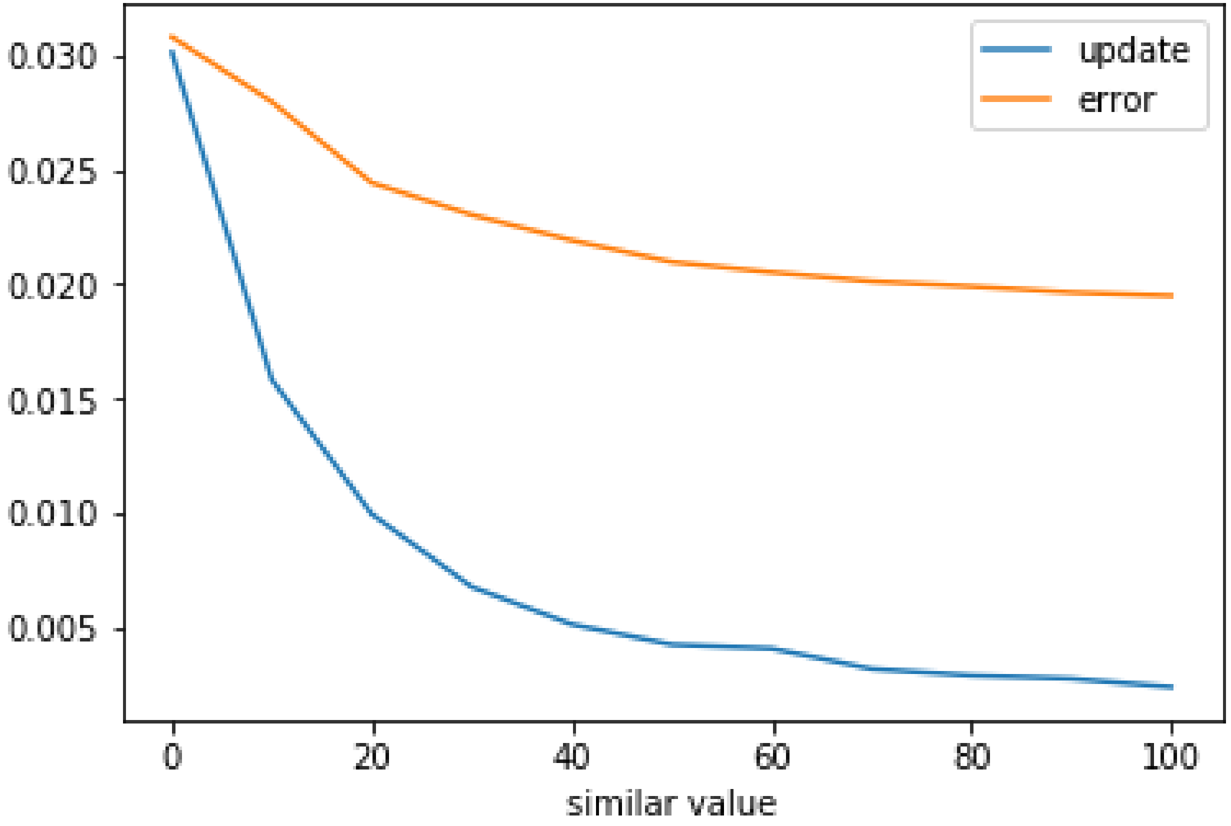


性能测试：facebookcombined

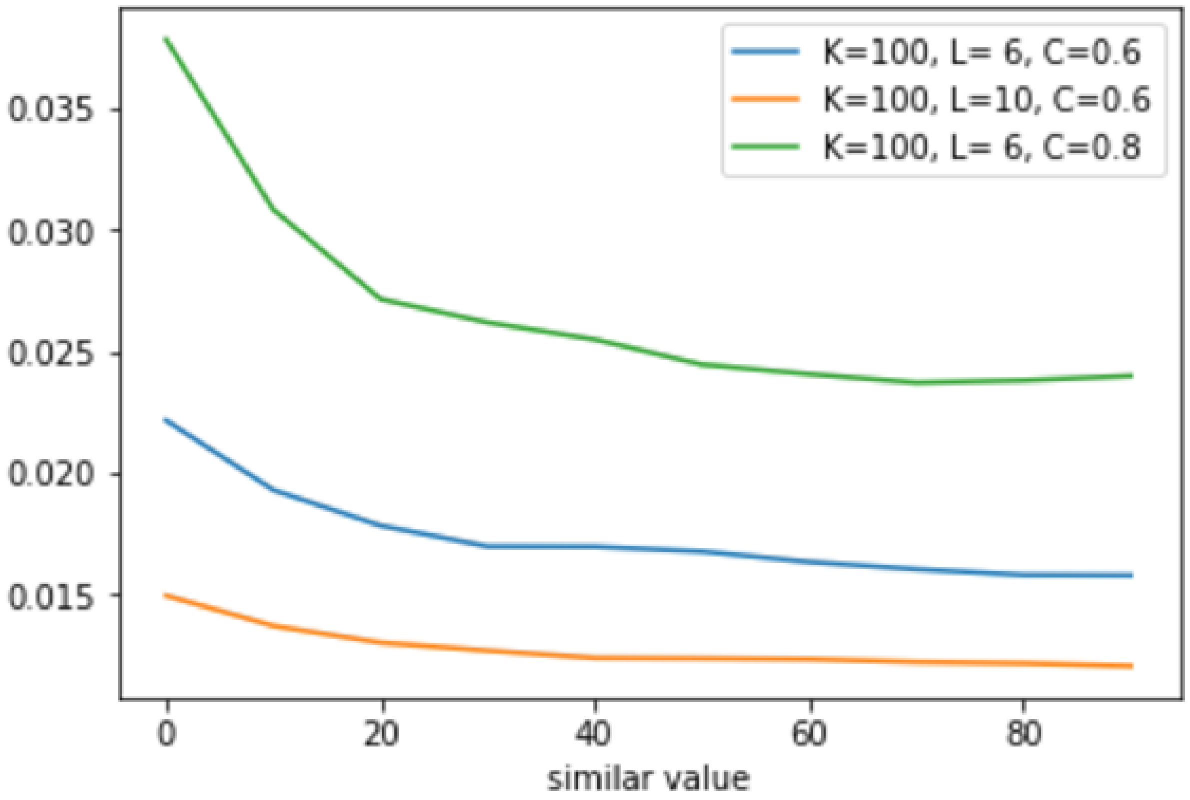
同步

异步

Mont
simr
Lite)



性能测试：facebookcombined



Search deeper;
Decay smaller;