1 Introduction

Algorithm 1 Heuristics improved (graph, k)

```
initialize SeedSet, h
for each $n \in graph$ do
    calculate $outdegree[n]$
end for
for each $n \in graph$ do
    $h[n] = 0$
    for each $neighbor \in N(n)$ do
        $h[n] += weight * outdegree[neighbor]$
    end for
end for
while $|Seedset| \neq k$ do
    $seed \leftarrow n$ with maximum $h(n)$
    SeedSet.add($seed$)
    for each $neighbor \in N(seed)$ do
        $h(neighbor) = (1 - weight) * (h[neighbor] - |intersection\ nodes|)$
    end for
end while
return SeedSet
```

Algorithm 2 CELF improved (graph, k, NodeSet)

```
1: initialize nodeHeap, preSpread, SeedSet
2: for each $n \in NodeSet$ do
3:     times = 100
4:     [low, high] = the 95% confidence interval of $ise(100times)$
5:     nodeHeap.add(node with high)
6: end for
7: for $i = 1; i < k; i++$ do
8:     $m = nodeHeap.top$
9:     while $m \notin calculate\ in\ this\ loop$ or m.times $\neq$ 10000 do
10:        if $m \in calculate\ in\ this\ loop$ then
11:            times = 10000
12:            spread = $ise(10000times) - preSpread$
13:            nodeHeap.(node with spread)
14:        else
15:            times = 100
16:            [low, high] = the 95% confidence interval of $ise(100times)$
17:            nodeHeap.add(node with high)
18:        end if
19:    end while
20:    seed = nodeHeap.pop()
21:    preSpread = ise(seed) + preSpread
22:    SeedSet.add(seed)
23: end for
24: return SeedSet
```