A Self-Stabilizing Algorithm for the K-Clustering Problem on Weighted Graphs

E. Caron\textsuperscript{1}, A. K. Datta\textsuperscript{2}, B. Depardon\textsuperscript{1} et L. L. Larmore\textsuperscript{2}

\textsuperscript{1} Université de Lyon. LIP.
UMR CNRS - ENS Lyon - INRIA - UCB Lyon 5668
\textsuperscript{2} University of Nevada, Las Vegas, USA

May 14\textsuperscript{th}, 2009
Introduction

Clustering
- Distributed algorithm
- Based on ID comparison
- Nodes at distance $\leq k$ of an elected node in cluster
- Fault-tolerant $\Rightarrow$ Self-stabilization

Self-Stabilization

Goal
- Distributed application
- Communications improvement
- Middleware deployment
- Automatic platform discovery
Outline

1. \textit{k}-clustering
2. Weighted-Clustering
3. Example
4. Performances
Outline

1. \textit{k}-clustering
2. Weighted-Clustering
3. Example
4. Performances
**k-clustering**

**Platform**
- Weighted graph $G = (V, E)$
- $\forall x, y \in V$, $w(x, y)$: shortest path weight between $x$ and $y$
  $\rightarrow$ weight: non negative integer
- $\text{radius}(G) = \min_{x \in V} \max_{y \in V} \{w(x, y)\}$

**Clustering**
- $C$ $k$-cluster: $C$ connected subgraph of $G$, $\text{radius}(C) \leq k$
- $x \in C$ is a clusterhead if $\forall y \in C$, there exists a path such that $w(x, y) \leq k$
- $k$-clustering of $G$: clustering of $V$ into disjoint $k$-clusters

$\implies$ Minimal $k$-clustering: $\mathcal{NP}$-hard problem
**Introduction**

- Clustering
- Weighted Clustering
- Example
- Performances
- Conclusion

---

**k-clustering**

Find a $k$-clustering for $k = 20$
Clustering definition

For each node $P$, we define the following values:

$$\mathcal{N}_P = \{\text{Set of neighbors of } P\}$$

$$MinId(P, d) = \min \{Q.id : w(P, Q) \leq d\}$$

$$MaxMinId(P, d) = \max \{MinId(Q, k) : w(P, Q) \leq d\}$$

$$Clusterhead\_Set = \{P : MaxMinId(P, k) = P.id\}$$

$$Dist(P) = \min \{w(P, Q) : Q \in Clusterhead\_Set\}$$

$$Parent(P) = \begin{cases} 
P.id 
& \text{if } P \in Clusterhead\_Set \\
\min \{Q.id : (Q \in \mathcal{N}_P) \land (Dist(Q) + w(P, Q) = Dist(P))\} 
& \text{otherwise}
\end{cases}$$

$$Clusterhead(P) = \begin{cases} 
P.id 
& \text{if } P \in Clusterhead\_Set \\
Clusterhead(Parent(P)) 
& \text{otherwise}
\end{cases}$$
Outline

1. k-clustering
2. Weighted-Clustering
3. Example
4. Performances
Algorithm *Weighted-Clustering* structure

**Description**

- 11 variables
- 26 functions
- 15 actions

**4 Phases**

**Leader election**  SSLE (Self-Stabilizing Leader Election), BFS tree, rooted at the lowest ID process

**Interval**  2 phases: MinId and MaxMinId

**Clustering**  k-clusters & trees creation

**Model**

- Shared memory: read/write own variables, read neighbors’
Algorithm Weighted-Clustering structure

Description
- 11 variables
- 26 functions
- 15 actions

4 Phases
- **Leader election**: SSLE (Self-Stabilizing Leader Election), BFS tree, rooted at the lowest ID process
  - **Interval**: 2 phases: MinId and MaxMinId
- **Clustering**: $k$-clusters & trees creation

Model
- Shared memory: read/write own variables, read neighbors’
### Algorithm Weighted-Clustering structure

#### Description
- 11 variables
- 26 functions
- 15 actions

+ SSLE

Memory space $O(\log n + \log k)$

#### 4 Phases

- **Leader election**: SSLE (Self-Stabilizing Leader Election), BFS tree, rooted at the lowest ID process
- **Interval**: 2 phases: MinId and MaxMinId
- **Clustering**: $k$-clusters & trees creation

#### Model
- Shared memory: read/write own variables, read neighbors’
## Algorithm Weighted-Clustering structure

### Description
- 11 variables + SSLE
- 26 functions
- 15 actions
- Memory space $O(\log n + \log k)$

### 4 Phases

#### Leader election
- SSLE (Self-Stabilizing Leader Election), BFS tree, rooted at the lowest ID process

#### Interval
- 2 phases: MinId and MaxMinId

#### Clustering
- $k$-clusters & trees creation

### Model
- Shared memory: read/write own variables, read neighbors’
Find a $k$-clustering for $k = 30$
0 ≤ d < 6 ≤ k + 1 = 31
Principle

\[ 6 \leq d < 12 \leq k + 1 = 31 \]
Principle

\[ 6 \leq d < 18 \leq k + 1 = 31 \]
Principle

\[ 6 \leq d < 20 \leq k + 1 = 31 \]
Principle

\[ 6 \leq d < 28 \leq k + 1 = 31 \]
Principle

\[ 28 \leq d < 31 \leq k + 1 = 31 \]
Outline

1. $k$-clustering
2. Weighted-Clustering
3. Example
4. Performances
Initial graph

Find a $k$-clustering for $k = 30$
Initial graph: errors

- Minimum research level: $minlevel$
- Minimum ID found so far: $minid$
- Maximum research level: $minhilevel$
Initial graph: errors

The initial graph contains nodes labeled A, B, C, D, E, with weights and connections as follows:

- **A** connected to **D** with weight 35:E:25
- **E** connected to **D** with weight 10:B:6
- **D** connected to **L** with weight 5:D:100
- **L** connected to **B** with weight 0:A:70
- **B** connected to **D** with weight 20:D:12
Initial graph: errors
MinId phase

\[ A_{hilevel} \Rightarrow \text{updates} \ minhilevel \]
\[ A_{minId} \Rightarrow \text{updates} \ minlevel, \ minid \text{ and } minhilevel \]
MinId phase
MinId phase
MinId phase

Graph showing connections and labels:
- A
  - 0:A:31
  - 25
- D
  - 0:D:22
  - 7
- L
  - 15:B:21
  - 15
- B
  - 0:B:30
- E
  - 6:D:20
  - 6

Nodes labeled with 'A_hilevel' and 'A_minId'. Connections indicate relationships between elements 0, 15, and 25.
MinId phase
MinId phase
MinId phase: final state

A: 0:A:31

E: 28:B:31

D: 25:A:31

L: 15:B:31

B: 0:B:31

25

6
MaxMinId phase

A
0:A:31

D
6:B:31

L
0:B:31

B
0:B:31

E
0:B:31

25

6

7

15
Termination

A
0

D
22

L
15

B
0

E
28

25

7

15

6
Outline

1. k-clustering
2. Weighted-Clustering
3. Example
4. Performances
Worst case

Number of clusterheads

\[(n - 1)OPT\]

Convergence

\[O(nk)\] rounds
Simulations, clusterheads
Simulations, clusterheads

59 processes
Radius = 163
Diameter = 282
Simulations, convergence

Number of rounds versus $k$

Unfair Daemon, different processor speeds
Fair Daemon, same processor speed
$n^*k$
Conclusion and Future work

Conclusion
- Self-stabilizing algorithm for the $k$-clustering problem on weighted graphs
- Based only on ID comparison
- Low memory consumption
- Bad worst cases,…
- …but in “practice” good results

Future work
- Reduce dependency towards ID
- Improve convergence time
- Use a message passing model
- Use $k$-clustering to improve grid middleware deployment, scheduling
Thanks for your attention!