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

E. Caron[1], A. K. Datta[2], B. Depardon[1] et L. L. Larmore[2]

[1] Université de Lyon. LIP.
UMR CNRS - ENS Lyon - INRIA - UCB Lyon 5668
[2] University of Nevada, Las Vegas, USA

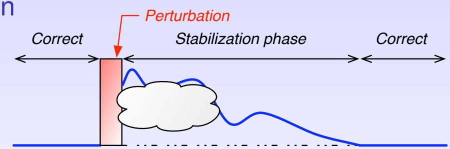
May 14th, 2009

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

Outline

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
- $radius(G) = \min_{x \in V} \max_{y \in V} \{w(x, y)\}$

Clustering

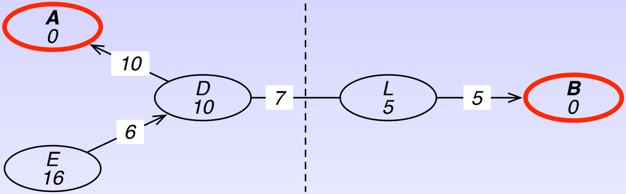
- C *k*-cluster: C connected subgraph of G, $radius(C) \leq k$
- $x \in C$ is a *clusterhead* if $\forall y \in C$, there exists a path such that

$$w(x, y) \leq k$$

- *k*-clustering of G: clustering of V into disjoint k-clusters

$\Longrightarrow$ Minimal k-clustering: $\mathcal{NP}$-hard problem

k-clustering



Find a k-clustering for $k = 20$

Clustering definition

For each node P, we define the following values:

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

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

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

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

Outline

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
- 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

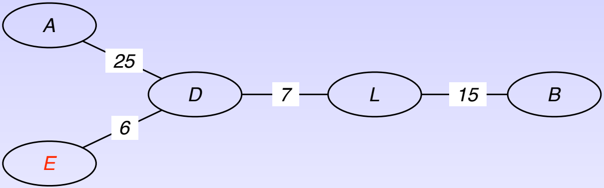
Interval *2 phases*: MinId and MaxMinId

Clustering k-clusters & trees creation

Model

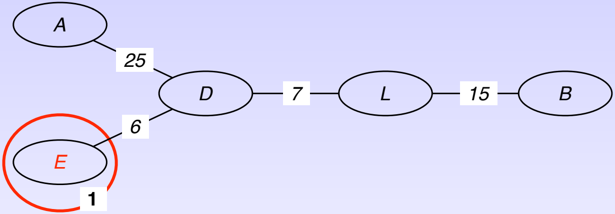
- Shared memory: read/write own variables, read neighbors'

Principle



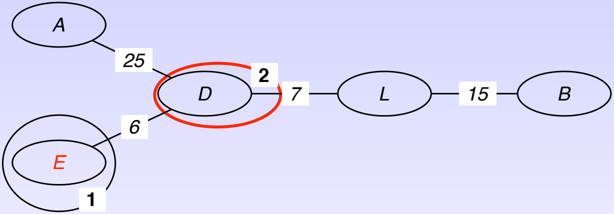
Find a k-clustering for $k = 30$

Principle



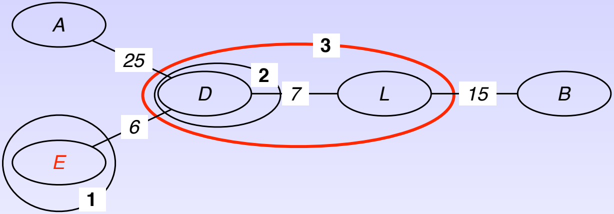
$$0 \leq d < 6 \leq k + 1 = 31$$

Principle



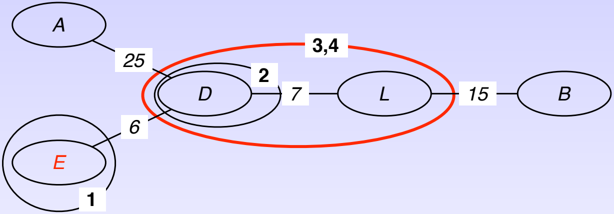
$$6 \leq d < 12 \leq k + 1 = 31$$

Principle



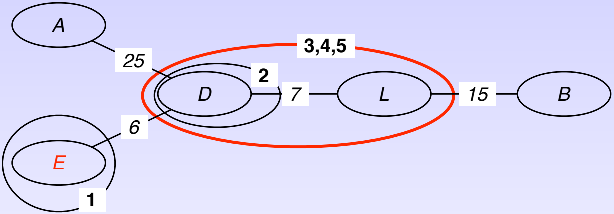
$$6 \leq d < 18 \leq k + 1 = 31$$

Principle



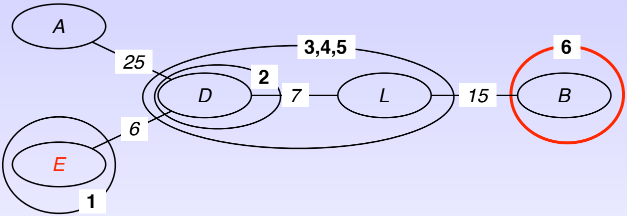
$$6 \leq d < 20 \leq k + 1 = 31$$

Principle



$$6 \leq d < 28 \leq k + 1 = 31$$

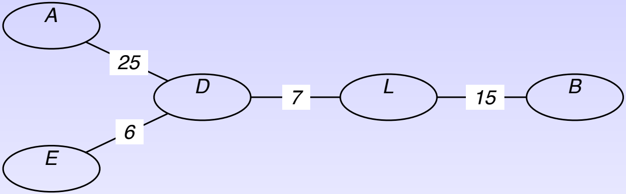
Principle



$$28 \leq d < 31 \leq k + 1 = 31$$

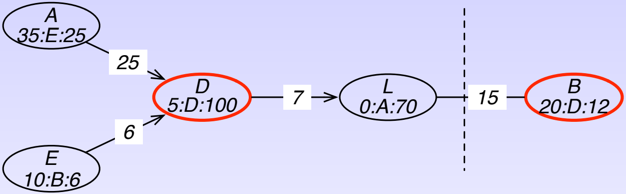
Outline

Initial graph



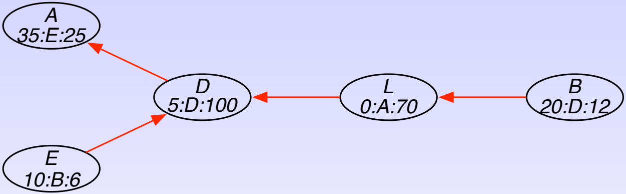
Find a k-clustering for $k = 30$

Initial graph: errors

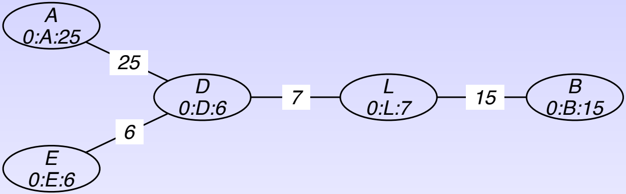


- Minimum research level: *minlevel*
- Minimum ID found so far: *minid*
- Maximum research level: *minhilevel*

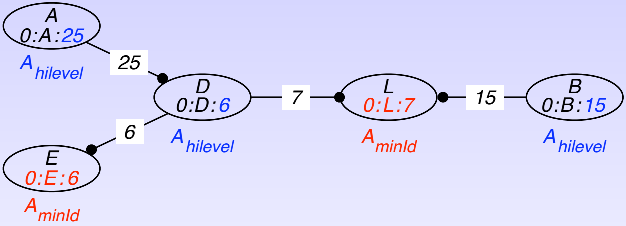
Initial graph: errors



Initial graph: errors



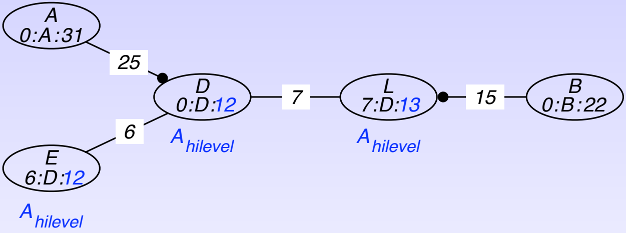
MinId phase



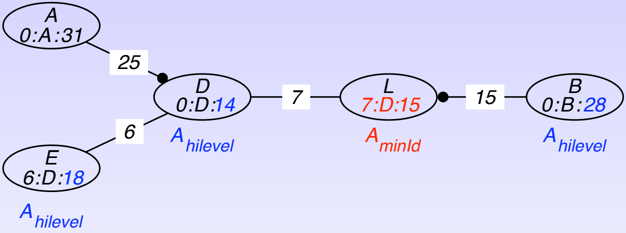
$A_{hilevel}$ $\Rightarrow$ updates *minhilevel*

A_{minId} $\Rightarrow$ updates *minlevel*, *minid* and *minhilevel*

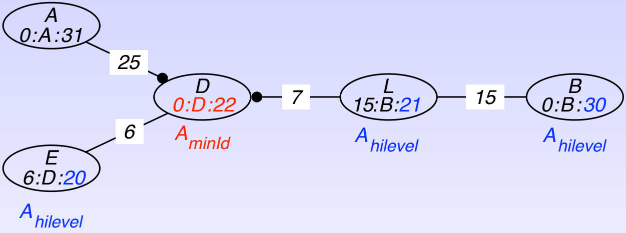
MinId phase


A
0:A:31
25
D
0:D:12
A hilevel
7
L
7:D:13
A hilevel
15
B
0:B:22
6
E
6:D:12
A hilevel

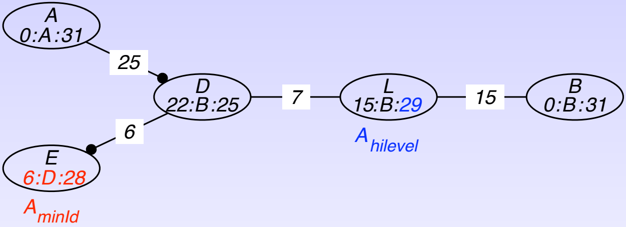
MinId phase


A
0:A:31
25
D
0:D:14
A hilevel
7
L
7:D:15
A minId
15
B
0:B:28
A hilevel
6
E
6:D:18
A hilevel

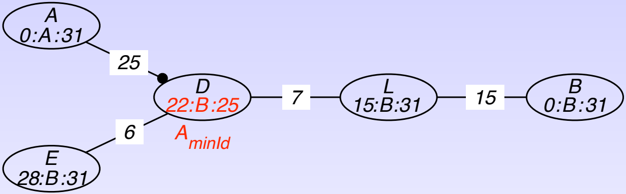
MinId phase


A
0:A:31
25
D
0:D:22
A minId
6
E
6:D:20
A hilevel
7
L
15:B:21
A hilevel
15
B
0:B:30
A hilevel

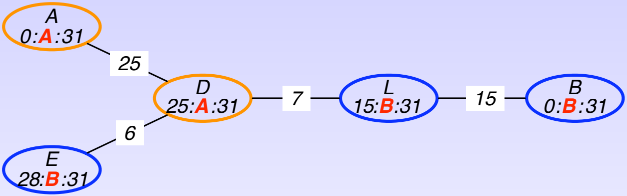
MinId phase



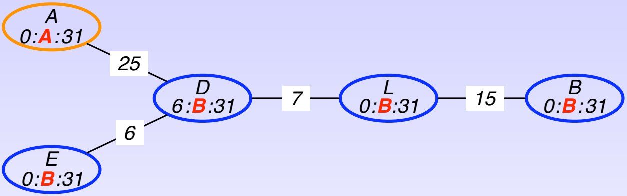
MinId phase



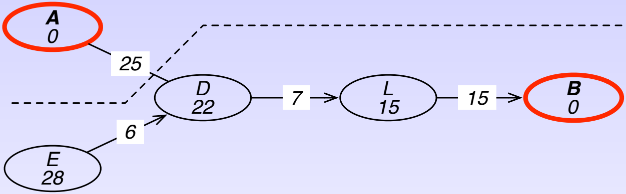
MinId phase: final state



MaxMinId phase



Termination

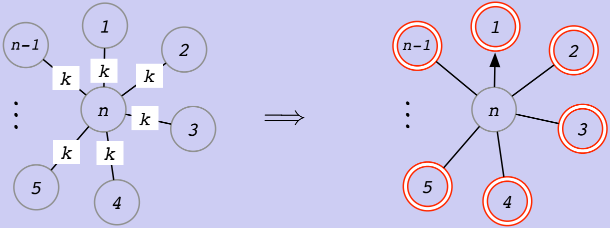


Outline

Worst case

Number of clusterheads

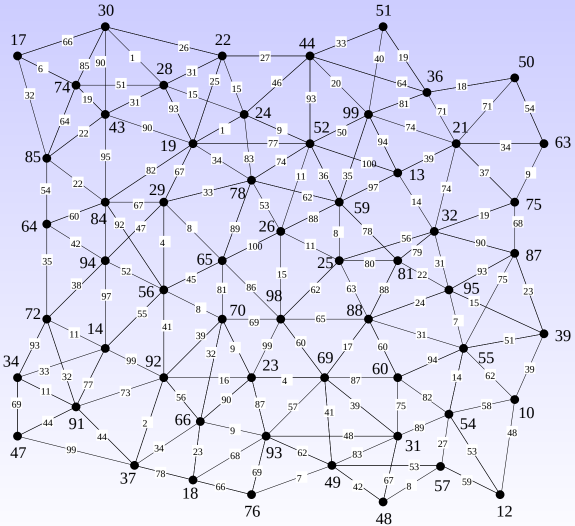
$(n-1)OPT$



Convergence

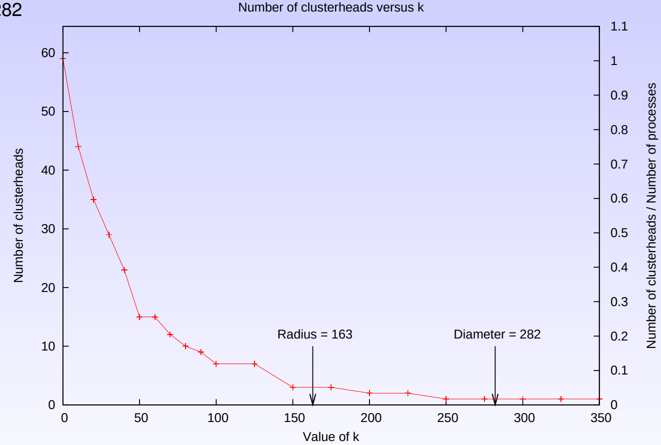
$O(nk)$ rounds

Simulations, clusterheads

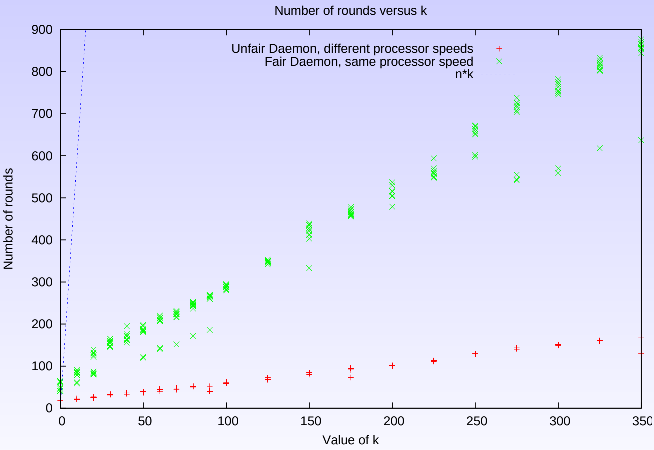


Simulations, clusterheads

59 processes
Radius = 163
Diameter = 282



Simulations, convergence



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!