1st Workshop on CyberSecurity

A Self-Stabilizing Algorithm for Maximal *p*-Star Decomposition of General Graphs

Brahim NEGGAZI [1], Volker TURAU [2], Mohammed HADDAD [1], Hamamache KHEDDOUCI [1]

[1] **Laboratoire d'InfoRmatique en Image et Systèmes d'information**
Université Claude Bernard Lyon (LIRIS/UCBL)

[2] **Institute of Telematics, Hamburg University of Technology,**
Hamburg, Germany (IT/TUHH)

To appear in the proceeding of the 15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2013)

Lyon- 17/10/2013

Team: Graphs, Algorithms and Multi-Agents (GrAMA)

Outlines

I. Introduction

II. Graph decomposition and self-stabilization

III. System Model

IV. Proposed self-stabilizing Algorithm for star decomposition

V. Proofs of proposed algorithm (Correcteness and convergence and complexity)

VI. Conclusion and Futur work

Introduction

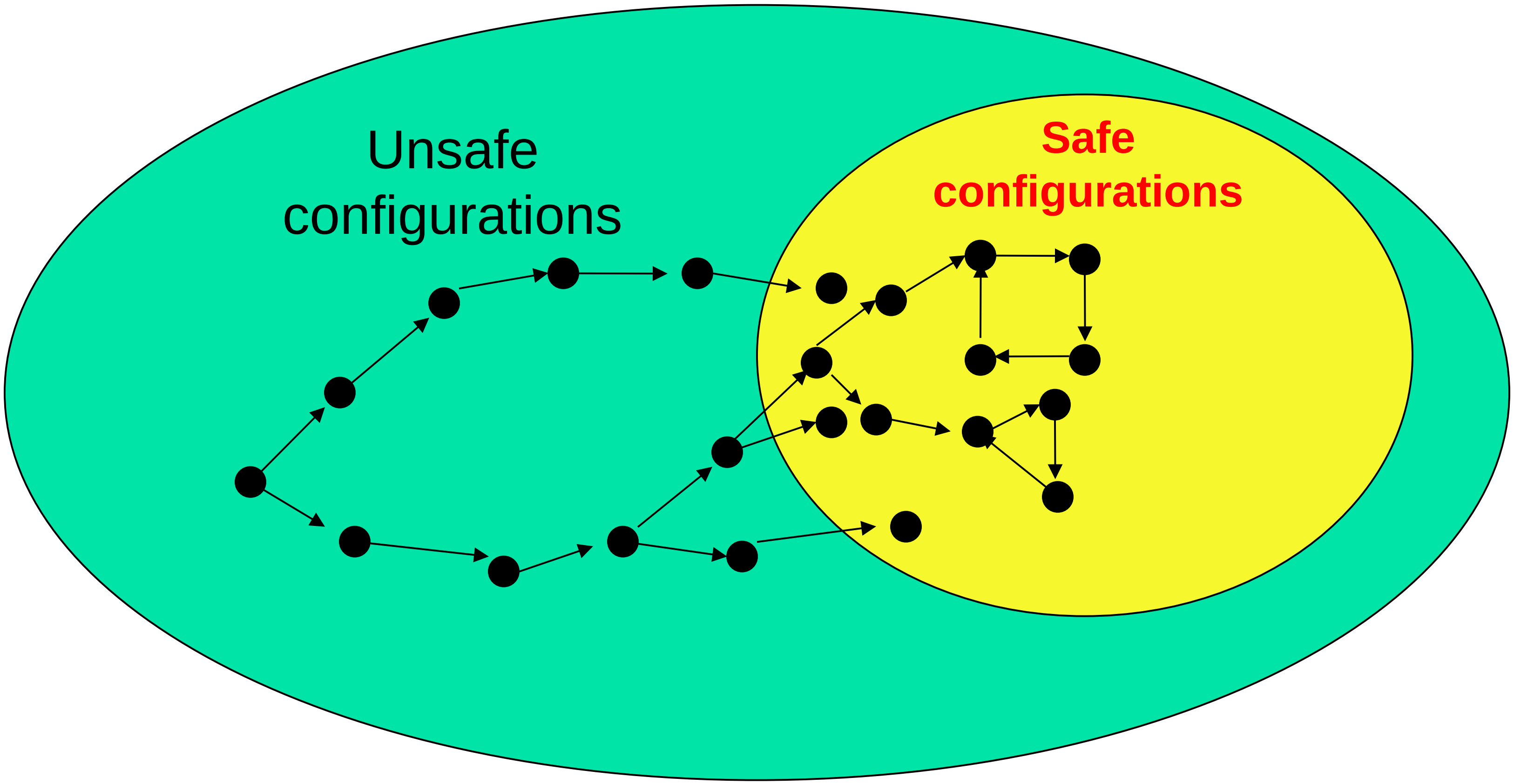
Larger computer Networks

Harder control
and management

Network decomposition

The decomposition problem is a way for partitioning a network into small components that satisfy some specific properties (topology, number of nodes, density, etc.).

Introduction



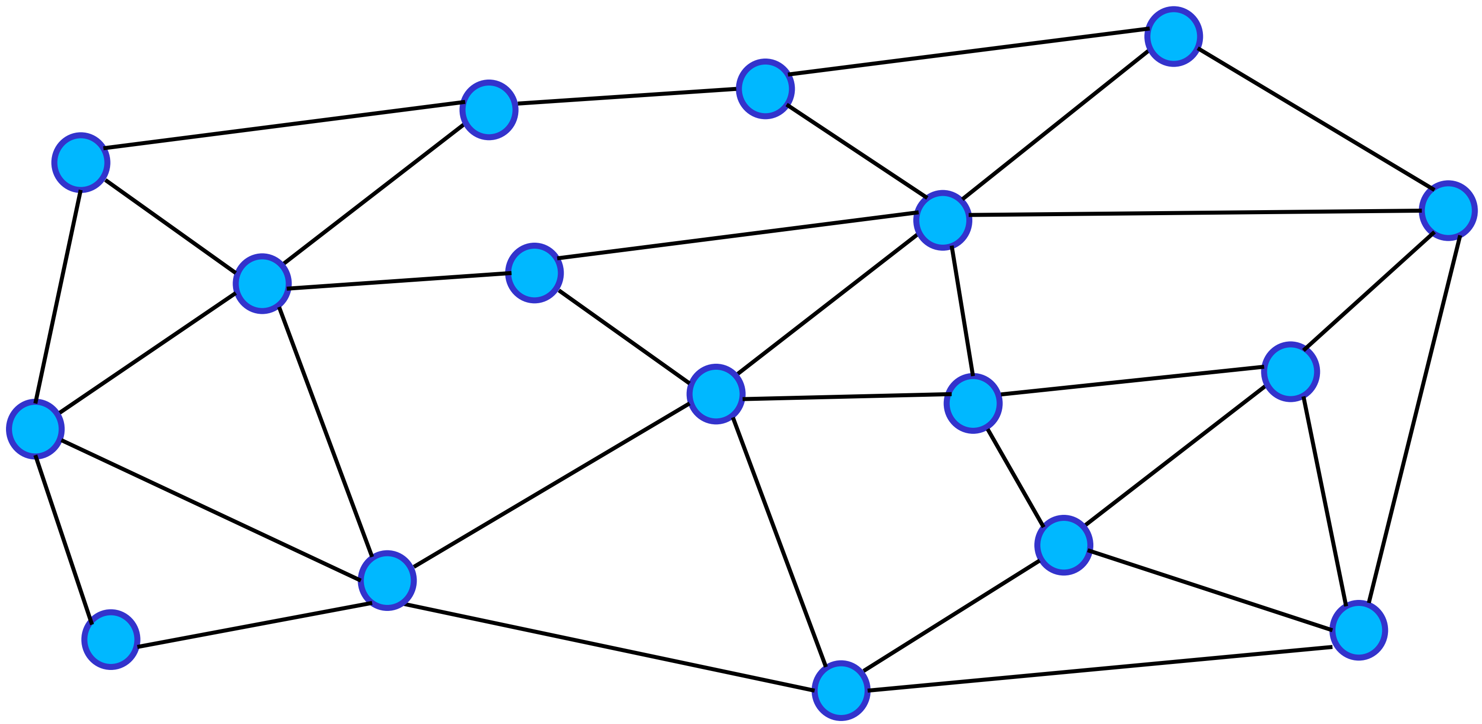
Self-stabilizing behavior of a system

Some self-stabilizing algorithms for graph decompositions

- F. Belkouch et al. in [IJPDC 02] considered a particular graph decomposition problem that consists in partitioning a graph of k^2 nodes into **k partitions** of order k.
- E. Caron et al. in [Euro-Par 09], C. Johnen et al. in [OPODIS 06], Bein et al. [ISPAN 05] focused on decomposing graphs into **clusters**.
- B. Neggazi et al. in [SSS 12] considered decomposition of graphs into **triangles**.

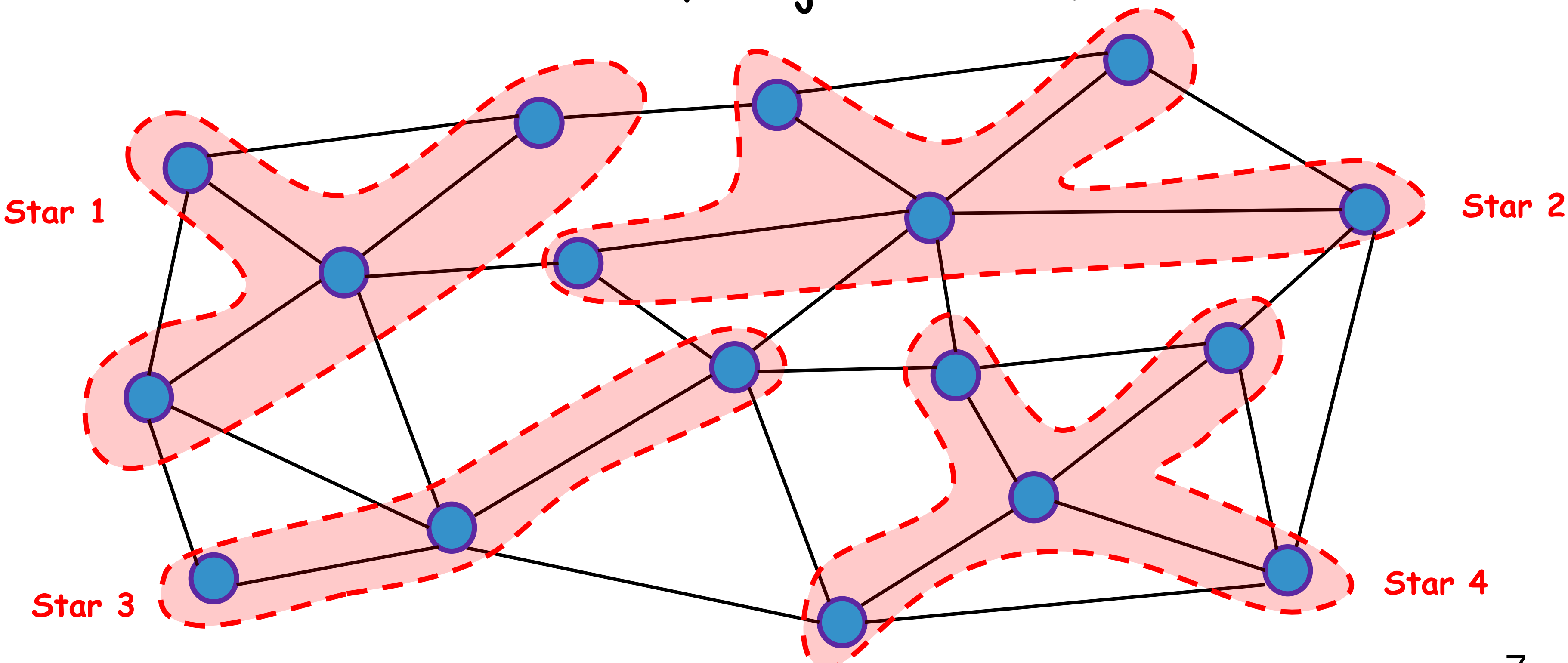
Star decomposition problem

This type of decomposition describes a graph as the union of disjoint stars.



Star decomposition problem

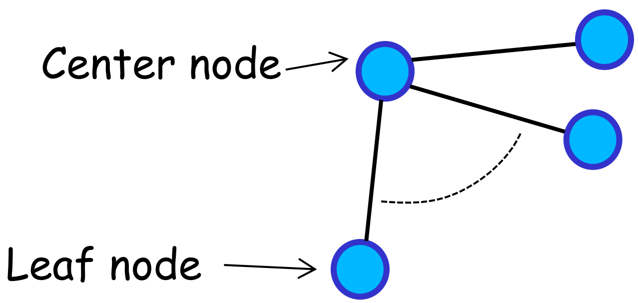
This type of decomposition describes a graph as the union of disjoint stars.



Star decomposition problem

A **uniform** decomposition into stars is one in which all stars have equal size.

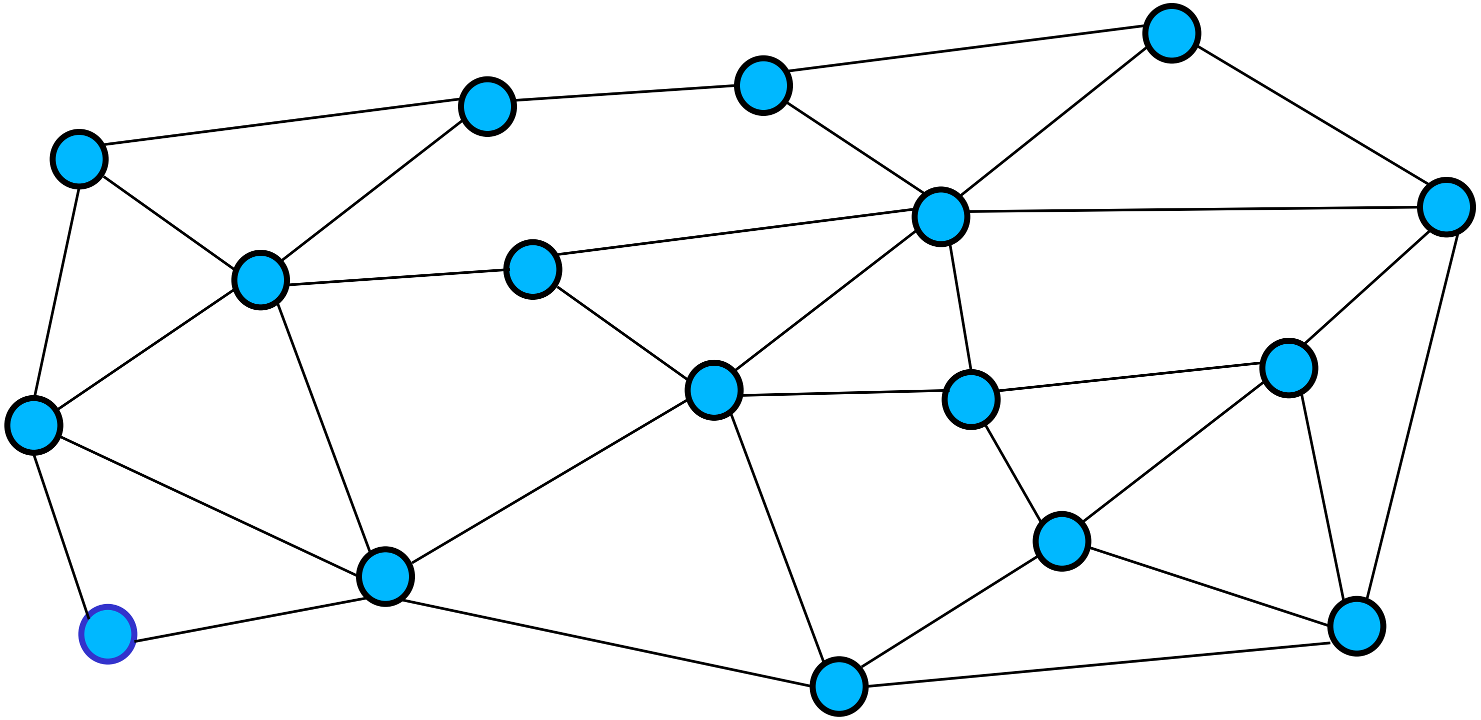
A p-star has one center node and p leaves where $p \geq 1$.



A **p-star decomposition** subdivides a graph into **p-stars**

Variant of generalized matchings and general graph factor problems that were proved to be NP-Complete [D. Kirkpatrick et al. in STOC 78] , Journ. Comp. 83]

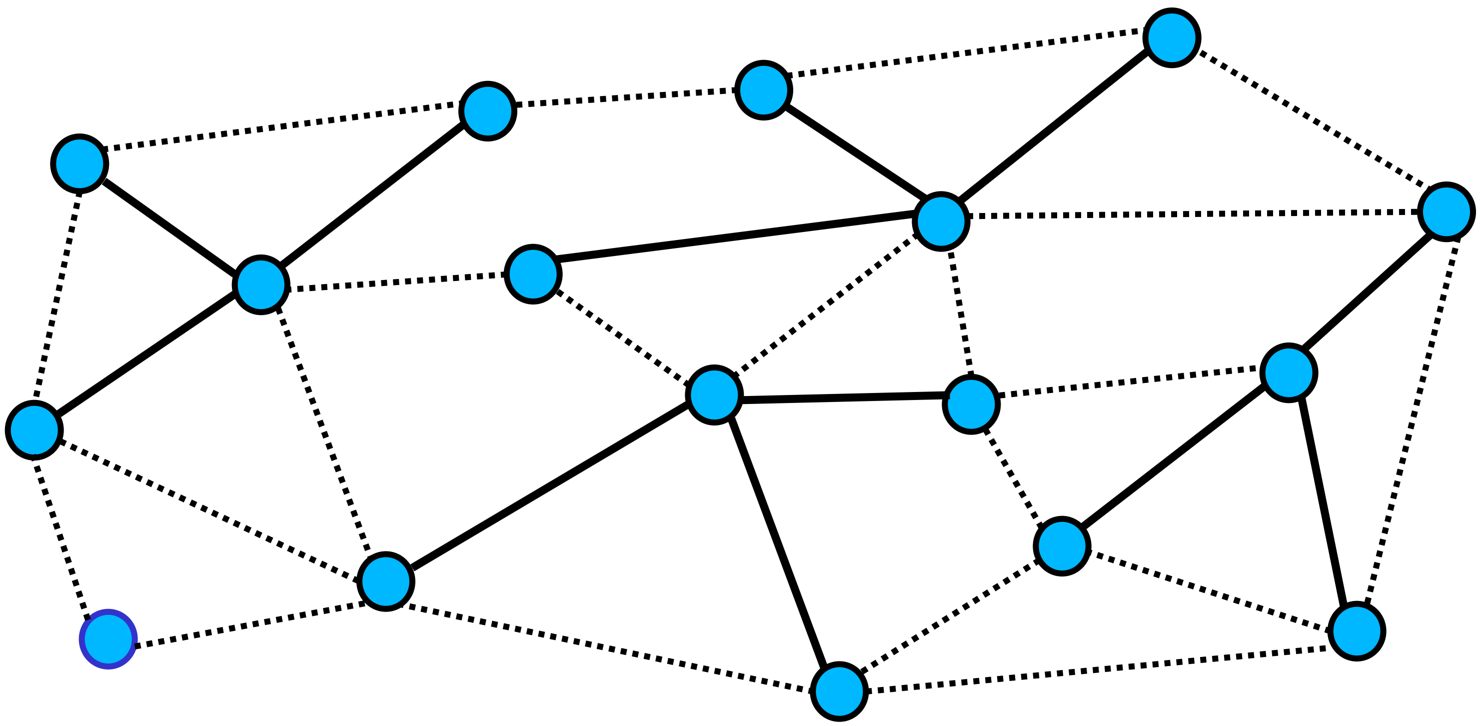
p-Star Decomposition of General Graphs



Graph G = (V,E)

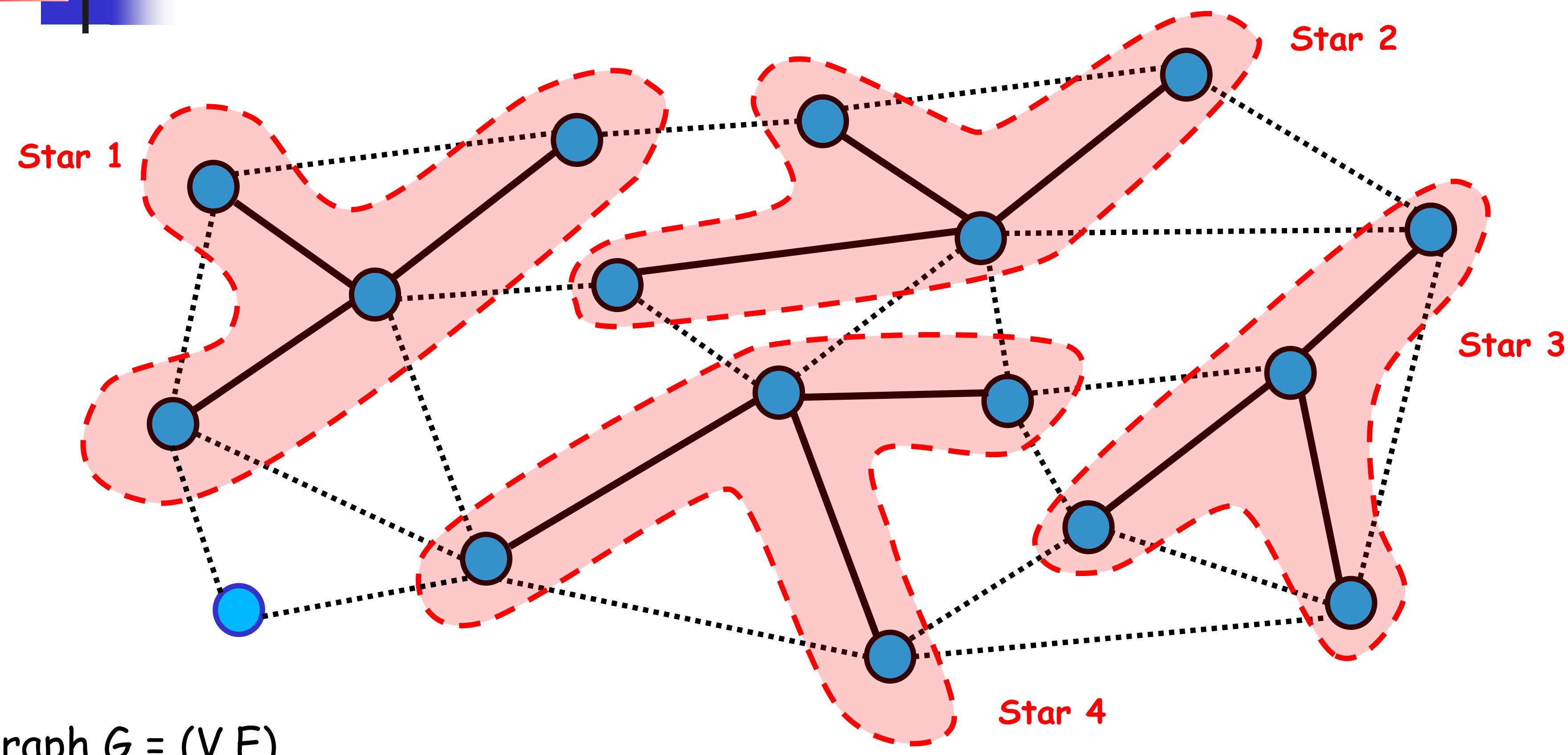
p=3

p-Star Decomposition of General Graphs



Graph $G = (V,E)$
$p=3$

p-Star Decomposition of General Graphs



Graph G = (V,E)

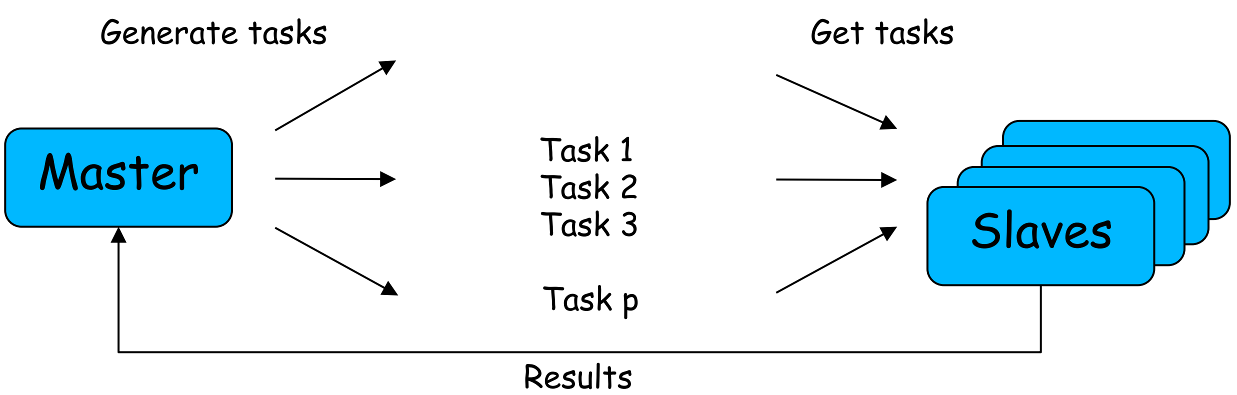
p=3

Maximal *p*-star Decomposition

p-Star Decomposition vs Master-Slaves paradigm

This decomposition offers similar paradigm as the **Master-Slaves paradigm** used in :

- ❖ Grid [M. Mezmaz PDP 07].
- ❖ P2P infrastructures [A. Bendjoudi Int. J. Grid Util. Comput 09].



Contribution

The purpose of this work is to

- Develop a distributed and **self-stabilizing algorithm** for decomposing a graph into p-stars.
- Operate with an unfair **Distributed Scheduler**.
- Suppose only local knowledge (Distance-1 knowledge).

System Model and Definitions

A **self-stabilizing** system, regardless of its initial configuration, converges in finite time, without any external intervention. [E.W. Dijkstra 74]

Each node v:

Begin

[If $p_1(v)$ then M_1] ; R_1
[If $p_2(v)$ then M_2]; R_2
....
[If $p_i(v)$ then M_i]; R_i

End

Self-stabilizing algorithm

p(v) is true -> v is enabled -> Move

System Model and Definitions

Two types of schedulers (daemons) :

- central (serial).
- **Distributed**.
  - Special case : Synchronous

Fairness:

- Fair.
- **Unfair (adversarial).**

System Model and Definitions

Two types of schedulers (daemons) :

- central (serial).
- **Distributed**.
  - Special case : Synchronous

Fairness:

- Fair.
- **Unfair (adversarial).**

NB. This work assumes the most general scheduler.

System Model and Definitions

Complexity :

- Moves
- Steps
- **Rounds**

System Model and Definitions

Graph *G* = (V,E),

Assume that each node "*v* " has "***id***" (locally distinct).

We denote : - **N(v)** open neighborhood,
- **d(v)** degree of a node v,
- ***p*** is a positive integer.

Let be S_i is a *p*-star

System Model and Definitions

Graph *G* = (V,E),

Assume that each node "*v*" has "***id***" (locally distinct).

We denote : - **N(v)** open neighborhood,
- **d(v)** degree of a node v,
- ***p*** is a positive integer.

Let be S_i is a *p*-star

Definition. A p-star Decomposition **D** of a graph ***G*** = **(V,E)** is a set of subgraphs of the form $S_i = (V_i, E_i)$ such that the sets $V_i \subseteq V$ are disjoint and each S_i is a p-star.

D is maximal if the subgraph induced by the nodes of ***G*** not contained in **D** does not contain a *p*-star as a subgraph.

Self-stablizing Algorithm for *p*-star Decomposition

Impossibility of finding a deterministic self-stabilizing algorithm for maximal **matching** in **anonymous** graph under a **distributed scheduler.** [F. Manne et al. TCS 2009]

p-star decomposition is a **generalization** of the **matching** problem for which $p = 1$

Self-stablizing Algorithm for *p*-star Decomposition

Impossibility of finding a deterministic self-stabilizing algorithm for maximal **matching** in **anonymous** graph under a **distributed scheduler.** [F. Manne et al. TCS 2009]

p-star decomposition is a **generalization** of the **matching** problem for which $p = 1$

Impossibility result remains valid for *p*-star **decomposition** for all $p \geq 1$

p-star decomposition algorithm requires a **mechanism for symmetry breaking**

Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

General idea

- **STEP 1 :** The node *v* with the smallest identifier having at least p neighbors becomes master.
- **STEP 2 :** The p neighbors $v_1, \ldots, v_p$ of v with the smallest identifiers become slaves of v.
- The previous steps are repeated for the subgraph of *G* consisting of all nodes except $v, v_1, \ldots, v_p$.

The challenge is to design an efficient distributed version of this algorithm under an unfair distributed scheduler.

Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

Let X be a set and *p* is a positive integer.

Two operators :

$$X^{p} = \begin{cases} \phi & \text{if } |X| < p \\ \text{the } p \text{ smallest elements of } X & \text{otherwise} \end{cases}$$

$$\min X = \begin{cases} null & \text{if } |X| = \phi \\ \text{the smallest element of } X & \text{otherwise} \end{cases}$$

Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

If identifier of v is smaller than identifier of u then we note $v < u$.

We define that $\forall v \in V:\ v < null$

Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

If identifier of v is smaller than identifier of u then we note $v < u$.

We define that $\forall v \in V:\ v < null$

Each node v maintains two variables:

- " **s** " contains the list of pointers to its p slaves
- " **m** " contains the pointer to the selected master.

Denote: $M(v) = \{w \in N(v) \mid v \in w.s\}$

$$S(v) = \{w \in N(v) \mid (w.s = \phi \wedge w.m \geq v) \vee (w.s \neq \phi \wedge w > v)\}$$

Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

SMSD uses the following code permitting a node ***v*** to compute its new values of $\mathbf{s_{new}}$ and $\mathbf{m_{new}}$.

$$
\begin{array}{l}
\textit{If } (\min M(v) < v \vee S(v)^p = \phi) \textit{ then} \\
\qquad v.s_{new} := \phi \,;\, v.m_{new} := \min M(v)\,; \\
\textit{else} \\
\qquad v.s_{new} := S(v)^p \,;\, v.m_{new} := null\,;
\end{array}
$$

Algorithm 1: Star Decomposition (SMSD)

Nodes: v is the current node

$$v.m \neq v.m_{new} \vee v.s \neq v.s_{new} \rightarrow v.m := v.m_{new}\,;\, v.s := v.s_{new}\,; \quad [R]$$

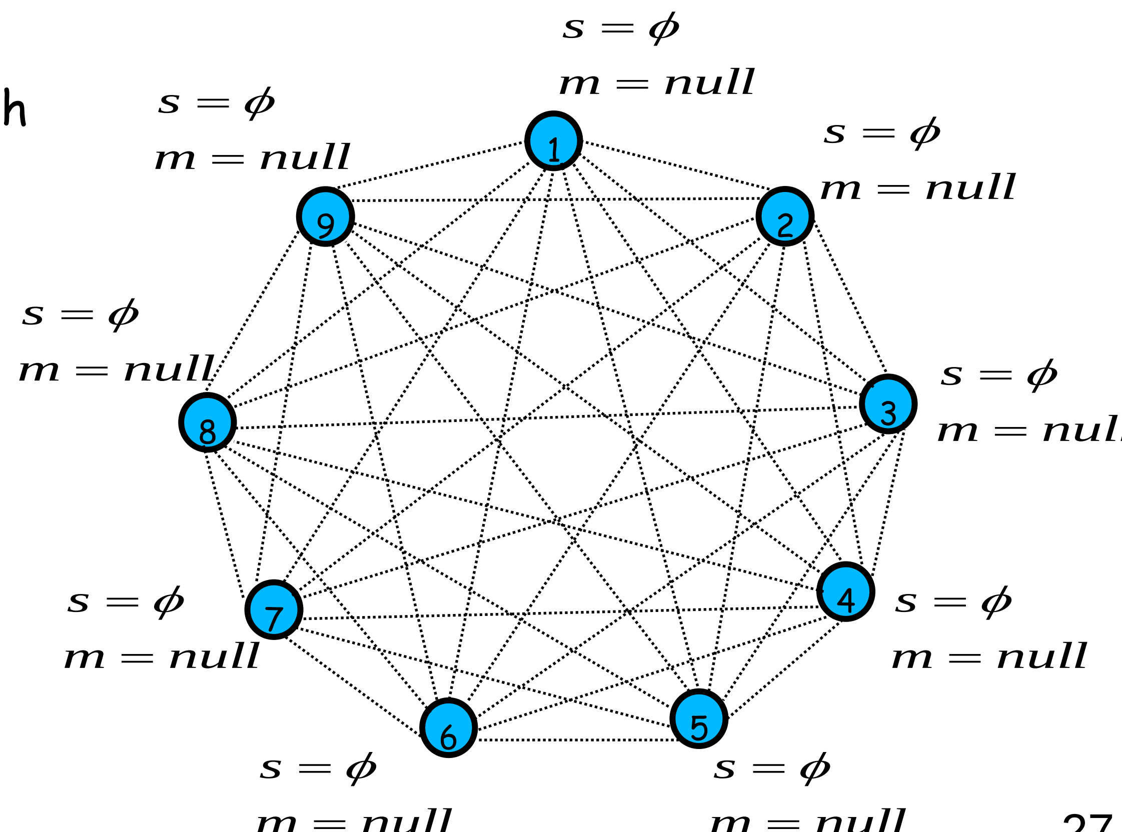
Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

Example of executing Algorithm SMSD
under the synchronous scheduler.

Graph G is a complete graph

Let ***p*** **=3**

Initial configuration



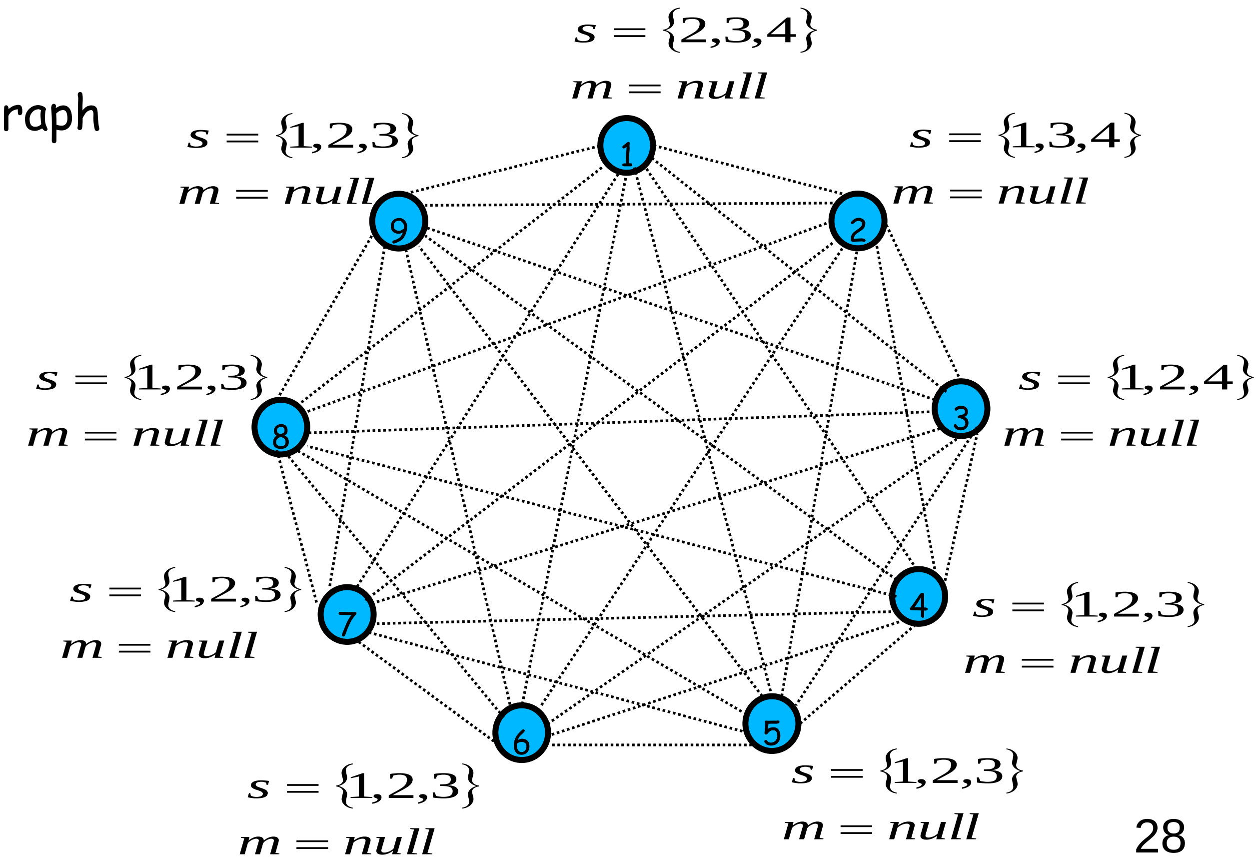
Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

Example of executing Algorithm SMSD
under the synchronous scheduler.

Graph G is a complete graph

Let ***p*** **=3**

Round 1



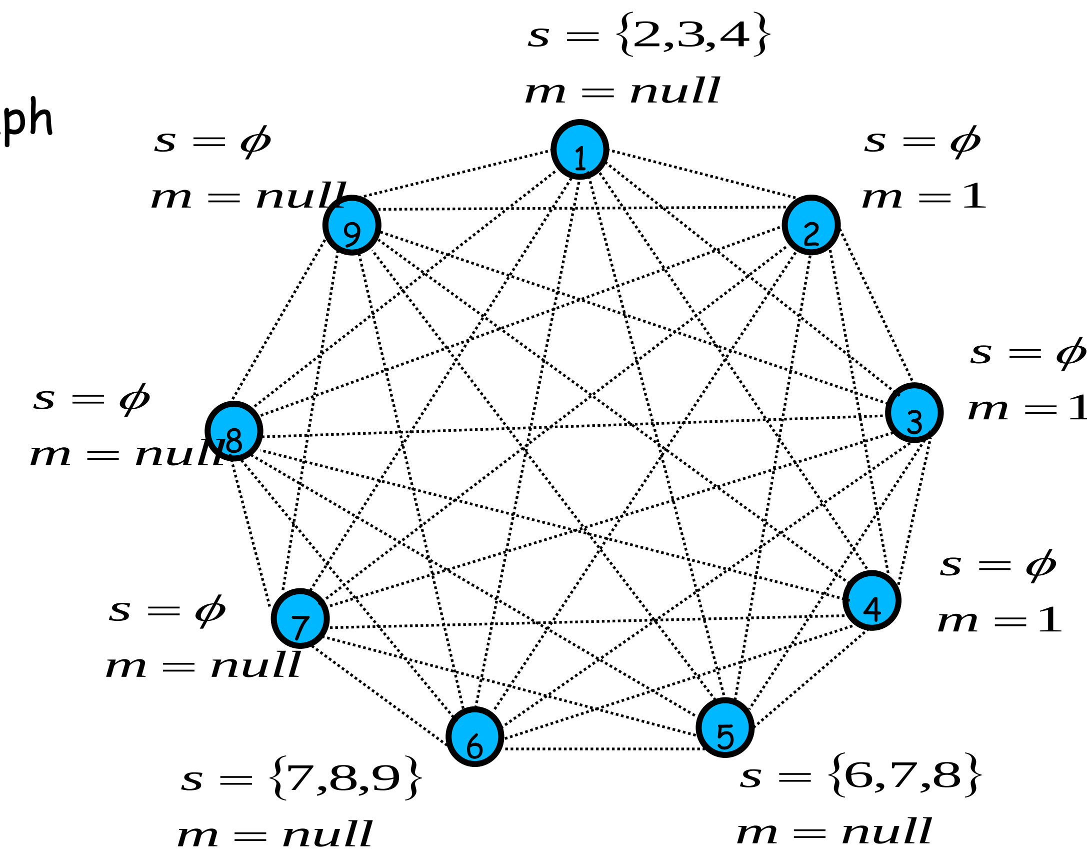
Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

Example of executing Algorithm SMSD
under the synchronous scheduler.

Graph G is a complete graph

Let ***p*** **=3**

Round 2



Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

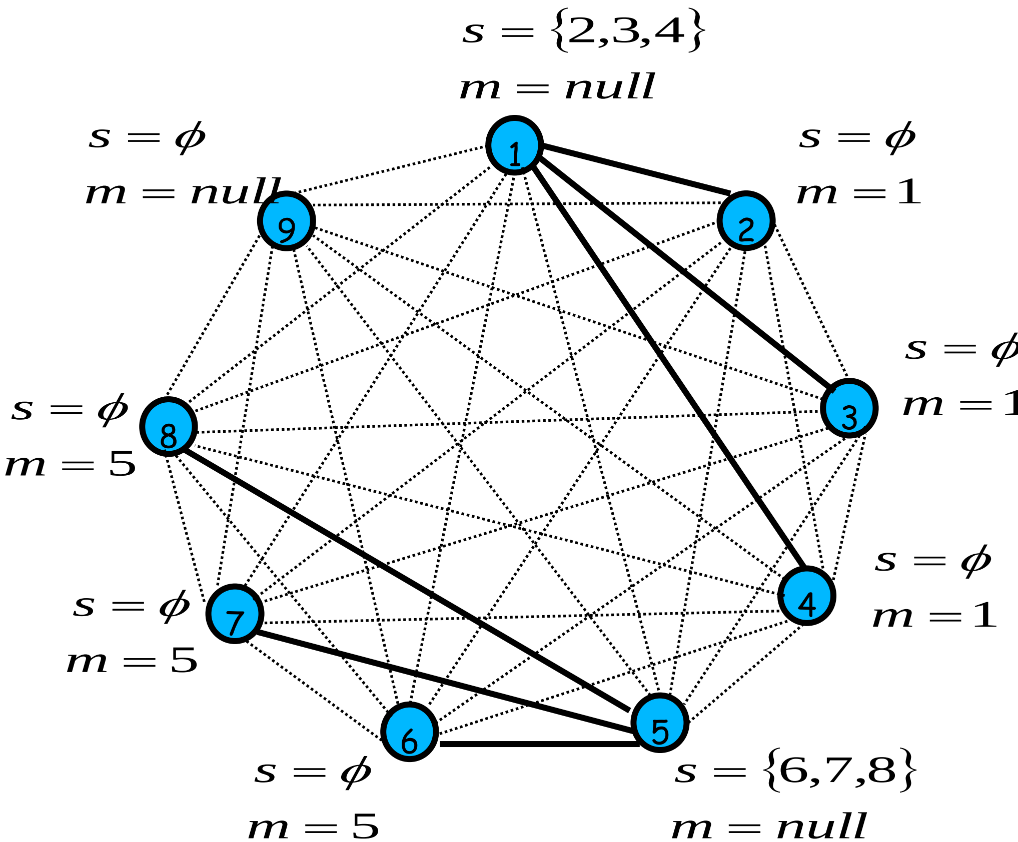
Example of executing Algorithm SMSD under the synchronous scheduler.

Graph G is a complete graph

Let ***p*** **=3**

Round 3

Final configuration



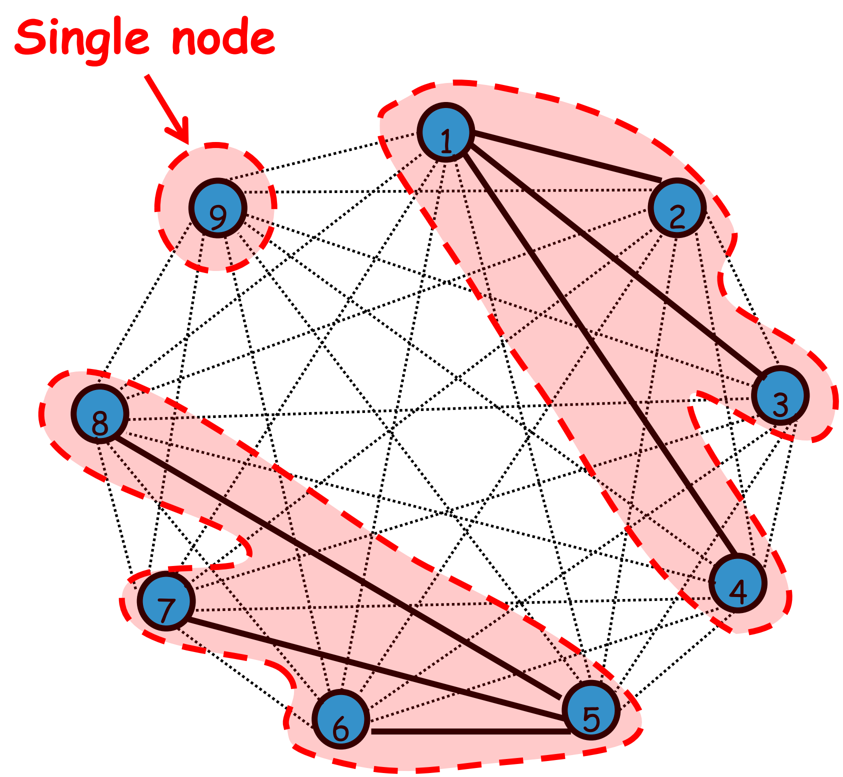
Self-stabilizing Algorithm for *p*-star Decomposition (SMSD)

Example of executing Algorithm SMSD
under the synchronous scheduler.

Graph G is a complete graph

Let p =3

Final configuration



Correctness proof

Lemma 1. In a configuration with no node is enabled, the following properties hold for each $v \in V$:

(a) $If\ v.s \neq \phi \quad then \quad v.s \subseteq N(v)\ and\ |v.s| = p\ and\ v.m = null.$

(b) $If\ v.m \neq null \quad then \quad v.m \in N(v).$

(c) $If\ v \in w.s \quad then \quad v.m = w\ and\ v.s = \phi.$

Correctness proof

Lemma 1. In a configuration with no node is enabled, the following properties hold for each $v \in V$:

(a) $If \;\; v.s \neq \phi \quad then \quad v.s \subseteq N(v) \; and \; |v.s| = p \; and \; v.m = null.$

(b) $If \;\; v.m \neq null \quad then \quad v.m \in N(v).$

(c) $If \;\; v \in w.s \quad then \quad v.m = w \; and \; v.s = \phi.$

Lemma 2. In a configuration with no enabled node the stars induced by all nodes v with $v.s \neq \phi$ form a maximal p-star decomposition of G.

Convergence under unfair distributed scheduler

The time complexity of the algorithm is measured in **rounds**.

A round under an unfair distributed scheduler may consist of an **infinite** number of **moves**.

Not sufficient to prove that the algorithm stabilizes after a finite number of rounds.

Theorem 1 : SMSD requires a finite number of moves.

Convergence under unfair distributed scheduler

The time complexity of the algorithm is measured in **rounds**.

A round under an unfair distributed scheduler may consist of an **infinite** number of **moves**.

Not sufficient to prove that the algorithm stabilizes after a finite number of rounds.

⇒

Theorem 1 : SMSD requires a finite number of moves.

For each node v : we distinguish

- ***m-move*** if v executes rule R and assigns a new value to *v.m*
- ***s-move*** if v executes rule R and assigns a new value to *v.s*

Note: A move can be a *m-move* and a *s-move* at the same time.

Convergence under unfair distributed scheduler

Lemma 3. Let $v \in V$ and "e" an execution of Algorithm SMSD such that no node u with $u < v$ makes an s-move in e. Then v makes at most $d(v)+2$ *s-moves* in e.

Convergence under unfair distributed scheduler

Lemma 3. Let $v \in V$ and "e" an execution of Algorithm SMSD such that no node u with $u < v$ makes an s-move in e. Then v makes at most $d(v)+2$ *s-moves* in e.

Lemma 4. The total number of s-moves in any execution of Algorithm SMSD is finite.

Convergence under unfair distributed scheduler

Lemma 3. Let $v \in V$ and "e" an execution of Algorithm SMSD such that no node u with $u < v$ makes an s-move in e. Then v makes at most $d(v)+2$ *s-moves* in e.

Lemma 4. The total number of s-moves in any execution of Algorithm SMSD is finite.

Lemma 5. Let Δ be the maximum node degree in the graph G. The total number of m-moves in any execution of Algorithm SMSD is at most $\Delta C + n$, here C denotes the total number of s-moves during the execution.

Convergence under unfair distributed scheduler

Theorem1. Algorithm SMSD is a self-stabilizing algorithm for computing a maximal p-star decomposition.

The complexity ??

Complexity analysis

Lemma 6. After round r_0 and in all following rounds, each node $v \in V$ satisfies the following properties.

(a) $v.m = null \quad or \quad v.m \in N(v).$

(b) $If\ v.s \neq \phi \quad then\ |v.s| = p \wedge v.s \subseteq N(v) \wedge d(v) \geq p \wedge v.m = null.$

Complexity analysis

Lemma 6. After round r_0 and in all following rounds, each node $v \in V$ satisfies the following properties.

(a) $v.m = null \quad or \quad v.m \in N(v).$

(b) $If\ v.s \neq \phi \quad then\ |v.s| = p \wedge v.s \subseteq N(v) \wedge d(v) \geq p \wedge v.m = null.$

Lemma 7. After round r_1 and in all following rounds, each node $v \in V$ with $v.m = u$ satisfies $d(u) \geq p$ and $v.s = \phi.$

Complexity analysis

Lemma 8. Let v^* be the smallest node in G such that $d(v^*) \geq p$. Then,

(a) After round r_2 and in all following rounds,

$$v^*.m = null \quad and \quad v^*.s = N(v^*)^p$$

(b) Let be $S^* = (v^* \cup v^*.s)$. After round r_3 and in all following rounds,

$$v.m \notin S^* \; and \; v.s \cap S^* = \phi \quad for \;\; all \;\; v \in V(G) \setminus S^*.$$

Complexity analysis

Lemma 8. Let v^* be the smallest node in G such that $d(v^*) \geq p$. Then,

(a) After round r_2 and in all following rounds,

$$v^*.m = null \quad and \quad v^*.s = N(v^*)^p$$

(b) Let be $S^* = (v^* \cup v^*.s)$. After round r_3 and in all following rounds,

$$v.m \notin S^* \ and \ v.s \cap S^* = \phi \quad for \ all \ v \in V(G) \setminus S^*.$$

Lemma 9. Algorithm SMSD stabilizes after at most $2\left\lfloor \frac{n}{p+1} \right\rfloor + 2$ rounds.

Complexity analysis

Theorem 2. Algorithm SMSD is self-stabilizing algorithm for maximal *p*-star decomposition and converges after at most $2\left\lfloor \frac{n}{p+1} \right\rfloor + 2$ rounds under the unfair distributed scheduler using *O(p log n)* memory.

Conclusions & future work

- First self-stabilizing algorithm for graph decomposition into disjoint p-stars (SMSD).
- SMSD operates under the unfair distributed scheduler and stabilizes after at most $2\left\lfloor \frac{n}{p+1} \right\rfloor + 2$ rounds.

Conclusions & future work

- First self-stabilizing algorithm for graph decomposition into disjoint p-stars (SMSD).
- SMSD operates under the unfair distributed scheduler and stabilizes after at most $2\left\lfloor \frac{n}{p+1} \right\rfloor + 2$ rounds.
- The proposed algorithm generalizes maximal matching algorithms where *p = 1*. The time complexity in rounds of SMSD has the same order as the best known self-stabilizing algorithm for maximal matching under the synchronous scheduler or the distributed scheduler.

Conclusions & future work

- SMSD requires at most $O(\frac{n^2}{p})$ moves using the synchronous scheduler.
- The exact move complexity of the algorithm under the unfair distributed scheduler is unknown.

Conclusions & future work

- SMSD requires at most $O(\frac{n^2}{p})$ moves using the synchronous scheduler.
- The exact move complexity of the algorithm under the unfair distributed scheduler is unknown.

As future works, we aim to

- Bound moves complexity of SMSD.
- Generalize SMSD to weighted graphs.

End