Spanning Tree Construction

MPRI

Lélia Blin

lelia.blin@irif.fr

2023
Spanning Tree Problem Specification

Let $G(V, E)$ a connected undirected graph. An acyclic subgraph that connects all the nodes of $G$ is called a spanning tree of $G$, denoted by $ST(G)$. 
Pointer notion

In a spanning tree, the acyclic property ensures that there are no loops or cycles, while the requirement for each node to have a unique pointer to a neighbor guarantees a distinct and unambiguous path between nodes.
Self-stabilizing Spanning Tree construction

Main difficulties

- Breaking cycle
- Empty Node pointer
Spanning Tree under constraint

- BFS Spanning Tree
- DFS Spanning Tree
- Minimum spanning Tree
- Minimum Degree Spanning Tree
- Minimum Diameter Spanning Tree
A self-stabilizing algorithm for constructing a spanning trees

1991 Chen Yu Huang IPL
Model

- Semi-uniform $\rightarrow$ root node denoted by $r$
- Anonymous
- Knowledge: $n$
- Scheduler : central (Only one node activate at each step)
Local variables

- Level: $L_v \in \{1, \text{dots}, n\}$
- Parent: $p_v \in \{0, \ldots, n\}$

- Remark: $r$ has no parent and is level is zero
- These variables uses $O(\log)$ bits of memory by node
Algorithm

\[ R_0 : L_v \neq n \land L_v \neq L_{p_v} + 1 \land L_{p_v} \neq n \rightarrow L_v = L_{p_v} + 1 \]
\[ R_1 : L_v \neq n \land L_{p_v} = n \rightarrow L_v = n \]
\[ R_2 : L_v = n \land \exists u \in N(v) \mid L_u < n - 1 \rightarrow L_v = L_u + 1; p_v = u \]
Example

\[\text{Init. Config.:} \quad \begin{array}{c}
\circlearrowleft R1 \\
3 \quad e \\
\circlearrowright R2 \\
2 \quad b \\
\circlearrowright R2 \\
0 \quad c
\end{array}
\]
Example
Legal Configuration

\[ GST \equiv (\forall v \in V \setminus \{r\} | L_v = L_{p_v} + 1) \]
Definitions

- A parent pointer is called **Well-Formed** (WF) pointer if
  \[ L_v \neq n \land L_{p_v} \neq n \land L_v = L_{p_v} + 1 \]
- \( S_v^{L_v} \) to denote the WF set rooted at \( v \) (tree structure)
  - Ex Fig1 initial \( S_c^0 = \{ c \}, S_a^1 = \{ a, b, e \}, S_d^5 = \{ d \} \).
Lemma 1: Before $GST$ is true, the algorithm does not terminate.

Proof: Before $GST$ is true, there must exist some WF set $S^L_v$, with $v \neq r$. Two possible cases need to be considered:

1. We consider $v \neq r$ if $L_v \neq n$; $v$ is activatable by $R_0$ or $R_1$ because either $L_{p_v} \neq n$ or $L_{p_v} = n$.

2. $L_v = n$ for every WF set $S^L_v$, $v \neq r$. Since $G$ is connected, there exists at least one edge between a node $u$ in $S^0_r$ and some node $v$, $v \neq r$ and $S^L_v = S^n_v = \{v\}$. Because $L_r = 0$ and $|S^0_r| < n$, we have $L_u < n - 1$. Hence, node $v$ can apply $R_2$ to make a move. □
Definitions of function $F$

- Let $\gamma$ be a configuration and let $t_k$: $0 \leq k \leq n$ be the the number of WF set $S_{L_v}^v$ such that $L_v = k$.
- $F(\gamma) = (t_0, t_1, \ldots, t_n)$ with $0 \leq i < n$
- The comparison of $F$ is lexicographic.
- $F$ is bounded function with the maximum value $(1, n_1, 0, \ldots, 0)$ and minimum $(1, 0, \ldots, 0)$. 
Init. Config.

Step 0: $S_0^0 = \{c\}, S_0^1 = \{a, b, e\}, S_0^2 = \{d\}$  \hspace{1cm}  $F = (1, 1, 0, 0, 0, 1)$

Step 1: $S_1^0 = \{c\}, S_1^1 = \{b, e\}, S_1^2 = \{a\}, S_1^3 = \{d\}$  \hspace{1cm}  $F = (1, 0, 1, 0, 0, 2)$

Step 2: $S_2^0 = \{c\}, S_2^1 = \{e\}, S_2^2 = \{a\}, S_2^3 = \{b\}, S_2^4 = \{d\}$  \hspace{1cm}  $F = (1, 0, 0, 1, 0, 3)$

Step 3: $S_3^0 = \{c, b\}, S_3^1 = \{e\}, S_3^2 = \{a\}, S_3^3 = \{d\}$  \hspace{1cm}  $F = (1, 0, 0, 1, 0, 2)$

Step 4: $S_4^0 = \{c, b, d\}, S_4^1 = \{e\}, S_4^2 = \{a\}$  \hspace{1cm}  $F = (1, 0, 0, 1, 0, 1)$

Step 5: $S_5^0 = \{c, b, d, e\}, S_5^1 = \{a\}$  \hspace{1cm}  $F = (1, 0, 0, 0, 0, 1)$

Step 6: $S_6^0 = \{c, b, d, e, a\}$  \hspace{1cm}  $F = (1, 0, 0, 0, 0, 0)$
Step 0: $S_0^0 = \{ c \}, S_0^1 = \{ a, b, e \}, S_0^2 = \{ d \}$

$F = (1, 1, 0, 0, 0, 1)$

Step 1: $S_1^0 = \{ c \}, S_1^1 = \{ b, e \}, S_1^2 = \{ a \}, S_1^3 = \{ d \}$

$F = (1, 0, 1, 0, 0, 2)$

Step 2: $S_2^0 = \{ c \}, S_2^1 = \{ e \}, S_2^2 = \{ a \}, S_2^3 = \{ b \}, S_2^4 = \{ d \}$

$F = (1, 0, 0, 1, 0, 3)$

Step 3: $S_3^0 = \{ c, b \}, S_3^1 = \{ e \}, S_3^2 = \{ a \}, S_3^3 = \{ d \}$

$F = (1, 0, 0, 1, 0, 2)$

Step 4: $S_4^0 = \{ c, b, d \}, S_4^1 = \{ e \}, S_4^2 = \{ a \}$

$F = (1, 0, 0, 1, 0, 1)$

Step 5: $S_5^0 = \{ c, b, d, e \}, S_5^1 = \{ a \}$

$F = (1, 0, 0, 0, 0, 1)$

Step 6: $S_6^0 = \{ c, b, d, e, a \}$

$F = (1, 0, 0, 0, 0, 0)$
Rule $R_0$

**Lemma 2:** $F$ monotonically decreases each time when rule $R_0$ is applied.
Example of execution

\[ R_0 : L_v \neq n \land L_v \neq L_{p_v} + 1 \land L_{p_v} \neq n \rightarrow L_v = L_{p_v} + 1 \]

\[ \gamma : S^0_r = \{r\}, S^2_{p_v} = \{p_v\}, S^3_x = \{x\}, S^4_v = \{v, u, w\} \rightarrow F(\gamma) = (1, 0, 1, 1, 1, 0, 0) \]

\[ \gamma' : S^0_r = \{r\}, S^2_{p_v} = \{p_v, v\}, S^3_x = \{x\}, S^5_u = \{u, w\} \rightarrow F(\gamma') = (1, 0, 1, 1, 0, 1, 0) \]

\[ F(\gamma) < F(\gamma') \]
Proof of lemma 2

- Let us consider a node \( v \), with \( k = L_v(\gamma) \) such that \( v \in S^k_v \) in \( \gamma \) so \( t_k \) is not null in \( \gamma \);
- Let \( p_v \) be the parent of \( v \) in configuration \( \gamma \), if \( v \) can execute \( R_0 \):
  \[
  k_p = L_{p_v}(\gamma) \neq k - 1.
  \]
- If the node \( v \) executes \( R_0 \) in configuration \( \gamma \), it becomes element of \( S^{k_p}_{p_v} \), so \( S^k_v \) disappears in \( \gamma' \) and \( t_k(\gamma') < t_k(\gamma) \).
Proof of lemma 2

- Let denoted by \( u_i \) the children of \( v \) in configuration \( \gamma \) such that \( L_{u_i} = k_i = k + 1 \), all the children was in \( S^k_v(\gamma) \) but in \( \gamma' \) the child \( u_i \in S^{k_i}_{u_i} \), so \( t_{k_i}(\gamma') > t_k(\gamma) \), remember that \( k_i = k + 1 \).

- As a consequence,
  \[
  F(\gamma) = (1, \ldots, t_k(\gamma), t_{k_i}(\gamma), \ldots) < F(\gamma') = (1, \ldots, t_k(\gamma'), t_{k_i}(\gamma'), \ldots)
  \]
  because \( t_k(\gamma) < t_k(\gamma') \) and \( t_{k_i}(\gamma) > t_{k_i}(\gamma') \).

\[\square\]
Rule $R_1$

**Lemma 3**: $F$ monotonically decreases each time when rule $R_1$ is applied.
Example of execution

\[ R_1 : L_v \neq n \land L_{p_v} = n \rightarrow L_v = n \]

\( \gamma \)

\[ \begin{array}{cccccc}
\gamma & 0 & 3 & 6 & 4 & 5 & 6 \\
 \ \\
 r & x & p_v & v & u & w \\
 \end{array} \]

\( \gamma' \)

\[ \begin{array}{cccccc}
\gamma' & 0 & 3 & 6 & 6 & 5 & 6 \\
 \ \\
 r & x & p_v & v & u & w \\
 \end{array} \]

\( \gamma : S_r^0 = \{r\}, S_x^3 = \{x\}, S_v^4 = \{v, u, w\}, S^6 = \{p_v\} \rightarrow F(\gamma) = (1, 0, 0, 1, 1, 0, 1) \)

\( \gamma' : S_r^0 = \{r\}, S_x^3 = \{x\}, S_u^5 = \{u, w\}, S_{p_v}^6 = \{p_v\}, S_v^6 = \{v\} \rightarrow F(\gamma') = (1, 0, 0, 1, 0, 1, 2) \)

\[ F(\gamma) < F(\gamma') \]
Proof of lemma 3

- Let us consider a node $v$, with $L_v(\gamma) = k, k < n$ and $L_{p_v} = n$
- So in $\gamma$ we have $v \in S^v_k$ and $p_v \in S^n_{p_v}$ as a consequence $t_k(\gamma)$ and $t_n(\gamma)$ are not nul;
- If the node $v$ executes $R_1$ in configuration $\gamma$, it becomes in $\gamma'$ element of $S^n_v$, and $S^k_v$ disappears so $t_k(\gamma') < t_k(\gamma)$ and $t_n(\gamma') > t_n(\gamma)$.
- Like $k < n$ we obtain $F(\gamma') < F(\gamma)$.

$\Box$
Rule $R_2$

**Lemma 4:** $F$ monotonically decreases each time when rule $R_2$ is applied.
Example of Execution of $R_2$

$R_2 : L_v = n \land \exists u \in N(v) \mid L_u < n - 1 \implies L_v = L_u + 1; p_v = u$

γ: $S_r^0 = \{r\}, S_x^3 = \{x\}, S_u^5 = \{u, w\}, S_{p_v}^6 = \{p_v\}, S_v^6 = \{v\} \implies F(\gamma') = (1, 0, 0, 1, 0, 1, 2)$

γ: $S_r^0 = \{r\}, S_x^3 = \{x, p_v\}, S_u^5 = \{u, w\}, S_v^6 = \{v\} \implies F(\gamma') = (1, 0, 0, 1, 0, 1, 1)$

→ $F(\gamma) < F(\gamma')$
Proof of lemma 4

Proof: Let us consider a node $v$ such that $v \in S_v^n$ in $\gamma$ so $t_n(\gamma)$ is not null in $\gamma$; after execution of rule $R_2$ by the node $v$ in $\gamma$ $S_v^n$ disappears and $t_n(\gamma') = t_n(\gamma) - 1$, now $v$ is in $S_u^k$ and $t_k(\gamma') = t_k(\gamma')$ because $v$ reaches a parent with the good level. By definition of $R_2$ $k < n - 1$ so we obtain $F(\gamma') < F(\gamma)$. □
**Theorem** Eventually, the system reaches a legitimate configuration.

**Proof:** Since the initial value for $F$ is finite, and the smallest possible value for $F$ is $(1, 0, \ldots, 0)$, by Lemmas 2, 3 and 4 the rules can only be applied a finite number of times. Hence, by Lemma 1, eventually $GST$ is true. □
Conclusion

- Silent self-stabilizing algorithm
- Centralized Scheduler
- Knowledge: $n$
- $O(\log_2 n)$ bits of memory per node
- Time complexity not provided

Question: Is it space optimal?
A self-stabilizing algorithm for constructing breadth-first trees

Chen Huang 1992
Model

- Semi-uniform $\rightarrow$ root node denoted by $r$
- Anonymous
- Knowledge: $n$
- Scheduler: Distributed fair scheduler
Local variables

- Level: $L_v \in \{1, \text{dots}, n\}$
- Parent: $p_v \in \{0, \ldots, n\}$

- Remark: $r$ has no parent and is level is zero
- These variables uses $O(\log)$ bits of memory by node
Algorithm

- $R_0 : L_v \neq L_{p_v} + 1 \land L_{p_v} \neq n \rightarrow L_v = L_{p_v} + 1$
- $R_1 : L_v > k \rightarrow L_v = k + 1; p_v = k_{id}$
  - with $k = \min\{L_u | u \in N(v)\}$
  - $k_{id} = \min\{id_u | u \in N(v) \land L_u = k\}$
**Configuration légale**

\[
BFT \equiv (\forall v \in V \setminus \{r\}| L_v = Lp_v + 1 \land L_{p_v} = \min\{L_u | u \in N(v)\})
\]

**Remarque:** \(BTF(v) \equiv L_v = Lp_v + 1 \land L_{p_v} = \min\{L_u | u \in N(v)\}\)
No Deadlock

Lemma 1: before $BFT$ is true the system never causes a deadlock

Proof

- if $n \geq 2$ the proof is trivial. Let us consider for $n > 2$.
- Proof by contradiction, so assuming $BFT$ is false and no node can apply a rule.
- $\forall v \in V \setminus \{r\} : \neg R_0(v) \land \neg R_1(v) = true$
- $BTF(v) \equiv L_v = L_{p_v} + 1 \land L_{p_v} = \min\{L_u | u \in N(v)\} = false$
  1. $L_v \neq L_{p_v} + 1$ and $\neg R_0(v) \rightarrow L_{p_v} = n$
  2. $L_{p_v} > \min\{L_u | u \in N(v)\}$ and $\neg R_1(v)$ contradiction

$\rightarrow$ (1) all the node $v$ have $L_v = n, L_r = 0$ so the neighbors of $r$ can applied $R_1$. 

Self-stabilization MPRI
Modifications of the rules for the Correctness

- $M_0 : L_v \leq L_{p_v} < n \rightarrow L_v = L_{p_v} + 1$
- $M_1 : L_v > L_{p_v} + 1 \rightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \rightarrow p_v = k_{id};$
  - with $k = \min\{L_u | u \in N(v)\}$ $k_{id} = \min\{id_u | u \in N(v) \land L_u = k\}$
Potential Function

- $F \equiv (F_1, F_2)$
- $F_1 = (t_2, t_3, \ldots, t_n)$ where $t_i$ is a number of nodes $v \in V$ such that $L_v = i$ and $L_v \leq L_{p_v}$ the node $v$ is called an $i -$ turn.
- $F_2 = \sum_{v \in V \setminus r}(L_v + L_{p_v})$
Examples

Init. Config.

F_1 = (1, 1, 0, 0)
F_2 = 23

F_1 = (0, 2, 0, 0)
F_2 = 27

F_1 = (0, 2, 0, 0)
F_2 = 25

F_1 = (0, 2, 0, 0)
F_2 = 23
Examples

\[ F_1 = (0, 2, 0, 0) \]
\[ F_2 = 22 \]

\[ F_1 = (0, 2, 0, 0) \]
\[ F_2 = 20 \]

\[ F_1 = (0, 1, 0, 0) \]
\[ F_2 = 21 \]

\[ F_1 = (0, 1, 0, 0) \]
\[ F_2 = 19 \]
Examples

\[ F_1 = (0, 1, 0, 0) \]
\[ F_2 = 17 \]

\[ F_1 = (0, 0, 0, 0) \]
\[ F_2 = 18 \]
Remark: To compute $F_1$ and $F_2$ only the tuple node parent of the node is considered.
**Lemma 2:** $F_1$ decreases each time when rule $M_0$ is applied.

Remember:

- $M_0 : L_v \leq L_{p_v} < n \rightarrow L_v = L_{p_v} + 1$
- $F_1 = (t_2, t_3, \ldots, t_n)$ where $t_i$ is a number of nodes $v \in V$ such that $L_v = i$ and $L_v \leq L_{p_v}$
Proof of lemma 2:

• Let $v$ be a $k$–*turn* node where $k = L_v(\gamma)$, so $v$ can execute $M_0$ in configuration $\gamma$ by definition of $M_0$ and $t$, after execution of node $v$, $v$ does not stay a $k$–*turn* node.

• Let now consider a node $u$ child of node $v$ such that

  ○ $L_u(\gamma) = L_v(\gamma) + 1$, so in $\gamma u$ is not a $(k + 1)$–*turn* node but becomes $(k + 1)$–*turn* after activation of $v$, but $k + 1 > k$ so in we obtain

  \[ F_1(\gamma) < F_1(\gamma') \]

  ○ $L_u(\gamma) \leq L_v(\gamma)$

  ○ $L_u(\gamma) > L_v + 1$
Lemma 3: $F_2$ decreases each time when rule $M_1$ or $M_2$ is applied.

Remember:

- $M_1 : L_v > L_{p_v} + 1 \rightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \rightarrow p_v = k_{id}$;
  - with $k = \min\{L_u | u \in N(v)\}$
  - $k_{id} = \min\{id_u | u \in N(v) \land L_u = k\}$
- $F_2 = \sum_{v \in V \setminus r} (L_v + L_{p_v})$
Proof of Lemma 3

If a node $v$ applied $M_1$ or $M_2$ then $L(v)$ decrease, the only way to increase $F_2$ is the use of $M_0$ but in this case thanks to lemma 2 $F_1$ decreases so $F$ decreases.
Lemma 4: $F_1$ does not increase each time when rule $M_1$ or $M_2$ is applied.

Remember:

- $M_1 : L_v > L_{p_v} + 1 \rightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \rightarrow p_v = k_{id}$
  - with $k = \min\{L_u \mid u \in N(v)\}$
  - $k_{id} = \min\{id_u \mid u \in N(v) \land L_u = k\}$
- $F_2 = \sum_{v \in V \setminus r} (L_v + L_{p_v})$
**Theorem:** Eventually, the system reaches a legitimate state.

**Proof:** Direct by lemmas 2, 3, 4.
Conclusion

- Silent self-stabilizing algorithm
- Distributed fair Scheduler
- Knowledge: $n$
- $O(\log_2 n)$ bits of memory per node
- Time complexity not provided

Question: Is it space optimal?
Main technique to break cycles

Distance to the root
Freeze: Technic to destroy cycle

Blin Tixeuil 2017

**Algorithm 4: Algorithm Freeze**

\[
\begin{align*}
\mathbb{R}_{\text{Error}} & : \quad \text{ErCycle}(v) \lor \text{ErST}(v) \\
\mathbb{R}_{\text{Froze}} & : \quad \neg\text{ErCycle}(v) \land \neg\text{ErST}(v) \land (\text{froz}_p = 1) \land (\text{froz}_v = 0) \\
\mathbb{R}_{\text{Prun}} & : \quad \neg\text{ErCycle}(v) \land \neg\text{ErST}(v) \land (\text{froz}_p = 1) \land (\text{froz}_v = 1) \land (\text{Ch}(v) = \emptyset)
\end{align*}
\]

\[\rightarrow \text{froz}_v := 1, \ p_v := \emptyset;\]
\[\rightarrow \text{froz}_v := 1;\]
\[\rightarrow \text{Reset}(v);\]

**Theorem 14.** Algorithm Freeze deletes a cycle or an impostor-rooted sub spanning tree in \(n\)-nodes graph in a silent self-stabilizing manner, assuming the state model, and a distributed unfair scheduler. Moreover, Algorithm Freeze uses \(O(1)\) bits of memory per node.

**Lemma 15.** Algorithm Freeze converges in \(O(n)\) steps.
Freeze : Technic to destroy cycle
Other techniques

Number of children
Other techniques

ID based algorithm: Unicity of the identity
Other technique: Non Silent, Semi-Uniform

Constant wave from the root
Fake subtree destruction
Space Optimality

Silent Self-Stabilizing Algorithm for Tree Construction
Proof Labeling Scheme vs Silent algorithm

Korman, Kutten, Peleg, 2007

- A proof-labeling scheme for a task is a pair such that:
- **Prover** assigns a certificate to every node.
- **Verifier** is a distributed algorithm such that

\[ V : \{ v \} \rightarrow \{ yes, no \} \]
Configurations

**Legal**

\((G, x)\) legal for \(T \Rightarrow P\) assign certificates such that \(V\) accepts at all nodes.

**Illegal**

\((G, x)\) illegal for \(T \Rightarrow P\) for every certificates, \(V\) must reject in at least one node.
PLS : Spanning Tree

- $T = \{(v, x(v)), v \in V\}$, $x(v)$ is the parent of $v$.
- $P(v) = (Root, dis)$
- $V(v) : (\forall u \in N(v) : (Root_u = Root_v)) \land (dis_v = dis_{p_v} + 1)$
Acyclicity

Theorem [Korman, Kutten, Peleg 2007]: Any PL for acyclicity requires certificate on $\Omega(\log_2 n)$ bits in $n$-node graph.
Proof

- Certificate $k$ bits with $k \leq \frac{1}{2} \log_2 n - 1$
- $m = n - 1$ edges
- edges $\leftrightarrow (a, b)$: $2k$ bits
- different pair $2^{2k} = 2^{\log_2 n - 2} = \frac{1}{4} 2^{\log_2 n} = \frac{n}{4}$
- $\exists e \neq e' \mid e \in E, e' \in E$ that correspond to the same pair
Silent Self-Stabilization vs. Proof-Labeling Scheme

Blin, Fraigniaud, Patt-Shamir 2014

<table>
<thead>
<tr>
<th></th>
<th>Size of registers</th>
<th>Number of rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower bound</td>
<td>$\Omega(\ell)$</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>$O(\ell + \log n)$</td>
<td>$O(n2^{n\ell})$</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>$O(n^2 + nk)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

- $n$-node networks, $k$-bit outputs, PLS of size at most $\ell$ bits

**Question:** Is there, for every task, a silent self-stabilizing algorithm for solving that task, that is simultaneously space-efficient and time-efficient?