Time Complexity of Link Reversal Routing
Talk is based on . . .

work with Bernadette Charron-Bost (CNRS, LIX), Jennifer L. Welch (Texas A&M University), and Josef Widder (TU Wien)

(Sirocco, ’11), (ACM Trans. on Algorithms, ’15).
Graph rewriting algorithms:
$G_0, G_1, G_2, \ldots$

Link reversal algorithms: on link directions

Applications:
- Routing algorithms
- Behavior of asynchronous circuits
- Behavior of Physarum (?)
The Routing Problem

Requirements:
- Computationally efficient algorithm.
- Adapt to failures/mobility fast.
Routing to a destination

Acyclic and destination oriented $\Rightarrow$
Routing is simple.
Routing to a destination

acyclic, destination oriented

acyclic
Full Reversal routing (Gafni and Bertsekas, 1981)

before failure
acyclic, destination oriented
Full Reversal routing (Gafni and Bertsekas, 1981)

before failure
acyclic, destination oriented

0

1 2 3

4 5 6

0

1 2 3

4 5 6

0

time 0
acyclic
Full Reversal routing (Gafni and Bertsekas, 1981)
Full Reversal routing (Gafni and Bertsekas, 1981)

before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic

time 2
acyclic
Full Reversal routing (Gafni and Bertsekas, 1981)

before failure
acyclic, destination oriented

1 2 3
4 5 6

0

1 2 3
4 5 6

0

1 2 3
4 5 6

0

1 2 3
4 5 6

0
Full Reversal routing (Gafni and Bertsekas, 1981)

before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic

time 2
acyclic

time 3
acyclic

time 4
acyclic, destination oriented
Full Reversal routing - alternative execution

before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic
Full Reversal routing - alternative execution

before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic

time 2
acyclic

before failure
acyclic, destination oriented
Full Reversal routing - alternative execution
Full Reversal routing - alternative execution

before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic

time 4
acyclic

time 3
acyclic

Full Reversal routing - alternative execution

before failure
acyclic, destination oriented
time 0
acyclic
time 1
acyclic
time 2
acyclic
time 3
time 4
acyclic
time 5
acyclic, destination oriented
Executions

Execution: $G_0, G_1, G_2, \ldots$

Two extremes:

- **Greedy** execution: all sinks in $G_i$ make a step
- **Lazy** execution(s): one sink in $G_i$ makes a step

... and many executions in between.
Distributed Algorithm

Local graph rewriting: \( \ldots, G_i, G_{i+1}, \ldots \)

Local mutex: no two sinks are neighbors

Asynchronous execution possible
Some Facts

Algorithm always terminates destination oriented.

At all steps, graph is acyclic.

Algorithm steps are commutative:
- Final graph is the same.
- Each node performs same number of steps.
Complexity

For an initial graph:

**work complexity** of node $i$
- number of steps made by node $i$ in *any* execution
- first exact expression established in (Busch et al., 2003)
- work complexity for more general algorithms in (Charron-Bost et al., 2009)

**time complexity** of node $i$
- time node $i$ makes its last step in *the greedy* execution
- problem: only approximate bounds (by work)
Complexity

For an initial graph:

**work complexity** of node $i$
- number of steps made by node $i$ in *any* execution
- first exact expression established in (Busch et al., 2003)
- work complexity for more general algorithms in (Charron-Bost et al., 2009)

**time complexity** of node $i$
- time node $i$ makes its last step in the greedy execution
- problem: only approximate bounds (by work)

Due to concurrency, understanding of work complexity is not sufficient to get exact time complexity.
A dynamical system

System state $\vec{W}(t) = (\vec{W}_0(t), \vec{W}_1(t), \ldots, \vec{W}_N(t))$ at time $t$.

$\vec{W}_i(t)$ . . . number of steps of node $i$ up to time $t$. 
A dynamical system

System state $\vec{W}(t) = \left( \vec{W}_0(t), \vec{W}_1(t), \ldots, \vec{W}_N(t) \right)$ at time $t$.

$\vec{W}_i(t)$ ... number of steps of node $i$ up to time $t$.

$\vec{W}_i(0) = 0$ for all nodes $i$

$\vec{W}_0(t) = 0$ for all times $t$

Looking for a function $F$ such that

$\vec{W}(t) = F \left( \vec{W}(t - 1) \right)$
The influence of links

Proposition

Between two consecutive steps by a node $i$, each neighbor of $i$ takes exactly one step.

Proposition

In any FR execution in which $i$ takes a step, before the first step by $i$, each node $j \in \text{In}_i$ takes no step and each node $k \in \text{Out}_i$ takes exactly one step.
The influence of links

Proposition

Between two consecutive steps by a node $i$, each neighbor of $i$ takes exactly one step.

Proposition

In any FR execution in which $i$ takes a step, before the first step by $i$, each node $j \in \text{In}_i$ takes no step and each node $k \in \text{Out}_i$ takes exactly one step.

$\Rightarrow$ Links induce strict alternation.
Recurrence

Strict alternation ⇒
\[ \tilde{W}_i(t) \leq \tilde{W}_j(t - 1) + 1 \quad \text{for } j \in \text{In}_i \]
\[ \tilde{W}_i(t) \leq \tilde{W}_k(t - 1) + 0 \quad \text{for } k \in \text{Out}_i \]

Theorem

In a greedy FR execution, for any node \( i \) other than 0 and any \( t \geq 1 \),

\[ \tilde{W}_i(t) \leq \min \left\{ \tilde{W}_j(t - 1) + 1, \tilde{W}_k(t - 1) + 0 : j \in \text{In}_i, \ k \in \text{Out}_i \right\} . \]
Recurrence

Strict alternation $\Rightarrow$

$\vec{W}_i(t) \leq \vec{W}_j(t - 1) + 1$ for $j \in \text{In}_i$

$\vec{W}_i(t) \leq \vec{W}_k(t - 1) + 0$ for $k \in \text{Out}_i$

Theorem

In a greedy FR execution, for any node $i$ other than 0 and any $t \geq 1$,

$\vec{W}_i(t) = \min \left\{ \vec{W}_j(t - 1) + 1, \, \vec{W}_k(t - 1) + 0 : j \in \text{In}_i, \, k \in \text{Out}_i \right\}$. 
Recurrence

\[ \tilde{W}_i(t) = \min \left\{ \tilde{W}_j(t-1) + 1, \tilde{W}_k(t-1) + 0 : j \in In_i, \ k \in Out_i \right\}. \]
Recurrence

\[ \vec{W}_i(t) = \min \left\{ \vec{W}_j(t-1) + 1, \vec{W}_k(t-1) + 0 : j \in \text{ln}_i, k \in \text{out}_i \right\}. \]

\( \Downarrow \) \quad \text{min-plus algebra}

\[ \vec{W}_i(t) = \sum_{j \in \text{ln}_i} \vec{W}_j(t-1) \otimes 1 + \sum_{k \in \text{out}_i} \vec{W}_k(t-1) \otimes 0. \]
Recurrence

\[ \vec{W}_i(t) = \min \left\{ \vec{W}_j(t - 1) + 1, \vec{W}_k(t - 1) + 0 : j \in \text{In}_i, \ k \in \text{Out}_i \right\}. \]

\( \vec{W}_i(t) = \sum_{j \in \text{In}_i} \vec{W}_j(t - 1) \otimes 1 + \sum_{k \in \text{Out}_i} \vec{W}_k(t - 1) \otimes 0. \)

\[ \vec{W}(t) = A \otimes \vec{W}(t - 1). \]
Recurrence

\[ \tilde{W}_i(t) = \min \left\{ \tilde{W}_j(t-1) + 1, \tilde{W}_k(t-1) + 0 : j \in \text{In}_i, k \in \text{Out}_i \right\}. \]

min-plus algebra

\[ \tilde{W}_i(t) = \sum_{j \in \text{In}_i} \tilde{W}_j(t-1) \otimes 1 + \sum_{k \in \text{Out}_i} \tilde{W}_k(t-1) \otimes 0. \]

matrix form

\[ \tilde{W}(t) = A \otimes \tilde{W}(t-1). \]

\[ \tilde{W}(t) = A^t \otimes \tilde{0}. \]
Recurrence

\[ \tilde{W}_i(t) = \min \left\{ \tilde{W}_j(t-1) + 1, \tilde{W}_k(t-1) : j \in \text{In}_i, k \in \text{Out}_i \right\}. \]

\[ \Rightarrow \text{min-plus algebra} \]

\[ \tilde{W}_i(t) = \sum_{j \in \text{In}_i} \tilde{W}_j(t-1) \otimes 1 + \sum_{k \in \text{Out}_i} \tilde{W}_k(t-1) \otimes 0. \]

\[ \Rightarrow \text{matrix form} \]

\[ \tilde{W}(t) = A \otimes \tilde{W}(t-1). \]

\[ \tilde{W}(t) = A^t \otimes \tilde{0}. \]

\[ \Rightarrow \text{discrete linear dynamical system in min-plus algebra} \]
Reduction to graph properties

- computing: $\tilde{W}(t) = A^t \otimes \vec{0} = (A^{t/2})^2 \otimes \vec{0}$
Reduction to graph properties

- computing: \( \tilde{W}(t) = A^t \otimes \vec{0} = (A^{t/2})^2 \otimes \vec{0} \)

- but:
  matrix \( A \) quite similar to initial graph’s adjacency matrix

- \( W_i(t) = \min \{ \text{weight}(p) : p \text{ is path to } i \text{ of length } t \} \)
Reduction to graph properties

- computing:  \( \vec{W}(t) = A^t \otimes \vec{0} = (A^{t/2})^2 \otimes \vec{0} \)

- but:
  matrix \( A \) quite similar to initial graph’s adjacency matrix

\( W_i(t) = \min \{ \text{weight}(p) : p \text{ is path to } i \text{ of length } t \} \)

\( W_i(t) = \min \{ r(c) : c \text{ is chain to } i \text{ of length } t \} \)
Simple work complexity proof

\[ w_i = \lim_{t \to \infty} W_i(t) = \lim_{t \to \infty} \min \{ \text{weight}(p) : p \text{ is path to } i \text{ of length } t \} \]

▶ A path long enough to reach 0, will loop there for free.

▶ \( w_i = \min \{ \text{weight}(p) : p \text{ is path from 0 to } i \} \)
Simple work complexity proof

\[ w_i = \lim_{t \to \infty} W_i(t) = \lim_{t \to \infty} \min \{ \text{weight}(p) : p \text{ is path to } i \text{ of length } t \} \]

- A path long enough to reach 0, will loop there for free.
- \( w_i = \min \{ \text{weight}(p) : p \text{ is path from 0 to } i \} \)
- \( w_i = \min \{ r(c) : c \text{ is chain from 0 to } i \} \)
Can we say something about time complexity?

Dual of $W_i(t)$ is $T_i(w)$

- $W_i(T_i(w)) = w$  $\Rightarrow$  $W_i(\theta_i) = w_i$
- $W_i(T_i(w) - 1) = w - 1$  $\Rightarrow$  $W_i(\theta_i - 1) = w_i - 1$
Can we say something about time complexity?

Dual of $W_i(t)$ is $T_i(w)$

$\quad\quad\quad\quad\quad\quad\quad W_i(T_i(w)) = w \quad \Rightarrow \quad W_i(\theta_i) = w_i$

$\quad\quad\quad\quad\quad\quad\quad W_i(T_i(w) - 1) = w - 1 \quad \Rightarrow \quad W_i(\theta_i - 1) = w_i - 1$

Theorem

The termination time $\theta_i$ of any node $i$ that takes a step is equal to

$$\theta_i = \max \{ \text{length}(c) : c \text{ is chain to } i \text{ with } r(c) = w_i - 1 \} + 1.$$
Can we say something about time complexity?

Dual of $W_i(t)$ is $T_i(w)$

$\blacktriangleright$ $W_i(T_i(w)) = w$ \quad $\Rightarrow$ \quad $W_i(\theta_i) = w_i$

$\blacktriangleright$ $W_i(T_i(w) - 1) = w - 1$ \quad $\Rightarrow$ \quad $W_i(\theta_i - 1) = w_i - 1$

\[\downarrow\]

**Theorem**

The termination time $\theta_i$ of any node $i$ that takes a step is equal to

$$\theta_i = \max \{\text{length}(c) : c \text{ is chain to } i \text{ with } r(c) = w_i - 1\} + 1.$$ 

\[\begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
0
\end{array}\]

\[\begin{array}{c}
w_3 = 2 \\
\theta_3 = 4
\end{array}\]
Full Reversal routing (Gafni and Bertsekas, 1981)

Before failure
acyclic, destination oriented

time 0
acyclic

time 1
acyclic

time 4
acyclic, destination oriented

time 3
acyclic

time 2
acyclic
Immediate results

Corollary

*There is an $N$ node graph family with FR time complexities scaled from* $\Theta(N)$ *to* $\Theta(N^2)$

Corollary

*In any tree with* $N + 1$ *nodes, the FR time complexity is at most equal to* $2N - 1$.

Corollary

*In general FR time complexity is unstable. Adding one link can increase it from* $\Theta(N)$ *to* $\Theta(N^2)$.
FR only for routing?

Distributed algorithm where no two sinks are neighbors.

⇒ Distributed scheduling with local mutex.
FR only for routing?

Distributed algorithm where no two sinks are neighbors.

⇒ Distributed scheduling with local mutex.

Work until termination: \( w_i = \min \{ r(c) : c \text{ is chain from 0 to } i \} \).

⇒ What if we remove the special node 0?
FR only for routing?

Distributed algorithm where no two sinks are neighbors.

⇒ Distributed scheduling with local mutex.

Work until termination: \( w_i = \min \{ r(c) : c \text{ is chain from 0 to } i \} \).

⇒ What if we remove the special node 0?

Will run forever.
Distributed scheduling

Scheduling frequency $\lim_{t \to \infty} \frac{W_i(t)}{t}$
Distributed scheduling

Again a dynamical system \( \vec{W}(t) \)

\[ \vec{W}_i(t) = \min \{ r(c) : c \text{ is chain to } i \text{ of length } t \} \]
Distributed scheduling

Again a dynamical system $\vec{W}(t)$

$\vec{W}_i(t) = \min \{ r(c) : c \text{ is chain to } i \text{ of length } t \}$

⇒ eventually $c$ will find a cycle $\gamma$ with minimal $r(\gamma)/\ell(\gamma)$. 
Distributed scheduling

Theorem
\[ \lim_{t \to \infty} \frac{W_i(t)}{t} = \min \left\{ \frac{r(\gamma)}{\ell(\gamma)} : \gamma \text{ is cycle} \right\} \]

\[ \lim_{t \to \infty} \frac{W_i(t)}{t} = 1/4 \]
Beyond Full Reversal

LR $\equiv$ Reverse only some links.

Proof idea:
- Not necessarily linear in $N$-dimensional system
- But: simulate nodes with one or two nodes (transformed graph).
- Relate chains in transformed graph to chain in original graph.

$\Rightarrow$ analogous results with other potentials $\Pi$.

Theorem

The termination time $\theta_i$ of any node $i$ that takes a step is equal to

$$\theta_i = \max \{ \text{length}(c) : c \text{ is chain to } i \text{ with } \Pi(c) = w_i - 1 \} + 1.$$