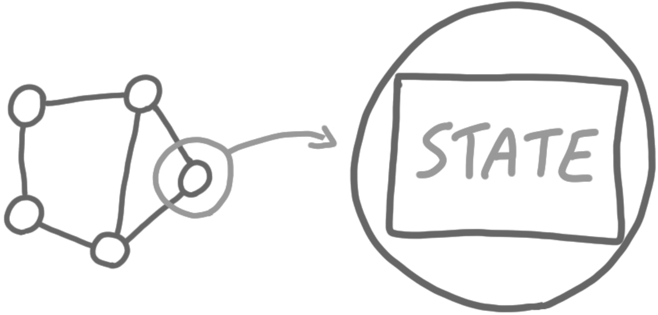
Silent MST approximation for tiny memory

Lélia Blin, Swan Dubois and Laurent Feuilloley

SSS 2020 · 18^{th} Novembrer 2020 · Virtual

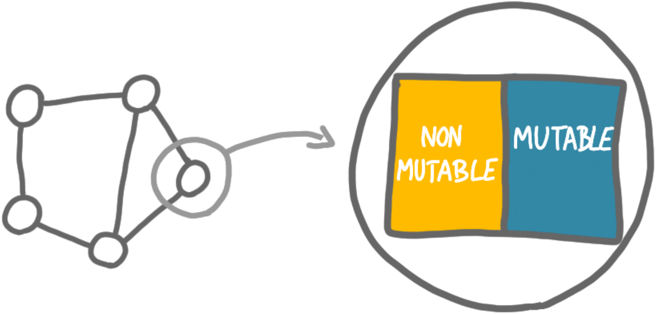
The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.



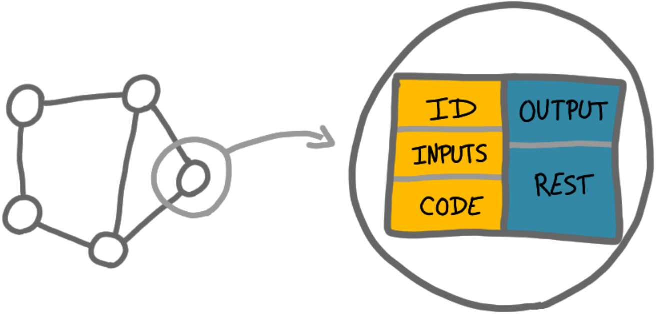
The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.



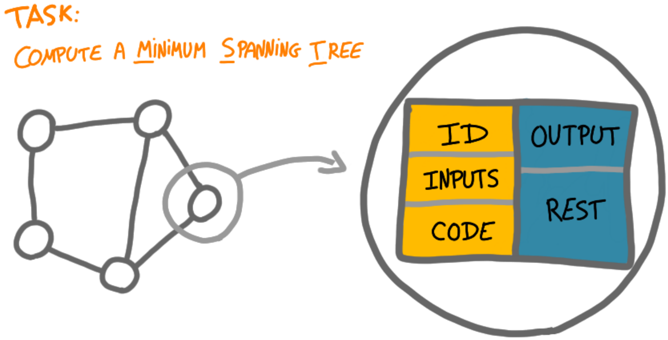
The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.
2. The non-mutable memory contains the ID, inputs and code.
3. At the end the mutable memory should contain the output.



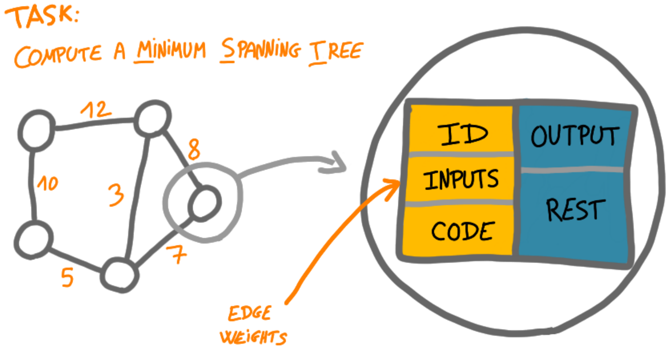
The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.
2. The non-mutable memory contains the ID, inputs and code.
3. At the end the mutable memory should contain the output.



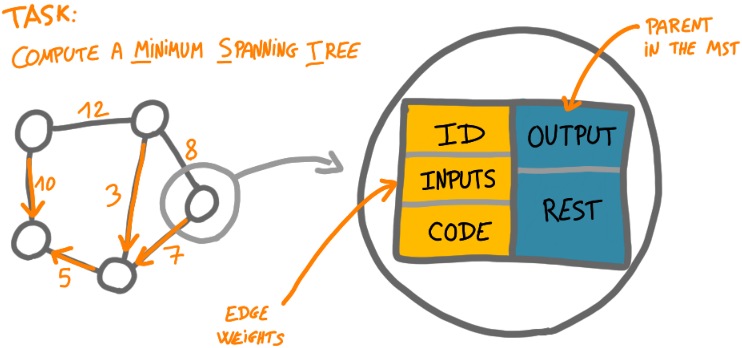
The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.
2. The non-mutable memory contains the ID, inputs and code.
3. At the end the mutable memory should contain the output.

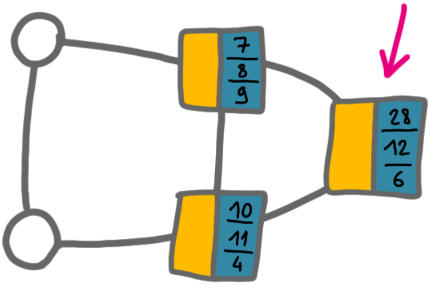


The state model and MST

1. Computation on a graph and every node has a state, with two parts : the mutable and the non-mutable memory.
2. The non-mutable memory contains the ID, inputs and code.
3. At the end the mutable memory should contain the output.

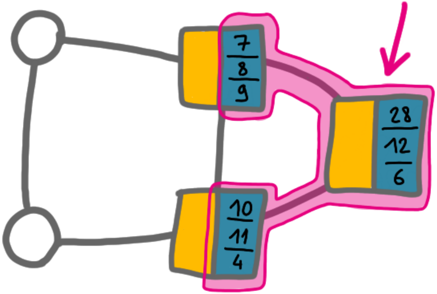


Computation in the state model



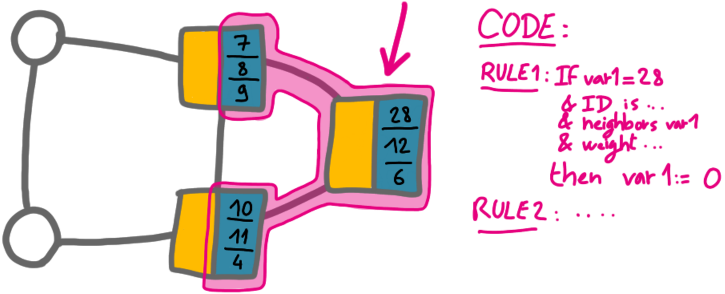
Computation in the state model

- Local view, modeling messages.



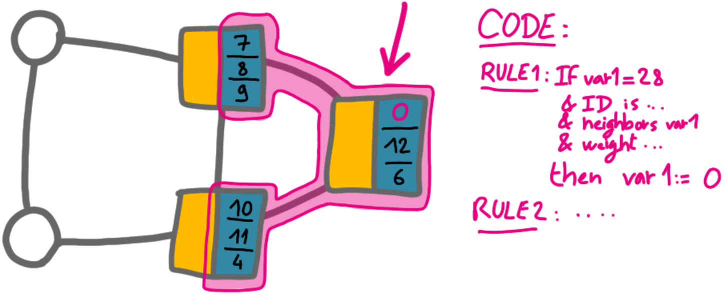
Computation in the state model

- Local view, modeling messages.
- Local "if" rules, activated/non-activated.



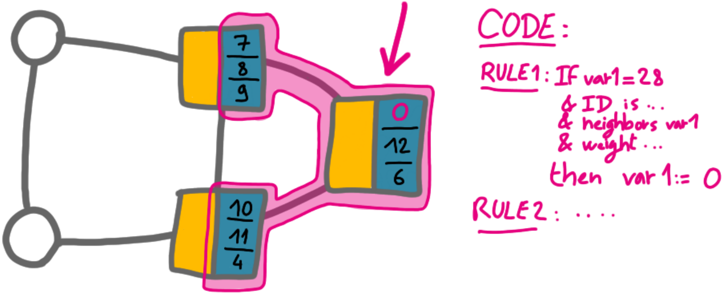
Computation in the state model

- Local view, modeling messages.
- Local "if" rules, activated/non-activated.



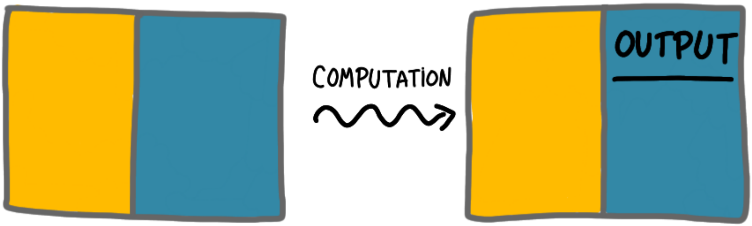
Computation in the state model

- Local view, modeling messages.
- Local "if" rules, activated/non-activated.
- Adversary scheduler : chooses the (active) nodes taking steps.



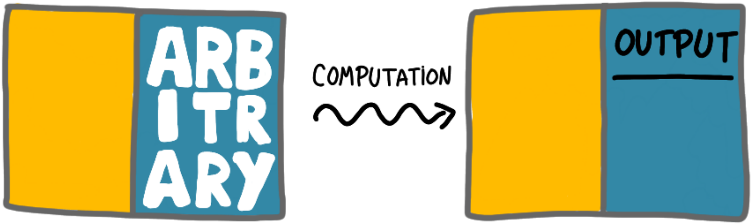
Silent self-stabilization

- Fault-free setting vs. self-stabilizing setting



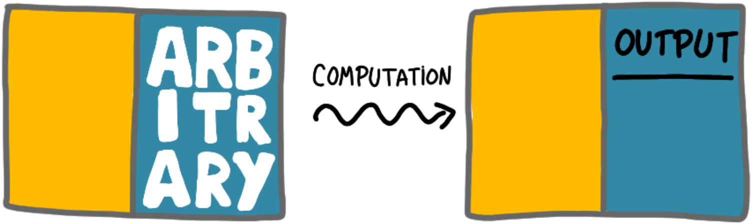
Silent self-stabilization

- Fault-free setting vs. self-stabilizing setting



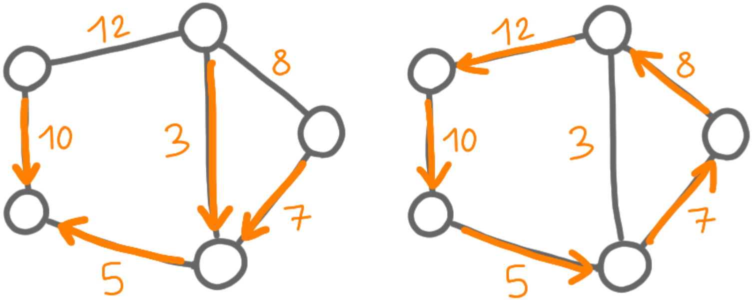
Silent self-stabilization

- Fault-free setting vs. self-stabilizing setting
- Silent self-stabilization : at the end, no activated nodes



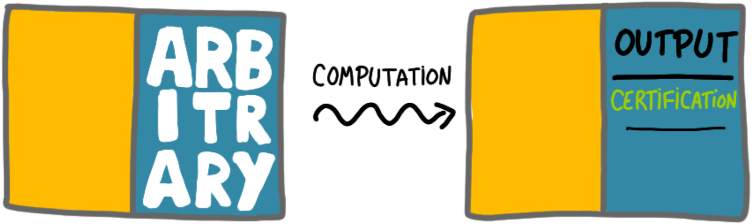
Silent self-stabilization

- Fault-free setting vs. self-stabilizing setting
- Silent self-stabilization : at the end, no activated nodes
- The output is not enough (for most tasks).



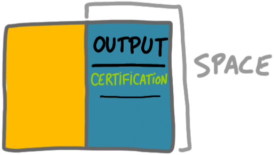
Silent self-stabilization

- Fault-free setting vs. self-stabilizing setting
- Silent self-stabilization : at the end, no activated nodes
- The output is not enough (for most tasks).
- Certification has been studied on its own (*Proof-labeling schemes* Korman, Kutten and Peleg)



Certification size and MST

- Focus on the space (keeping polynomial time).
- $\rightarrow$ minimize size of certification and do not use more space.



Theorem [Korman, Kutten] : The optimal certification size for minimum spanning tree is $\Theta(\log n \times s)$, when weights are encoded on s bits.
$\rightarrow$ For weights in a $poly(n)$ range $s = \log n$, and the optimal size is $\Theta(\log^2 n)$.

Our results

Theorem 1 : A (full) self-stabilizing MST algorithm with optimal space $O(\log n \times s)$.

Proof : First : build a non-certified MST. Second : build the optimal certification on top.

Our results

Theorem 1 : A (full) self-stabilizing MST algorithm with optimal space $O(\log n \times s)$.

Proof : First : build a non-certified MST. Second : build the optimal certification on top.

What if we cannot afford $\log^2 n$ bits ? Can we still get something useful ? $\rightarrow$ Yes with approximation.

Our results

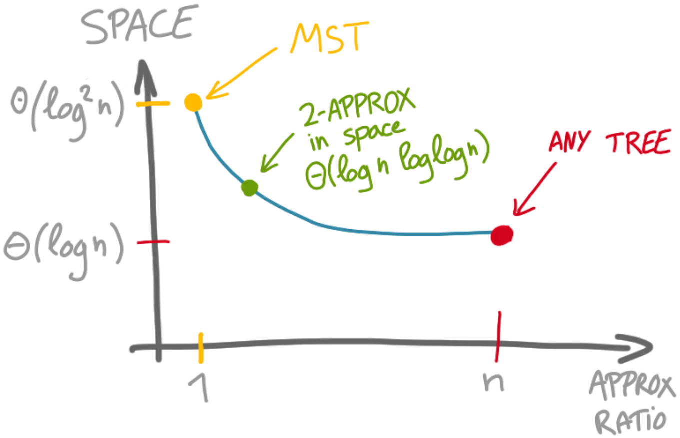
Theorem 1 : A (full) self-stabilizing MST algorithm with optimal space $O(\log n \times s)$.

Proof : First : build a non-certified MST. Second : build the optimal certification on top.

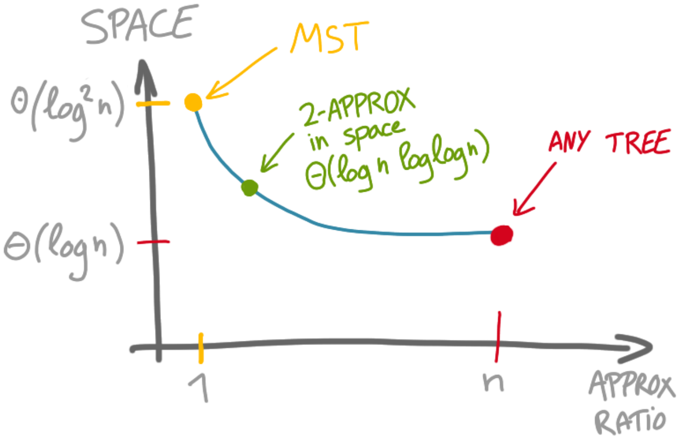
What if we cannot afford $\log^2 n$ bits ? Can we still get something useful ? $\rightarrow$ Yes with approximation.

Theorem 2 : We can parameterize the algorithm such that there is tradeoff between space complexity and quality of the solution.

Approximation and trade-off



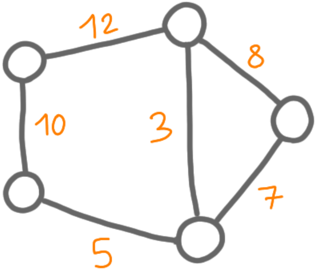
Approximation and trade-off



Approximation in certification : Censor-Hillel, Paz and Perry (2017), and Emek and Gil (2020).

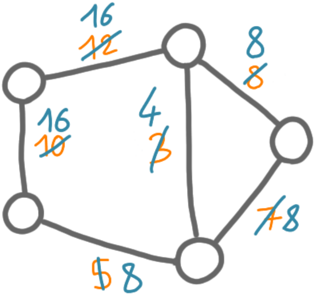
Example : 2-approximation

- Round every weight to the next power of 2.



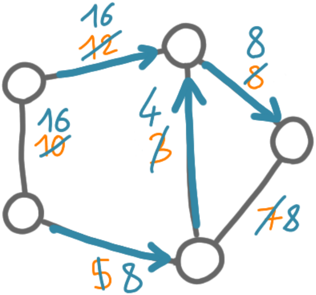
Example : 2-approximation

- Round every weight to the next power of 2.



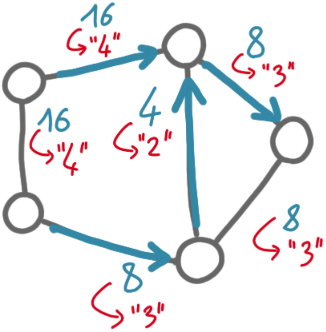
Example : 2-approximation

- Round every weight to the next power of 2.
- An MST on these new weights is a 2 -approximation.



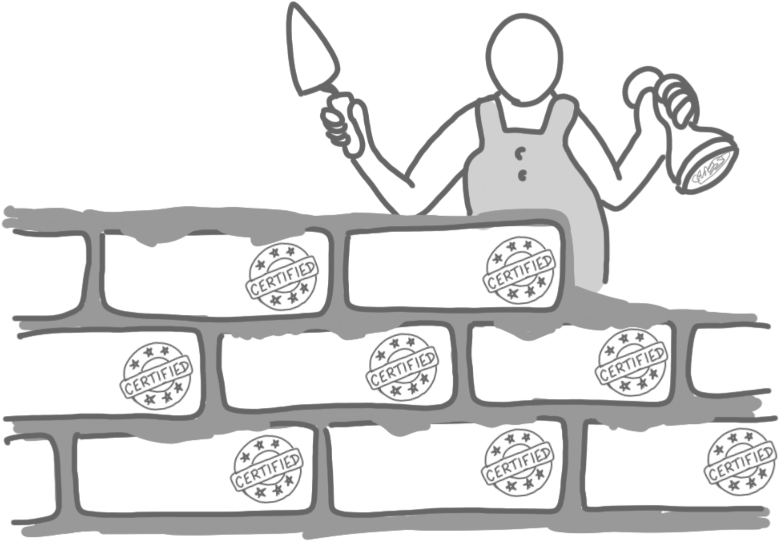
Example : 2-approximation

- Round every weight to the next power of 2.
- An MST on these new weights is a 2 -approximation.
- Weights can be encoded in a compact way : 2^p encoded as p.
- This reduces the weight exponentially $\log n \to \log\log n$.
- Using Theorem 1, we get a 2-approx in space $\log n \log\log n$.

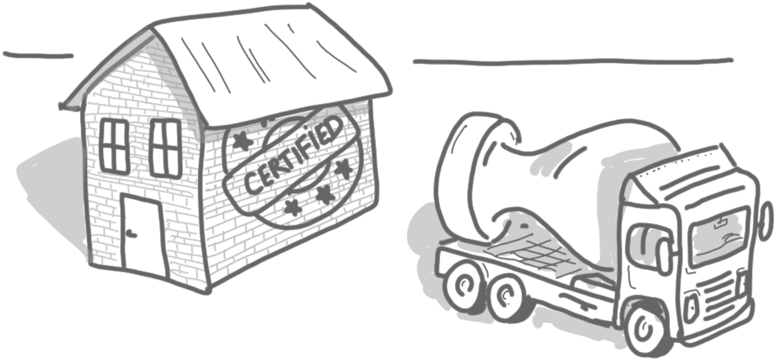


To finish : build vs certify

To finish : build vs certify



To finish : build vs certify



Thanks for your attention !