

Locally optimal load balancing

Laurent Feuilloley
Université Paris Diderot

November 23, 2015 · Journées du GT CoA

Joint work with :



Jukka Suomela



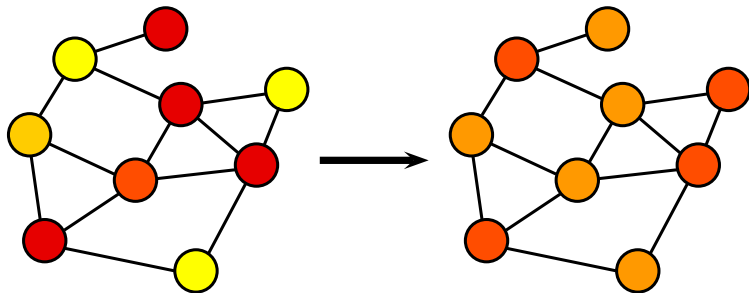
Juho Hirvonen

The paper appeared in DISC 2015, and is available on Arxiv.

Load Balancing

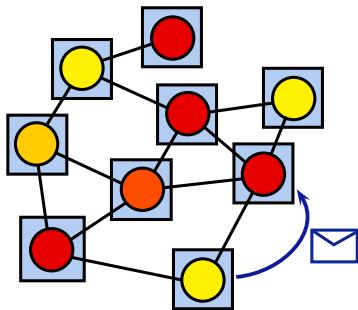
Input : a network and loads

Task : balance the load.



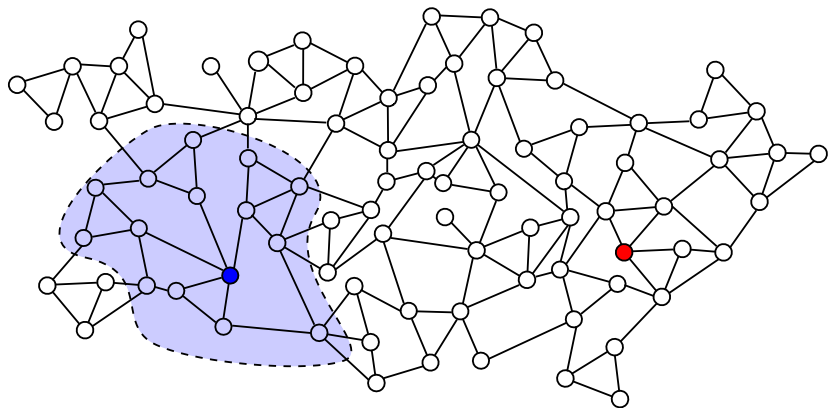
Network distributed computing

The input graph is the communication graph.



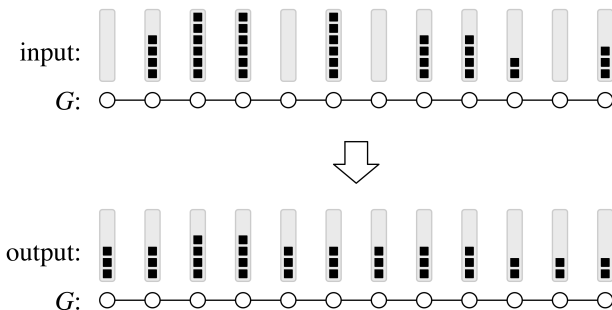
Locality

Limited number of rounds of communication
is equivalent to
Local vision of the graph.



Locally optimal load balancing

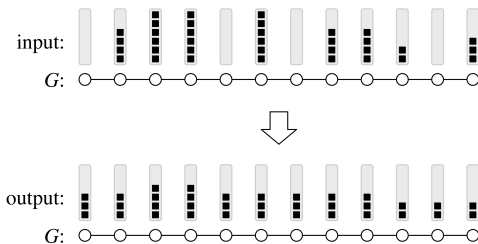
Locally optimal : two adjacent nodes have a load difference of at most one.



The distribution is *smooth*.

Three remarks

- Two versions : fractional and integral.
- For this talk the graph is a line.
- Maximum load L



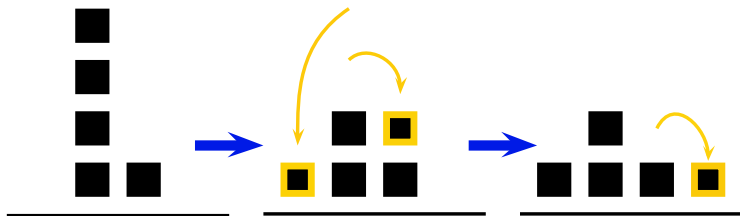
What we want

We look for algorithms that are :

- deterministic
- very local : complexity independent of the number of nodes n .

Algorithm 1 : Match-and-balance

Until it is locally optimal:
 Balance the load between neighbours ;



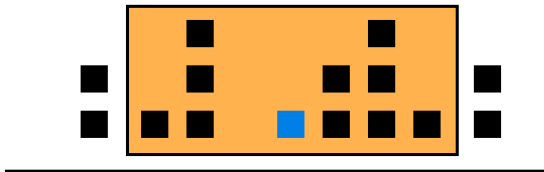
Algorithm 1 : Match-and-balance

How to balance ?

→ Actually all the balancing policies are slow.

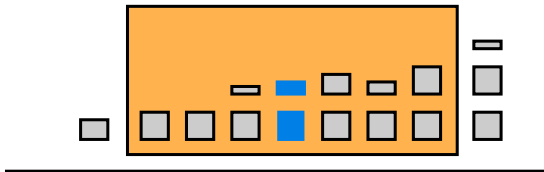
Algorithm 2 : Sliding window

Smoothing
with an *averaging sliding window*.



Algorithm 2 : Sliding window

Smoothing
with an *averaging sliding window*.



Algorithm 2 : Sliding window

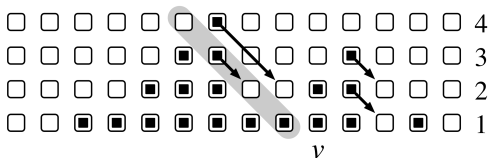
Smoothing


with an *averaging sliding window*.

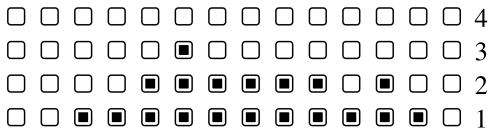
- Works only in the fractional setting
- Is *oblivious*, and then cannot be generalised to general graphs.

Algorithm 3 : Push algorithm

Push along the descending diagonals,
In one direction, then in the other.



1-push 



Algorithm 3 : Push algorithm

How to simulate a global orientation ?

→ At every node :

Divide the loads into two parts

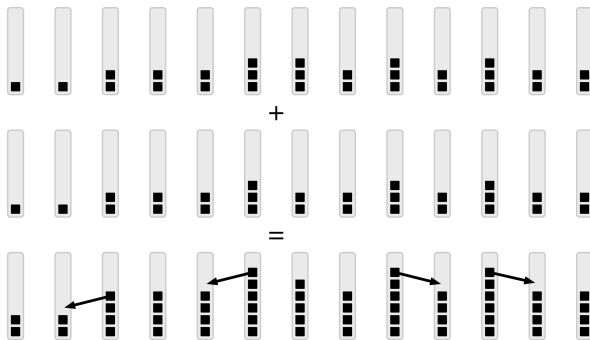
Manage two parallel instances

(Tokens $0 \rightarrow 1$ and $1 \rightarrow 0$)

Recombine the two instances.

Algorithm 3 : Push algorithm

Recombining 3-stable configurations

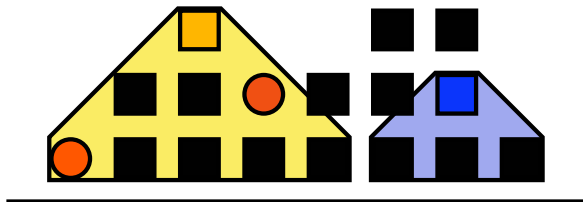


Algorithm 3 : Push algorithm

- Optimal complexity on the path : $O(L)$.
- But difficult to generalise to bounded degree graphs.

Algorithm 4 : Cone algorithm

The notion of stable cone.



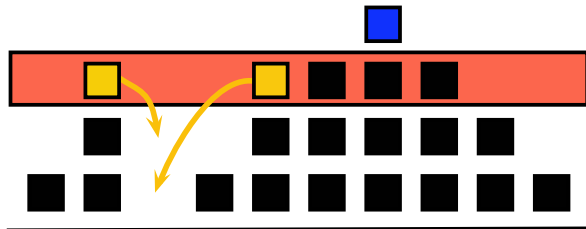
Algorithm 4 : Cone algorithm

For level i from L down to 0:

For every token at the level i :

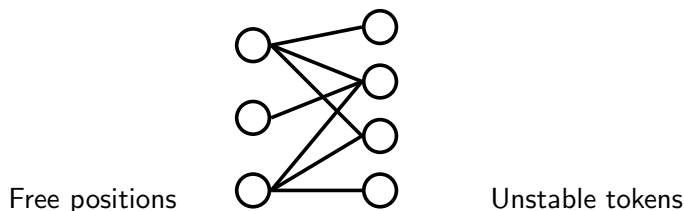
If it is stable, do not move it.

If it is unstable, try to place it in a free position of its cone.



Algorithm 4 : Cone algorithm

Try to place it in a free position
→ **bipartite maximal matching** (BMM) in a graph of degree order of $\Theta(L\Delta^L)$.



Algorithm 4 : Cone algorithm

BMM with maximum degree d :

- Exact complexity unknown
- In the discrete case : $O(d)$ (tight ?) .
- In the fractional case : approximate BMM in $O(\log(d))$

Algorithm 4 : Cone algorithm

For locally load balancing :

→ $O(L^3 \log(\Delta))$ in the fractional case

→ $O(\Delta^L L^3)$ in the discrete case

Wrap-up and questions

- For the line : $\Theta(L)$.
- For bounded degree graphs :
 - fractional : $\text{poly}(\Delta, L)$
 - discrete : $\Delta^L \text{poly}(L)$, optimal?
- Is BMM really in $\Theta(d)$?