Locally Optimal Load Balancing

Laurent Feuilloley (laurent.feuilloley@ens-cachan.fr)

ENS Cachan

Aalto University

This poster

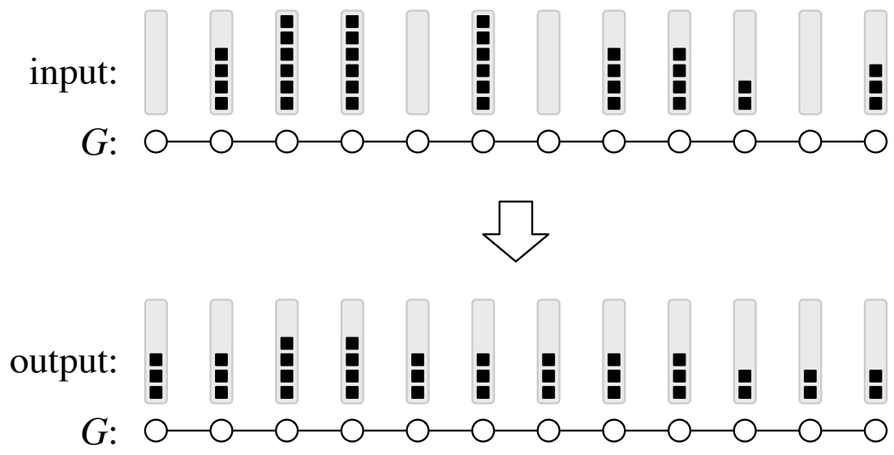
This poster presents part of the results of my ARPE. This work has been done during a visit at Aalto University in Helsinki, in automn-winter 2014, under the supervision of **Jukka Suomela** and with **Juho Hirvonen**. It be will presented at the conference DISC in October 2015. We present the ideas of the paper, avoiding the technicalities.

Context

Load balancing is a well-studied subject, that paradoxically had not been investigated from the point of view of local distributed computing. This work present the first results on this matter.

Problem

Computational problem :
Input : A line of processors, with their loads represented by tokens. The maximum load is L.
Task : Balance the load such that two adjacent processors have a load difference of at most one token.



Model and Complexity measure

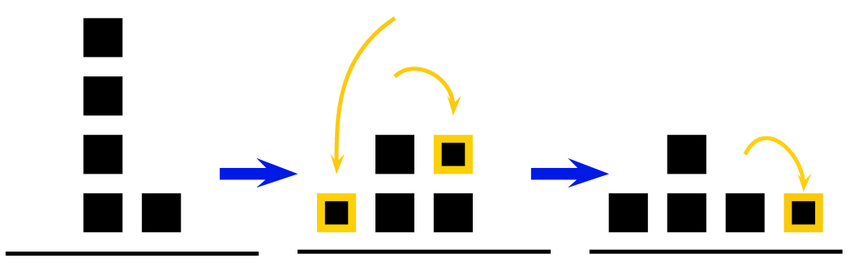
We use the local model of distributed computing.

- Every processor is a computing machine.
- There are rounds of communication, and the messages travel between the processors with the speed of one edge per round.
- The edges have port numbers.

The measure of the complexity (that we want to minimise) is the number of rounds of communication needed to balance the load. Basically we want the computation to be as local as possible.

Match-and-balance Algorithms

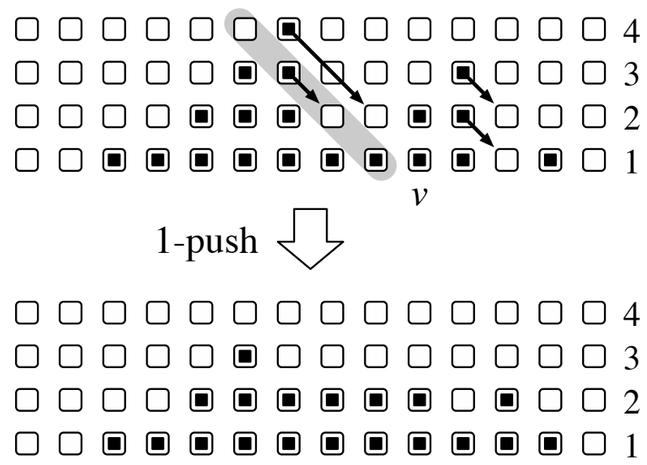
The most natural idea of algorithm is to balance the load between neighbours repeatedly.



We show in the paper that this strategy is not very efficient in general.

Push Algorithm

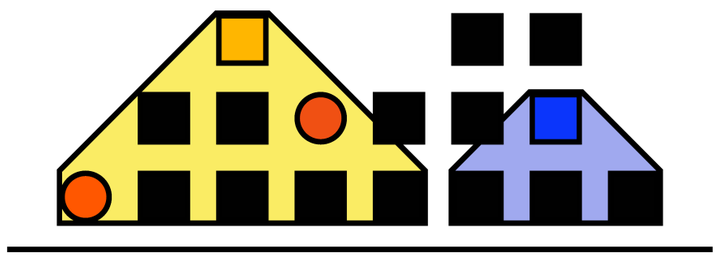
We propose an algorithm with complexity $O(L)$. The principle is to *push* the tokens along the descending diagonals. We push to the right (see the figure below), and then to the left.



The algorithm is actually more complex, because we need to simulate a global orientation.

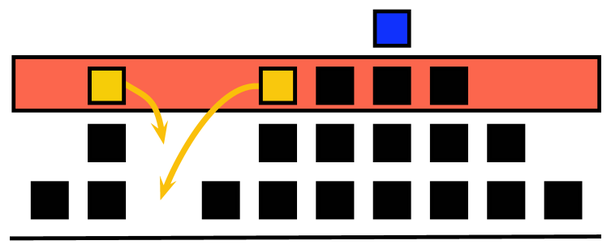
Cone Algorithm

It is difficult to generalise the push algorithm to larger graph classes. Therefore we build a new algorithm. We say that a token is stable if the positions in its downward cone are full.



The disks are empty positions, the yellow token is unstable, the blue one is stable.

Algorithm :
For a level i from L down to 0:
  For every token at the level i :
    If it is stable, do not move it.
    It it is unstable, try to place it in a free position of its downward cone.



The red line is the level i, the yellow tokens are the ones that are moving, the blue token is stable at its final position.

Placing a token in a free position requires coordination, which boils down to bipartite maximal matching. The complexity of this problem in general graphs is a long-standing open problem.

Note: the two black and white pictures were designed by Jukka Suomela.