On the Load Balancing Problem

Gabriel Semanišin

joint work with

František Galčík and Ján Katrenič

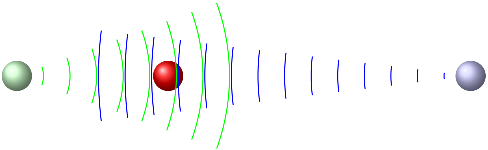
Institute of Computer Science, Faculty of Science, P.J. Šafárik University in Košice, Slovakia

MMCP 2017 07.07.2017

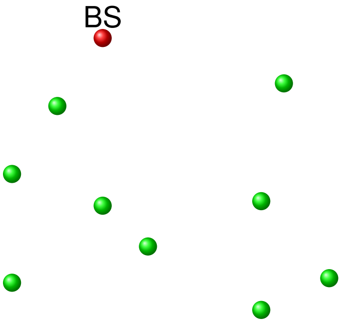
Wireless sensor networks (WSN)

WSN is a special type of ad-hoc wireless networks such that its nodes are devices with embedded

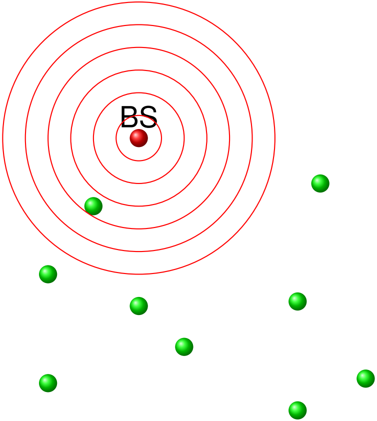
- microcontroller,
- sensor,
- FM radio and
- power source.



Building WSN

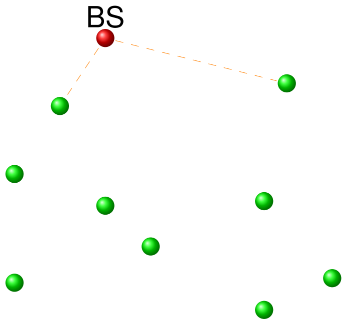


Building WSN



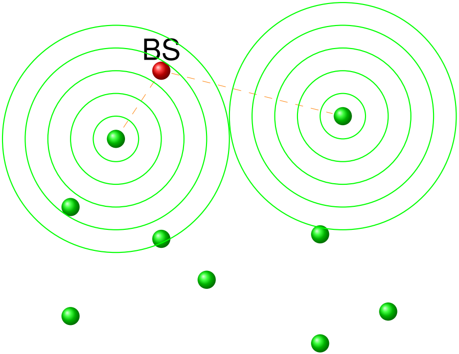
1. Building reachability graph

Building WSN



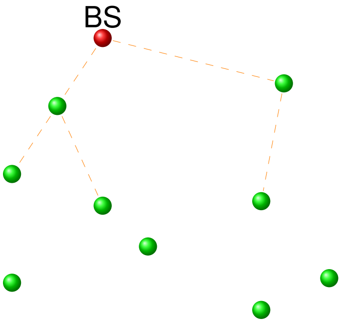
1. Building reachability graph

Building WSN



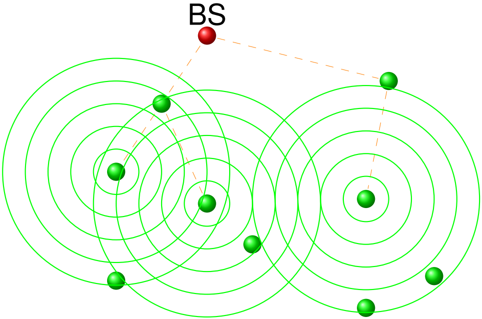
1. Building reachability graph

Building WSN



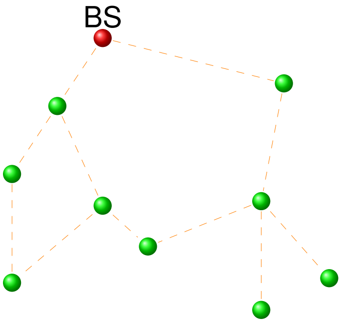
1. Building reachability graph

Building WSN



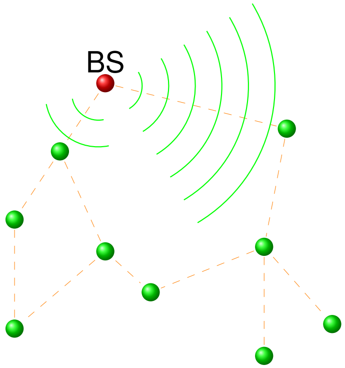
1. Building reachability graph

Building WSN



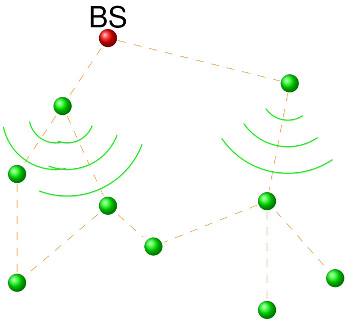
1. Building reachability graph

Building WSN



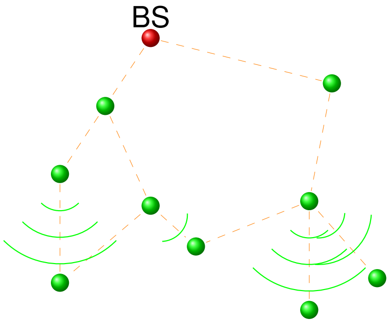
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



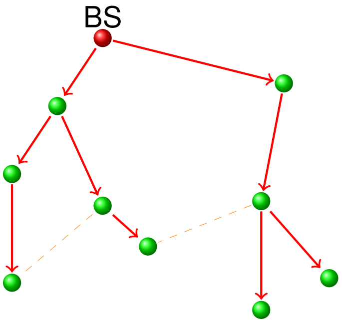
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



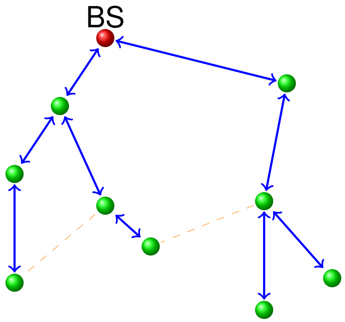
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



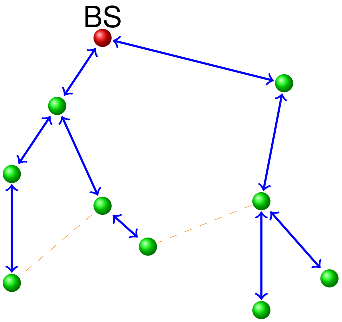
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



1. Building reachability graph
2. Building BFS tree, establishing connections
3. Establishing secure connections

Building WSN

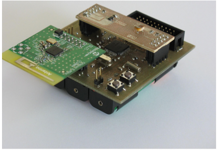


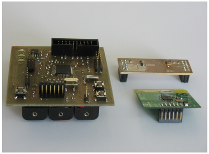
1. Building reachability graph
2. Building BFS tree, establishing connections
3. Establishing secure connections
4. Secure communication

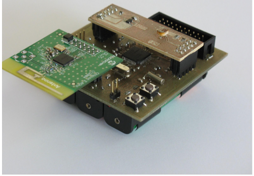
WSN based on CDMA technology

WSN based on CDMA technology

- consists of node that **are able to communicate each other with respect to their physical limitations** and **mutual distance**,
- the sink of the network (base station) **has relatively large computational capabilities and energy sources**,
- the number of communication channels available at the sensors is limited (say 16).







k-Path Vertex Cover Problem

(hereditary)

Generalised Scheduling Problem

(induced hereditary reductions)

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

Bad news

The problem of finding spanning tree with bounded degree is NP-complete.

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

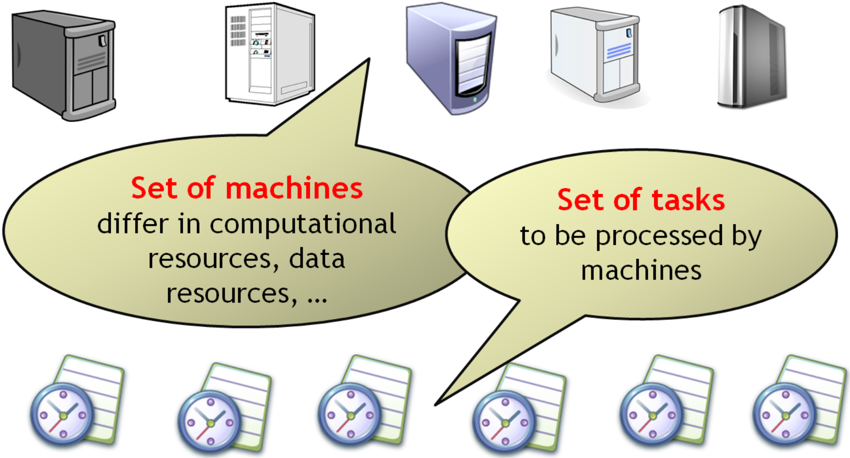
Bad news

The problem of finding spanning tree with bounded degree is NP-complete.

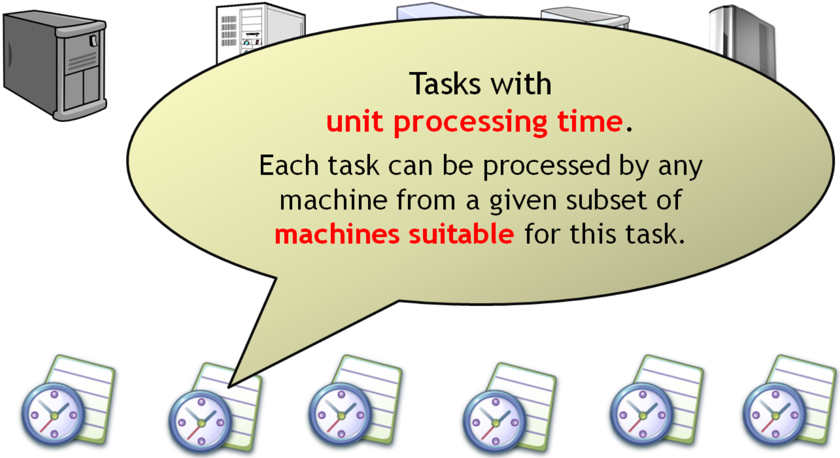
Good news

We can restrict our consideration to a bipartite graph that is formed by two layers in BSF-tree.

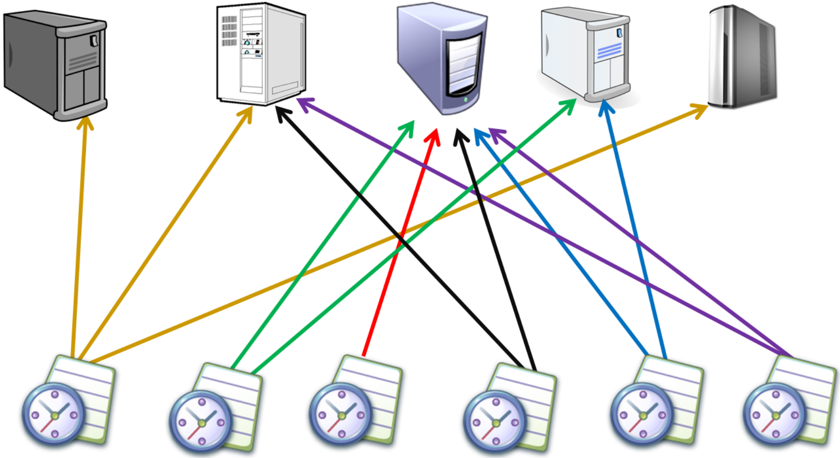
Load balancing problem



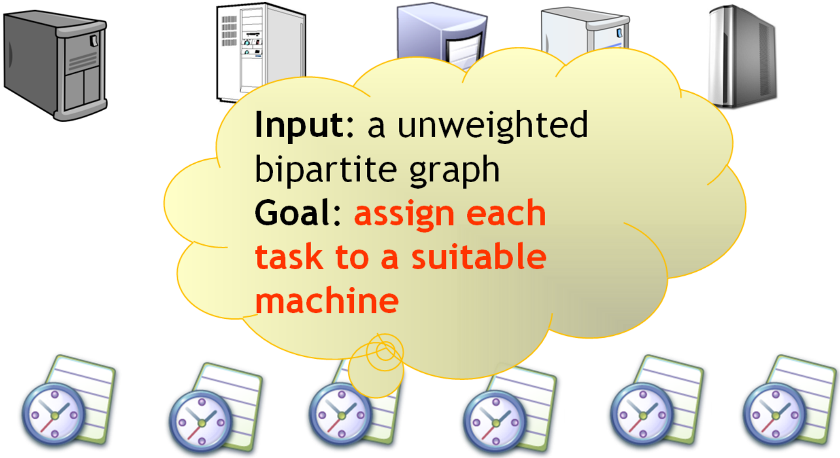
Load balancing problem



Load balancing problem



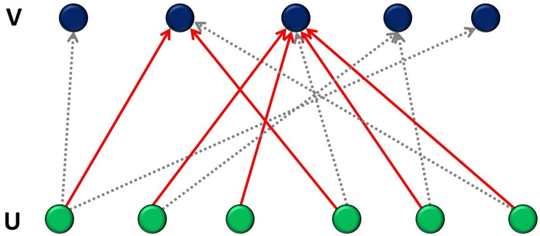
Load balancing problem



Semi-matchings

Semi-matching in a bipartite graph $G = (U, V, E)$:

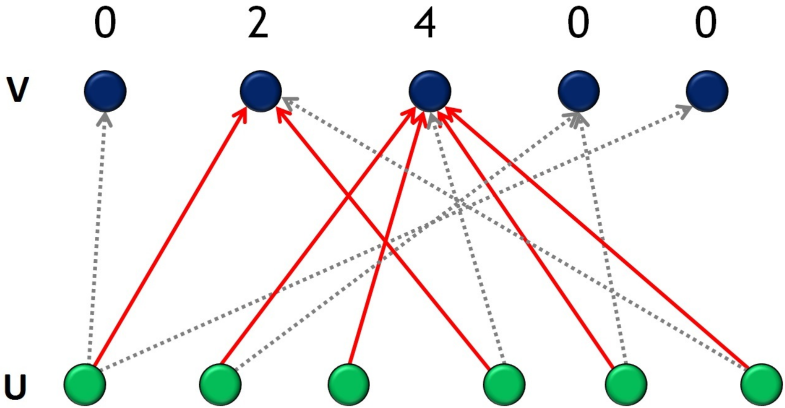
- any subset $M \subseteq E$ such that $deg_M(u) \leq 1$ for all $u \in U$
- *each task is assigned to at most one machine*



Maximum semi-matching - maximizes the number of assigned tasks; if there is *no other restriction* then

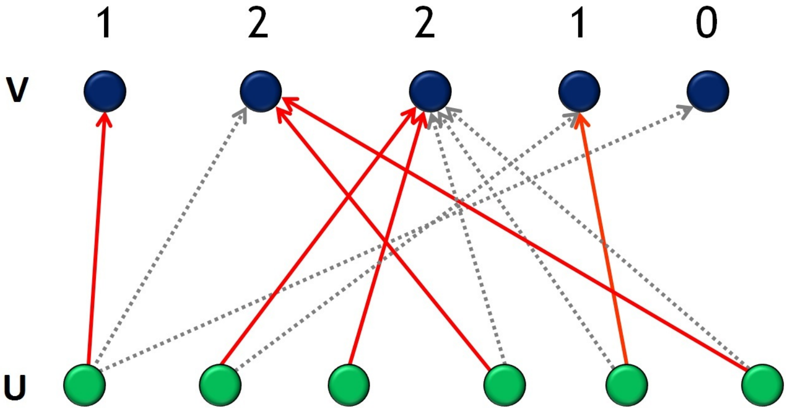
- any subset $M \subseteq E$ such that $deg_M(u) = 1$ for all $u \in U$
- always exists, many maximum semi-matchings

Which semi-matching is better?



Workload distribution (sorted loads): **4, 2, 0, 0, 0**

Which semi-matching is better?



Workload distribution (sorted loads): **2, 2, 1, 1, 0**

Optimal semi-matchings

Cost of a semi-matching M (the total completition time):

$$cost(M) = \sum_{v \in V} \frac{deg_M(v).(deg_M(v) + 1)}{2}$$

Optimal semi-matching

- a maximum semi-matching M such that $cost(M)$ is minimal
- a maximum semi-matching M such that its degree (workload) distribution is lexicographically minimal
  - shown by Bokal et al. to be equivalent with *cost*-minimal semi-matching (and also other cost measures)
  - in the previous example: $(4, 2, 0, 0, 0)$ vs. $(2, 2, 1, 1, 0)$

Our optimality criterion: **lexicographical minimality**

Previous work

Algorithms for computing **an optimal semi-matchings**:

- $O(n^3)$ by Horn (1973) and Bruno et al. (1974)
- $O(n \cdot m)$ by Lovász et al. (2006, JAlgor)
- $O(\min\{n^{3/2}, m \cdot n\} \cdot m)$ by Lovász et al. (2006, JAlgor)
- $O(n \cdot m)$ by Bokal et al. (2009) for generalized setting
- $O(\sqrt{n} \cdot m \cdot \log n)$ by Fakcharoenphol et al. (2010, ICALP)

Algorithms are based on finding (cost-reducing) alternating paths with some properties.

Maximum matchings in bipartite graphs:

- $O(\sqrt{n} \cdot m)$ by Micali and Vazirani (1980)
- $O(n^\omega)$ by Mucha and Sankowski (2004)
  - ω is the exponent of the best known matrix multiplication algorithm
  - randomized algorithm, better for dense graphs

Our work

Can we construct an algorithm for computing an optimal semi-matching that breaks through $O(n^{2.5})$ barrier for dense graphs?

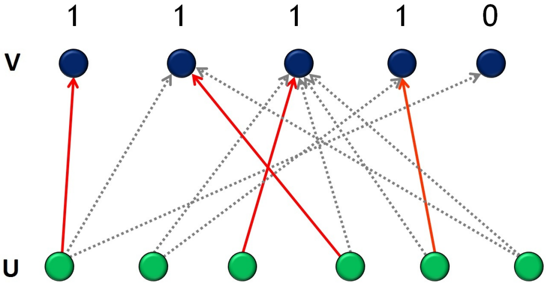
Answer: YES, we can

And moreover (side results):

- **new approach** for computing an optimal semi-matching: divide and conquer strategy instead of cost-reducing alternating paths
  - divide and conquer = more suitable for parallel computation
- **reduction** to a variant of *maximum bounded-degree semi-matching*
  - can be solved by different algorithms and approaches (e.g. maximum matchings, reduction to matrix multiplication)

Limited workload for V-vertices

Restriction: a machine can process only limited number of tasks, e.g. 1 task:

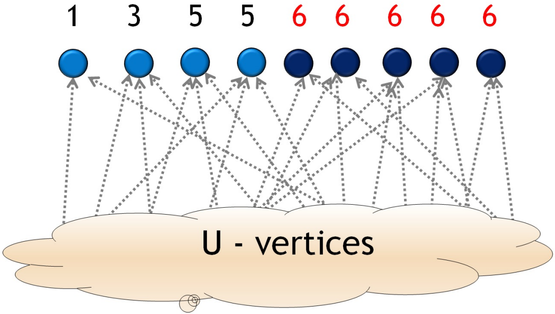


Intuition:

- there can be unassigned tasks
  - *U*-vertices not incident to a matching edge
- larger workload limit for machines = more assigned tasks

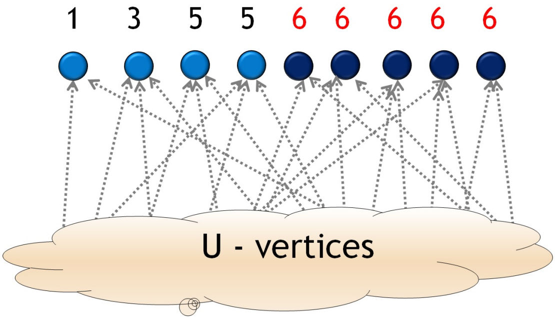
Limited workload for V-vertices

Maximum semi-matching with workload limit 6
(max. 6 tasks per machine):



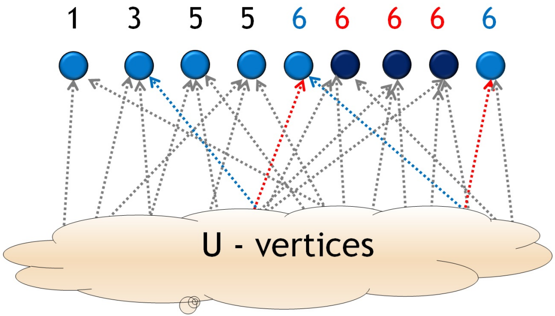
Is it necessary to increase workload limit for all V-vertices (machines) in order to match all U-vertices?

Intuition related to limited workload



- **no sense** to increase the workload limit for vertices (machines) that are not fully loaded in a given maximum semi-matching

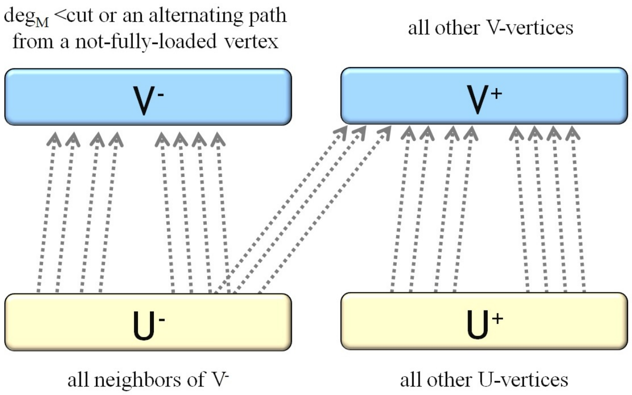
Are all fully-loaded vertices good candidates?



- **no sense** to increase the workload limit for fully loaded vertices (machines) that are endpoints of an alternating path starting in a non-fully loaded vertex

Intuition: How to divide the problem

Maximum semi-matching M respecting a workload limit *cut*:



Find an optimal semi-matching

- in $G^- = (U^-, V^-, E^-)$ by "decreasing" workload limits
- in $G^+ = (U^+, V^+, E^+)$ by "increasing" workload limits

(Sub)problem instances

$LSM(G)$ - a set of all optimal semi-matchings for G

Input/problem instances: $(G, down, up, M_f)$

- an input bipartite graph $G = (U, V, E)$ such that
  - $\forall M \in LSM(G), \forall v \in V : down \leq deg_M(v) \leq up$
- a semi-matching M_f in G such that
  - $\forall v \in V : deg_{M_f}(v) \geq down$

Goal: if $(G, down, up, M_f)$ is an input, compute an optimal semi-matching for G

Starting point: $(G, 0, \infty, \emptyset)$

- G is a graph, in which we want to find an optimal semi-matching
- all preconditions are satisfied

(Sub)problem instances

$LSM(G)$ - a set of all optimal semi-matchings for G

Input/problem instances: $(G, down, up, M_f)$

- an input bipartite graph $G = (U, V, E)$ such that
  - $\forall M \in LSM(G), \forall v \in V : down \leq deg_M(v) \leq up$
- a semi-matching M_f in G such that
  - $\forall v \in V : deg_{M_f}(v) \geq down$

Divide phase for *cut* ($down \leq cut \leq up$):

$$\begin{array}{ccc} & (G, down, up, M_f) & \\ \swarrow & & \searrow \\ (G^-, down, cut, M_f^-) & & (G^+, cut, up, M_f^+) \end{array}$$

Key property:

- $\forall M^- \in LSM(G^-), \forall M^+ \in LSM(G^+)$: $M^- \cup M^+ \in LSM(G)$

Trivial case (or why is M_f required)

Input: $(G, down, up, M_f)$, where $up - down \leq 1$
Problem: How to compute $M \in LSM(G)$?

First idea:

- compute a maximum semi-matching M for load limit up
- it can happen that $M \notin LSM(G)$:
  - $(3, 2, 2, 2, 2, 2) \in LSM(G)$ vs.$(3, 3, 3, 3, 1, 0) \notin LSM(G)$

Solution:

- utilizing M_f with $deg_{M_f}(v) \geq down$ for all $v \in V$, **transform** semi-matching M to a semi-matching M_B such that
  - $|M| = |M_B|$
  - $down \leq deg_{M_B}(v) \leq up$ for all $v \in V$
- it can be shown that $M_B \in LSM(G)$
- transformation can be realized in the linear time

Dividing subroutine - idea

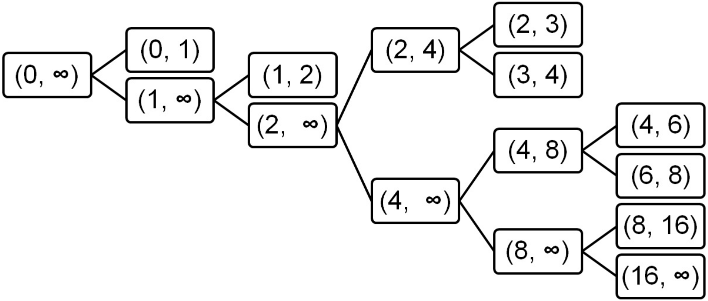
Input instance: $(G, down, up, M_f)$

Computation:

1. compute a maximum semi-matching M for workload limit cut
2. compute M_B by rebalancing M with respect to M_f
3. compute V^-, V^+, U^-, and U^+ considering workload of V-vertices
4. compute induced subgraphs $G^- = (U^-, V^-, E^-)$ and $G^+ = (U^-, V^+, E^+)$
5. compute $M_f^- = M_B \cap E^-$ and $M_f^+ = M_B \cap E^+$
6. return $(G^-, down, cut, M_f^-)$ and (G^+, cut, up, M_f^+)

Main algorithm - Divide and conquer

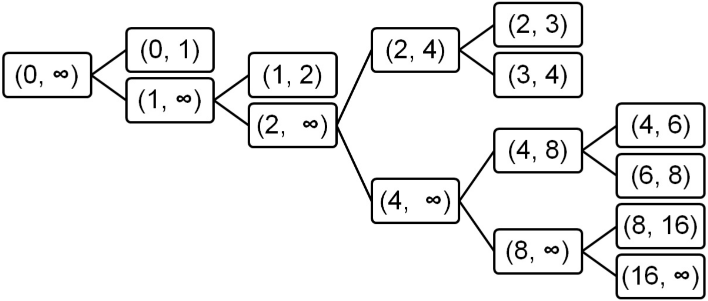
Computational tree starting with $(G, 0, \infty, \emptyset)$:



- Divide and conquer: $(down, up)$ is always divided into 2 subintervals (of almost equal size)
- Doubling: $(down, \infty)$ is divided to $(down, 2 \cdot down)$ and $(2 \cdot down, \infty)$

Main algorithm - Computation

Computational tree starting with $(G, 0, \infty, \emptyset)$:



- after $O(\log n)$ levels, graphs of subproblems are empty
  - there is no subgraph of G for which a semi-matching with load of a V-vertex at least $n+1$ exists

Maximum semi-matching with workload limits?

- in each step of the algorithm, we need a maximum semi-matching that respects the workload limits

Problem (Bounded-degree semi-matching)

Instance: *A bipartite graph* $G = (U, V, E)$ *with* $n = |U| + |V|$ *vertices and* $m = |E|$ *edges; a capacity mapping* $c : V \to \mathbb{N}$ *satisfying* $\sum_{v \in V} c(v) \leq 2 \cdot n$.

Question: *Find a semi-matching* M *in* G *with maximum number of edges such that* $deg_M(v) \leq c(v)$ *for all* $v \in V$.

Time complexity notation: $T_{BDSM}(n, m)$ *for a graph n vertices and m edges.*

Total time for computing an optimal semi-matching:

$$O((n + m + T_{BDSM}(n, m)) \cdot \log n)$$

Bounded-degree semi-matching

Reduction to maximum matching:

- make $c(v)$ **copies** of each V-vertex v
- new graph has at most $3 \cdot n$ vertices
- apply algorithm for maximum matching in $O(n^{\omega})$ by Mucha and Sankowski

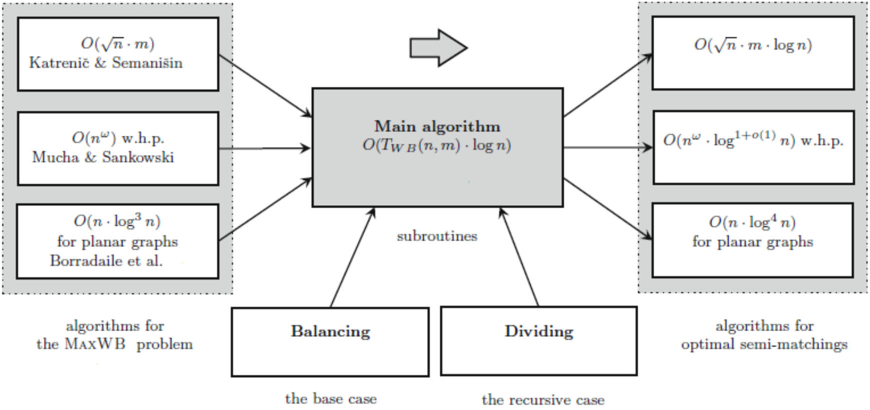
$$O(n^{\omega} \cdot \log n)$$

Reduction to $(1, c)$-semi-matchings:

- $(1, c)$-semi-matching is bounded-degree semi-matching **without** condition $\sum_{v \in V} c(v) \leq 2 \cdot n$
- due to algorithm by Katrenič and G.S., $(1, c)$-semi-matching can be computed in time $O(\sqrt{n} \cdot m)$

$$O(\sqrt{n} \cdot m \cdot \log n)$$

Conclusion


$O(\sqrt{n} \cdot m)$
Katrenič & Semanišin
$O(n^\omega)$ w.h.p.
Mucha & Sankowski
$O(n \cdot \log^3 n)$
for planar graphs
Borradaile et al.
algorithms for
the MAXWB problem
Main algorithm
$O(T_{WB}(n, m) \cdot \log n)$
subroutines
Balancing
the base case
Dividing
the recursive case
$O(\sqrt{n} \cdot m \cdot \log n)$
$O(n^\omega \cdot \log^{1+o(1)} n)$ w.h.p.
$O(n \cdot \log^4 n)$
for planar graphs
algorithms for
optimal semi-matchings

Related works

Semi-matching problem

- algorithm for optimal weighted semi-matching with time complexity $O(n^2.m)$ (Harada et al. 2007)
- distributed deterministic 2-approximation algorithms with time complexity $O(\Delta^5)$ (Czygrinow et al. 2012, 2016)
- a deterministic one-pass streaming algorithm that for any $0 \leq \epsilon \leq 1$ uses computes an $O(n^{(1-\epsilon)/2})$ and with $O(\log n)$ passes computes an $O(\log n)$ approximation (Konrad et al. 2016)

Distributed Backup Placement problem

- a distributed algorithm which finds placement in polylogarithmic time and approximation ratio $O(\frac{\log n}{\log \log n})$ (Halldórsson et al. 2015)

Future work

A generalisation introduced by Bokal et al.
An (f, g)**-quasi-matching** in a bipartite graph $G = (U \cup V, E)$: any set of edges $M \subseteq E$ such that

- each vertex $u \in U$ is incident with at most $f(u)$ edges of M
- each vertex $v \in V$ is incident with at least $g(v)$ edges of M.

Algorithm complexity

- Bokal et al. provided algorithm with complexity $O(m.g(B)$, where $g(B) = \sum_{v \in B} g(v)$
- our reduction lemma should allow an **improvement of this result by factor** $\log n$ **and factor** $\sqrt{(g(B))}$.

References

D. Bokal, B. Brešar, J. Jerebic:
A generalization of Hungarian method and Hall's theorem with applications in wireless sensor networks
Discret. Appl. Math. 160(4–5), 460–470 (2012)

J. Fakcharoenphol, B. Laekhanukit, D. Nanongkai
Faster algorithms for semi-matching problems
ACM Trans. Algorithms 10(3), 14:1–14:23 (2014)

Katrenič, J., Semanišin, G.
Maximum semi-matching problem in bipartite graphs
Discuss. Math. Graph Theory 33, 559–569 (2013)

F. Galčík, J. Katrenič, G. Semanišin
On computing an optimal semi-matching
(to appear in Algorithmica, already published electronically)

Thank you for your attention

Gabriel Semanišin

e-mail: gabriel.semanisin@upjs.sk