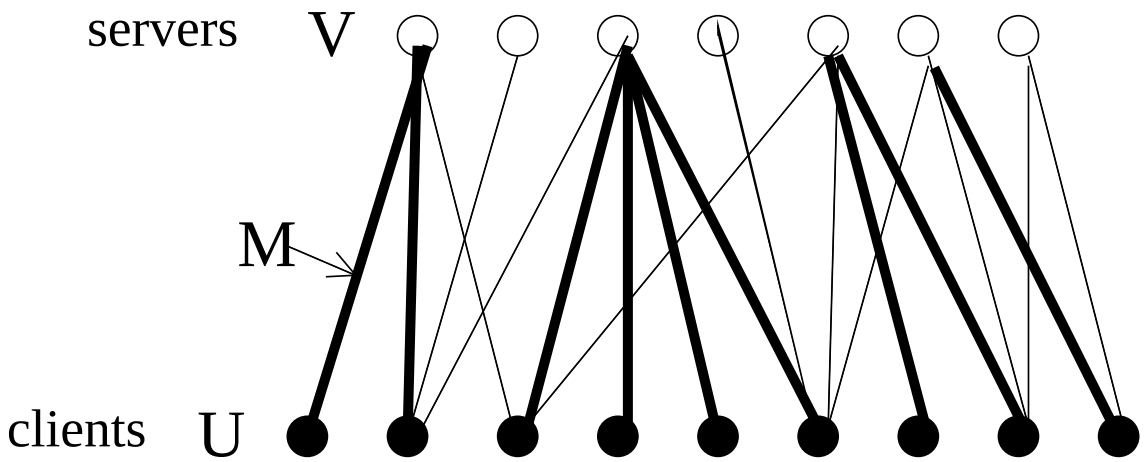
Distributed 2-approximation algorithm for the semi-matching problem

W. Wawrzyniak, A. Czygrinow, M. Hańćkowiak,
and E. Szymańska

November 14, 2013

The semi-matching problem



- each client needs 1 server ($d_M(u) = 1$ for all $u \in U$)
- all servers should have $\approx$ equal "load" ($d_M(v)$),
  that is we want to minimize the following cost function:

$$\sum_{v \in V} \binom{d_M(v)+1}{2} \quad \left(\text{or} \quad \sum_{v \in V} d_M^2(v) \right)$$

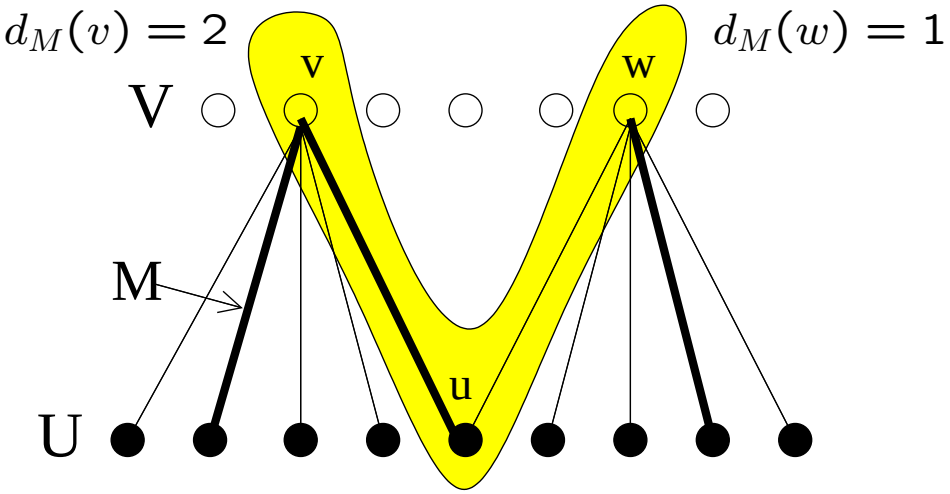
(the first is the "total completion time" interpretation)

Related Work

- "load balancing"
- N. J. A. Harvey, R. E. Ladner, L. Lovasz, T. Tamir
  Semi-matchings for bipartite graphs and load balancing,
  J. Algorithms 59 (1), (2006)
- J. Fakcharoenphol, B. Laekhanukit, D. Nanongkai
  Faster algorithms for semi-matching problems
  ICALP'10, (2010)
- F. Galčík, J. Katrenič, G. Semanišin
  On computing an optimal semi-matching.
  Graph-Theoretic Concepts in Computer Science, (2011)
- A. Czygrinow, M. Hańćkowiak, K. Krzywdziński, E. Szymańska, W. Wawrzyniak
  Distributed approximations for the semi-matching problem,
  Brief Announcement: DISC 2011

Key definition (of non-swappable)

a semi-matching M is non-swappable
if for all paths vuw, $vu \in M$, $uw \in E \setminus M$: $d_M(v) - d_M(w) \leq 1$

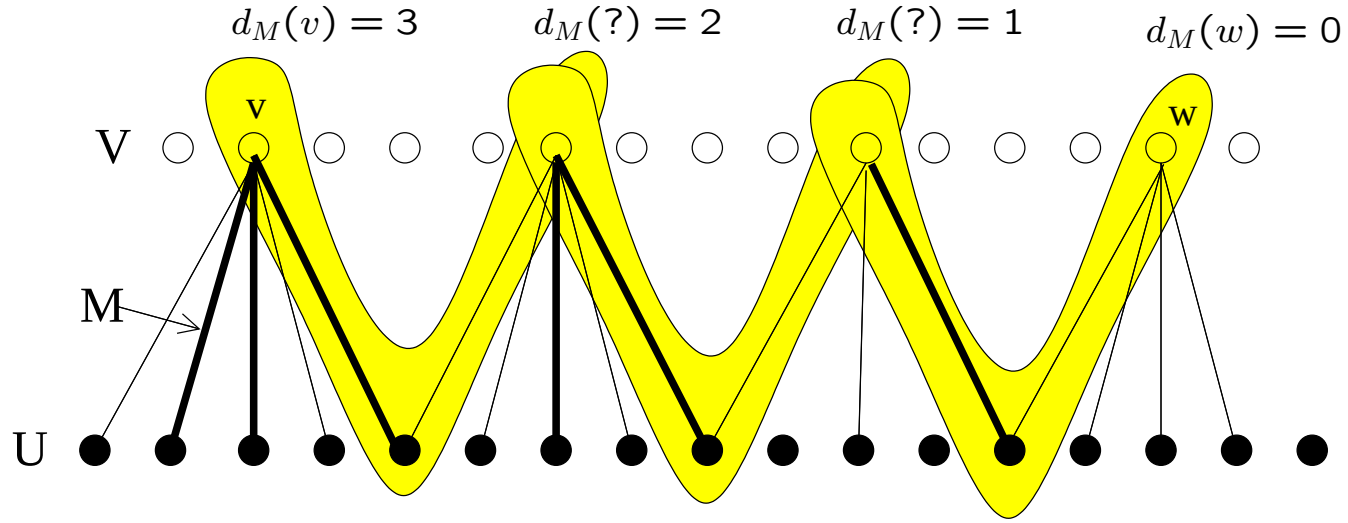


when $d_M(v) - d_M(w) > 1$ then vuw is called **bad path** !

Main theorem

a non-swappable semi-matching M is a 2 (or 3) - approximation of the minimum cost semi-matching

Sketch of the proof of the main theorem



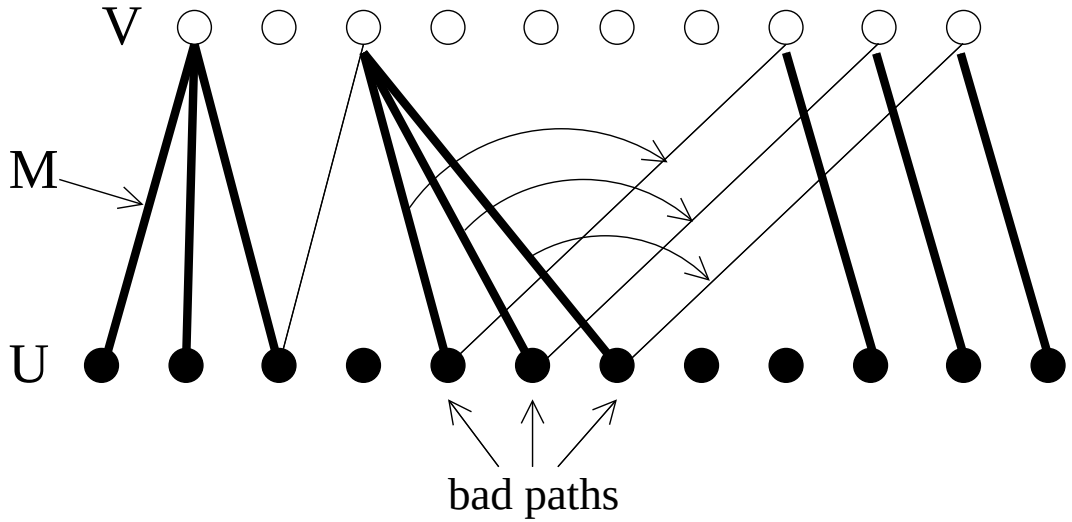
- M - non-swappable semi-matching
- M_1 - after swapping "yellow" paths
- $cost(M) - cost(M_1) = 2(d_M(v) - d_M(w) - 1) \leq 2(k-1) \leq 2k$ where k is the number of yellow paths
- M, M_1, M_2, ..., M^* - optimal semi-matching
- $cost(M) - cost(M^*) \leq 2|U|$, $|U| \leq cost(M^*)$, $cost(M) \leq 3cost(M^*)$

What are "long yellow paths" ?

- M - input semi-matching, M^* - optimal semi-matching
- we define a digraph D with :
  $V(D) = V$
  $(u, v) \in E(D) \iff \{u, a\} \in M$ and $\{a, v\} \in M^*$ for $a \in U$
- fact: D can be decomposed into arc-disjoint open trails
- these trails correspond to "long yellow paths" ...

Our main algorithm (for small $\Delta(V)$)

- in a distributed/synchronous model of computation
- take arbitrary semimatching M and make it non-swappable
- it must be done very carefully, because...



- our algorithm is working in $O(\Delta(V)^5)$-rounds
- it computes 2 (or 3) -approximation of semi-matching problem

Our main algorithm (for small $\Delta(V)$)

Procedure SemiMatch($G = (V, U, E)$)

1. $\forall u \in U$ pick an arbitrary edge e_u incident to u and let $M = \bigcup_{u \in U} e_u$.
2. for $k = 0$ to $\Delta - 2$
   for $i = 0$ to 2Δ
   (a) $\forall_{v \in V}\, l(v) = d_M(v)$
   $\forall_{t=0,\ldots,\Delta}\, L_t = \{v \in V | l(v) = t\}$
   (b) $X = Bad_{ind}(V_{>k+1}, V_k)$
   (c) $S = Ends(X)$, $S^c = V \setminus S$
   (d) $M = M \oplus X$
   (e) for $j = 0$ to $2\Delta^2$
   - $Y = \bigcup_{t=1}^{k} Bad_{ind}(L_t \cap S, L_{t-1} \cap S^c)$
   - $S = S \cup Ends(Y) \setminus Starts(Y)$
     $S^c = V \setminus S$
   - $M = M \oplus Y$
3. return M

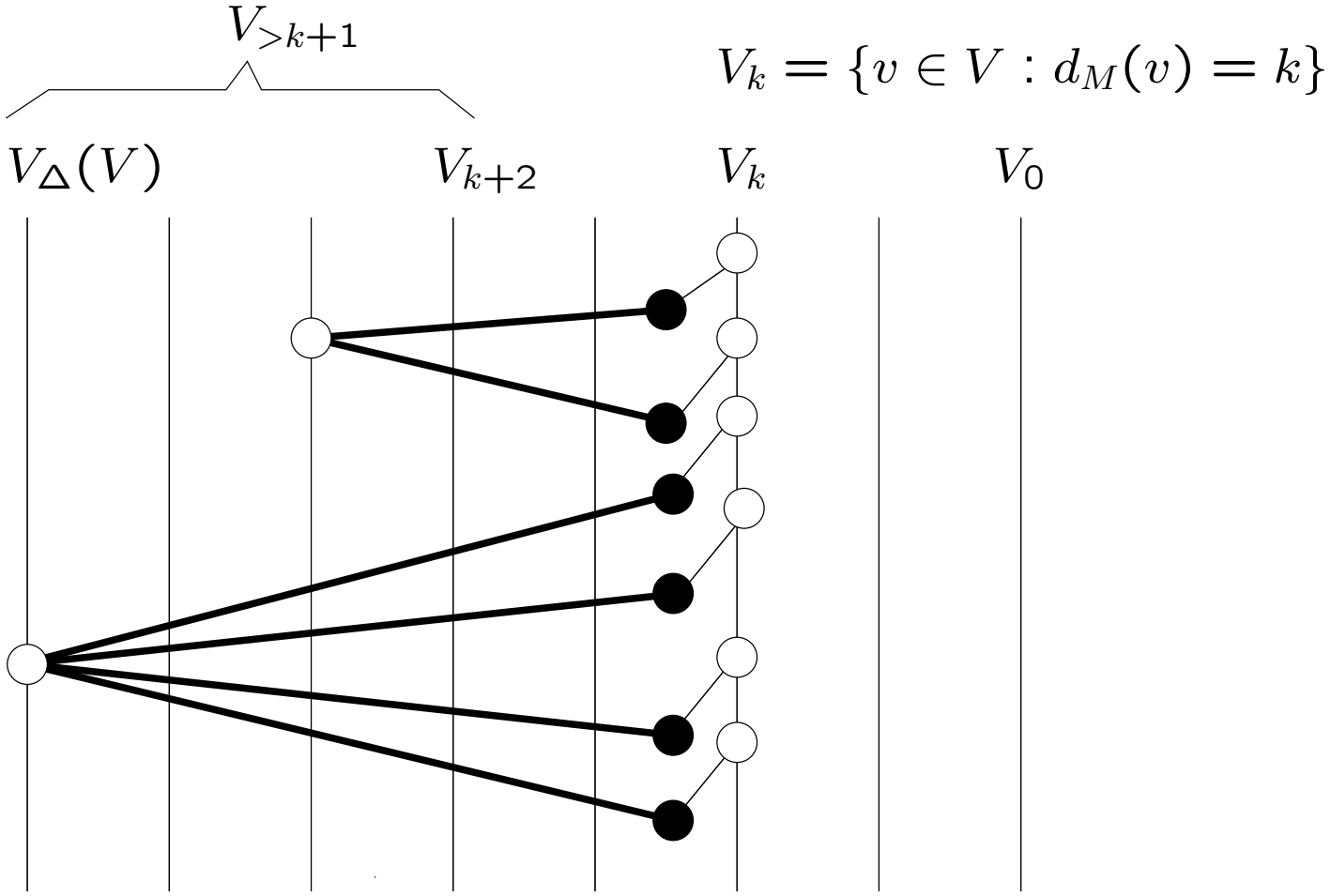
$V_k := \{v \in V : d_M(v) = k\}$
$V_{>x} := \{v \in V : d_M(v) > x\}$

$Bad(A, B)$ - a set of bad paths from set A to B

$Bad_{ind}(A, B)$ - a procedure finding "independent" set of pahts in $Bad(A, B)$

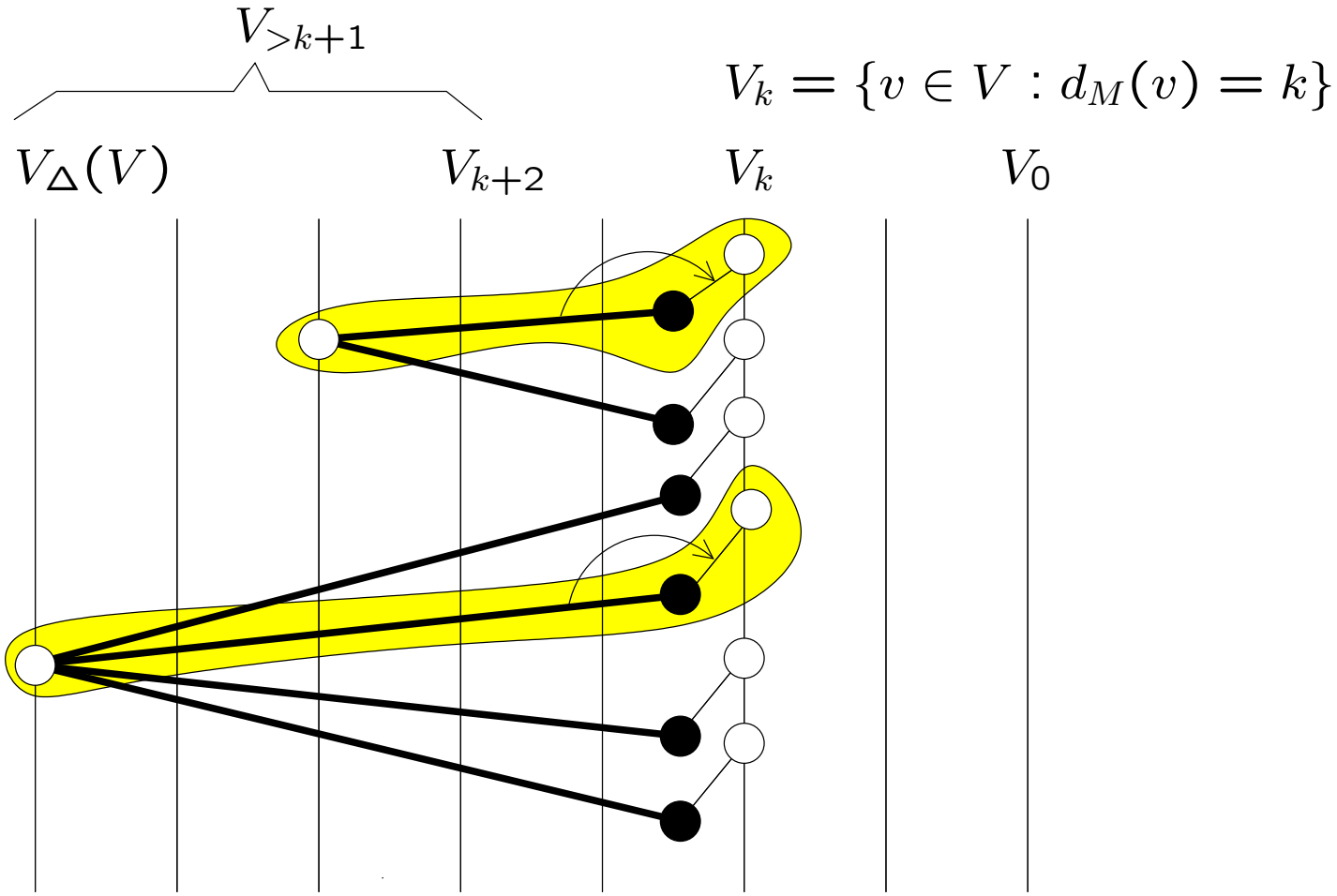
Our main algorithm (for small $\Delta(V)$)

- in k-th iteration of the main loop
  we eliminate bad paths from $Bad(V_{>k+1}, V_k)$ ...



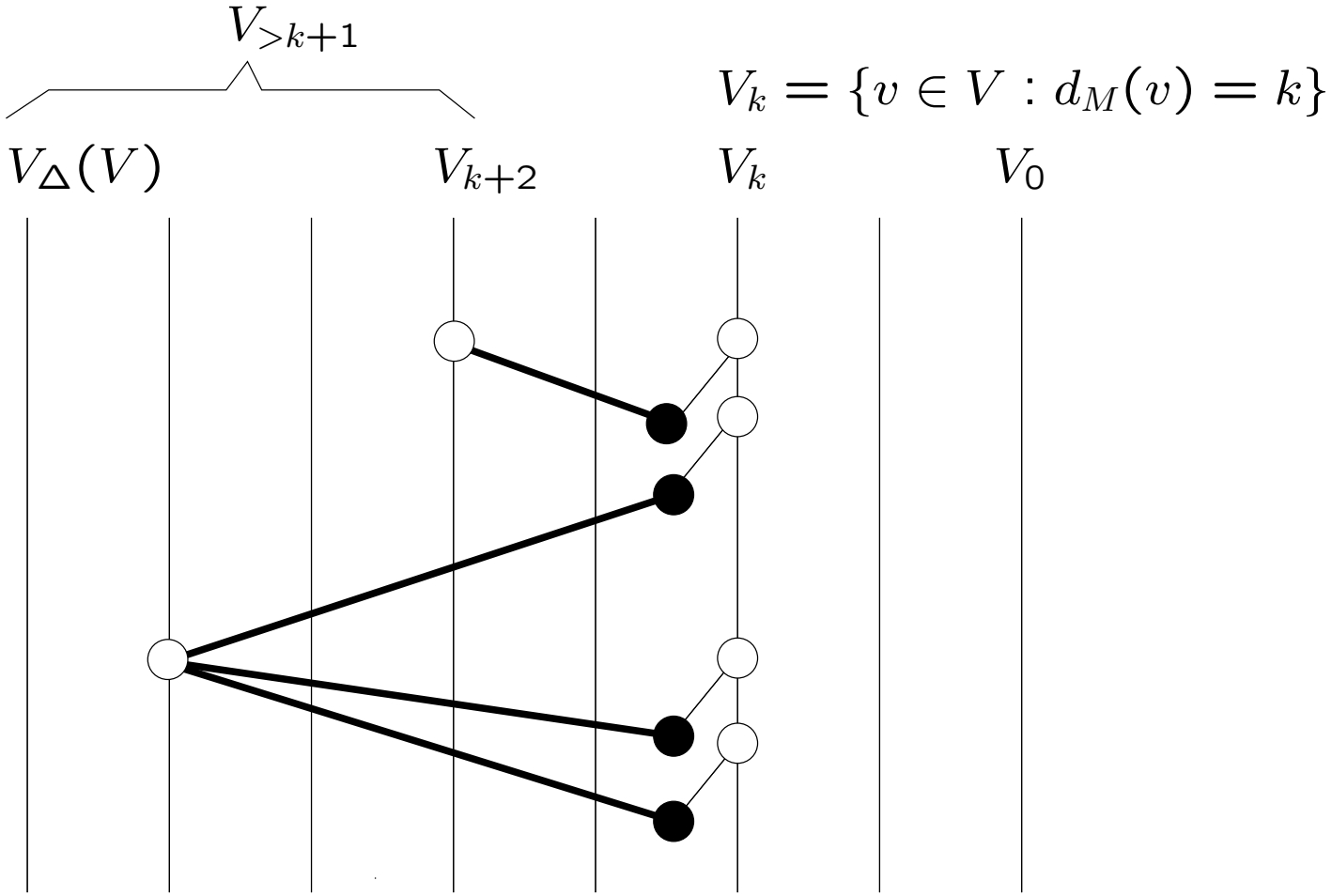
Our main algorithm (for small $\Delta(V)$)

- we call $Bad_{ind}(V_{>k+1}, V_k)$
  to find independent paths and "swap" them ...



Our main algorithm (for small $\Delta(V)$)

- ... after $O(\Delta(V))$ -iterations $Bad(V_{>k+1}, V_k) = \emptyset$



The algorithm for large $\Delta(V)$

- the reduction to MSSC (Min Sum Set Cover) problem on a hipergraph $H = (V(H), E(H))$, $n = |V(H)|$
- MSSC problem: find a bijection $h : V(H) \mapsto \{1, ..., n\}$ to minimize the function

$$cost_{MSSC}(h) := \sum_{e \in E(H)} \min\{h(v) : v \in e\}$$

- for $i := 1$ to n do:
  - find a vertex v with the largest $d_H(v)$
  - $h(v) := i$, remove from H all hyper-edges containing v
- greedy algorithm finds 4 -approximation of MSSC problem:
  U. Feige, L. Lovasz, P. Tetali
  Approximating Min Sum Set Cover
  Algorithmica 40(4), (2004)

The algorithm for large $\Delta(V)$

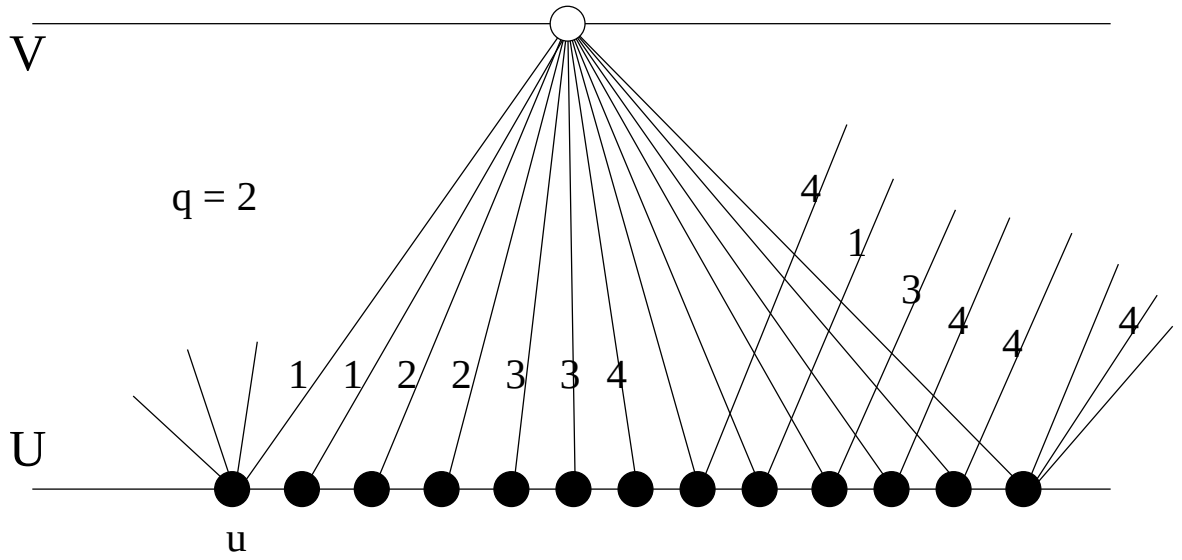
- definition: q-matching M is a set of edges such that for all $v \in V$ $d_M(v) \leq q$ and for all $u \in U$ $d_M(u) \leq 1$

- we define a hipergraph H as follows:
  $V(H)$ - a set of all q-matchings in G
  $E(H) = U$

- note that for $v \in V(H)$ and $e \in E(V)$:
  $v \in e \Longleftrightarrow$ that q-matching "v" matches a vertex "e" from U
  $d_H(v) =$ size of q-matching "v"

- while U is not empty
  - find a maximal q-matching M
    (and assign to its edges the next number)
  - remove all vertices from U matched by M

The algorithm for large $\Delta(V)$



$$cost_{MSSC}(M) = \sum_{u \in U} g(u) \approx \sum_{v \in V} \begin{cases} d_M(v), d_M(v) < q \\ \frac{1}{q} d_M^2(v), d_M(v) \geq q \end{cases}$$

- it works only for special graphs !
  when there exists an optimal semi-matching M^*
  s.t. $\forall_{v \in V} d_{M^*}(v) \geq q$

The algorithm for large $\Delta(V)$

- let G satisfies:
  $\delta(V) \geq \Delta(V)/a$ and $\Delta(U) \leq b$; a, b -const

- there exists an optimal semi-matching M^*
  with $\forall_{v \in V} d_{M^*}(v) \geq \Delta(V)/(ab)$

- the greedy algorithm with $q := \Delta(V)/(ab)$
  finds 36-approx of semi-matching problem,
  in $O(ab^2)$ -rounds

Open Problems

1. const -approx of the semi-matching problem for general graphs
   (or when $\Delta(U)$ is const.)

2. PTAS, that is $(1+\epsilon)$ -approx, in $O(\Delta(V)^{O(1/\epsilon)})$ -rounds

3. lower bounds
   (we only know that finding optimal semi-matching
   requires $\Omega(|V|)$ -rounds...)

THANK YOU FOR YOUR ATTENTION !