# *Minimum sum multicoloring on the edges of planar graphs and partial $k$-trees*

Dániel Marx

Budapest University of Technology and Economics

dmarx@cs.bme.hu

Workshop on Approximation and Online Algorithms

14th September, 2004

Bergen, Norway

# *Minimum sum multicoloring*

- **Given:** a graph $G(V, E)$, and demand function $x$: $V \longrightarrow \mathbb{N}$
- **Find:** an assignment of $x(v)$ colors (integers) to every vertex $v$, such that neighbors receive disjoint sets

**Finish time:** $f(v)$ of vertex $v$ is the largest color assigned to it in the coloring.

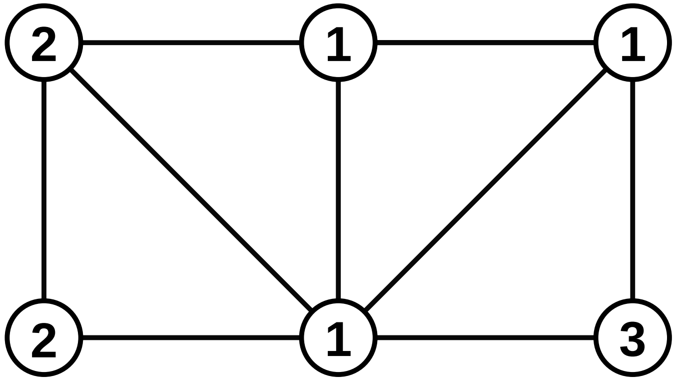
- **Goal:** Minimize $\sum_{v \in V} f(v)$, the **sum of the coloring**.

# *Minimum sum multicoloring*

- **Given:** a graph $G(V, E)$, and demand function $x$: $V \longrightarrow \mathbb{N}$
- **Find:** an assignment of $x(v)$ colors (integers) to every vertex $v$, such that neighbors receive disjoint sets

**Finish time:** $f(v)$ of vertex $v$ is the largest color assigned to it in the coloring.

- **Goal:** Minimize $\sum_{v \in V} f(v)$, the **sum of the coloring**.

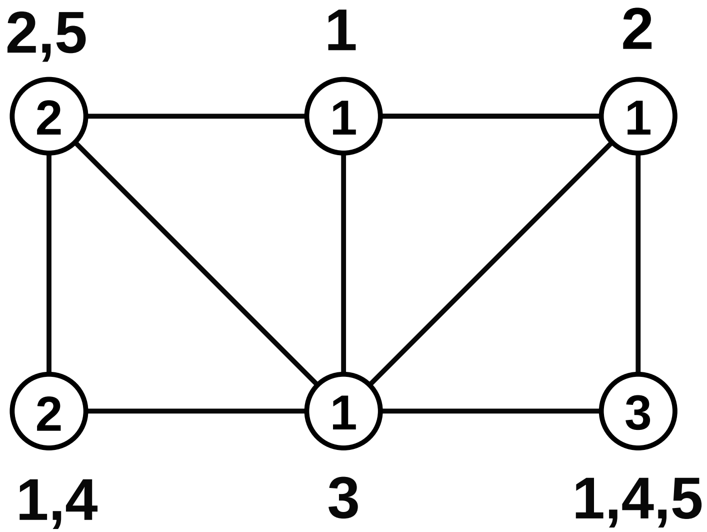


# *Minimum sum multicoloring*

- **Given:** a graph $G(V, E)$, and demand function $x$: $V \longrightarrow \mathbb{N}$
- **Find:** an assignment of $x(v)$ colors (integers) to every vertex $v$, such that neighbors receive disjoint sets

**Finish time:** $f(v)$ of vertex $v$ is the largest color assigned to it in the coloring.

- **Goal:** Minimize $\sum_{v \in V} f(v)$, the **sum of the coloring**.

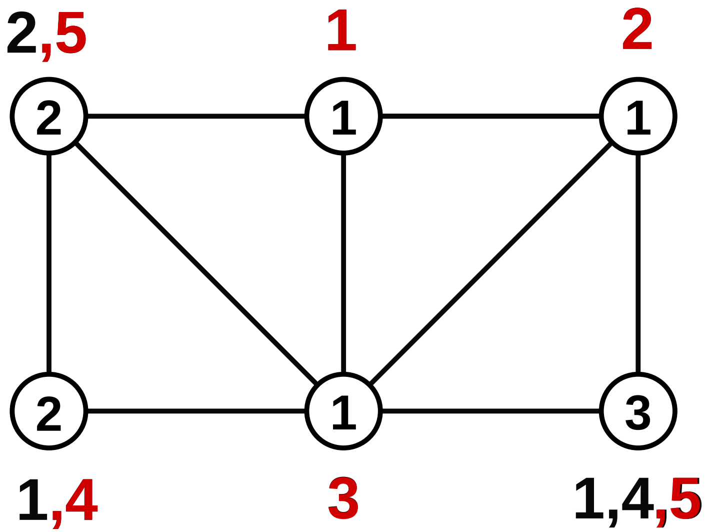


# *Minimum sum multicoloring*

- **Given:** a graph $G(V, E)$, and demand function $x$: $V \longrightarrow \mathbb{N}$
- **Find:** an assignment of $x(v)$ colors (integers) to every vertex $v$, such that neighbors receive disjoint sets

**Finish time:** $f(v)$ of vertex $v$ is the largest color assigned to it in the coloring.

- **Goal:** Minimize $\sum_{v \in V} f(v)$, the **sum of the coloring**.




**Sum of the coloring:**

5 + 1 + 2 + 4 + 3 + 5 = 20

# ***Edge coloring version***

Assign $x(e)$ colors to each edge $e$, minimize the sum of the finish times of the edges. Each color can appear at most once at a vertex.

**Line graph:** in the line graph of $G$ there is one vertex for each edge of $G$, vertices corresponding to adjacent edges are connected.

Edge coloring is the same as coloring the **line graph** of the graph.

# *Approximation schemes*

- [Halldórsson et al. 2003] PTAS for the minimum sum multicoloring of trees.
- [Halldórsson and Kortsarz 2002] PTAS for the minimum sum multicoloring of partial $k$-trees and planar graphs.
- [M. 2003] PTAS for the minimum sum **edge** multicoloring of trees.
- [This paper] PTAS for the minimum sum **edge** multicoloring of partial $k$-trees and planar graphs.

# *Approximation schemes*

- [Halldórsson et al. 2003] PTAS for the minimum sum multicoloring of trees.
- [Halldórsson and Kortsarz 2002] PTAS for the minimum sum multicoloring of partial $k$-trees and planar graphs.
- [M. 2003] PTAS for the minimum sum **edge** multicoloring of trees.
  - bounded degree trees / almost bounded degree trees / arbitrary trees
- [This paper] PTAS for the minimum sum **edge** multicoloring of partial $k$-trees and planar graphs.
  - bounded degree partial $k$-trees / almost bounded degree partial $k$-trees / arbitrary partial $k$-trees / planar graphs

**Definition:** a graph has **almost bounded degree** if it has bounded degree after deleting the degree 1 nodes.

# *Partial $k$-trees*

**$k$-trees** are defined recursively:

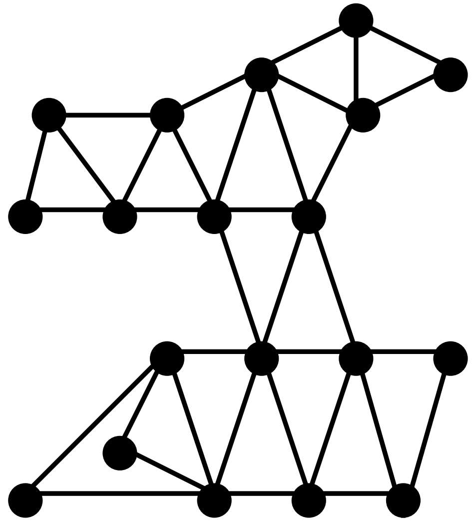
- A $k$-clique is a $k$-tree.
- Attaching a vertex to a $k$-clique of a $k$-tree gives another $k$-tree.
- Every $k$-tree can be obtained with these two rules.

**Partial $k$-tree:** subgraph of a $k$-tree.

Partial $\mathbf{1}$-trees $=$ forests

Many problems are efficiently solvable on partial $k$-trees using dynamic programming.

A $\mathbf{2}$-tree:



# *Approximate color sets*

**Idea:** A collection of color sets is assigned to each vertex, such that there is a good approximate coloring using these sets. Then we use dynamic programming to find the best coloring with these sets.

Dynamic programming can be used for partial $k$-trees:

- Trees $\checkmark$
- Partial $k$-trees $\checkmark$
- Line graph of a tree with max degree $d \Rightarrow$ partial $(d-1)$-tree $\checkmark$
- Line graph of a partial $k$-tree with max degree $d \Rightarrow [(k+1)d-1]$-tree $\checkmark$

There is a more efficient direct algorithm for the edge coloring of trees and partial $k$-trees, which works even for **almost bounded degree graphs.**

# *Finding approximate color sets*

How to construct these collections of color sets?

**Trees:** Using some easy properties of bipartite graphs.

**Partial $k$-trees:** Sophisticated probabilistic arguments (Chernoff's Bound).

**Edges of trees:** Greedy algorithm.

**Edges of partial $k$-trees:** we use the following theorem:

**Theorem:** [Kahn, 1996] For every $\epsilon > 0$, there is a constant $D(\epsilon)$ such that if a multigraph has maximum degree at least $D(\epsilon)$, then its chromatic index is at most $(1 + \epsilon)$-times the fractional chromatic index.

# ***Rounding the demand***

**Theorem:** If the graph is a **tree**, then increasing the demand of each edge to the next power of $(1+\varepsilon)$ increases the sum by at most a factor of $(1+\varepsilon)$.

Weaker property:

**Theorem:** If the graph is a **partial $k$-tree**, then increasing the demand of each edge to the next power of $(1+\varepsilon)$ increases the sum by at most a factor of $(1+2(k+1)\varepsilon)$.

$\Rightarrow$ we can assume that each demand is of the form $(1+\varepsilon)^i$

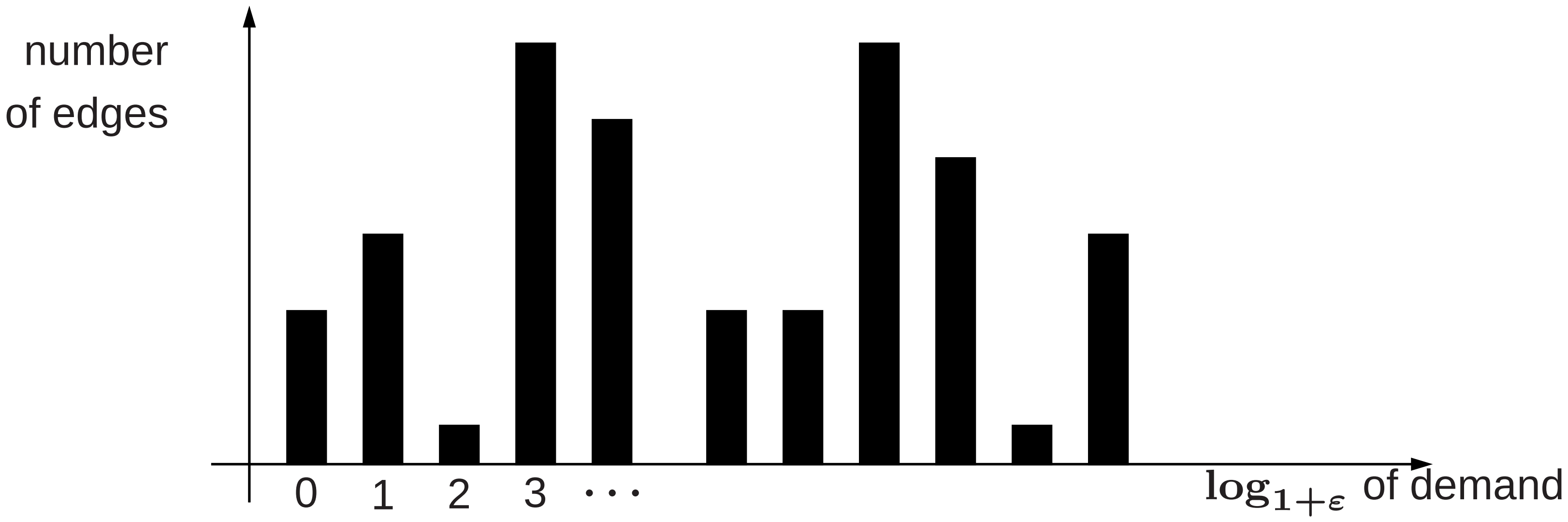
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:



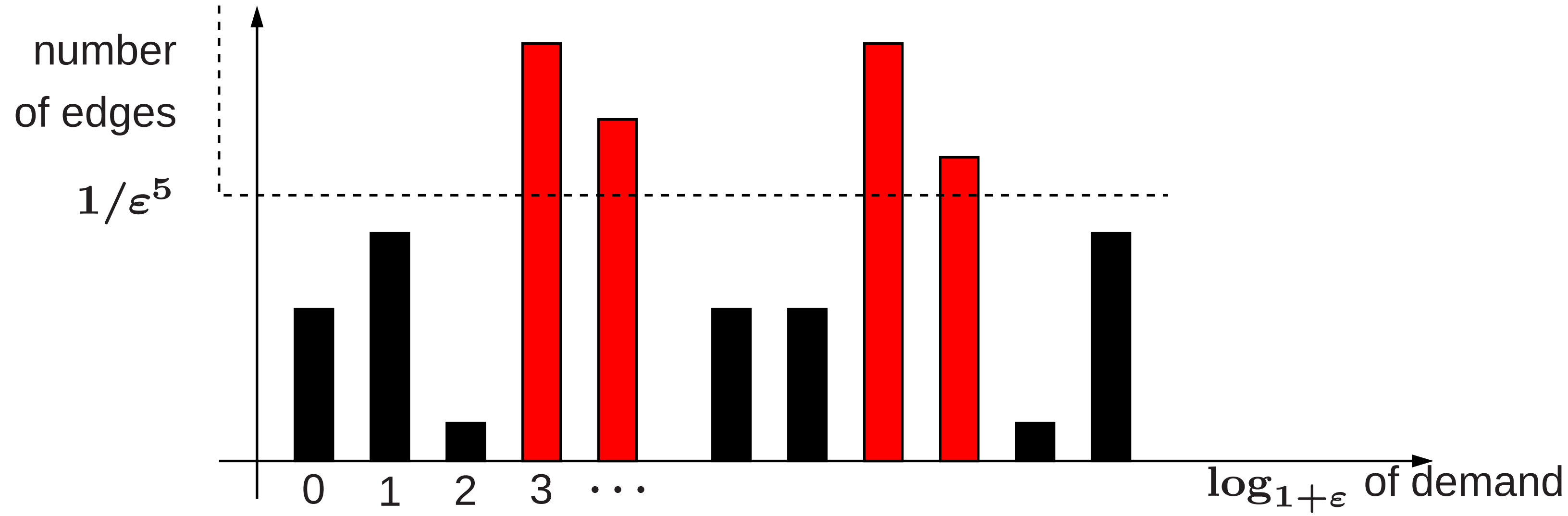
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:



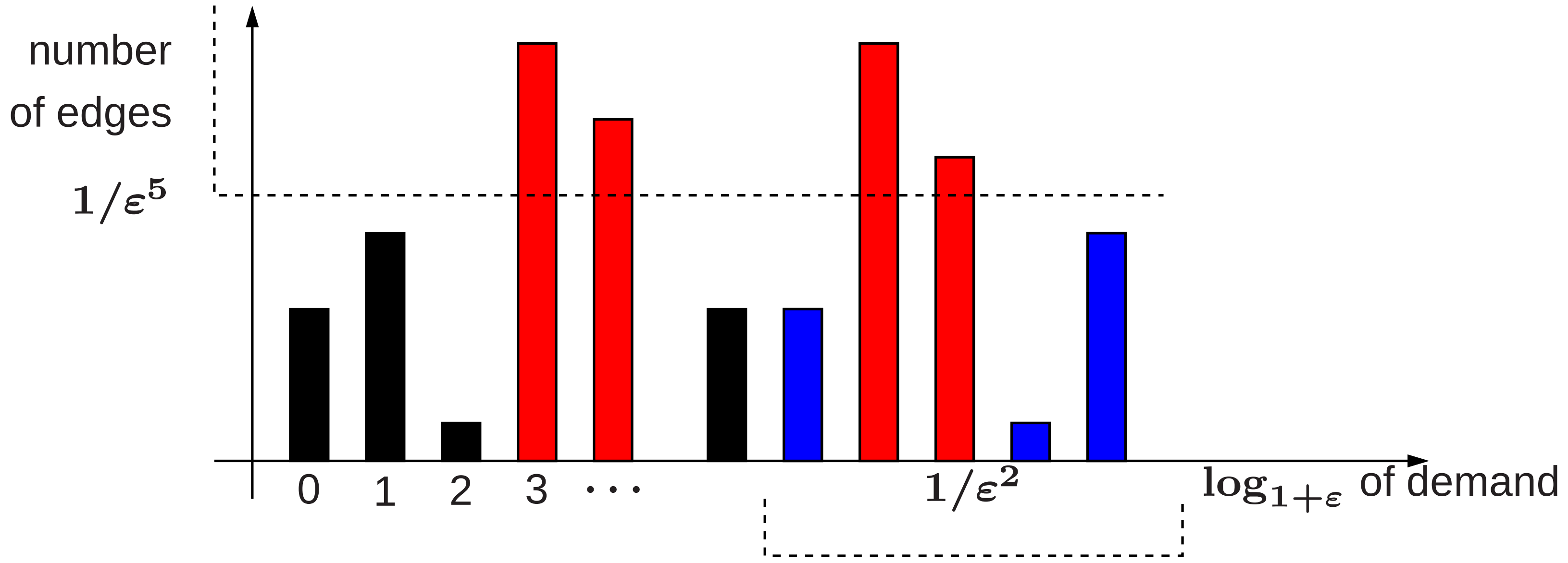
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:



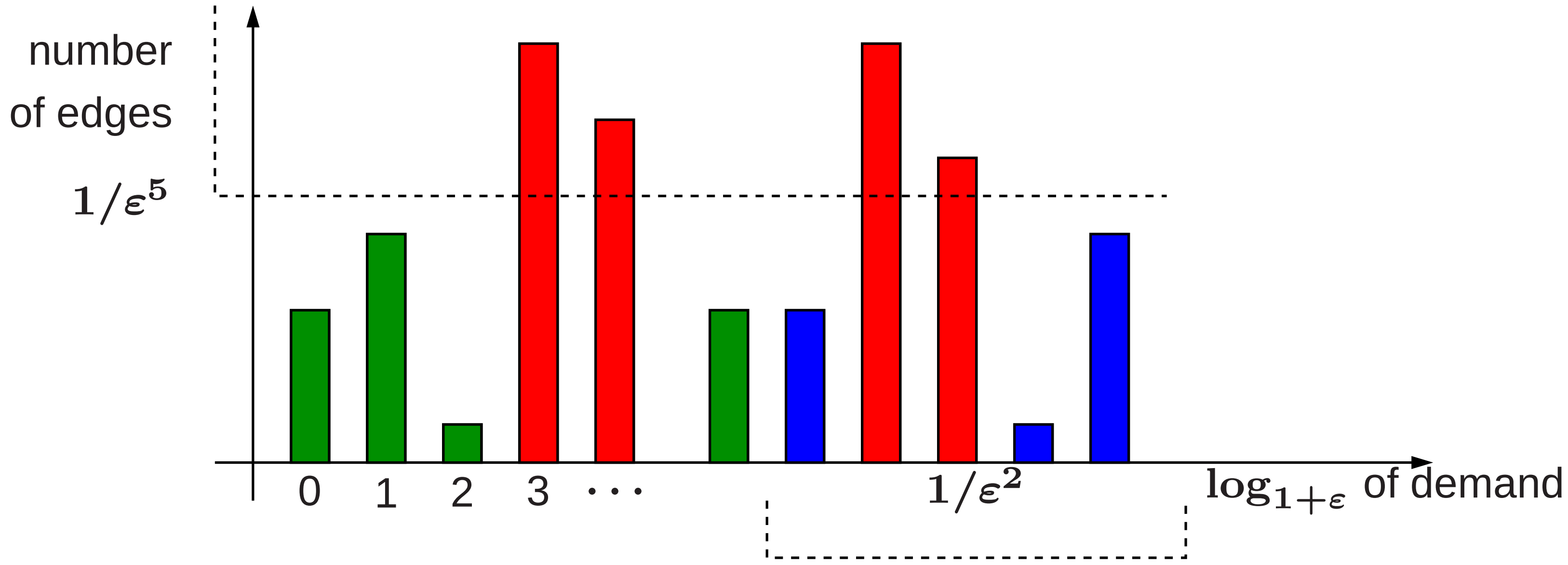
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:



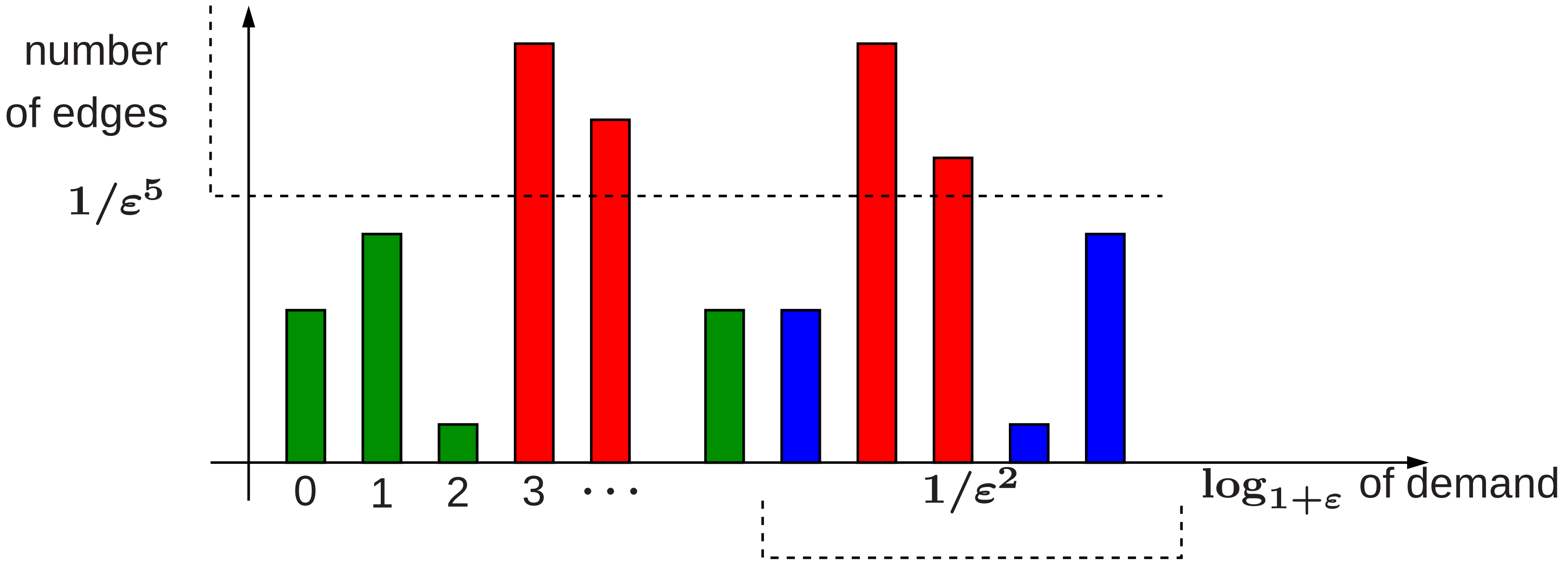
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:




Total demand of the small edges is very small, they can be thrown away.

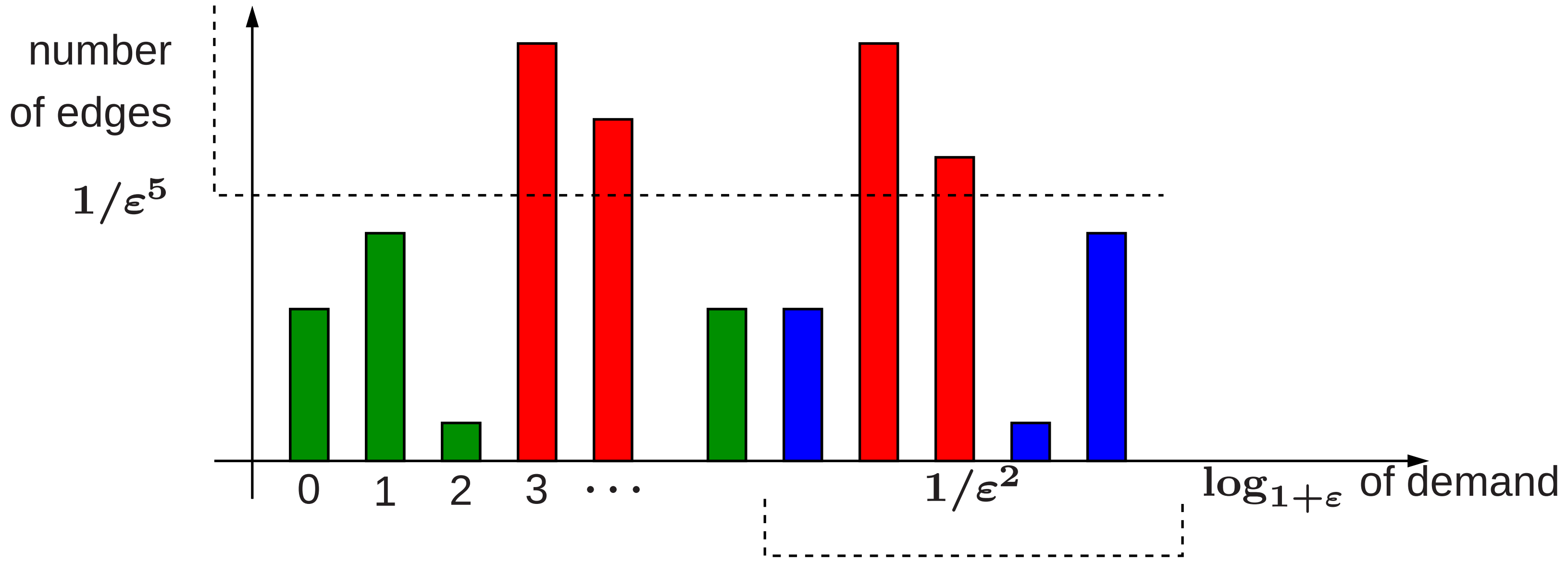
# *The Small, the Large, and the Frequent*

We orient the edges such that at most $k$ edges leave every vertex.

The edges entering a node are divided into small, large, and frequent edges.

Every demand is of the form $(1+\varepsilon)^i$.

Number of edges for each demand size:

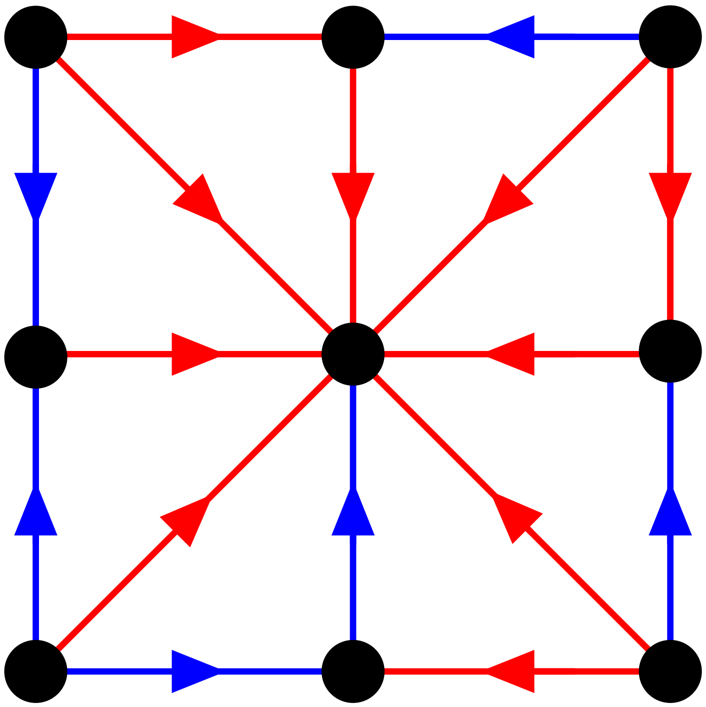



Total demand of the small edges is very small, they can be thrown away.

At most a constant number ($\leq 1/\varepsilon^7$) of large edges enter each vertex.

# *Splitting the graph*

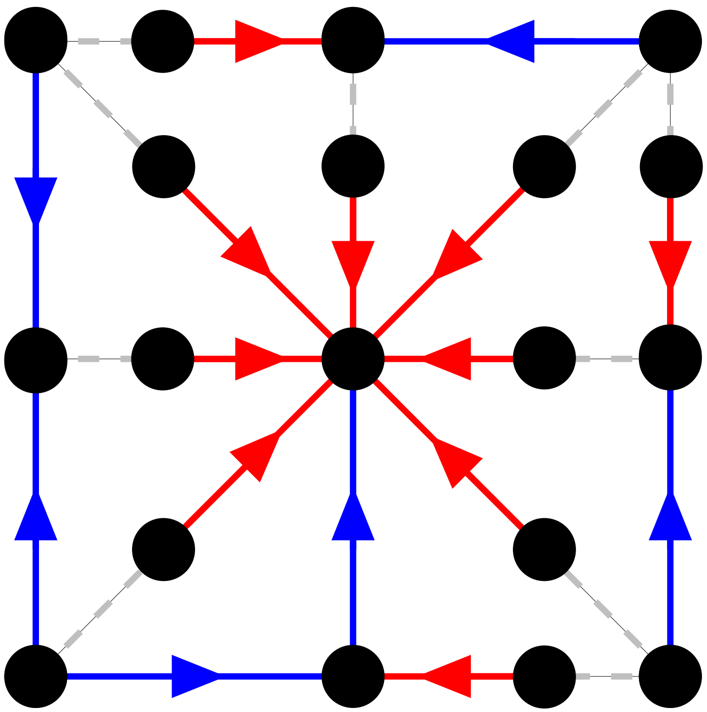
We split the frequent edges:



- At most a constant number of large edges enter each vertex.
- The graph is split at the frequent edges.
- Almost bounded degree: after deleting the degree one vertices only the large edges remain.
- Thus the PTAS can be used.

# *Splitting the graph*

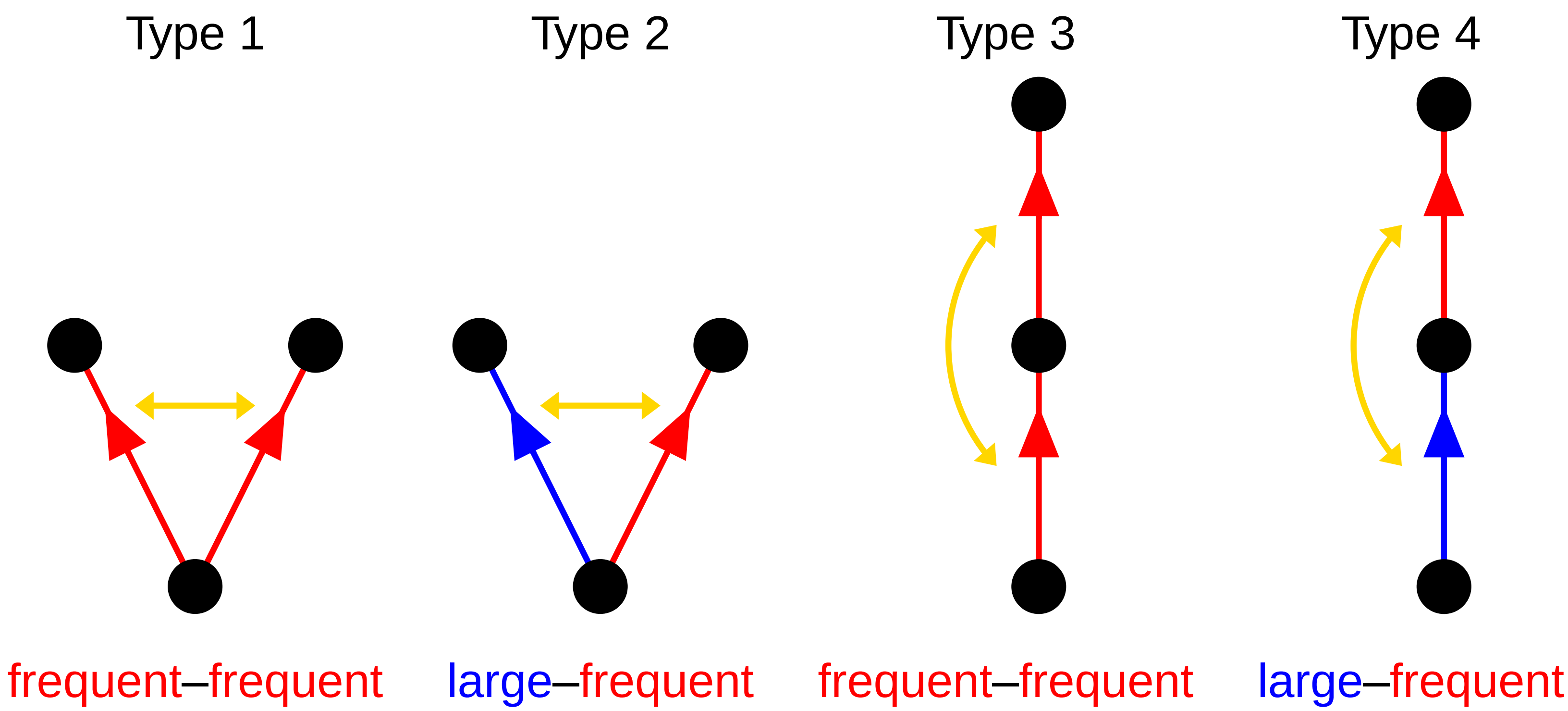
We split the frequent edges:



- At most a constant number of large edges enter each vertex.
- The graph is split at the frequent edges.
- Almost bounded degree: after deleting the degree one vertices only the large edges remain.
- Thus the PTAS can be used.

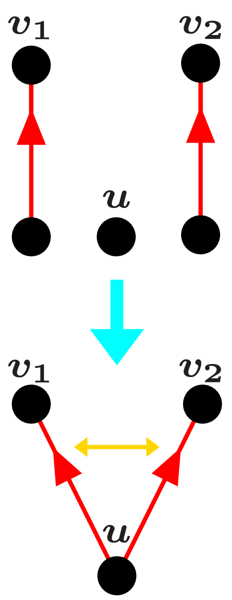
# Conflicts

The following types of conflicts can arise when we restore the frequent edges:

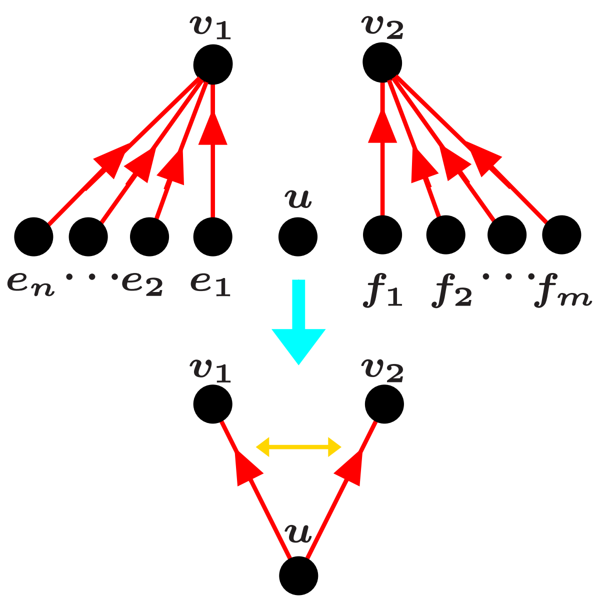



We resolve each type of conflict with only a small increase of the sum.

# Type 1 conflicts

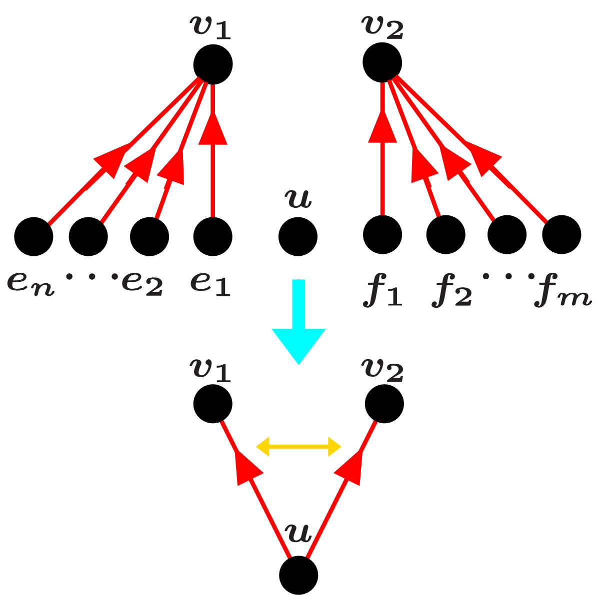


# *Type 1 conflicts*

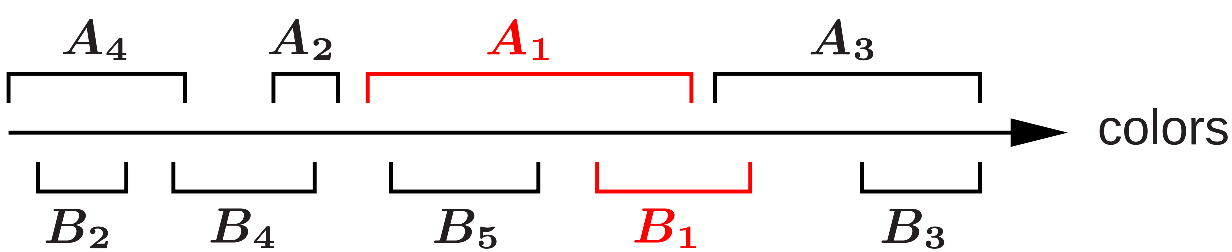



- Edges $e_1, \ldots, e_n$ are identical in every sense, the colors sets can be randomly reordered (similarly for $f_1, \ldots, f_m$).

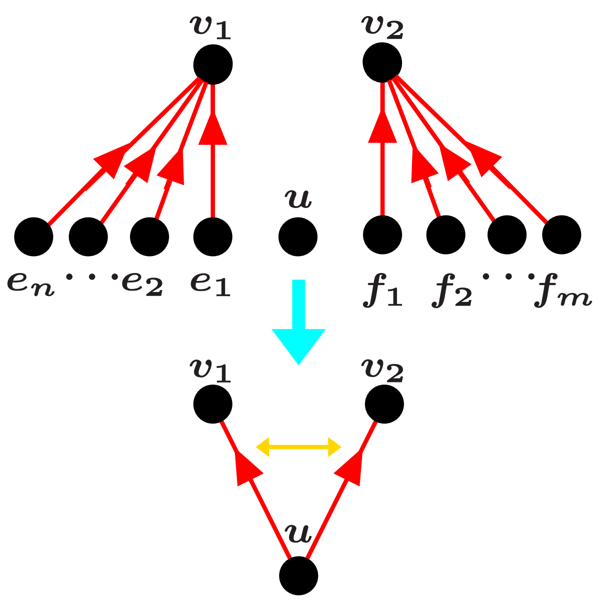
# Type 1 conflicts



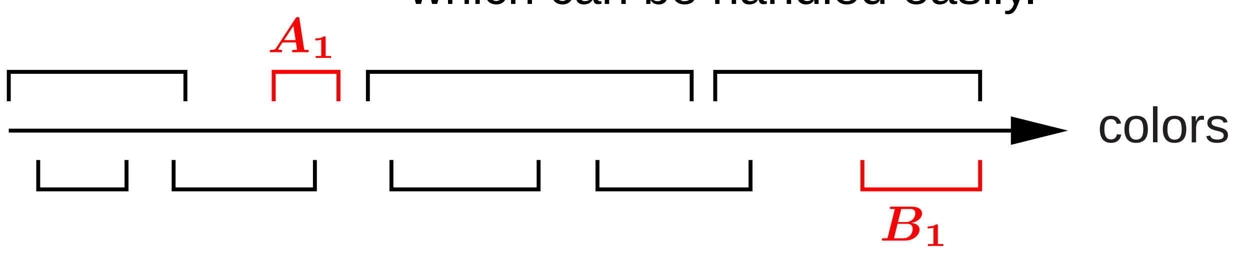

- Edges $e_1, \ldots, e_n$ are identical in every sense, the colors sets can be randomly reordered (similarly for $f_1, \ldots, f_m$).
- $A_i$: the range of colors assigned to $e_i$. The $A_i$'s do not overlap.



# Type 1 conflicts




- Edges $e_1, \ldots, e_n$ are identical in every sense, the colors sets can be randomly reordered (similarly for $f_1, \ldots, f_m$).
- $A_i$: the range of colors assigned to $e_i$. The $A_i$'s do not overlap.
- If $n$ and $m$ are large, then with high probability $A_1$ and $B_1$ do not intersect.
- With high probability there are only a few conflicts, which can be handled easily.



# Conclusions

- Problem: edge coloring version of minimum sum multicoloring on trees.
- A linear time PTAS for edge coloring partial $k$-trees. First for almost bounded degree partial $k$-trees.
- PTAS for the edge coloring of planar graphs using Baker's layering technique.
- General techniques useful for multiple problems.