Minimum sum multicoloring on the edges of trees

Dániel Marx
Budapest University of Technology and Economics
dmarx@cs.bme.hu

Workshop on Approximation and Online Algorithms
Budapest, 2003

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.

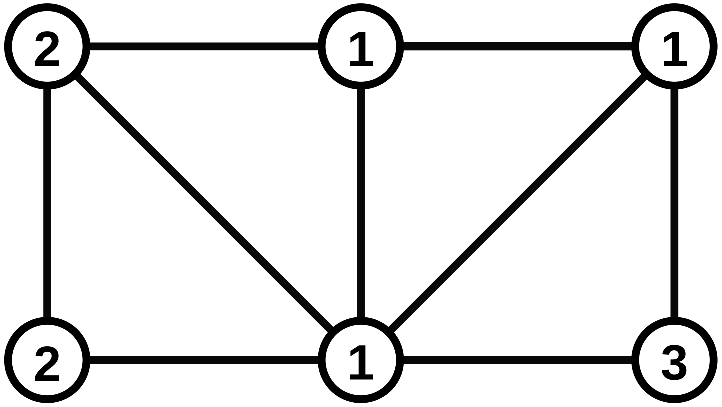
- **Goal:** Minimize $\sum_{v \in V} f(v)$, the **sum of the coloring**.

Minimum sum multicoloring

- **Given:** a graph $G(V, E)$, and demand function x: $V \rightarrow \mathbb{N}$
- **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**.

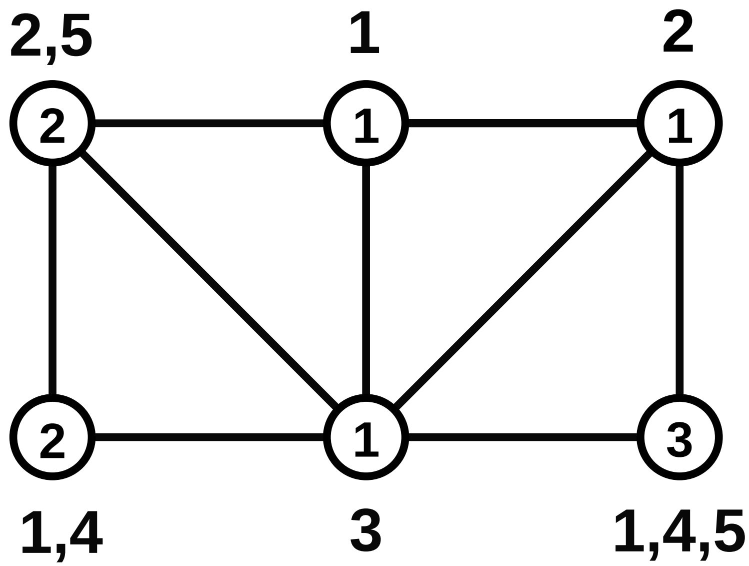


Minimum sum multicoloring

- **Given:** a graph $G(V, E)$, and demand function x: $V \rightarrow \mathbb{N}$
- **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**.

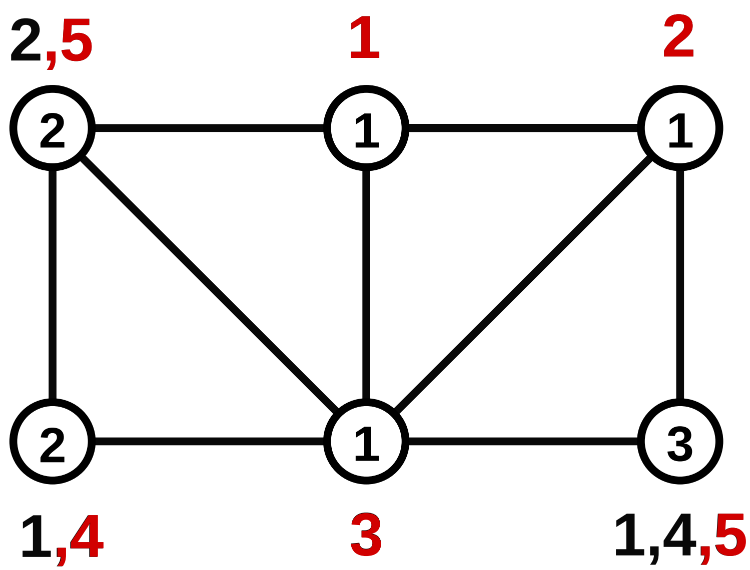


Minimum sum multicoloring

- **Given:** a graph $G(V, E)$, and demand function x: $V \rightarrow \mathbb{N}$
- **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

Known results

Special case: the chromatic sum problem: $x(v) = 1, \ \forall v \in V$

- **Trees:**
  - ⋆ polynomial time solvable if every demand is 1 [Kubicka, 1989],
  - ⋆ sum multicoloring is **NP**-hard for binary trees [Marx, 2002]
  - ⋆ $(1 + \varepsilon)$-approximation for sum multicoloring [Halldórsson et al., 1999]

- **Partial k-trees:**
  - ⋆ $(1 + \varepsilon)$-approximation for sum multicoloring [Halldórsson and Kortsarz, 1998]

- **Bipartite graphs:**
  - ⋆ **APX**-hard, even if every demand is 1 [Bar-Noy and Kortsarz, 1998]
  - ⋆ 1.5-approximation for sum multicoloring [Bar-Noy et al., 1998]

Edge coloring version

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

Application: scheduling dedicated biprocessor tasks

Each task requires the simultaneous work of two preassigned processors for a given number of time slots. Goal: minimize the sum of completion times.

vertices	$\Longleftrightarrow$	processors
edges	$\Longleftrightarrow$	jobs
demand	$\Longleftrightarrow$	length of job
colors	$\Longleftrightarrow$	time slots

Preemptive scheduling: jobs can be interrupted and continued later

Bipartite graphs: processors are divided into clients and servers

Edge coloring results

- **Known results:**
  - ★ Polynomial time solvable on trees with demand 1 [Giaro and Kubale 2000]
  - ★ **NP**-hard on bipartite graphs even if every demand is 1 [Giaro and Kubale 2000]
  - ★ 1.796-approximation for bipartite graphs with demand 1 [Halldórsson et al.]
  - ★ 2-approximation for general graphs and general demand [Bar-Noy et al., 2000]

Edge coloring results

- **Known results:**
  - ⋆ Polynomial time solvable on trees with demand 1 [Giaro and Kubale 2000]
  - ⋆ **NP**-hard on bipartite graphs even if every demand is 1 [Giaro and Kubale 2000]
  - ⋆ 1.796-approximation for bipartite graphs with demand 1 [Halldórsson et al.]
  - ⋆ 2-approximation for general graphs and general demand [Bar-Noy et al., 2000]

- **Our results:**
  - ⋆ **NP**-hard on trees even if every demand is 1 or 2
  - ⋆ $(1+\varepsilon)$-approximation on trees with arbitrary demand

Scaling the demand

Theorem: If the graph is a tree, then multiplying the demand of each edge by integer q multiplies the minimum sum by *exactly* q.

Scaling the demand

Theorem: If the graph is a tree, then multiplying the demand of each edge by integer q multiplies the minimum sum by *exactly* q.

$\Rightarrow$ The problem can be solved in polynomial time on trees if every edge has the same demand

Scaling the demand

Theorem: If the graph is a tree, then multiplying the demand of each edge by integer q multiplies the minimum sum by *exactly* q.

$\Rightarrow$ The problem can be solved in polynomial time on trees if every edge has the same demand

$\Rightarrow$ Increasing the demand to the next power of $(1+\varepsilon)$ increases the sum by at most a factor of $(1+\varepsilon)$ $\Rightarrow$ we can assume that each demand is of the form $(1+\varepsilon)^i$

Bounded degree trees

Theorem: Minimum sum edge coloring admits a linear time PTAS in bounded degree trees.

Method:

The line graph of a tree with max degree d is a partial $(d-1)$-tree, hence the PTAS of Halldórsson and Kortsarz can be used.

In a partial k-tree we can compute a polynomial number of color sets for each vertex such that there is a good approximate solution using only these sets $\Rightarrow$ PTAS with standard dynamic programming

Bounded degree trees

Theorem: Minimum sum edge coloring admits a linear time PTAS in bounded degree trees.

Method:

The line graph of a tree with max degree d is a partial $(d-1)$-tree, hence the PTAS of Halldórsson and Kortsarz can be used.

In a partial k-tree we can compute a polynomial number of color sets for each vertex such that there is a good approximate solution using only these sets $\Rightarrow$ PTAS with standard dynamic programming

Bounded degree trees: In edge coloring bounded degree trees, a *constant* number of color sets is sufficient for each edge $\Rightarrow$ linear time PTAS

Bounded degree trees

Theorem: Minimum sum edge coloring admits a linear time PTAS in bounded degree trees.

Method:

The line graph of a tree with max degree d is a partial $(d-1)$-tree, hence the PTAS of Halldórsson and Kortsarz can be used.

In a partial k-tree we can compute a polynomial number of color sets for each vertex such that there is a good approximate solution using only these sets $\Rightarrow$ PTAS with standard dynamic programming

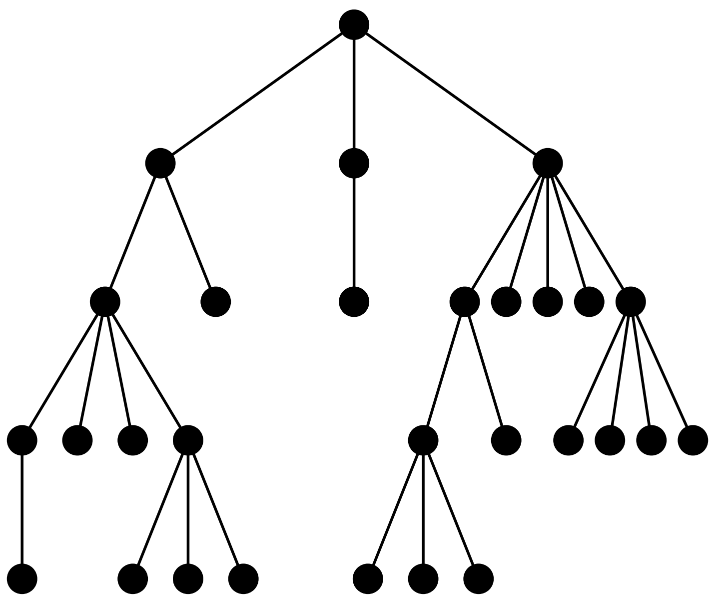
Bounded degree trees: In edge coloring bounded degree trees, a *constant* number of color sets is sufficient for each edge $\Rightarrow$ linear time PTAS

Almost bounded degree trees: trees that have bounded degree after deleting the degree 1 nodes. Algorithm works for such trees as well.

General case

Theorem: Linear time PTAS for general trees.

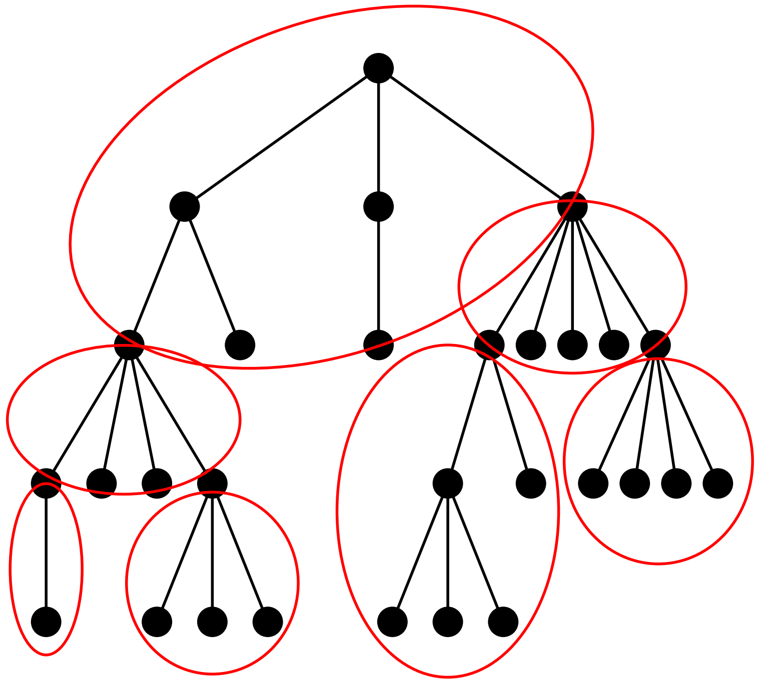
- Partition the tree into almost bounded degree subtrees
- Use the PTAS for subtrees
- Merge the colorings of the subtrees to a coloring of the whole tree



General case

Theorem: Linear time PTAS for general trees.

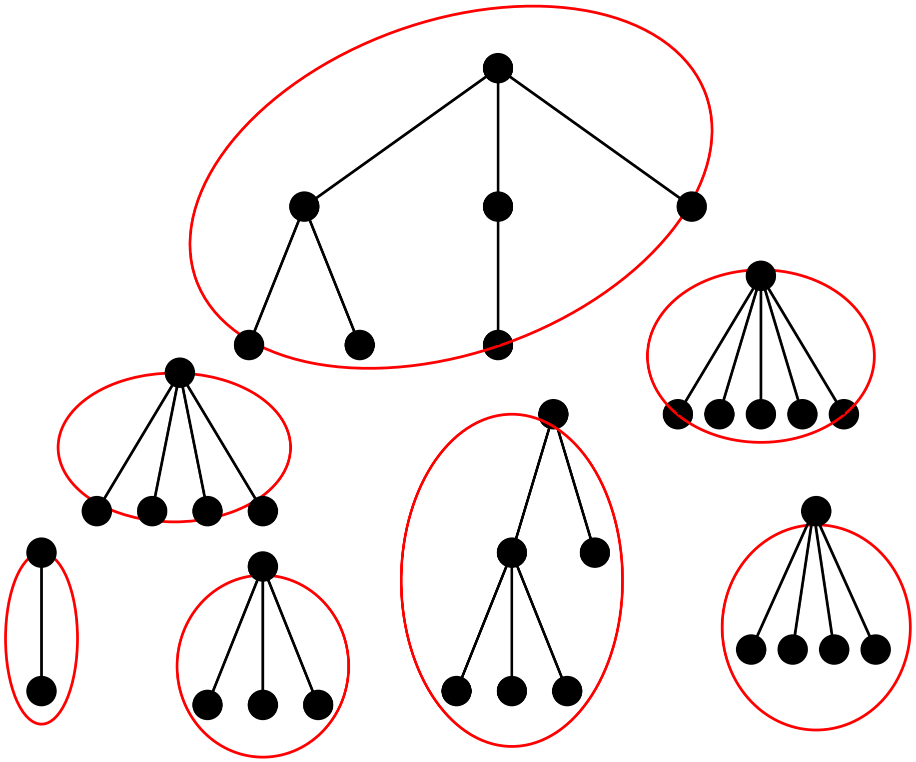
- Partition the tree into almost bounded degree subtrees
- Use the PTAS for subtrees
- Merge the colorings of the subtrees to a coloring of the whole tree



General case

Theorem: Linear time PTAS for general trees.

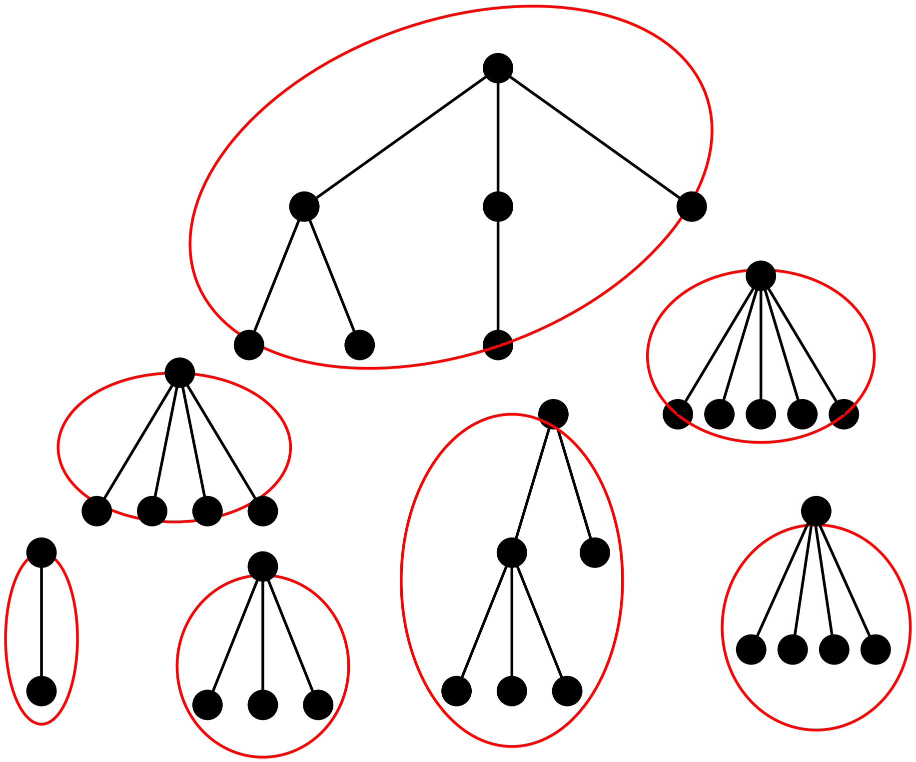
- Partition the tree into almost bounded degree subtrees
- Use the PTAS for subtrees
- Merge the colorings of the subtrees to a coloring of the whole tree



General case

Theorem: Linear time PTAS for general trees.

- Partition the tree into almost bounded degree subtrees
- Use the PTAS for subtrees
- Merge the colorings of the subtrees to a coloring of the whole tree



?

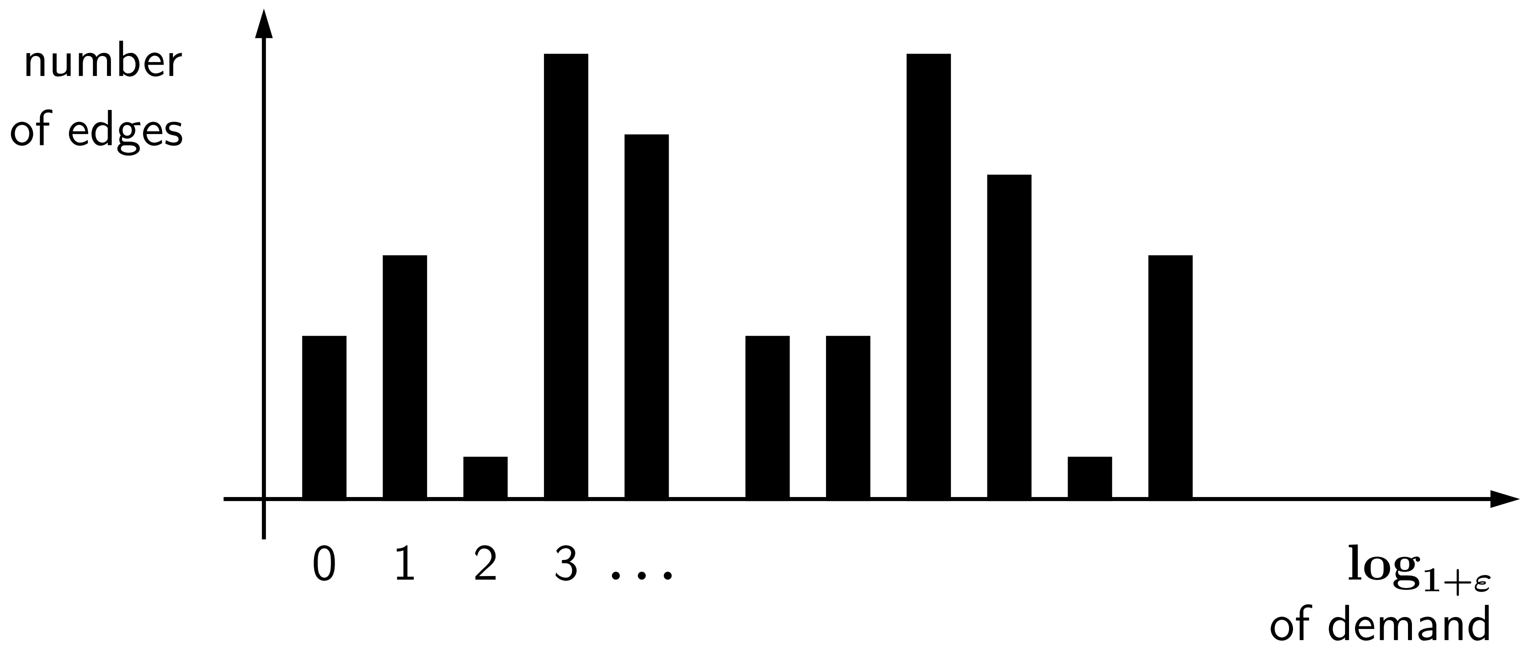
How to partition the tree?
How to resolve the conflicts when merging the colorings?

The Small, the Large, and the Frequent

The child edges of a given 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:

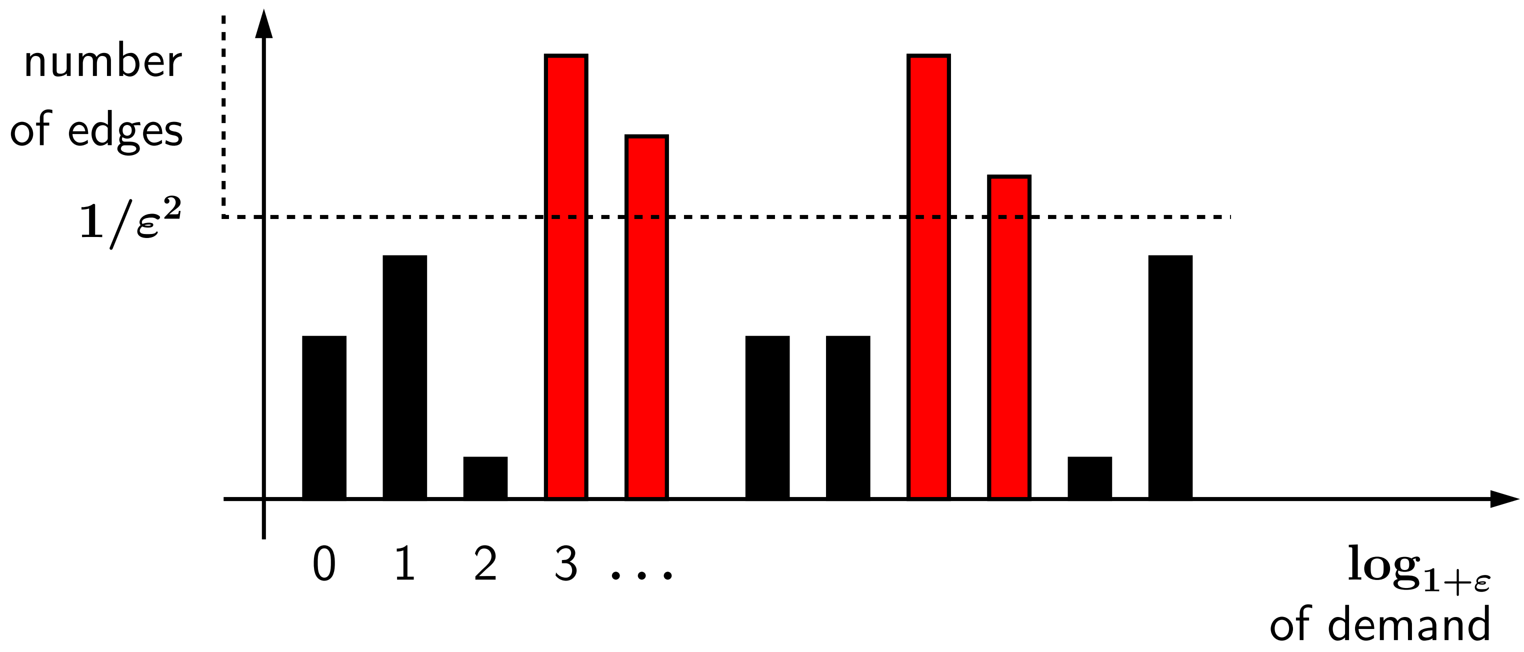


The Small, the Large, and the Frequent

The child edges of a given 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:

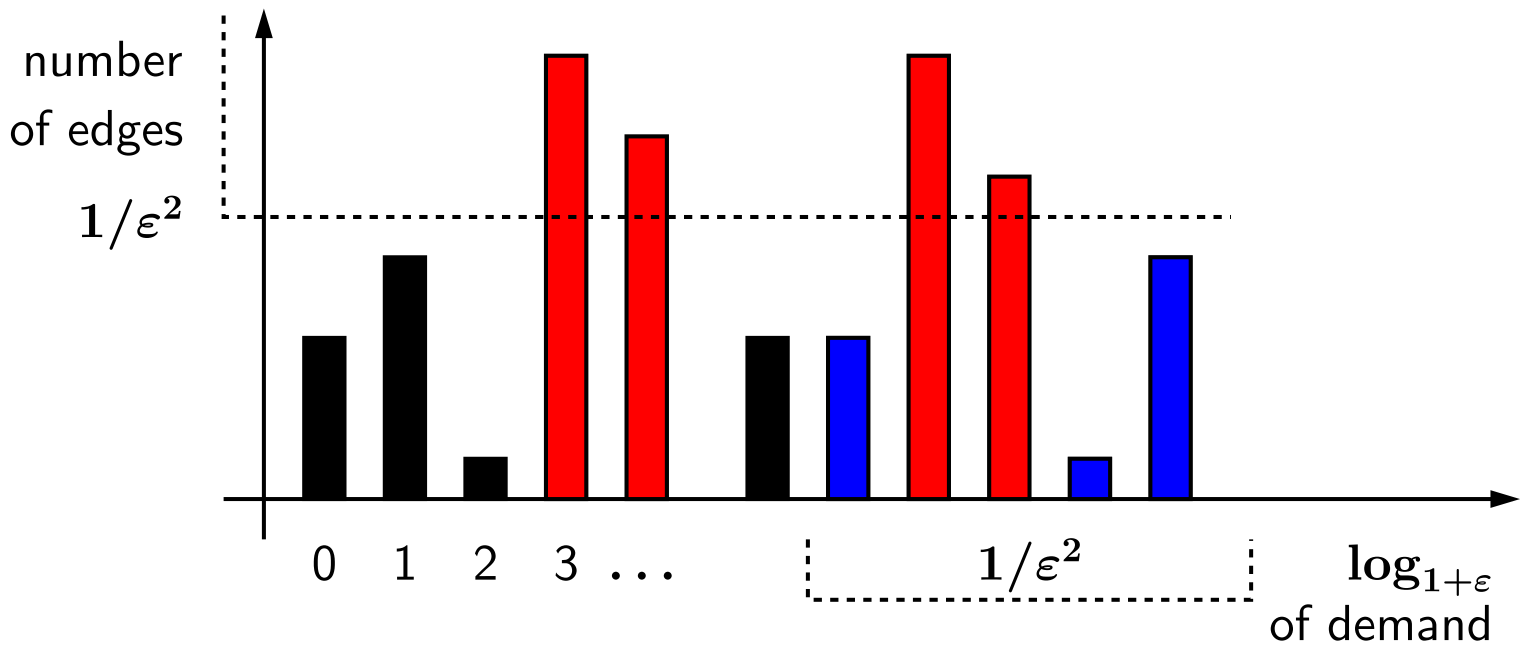


The Small, the Large, and the Frequent

The child edges of a given 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:

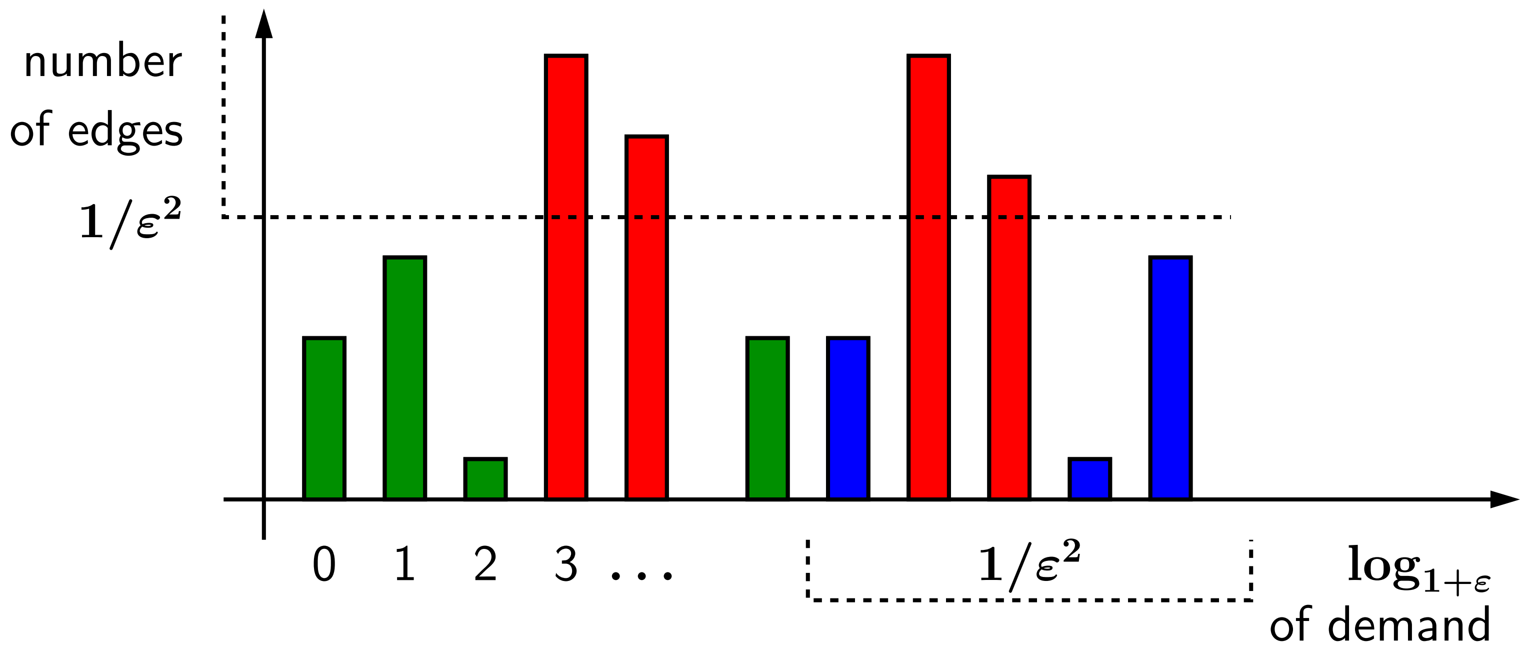


The Small, the Large, and the Frequent

The child edges of a given 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:

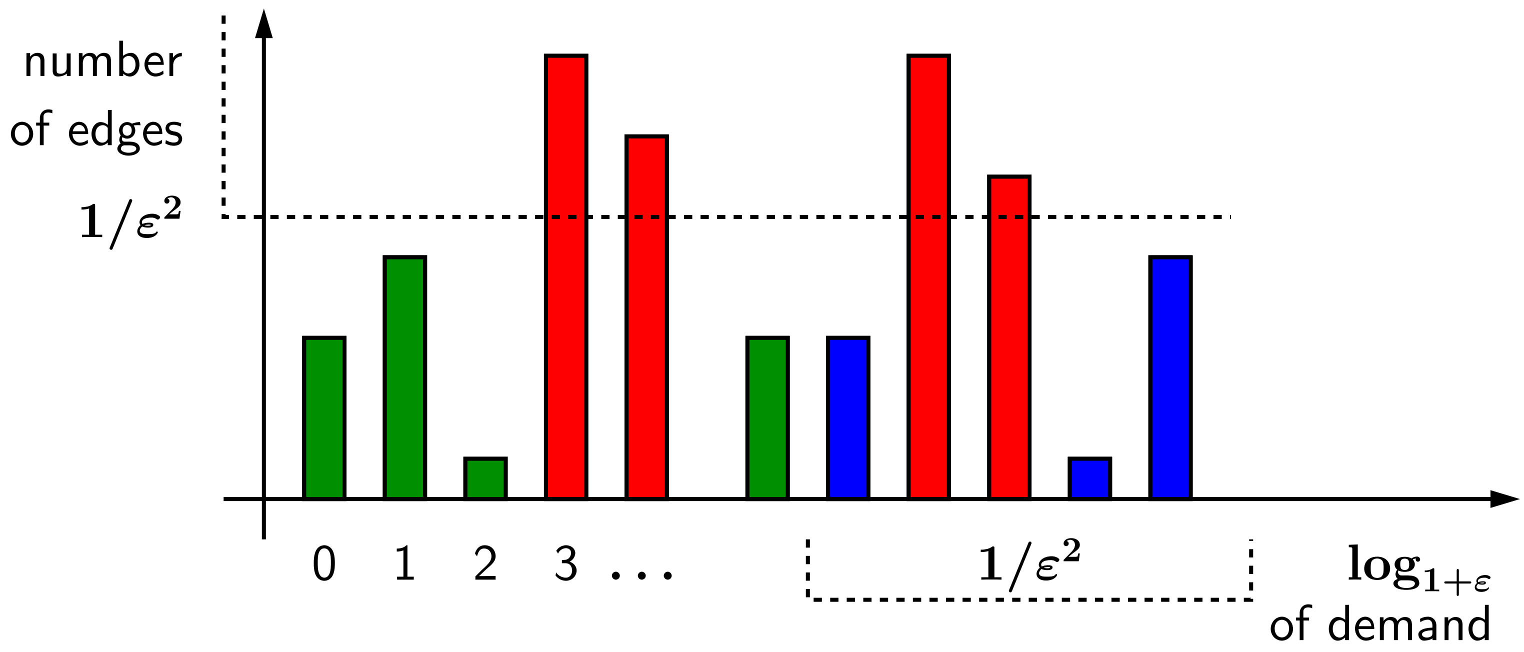


The Small, the Large, and the Frequent

The child edges of a given 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:



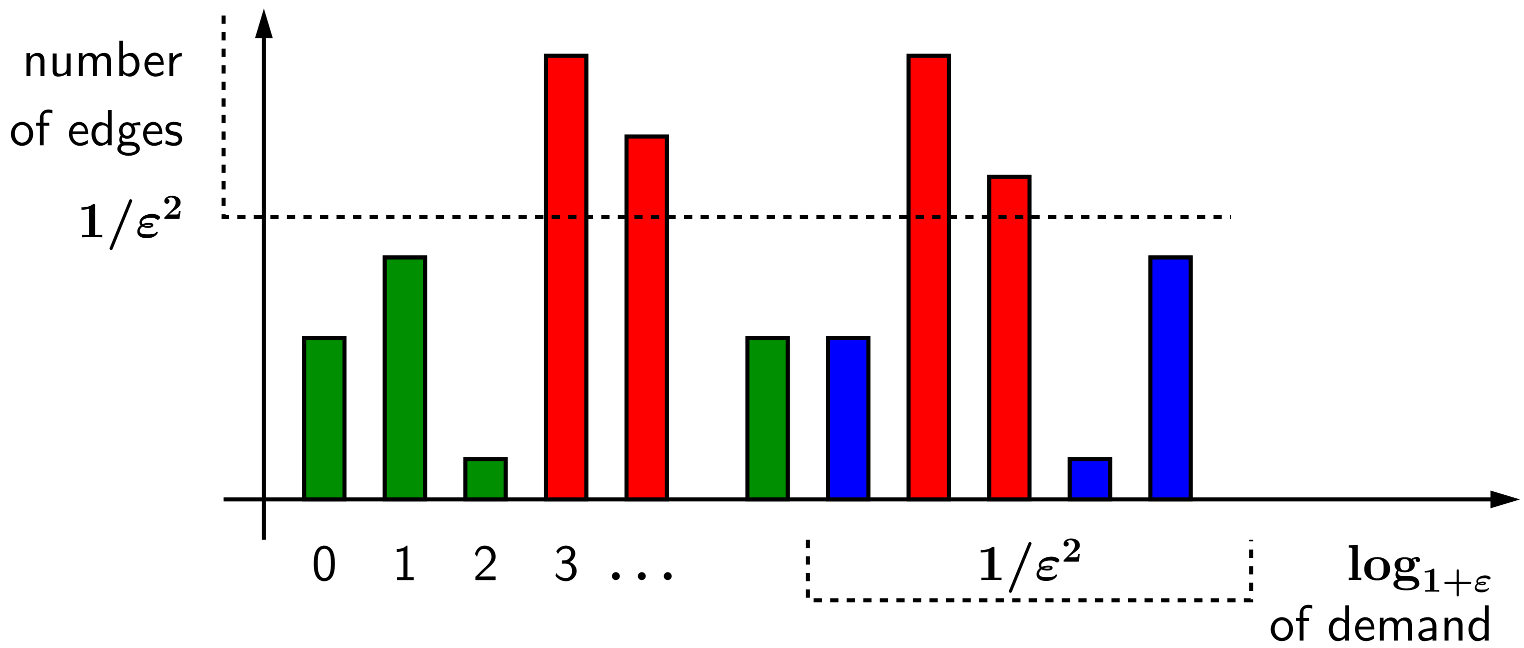
Total demand of the small edges is very small, they can be thrown away.

The Small, the Large, and the Frequent

The child edges of a given 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:

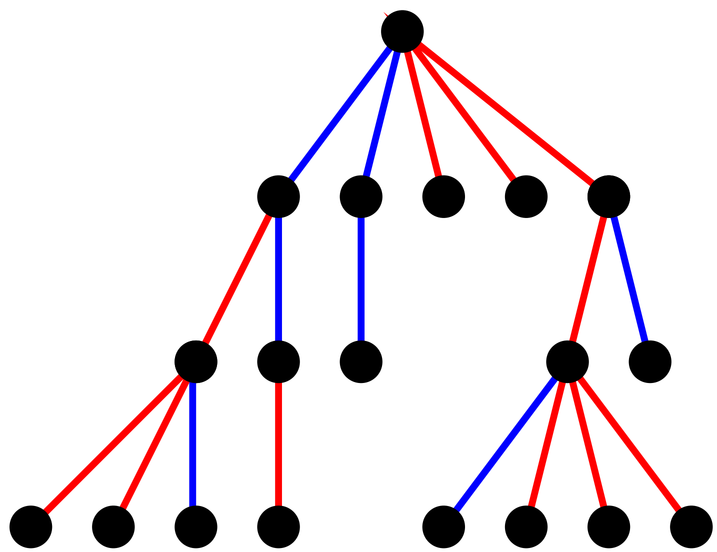


Total demand of the small edges is very small, they can be thrown away.

Each node has at most a constant number ($\leq 1/\varepsilon^4$) of large child edges.

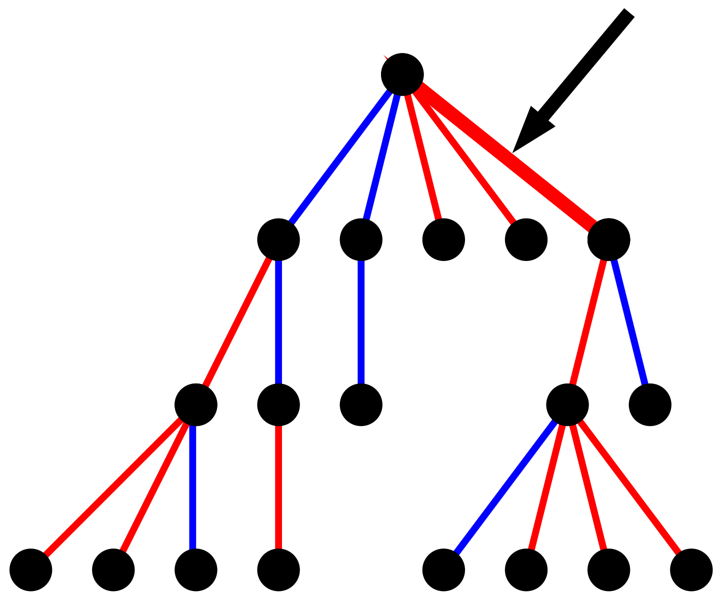
Partitioning the tree

The tree is split at the frequent edges:



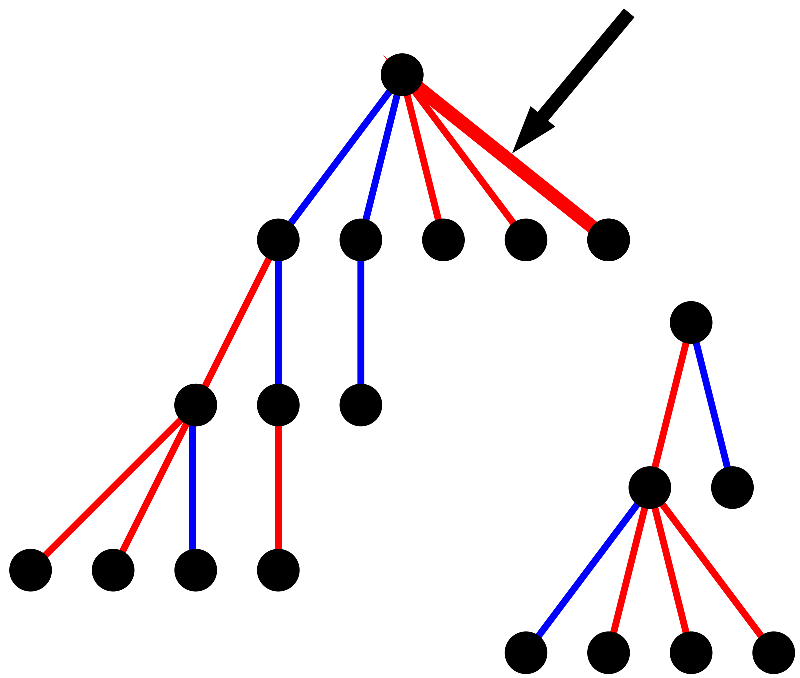
Partitioning the tree

The tree is split at the frequent edges:



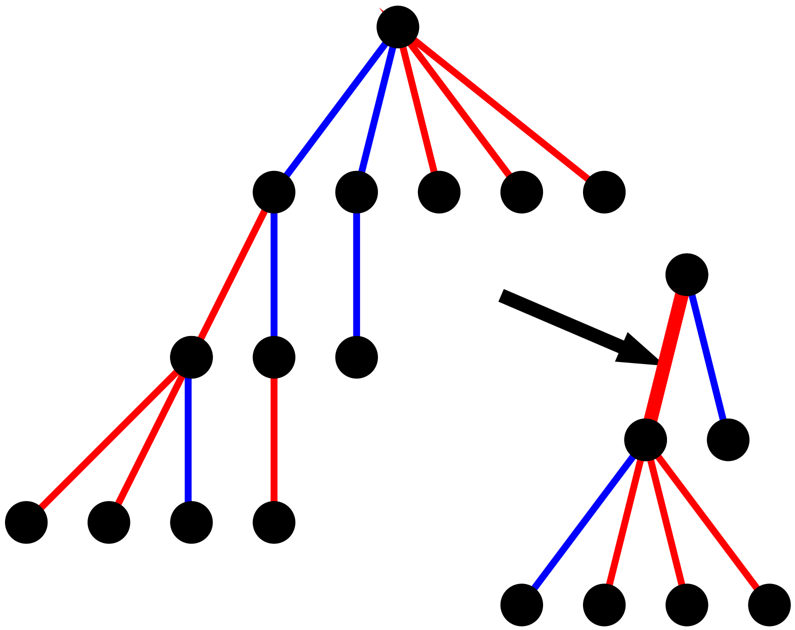
Partitioning the tree

The tree is split at the frequent edges:



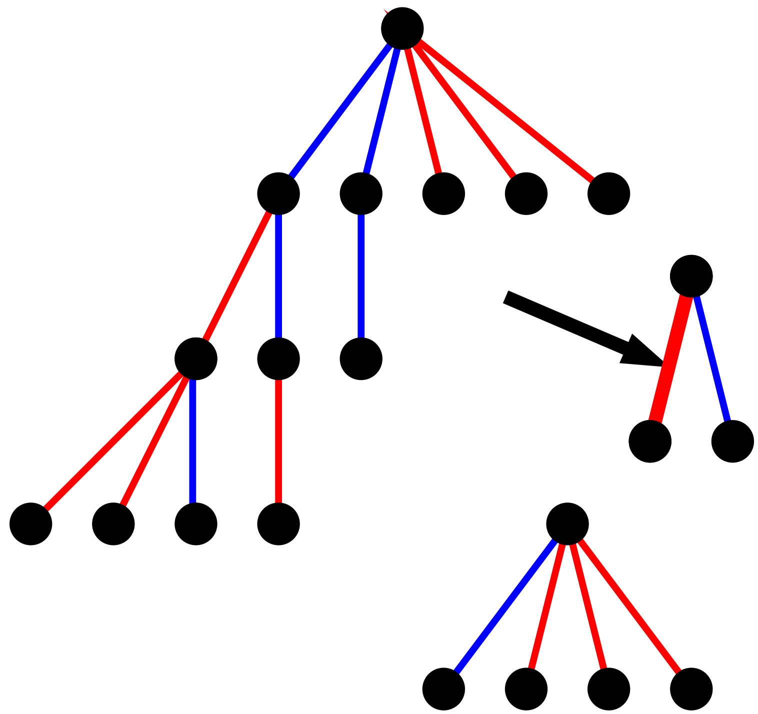
Partitioning the tree

The tree is split at the frequent edges:



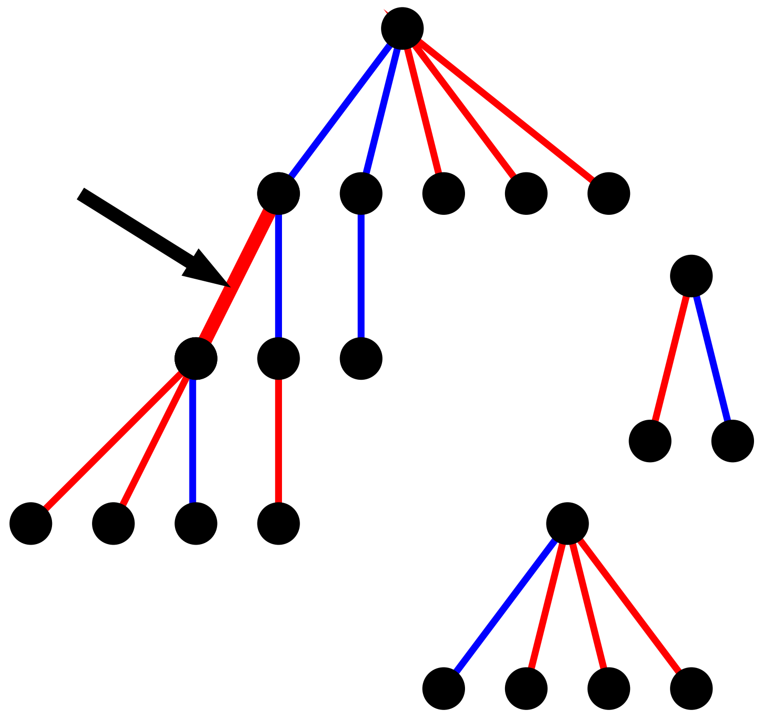
Partitioning the tree

The tree is split at the frequent edges:



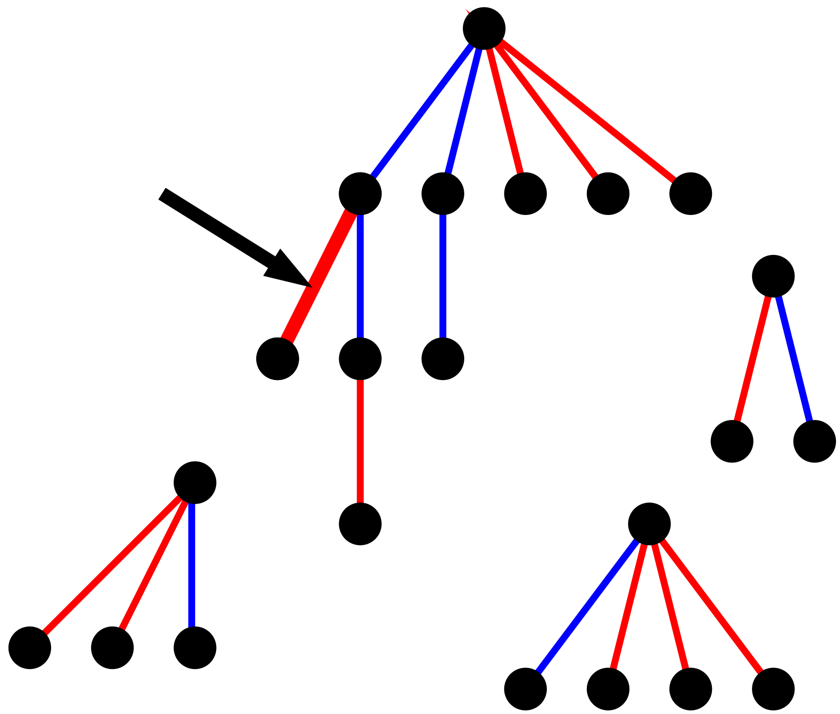
Partitioning the tree

The tree is split at the frequent edges:



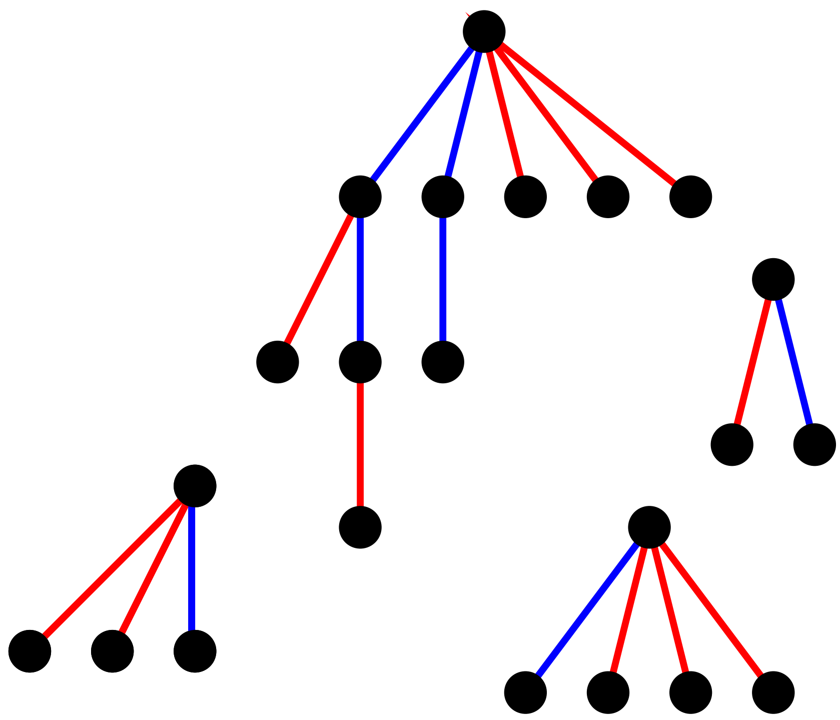
Partitioning the tree

The tree is split at the frequent edges:



Partitioning the tree

The tree is split at the frequent edges:



Claim: Each subtree is an almost bounded degree tree $\Rightarrow$ PTAS can be used

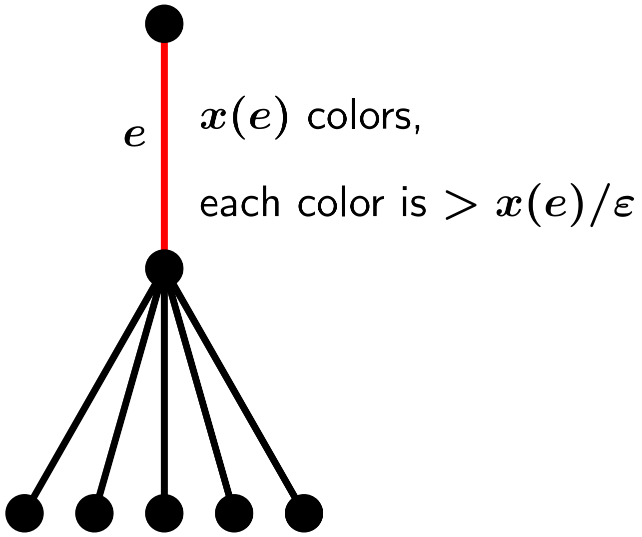
Proof:

- Deleting the degree 1 nodes deletes every frequent edge
- Only the large edges remain
- Each node has at most a constant number of large child edges

How to merge the colorings?

Shifting the frequent edges: We modify the coloring such that each frequent edge e uses only colors above $x(e)/\varepsilon$. This can be done with only a small increase of the sum.

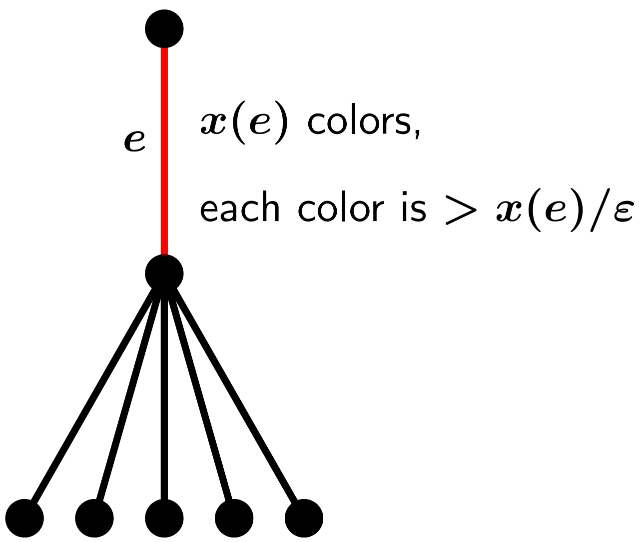
Resolving the conflicts: remove the conflicting colors from the child edges.



How to merge the colorings?

Shifting the frequent edges: We modify the coloring such that each frequent edge e uses only colors above $x(e)/\varepsilon$. This can be done with only a small increase of the sum.

Resolving the conflicts: remove the conflicting colors from the child edges.



We remove at most $x(e)$ colors, each of them is greater than $x(e)/\varepsilon$.

To replace these colors, it is easy to find $x(e)$ unused colors below $x(e)/\varepsilon$.

Conclusions

- Problem: edge coloring version of minimum sum multicoloring on trees.
- PTAS for vertex coloring partial k-trees implies a PTAS for edge coloring bounded degree trees. Linear time PTAS with additional techniques.
- Linear time PTAS for general trees uses the algorithm for bounded degree trees as a subroutine
- Minimum sum edge multicoloring is **NP**-hard on trees.