

Master's Thesis

An optimal minimum spanning tree algorithm

Claus Andersen
Aarhus University
December 19, 2008



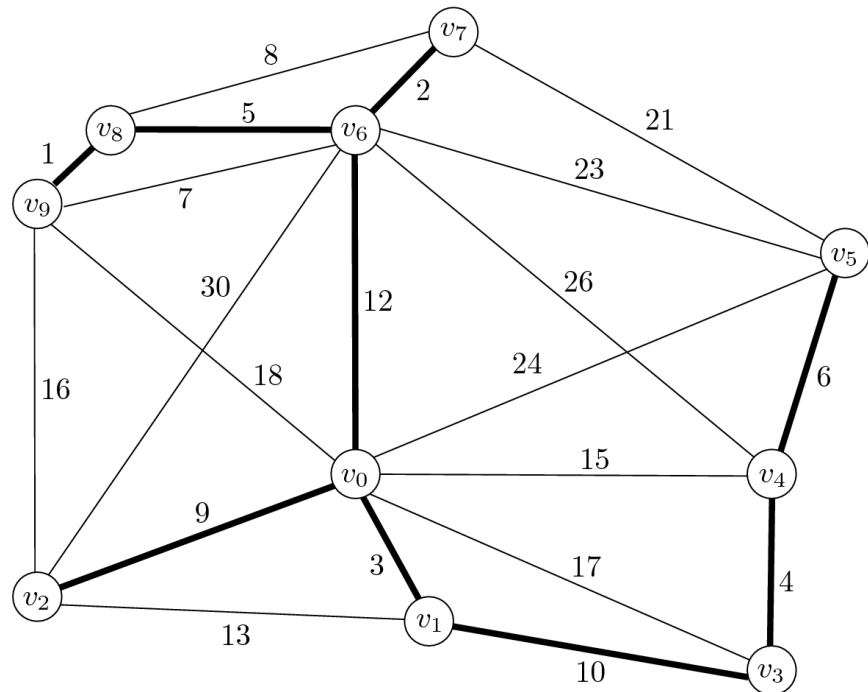
Programme

- Introduction to MST
- Overview of MST algorithms
- The optimal MST algorithm
 - Brief analysis
- Experimental results
 - Soft heap versions
- Conclusion



Minimum spanning tree

- Weighted undirected graph
- Spanning tree with minimum total weight
- Cycle property
- Cut property
- Uniqueness



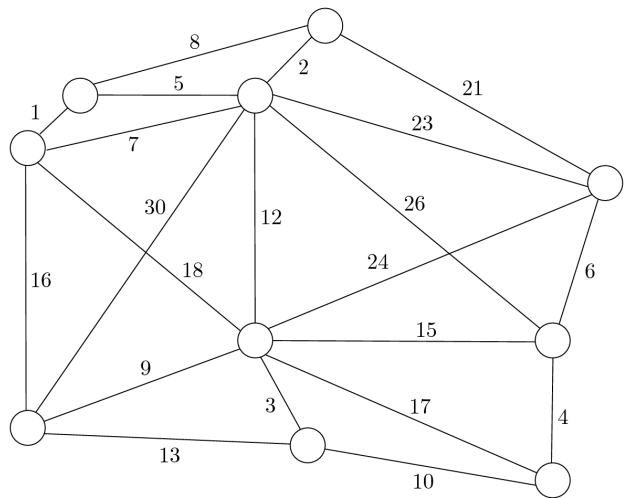
MST History

- Borůvka, 1926 – Electrical network
- Jarník, 1930 (Prim and Dijkstra)
- Fredman and Tarjan, 1987 – Fibonacci Heaps,
 $O(m \cdot (\log^*(n) - \log^*(m/n)))$
- Chazelle, 2000, $O(m \cdot \alpha(m, n))$ – Best upper bound
- **Pettie and Ramachandran, 2002,**
Order of "optimal" time

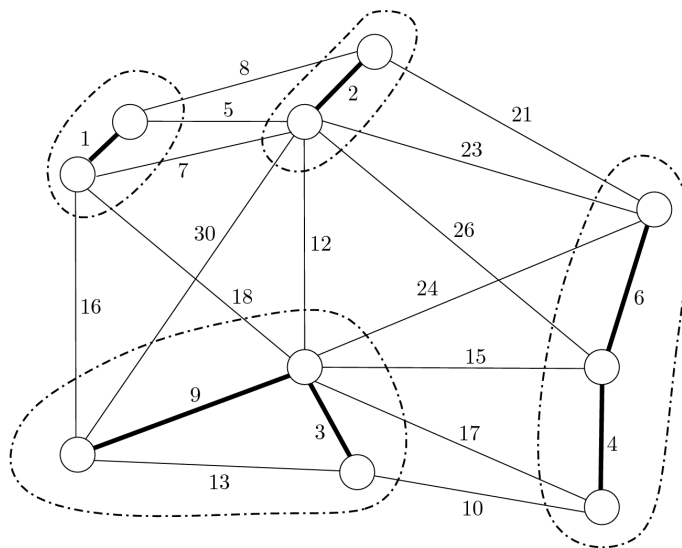
Borůvka's algorithm

- Borůvka step:
 - For each vertex: Add the lightest incident edge to MST
 - Contract graph along MST edges
- Step time: $O(m)$
- Total time: $O(\min\{m \cdot \log(n), n^2\})$
 - Very sparse graphs: $O(m)$

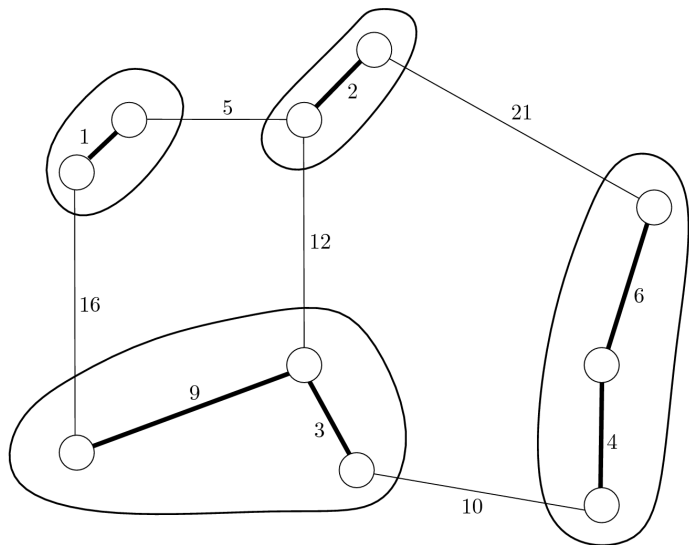
1



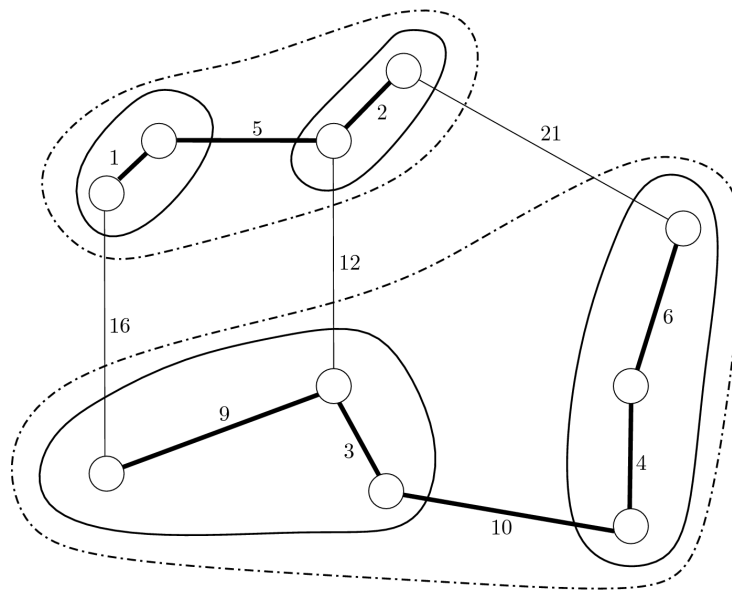
2



3



4

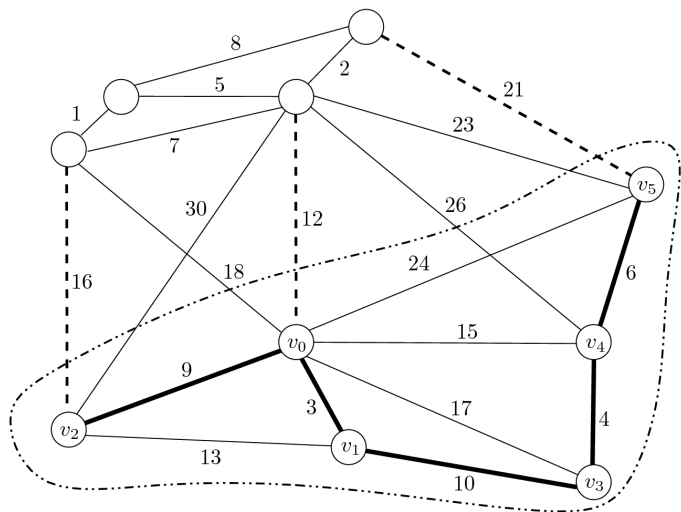


6

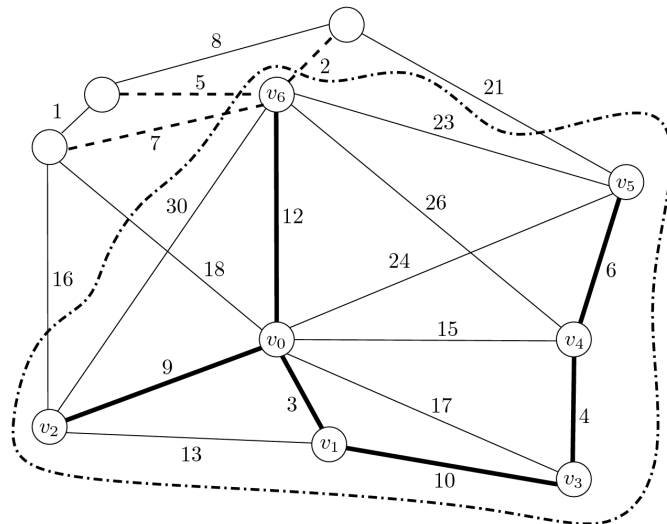
The DJP algorithm

- Repeatedly augments a tree, T , of MST edges
- Priority queue (PQ) of edges connecting T to neighbouring vertices.
 - Key is edge weight
- Time depends on PQ operation times
- Time, fibonacci heap: $O(n \cdot \log(n) + m)$

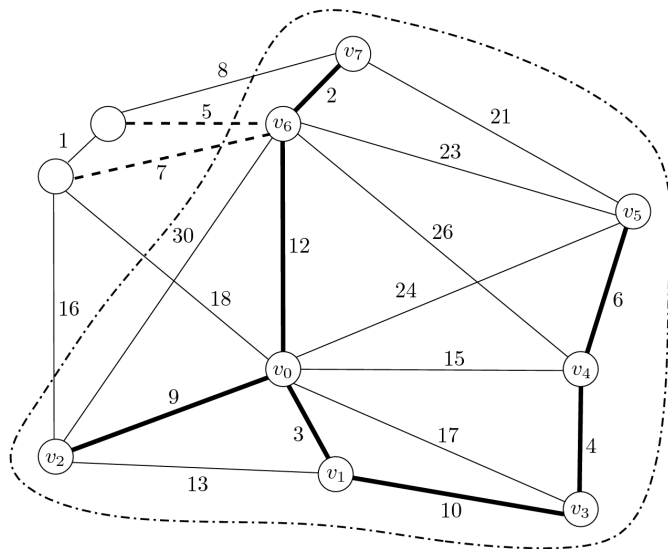
1



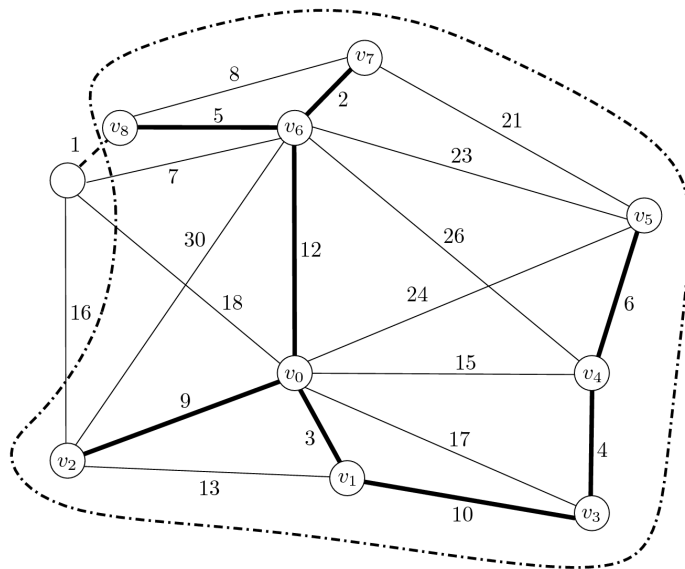
2



3



4



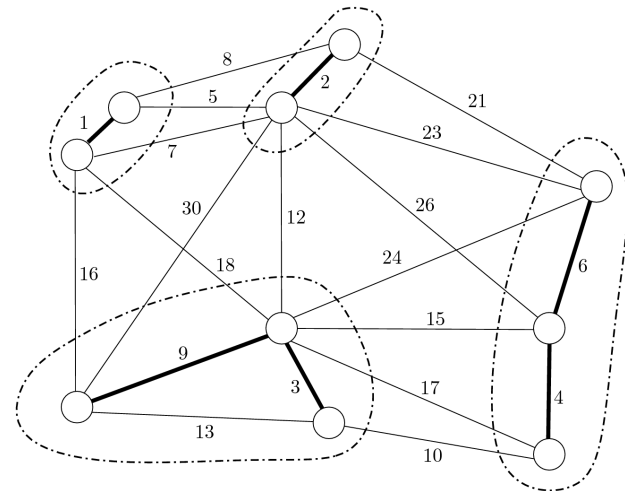
8

Fredman & Tarjan ("Dense Case")

- Dense Case pass:
 - Input: t vertices, m' edges. Heap bound: $k=2^{2m/t}$
 - Repeatedly run DJP with heap bound k
 - Contract graph along MST edges
- First pass: $k=2^{2m/n}$, Last pass: $k \geq n$
- Number of trees: $t' \leq 2m'/k$
- Next heap bound: $k' = 2^{2m/t'} \geq 2^{2mk/2m'} \geq 2^k$

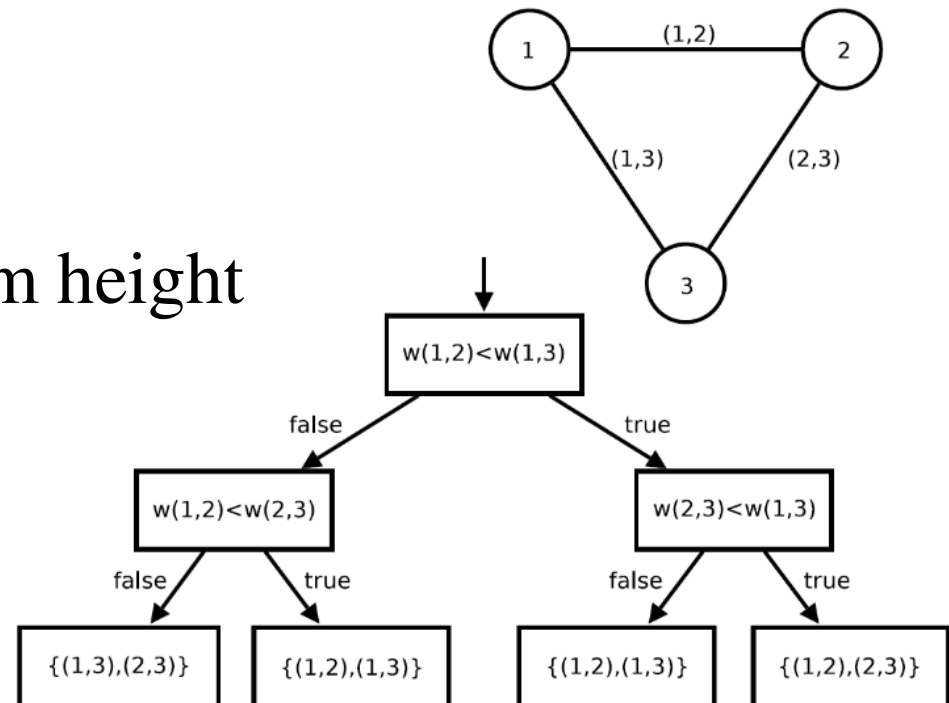
Fredman & Tarjan ("Dense Case")

- Beta function: $\beta(m, n) = \min\{ i : \log^{(i)}(n) \leq m/n \}$
- Contracted graph:
 - $n' \leq n / \log^{(3)}(n)$ vertices and $m' \leq m$ edges ($n \leq m$)
 - Nominal density: $m/n' \geq m \cdot \log^{(3)}(n) / n \geq \log^{(3)}(n)$
 - $\beta(m, n') \leq 3$ (passes)



MST decision tree (DT)

- Rooted binary tree hardwired to fixed graph
 - Internal node: Edge-weight comparison (true/false)
 - Leaf node: MST edge set
- Optimal DT
 - Correct DT with minimum height
 - Unknown height, but an upper bound exists



MST decision trees (2)

- Brute force searching for graphs with $\leq r$ vertices
 - Generate all possible DT's with height r^2
 - For each graph G :
 - Run DJP algorithm for each edge-weight permutation
 - Find an optimal DT for G
 - Time: $O(2^{2^{(r^2+o(r))}})$ – Very slow or very small r !
 - Time for $r = \log^{(3)}(n)$: $o(n)$
 - In practice: $r \leq 3$

Soft heap – Approximate PQ

- Chazelle, 2000 – Utilized by his MST algorithm
- Kaplan & Zwick, 2009 – Simplified version
- Artificially raising the key of some elements
- Initialized with error parameter: $0 < \varepsilon < 1/2$
- Soft heap instance after n insertions:
 - Maximum εn corrupted elements
 - Time, insert: $O(\log(1/\varepsilon))$, other: $O(1)$

Key lemma

- Some number of DJP steps using a soft heap
 - Some edges corrupted (potentially deleted to late): M
 - "DJP-contractible" sub graph induced by DJP tree: C
 - Edges in M with one endpoint in C : M_C
- $\text{MSF}(G) \subseteq \text{MSF}(C) \cup \text{MSF}(G \setminus C - M_C) \cup M_C$
- Prove, using the cycle property, that edges not in the superset are not in $\text{MSF}(G)$

Partition procedure

- Input: Graph G , partition *maxsize*, error rate ε
- Repeatedly grow DJP-contractable partitions, C_i , in G from a live vertex using a fresh soft heap
- Output: Partitions, C , and corrupted edges, M
- Key lemma applied multiple times:

$$\text{MSF}(G) \subseteq \bigcup_{j=1}^k \text{MSF}(C_j) \cup \text{MSF}\left(G \setminus \bigcup_{j=1}^k C_j - \bigcup_{j=1}^k M_{C_j}\right) \cup \bigcup_{j=1}^k M_{C_j}$$

The optimal MST algorithm

- Precomputation: Build decision trees for graphs with $\leq \log^{(3)}(n)$ vertices

Algorithm 15.1: $\text{optimalMST}(G)$ - The optimal MST algorithm

Input : Connected graph G .
Output: The MST of G .
if $E(G) = \emptyset$ **then return** $\emptyset$
 $r \leftarrow \lceil \log^{(3)}(|V(G)|) \rceil$
 $(M, C) \leftarrow \text{Partition}(G, r, \varepsilon)$
 $F \leftarrow \text{DecisionTree}(C)$
Let $k \leftarrow |C|$ and $F = \{F_1, \dots, F_k\}$
 $G_a \leftarrow G \setminus (F_1 \cup \dots \cup F_k) - M$
 $F_0 \leftarrow \text{DenseCase}(G_a, |E(G)|)$
 $G_b \leftarrow F_0 \cup F_1 \cup \dots \cup F_k \cup M$
 $(T', G_c) \leftarrow \text{Boruvka2}(G_b)$
 $T \leftarrow \text{OptimalMST}(G_c)$
return $(T \cup T')$

Analysis 1/4

- Error rate $\varepsilon = 1/8$
- Partition: $O(m \cdot \log(1/\varepsilon)) = O(m)$
- Decision tree: Unknown, but order of optimal for each partition
- Dense Case: $O(m)$ for G_a
- Boruvka2: $O(m)$
 - $n_c \leq n/4, \quad m_c \leq m/2$

Analysis 2/4

- Specific graph H :
 - Optimal number of comparisons: $\mathcal{T}^*(H)$
 - Class of graphs with n vertices and m edges:
$$\mathcal{T}^*(m, n) = \max \{ \mathcal{T}^*(H) : |V(H)| = n \wedge |E(H)| = m \}$$
$$\mathcal{T}^*(H) \leq \mathcal{T}^*(|V(H)|, |E(H)|)$$
- Total time:
$$T(m, n) \leq \sum_i c_1 \mathcal{T}^*(C_i) + T\left(\frac{m}{2}, \frac{n}{4}\right) + c_2 m$$

Analysis 3/4

- Let H be the union of grown partitions C_i
- Lemma 15.1: $\text{MSF}(H) = \bigcup_{i=1}^k \text{MSF}(C_i)$
- Corollary of lemma 15.3:

$$\sum_i \mathcal{T}^*(C_i) = \mathcal{T}^*(H) \leq \mathcal{T}^*(m_H, n_H) \leq \mathcal{T}^*(m, n)$$

Analysis 4/4

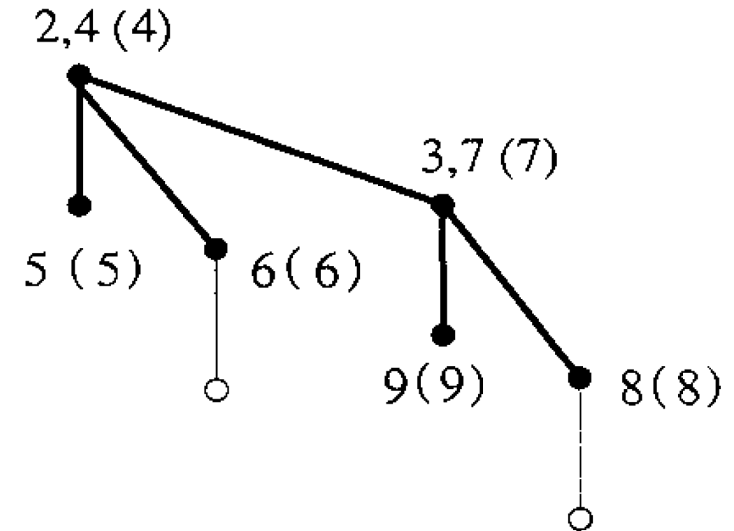
- For $c \geq 2c_1 + 4c_2$:

$$\begin{aligned} T(m, n) &\leq c_1 T^*(m, n) + T\left(\frac{m}{2}, \frac{n}{4}\right) + c_2 m \\ &\leq c T^*(m, n) \end{aligned}$$

- Deterministic complexity: $O(T^*(m, n))$
 - Decision tree complexity
- Linear time for realistic input

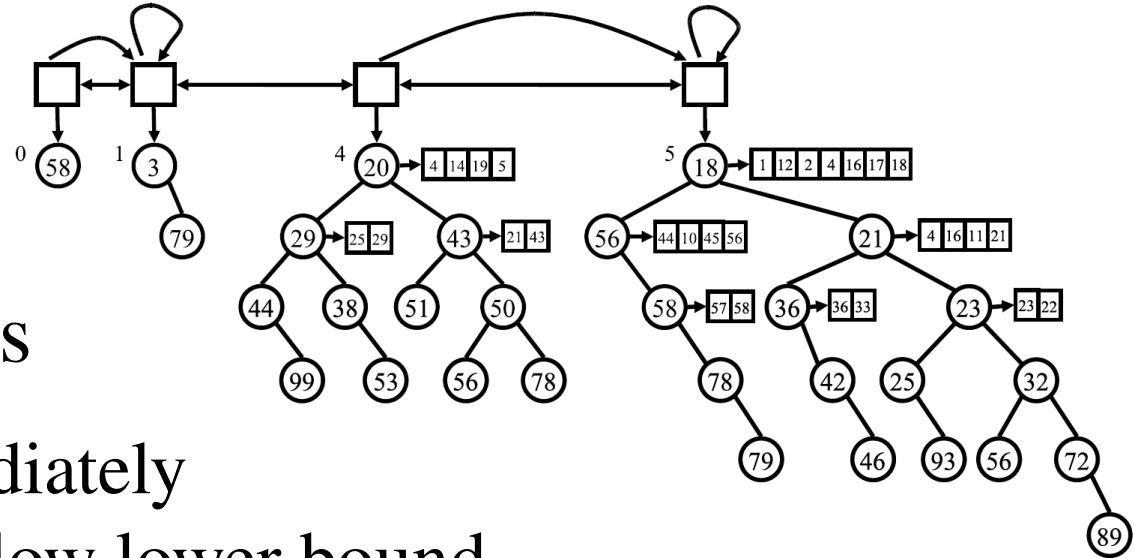
Soft heap: Chazelle

- Partial binomial trees
 - Represented as binary trees
- Clean insert
- Lazy deleteMin
 - Delayed clean-up, if item list is empty
 - Remelding if root has too few children
 - Sifting (relinking), otherwise – Maybe multiple times



Soft heap: Kaplan & Zwick

- Binary trees
- Insert with sifting
- Item list size bounds
 - Lists refilled immediately when size drops below lower bound
 - Recursive sifting of elements from child node
 - Clean deleteMin operation



Soft heap versions

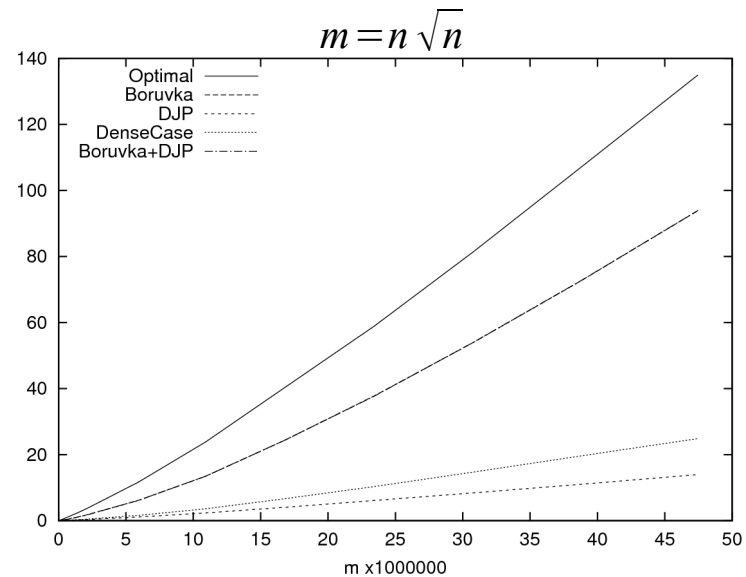
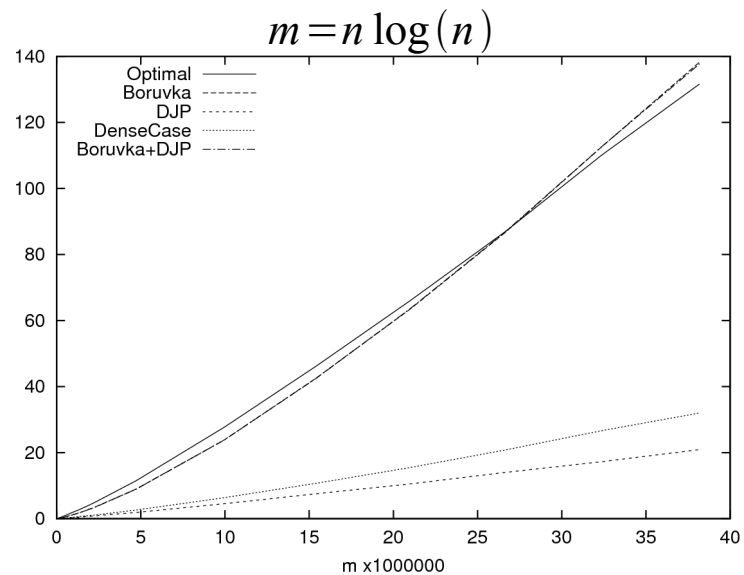
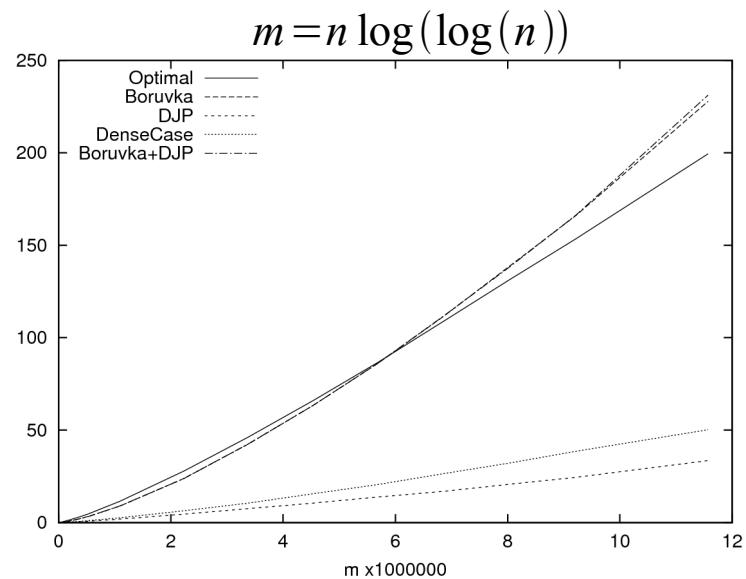
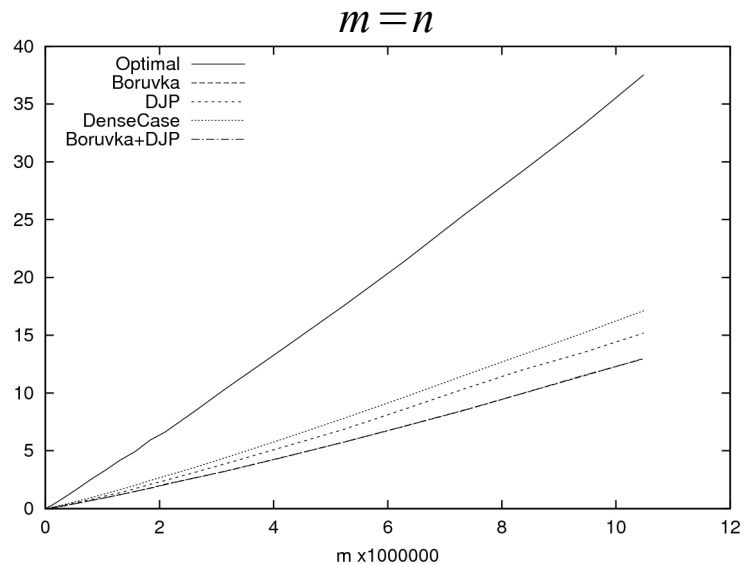
- Optimal MST algorithm profile (Chazelle):
 - Insert: 10% - 15% (All edges)
 - DeleteMin: $< 1\%$ (Very few edges)
- Pros and cons for the optimal MST algorithm:

	Pros :-)	Cons :-)
Chazelle	- Insert without sift - Delayed clean-up	- Complicated analysis - Larger Big-Oh constant?
Kaplan Zwick	- Intuitive implementation - Smaller Big-Oh constant?	- Insert with sift - Immediate Clean-up

Experimental results

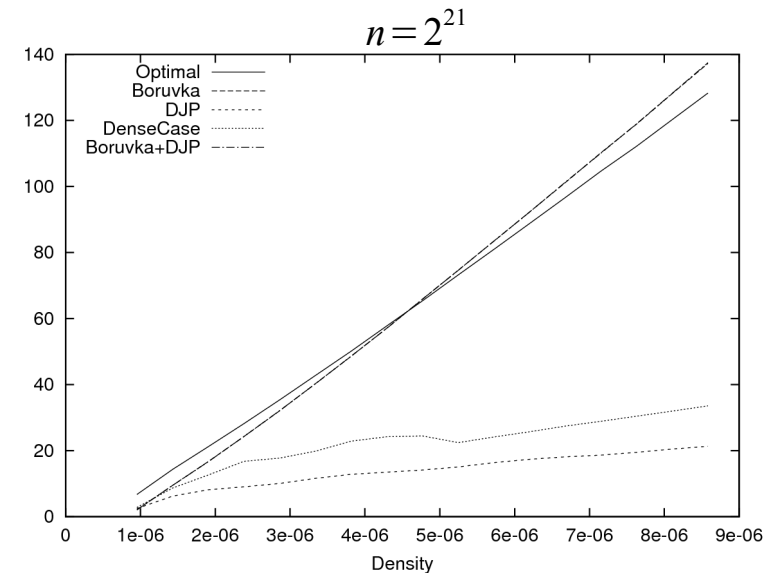
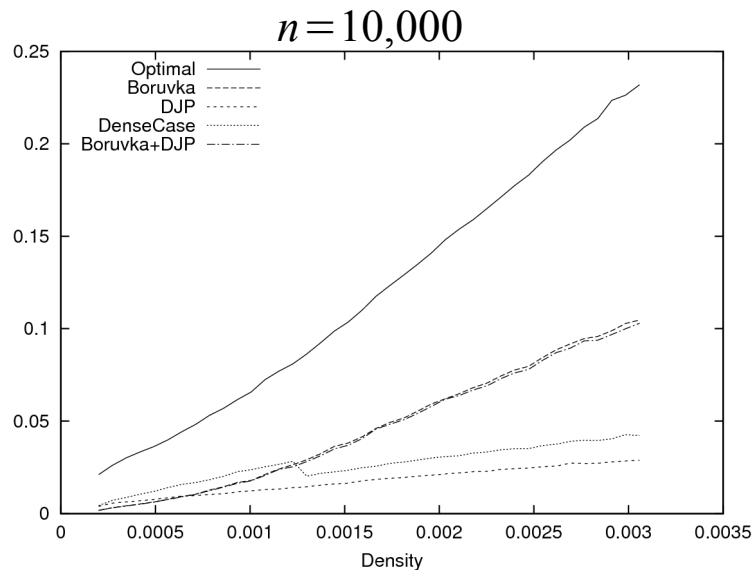
- Advanced algorithm
 - Many "sub algorithms"
 - Large Big-Oh constant
- Can not beat linear time algorithms

Constant density



Variable density

Density function: m/m_{max}



Conclusion for realistic input

- Experiment winner: DJP
 - 2nd: Dense Case
- Experiment loser: Optimal
 - 2nd: Borůvka
- Optimal vs. Borůvka
 - Optimal is fastest for worst case Borůvka graphs (narrow interval of densities)
 - Otherwise, Borůvka is fastest

The end

Any questions?

