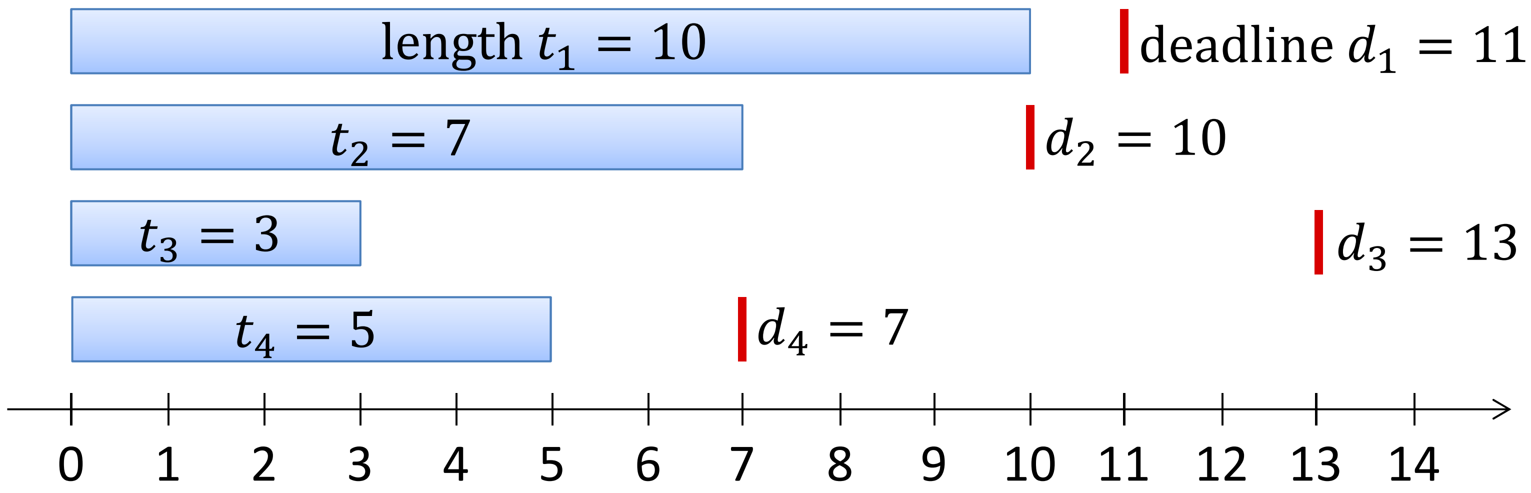
Chapter 2
Greedy Algorithms

Algorithm Theory
WS 2014/15

Fabian Kuhn

Scheduling Jobs with Deadlines

- Given: n requests / jobs with deadlines:



- Goal: schedule all jobs with minimum lateness L
  - Schedule: $s(i)$, $f(i)$: start and finishing times of request i
    Note: $f(i) = s(i) + t_i$

- Lateness $L := \max\left\{0, \max_i\{f(i) - d_i\}\right\}$
  - largest amount of time by which some job finishes late

- Many other natural objective functions possible…

Greedy Algorithm

Schedule by earliest deadline?

- Schedule in increasing order of d_i
- Ignores lengths of jobs: too simplistic?

- Earliest deadline is optimal!

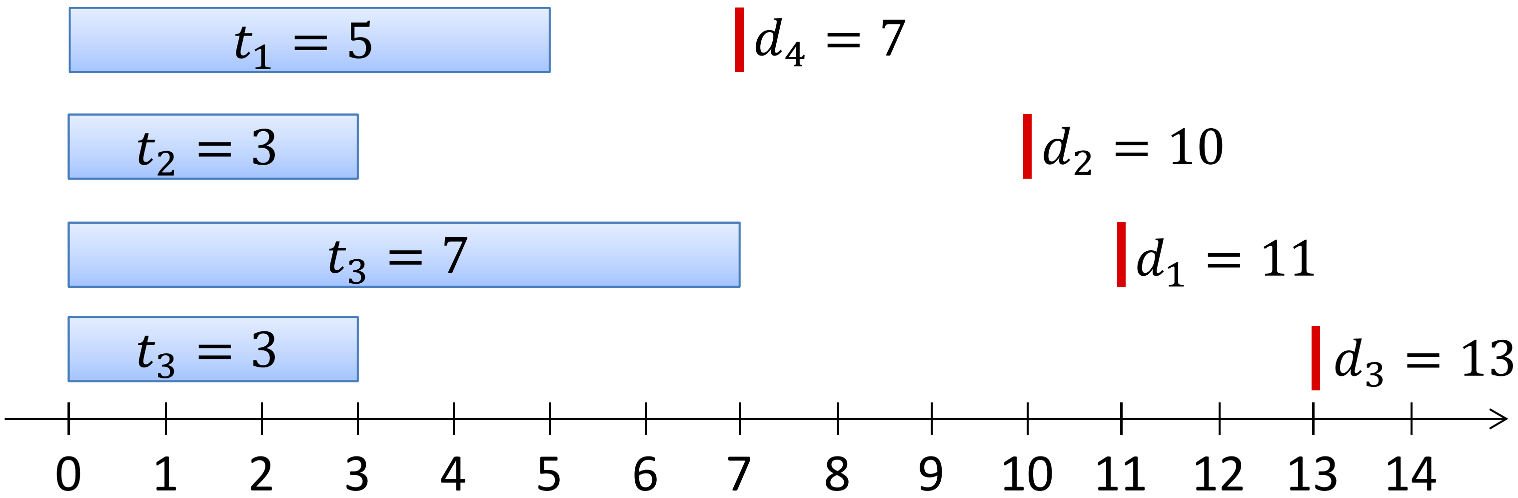
Algorithm:

- Assume jobs are reordered such that $d_1 \leq d_2 \leq \cdots \leq d_n$
- Start/finishing times:
  - First job starts at time $s(1) = 0$
  - Duration of job i is t_i: $f(i) = s(i) + t_i$
  - No gaps between jobs: $s(i + 1) = f(i)$

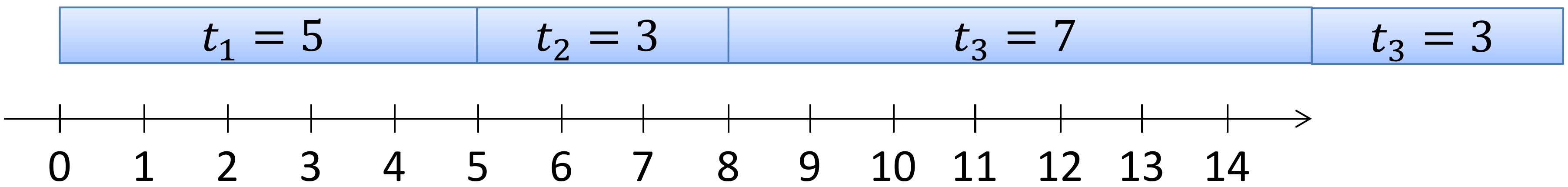
  (idle time: gaps in a schedule → alg. gives schedule with no idle time)

Example

Jobs ordered by deadline:



Schedule:



Lateness: job 1: 0, job 2: 0, job 3: 4, job 4: 5

Basic Facts

1. There is an optimal schedule with no idle time
   - Can just schedule jobs earlier…

2. Inversion: Job i scheduled before job j if $d_i > d_j$
   Schedules with no inversions have the same maximum lateness
   - In schedules with no inversions, jobs are sorted by deadline
   - Only jobs with the same deadline can be permuted
   - For each deadline d, the maximum lateness remains the same if these jobs are reordered

Earliest Deadline is Optimal

Theorem:
There is an optimal schedule $\mathcal{O}$ with no inversions and no idle time.

Proof:

- Consider optimal schedule $\mathcal{O}'$ with no idle time
- If $\mathcal{O}'$ has inversions, ∃ pair (i, j), s.t. i is scheduled immediately before j and $d_j < d_i$
- Claim: Swapping i and j gives schedule with
  1. Less inversions
  2. Maximum lateness no larger than in $\mathcal{O}'$

Earliest Deadline is Optimal

Claim: Swapping i and j: maximum lateness no larger than in $\mathcal{O}'$

Exchange Argument

- General approach that often works to analyze greedy algorithms

- Start with any solution
- Define basic exchange step that allows to transform solution into a new solution that is not worse
- Show that exchange step move solution closer to the solution produced by the greedy algorithm
- Number of exchange steps to reach greedy solution should be finite…

Another Exchange Argument Example

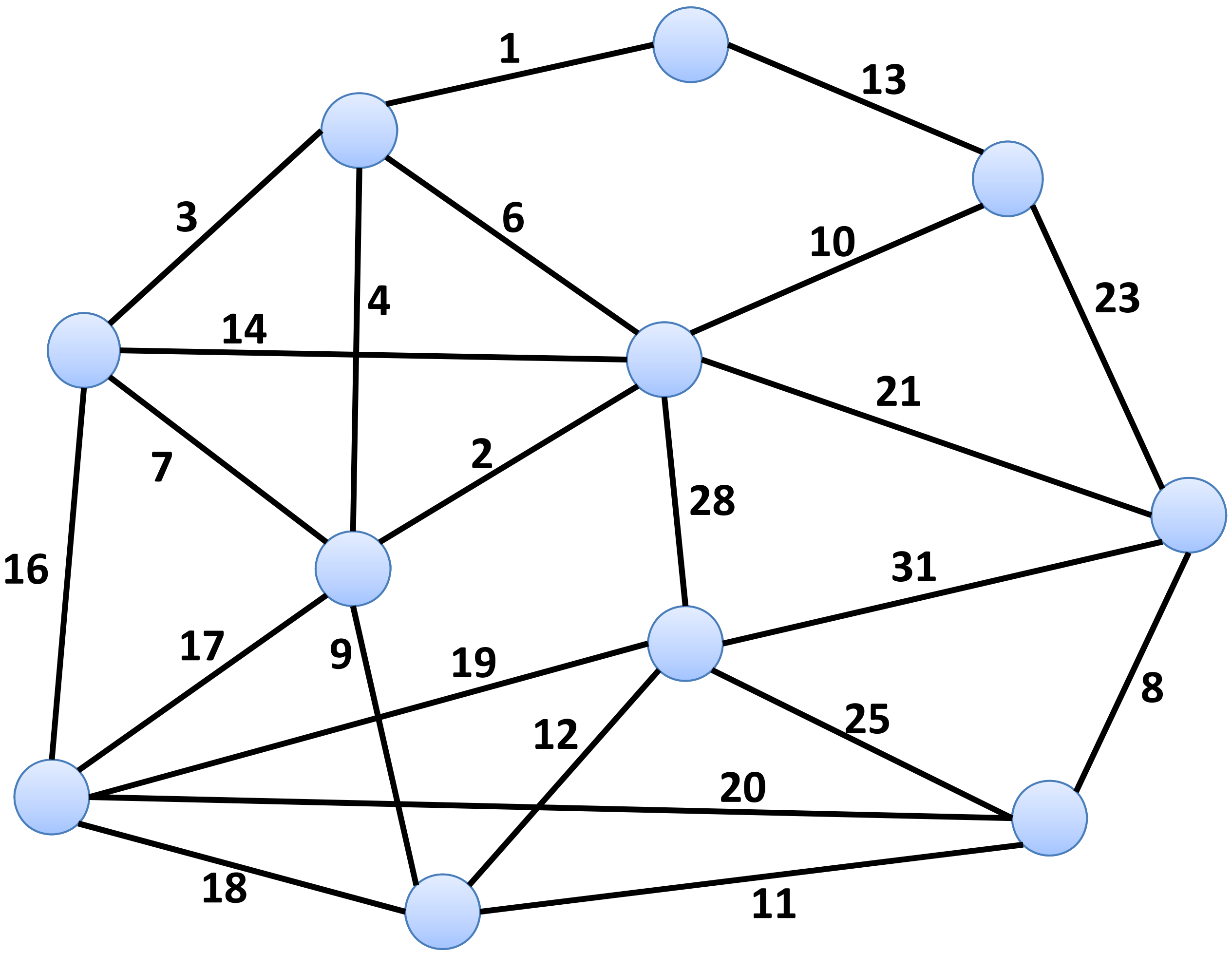
- **Minimum spanning tree (MST)** problem
  - Classic graph-theoretic optimization problem

- **Given**: weighted graph
- **Goal**: spanning tree with min. total weight

- Several greedy algorithms work

- Kruskal's algorithm:
  - Start with empty edge set
  - As long as we do not have a spanning tree:
    add minimum weight edge that doesn't close a cycle

Kruskal Algorithm: Example



Kruskal is Optimal

- Basic exchange step: swap to edges to get from tree T to tree T'
  – Swap out edge not in Kruskal tree, swap in edge in Kruskal tree
  – Swapping does not increase total weight

Matroids

- Same, but more abstract…

Matroid: pair $(\boldsymbol{E}, \boldsymbol{I})$

- $\boldsymbol{E}$: set, called the **ground set**
- $\boldsymbol{I}$: finite family of finite subsets of E (i.e., $I \subseteq 2^E$), called **independent sets**

(E, I) needs to satisfy 3 properties:

1. Empty set is independent, i.e., $\emptyset \in \boldsymbol{I}$ (implies that $I \neq \emptyset$)
2. **Hereditary property**: For all $A \subseteq E$ and all $A' \subseteq A$,

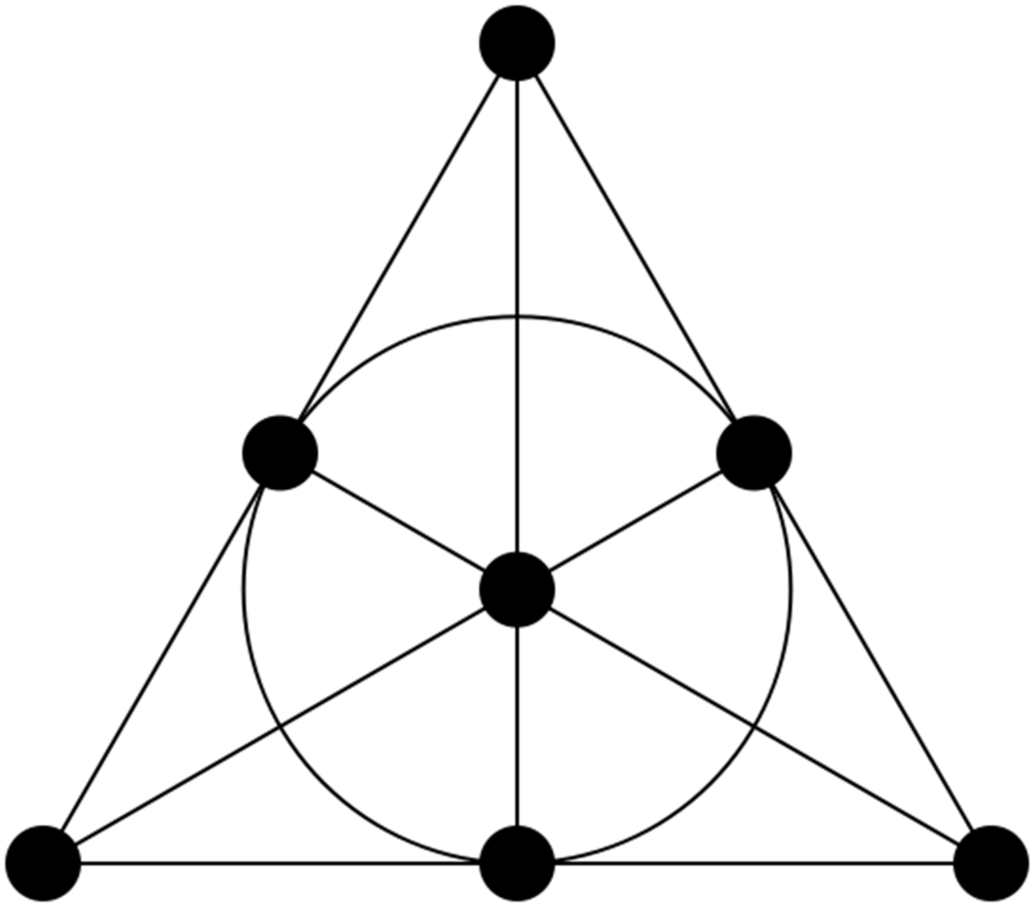
   if $\boldsymbol{A} \in \boldsymbol{I}$, then also $\boldsymbol{A}' \in \boldsymbol{I}$

3. **Augmentation / Independent set exchange property:**
   If $A, B \in I$ and $|A| > |B|$, there exists $x \in A \setminus B$ such that

$$\mathbf{B}' := \boldsymbol{B} \cup \{\boldsymbol{x}\} \in \boldsymbol{I}$$

Example

- Fano matroid:
  - Smallest finite projective plane of order 2…



Matroids and Greedy Algorithms

Weighted matroid: each $e \in E$ has a weight $w(e) > 0$

Goal: find maximum weight independent set

Greedy algorithm:

1. Start with $S = \emptyset$
2. Add max. weight $e \in E \setminus S$ to S such that $S \cup \{e\} \in I$

Claim: greedy algorithm computes optimal solution

Greedy is Optimal

- S: greedy solution A: any other solution