Chapter 2
Greedy Algorithms

Algorithm Theory
WS 2016/17

Fabian Kuhn

Greedy Algorithms

- No clear definition, but essentially:

In each step make the choice that looks best at the moment!

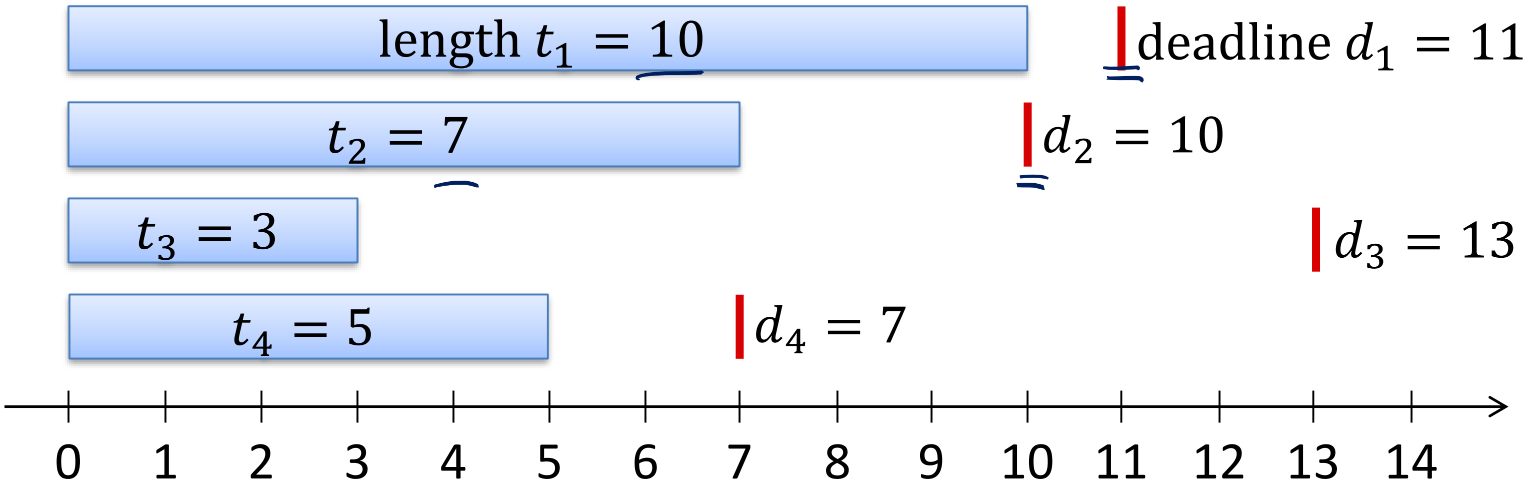
no backtracking

- Depending on problem, greedy algorithms can give
  - Optimal solutions
  - Close to optimal solutions
  - No (reasonable) solutions at all
- If it works, very interesting approach!
  - And we might even learn something about the structure of the problem

Goal: Improve understanding where it works (mostly by examples)

Scheduling 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)

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

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, $\exists$ pair (i, j), s.t. i is scheduled immediately before j and $d_j < d_i$

- **Swapping i and j** gives schedule with
  1. Less inversions
  2. 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

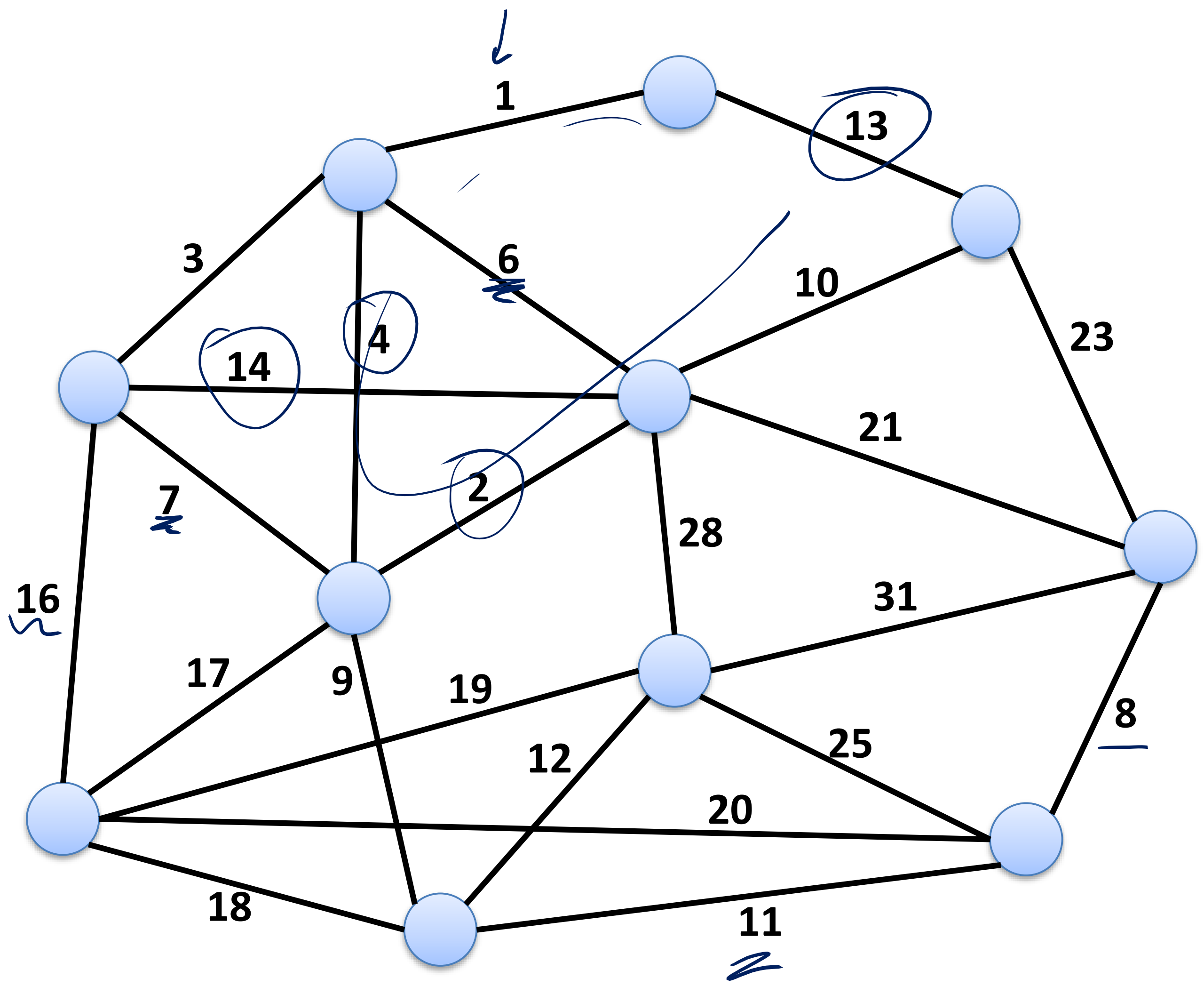
- **Minimum spanning tree (MST)** problem
  - Classic graph-theoretic optimization problem

- **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
- For simplicity, assume, weights are unique:

T: any spanning tree T_k: Kruskal tree

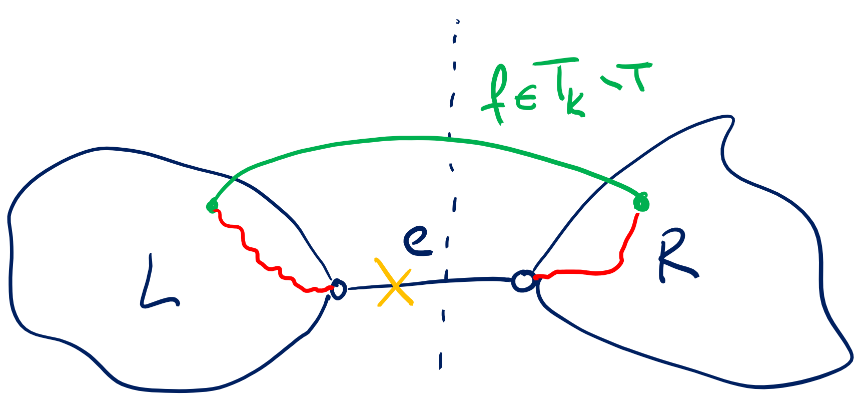
$T \neq T_K$ $e \in T \setminus T_K$

$w(f) < w(e)$:

assume otherwise:

Kruskal considers e before f

Kruskal would have added e ↯



$T' := T \setminus \{e\} \cup \{f\}$ $w(T') < w(T)$

Matroids

$E = \{1,2,3,4\}$

$I = \{\emptyset, \{1\}, \dots, \{4\}, \{1,2\}, \{1,3\}, \dots, \{3,4\}\}$

$\emptyset \in I$ ✓ $\quad \{1,3\} \quad \{2\} \quad \{3\}$

- Same, but more abstract…

set system

Matroid: **pair (E, I)**

- $\boldsymbol{E}$: set, called the **ground set** set of elements
- $\boldsymbol{I}$: 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 $A' \subseteq A$,

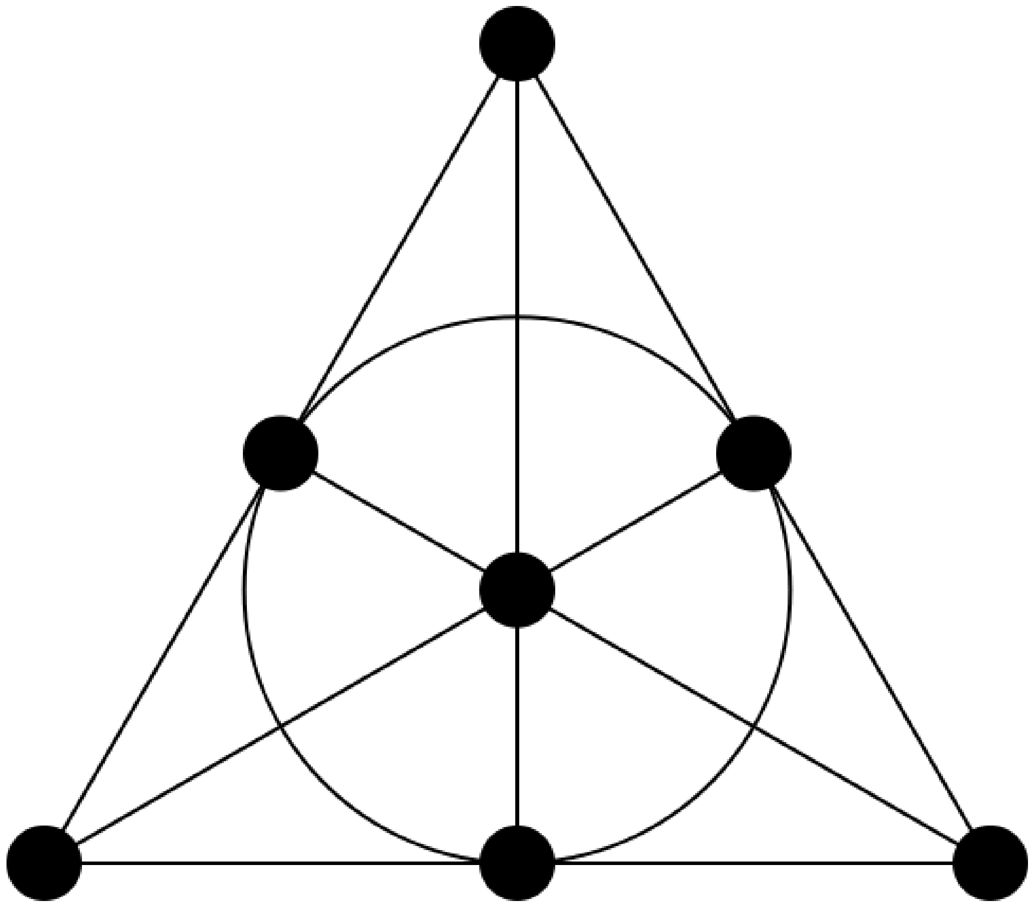
   if $\boldsymbol{A} \in \boldsymbol{I}$, then also $\boldsymbol{A}' \in \boldsymbol{I}$

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

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

Example

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



Matroids and Greedy Algorithms (E, I)

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

$(E, \mathcal{I})$ $S \in \mathcal{I}$ $A \in \mathcal{I}$ $s = |S|$ $a = |A|$

- S: greedy solution A: any other solution

$S \subseteq E,\ S \in \mathcal{I}$ $A \subseteq E,\ A \in \mathcal{I}$

$|S| \geq |A|$:

for contradiction, assume $|A| > |S|$ augm. prop: $\exists x \in A \setminus S$: $S \cup \{x\} \in \mathcal{I}$

greedy would have added x

$\boxed{w(S) \geq w(A)}$:

for contradiction, assume $\boxed{w(S) < w(A)}$

$S = \{x_1, x_2, \dots, x_s\}$ $w(x_1) \geq w(x_2) \geq \dots \geq w(x_s)$

$A = \{y_1, y_2, \dots, y_a\}$ $w(y_1) \geq w(y_2) \geq \dots \geq w(y_a)$

will show:

$\forall i \in \{1, \dots, a\}: w(x_i) \geq w(y_i)$ (*)

$\hookrightarrow w(S) \geq w(A)$

$\neg(*) \Rightarrow$ there is a smallest k s.t. $w(x_k) < w(y_k)$

$S' = \{x_1, \dots, x_{k-1}\}$ augm. property: $\exists y \in A' \setminus S'$: $S' \cup \{y\} \in \mathcal{I}$

$A' = \{y_1, \dots, y_k\}$

$w(y) \geq w(y_k) > w(x_k)$

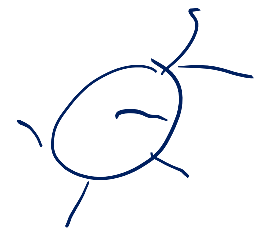
greedy considers y before x_k

greedy would add y $\lightning$

Matroids: Examples

Forests of a graph $G = (V, E)$:

- forest F: subgraph with no cycles (i.e., $F \subseteq E$)
- $\mathcal{F}$: set of all forests → $(E, \mathcal{F})$ is a matroid
- Greedy algorithm gives maximum weight forest (equivalent to MST problem)



Bicircular matroid of a graph $G = (V, E)$:

- $\mathcal{B}$: set of edges such that every connected subset has ≤ 1 cycle
- $(E, \mathcal{B})$ is a matroid → greedy gives max. weight such subgraph

Linearly independent vectors: $\mathbb{R}^d$

- Vector space V, E: finite set of vectors, I: sets of lin. indep. vect.
- Fano matroid can be defined like that

Forest Matroid $G=(V,E)$

<u>matroid</u> $(E, \mathcal{F})$ — set of forests

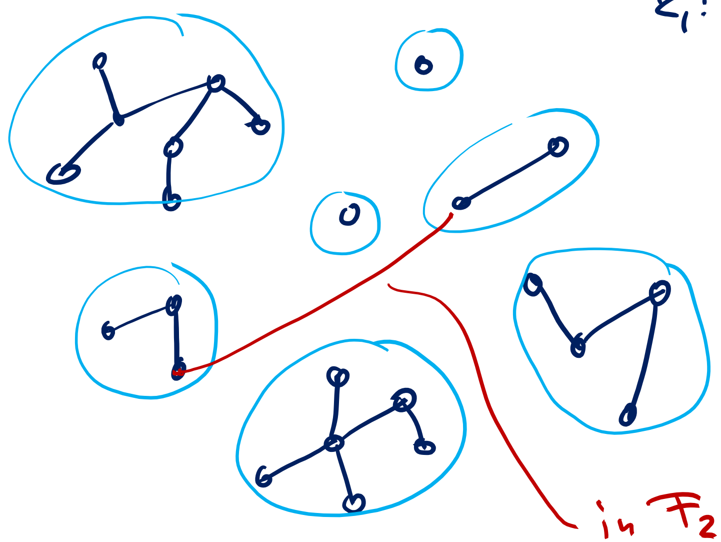
E: set of edges of G

1) $\emptyset \in \mathcal{F}$ ✓

2) $F \in \mathcal{F} \wedge F' \subseteq F \Longrightarrow F' \in \mathcal{F}$ ✓

3) <u>augm. property:</u> forests F_1, F_2 $\quad |F_1| < |F_2|$

<u>F_1:</u>



k_1: # of components $\quad (n = |V|)$

$|F_1| = n - k_1$

comp.

$|F_2| > |F_1| \Longrightarrow k_2 < k_1$

$\square$

Greedoid

- Matroids can be generalized even more

- Relax hereditary property:

  Replace $\quad A' \subseteq A \subseteq I \quad \Longrightarrow \quad A' \in I$

  by $\quad \emptyset \neq A \subseteq I \quad \Longrightarrow \quad \exists a \in A,\ \text{s.t. } A \setminus \{a\} \in I$

(Augm)

- Exchange property holds as before

- Under certain conditions on the weights, greedy is optimal for computing the max. weight $A \in I$ of a greedoid.
  - Additional conditions automatically satisfied by hereditary property

- More general than matroids