Lecture 2 (2011-10-13)**:**

Definition 2.1 (connected)**:**
A graph is called *connected (zusammenhängend)* if there exists a [s,t]-Path be-
tween all pairs of vertices $s, t \in V$.

Definition 2.2 (forrest, tree, spanning, forest problem, minimum spanning tree)**:**
A *forest (Wald)* is a graph that does not contain a cycle (Kreis). A connected
forest is called a *tree (Baum)*. A tree in a graph (as subgraph) is called *spanning
(aufspannend)*, if it contains all vertices.
Given a graph $G = (V, E)$ with edge weights $c_e \in \mathbb{R}$ for all $e \in E$, the task to find
a forest $W \subset E$ such that $c(W) := \sum_{e \in W}$ is maximal, is called the *Maximum Forest
Problem (Problem des maximalen Waldes)*. The task to find a tree $T \subset E$ which
spans G and which weight $c(T)$ is minimal, is called the *Minimum Spanning Tree
(MST) problem (minimaler Spannbaum)*.

Lemma 2.3:
A tree $G = (V, E)$ with at leat 2 vertices has at least 2 vertices of degree 1.

Proof. Let v be arbitrary. Since G is connected, $deg(v) \geq 1$. Assume $deg(v) =$
1. So $\delta(v) = \{vw\}$. If $deg(w) = 1$, we found two vertices with $degree$ 1. If
$deg(w) > 1$, there exist a neighbour of w, different from $v : u$. Now, again u has
$degree$ 1 or higher. If we repeat this procedure we either find a vertix of degree
1 or find again *new* vertices. Hence, after at most $n - 1$ vertices we end up at
a vertex of degree 1. Now, if $deg(v) \geq 2$, we do the same and find a vertex of
degree 1, say w. Then repeat the above, staring from w to find a second vertex
of degree 1.   □

Corollary 2.4:
A tree $G = (V, E)$ with maximum degree Δ has at least Δ vertices of degree 1.

Lemma 2.5: (a) For every graph $G = (V, E)$ it holds that $2|E| = \sum_{u \in V} deg(u)$

   (b) for every tree $G = (v, E)$ it holds that $|E| = |V| - 1$.

Proof.   (a) trivial

   (b) Proof by induction. Clearly, if $|V| = 1$ or $|V| = 2$ it holds. Assumption: true
       for $n \geq 2$. Let G be a tree with $n + 1$ vertices. By Lemma 2.3, there exists
       a vertex $v \in G$ with $deg(v) = 1. G - v = G[V \setminus \{v\}$ is a tree again with n
       vertices and thus $|E(G - v)| = V(G - v)| - 1$. Since G differs by one vertex
       and one edge from $G - v$, the claim holds got G as well.

                                                                                          □

Lemma 2.6:
If $G = (V, E)$ whith $|V| \geq 2$ has $|E| < |V| - 1$, G is not connected.

Algorithm MST

$min_{x \in X} = -max_{x \in X} - c(x)$ ~~maximal forest~~
X spanning trees
$min_{x \in X} + (n-1)D = -max_{x \in X} - c(x)(n-1)D = max_{x \in X} \sum \underbrace{D - C_{ij}x_{ij}}_{\geq 0 \, if \, D \geq max_{ij \in E} c_{ij}}$

Theorem 2.7:
Kruskal's Algorithm returns the optimal solution.

Proof. Let T be Kruskal's tree and assume there exists a tree T' with $c(T') < c(T)$. Then there exist an edge $e' \in T' \setminus T$. Then $T \cup \{e'\}$ contains a cycle $\{e_1, e_2, \ldots, e_k, e'\}$. Let $c_f = max_{i=1,\ldots k} c_{l_i}$. At the moment Kruskal chooses edge f, edge e' cannot be added yet and therefore $c(e') \geq c(f)$. Now exchange e' by f in T'. Hence the number of differences beetween T' and T is reduced by one, $C(T'_{new}) \leq c(T') < c(T)$. Repeating the procedure results in $c(T) \leq \ldots < c(T)$, a contradiction. $\square$

Lecture 3 (2011-10-17)**:**

Definition 2.7(+1):
The *running time of algorithms (Laufzeit)* of an algorithm is measured by the number of operations needed in worst case of a function of the input size. We use the $O(\cdot)$ notation (Big-O-notation) ot focus on the most important factor of the running time, ignoring constants and smaller factors.

Example 2.7(+2):
If the running time is $3n \cdot \log n + 26n$, the algorithm runs in $O(n \cdot \log n)$. If the running time is $3n \cdot \log n + 25n^2$, the algorirthm runs in $O(n^2)$.

For graph Problems, the running is expressed in the number of vertices $n = |V|$ and the number of edges $m = |E|$. Sometimes m is approximated by n^2.

Example 2.7(+3) (Kruskal's Algorithm)**:**
First, the edged are sorted according to nondecreasing weights. This can be done in $O(m \cdot \log m)$. Next, we repeatedly select an edge or reject its selection until $n-1$ edges are selected. Since the last selected edge might be after m steps, this routine is performed at most $O(m)$ times.

Checking whether the end nodes of $\{u, v\}$ are already in the same tree can be done in constant time, if we label the vertices of the trees selected so far: $r(u) = \#trees\, containing\, u$. If $r(u) \neq r(v)$, the trees are connected by $\{u, v\}$ to a new tree.

Without going into details, the resetting of labels in one of the old trees, can be done $O(\log n)$ on average. Since this update has to be done at most $n-1$ times, it takes $O(n \cdot \log n)$.

Overall, Kruskal runs in

$$O(n \log m + m + n \cdot \log n) = O(m \cdot \log m) = O(m \cdot \log n^2) = O(m \cdot \log n)$$

Definition 2.7(+4) (Shortest paths in acyclic digraphs)**:**
A directed graph (digraph) $D = (V, A)$ is called *acyclic (azyklisch)* if it does not
contain any *directed cycles ()*, i.e. a *chain (Kette)* $(v_0, a_1, v_1, a_2, v_2, \ldots a_k, v_k)$,
$k \geq 0$, with $a_i(v_{i-1}, v_i) \in A$ and $v_k = v_0$. In particular, D does not contain
antiparallel () arcs: if $(u, v) \in A$, $(v, u) \notin A$. With $\delta_D^+(v)$ we denote the arcs
leaving vertex v:

$$\delta_D^+(v) = \{(u, w) \in A : u = v\}$$

similarly:

$$\delta_D^-(v) = \{(u, w) \in A : w = v\}$$

are the arcs entering v.

The *outdegree ()* of v is $\deg_D^+(v) = |\delta^+(v)|$ (assuming simple digraph)
The *indegree ()* of v is $\deg_D^-(v) = |\delta^-(v)|$

Definition 3.1:
The *shortest path ()* problem in a acyclic digraph is, given an acyclic digraph
$D = (V, A)$, a length function $C : A \to \mathbb{R}$ and two vertices $s, t \in V$, find a
$[s, t]$-path of minimal length.

Question 3.1.1:
Does there exist a $[s, t]$-path at all?

Theorem 3.2:
A digraph $D = (V, A)$ is acyclic, if and only if there exists a permutaion $\sigma : V \to$
$\{1, .., n\}$ of the vertices such that $\deg_{D[v_1, \ldots, v_n]}^-(v_i) = 0$ for all $i = 1, \ldots, n$ with
$v_i = \sigma^{-1}(i)$.

Proof. By induction:

For digraph with $|V| = 1$, the statement is true. Assume the statement is true for
all digraphs with $|V| \leq n$ and consider $D = (V, A)$ acyclic with $n + 1$ vertices. If
there does not exist a vertex with $\deg_D^-(v) = 0$, a directed cycle can be detected
by following incoming arcs backwards until a vertex is repeated, a contradiction
regarding the acyclic property of D.

Hence, let v be a vertex with $\deg_D^-(v) = 0$. Set $v_1 = v$. The digraph $D - v_1$ has
n vertices and is acyclic, and thus has a permutation $(v_2, \ldots, v_{n+1})$ with

$$\deg_{D[v_i, \ldots, v_{n+1}]}^-(v_i) = 0 \qquad \forall i = 2, \ldots, n + 1$$

Now, $(v_1, \ldots, v_{n+1})$ is a permutation fulfilling the condition.

In reverse, if there exists a permutation $(v1, \ldots, v_{n+1})$, $\deg_D^-(v_1) = 0$ and there
cannot exist a directed cycle containing v_1. By induction, neither cycles containing
$v_i, i = 2, \ldots, n + 1$ exist.   □

Theorem 3.3:
A $[s, t]$-path exists in a acyclic Digraph $D = (V, A)$ if and only if in all permutations
$\sigma : V \to \{1, \ldots, n\}$ with $\deg_{D[v_i, \ldots, v_n]}^-(v_i) = 0$ for all $i = 1, \ldots, n$, it holds that
$\sigma(s) < \sigma(t)$.

Proof. Assume there exists a permutation σ with $\sigma(s) > \sigma(t)$. Since outgoing arcs only go to higher ordered vertices, there does not exist a path from s to t in D.

In reverse, if there does not exist a path from s to t, we order all vertices with paths to t first, followed by t and s afterwards.   □

Question 3.3.1:
How do we find the shortest $[s, t]$-path if it exists?

To simplify notation, let $V = \{1, \ldots, n\}, s = 1, t = n$ and $(i, j) \in A \Rightarrow i < j$. Let $D(i)$ be the distance from i to n and $NEXT(i)$ be the next vertex on the shortest path from i to n.

Bellman's Algorithm

```
1   D(i) = {∞ : i < n and NEXT(i) = NIL, 0 : i = n}
2   FOR  i = n − 1 DOWNTO 1 DO
3            D(i) = min_{j=j+1,...,n}{D(j) + c(i,j)} with c(i,j) = ∞ if
                 (i,j) ∉ A
4            NEXT(i) = Argmin_{j=i+1,...,n}{D(j) + c(i,j)}
```

Theorem 3.4:
Bellman's Algorithm is correct and runs in $O(m + n)$ time.

Proof. Every path from 1 to n passes through vertices of increasing ID. Assume there exists a path $(a_1, \ldots, a_k)$ with $\sum_{i=1}^{k} c(a_i) < D(1)$. Let $a_1 = (1, j_1)$. Since $D(1) \leq c(a_1) + D(j_1)$, it should hold that

$$\sum_{i=2}^{2} c(a_i) < D(j_1)$$

But $D(j_1) \leq c(a_2) + D(j_2)$ with $a_2 = (j_1, j_2)$, etc.

In the end, $c(a_k) < D(j_{k-1})$ but $D(j_{k-1}) \leq c(a_k) + D(n) = c(a_k)$, contradiction.
                                                                                               □

Lecture 4 (2011-10-20)**:**

Theorem 3.5:
Bellman's Algorithm is correct and runs in $O(m + n) = O(n)$.

Proof. of runtime:

$$D(i) = \min_{(i,j) \in A} D(j) + D(i, j)$$

$\Rightarrow$ Every arc is considered once, and thus overall $O(m)$ computations are needed. Initialization costs $O(n)$.   □

Bemerkung 3.5(+1):
The running time does not contain the time to find the permutation.

Observation 1: We not only found the shortest path from 1 to n, but also from i to n, $i = 2, ..., n$.

Observation 2: We can use a similar procedure for the shortest path from 1 to i, $i = 2, ..., n$. (with $PREV(i)$ for previous instead of $NEXT(i)$).

Question 3.5.1:
Can we find a shortest path from 1 to i in a digraph that is not acyclic, i.e. it contains cycles?

Theorem 4.1:
The Moore-Bellman-Algorithm returns the shortest paths from 1 to $i = 1, ..., n$ provided D does not contain negative-weighted directed cycles.

Proof. We call an arc $(i, j) \in A$ an *upgoing* arc (Aufwärtsbogen) if $i < j$ and a *downgoing* arc (Abwärtsbogen) if $i > j$.

A shortest path from 1 to i contains at most $n - 1$ arcs. If an upgoing arc is followed by a downgoing arc (or vice versa), we have a *change of direction* (Richtungswechsel). With at most $n - 1$ arcs, at most $n - 2$ changes of direction are possible.

Let $D(i, m)$ be the value of $D(i)$ at the end of the m-th iteration. We will show (and this is enough):

$$D(i, m) = min\{c(W) : W \text{ is the directed } [1, i]\text{-path with at most } m \text{ changes of directions}\}$$

We prove it by induction on m.

- For $m = 0$, the algorithm is equivalent to Bellman's algorithm for acyclic grpahs. Thus, $D(i, 0)$ is the length of the shortest path without any changed of direction.

- Now, let us assume, that the statement is true for $m \geq 0$ and the subroutine is executed for the $m + 1$-st time. The set of $[1, i]$-paths with at most $m + 1$ changes of direction consists of

  (a) $[1, i]$-paths with $\leq m$ changes of direction

  (b) $[1, i]$-paths with exactly $m + 1$ changes of direction

  $\Rightarrow D(i, m)$

- Since every path starts with an upgoing arc $(1, k)$, the last arc after $m + 1$ changes is either a downgoing arc if $m + 1$ is odd or an upgoing arc if $m + 1$ is even. We restrict ourselves to $m + 1$ odd ($m + 1$ even is similar).

  To compute the minimum length path in (b) we use an additional induction on $i = n, n - 1, ..., j + 1$. Since every path ending at n ends with an upgoing arc, there do not exist such $[1, n]$-paths. Hence, $D(n, m + 1) = D(n, m)$.

Now assume that $D(k, m+1)$ is correctly computed for $i \leq k \leq n$. The shortest path from 1 to $i-1$ with exactly $m+1$ changes ends with a downgoing arc $(j, i-1), j > i-1$.

$D(j, m+1)$ is already computed correctly. If $PREV(j) > j$, no change of direction is required in j and $D(i-1, m+1) = D(j, m+1) + c(j, i-1)$ If $PREV(j) < j$, the last avec of the $[1, j]$-Path is upgoing, and thus $D(i-1, m+1) = D(j, m) + c(j, i)$. The last change of direction at j is thus, in worst case, the $(m+1)$-st change. Hence, $D(i-1, m+1)$ fulfills the statement.

$\square$

Remark 1:

In fact, the algorithm finds the minimum length of a chain (kette) with at most $n-2$ changes of direction. In case of negative weighted cycles these might be in a chain several times.

In case no negative weighted cycles exist, the min. length chains are indeed paths. Hence, the algoithm only works correctly if *all* cycles are non-negative weighted.

Remark 2:

If a further executing of the subroutine ($m = n-1$) results in at least one change of a value $D(i)$, then the digraph contains negative weighted cycles.

Remark 3:

A more efficient implementation is given by E'sopo-Pape-Variant.

Dijkstra's Algorithm for non-negative weights

Theorem 4.2:

Dijkstra returns the shortest paths from 1 to $i, i = 1 \ldots n$, provided all weights ≥ 0.

Proof. Each step, one vetex is moved from T to S. At the end of a step, $D(j)$ is the shortest path from 1 to j via vertices in S.

If $S = V(T = \emptyset), D(i)$ is thus the shortiest $[1, i]$-path $\hspace{2cm}\square$

Lecture 5 (2011-10-24)**:**

Shortest paths between all pairs of vertices

Solution 1: Apply Moore-Bellman or Dijkstra to all vertices i as starting vertex

Solution 2: Apply Floyd's Algorithm

Notation:

w_{ij} is the length of the shortest $[i, j]$-path, $i \neq j$

w_{ii} is the length of the shortest directed cycle containing i

p_{ij} is the predecessor of j on the shortest $[i, j]$-path (cycle)

$W = (w_{ij})$ is the *shortest path length matrix (???)*

Theorem 5.1:
The Floyd Algorithm works correctly if and only if $D = (V, A)$ does not contain any negative weighted cycles.

D contains a negative weighted cycle if and only if one of the diagonal elements $w_{ii} < 0$.

Proof. Let W^k be the matrix W after iteration k, with W^0 being the initial matrix. By induction on $k = 0, \ldots, n$ we show that W^k is the matrix of shortest path lengths with vertices $1, \ldots, k$ as *possible* internal vertices, provided D does not contain a negative cycle on these vertices.

If D has a negative cycle, then $w_{ii}^k < 0$ for an $i \in \{1, \ldots, n\}$

For $k = 0$, the statement clearly true.

Assume, it is correct for $k \geq 0$, and we have executed the $(k + 1)$st iteration.

It holds that $w_{ij}^{k+1} = \min\{w_{ij}^k, w_{i,k+1}^k + w_{k+1,j}^k\}$. Note that, provided no negative cycle exists, $w_{i,k+1}^{k+1}$ does not have any vertex $k + 1$ as internal vertex, and thus $w_{i,k+1}^{k+1} = w_{i,k+1}^k$ (similarly, $w_{k+1,j}^{k+1} = w_{k+1,j}^k$).

$w_{i,k+1}^k$ is the minimal length of a $[i, k+1]$-path with $\{1, \ldots, k\}$ as allowed internal vertices. Similarly, $w_{k+1,j}^k$.

Thus, $w_{i,k+1}^k + w_{k+1,j}^k$ is the minimal length of an $[i, j]$-path (not necessarily simple) containing $k + 1$ (mandatory) and $\{1, \ldots, k\}$ (voluntary). If the shortest path from i to j using $\{1, \ldots, k+1\}$ does not contain $k+1$, it only contains $\{1, \ldots, k\}$ (voluntary) and, hence, w_{ij}^k is the right value.

What remains to show is that the connection of the $[i, k + 1]$-path with the $[l + 1, j]$-path is indeed a simple path.

Let K be this chain. After removal of cycles, the chain K contains (of course) a simple $[i, j]$-path $\bar{K}$. Since such cycles may only contain vertices from $\{1, \ldots, k + 1\}$, one cycle must contain $k + 1$. If this cycle is not negatively weighted, then path $\bar{K}$ is shorter and $w_{ij} < w_{i,k+1}^k + w_{k+1,j}^k$.

If this cycle is negatively weighted, $w_{k+1,k+1}^k < 0$ (the cycle only contains internal vertices from $\{1, \ldots, k\}$) and algorithm would have stopped earlier. $\qquad\square$

Min-Max-Theorems for combinatorial Optimization Problems

From "Optimierung A": Duality of linear programs

$$\max_{\text{s. t.}, Ax \leq b, x \geq 0} c^T x = \min_{\text{s.t.}, A^T y \geq c, y \geq 0} b^T y$$

For several combinatorial problems $\min\{c(x) : x \in X\}$

We can define a second set Y and a function $b(y)$ with $\max\{b(y) : y \in Y\} = \min\{c(x) : x \in X\}$ where Y and b(y) have a graph theoretical interpretation.

Existence of such a "Dual" Problem indicates often that the problem can be solved "efficiently". For the shortest path problem several max-min-theorems exist.

Definition 5.2:
An (s, t)-*cut ((s, t)-Schnitt)* in a digraph $D = (V, A)$ with $s, t \in V$ is a subset

$B \subset A$ of the arcs with the property that every (s, t)-path contains at least one arc of B.

Stated otherwise, for every cut B, there exists a vertex set $W \subset V$ such that

- $s \in W, t \in V \setminus W$

- $\delta^+(w) = \{(i, j) \in A : i \in W, j \in V \setminus W\} \subseteq B$

Theorem 5.3:
Let $D = (V, A)$ be a digraph, $c(a) = 1 \, \forall a \in A, s, t \in V, s \neq t$. Then the minimum length of a $[s, t]$-path equals the maximum number of arc-disjoint (s, t)-cuts.

Proof. Folows from **??**   □

Theorem 5.4:
Let $D = (V, A)$ be a digraph, $c(a) \in \mathbb{Z}_+ \, \forall a \in A \wedge s, t \in V \wedge s \neq t$. Then the min length of an $[s, t]$-path equals the maximum number d of (not necessarily different) (s, t)-cuts $C_1, \ldots, C_d$ such that every arc $a \in A$ is contained in at most $c(a)$ cuts.

Proof. We define (s, t)-cuts $C_i = \delta^+(v_i)$ with $V_i = \{v \in V : \exists (s, v)\text{-path with } c(P) \leq i - 1\}$

$$v_1 = \{s\}$$
$$v_2 = \{5, 3, 4\}$$
$$v_3 = \{5, 2, 3, 4\}$$
$$v_4 = v_3 \cup \{6\}$$

(for the example graph on the board)

The shortest $[s, t]$-path P consists of arcs $a_1, \ldots a_k$ with arc a_j contained in (s, t)-cuts C_i, $i \in \{\sum_{l=1}^{j-1} c(a_l) + 1, \ldots, \sum_{l=1}^{j} a(a_l)\}$: exactly $c(a)$ cuts.   □

Lecture 6 (2011-10-26)**:**

Knapsack problem

Definition 6.1:
The *Knapsack problem (Knapsack Problem)* is defined by a set of items $N = \{1, \ldots, n\}$ weights $a_i \in \mathbb{N}$, value $c_i \in \mathbb{N}$, and a bound $b \in \mathbb{N}$. We search for a subset $S \subset \mathbb{N}$ such that

$$a(S) = \sum_{i \in S} a_i \leq b \text{ and } c(S) = \sum_{i \in S} c_i \text{ maximum}$$

Approach 1:
Greedy algorithm

Idea: Items with small weight but high value are the most atrractive ones.

Procedure:

```
1  Sort the items such that  c₁/a₁ ≤ c₂/a₂ ≤ ... ≤ cₙ/aₙ .
2
3  Set S = ∅.
4  For i = 1 to n do
5          if (a(s) + aᵢ ≤ b) then
6                  S = S ∪ {i}
7          endif
8  endfor
9  return S and c(S)
```

Theorem 6.2:
The greedy algorithm does *not* guarantee an optimal solution.

Proof. Let $b = 10$, $n = 6$

i	2	3	4	5	6
a_i	9	2	2	2	2
c_i	19	4	4	4	4

Greedy: $S = \{1\}$, $c(s) = 20$
Optimal: $S = \{2, 3, 4, 5, 6, \}$, $c(S) = 20$ □

Approach 2: Integer Linear Programming

The set of solutions X of a combinatorial optimization problem can (almost always) be written as the intersection of integer points in $\mathbb{N}_0^n$ and a polyhedron $\{x \in \mathbb{R}^n : Ax \le b\}$

Let $x \in \{0, 1\}^n$ be a vector representing all solutions of the knapsack problem:

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

$X = \{0, 1\} \cap \{x \in \mathbb{R}^n : \sum_{i=0}^{n} a_i x_i \le b\}$

Knapsack: $\max \sum i = 0^n c_i x_i$

The *linear relaxation (Lineare Relaxierung)* of an ILP is the linear program optained by relaxing the integrality of the variables:

$\max \sum_{i=1}^{n} c_i x_i$
s. t. $\sum_{i=1}^{n} a_i x_i \le b, 0 \le x_i \le 1 \qquad \forall i \in \{1, \dots, n\}$

Theorem 6.3:
An optimal solution $\tilde{x}$ of the linear relaxation of the knapsack problem is:
There exists a $k \in \{1, \dots, n\}$ such that

$$\tilde{x}_i = \begin{cases} 1 & \text{if } i \le k \\ 0 & \text{if } i > k + 1 \\ (b - \sum_{i=1}^{k} a_j)/a_{k+1} & \text{if } i = k + 1 \end{cases}$$

where $c_1/a_1 \ge c_2/a_2 \ge \dots \ge c_n/a_n$.

Proof. Let x^* be an optimal solution with $c^T c^* > c^T \tilde{x}$. If $x_i^* < 1$ for $i \leq k$, there must exist a $j \geq k+1$ with $x_j^* > \tilde{x}_j$.

We define $\bar{x}$ with $\varepsilon \leq x_j^* - \tilde{x}_j$ as

$$\bar{x}_l = \begin{cases} x_k^* & \text{for } l \notin \{i, j\} \\ x_l^* - \varepsilon & \text{for } l = j \\ x_l^* + \frac{a_j}{a_l} \cdot \varepsilon & \text{for } l = i \end{cases}$$

Then $\bar{x}$ is feasible and

$$c^T \bar{x} = \sum_{l=1}^n c_l \bar{x}_l = \sum_{l=1}^n c_l x_l^* + \underbrace{c_i \cdot \frac{a_j}{a_i} \varepsilon - c_j \varepsilon}_{\geq 0} \geq c^T x^* > c^T \tilde{x}$$

Repetition yields $c^T \bar{x} > c^T \bar{x}$, a contradiction. $\qquad\square$

Note:

If $\tilde{x}$ is integer valued, then the solution is also optimal for the knapsack problem. In this case, also the greedy algorithm is optimal.

Approach 3: Dynamic Programming

A dynamic program algorithm to solve a problem first solves similar, but smaller subproblems in order to use their solution to solve the original problem.

The problem should conform to the *optimality principle* of Bellman: Given an optimal solution for the original problem, a partial solution restricted to a subproblem is also optimal for the subproblem.

Let $f_k(b)$ be the optimal solution value of the knapsack problem with total weight equal to b and items from $\{1, \ldots, k\}$.

Theorem 6.4:
$f_{k+1}(b) = max\{f_k(b), f_k(b - a_{k+1} + c_{k+1}\}$.

Proof. An optimal solution of $f_{k+1}(b)$ either contains item $k+1$ or not. If $k+1$ is not contained, the problem is identical to $f_k(b)$. If $k+1$ is contained, other items in the solution should have total weight $b - a_{k+1}$.

Hence, $f_k(b - a_{k+1})$ is an optimal solution for the remaining items $+c_{k+1}$ for the item $k+1$. $\qquad\square$

Corollary 6.5:
The knapsack problem can be solved in $O(nb)$ with value $max_{d=0,\ldots,b} f_n(d)$.

Lecture 7 (2011-10-31)**:**

Matchings, Stable Sets, Vertex Covers and Edge Covers

Definition 7.1:
Let $G = (V, E)$ be an undirected graph. A *matching (Paarung)* is a subset $M \subset E$ such that $e \cap e' = \emptyset \forall e, e' \in M$ with $e \neq e'$.

A matching M is called *perfect* if all vertices are incident to some edge in M.

Example 7.1(+1):
An airline has to allocate two pilots to each of the (round)trips on a single day. Only certain pairs of pilots can work together, due to experience, qualification, location, etc. By defining a graph with vertex set the pilots and edges if two pilots can work together, a daily corresponds to a matching since a pilot can work at two trips at the same time.

Definition 7.2:
A matching $M \subseteq E$ is called *maximal* if no further edges can be added.

A *maximum* matching is a matching M with maximal cardinality, i.e. no other matching M' exists with $|M'| > |M|$.

$\nu(G) = $ *matching number (Paarungszahl)*, size of a maximum matching.

Further related graph parameters

$\alpha(G) = $ *Stable set number / independent set number (Stabile-Mange-Zahl)*, size of a maximum stable set in G: a subset $S \subseteq V$ such that $\forall_{\{v,w\} \in E} |\{v, w\} \cap S| \leq 1$

$\rho(G) = $ *Edge cover number (Kantenüberdeckungszahl)*, minimum size of an edge cover of G: a subset $F \subseteq E$ such that $\forall v \in V : \exists e \in F : e = \{v, w\}$.

$\tau(G) = $ Vertex *cover number (Knotenüberdeckungszahl)*, minium size of a vertex cover of G: a subset of $W \subseteq V$ such that $\forall e \in E : e \cap W \neq \emptyset$.

Lemma 7.3:
$\rho(G) = \infty$ if and only if G contains isolated vertices.

Proof. If G contains isolated vertices, such vertices cannot be covered by any edge, hence $\rho(G) = \infty$. If G does not contain isolated vertices, then E is an edge cover itself, thus $\rho(G) \leq |E|$. $\qquad\square$

Lemma 7.4:
$S \subseteq V$ is a stable set if and only if $V \setminus S$ is a vertex cover.

Proof. Exercise sheet. $\qquad\square$

Lemma 7.5:
$\alpha(G) \leq \rho(G)$.

Proof. If $\rho(G) = \infty$, then $\alpha(G) \leq \rho(G)$ follows automatically. If $\rho(G) < \infty$, then every vertex has degree at least one. Let F be an edge cover in G. Since the vertices of a stable set are not adjacent, there must exist an edge $e \in F$ for all $v \in S$ such that $v \in e_v$ and $e_v \neq e_w$ for all $v, w \in S, v \neq w$. Hence, $|F| \geq |S|$ and it follows $\alpha(G) \leq \rho(G)$. $\qquad\square$

Lemma 7.6:
$\nu(G) \leq \tau(G)$

Proof. To cover all edges of a matching M by vertices, we need at last $|M|$ vertices. Hence, $\nu(G) \leq \tau(G)$. $\qquad\square$

Theorem 7.7 (Gallai's Theorem)**:**
For every graph $G = (V, E)$ without isolated vertices, it holds that

$$\alpha(G) + \tau(G) = |V| = \nu(G) + \rho(G)$$

Proof. $\alpha(G) + \tau(G) = |V|$ follows directly from Lemma 7.4.

To show $\nu(G) + \rho(G) = |V|$, consider a maximum matching M ($|M| = \nu(G)$). M covers all $2|M|$ vertices in M. For every vertex not covered by M, add an incident edge to M. The resulting edge cover has

$$|M| + (|V| - 2|M|) = |V| - |M| = |V| - \nu(G)$$

edges. Hence, $\rho(G) \leq |V| - \nu(G)$.

Now, let F be an edge cover with $|F| = \rho(G)$. Remove for every vertex $v \in V$ $deg_F(v) - 1$ edges incident to v. The resulting edge set is a matching with at least

$$\nu(G) \geq |F| - \sum_{v \in V}(deg_F(v) - 1) = |F| - 2|F| + |V| = |V| - |F| = |V| - \rho(G)$$

$\square$

For all graph parameters, there exists a weighted version: $\nu(G, w), \tau(G, w), \alpha(G, w), \rho(G, w)$.

Question 7.7.1:
How do we find a maximum (weighted) matching?

Definition 7.8:
Let M be a matching in G. A path $P = (v_0 e_1 v_1 ... e_r v_r)$ in G is called *M-alternating (M-alternierend)*, if M contains either all edges e_i with i even or all edges e_i with i odd.

A *M*-alternating path P is called *M-augmenting (M-augmentierender)* path if v_0 and v_r are not matched in M, i.e. $v_0, v_r \notin \bigcup_{e \in M} e$.

Lemma 7.9:
If P is *M*-augmenting, then r odd and

$$M' = M \setminus \{e_2, e_4, ..., e_{r-1}\} \cup \{e_1, e_3, ..., e_r\}$$

is a matching with $|M'| = |M| + 1$.

Proof. Trivial. $\square$

Lemma 7.10:
(Berge)

Let $G = (V, E)$ be a graph and M a matching in G. Then either M is a maximum matching or there exists a *M*-augmenting path.

Lecture 8 (2011-11-04)**:**

Matchings in bipartite graphs

Theorem 8.1:
(Berge) Let $G = (V, E)$ be a grpah an M a matching in G. Then either M is a maximum matching or there exists an M-augmentige path.

Proof. If M is a maximum matching, no M-augmenting path can exist, since M' would be a larger matching.
Let $\bar{M}$ a matching with $|\bar{M}| > |M|$. Consider the components (Komponente) of $G' = (V, M \cup \bar{M})$. Since the the degree of vertices in (V, M) and $(V, \bar{M})$ is at most one, the degree in G' is at most two. Thus, each component of G' is either a path (possibly of length zero) or a cycle. Since $|\bar{M}| > |M|$, at least one component of G' has to have more edges from $\bar{M}$ than from M. Such a component cannot be a cycle and thus is a path, better an M-augmenting path since the end nodes are not matched in M.   □

Definition 8.2:
A graph $G = (V, E)$ is called *bipartite (bipartit)* if and only if $V = U \cup W (U \cap W = \emptyset)$ such that $\{v, w\} \in E \Rightarrow v \in U, w \in W$ (or vice versa). The set U and W are called the color classes of V.

Example 8.2(+1):
n workers, m Jobs. Not every worker is qualified for every job. How many jobs can be processed simultaneously? U = set of worker. W = set of jobs, $\{u_i, w_j\} \in E \Leftrightarrow$ worker i is qualified for job j.

$\nu(G) \leq \tau(G)$ vertex cover

Theorem 8.3:
(König's Matching Theorem, 1931) For every bipartite graph $G = (V, E) : \nu(G) = \tau(G)$

Proof. $\nu(G) \leq \tau(G)$ hols by Lemma 7.6. We therefore only show $\nu(G) \geq \tau(G)$. We may assume that G has at least one edge (otherwise $\nu(G) = \tau(G) = 0$). We will show that G has a vertex v that is matched in every maximum matching. Let $\{v, w\} = e$ be an arbitrary edge in G and assume that M and N are two maximal mathcings with u not matched in M and v not matched in N. Define P as the component of $(V, M \cup N)$ containing u. Since u is only matched in N, $deg_P(u) = 1$ and thus u is an end node of the path P. Since M is maximum, P is *not* an M-augmenting path. Hece, the length of P is even. Consequently, P does not contain v (otherwise P ends at v which contradict the bipartiteness of $G : P \cup \{u, v\}$ would be an odd cycle). The path $P \cup \{v, v\}$ is thus odd, starts with vertex v not matched in N and ends with another vertex not matched in N. Hence $P \cup e$ is N-augmenting path; contracdiction since N is a maximum matching.
So, either u or v must be contained in all maximum matchings, let's say v. Now, consider $G' := G - v$. It holds that $\nu(G) = \nu(G) - 1$. By induction on $n = |V|$ we may assume that G' has a vertex cover W with $|W| = \nu(G')$. Then $W \cup \{v\}$ is a vertex cover of G of size $\nu(G') + 1 = \nu(G)$. It follows $\tau(G) \leq \nu(G)$.   □

Corollary 8.4:
(König's Edge Cover Theorem) Every bipartite graph has $\alpha(G) = \rho(G)$.

Proof. Foolows from Thm 7.7(Gallai's Thm) and 8.3   □

Matching augmenting algorithm for bipartite graphs

Input:bipartite graph $G = (V, E)$ and a matching M

Output: matching M' with $|M'| > |M|$ (if it exists)

Description: Let U, W be the color classes of G. Orientate every edge $e = \{v, w\} (u \in U, w \in W)$ as follows:
if $e \in M$, then orientate from w to u
if $e \notin M$, then orientate from u to w
Let D be the digraph constructed this way. Consider

$$U' := U \setminus \bigcup_{e \in M} e \quad \text{and} \quad W' := W \setminus \bigcup_{e \in M} e$$

An M-augmenting path exists if and only if a directed path in D exists starting at a vertex U' and ending at a vertex W'. Augment M with this path return M'.

Theorem 8.5:
A maximum matching can be found by applying at most $\frac{1}{2}|V|$ times the above algorithm.

Proof. Thm 8.2 says that either a maximum matching or an M-augmenting path exists. This path can be found by the digraph as it has to start with an unmatched vertex and alternates between matched and not matched edges. Thes is guaranteed by the direction of the edges in the digrpah.. If we start with $M = \emptyset$, the size increases by one in every iteration. A max matching can at most $\frac{1}{2}|V|$ edges.   □

Lecture 9 (2011-11-07)**:**

Corollary 9.1 (Frobenius theorem)**:**
A bipartite graph $G = (V, E)$ has a perfect, iff all the vertex covers contain at least $\frac{1}{2}|V|$ nodes.

Proof. Let $\tau(G) \geq \frac{1}{2}|V|$ hold. We derive $\nu(G) \geq \frac{1}{2}|V|$ from König's theorem, but $\nu(G) \leq \frac{1}{2}|V|$ holds as well. If $\tau(G) < \frac{1}{2}|V|$ holds, $\nu(G) < \frac{1}{2}|V|$ holds as well and therefore no perfect matching exists.   □

Corollary 9.2:
Every regular bipartite graph $(\deg(r) = \deg(W) \ \forall v, w \in B)$ (with positive degree) has a perfect matching

Proof. exercise   □

Theorem 9.3 (Hall's theorem (Hochzeitssatz))**:**
Let $G = (U \cup W, E)$ be a bipartite graph with classes U, W. For each subset $X \subseteq U$, $N(X)$ describes the set of nodes in W, which are adjacent to a node in X (neighborhood)

$$\nu(G) = |U| \Leftrightarrow |\nu(X)| \geq |X| \ \forall X \subseteq U$$

Proof. necessity. If $|N(X)| < |X|$ holds, then a matching M can contain at most $|N(X)|$ edges with ending nodes in X. Therefore noes remain in X which cannot be covered by a matching.

Sufficient: Assume $\nu(G) < |U|$, but $|N(X)| \geq |X| \; \forall X \subseteq U$. Then there exists a vertex cover Z for G with $|Z| < |U|$.

Define $X := U \setminus Z$. Then it follows that $N(X) \subseteq W \cap Z$ and therefore

$$|X| - |N(X)| \geq \underbrace{|U| - |U \cap Z|}_{=|X|} - \underbrace{|W \cap Z|}_{\geq |N(X)|} = |U| - |Z| > 0$$

produces a contradiction $\qquad \square$

Maximum Weighted Matching on bipartite graphs

Let $G = (V, E)$ be a graph and let $\omega : E \to \mathbb{R}$ be a weight function For any subset M of E define the weight $\omega(M)$ of M by

$$\omega(M) = \sum_{e \in M} \omega(e)$$

Definition 9.3(+1):
We call a matching extreme if it has maximum weight among all matchings of cardinality $|M|$ (Kardnialität $|M|$)

$$l(e) = \begin{cases} w(l), & \text{if } l \in M \\ -w(l), & \text{if } l \notin M \end{cases}$$

Lemma 9.4:
Let $P\{e_1, e_2, \ldots, e_r\}$ be an M-augmenting path of minimal length. If M is extreme, then $M' = M \setminus \underbrace{\{e_2, \ldots, e_{n-1}\}}_{\in M} \cup \underbrace{\{e_1, e_2, \ldots, e_r\}}_{\notin M}$ is again extreme

Proof. Let N be any extreme matching of size $|M| + 1$. As $|N| > |M|$, $M \cup N$ has a component $Q = \{f_1, f_2, \ldots, f_q\}$ that is an M-augmenting path. As P is a shortest M-augmenting path, we know $l(P) \leq l(Q)$. Moreover, as $M^* = N \setminus \{f_1, f_3, \ldots, f_q\} \cup \{f_2, \ldots, f_{q-1}\}$ is a matching of size $|M|$ and as M is extreme, we know that $\omega(M^*) \leq \omega(M)$ Hence $\omega(N) = \omega(M^*) - l(Q) \leq \omega(M) - l(P) = \omega(M')$. Hence M' is extreme. $\qquad \square$

Hungarian method for maximum weighted matching

"find iteratively extreme matchings $M_1, M_2, \ldots, M_j$ with $|M_k| = K$. Then the matching among $M_1, \ldots, M_j$ of maximum weight is a maximum weight matching."

Define D as in the maximum cardinality matching algorithm.

set
$$U' = U \setminus (\cup_{l \in M} e) \qquad W' = W \setminus (\cup_{l \in M} e)$$

extend D with nodes s and t and arcs $(s, u) \; \forall u \in U'$ with length 0 and $(w, t) \; \forall w \in W'$ with length 0.

Now we find a shortest path form s to t, to get an extreme matching M' from extreme matching M ($|M'| > |M|$).

Theorem 9.5:
Let M be an extreme matching. Then D does not contain a directed circuit of negative length.

Proof. Suppose C is a directed circuit in D with length $l(C) < 0$. We may assume $C = (u_0, w_1, u_1, \ldots, w_t, u_t)$ with $u_0 = u_t$, $u_1, \ldots, u_t \in U$ and $w_1, \ldots, w_t \in W$. Then $w_1 u_1, \ldots, w_t u_t$ belong to M and the edges $u_0 w_1, \ldots, u_{t-1} w_t$ do not belong to M. Then $M'' = M \setminus \{w_1 u_1, \ldots, w_t u_t\} \cup \{u_0 w_1, \ldots, u_{t+1} w_t\}$ matching of cardinality $|M''| = |M|$ with weight: $\omega(M'') = \omega(M) - l(C) > \omega(M)$. This contradicts to M being extreme. $\qquad\square$

Corollary 9.6:
The maximum weighted matching in bipartite graphs can be found by using a shortest path algorithm $\frac{1}{2}|V|$-times.

Lecture 10 (2011-11-10)**:**

Matchings in non-bipartite graphs

Definition 10.1:
A component of a graph is called *odd (ungerade)*, if its number of vertices is odd. We define $o(G) :=$ number of odd components of G. $G - U := G[V \setminus U]$ denotes the subgraph induced by $V \setminus U$.

Theorem 10.2 (Tutte-Berge-Formula)**:**
For a graph $G = (V, E)$ it holds that $D(G) = \min_{U \subset V} \left\{ \frac{1}{2}(|V| + |U| - o(G - U)) \right\}$

Proof. We first prove $\leq$. For $U \subseteq V$ it holds that

$$
\begin{aligned}
\nu(F) &\leq |U| + \nu(G - U) \leq |U| + \frac{1}{2}(|V \setminus U| - o(G - U)) \\
&= \frac{1}{2}(|U \setminus V| + 2|U| - o(G - U)) \\
&= \frac{1}{2}(|V| + |U| - o(G - U))
\end{aligned}
$$

Now, we prove $\geq$. We apply inductions on $|V|$. For $|V| = 0$, the statement is trivial. Furterh, we may assume w.l.o.g (o.B.d.A) that G is connected, otherwise the result follows by induction on the components of G.

Case 1:
There exists a vertex v that is matched in all maximum matches (like in the bipartite case). Thus $\nu(G - V) = \nu(G) - 1$ and by induction there exists a subset $U' \subseteq V - v$ with $\nu(G - V) = \frac{1}{2}(|V \setminus \{v\}| + |U'| - o(G - v - U'))$ Set $U := U' \cup \{V\}$.

Then

$$\nu(G) = \nu(G - v) - 1 = \frac{1}{2}(|V \setminus \{v\}| + |U'| - o(G - U' - v)) - 1$$

$$= \frac{1}{2}(|V| - 1 + |U| - 1 - o(G - U) + 2)$$

$$\leq \min_{T \subseteq V} \frac{1}{2}(|V| + |T| - O(G - T))$$

Case 2:

There does (not) exist a vertex matched in all maximum matchings. So for all $v \in V$, there exists a maximum matching without v. Then, in particular $\nu(G) < \frac{1}{2}|V|$. We will show that there exists in this case a matching of size $\frac{1}{2}(|V| - 1)$. By this, we have proven the theorem (set $U = \emptyset$).

If there does not exist a matching of size $\frac{1}{2}(|V| - 1)$, then for every matching M, there exist two vertices u and v such that both are not matched. Among all maximum matchings, select a triple (M, u, v) for which $\text{Dist}(u, v)$ is minimum there, $\text{Dist}(u, v)$ is the minimum number of edges or a path between u and v. That is, for any other maximum matching N and pair of unmatched vertices s, t, $\text{Dist}(s, t) \geq \text{Dist}(u, v)$.

If $\text{Dist}(u, v) = 1$, then u and v are adjacent and we can extend M with $\{u, v\}$, a contradiction. Thus, $\text{Dist}(u, v) \geq 2$ and hence there exists a vertex t on the shortest path from u to v. Not that t is matched in M, otherwise $\text{Dist}(u, t) < \text{Dist}(u, v)$; a contradition. For t there exist other maximum matchings not covering t. Choose N such that $|M \cap N|$ is maximal.

Also, u and v are covered by N, since otherwise N would have a pair (t, u) (or (t, v)) of unmatched vertices with $\text{Dist}(u, t) < \text{Dist}(u, v)$ (respectively $\text{Dist}(t, v) < \text{Dist}(u, v)$)

Since $|M| = |N|$, there must be a second vertex $x \neq t$ which is not covered by N, but covered by M. Let $e = \{x, y\} \in M$. Then y is also covered by N, otherwise N could be extended by $\{x, y\}$, contradiction to maximality of $|N|$. Let $f = \{y, z\} \in N$. Now replace N by $N \setminus \{f\} \cup \{e\}$. The new matching has one more edge in common with M, contradiction.

Hence, a maximum matching cannot mis two or more vertices. Thus $\nu(G) = \frac{1}{2}(|V| - 1)$

$\square$

Corollary 10.3 (Tutte's 1-Factor Theorem)**:**
A graph $G(V, E)$ has a perfect matching if and only if $G - U$ contains at most $|U|$ odd components for all $U \subseteq V$.

Corollary 10.4:
Let $G = (V, E)$ be a graph without isolated vertices. Then

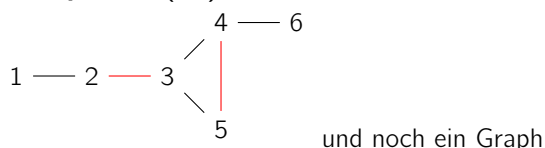$$\rho(G) = \max_{U \subseteq V} \frac{|U| + o(G[U])}{2}$$

Proof. Homework $\square$

Edmond's Matching Algroithm (the blossom shrinking algorithm) 1965

Again we are looking for M-augmenting paths. In bipartite graphs we just have to find a shortest path in the orientation of G by M.

Example 10.4(+1):

```
              4 —— 6
             ╱    │
  1 —— 2 —— 3     │
             ╲    │
              5          und noch ein Graph
```

Example 10.4(+2):
Weitere 2 graphen

We define for sets X and Y:

$$X/Y := \begin{cases} X & \text{if } X \cap Y = \emptyset \\ (X \setminus Y) \cup \{Y\} & \text{if } X \cap Y \neq \emptyset \end{cases}$$

Thus, if $G = (V, E)$ is a graph and $C \subseteq V$, then V/C is the set of ver- tices where all vertices in C are replaced by a singe vertex C. For an edge $e \in E$, $e/C = e$ if e and C are disjoint, whereas $e/C = \{u, C\}$ if $e \in \{u, v\}$ with $u \notin C$, $v \in C$, and $e/C = \{C, C\}$ if $e = \{u, v\}$ with $u, v \in C$.

The last type is unimporteant fo mathings and can be ignored. Further, for $F \subseteq E$, we have $F/C := \{f/C : f \in F\}$ and thus $G/C := (V/C, E/C)$ is again a graph which results from *shrinking (Schrumpfen)* of C.

Lecture 11 (2011-11-21)**:**

Edmonds' Matching Algroithm

Let G be a graph, M a matching in G and W the set of unmatched vertices. A M-augmenting path is a M-alternating W-W chain of positive length where all vertices are distinct.

We call a M-alternating chain $P = \{v_0, v_0, \ldots, v_t\}$ a *M-blossom (M-Blüte)* if $v_0, \ldots, v_{t-1}$ are dstinct, v_0 is not matched in M and $v_t = v_0$

Theorem 11.1:
Let C be a M-blossom in G. M is a maximum matching if and only if M/C is a maximum matching in G/C.

Proof. Let $C = \{v_0, v_1, \ldots, v_t\}$, $G' = G/C$, $M' = M/C$. First, let P be a M-augmenting path in G. W.l.o.g we may assume that P does not start in v_0 (therwise turnaround P). If P does not visit any vertex of C, then P is a M-augmenting path in G' as well. If P visits a vertex of C, then we can decompose P in Q and R with Q ending at the first vertex of C. Replace this vertex in Q

with vertex C in G'. Then Q is a M- augmenting path in G'. Now, on the reverse, let P' be an M'-augmenting path in G'. If P' does not visit vertex C, then P' is M-augmenting path in G. If vertex C is visited by P', we may assume that C is the end vertex (since M' does not cover C). We thus can replace vertex C with a suitable vertex $v_i \in C$ such that the new path Q ends at v_2 in G. If i is odd, then we extend Q with $v_{i+1}, v_{i+2}, \ldots, v_t = v_0$. If i is even, then we extend Q with $v_{i-1}, v_{i-2}, \ldots, v_0$. The resulting path is a M-augmenting path in G ☐

Edmonds' Algorithm

Input: Mathing M with set $W \subseteq V$ of unmatched vertices Output: Mathing N with $|N| = |M| + 1$ or a certificate that M is a maximum matching. Description:

```
1  Case 1: No M-alternating W-W chain exists:
2          Then M is an maximum matching.
3          STOP.
4  Case 2: A M-alternating W-W chain exists:
5          Let P = (v_0, v_1, ..., v_t) be a shortest M-alternating
6          W-W chain
7          Case 2A: P is a path.
8                  Then P is M-augmenting and N := MΔP.
9          Case 2B: P is not a path.
10                 Choose i < j with v_i = v_j and j as small as
                        possible
11                 Replace M by MΔ{v_0, ..., v_i}
12                 Then C := {v_i, v_{i+1}, ..., v_j} is a M-blossom.
13                 Apply the Algorithm recursively
14                 until G' := G/C and M' := M/C.
15
16                 * If a M'-augmenting path P in G'
17                   is returned, transform P' to a
18                   M-autmenting path in G
19                   (by \todo{Thm 11.1})
20                 * If M' is a maximum matching in G',
21                   then M is a maximum matching in G
22                   (\todo{Thm 11.1})
```

Theorem 11.2:
Edmonds' Algorithm is correct and needs at most $\frac{1}{2}|V|$ repetitions to find a maximum matching.

Proof. Follows directly from and .   ☐ ┌─ thm 7.11
                                                                                            └─ thm 11.1

How do we find a M-alternating W-W chain?

Definition 11.3:
A *M-alternating forest (M-alternierender Wald)* (V, F) is a forest with $M \subseteq F$,

each component of (V, F) contains *either* a vertex for W *or* exists of a single edge in M, and each path in (V, F) starting with a vertex in W is M-alternating.
Let

$$even(F) = \{v \in V : F \text{ contains a } W\text{-}v \text{ path of even length}\}$$
$$odd(F) = \{v \in V : F \text{ contains a } W\text{-}v \text{ path of odd length}\}$$
$$free(F) = \{v \in V : F \text{ does not contain a } W\text{-}v \text{ path}\}$$

insert Figure ELISA 1

Note that each vertex $u \in odd(F)$ is incident to a unique edge in $F \setminus M$ and a unique edge in M.

Lemma 11.4:
If no edge $e \in E$ exists that connects $even(F)$ with $even(F) \cup free(F)$, then M is a maximum matching.

Proof. If no such edge exists, then $even(F)$ is a stable set in $G - odd(F)$. In fact every vertex $u \in even(F)$ is an odd component in $G - odd(F)$. Let $U := odd(F)$.

$$o(G - U) \geq |even(F)| = |W| + |odd(F)| = |V| - 2|M| + |U|$$
$$\Leftrightarrow 2|M| \geq |V| + |U| - o(G - U).$$

By Tutte-Berge-Formula, M is a maximum matching. $\square$

Construction of an M-alternating forest

Intialization: $F := M$. Choose for every vertex $v \in V$ an edge $e_v = vu$ with $u \in W$ (if possible).

Iterate: Find $v \in even(F) \cup free(F)$ for which $e_v = vu$ exists.
Case 1 : $v \in free(F)$: Add uv to F. Let $vw \in M$. For all $wx \in E$, set $e_x = \{x, w\}$
Case 2: $v \in even(F)$: Find $W - u$ respectively $W - v$ paths P and Q in F
2a: If P and Q are disjoint, then $F \cup \{uv\} \cup Q$ is a M-augmenting path
2b: If P and Q are not disjoint, then $P \cup Q \cup \{uv\}$ contains a M-blossom C. For

insert Figure ELISA 2

all edges cx with $c \in C$ and $x \notin C$, set $e_x = C_x$. Replace G by G/C.

$$F = M \cup \{\}$$
$$W = \{1, 16, 18\}$$

v	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$\{v, u\} = e_v$		1								18														

Lecture 12 (2011-11-24)**:**

Weighted Matchings in general graphs

Instead of a maximum weighted matching, we search for a minimum weighted perfect matching. Without loss of generallity we may assume that G contains at

least one perfect matching: construct a new graph H by copying G, add $n = |V(G)|$ new vertices, each one adjacent to one vertex of G, an add a clique on the new vertices. All new edges have weight zero. To turn a maximum weight perfec matching into a minimum weight perfect matching, we define new weights

insert figure NIKLAS 1

$$w'(e) := W - w(e) \quad \text{with} \quad W = \max_{e \in E} w(e)$$

Lemma 12.1:
A maximum weight matching in (G, w) corresponds to a minimum weight perfect matching in (H, w').

Proof. Let M be a perfect matching in H.

$$w'(M) = n \cdot W - \sum_{e \in M \cap E(G)} w(e)$$

Hence

$$\min_{M \in E(H)} w'(M) = n \cdot W - \max_{M \in E(G)} w(M)$$

In contrast to the maximum cardinality matching problem, we also might have to "unshrink" a blossom: Expand $\square$

insert NIKLAS 3

Definition 12.2:
A collection Ω of odd subsets of V is called *nested (verschachtelt)*, if for all $U, W \Omega$ either $U \cap W = \emptyset$ or $U \subseteq W$ or $W \subseteq U$.

Example 12.2(+1):

$$\Omega = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{1,2,3\}, \{4,5,6\}, \{4,\ldots,7\}\}$$

if

insert NIKLAS 4

$$\Omega = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{1,2,3\}, \{4,5,6\}\}$$

then

$$\Omega^{max} = \{\{1,2,3\}, \{4,5,6\}, \{7\}\}$$

We will assume that $\{v\} \in \Omega$ for all $v \in V$. As a consequence Ω covers V. Further, there exist inclusionwise maximal elements in Ω. Let Ω^{max} denote the subsets of Ω that are inclusionwise maximal. Therefor Ω^{max} is a partition of V. The algorithm for finding a minimum weight perfect matching is "primal-dual", i.e., during the algorithm both a matching and a dual object, a function $\Pi : \Omega \to \mathbb{Q}$ is carried on. We consider functions $\Pi : \Omega \to \mathbb{Q}$ satisfying following conditions:

$$\begin{cases} \Pi(U) \geq 0 & \text{if } U \in \Omega \text{ with } |U| \geq \varepsilon \\ \sum_{U \in \Omega : e \in \delta(U)} \Pi(U) \leq w(e) & \text{for all } e \in E \end{cases} \quad (1)$$

Lemma 12.3:
Let N be a perfect matching. Then $w(N) \geq \sum_{U \in \Omega} \Pi(U)$

Proof.

$$w(N) = \sum_{e \in N} w(e) \overset{1}{\geq} \sum_{e \in N} \sum_{U \in \Omega : e \in \delta(U)} \Pi(U) = \sum_{u \in \Omega} \Pi(U) \cdot |N \cap \delta(U)| \overset{*}{\geq} \sum_{U \in \Omega} \Pi(U)$$

(2)

insert NIKLAS 5

(*): since all elements of Ω are of odd size, there always must be at least one vertex of U matched with a vertex of $V \setminus U$ in every perfect m.

So if we have a perfect matching N and a function Π fullfilling 2 with equality, we are done. Thus, a given $\Pi : \Omega \to \mathbb{Q}$, we define

$$w_\Pi(e) := w(e) - \sum_{U \in \Omega : e \in \delta(U)} \Pi(U) \overset{1}{\geq} 0$$

This in the end $w_\Pi(e) = 0$ for all $e \in N$. Further, leth $G \setminus \Omega$ be the graph with all $U \in \Omega^{max}$ shrunken (with U as shrunken vertex). Thus $G \setminus \Omega$ has vertex set Ω^{max} and $U, W \in \Omega^{max}$ are adjacent if and only if there exist $u \in U$, $w \in W$ with $\{u, w\} \in E$. Finally, we only consider collections Ω such that for all $U \in \Omega$ with $|U| \geq 3$, the graph H_U contains a Hamilton cycle C_U with edges e having $w_\Pi(e) = 0$. Hence H_U is the graph obtain by shrinking all inclusionwise minimal subsets in $G[U]$ □

Edmonds' Minimum Weighted Perfect Matching Algorithm

```
1  Initialize:
2          M := ∅,
3          F := ∅,
4          Ω := {{v}, v ∈ V},
5          Π({v}) := 0
6  As long as M is not perfect in G \ Ω iterate as follows
7      a) Select α maximal such that
8              Π(U) − α ≥ 0∀U ∈ Ω, |U| ≥ ε, U ∈ odd(F),
                 Π(U) + α ≥ 0∀U ∈ Ω, |U| ≥ ε, U ∈ even(F), and
```

$$\sum_{U \in \Omega \cap odd(F) : e \in \delta(U)} (\Pi(u) - \alpha) + \sum_{U \in \Omega \cap even(F) : e \in \delta(U)} (\Pi(u) - \alpha) \leq w(e) \forall e \in E$$

```
9          Reset Π(U) := Π(U) − α for U ∈ odd(F) and
               Π(U) := Π(U) + α for U ∈ even(F)
10         The new Π fullfills 1 and in addition either
               of the following holds:
11         i) ∃e ∈ G \ Ω with w_Π(e) = 0 such that e intesects
               with even(F) but not with odd(F)
12         ii) ∃U ∈ odd(F) with |U| ≥ ε and Π(U) = 0
```

13   b) If (i) holds, and only 1 end vertex of *e*
            belongs to *even(F)* and the other one is in
            free(F) then extend F with e GROW
14   If (i) holds, and both end vertices of *e*
               belong to *even(F)* and $F \cup \{e\}$ contains a cycle
               U, add *U* to Ω with $\Pi(U) := 0$, replace *F* by
               $F \setminus U$ and *M* by $M \setminus U$ SHRINK
15   If (i) holds, and both ... *even(F)* and $F \cup \{e\}$
               contains a *M*-augmenting path, augment *M* an
               replace $F := M$. AUGMENT
16   c) If (ii) holds, remove *U* from Ω, replace *F* by
            $F \cup P \cup N$ and *M* by $M \cup N$, where *P* is the path of
             even length in C_U connection the two vertices
             incident to the edges in *F* adjacent to *U* and
             N the matching C_U which covers all vertices in
             U not covered by *M* EXPAND
17 END

Lecture 13 (2011-11-25)**:**

The Traveling Salesman Problem

Definition 13.1:
A (directed) cycle (path) with $|V|$ (resp. $|V| - 1$) edges is called a (directed)
Hamilton cycle (Hamiltonkreis) (*Hamilton path (Hamiltonpfad)*).

Occasionally, a Hamilton cycle will be called a *tour (Tour)*.

The *traveling salesman problem (TSP) (Problem des Handelsreisenden)* is to find,
given distances c_{ij} for all i, j, a minimum length Hamilton cycle in a complete
graph.

If $c_{ij} = c_{ji} \forall i, j$, the problem is *symmetric (symmetrisch)*, otherwise it is *assymmetric (assymetrisch)*.

If c_{ij} represent euclidean distances in $\mathbb{R}^2$, the TSP is euclidean.

Lemma 13.2:
If $c_{ij} < \infty$ fulfills the triangle inequality $c_{ij} + c_{jk} \geq c_{ik}$, the vertices can be placed
in $\mathbb{R}^2$ such that the TSP is euclidean. $\square$

Let $C(S, k) =$ minimum length of a path from vertex 1 to vertex k which visits
all vertices in S exactly once (and no other vertices). Hence, the TSP is solved
as soon as we have computed $C(\{2, ..., n\}, 1)$.

Lemma 13.3:

$$C(S, k) = min_{j \in S} \{C(S \setminus \{j\}, j) + c_{jk}\}$$

Proof. The optimal path from 1 to k via S has as last-but-one vertex a vertex
$j^* \in S$. The path from 1 to j^* via $S \setminus \{j^*\}$ should be optimal (i.e. has value

$C(S \setminus \{j^*\}, j^*))$, otherwise we can improve it. Vertex j^* can be determined by taking the minimum among all $j \in S$. $\qquad\qquad\square$

Acronyms

(s, t)-**Schnitt** (s, t)-cut. 7

M-**Blüte** M-blossom. 18

M-**alternierender Wald** M-alternating forest. 19

??? shortest path length matrix. 6

assymetrisch assymmetric. 23

aufspannend spanning. 1

azyklisch acyclic. 3

Baum tree. 1

bipartit bipartite. 13

Hamiltonkreis Hamilton cycle. 23

Hamiltonpfad Hamilton path. 23

Kantenüberdeckungszahl Edge cover number. 11

Kette chain. 3

Knapsack Problem Knapsack problem. 8

Knotenüberdeckungszahl cover number. 11

Laufzeit running time of algorithms. 2

Lineare Relaxierung linear relaxation. 9

minimaler Spannbaum Minimum Spanning Tree (MST) problem. 1

Paarung matching. 10

Paarungszahl matching number. 11

Problem des Handelsreisenden traveling salesman problem (TSP). 23

Problem des maximalen Waldes Maximum Forest Problem. 1

Schrumpfen shrinking. 18

Stabile-Mange-Zahl Stable set number / independent set number. 11

symmetrisch symmetric. 23

Tour tour. 23