Exercise Session 4

14.05.

- Gurobi vs. Cplex
- Assignment Problem
- ~~Paths & Matchings~~ next week

Gurobi vs. Cplex: ~> alternative for the programming exercise

Cplex:
- available in Unixpool
- private PC/Laptop: only compilation

Gurobi
- available as student license
- very similar to Cplex
- not avail. in Unix pool
- nicer API

Example:

```java
package examples;

import gurobi.*;

/**
 * Example of the usage of the Gurobi API with Java.
 * The code shows how to build an LP for a multi commodity
 * flow problem with four given paths.
 *
 * @author Torsten Gellert
 */

public class SolvingLpWithGurobi{

    /** Contant for the cost of artifical arcs for this particular example. */
    public static final double M = 1000.0;
    /** Hide constructor, since this class is an simple example. */
    private SolvingLpWithGurobi() {
    }

    /**
    * Creates a path based LP formulation of the multi commodity flow problem
    * for the instance shown in the exercise session.
    * Solves and prints all variables and additional available information.
    *
    * @param  args The argument list . Not used by the method.
    */

```

```java
    public static void main( String[] args ) {
        try {
            GRBEnv    env   = new GRBEnv("lp.log");
            GRBModel  model = new GRBModel(env);

            populateByColumn(model);

            model.optimize();

            int status = model.get(GRB.IntAttr.Status);

            if (status == GRB.Status.INF_OR_UNBD ||
                status == GRB.Status.INFEASIBLE  ||
                status == GRB.Status.UNBOUNDED     ) {
                System.out.println("The model cannot be solved because it is "
                    + "infeasible or unbounded");
                System.exit(1);
            }

            System.out.println("Solution status = " + status);
            System.out.println("Solution value  = " +
                               model.get(GRB.DoubleAttr.ObjVal));


            GRBVar[] vars =  model.getVars();  get all variables in model
            for (int j = 0; j < vars.length; ++j) {
                System.out.println(vars[j].get(GRB.StringAttr.VarName)
                         + " " +vars[j].get(GRB.DoubleAttr.X)
                         + "\tReduced cost = "
                         +vars[j].get(GRB.DoubleAttr.RC));
            }

            GRBConstr[] cons  = model.getConstrs();
            for (int i = 0; i < cons.length; ++i) {
                System.out.println(cons[i].get(GRB.StringAttr.ConstrName) +
                                   ":\tSlack = " +
                                   cons[i].get(GRB.DoubleAttr.Slack) +
                                   "\tPi = "
                                   + cons[i].get(GRB.DoubleAttr.Pi));
            }                                                  dual variable Pi

            model.dispose();
            env.dispose();
        } catch (GRBException e) {
            System.out.println("Error code: " + e.getErrorCode() + ". " +
                 e.getMessage());
        }
    }

    /**
    * Builds all variables and constraints(ranges). These are
    * stored in a arrays and these array have to be placed in the
    * first position of the given two dimentional arrays.
    *
    * @param model The model for handling with LPs.
    * @throws GRBException if there occurs an error while build the LP model.
```

```java
    static void populateByColumn(GRBModel model) throws GRBException {
        //                          lb,  ub,              obj,     type,
  name
        GRBVar p_1_1 = model.addVar(0.0, Double.MAX_VALUE, M, GRB.CONTINUOUS, "p_1_1");
        GRBVar p_2_1 = model.addVar(0.0, Double.MAX_VALUE, M, GRB.CONTINUOUS, "p_1_2");
        GRBVar p_1_2 = model.addVar(0.0, Double.MAX_VALUE, 4, GRB.CONTINUOUS, "p_2_1");
        GRBVar p_2_2 = model.addVar(0.0, Double.MAX_VALUE, 4, GRB.CONTINUOUS, "p_2_2");

        // Integrate new variables
        model.update();

        //add some constraints to the model
        GRBLinExpr expr = new GRBLinExpr();
        expr.addTerm(1.0, p_1_2);
        expr.addTerm(1.0, p_2_2);
        model.addConstr(expr, GRB.LESS_EQUAL, 3.0, "arc_1_3");

        expr = new GRBLinExpr();
        model.addConstr(expr, GRB.LESS_EQUAL, 1.0, "arc_1_4");

        expr = new GRBLinExpr();
        expr.addTerm(1.0,p_2_2);
        model.addConstr(expr, GRB.LESS_EQUAL, 1.0, "arc_2_1");

        expr = new GRBLinExpr();
        model.addConstr(expr, GRB.LESS_EQUAL, 1.0, "arc_2_4");

        expr = new GRBLinExpr();
        expr.addTerm(1.0,p_1_2);
        model.addConstr(expr, GRB.LESS_EQUAL, 1.0, "arc_3_4");

        expr = new GRBLinExpr();
        expr.addTerm(1.0,p_1_1);
        expr.addTerm(1.0,p_1_2);
        model.addConstr(expr, GRB.GREATER_EQUAL, 2, "commodity_1");

        expr = new GRBLinExpr();
        expr.addTerm(1.0,p_2_1);
        expr.addTerm(1.0,p_2_2);
        model.addConstr(expr, GRB.GREATER_EQUAL, 2, "commodity_2");
    }
}
```

$1 \cdot p_{12} + 1 \cdot p_{22} \leq 3$ // arc 1->3

```
/*
Output:

Optimize a model with 7 rows, 4 columns and 8 nonzeros
Presolve removed 7 rows and 4 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0080000e+03   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective  2.008000000e+03
Solution status = 2
Solution value  = 2008.0
p_1_1 1.0       Reduced cost = 0.0
p_1_2 1.0       Reduced cost = 0.0
p_2_1 1.0       Reduced cost = 0.0
p_2_2 1.0       Reduced cost = 0.0
arc_1_3:        Slack = 1.0     Pi = 0.0
arc_1_4:        Slack = 1.0     Pi = 0.0
arc_2_1:        Slack = 0.0     Pi = -996.0
arc_2_4:        Slack = 1.0     Pi = 0.0
arc_3_4:        Slack = 0.0     Pi = -996.0
commodity_1:    Slack = 0.0     Pi = 1000.0
commodity_2:    Slack = 0.0     Pi = 1000.0

*/
```

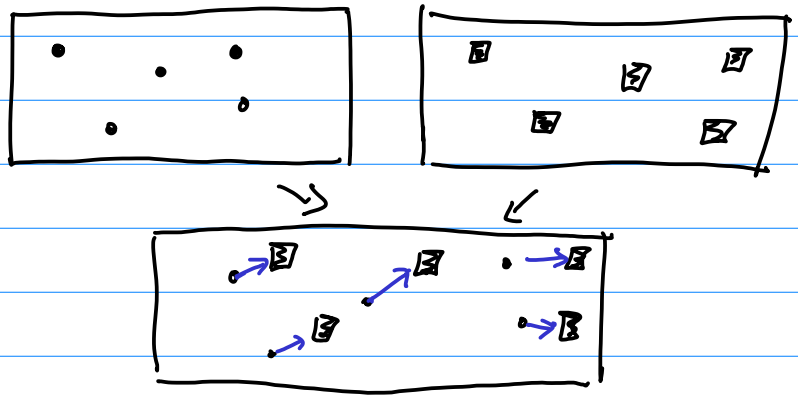
Java Tab Width

Assignment Problem:

Job Assignment Problem: jobs $j \in J$, each with processing time 1
m machines, set $X \subset J \times M$, $(j,m) \in X \Leftrightarrow$ mach. m can process j

Question: is it possible to do all jobs in 1 unit of time

examples:
- lectures — room (add. time slots
- worker — tasks
- detecting movements on succ. images



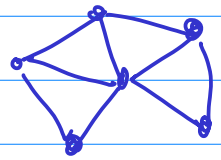
Same problem as Cardinality Matching Problem: in bipartite graphs!

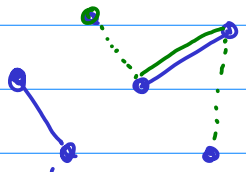
Instance: undir. Graph $G=(V,E)$, G bipartite

Task: find a maximum cardinality set of disjoint edges

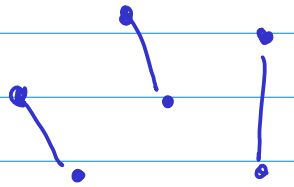
Inclusion wise Maximal Matching — Card. Maximal Matching

— : M augm. Path





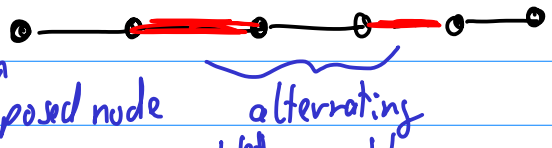
Matching is
- incl. maximal.



- Card. maximal (perfect)
- incl. maximal

finding incl. max Matching: trivial (Greedy)

— " — card. — " — : M-augmenting path



exposed node — alternating bet. match — non-matching — exposed node

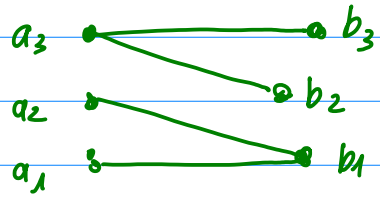
Thm: (Hall): $V = A \dot\cup B$, $G=(V,E)$ has a Matching M $|M| = |A|$ iff

$|\Gamma(X)| \geq |X| \quad \forall\, X \subseteq A$

in short: each set X has enough possible matches

$\Gamma(x)$ = neighborhood of X

example:



$|\Gamma(\{a_1, a_2\})| = |\{b_1\}|$

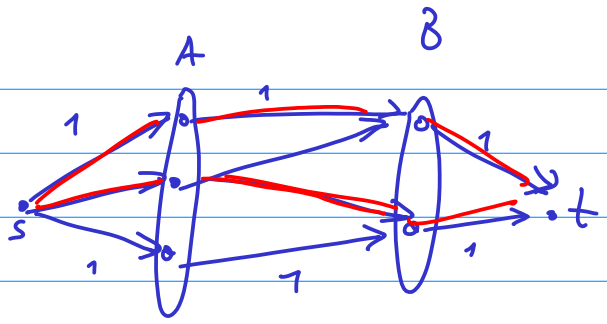
→ special case $|A| = |B|$

Marriage theorem: version of Hall's thm for $|A| = |B|$, edges are possible candidates for a marriage

Optimality: (Berge) Matching M is maximum iff there is no M-augm. path

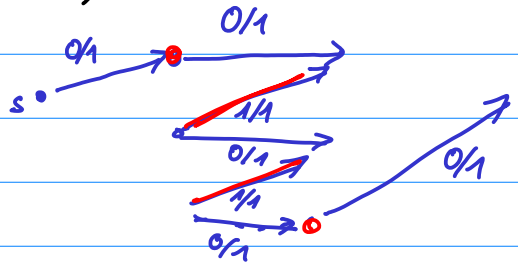
Solving: bipartite Matching:

- build s-t flow
- find integral max s-t flow
- → path decompos. contains Matching



Running time: Edmonds Karp $O(n^3)$ at most $O(n)$ augm. paths

flow augm path: contains M-augm. path



Hopcraft-Karp Algo. (1973)

idea: find a lot of augm. paths at once

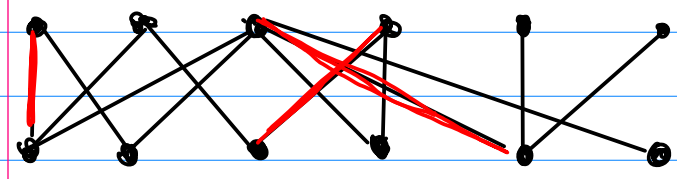
maximal set of disjoint min. length augm. paths:

- set of M-augm. paths
- vertex set is disjoint
- all path have same length as a min. M-augm. path
- every other augm. path with same length shares a vertex

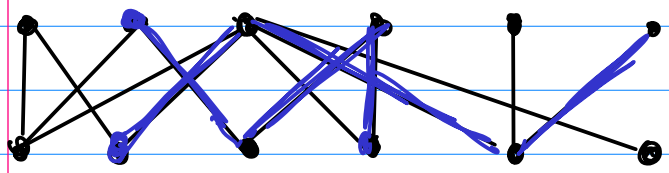
Algo:

- $M = \emptyset$
- find max. set of dis. min. length augm. pats
  $S := \{p_1, \dots, p_k\}$
- while $S \neq \emptyset$
  $M := M \oplus S$ // augment M with path $p_1, \dots, p_k$
  find new set S
- return M

example:

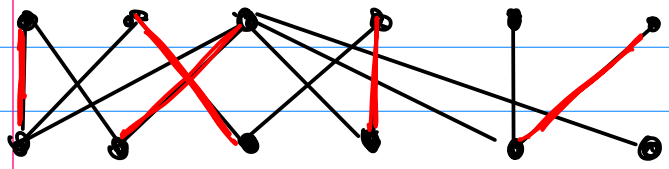


initial Matching, M



Set of augm paths, 2 paths S

$M \oplus S$



Running Time:

Lemma 1: (augm. path are increasing in length)

M is a matching, p is a shortest M-augm path

$M' := M \oplus p$, Let q' be a M' augm. path

$\Rightarrow |q| \geq |p| + 2|p \cap q|$ (set S maximal $\Rightarrow$ new paths at least longer by 2

Proof: • p, q vertex disjoint $\Rightarrow$ ✓ p was shortest

• $p \cap q \neq \phi$, $M'' := M' \oplus q = M \oplus p \oplus q$

$\rightarrow |M''| = |M| + 2 \Rightarrow M' \oplus M$ contains 2 odd length disjoint M-augm paths p_1, p_2

$\Rightarrow |p_1| |p_2| \geq |p| \rightarrow p$ was shortest

$M'' \oplus M = \underbrace{M' \oplus q}_{M''} \oplus M = M \oplus p \oplus q \oplus M = q \oplus p \Rightarrow$ 2 M augm paths in $p \oplus q$, not shorter than p

$\Rightarrow |p \oplus q| \geq 2|p|$

$|p \oplus q| = |p| + |q| - 2|p \cap q| \Rightarrow |p| + |q| - 2|p \cap q| \geq 2|p|$

$\Rightarrow |q| \geq |p| + 2|p \cap q|$ □

- we call finding a set S a phase

$\Rightarrow$ after a phase, the length of shortest M-augm. paths grows by at least 2

Lemma: Algo runs in $O(\sqrt{n} \cdot m)$ or $O(n^{2.5})$

Proof:
- Set S can be found in $O(m)$ $\rightarrow$ bfs starting at exposed vertices, stops if m-augm paths found
- number of phases is in $O(\sqrt{n})$
  - M is matching after $\sqrt{n}$ phases and not maximum
  - M^* is maximum, $\Rightarrow |M^*| - |M|$ augm. paths to find
    $|M| \oplus |M^*|$ has $|M^*| - |M|$ vertex disjoint augm. paths
  - (Lemma 1) each path has length $\geq 2\sqrt{n} + 1$
  - # augm. paths in $M^* \oplus M \leq \frac{n}{2\sqrt{n}+1} \leq \frac{\sqrt{n}}{2}$ (disjoint and length $\geq 2\sqrt{n}$)
  - each phase adds at least 1 augm. path
  - $\Rightarrow \frac{\sqrt{n}}{2}$ following phases
  - $\Rightarrow O(\sqrt{n} + \frac{\sqrt{n}}{2}) = O(\sqrt{n})$ phases

each phase in $O(m) \Rightarrow O(m \cdot \sqrt{n})$ $\square$

$\rightarrow$ Similar to Dinics Blocking Flow Algorithm