Instytut Telekomunikacji
Zakład Sieci i Usług Teleinformatycznych
semestr: lato 2021-2022

OAST Optymalizacja i analiza sieci teleinformatycznych

Optymalizacja (blok 2: 10 tygodni)
Analiza (blok 1: 5 tygodni)

OAST – zajęcia i komunikacja

- wykład i ćwiczenia
  - prof. Michał Pióro (blok 2)
  - dr inż. Piotr Gajowniczek (blok 1)
- projekt
  - mgr inż. Bartłomiej Ostrowski (blok 2)
  - dr inż. Piotr Gajowniczek (blok 1)
- komunikacja
  - Michał Pióro m.pioro@tele.pw.edu.pl (p. 345)
  - Piotr Gajowniczek p.gajowniczek@tele.pw.edu.pl (p. 44A)
  - Bartłomiej Ostrowski b.ostrowski@tele.pw.edu.pl (p. CS306)
  - lista mailowa 103A-TLTIC-MSP-OAST@elka.pw.edu.pl
  - strona przedmiotu
    https://studia3.elka.pw.edu.pl/f-pl/21Z/103A-TLTIC-MSP-OAST/priv/
- konsultacje
  - MP czw. 14-16
  - PG czw. 12-14
  - BO śr. 18-20

OAST: regulamin

- ćwiczenia i projekt:
  - obowiązkowe*
  - dopuszczalne 2 nieobecności nieusprawiedliwione

- punktacja:

kolokwium (koniec semestru)	50	20+30	(min 25)
2 projekty	50	20+30	(min 25)
prace domowe	5		

	105		

- ocena:

51-60	3.0
61-70	3.5
...	
≥ 91	5.0

* obecność na niektórych zajęciach projektowych nie będzie obowiązkowa

blok 2: optymalizacja sieci

- sieci przepływów wielotowarowych jako model matematyczny służący do optymalizacji sieci teleinformatycznych
- sformułowania reprezentatywnych problemów optymalizacji sieci
- metody programowania liniowego (algorytm simpleks) i programowania całkowitoliczbowego (algorytm podziału i ograniczeń) w zastosowaniu do rozwiązywania problemów optymalizacji sieci; metody heurystyczne
- problem WOST – wymiarowanie optycznych sieci transmisyjnych w technologii DWDM (projekt)

literatura do bloku 2

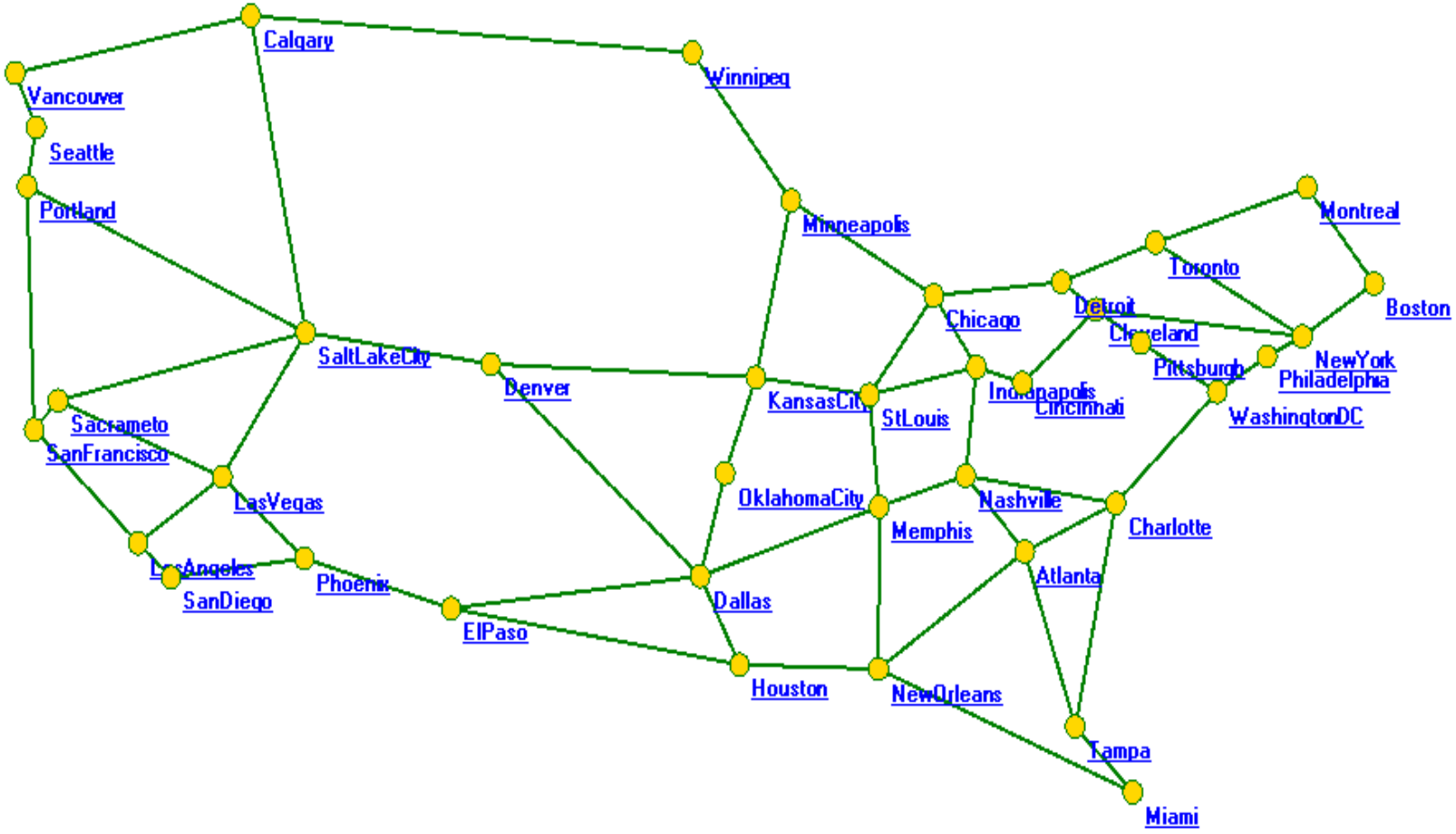
- M. Pióro and D. Medhi: *Routing, Flow, and Capacity Design in Communication and Computer Networks*, Morgan Kaufmann, 2004 , chapters 2-5 (9-10)

- L. Lasdon: *Optimization Theory for Large Systems*, McMillan, 1972

- L.A. Wosley: *Integer Programming*, Wiley, 1998

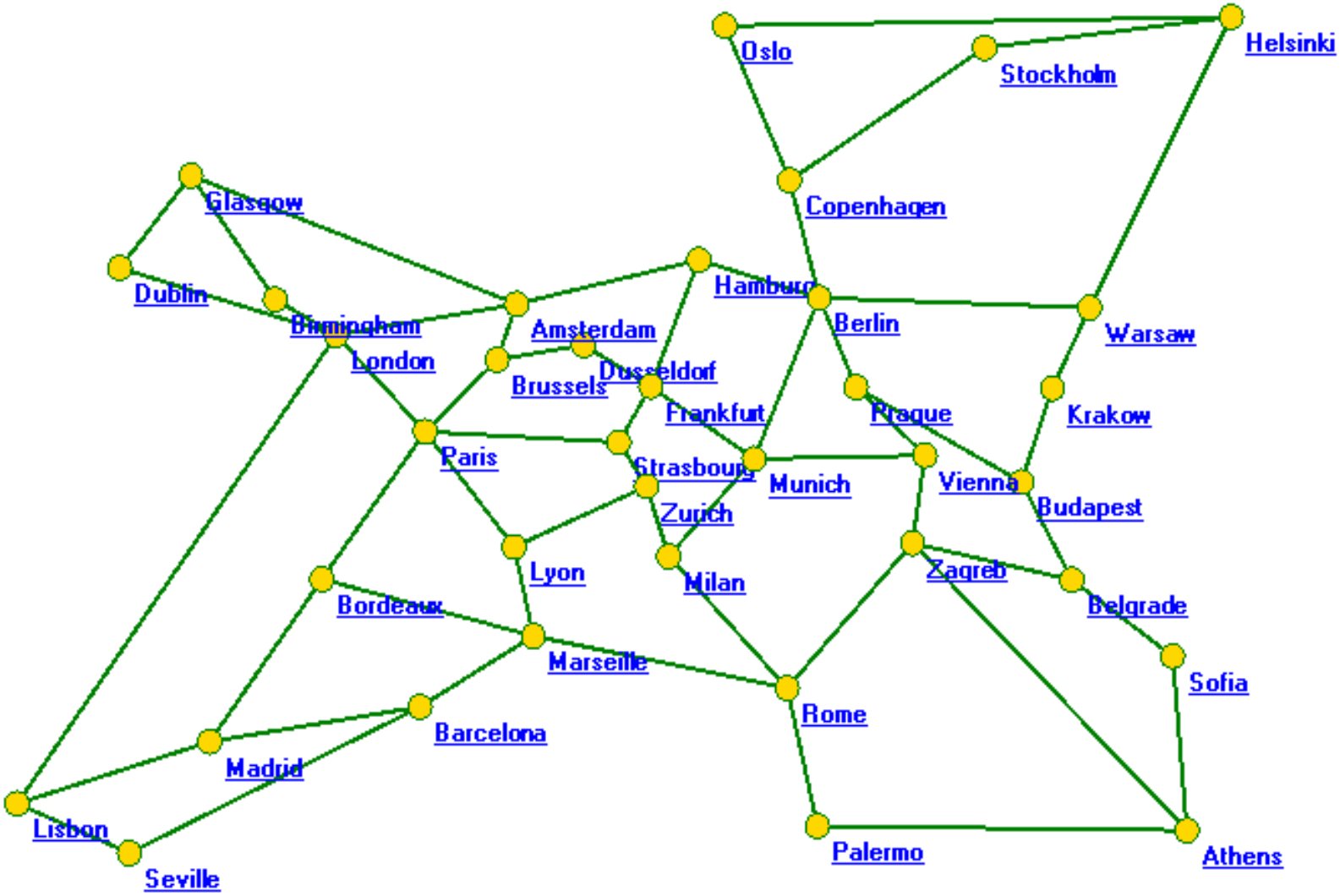
backbone (optical) network 1

sndlib.zib.de

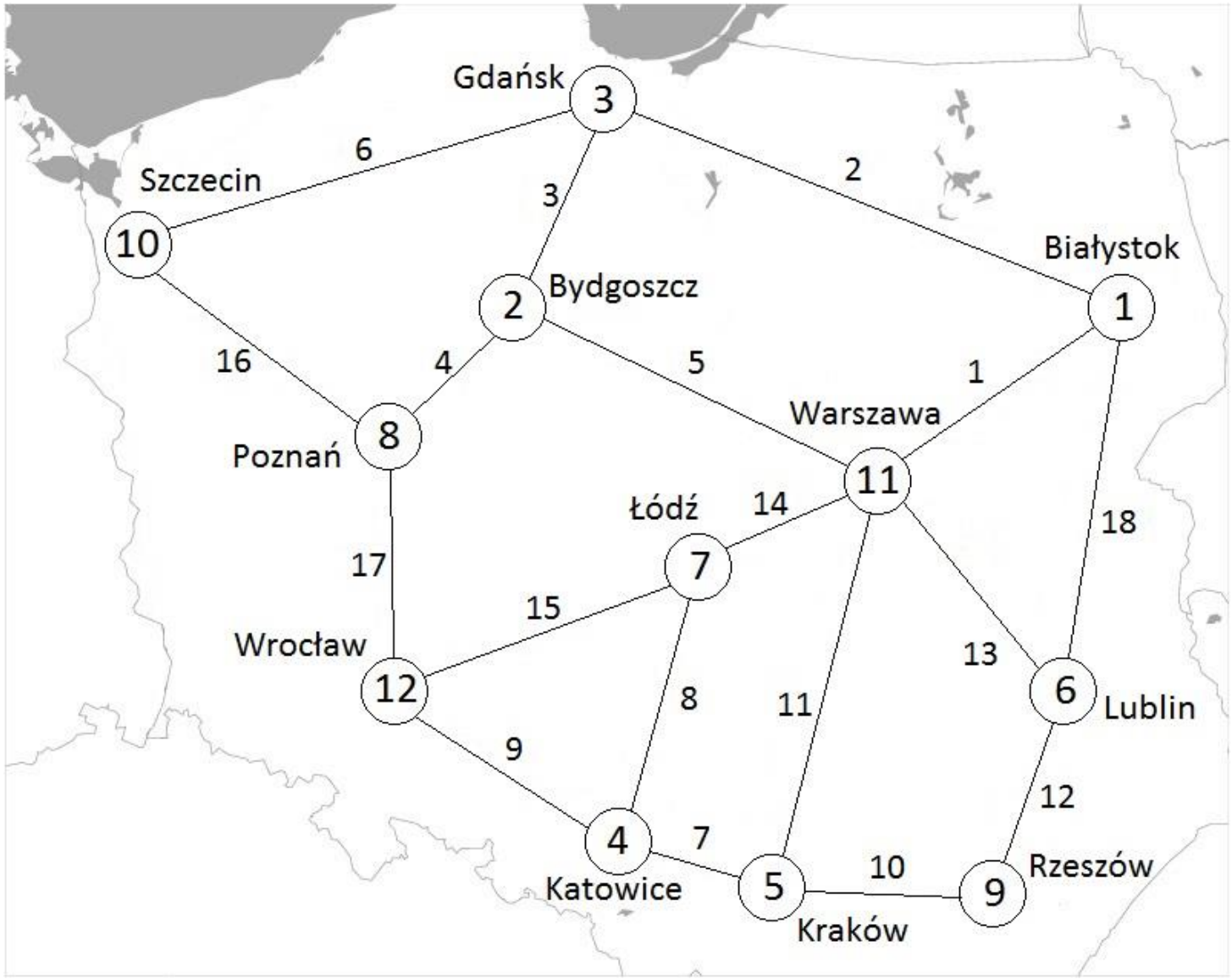


backbone (IP) network 2

sndlib.zib.de



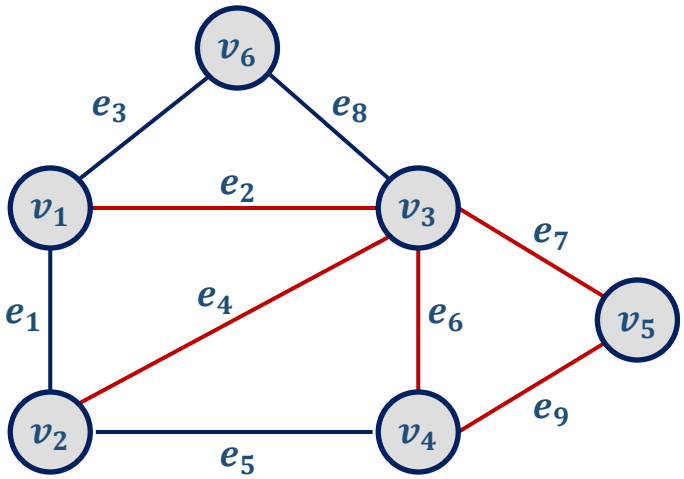
backbone (optical) network 3 (case study)



basic notions (undirected graphs)

- nodes (vertices) $v = 1,2,\dots,V$
- links (edges, arcs) $e = 1,2,\dots,E$
  - unordered pairs of nodes $\{v,w\}$ e: $(\{a(e),b(e)\})$
- paths $p = 1,2,\dots,P$
  - $\mathcal{P}_p$ path no. p
  - simple path $\mathcal{P}_p \subseteq \mathcal{E}$
  - binary coefficients specifying a path: $\delta_{ep} = 1$ if, and only if, $e \in \mathcal{P}_p$
- examples (paths between nodes v_1 and v_2)
  - not simple: $(v_1, e_2, v_3, e_7, v_5, e_9, v_4, e_6, v_3, e_4, v_2)$ (contains loop)
  - simple: $(v_1, e_2, v_3, e_4, v_2, e_1, v_1)$ (loop)
  - simple: (v_1, e_1, v_2) (direct)
    $\mathcal{P} = \{e_2\}$
  - simple: $(v_1, e_2, v_3, e_7, v_5, e_9, v_4, e_5, v_2)$
    $\mathcal{P} = \{e_2, e_5, e_7, e_9\}$
  - simple: $(v_1, e_2, v_3, e_6, v_4, e_5, v_2)$
    $\mathcal{P}_3 = \{e_2, e_5, e_6\}$ $\delta_{23} = 1, \delta_{53} = 1, \delta_{63} = 1$
    and $\delta_{e3} = 0$, otherwise
- **remark:** links can repeat as well

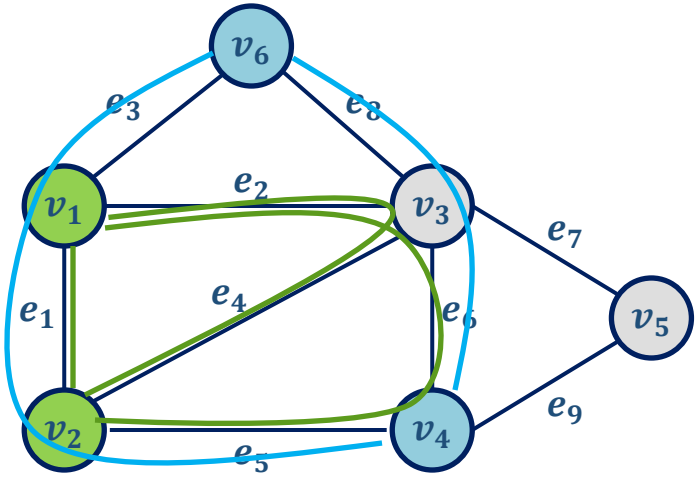
directed graphs are equally important



- simple path $\mathcal{P}_p$ is identified by its characteristic vector: $\delta(p) = (\delta_{1p}, \delta_{2p}, \dots, \delta_{Ep})$
  - $\delta(3) = (\delta_{13}, \delta_{23}, \dots, \delta_{93}) = (010011000)$

multicommodity flows (przepływy wielotowarowe)

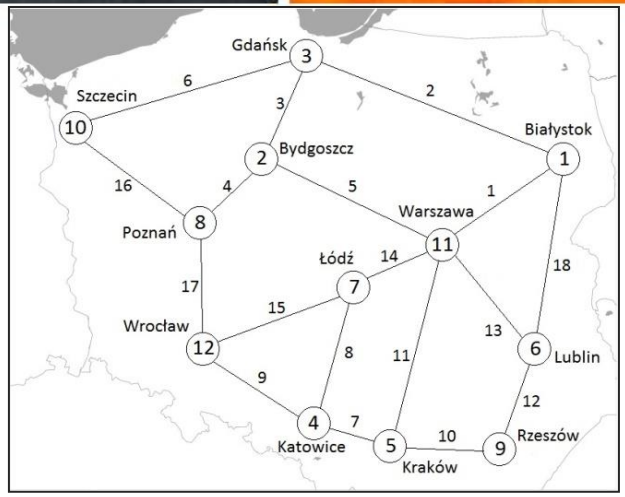
- commodity (towar) of a given volume to be sent from a source node to a destination node
- by means of (path) flows (przepływy ścieżkowe)
- the paths realizing flows for a given commodity do not to be disjoint
- the sum of all flows using a given link cannot exceed its capacity
- different flows realizing the same commodity can use a common link(s)



- good example (besides telecommunication networks): road traffic

basic notions (undirected graphs)

Book: Chapter 2 – network design problems



- nodes (vertices) $v = 1,2,\dots,V$
- links (edges, arcs) $e = 1,2,\dots,E$
  - y_e , c_e number of modules installed on link e (variable or given)
  - M link module (in demand units, e.g. Mbps)
  - $C_e = M \cdot c_e$ (or $C_e = M \cdot y_e$) link capacity (in demand units)
- demands $d = 1,2,\dots,D$ (realized by means of path flows)
  - unordered pairs of nodes $\{o(d), t(d)\}$
  - h_d demand's volume (demand units, e.g., Mbps)
  - $\mathcal{P}_{dp}$ path no. p of demand d $(p = 1,2,\dots,P_d)$
    predefined lists of paths (routes) linking $o(d)$ and $t(d)$
  - simple path $\mathcal{P}_{dp} \subseteq \mathcal{E}$, $d = 1,2,\dots,D$, $p = 1,2,\dots,P_d$
  - $\delta_{edp} = 1$ if, and only if, $e \in \mathcal{P}_{dp}$ (binary coefficient)

routing of demands and resulting link loads

DAP

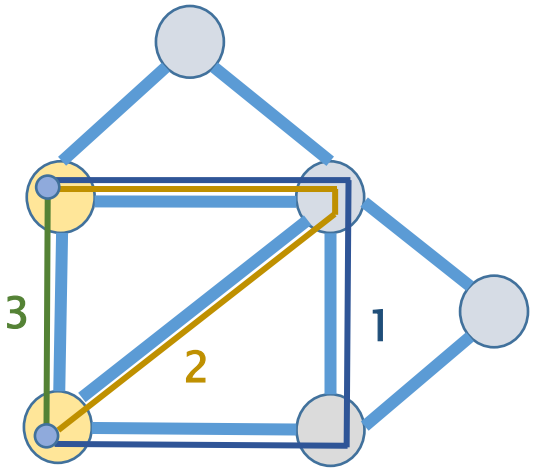
flow allocation x: defines path flows (routing/allocation of demands)

$x(d,p)$ number of demand units realized on path p of d

$$x(d,1) + x(d,2) + \cdots + x(d,P_d) = h_d$$

$$\sum_p x(d,p) = h_d$$

$x(d,p)$ – nonnegative integers



demands $d = 1,2,\dots,D$

paths $p = 1,2,\dots,P_d$

$d=1$	$d=2$		$d=D$
0	3		1
3	2		3
4	1	…	0
0			1
1			
$h_1 = 8$	$h_2 = 6$		$h_D = 5$

link capacities (given)

- $C_e \quad (e = 1,2,\dots,E)$

total number of demand units using link e

- $l(e,x) = \sum_d \sum_p \delta_{edp} x(d,p)$

link overload

- $l(e,x) - C_e \quad (e = 1,2,\dots,E)$

maximum (over all links)

- $F(x) = max_e \{ l(e,x) - C_e \}$

objective

- minimize $F(x)$

link loads calculation

DAP

- input data: paths, path flows $x(d,p)$:s
- find link load $l(e,x)$ $(e = 1,2,\dots,E)$:
  - total number of demand units using link e
  - $l(e,x) = \sum_d \sum_p \delta_{edp} x(d,p)$

```
begin
    for e := 1 to E do l(e,x) := 0;
    for d := 1 to D do
      for p := 1 to P_d do
        for e := 1 to E do if e ∈ P_dp then l(e,x) := l(e,x) + x(d,p)
end
```

demand allocation problem

DAP

- given
  - network graph
  - link capacities C_e $e = 1,2, \dots, E$
  - demand volumes h_d $d = 1,2, \dots, D$
  - lists of paths (routes) $\mathcal{P}_{dp}$ $p = 1,2, \dots, P_d$

- find allocation of flows x that minimizes the max load function:

$$F(x) = max_e \{ l(e,x) - C_e \}$$

minimize $F(x)$

demand allocation problem - formulation

DAP

Book: Section 4.2.4 ($L_d = 1$)

- given parameters
  - link capacity $C_e \quad e = 1,2,\dots,E$
  - demand volumes $h_d \quad d = 1,2,\dots,D$
  - lists of paths (routes) $\mathcal{P}_{dp} \quad d = 1,2,\dots,D, \quad p = 1,2,\dots,P_d$
  - link/path structure $\delta_{edp} \quad e = 1,2,\dots,E, \quad d = 1,2,\dots,D, \quad p = 1,2,\dots,P_d$

- variables x_{dp} path flows

- objective minimize z (where z is a real number)
- subject to $\sum_{p=1}^{P_d} x_{dp} = h_d \quad d = 1,2,\dots,D$

  $\sum_{d=1}^{D} \sum_{p=1}^{P_d} \delta_{edp} x_{dp} \leq C_e + z \quad e = 1,2,\dots,E$

  $x_{dp} \in Z_+ \quad d = 1,2,\dots,D, \quad p = 1,2,\dots,P_d$

Z_+ – set on non-negative integers

routing of demands and resulting link capacities

DDAP

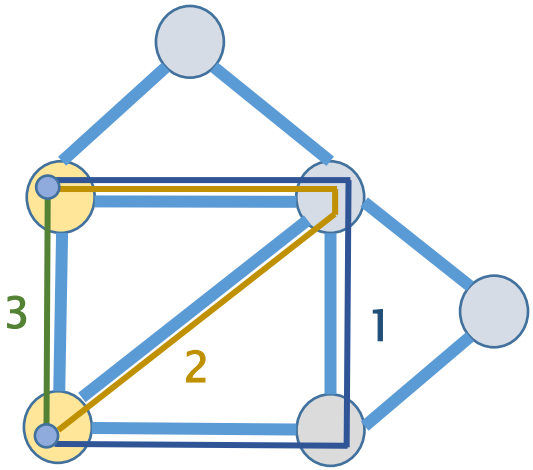
flow x defines flow allocation (demands' routing/allocation)

$x(d,p)$ number of demand units realized on path $p \epsilon P(d)$

$x(d,1) + x(d,2) + \cdots + x(d,P_d) = h_d$

$\sum_p x(d,p) = h_d$

$x(d,p)$ – nonnegative integers



demands $d = 1,2,\ldots,D$

paths $p = 1,2,\ldots,P_d$

$d=1$	$d=2$		$d=D$
0	3		1
3	2		3
4	1	...	0
0			1
1			
$h_1 = 8$	$h_2 = 6$		$h_D = 5$

- for a given feasible flow x
- find link load $l(e,x)$ $(e = 1.2,\ldots,E)$:
  - total number of demand units using link e
  - $l(e,x) = \sum_d \sum_p \delta_{edp} x(d,p)$
- then compute link size $y(e)$, i.e., the number of modules required to carry $l(e,x)$:
  - $y(e,x) = \left\lceil \frac{l(e,x)}{M} \right\rceil$

link loads calculation

DDAP

- input data: M, paths, path-flows $x(d,p)$:s
- find link load $l(e,x)$ $(e = 1,2, \dots E)$:
  - total number of demand units using link e
  - $l(e,x) = \sum_d \sum_p \delta_{edp} x(d,p)$
- then compute link size $y(e,x)$, i.e., the number of modules required to carry $l(e,x)$:
  - $y(e,x) = \left\lceil \frac{l(e,x)}{M} \right\rceil$

begin

for $e := 1$ **to** E **do** $l(e,x) := 0$;

for $d := 1$ **to** D **do**

for $p := 1$ **to** P_d **do**

for $e := 1$ **to** E **do if** $e \in \mathcal{P}_{dp}$ **then** $l(e,x) := l(e,x) + x(d,p)$;

for $e := 1$ **to** E **do** $y(e,x) := \left\lceil \frac{l(e,x)}{M} \right\rceil$

end

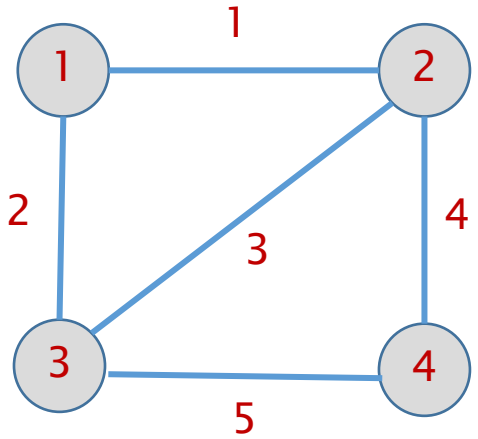
exercise 1

input data

- $V = 4$ nodes $(v = 1,2,3,4)$
- $D = 6$ demands $(d = 1,2, \dots, 6)$
- $E = 5$ links $(e = 1,2, \dots, 5)$
- d: (1,2), (1,3), (1,4), (2,3), (2,4), (3,4)
- h_d: 3, 4, 5, 2, 3, 4 [Mbps]
- 2 or 3 routes for each demand
- $M = 2$ [Mbps] : link capacity module

For the flow allocation given in the next slide determine the corresponding link loads and the number of capacity modules required to realize those loads on each link.

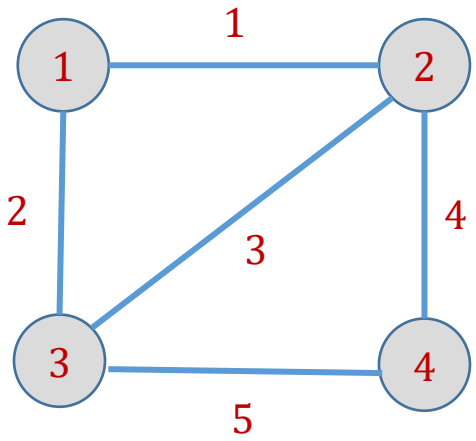
(node and link numbers)



(we write $P(d,p)$ instead of $\mathcal{P}_{dp}$
paths (routes):
$P(1,1) = \{1\}, P(1,2) = \{2,3\}, P(1,3) = \{2,4,5\}$
$P(2,1) = \{2\}, P(2,2) = \{1,3\}, P(2,3) = \{1,4,5\}$
$P(3,1) = \{1,4\}, P(3,2) = \{2,5\}$
$P(4,1) = \{3\}, P(4,2) = \{1,2\}, P(3,3) = \{4,5\}$
$P(5,1) = \{4\}, P(5,2) = \{3,5\}, P(5,3) = \{1,2,5\}$
$P(6,1) = \{5\}, P(6,2) = \{3,4\}, P(6,3) = \{1,2,4\}$

exercise 1 (continued)

$x(d,p), \quad d = 1,2,\dots,D, p = 1,2,\dots,P_d$



d: (1,2), (1,3), (1,4), (2,3), (2,4), (3,4)

flow pattern:

d	1	2	3	4	5	6
$x(d,1)$	2	0	3	1	2	2
$x(d,2)$	0	2	2	1	0	0
$x(d,3)$	1	2		0	1	2
h_d	3	4	5	2	3	4

dimensioning and demand allocation problem cntd.

DDAP

- given
  - network graph
  - link module M
  - cost of installing one module ξ_e $e = 1,2,\dots,E$
  - demand volumes h_d $d = 1,2,\dots,D$
  - lists of paths (routes) $\mathcal{P}_{dp}$ $p = 1,2\dots,P_d$

- find allocation of flows x that minimizes the cost of links:

$$F(x) = \sum_e \xi_e \cdot y(e,x)$$

minimize $F(x)$

dimensioning and demand allocation problem –formulation

DDAP

Book: Section 4.3.1

- given parameters
  - link module — M
  - link unit cost — $\xi_e \quad e = 1,2,\dots,E$
  - demand volumes — $h_d \quad d = 1,2,\dots,D$
  - lists of paths (routes) — $\mathcal{P}_{dp} \quad d = 1,2,\dots,D, \quad p = 1,2,\dots,P_d$

- variables
  - x_{dp} path flows
  - y_e link capacities

- objective — minimize $F(y) = \Sigma_e \, \xi_e y_e$
- subject to

$$\sum_p x_{dp} = h_d \qquad d = 1,2,\dots,D$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \le M y_e \qquad e = 1,2,\dots,E$$

$$x_{dp} \in Z_+ \qquad d = 1,2,\dots,D, \; p = 1,2,\dots,P_d$$

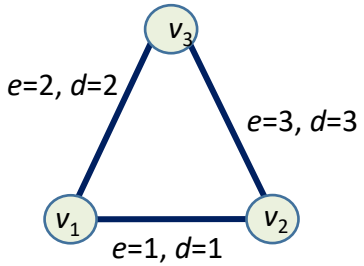
$$y_e \in Z_+ \qquad e = 1,2,\dots,E$$

Z_+ – set on non-negative integers

exercise 2: DDAP

$$E = 3, D = 3$$
$$M = 2, \xi_e \equiv 1, h_d \equiv 1$$

$$\mathcal{P}_{11} = \{1\}, \mathcal{P}_{12} = \{2,3\}$$
$$\mathcal{P}_{21} = \{2\}, \mathcal{P}_{22} = \{1,3\}$$
$$\mathcal{P}_{31} = \{3\}, \mathcal{P}_{32} = \{1,2\}$$



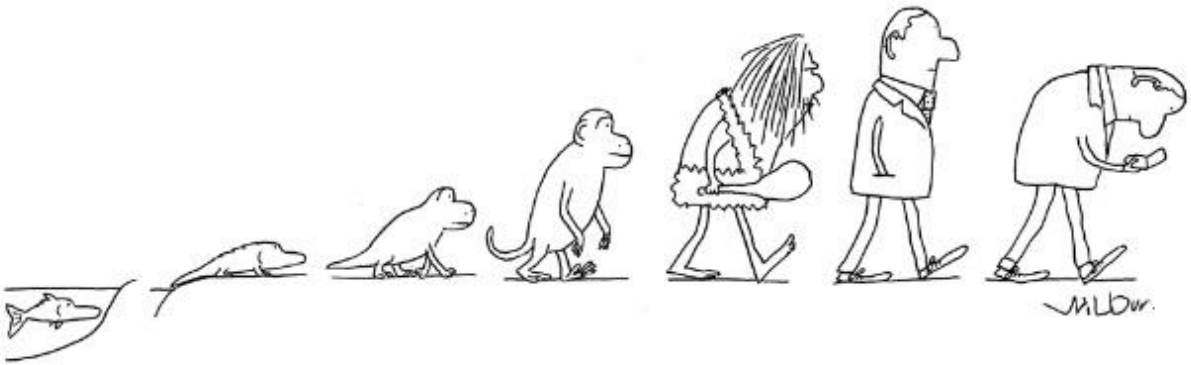
1. Write down the general DDAP formulation.
2. Write down the instance of DDAP for the above network.
3. What are the optimal solutions of this instance?

evolutionary algorithm (metaheuristic)

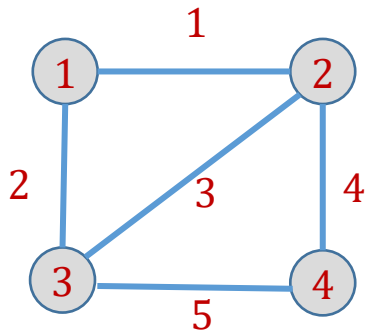
EA

- population = a set of N chromosomes (individuals, candidate solutions)
- generation = a consecutive population
- chromosome = a sequence of genes
  - individual, solution (point of the solution space)
  - genes represent internal structure of a solution
- fitness function = objective function

Book: Section 5.3.3



EA for DDAP



d	1	2	3	4	5	6
$x(d,1)$	0	2	2	1	1	2
$x(d,2)$	3	0	3	0	2	2
$x(d,3)$	0	2	-	1	0	0
h_d	3	4	5	2	3	4

chromosome (encodes a complete feasible solution)
$x = (x(1), x(2), \dots, x(D))$

gene (encodes an allocation pattern for demand d)
$x(d) = (x(d,1), x(d,2), \dots, x(d,P_d))$

link loads calculation

- input data: M, paths, $x(d,p)$:s
- find link load $l(e,x)$ $(e = 1,2,\dots,E)$:
  - total number of demand units using link e
  - $l(e,x) = \sum_d \sum_p \delta_{edp} x(d,p)$
- then compute link size $y(e,x)$, i.e., the number of modules required to carry $l(e,x)$:
  - $y(e,x) = \left\lceil \frac{l(e,x)}{M} \right\rceil$

```
begin
    for e := 1 to E do l(e,x) := 0;
    for d := 1 to D do
      for p := 1 to P_d do
        for e := 1 to E do if e ∈ P_dp then l(e,x) := l(e,x) + x(d,p);
    for e:=1 to E do y(e,x) := ⌈l(e,x)/M⌉
end
```

genetic operators

- mutation
  - is performed on a chromosome with a certain (low) probability
  - it perturbs the values of the chromosome genes, also with a certain low probability

- crossover
  - exchanges genes between two parent chromosomes to produce two offsprings
  - in effect both offspring have genes from both parents
  - chromosomes with better fitness function have greater chance to become parents

in general, the operators are problem-dependent

genetic operators for DDAP

- crossover of two chromosomes
  - each gene of the two offspring is taken from one of the parents x', x''
    - for each $d = 1,2 \dots, D$: $x(d) \coloneqq x'(d)$ with probability 0.5
      $x(d) \coloneqq x''(d)$ with probability 0.5
    - and the complement of x
  - better fit chromosomes have a greater chance to become parents
- mutation of a chromosome
  - for a mutated gene shift one unit of demand from one path to another
  - everything at random

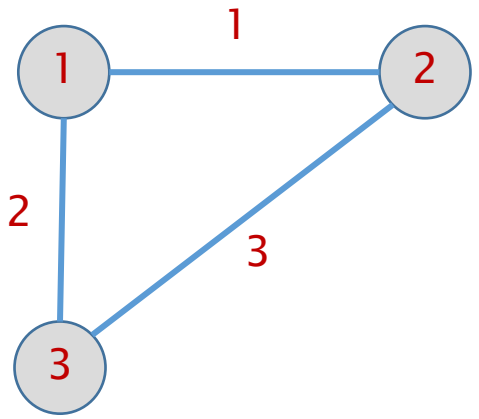
(N+K) evolutionary algorithm

```
begin
    n:= 0; initialize(P(0));
    while stopping criterion not true
          O(n):= ∅;
          for i:= 1 to K do O(n):= O(n) ∪ crossover[P(n)];
          for x ∈ O(n) do mutate(x);
          n:= n+1,
          P(n):= select_best_N[O(n−1)∪P(n−1)];
     end while
end
```

(N+K) EA - example

input data (N = 2, K=1)

- $V = 3$ nodes $(v = 1,2,3)$
- $D = 3$ demands $(d = 1,2,3)$
- $E = 3$ links $(e = 1,2,3)$
- d: (1,2), (1,3), (2,3)
- h_d: 2, 2, 2 [Mbps]
- 2 routes for each demand
- $M = 4$ [Mbps] : link capacity module
- unit link costs: $\xi_1 = \xi_2 = 1, \xi_3 = 2$



d	1	2	3
$x'(d,1)$	2	2	2
$x'(d,2)$	0	0	0
h_d	2	2	2

d	1	2	3
$x''(d,1)$	0	0	0
$x''(d,2)$	2	2	2
h_d	2	2	2

crossover:

2	0	0
0	2	2

0	2	2
2	0	0

mutation of x'':

0	1	0
2	1	2

some issues

- selection of parents
  - depending on the value of fitness function (queen of the bees)
  - purely random

- initialization
  - random population, random population without worst N chromosomes, population of individually optimized (simulated annealing, steepest descent) chromosomes, good chromosomes (other optimization algorithms, expert knowledge, etc.)

- convergence criteria, i.e., when to stop ?
  - after a predefined number of generations, good enough solution, no progress for a certain number of generations (individually the best chromosome, population's average, standard deviation)

- elitism and uniformity
  - best chromosomes should retain between generations
  - do not create duplicates (low fraction of duplications is advantageous)

exact optimization

- it's not at all easy to solve DDAP and DAP to optimality
- heuristics – no guarantee of optimum
- brute force – not effective
- mathematical formulations of DDAP and DAP and integer programming problems – proper approach

algorithms for DDAP and DAP

EA heuristic (evolutionary algorithm)

try to find a reasonable solution in reasonable time

BFA exact state space enumeration method (brute force)

generate all solutions x and identify the one that minimizes $F(x)$

excessive computation time but exact solution

BBA exact (B&B: branch and bound)

basic method of integer programming

BFA: number of flow patterns x

$$\prod_{d=1}^{D} \binom{h_d + P_d - 1}{P_d - 1}$$

where $\binom{n}{k} = \frac{n!}{n!(n-k)!}$

project

- a common parser for input data
- 3 parts + report(EA+BF) + final report(IP+comparisons)
  - EA own implementation (parameter tunning)
  - BFA own implementation
  - B&B (IP) AMPL+CPLEX
- we will start with DDAP and DAP but then different teams will implement slightly different problems

BFA – subproblem pseudocode (one demand)

```
begin
    F := +∞;
    for x(1) := 0 to h do
      for x(2) := 0 to h − x(1) do
        for x(3) := 0 to h − x(1) − x(2) do
          .....
            for x(P) := 0 to h − x(1) − x(2) − ⋯ − x(n − 1) do
              if F(x) < F then
                begin F := F(x); x^best := x end
end
```

BFA – pseudocode

```
rec(1,1,h(1),x);
function rec(curd,curp,lefth,x)
begin
    if curp = P(curd) then
        x(curd,curp)=lefth;
        if curd < D then rec(curd+1,1,h(curd+1),x)
        else print(x)
        endif;
    else
        for parth=0 to lefth
            x(curd,curp)=parth;
            rec(curd,curp+1,lefth-parth,x)
        endfor
    endif
end
```

- h(d) – demand's volume
- P(d) – number of demand's paths
- curd – current demand id, 1..D
- curp – current path id, 1..P(d)
- lefth – remaining part of current demand's volume
- x – solution, two dimensional array of path-flows

how to solve DDAP (and DAP) exactly?

- general method for integer programming problems: branch-and-bound (B&B)

- the B&B algorithm makes use of linear relaxation (LR) of the problem in hand
  - LR provides a lower bound for the integer (true) version of DDAP
  - in addition, LR gives a suboptimal solution (rounding up fractional y_e :s) which is an upper bound for the exact integer solution

- LR is a linear programming problem
  - general method for linear programming problems: simplex algorithm