Contents college 5 and 6

Branch and Bound; Beam Search
(Chapter 3.4-3.5, book)

$\rightarrow$ general introduction

Job Shop Scheduling
(Chapter 5.1-5.3, book)

- branch and bound (5.2)
- shifting bottleneck heuristic (5.3)

JOB SHOP

- multi-operational model
- several machines (workcenters)
- individual routes for the jobs
- with/without recirculation

JOB SHOP - cont.

Applications:

- wafer production
- patients in a hospital

Remark:

Job shop is a special case of the resource constraint project scheduling problem

---

We concentrate on

- machines (no workcenters)
- no recirculation
- makespan minimization

JOB SHOP - Definition

- n jobs to be processed on
- m machines
- operation (i, j): processing of job j on machine i
- order of the operations of a job is given:
  $(i, j) \rightarrow (k, j)$ specifies that job j has to be processed on machine i earlier than on machine k
- p_{ij}: duration of operation (i, j)

- Goal: Schedule the jobs on the machines
  – without overlapping on machines
  – without overlapping within a job
  – minimizing the makespan (latest completion of a job)

JOB SHOP - Example

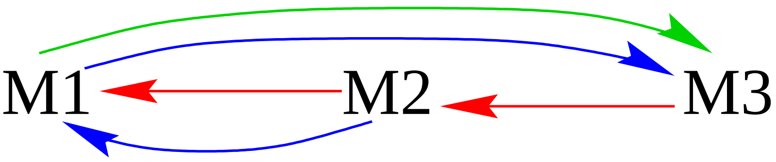
Data:

Machines: M1, M2, M3

Jobs: J1 (3,1)-->(2,1)-->(1,1)

J2 (1,2)-->(3,2)

J3 (2,3)-->(1,3)-->(3,3)



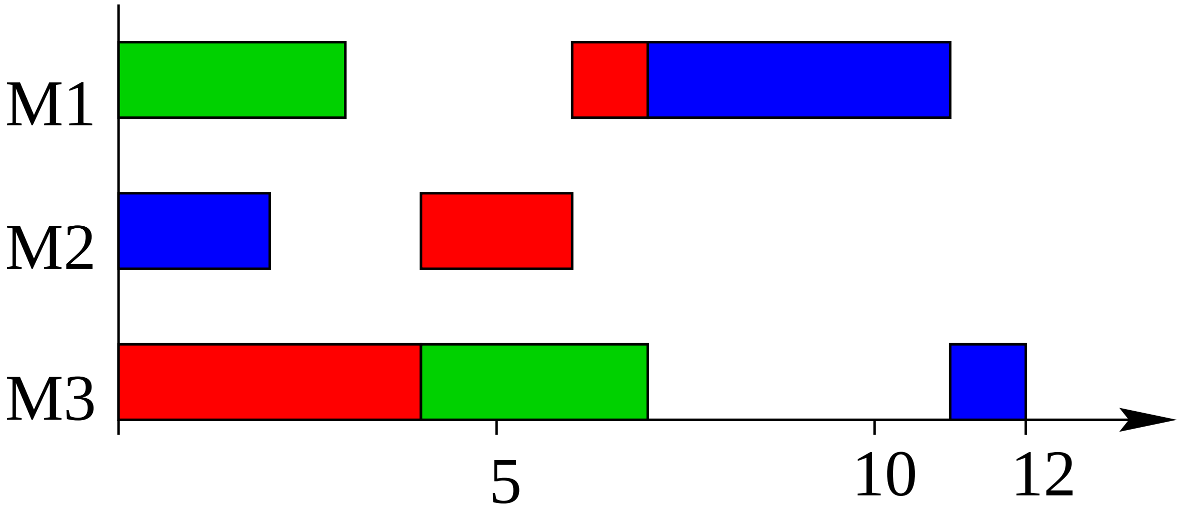
Durations: p31=4, p21=2, p11=1

p12=3, p32=3

p23=2, p13=4, p33=1

JOB SHOP - Example (cont.)

Schedule:



Makespan Cmax=12

Disjunctive Graph Representation

Graph $G = (N, A, B)$

- nodes correspond to operations
  $N = \{(i,j) | (i,j)$ is operation$\}$

- <u>conjunctive</u> arcs A
  represent order of the operations of jobs

$$(i,j) \to (k,j) \in A$$
$$\iff$$

operation (i,j) preceeds (k,j)

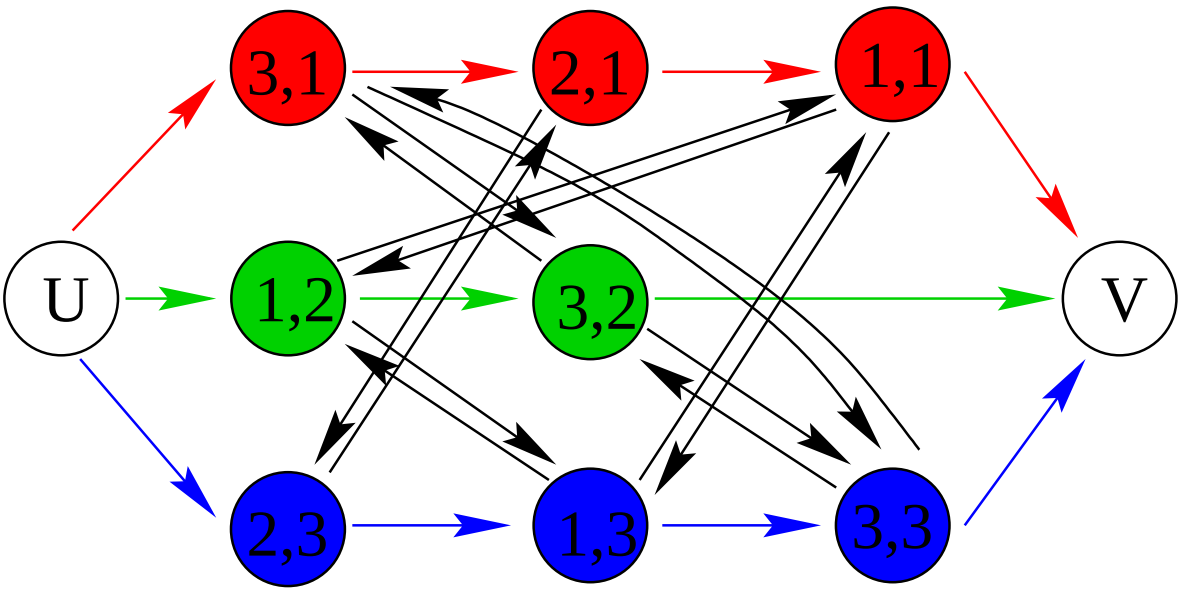
- <u>disjunctive</u> arcs B
  represent conflicts on machines
  Two operations (i,j) and (i,l) are connected by two arcs going in opposite direction

Disjunctive Graph Rep. (cont.)

- Two dummy nodes U and V representing source and sink
- arcs from U to all first operations of jobs
- arcs from all last operations of jobs to V

Remark: The disjunctive arcs form m cliques of double arcs (one for each machine)

Disjunctive Graph - Example



Conjunctive arcs

Disjunctive arcs (double arcs)

Selection

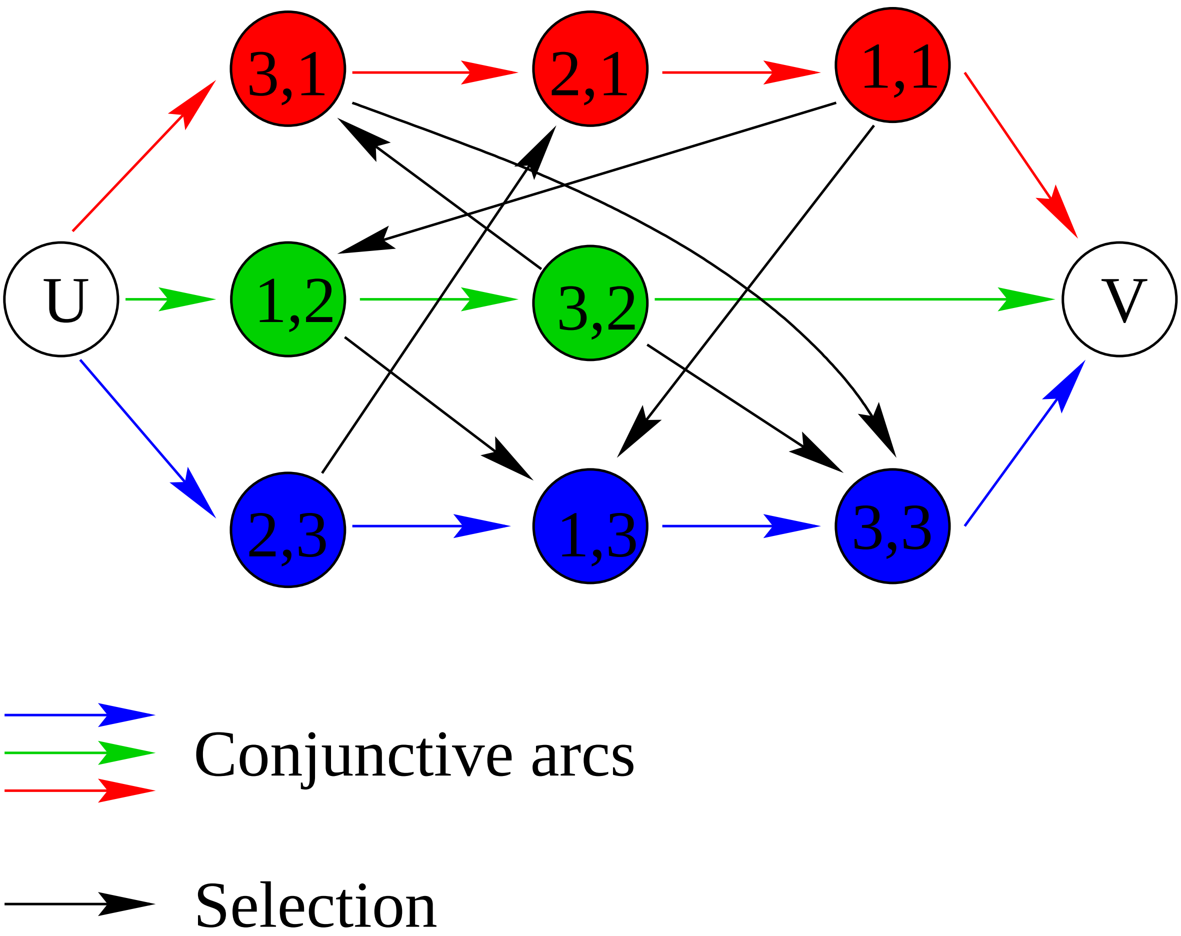
A subset $D \subset B$ is called a selection if it contains from each pair of disjunctive arcs exactly one

A selection D is feasible if the resulting directed graph $G(D) = (N, A \cup D)$ (graph with conjunctive and selected disjunctive arcs) is acyclic

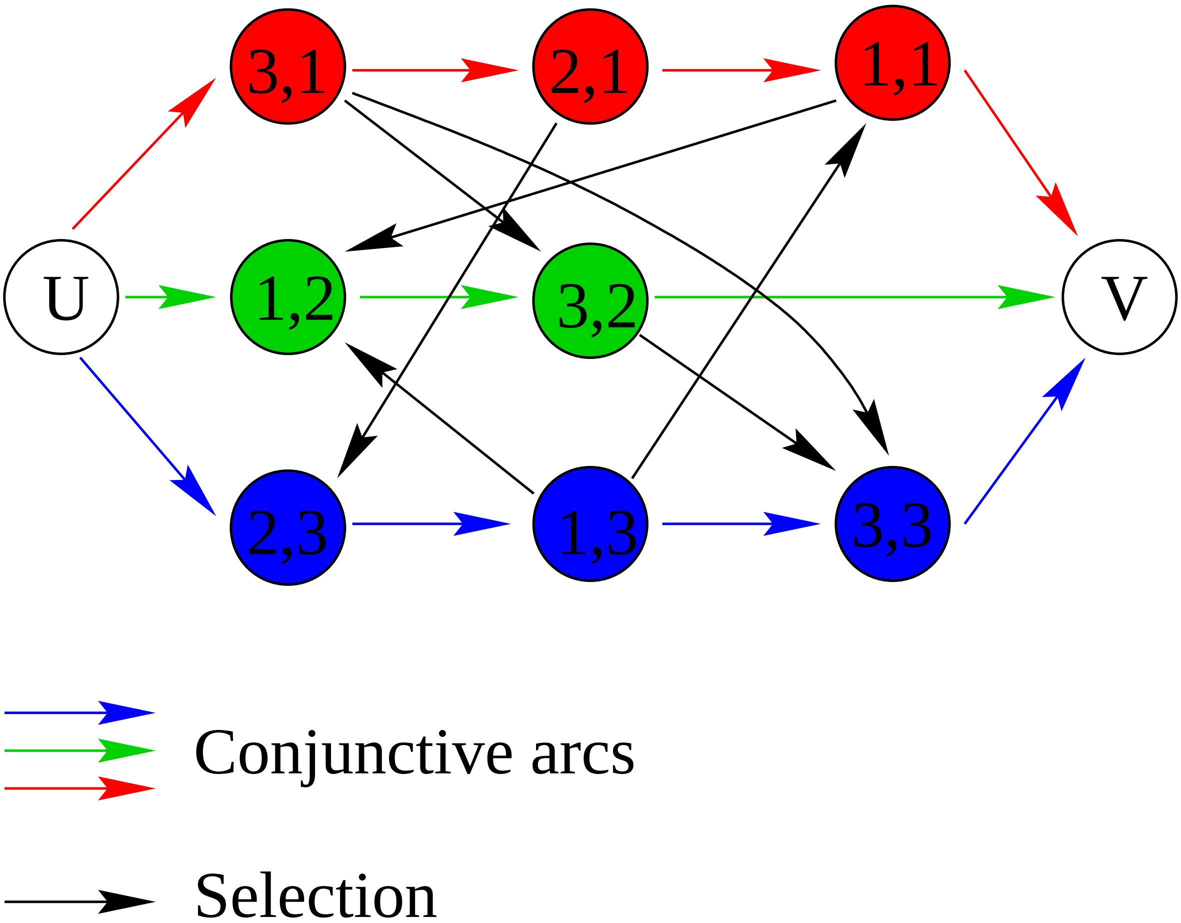
Remarks:

1. a feasible selection leads to a sequence in which operations have to be processed on machines
2. a feasible solution induces to a feasible selection
3. each feasible selection leads to a feasible schedule

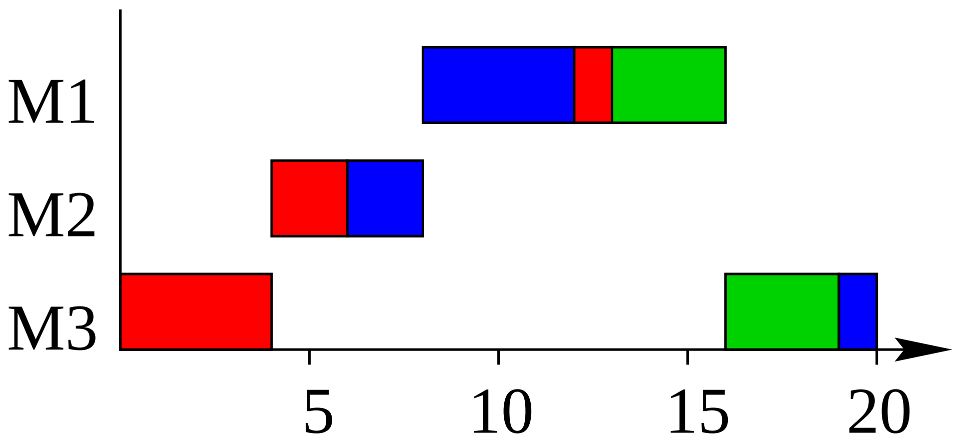
Infeasible selection - Example



Feasible selection - Example

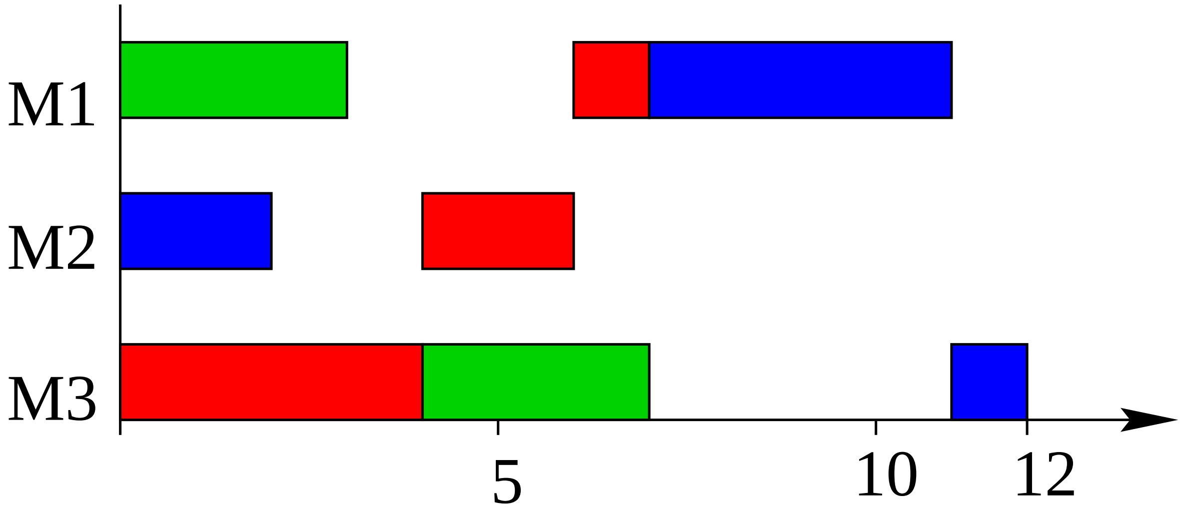


Corresponding Schedule



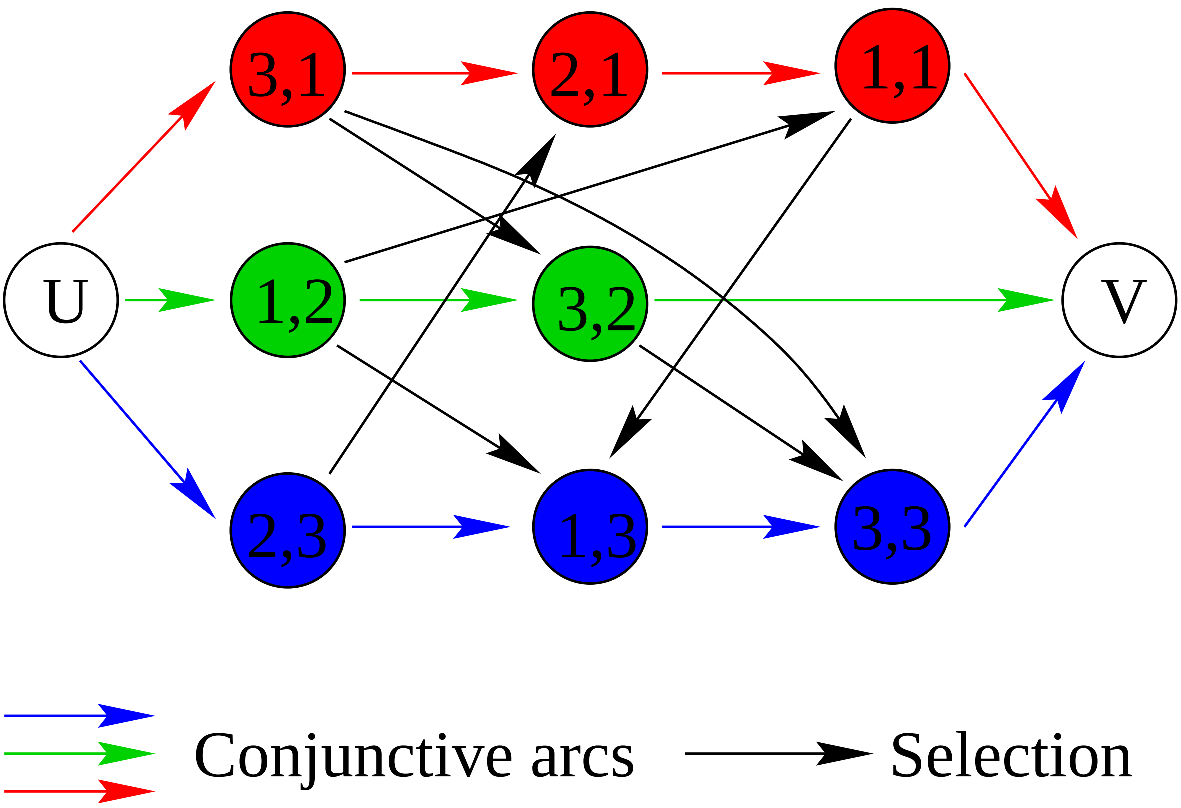
Makespan Cmax=20

Selection for given Schedule



Makespan Cmax=12

Corresponding selection



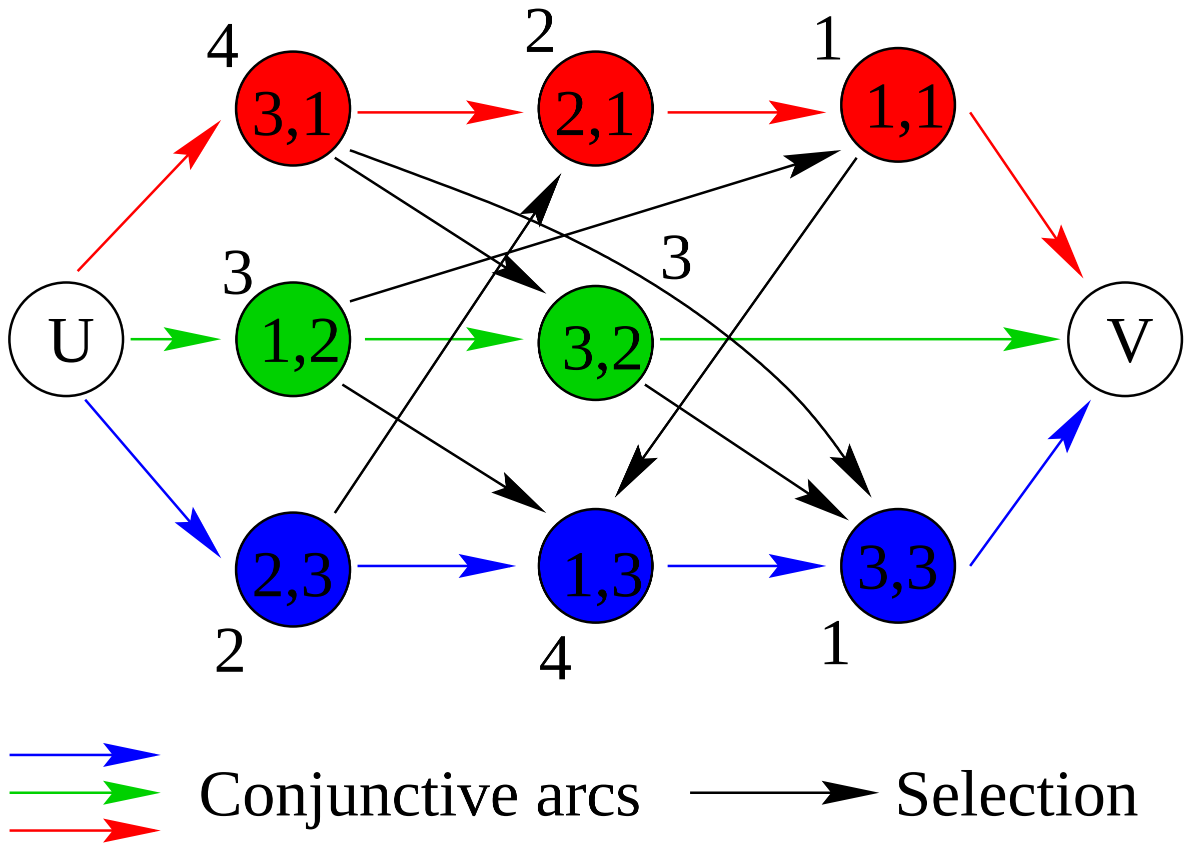
Calculate Schedule for Selection

Method: Calculated longest paths from U to all other nodes in $G(D)$

Technical description:

- weight nodes (i, j) by p_{ij} and node U by 0
- length of a path $i_1, i_2, \ldots, i_r$ = sum of the weights of the nodes $i_1, i_2, \ldots, i_{r-1}$
- calculate length l_{ij} of the longest path from U to (i, j) and V (using e.g. Dijkstra)
- start operation (i, j) at time l_{ij}
- the length of a longest path from U to V (such paths are called critical paths) is equal to the make-span of the solution

Calculate Schedule for Selection - Example



Calculation l_{ij}'s:

Node	(3,1)	(1,2)	(3,2)	(2,3)	(2,1)
Length	0	0	4	0	4

Node	(1,1)	(1,3)	(3,3)	V
Length	6	7	11	12

Disjunctive Programming Formulation

y_{ij} denotes starting time of operation (i,j)

minimize C_{max}

subject to

$$y_{kl} - y_{ij} \geq p_{ij} \quad \text{for all } (i,j) \rightarrow (k,j) \in A$$

$$C_{max} - y_{ij} \geq p_{ij} \quad \text{for all } (i,j) \in N$$

$$y_{ij} - y_{il} \geq p_{il} \quad \text{or} \quad y_{il} - y_{ij} \geq p_{ij} \quad \text{for all } (i,l),(i,j);\ i = 1,\ldots,m$$

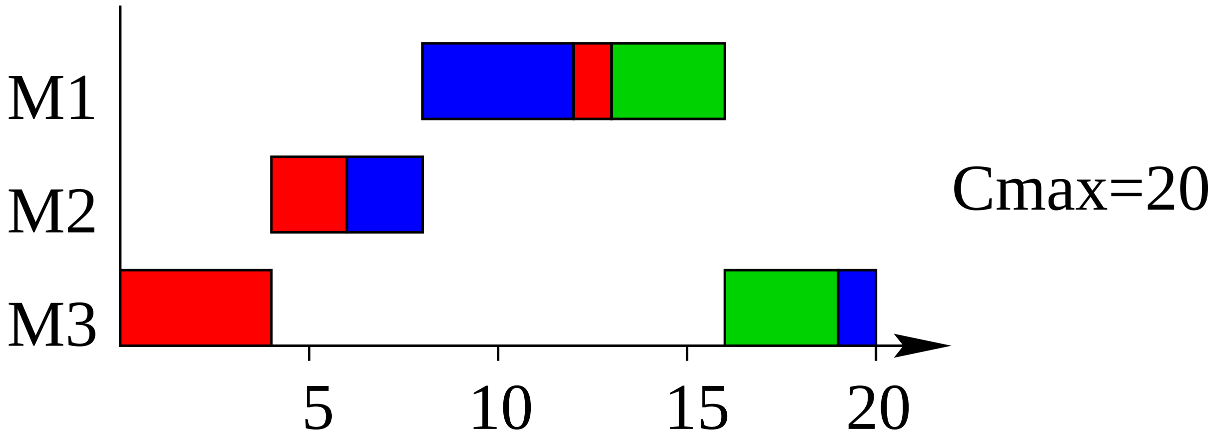
$$y_{ij} \geq 0 \quad \text{for all } (i,j) \in N$$

Active Schedules

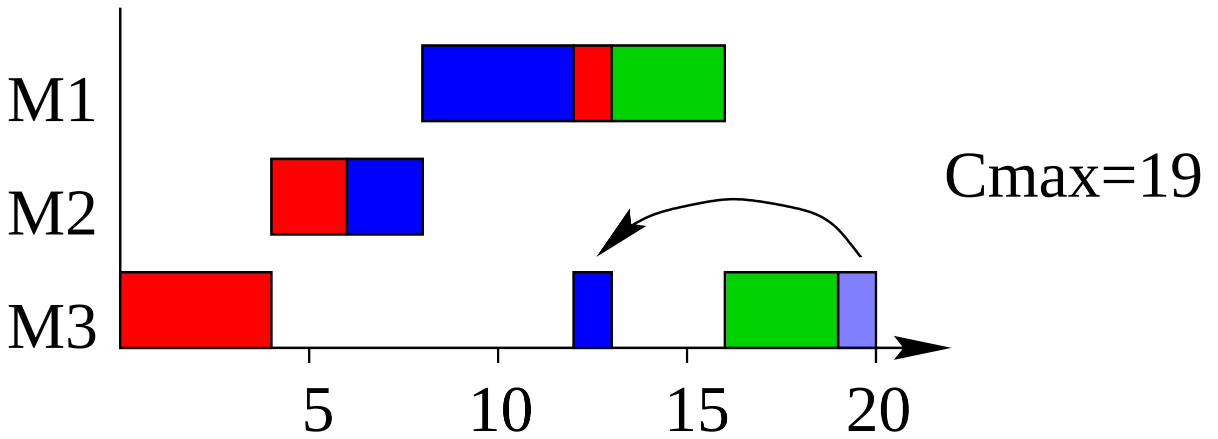
A schedule is <u>active</u> if no operation can be scheduled earlier without delaying another operation

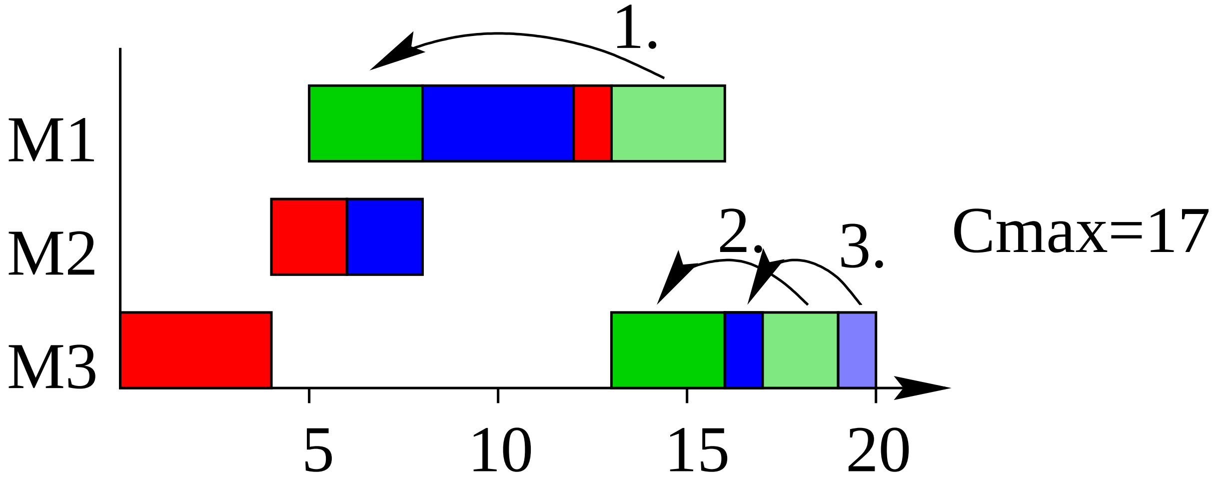
- reducing the makespan of an active schedule is only possible by increasing the starting time of at least one operation
- there exist an optimal schedule which is active

Example of a non-active schedule



Possible modifications





Base of Branch and Bound

Remark: The set of all active schedules contains an optimal schedule

Solution method: Generate all active schedules and take the best

Improvement: Use the generation scheme in a branch and bound setting

Consequence: We need a generation scheme to produce all active schedules for a job shop

Generation of all active schedules

Notations:

- Ω: set of all operations which predecessors have already been scheduled
- r_{ij}: earliest possible starting time of operation $(i, j) \in \Omega$
- Ω': subset of Ω

Remark: r_{ij} can be calculated via longest path calculations

Generation of all active schedules - cont.

Step 1 (Initial Conditions)
$\Omega :=$ {first operations for each job}
$r_{ij} := 0$ for all $(i, j) \in \Omega$

Step 2 (Machine selection)
Compute for current partial schedule

$$t(\Omega) := \min_{(i,j)\in\Omega} \{r_{ij} + p_{ij}\}$$

$i^* :=$ machine on which minimum is achieved

Step 3 (Branching)
$\Omega' := \{(i^*, j) | r_{i^*j} < t(\Omega)\}$
FOR ALL $(i^*, j) \in \Omega'$ DO
extend partial schedule by scheduling (i^*, j) next on machine i^*;
delete (i^*, j) from Ω;
add job-successor of (i^*, j) to Ω;
Return to Step 2

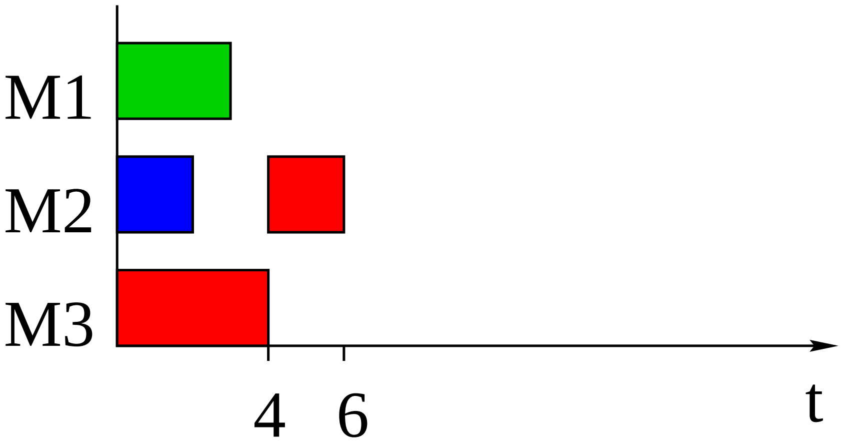
Generation ...- example

Jobs: J1 (3,1)-->(2,1)-->(1,1)

J2 (1,2)-->(3,2)

J3 (2,3)-->(1,3)-->(3,3)

Part. Schedule

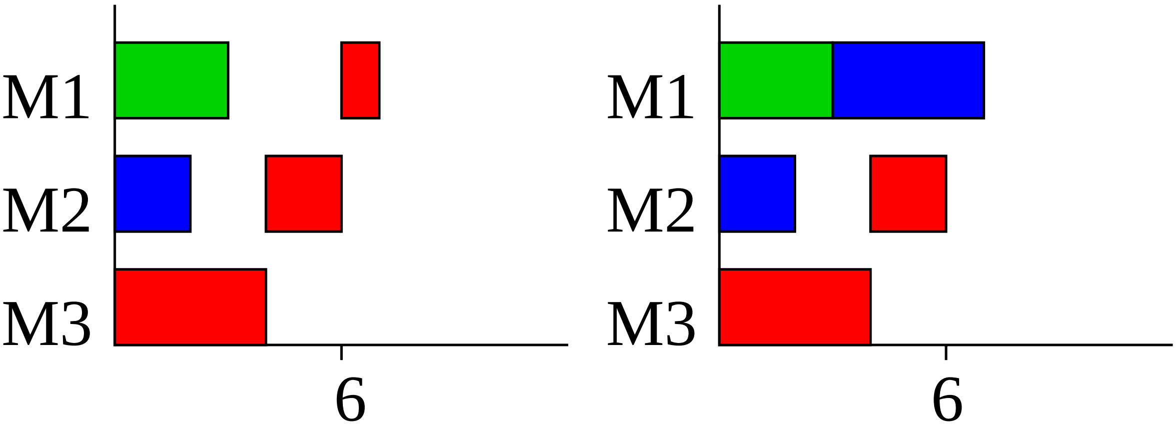


$\Omega = \{(1,1),(3,2),(1,3)\}, r_{11} = 6, r_{32} = 4, r_{13} = 3$

$t(\Omega) = \min\{6+1, 4+3, 3+4\} = 7$; $i^* = M1$

$\Omega' = \{(1,1),(1,3)\}$

<u>Extended partial schedules</u>:



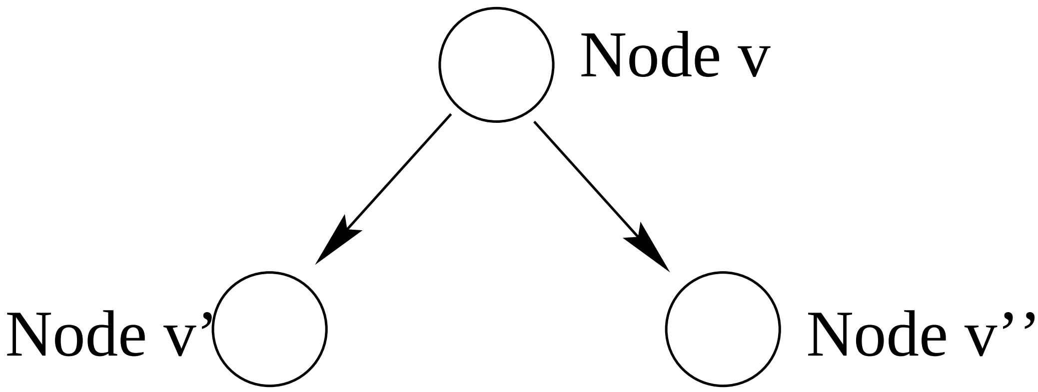
Generation of all active schedules - cont.

Remarks on the algorithm:

- the given algorithms is the base of the branching
- nodes of the branching tree correspond to partial schedules
- Step 3 branches from the node corresponding to the current partial schedule
- the number of branches is given by the cardinality of Ω'
- a branch corresponds to the choice of an operation (i^*, j) to be schedules next on machine i^*
  $\rightarrow$ a branch fixes new disjunctions

Disjunctions fixed by a branching

$$\Omega' = \{(i^*, j), (i^*, l)\}$$



selection (i^*, j)

Add disjunctions $(i^*, j) \rightarrow (i^* k)$ for all unscheduled operations (i^*, k)

selection (i^*, l)

Add disjunctions $(i^*, l) \rightarrow (i^* k)$ for all unscheduled operations (i^*, k)

Consequence: Each node in the branch and bound tree is characterized by a set D' of fixed disjunctions

Lower bounds for nodes of the branch and bound tree

Consider node V with fixed disjunctions D':

Simple lower bound: Calculate critical path in $G(D')$
$\rightarrow$ Lower bound $LB(V)$

Better lower bound:

- consider machine i
- allow parallel processing on all machines $\neq i$
- solve problem on machine i

Resulting 1-machine problem

1. calculate earliest starting times r_{ij} of all operations (i,j) on machine i (longest paths from source in $G(D')$)
2. calculate minimum amount Δ_{ij} of time between start of (i,j) and end of schedule (longest path to sink in $G(D')$)
   $\rightarrow$ due date $d_{ij} = LB(V) - \Delta_{ij} + p_{ij}$
3. solve single machine problem on machine i:
   - respect release dates
   - no preemption
   - minimize maximum lateness

   (see Section 3.4)

Result: maximum lateness L_i

Better lower bound

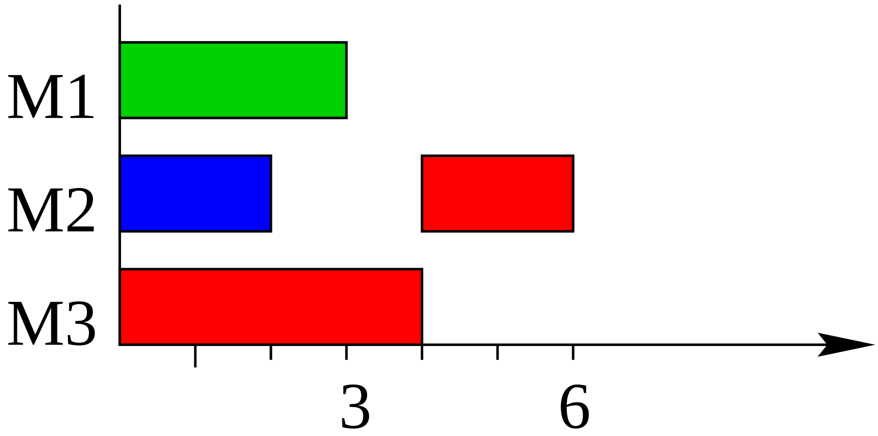
- solve 1-machine problem for all machines
- this results in values $L_1, \ldots, L_m$
- $$LB^{new}(V) = LB(V) + \max_{i=1}^{m} L_i$$

Remarks:

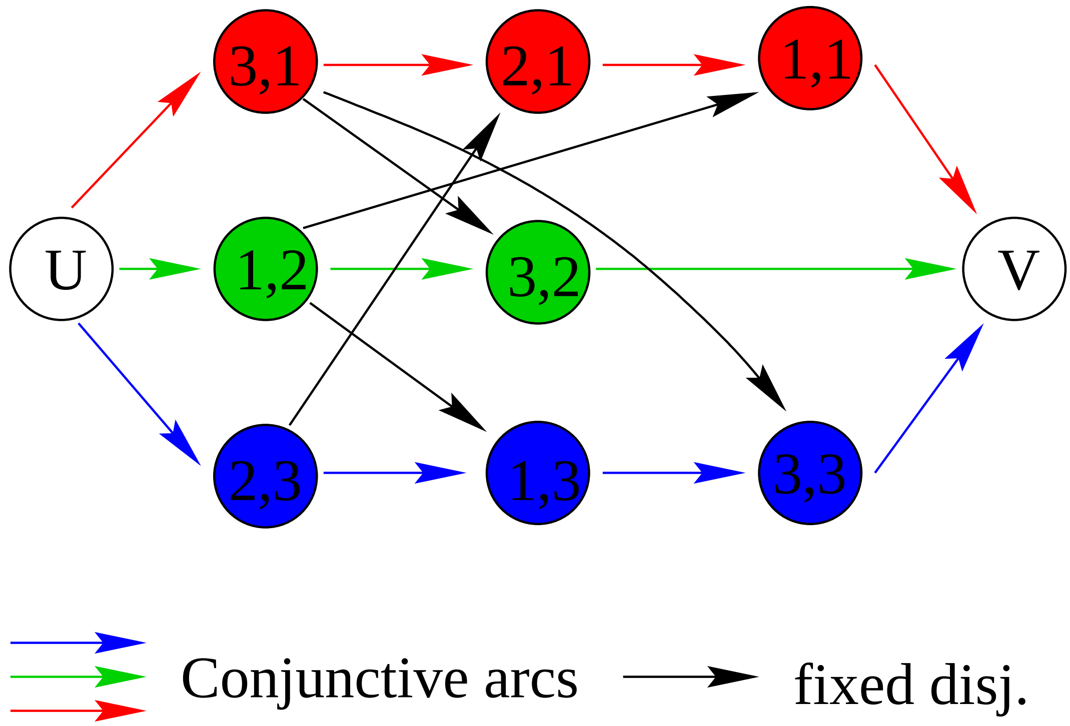
- 1-machine problem is NP-hard
- computational experiments have shown that it pays of to solve these m NP-hard problems per node of the search tree
- 20×20 instances are already hard to solve by branch and bound

Better lower bound - example

Partial Schedule:

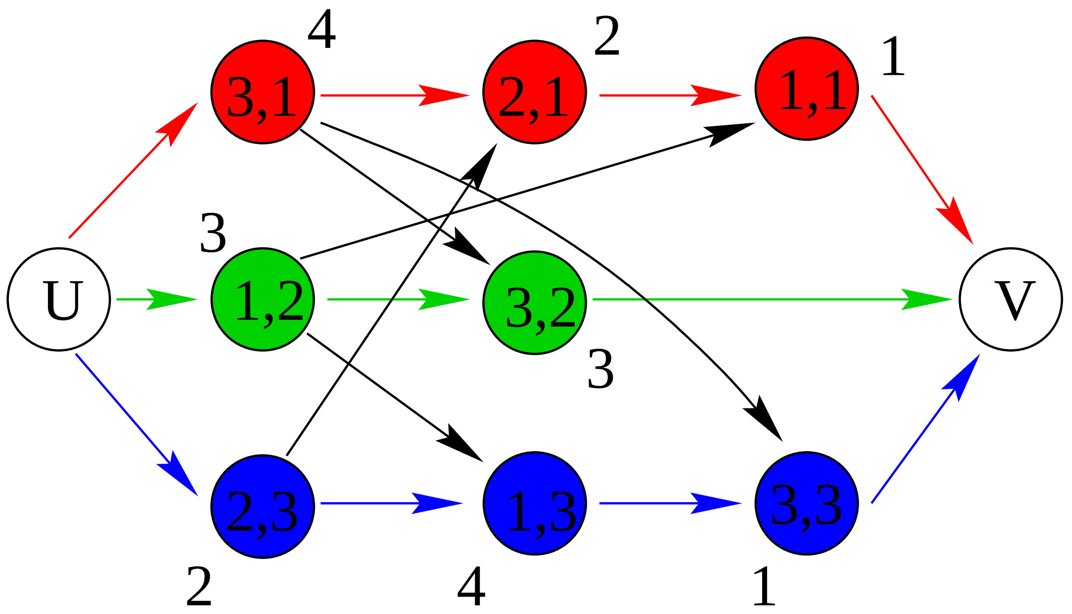


Corresponding graph $G(D')$:



Better lb - example - cont.

Graph $G(D')$ with processing times:

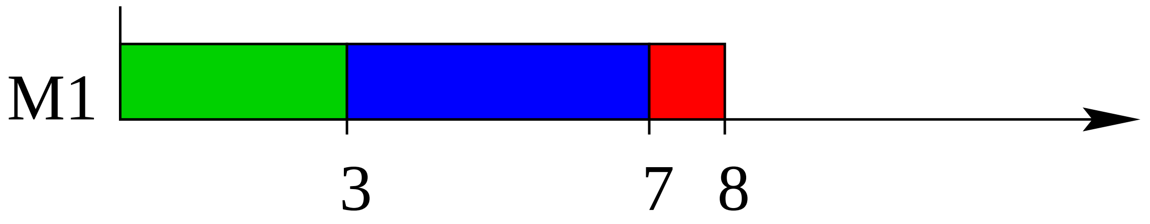


$LB(V)$=$l(U,(1,2),(1,3),(3,3),V)$=8 (unique)

Data for jobs on Machine 1:

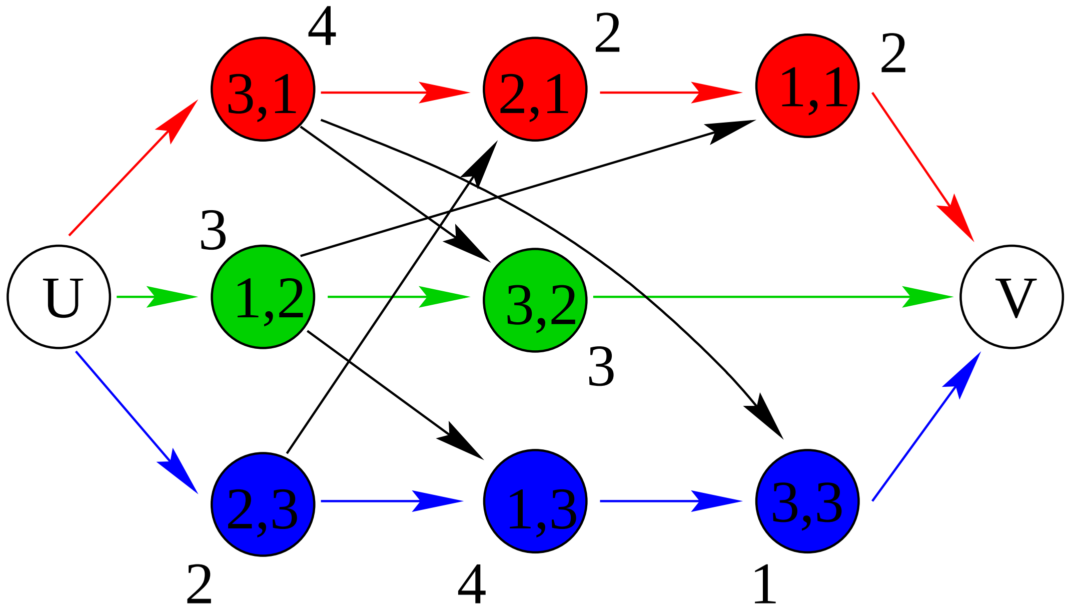
green	blue	red
$r_{12}=0$	$r_{13}=3$	$r_{11}=6$
$\Delta_{12}=8$	$\Delta_{13}=5$	$\Delta_{11}=1$
$d_{12}=3$	$d_{13}=7$	$d_{11}=8$

Opt. solution: $L_{max}=0$, $LB^{new}(V)=8$



Better lb - example - cont.

Change p_{11} from 1 to 2!

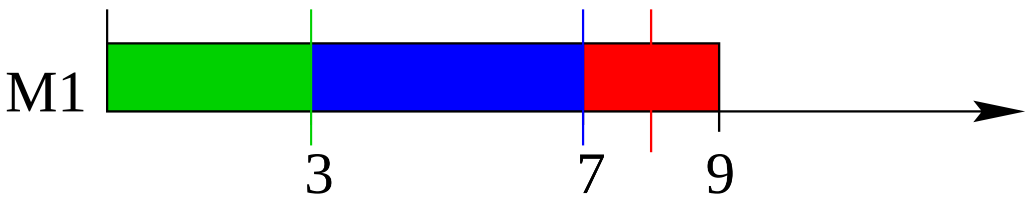


$$LB(V) = l(U,(1,2),(1,3),(3,3),V)$$
$$= l(U,(3,1),(2,1),(1,1),V) = 8$$

Data for jobs on Machine 1:

green	blue	red
$r_{12} = 0$	$r_{13} = 3$	$r_{11} = 6$
$\Delta_{12} = 8$	$\Delta_{13} = 5$	$\Delta_{11} = 2$
$d_{12} = 3$	$d_{13} = 7$	$d_{11} = 8$

Opt. solution: $L_{max} = 1$, $LB^{new}(V) = 9$



Opgaven voor werkcollege

Date: Time:	Thursday, 31 May, 2001 13.45-15.30 (5+6)
Room:	WB H-IV-206
Exercises:	5.1, 5.3, 5.6 (a)