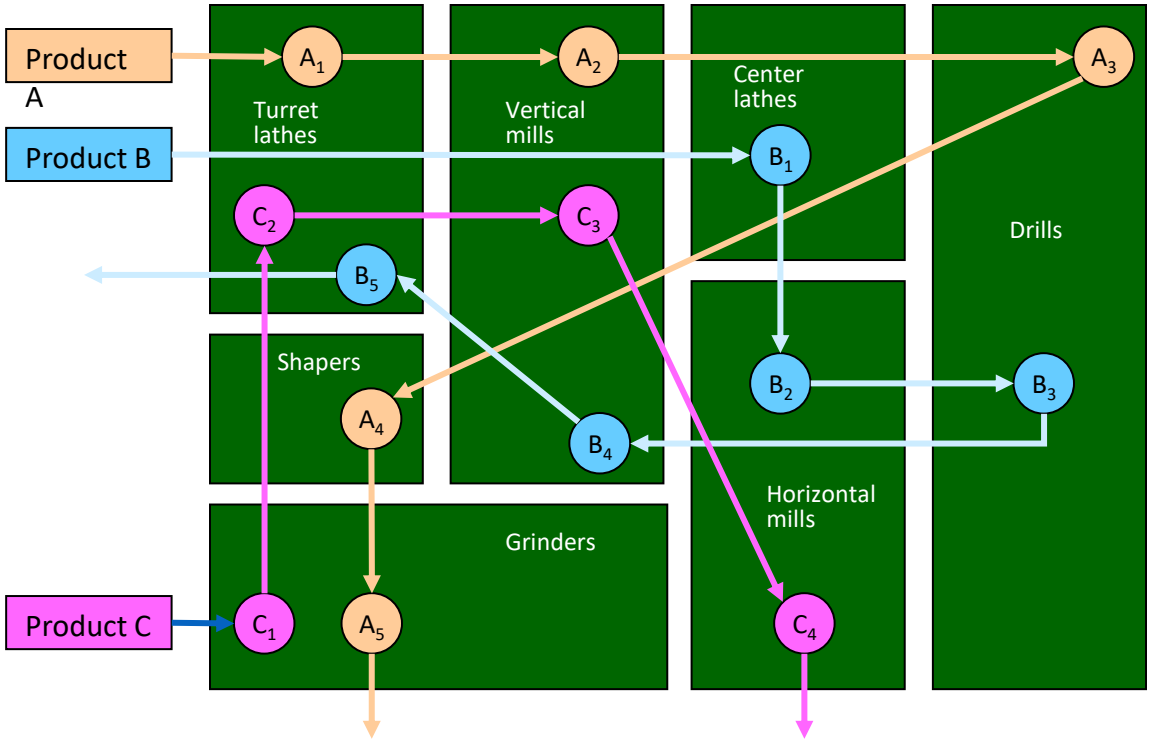
PRO5802 - Programação de Produção Intermitente (2021)

Aula 8 – Teoria de JobShop

Prof. Daniel de Oliveira Mota

Job Shop



From "Fundamentals of Operations Management" by Davis, Aquilano, Chase (1999)

Description of Job Shop Scheduling

- A finite set of n jobs
- Each job consists of a chain of operations
- A finite set of m machines
- Each machine can handle at most one operation at a time
- Each operation needs to be processed during an uninterrupted period of a given length on a given machine
- Purpose is to find a schedule, that is, an allocation of the operations to time intervals to machines, that has minimal length

Formal Definition of JSS

- Job set $J = \{ j_1, j_2, \ldots j_n \}$
- Machine set $M = \{m_1, m_2, \ldots m_m\}$
- Operations $O = \{o_1, o_2, \ldots o_n\}$ $O_i = \{o_{i1}, o_{i2}, \ldots o_{im_i}\}$
- Each operation has processing time $\{\tau_{i1}, \tau_{i2}, \ldots \tau_{im_i}\}$
- On O define A, a binary relation represent a precedence between operations. If $(v, w) \in A$ then v has to be performed before w.
- A induce the total ordering belonging to the same job; no precedence exist between operations of different jobs.

Formal Definition of JSS cont.

- A schedule is a function $S : O \to IN \cup \{0\}$ that for each operation v defines a start time S(v).
- A schedule S is feasible if

$$\forall v \in O: \qquad S(v) \geq 0$$

$$\forall v, w \in O,\ (v, w) \in A: \qquad S(v) + \tau(v) \leq S(w)$$

$$\forall v, w \in O,\ v \neq w,\ M(v) = M(w): \ S(v) + \tau(v) \leq S(w)\ or$$

$$S(w) + \tau(w) \leq S(v)$$

- The length of a schedule S is $len(S) = \max_{v \in O}(S(v) + \tau(v))$
- The goal is to find an optimal schedule, a feasible schedule of minimum length., min(len(S)).

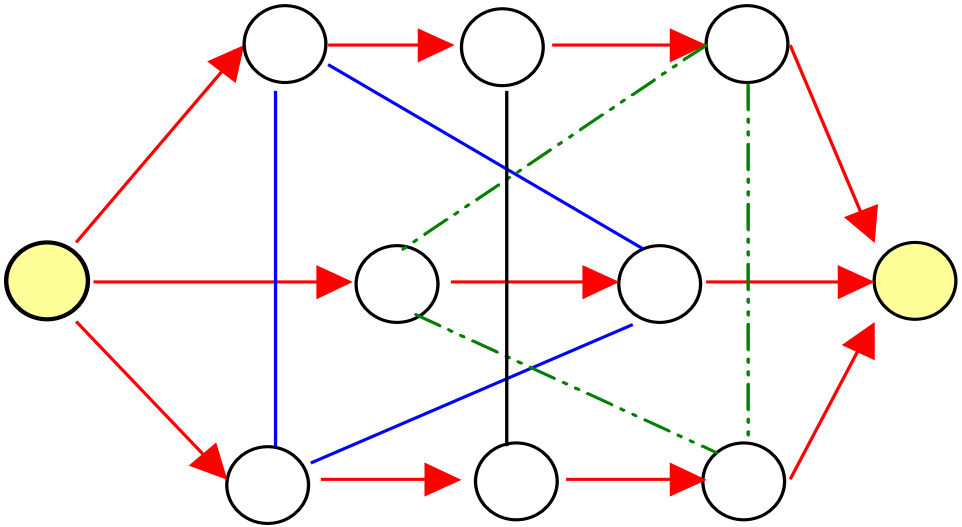
Disjunctive Graph

- An instance of the JSS problem can be represented by means of a **disjunctive graph** G=(O, A, E).
- The vertices in O represent the operations
- The arcs in A represent the given precedence between the operations
- The edge in

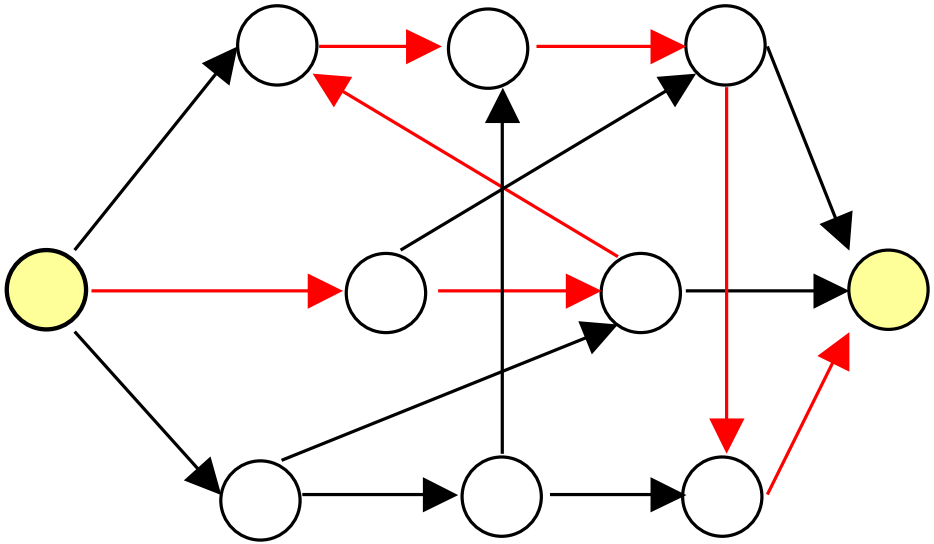
  represent the machine capacity constraints $E=\{(v,w) \mid v,w \in O,\ v \neq w,\ M(v)=M(w)\}$
- Each vertex v has a weight, equal to the processing time

$$\tau(v)$$

Example of Disjunctive Graph



Disjunctive Graph with Edge Orientations



Disjunctive Graph Cont.

- Finding an optimal feasible schedule is equivalent to finding an orientation E’ that minimizes the longest path length in the related digraph.

Why JSS Problem

- It is considered to be a good representation of the general domain and has earned a reputation for being notoriously difficult to solve
- JSS is considered to belong to the class of decision problems which are NP
- Lenstra et al(1977) Show that
  - 3*3 problem
  - N*2 instance with no more than 3 operations per job
  - N*3 problem with no more than 2 operations per job
  - N*3 problem where all operations are of unit processing time

  Belong to the set of NP instances.

Methods to Solve JSS

- **Mathematical Formulations:**
  - mixed integer linear programming (1960)
- **Branch and Bound**
- **Approximation Methods**
  - Priority dispatch rules
  - Bottleneck based heuristics
  - Artificial intelligence(constraint satisfaction approach, neural networks)
  - Local search methods

Branch and Bound

- Using a dynamically constructed tree structure represents the solution space of all feasible sequences
- Search begins at topmost node and a complete selection is achieved once the lowest level node has been evaluated
- Each node at a level p in the search tree represent a partial sequence of p operations
- From an unselected node the branching operation determines the next set of possible nodes from which the search could progress
- The bounding procedure selects the operation which will continue the search and is based on an estimated LB and currently best achieved UB. IF at any node the estimated LB is found to be greater than the current best UB, this partial selection and all its subsequent descendants are disregarded.

Priority Dispatch Rules

- At each successive step all the operations which are available to be scheduled are assigned a priority and the operation with the highest priority is chosen to be sequenced.
- Usually several runs of PDRs are made in order to achieve valid results.

Constraint Satisfaction Approach

- Aiming at reducing the effective size of the search space by applying constraints that restrict the order in which variables are selected and the sequence in which possible values are assigned to each variable
- Constraint propagation
- Backtracking
- Variable heuristic
- Value heuristic

Neural Networks

- Hopfield networks
- Back-error propagation networks

Local Search Method

- Configurations: a finite set of solutions.
- Cost function to be optimised.
- Generation mechanism, generating a transition from one configuration to another.
- Neighborhood, N(x), is a function which defines a simple transition from a solution x to another solution by inducing a change.
- Selection of neighborhood:
  - chose the first lower cost neighbor found;
  - select the best neighbor in the entire neighborhood;
  - Choose the best of a sample of neighbors.

Summary

- The definition of JSS
- The disjunctive graph
- The methods to solve JSS

Job Shop: Two Machines

Consider now problem $J2|\ |C_{max}$.

The set of jobs $N=\{1,\ldots,n\}$ can be split into two subsets:
N_{AB} - the jobs that consist of two operations with the processing route (A, B)
N_{BA} - the jobs that consist of two operations with the processing route (B, A)

The algorithm presented below is due to R. Jackson (1956).

Jackson's Algorithm

1. Run Johnson's algorithm with the set of jobs N_{AB} and find the corresponding sequence R_{AB}.
2. Run Johnson's algorithm with the set of jobs N_{BA} and find the corresponding sequence R_{BA}.
3. On machine A, first schedule N_{AB} according to R_{AB} and then N_{BA} according to R_{BA}.
4. On machine B, first schedule N_{BA} according to R_{BA} and then N_{AB} according to R_{AB}.

Job Shop: Two Machines

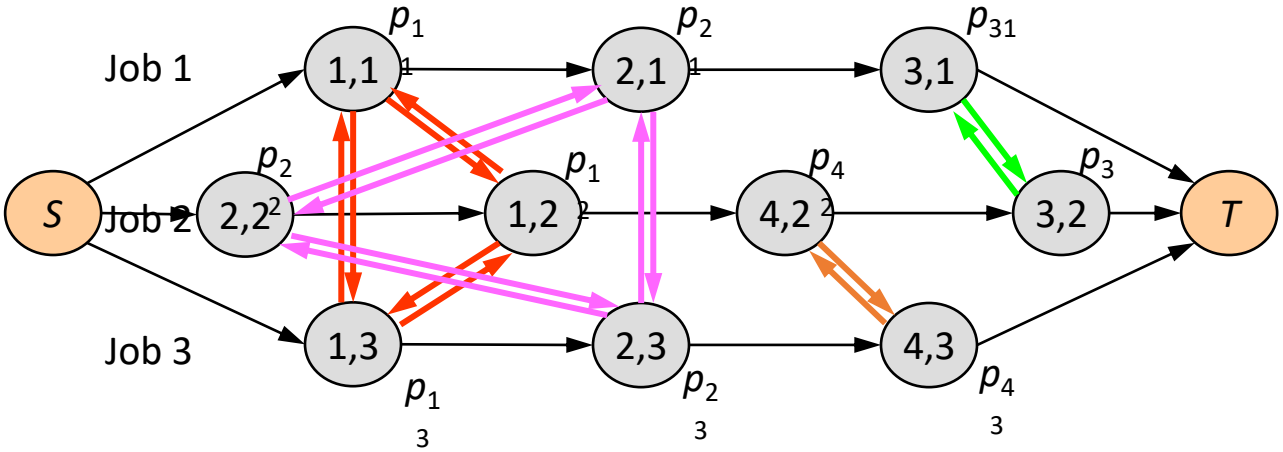
Example

Consider Problem $J2|\ |C_{max}$

j	1	2	3	4	5	6	7	8	9	10	11
a_j	3	2	1	1	2	4	3	1	2		
b_j	2	1	2	1	4	8	9			2	1

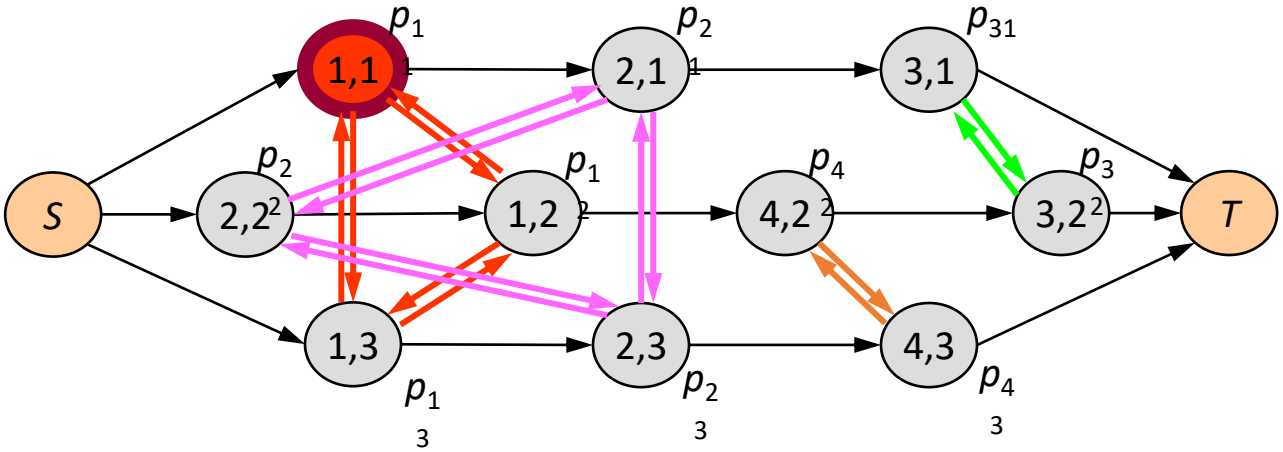
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



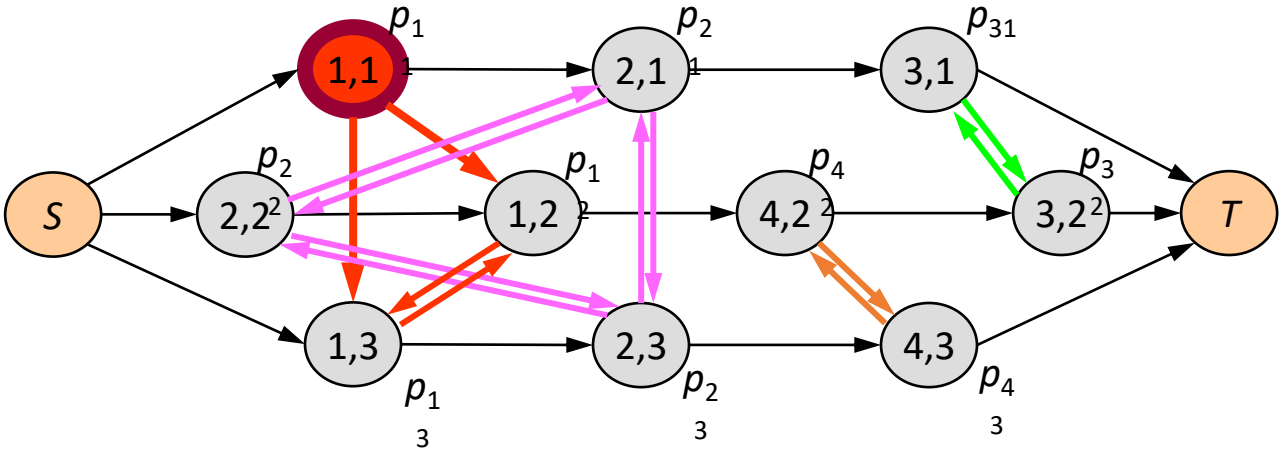
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



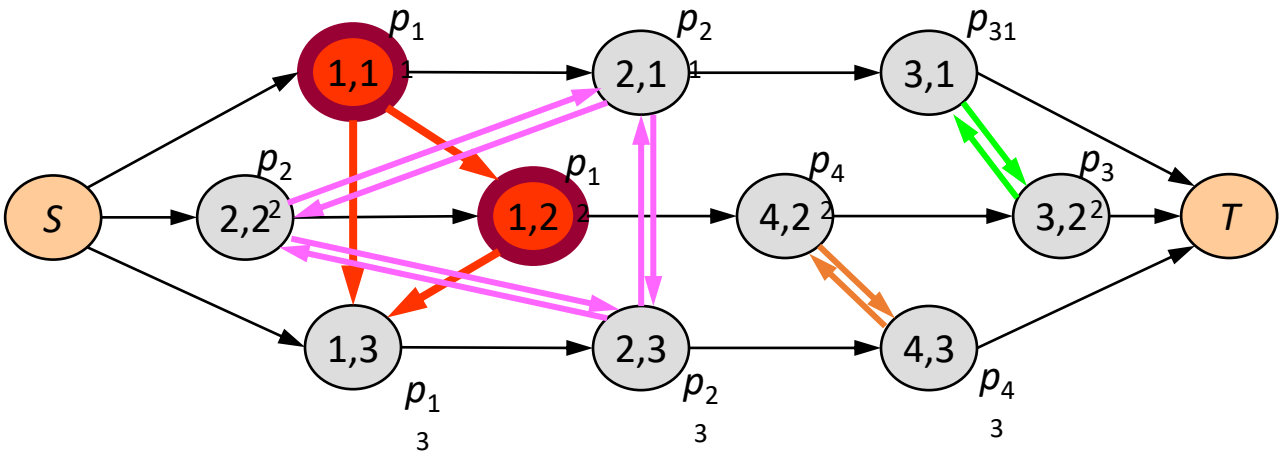
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



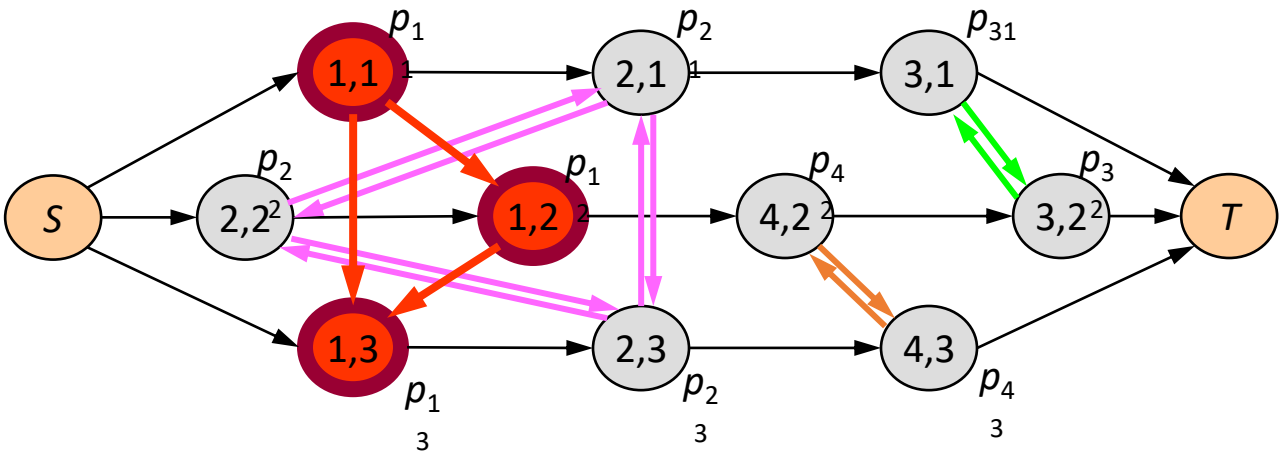
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



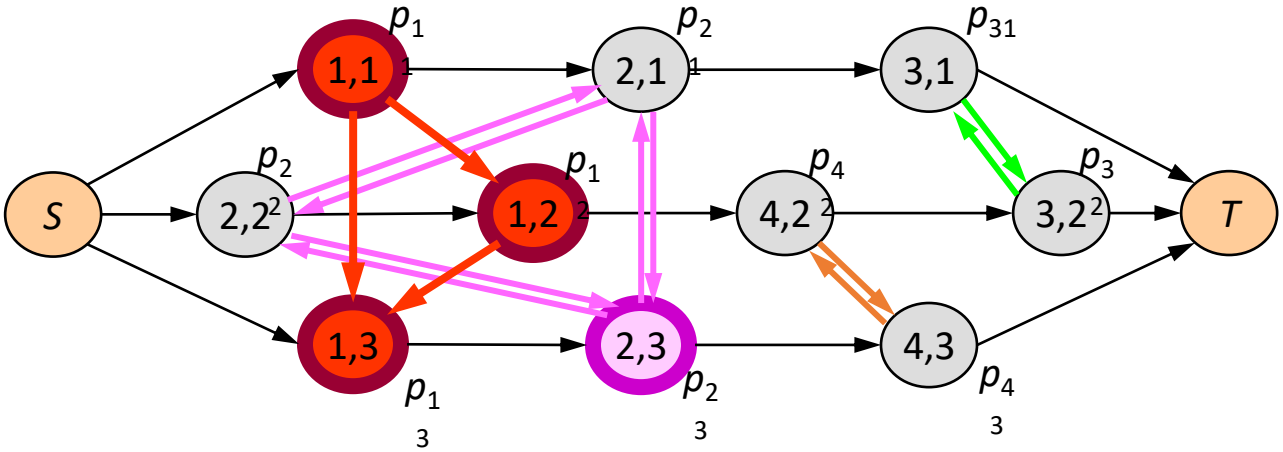
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



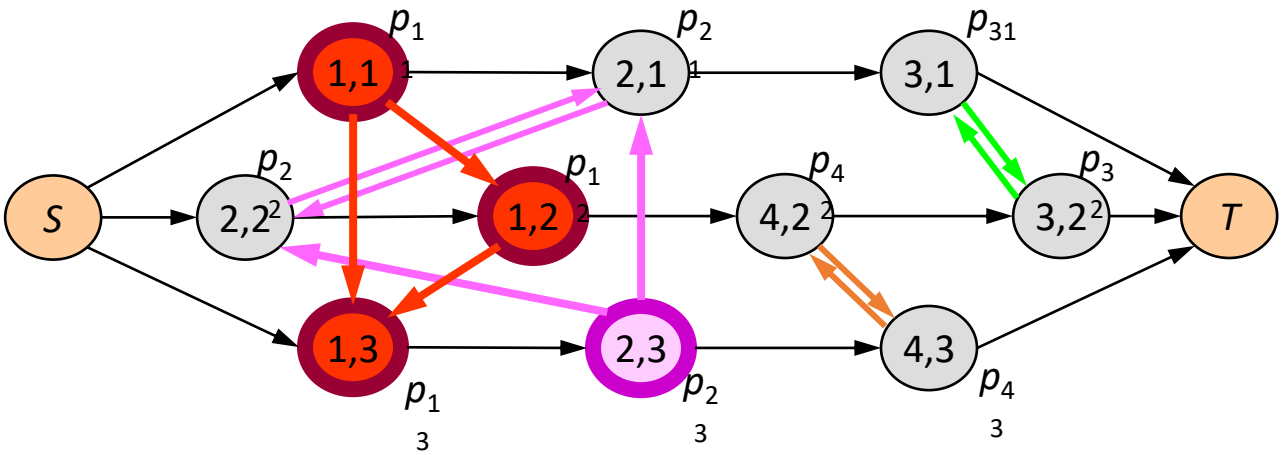
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



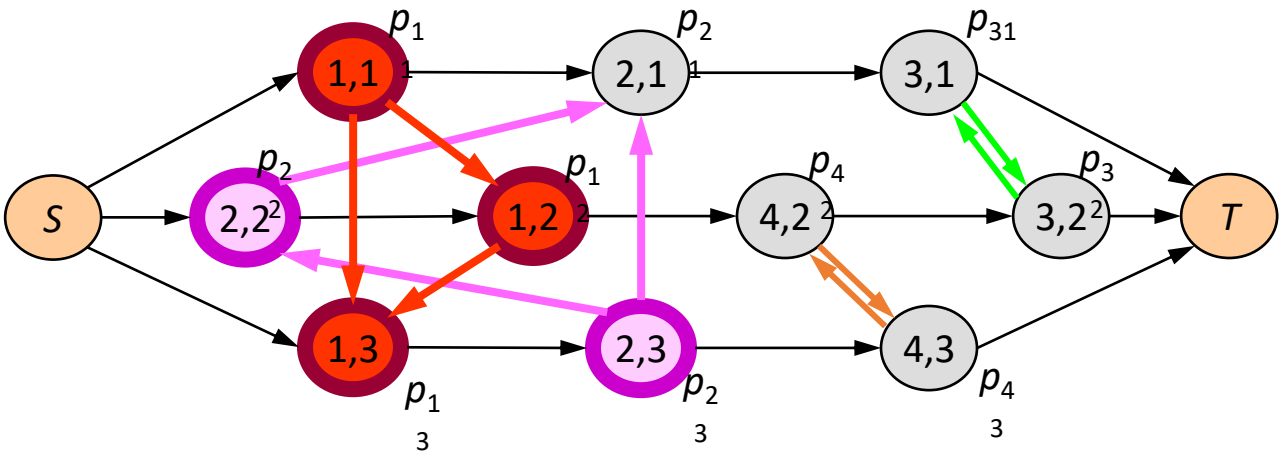
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



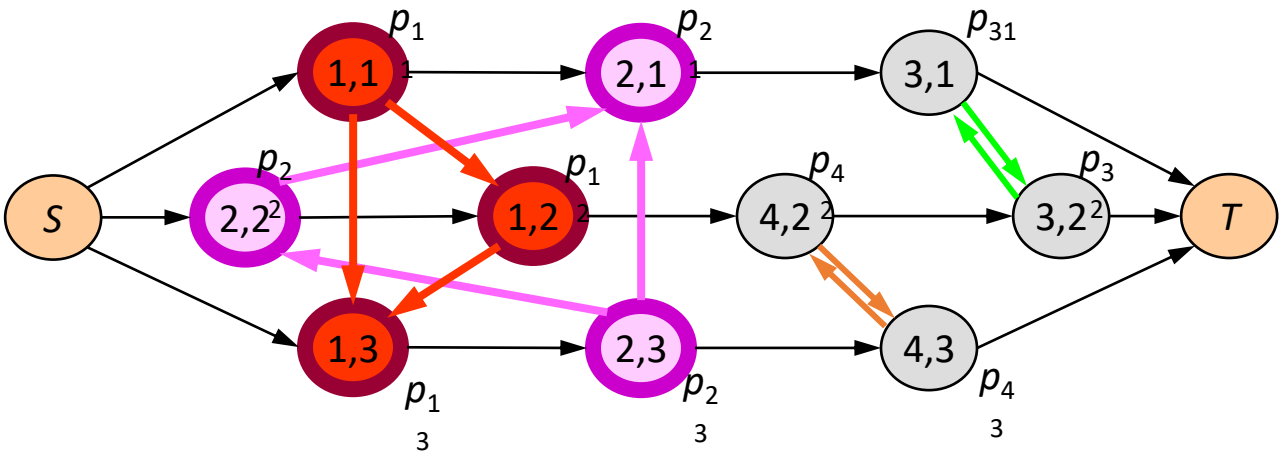
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



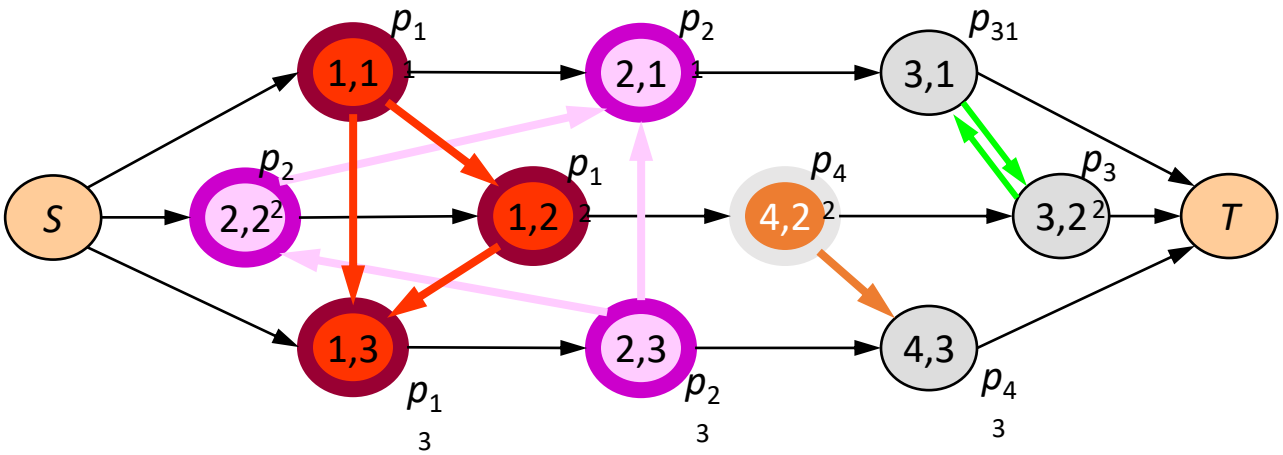
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



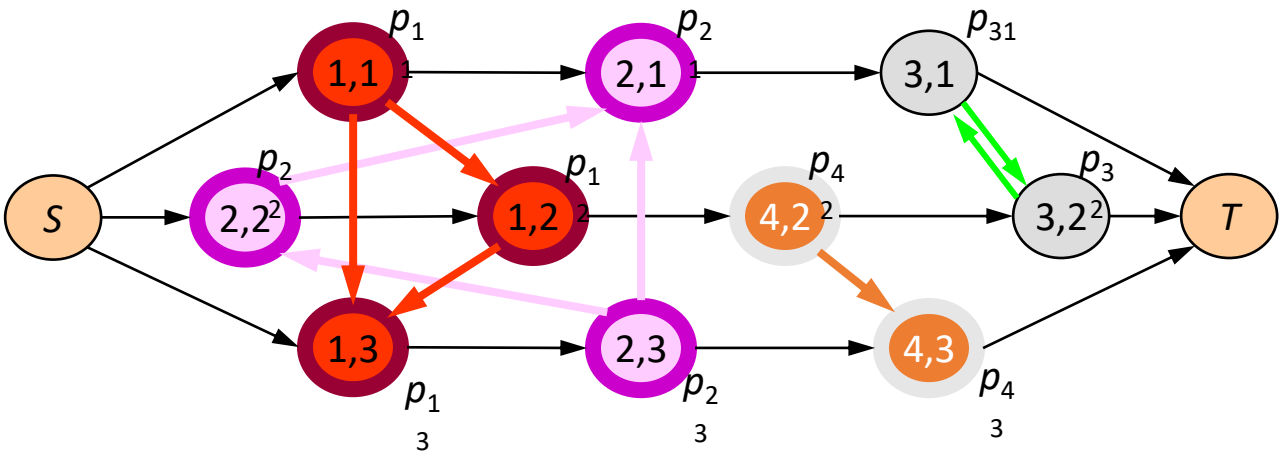
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



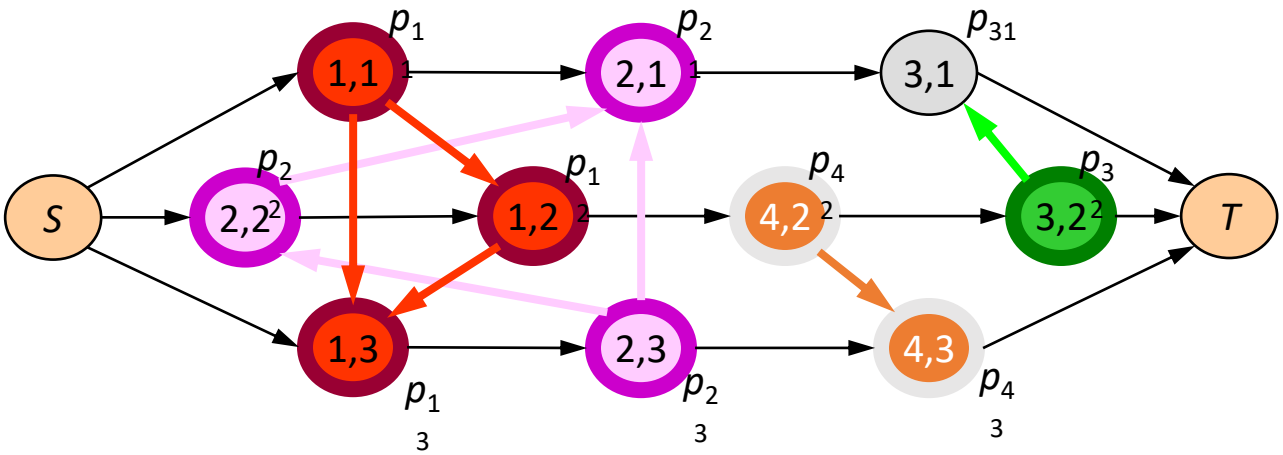
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



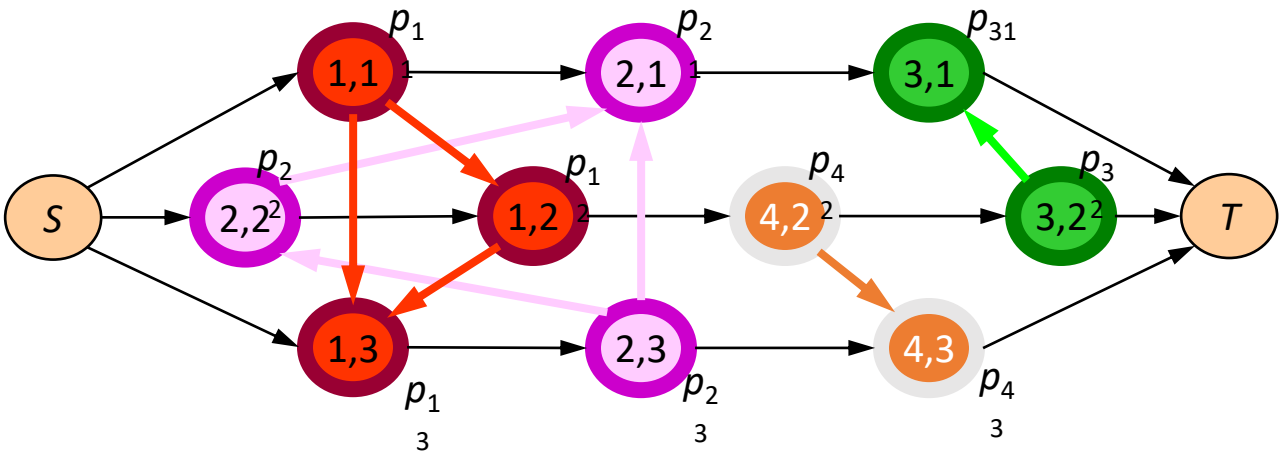
Job Shop: Disjunctive Graph Model

Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



Job Shop: Disjunctive Graph Model

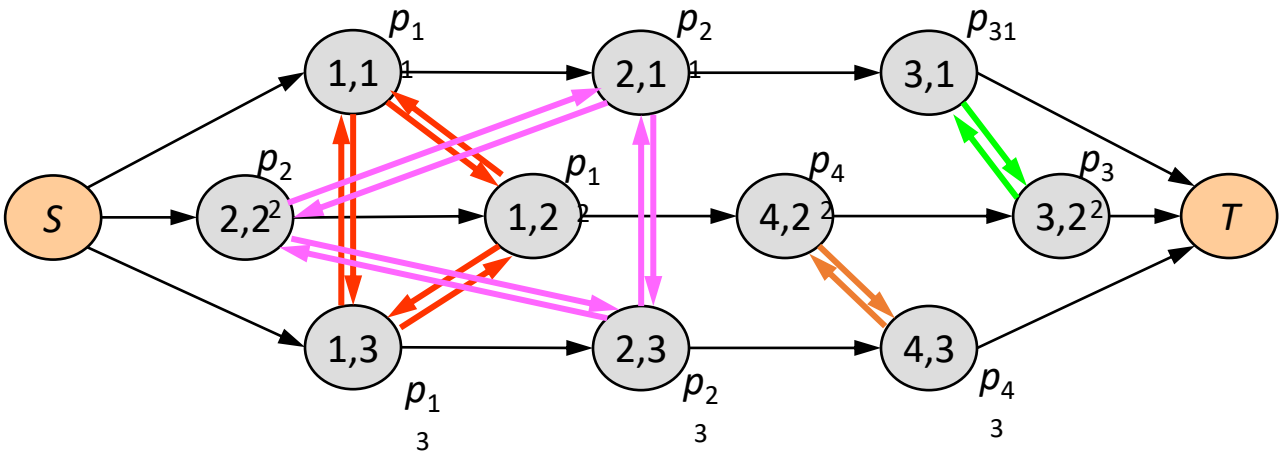
Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3



Job Shop: Disjunctive Graph Model

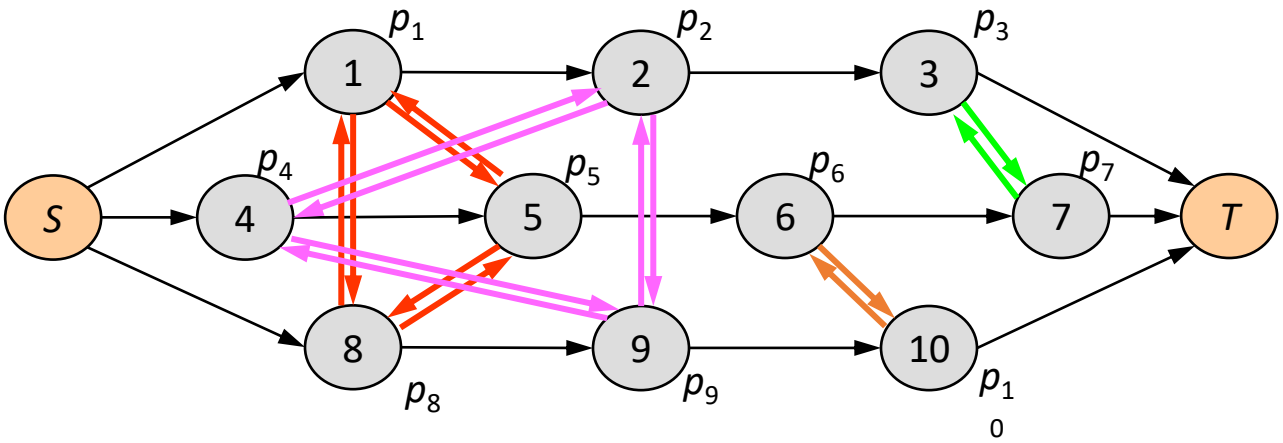
Disjunctive graph $G=(N, A, B)$:

- Nodes N correspond to all operations
- Conjunctive arcs A represent the precedence relations between operations of a single job
- Disjunctive arcs B link the operations processed by the same machine

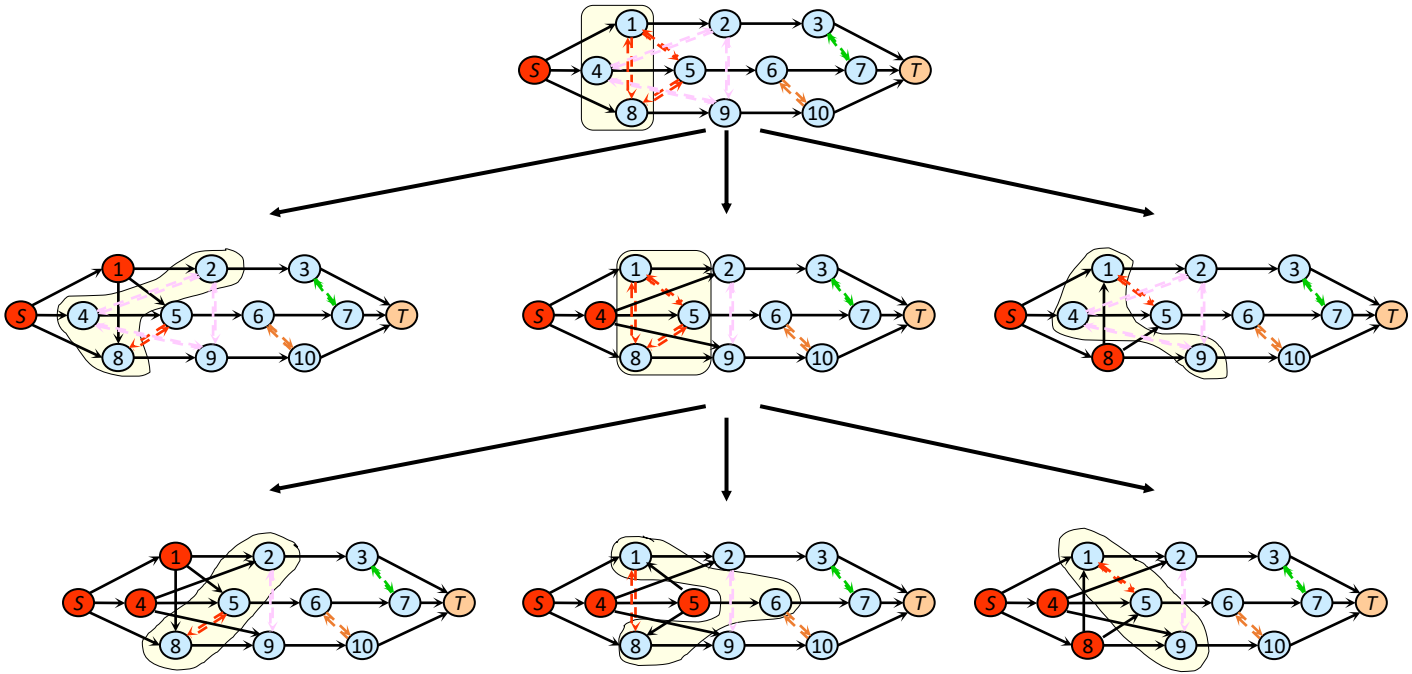


Job Shop: Branch & Bound

- Operations 1, 2, …, *q*
- *O* denotes the set of "schedulable operations" (whose predecessors have already been scheduled)
- "Scheduling" of operation *i* means replacing each pair of disjunctive arcs incident to *i* by a conjunctive arc starting from *i*

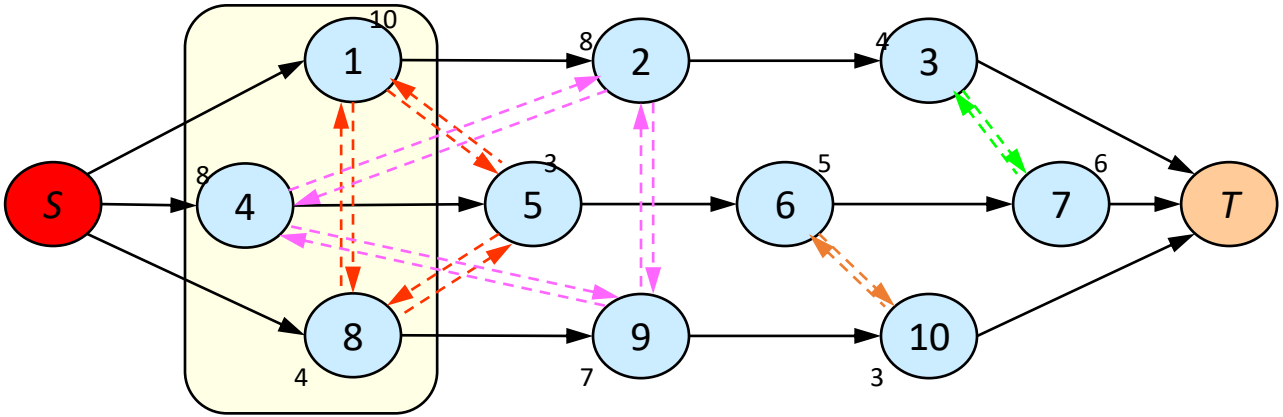


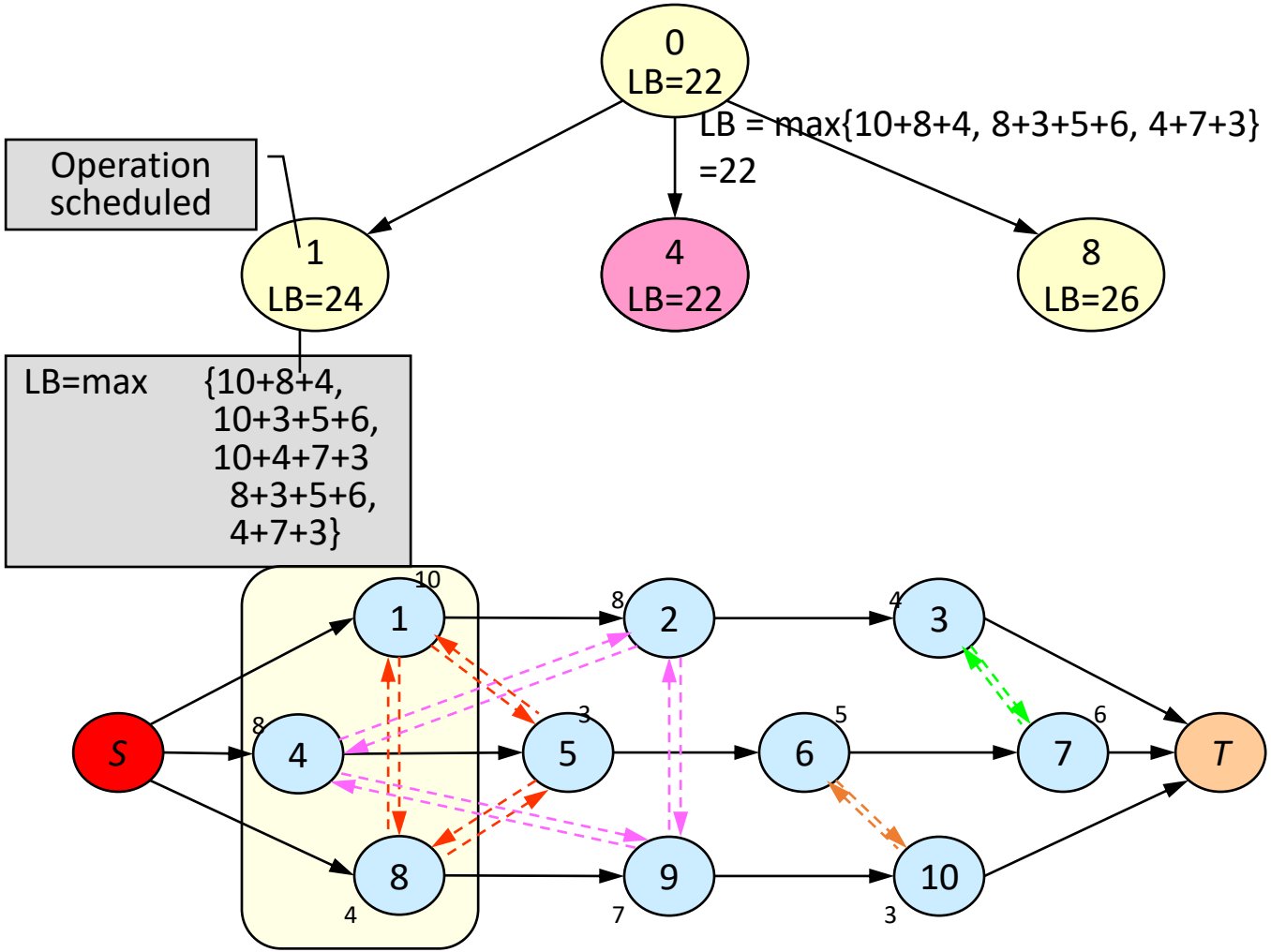
Job Shop: Branch & Bound

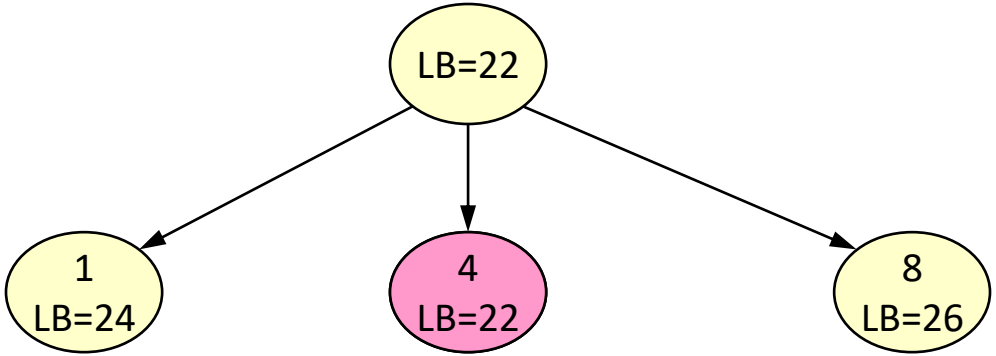


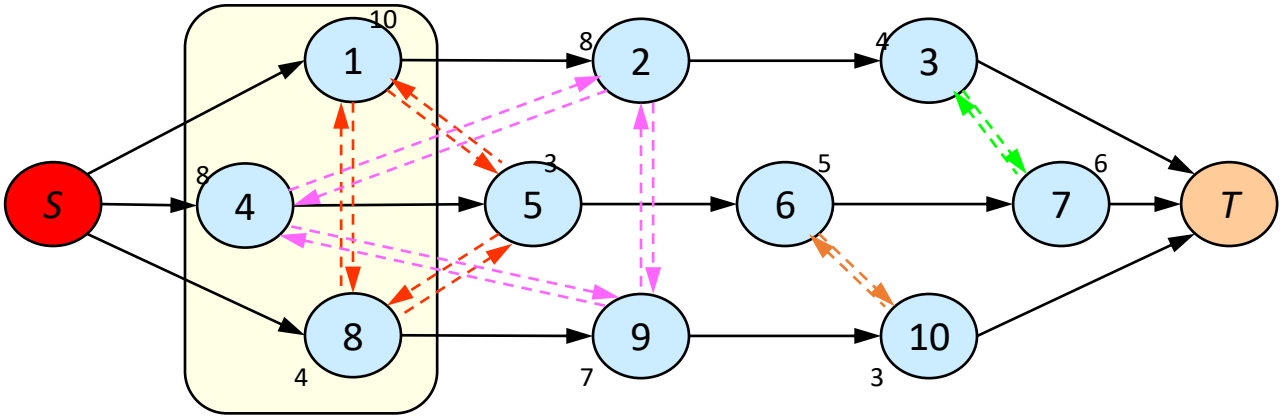
Job Shop: Branch & Bound

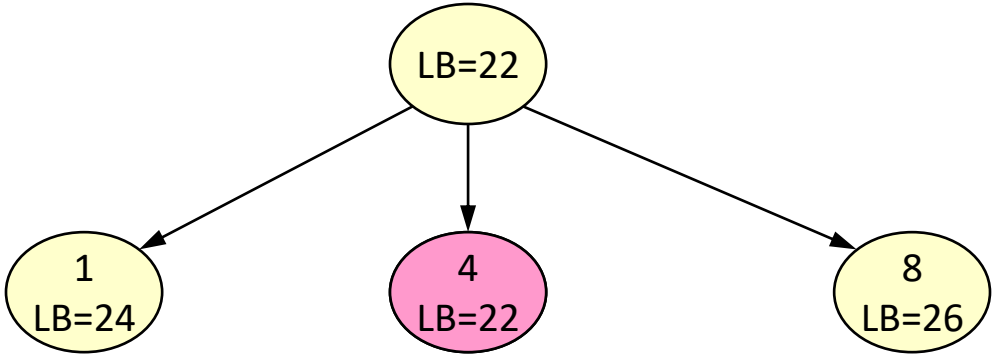
Jobs	Machine sequence	Processing times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{12}=3, p_{22}=8, p_{32}=6, p_{42}=5
3	1, 2, 4	p_{13}=4, p_{23}=7, p_{43}=3

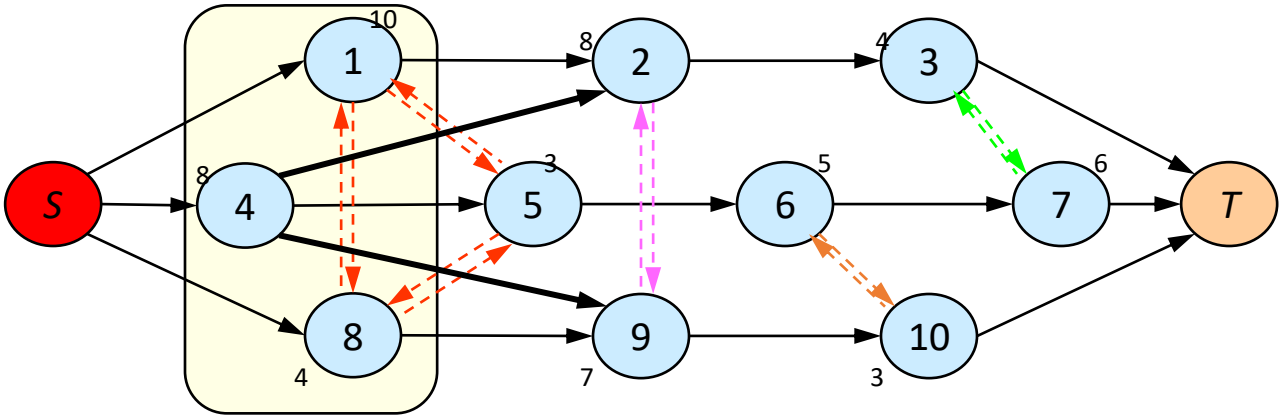


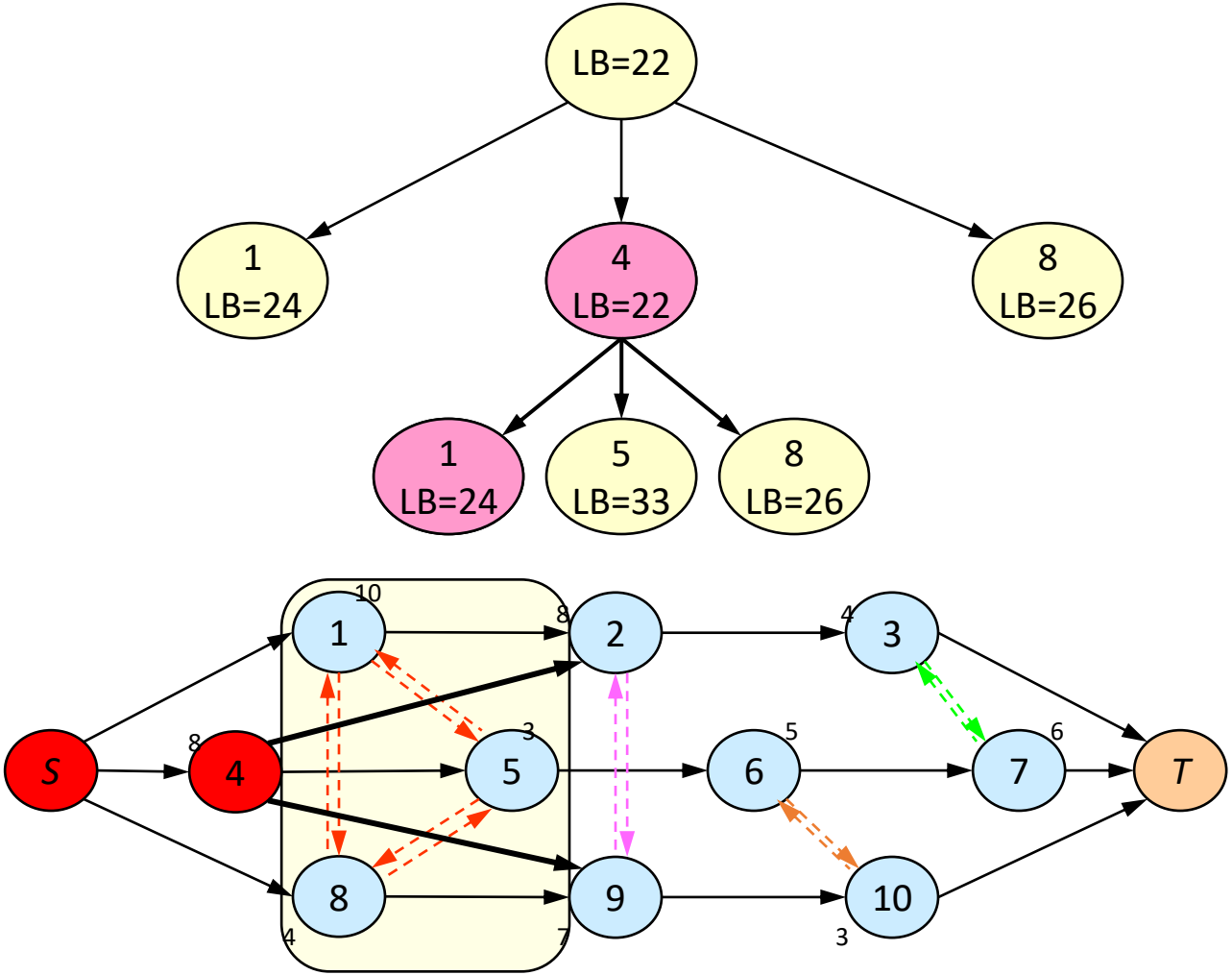

0
LB=22
LB = max{10+8+4, 8+3+5+6, 4+7+3}
=22
Operation scheduled
1
LB=24
4
LB=22
8
LB=26
LB=max {10+8+4,
10+3+5+6,
10+4+7+3
8+3+5+6,
4+7+3}
S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
3

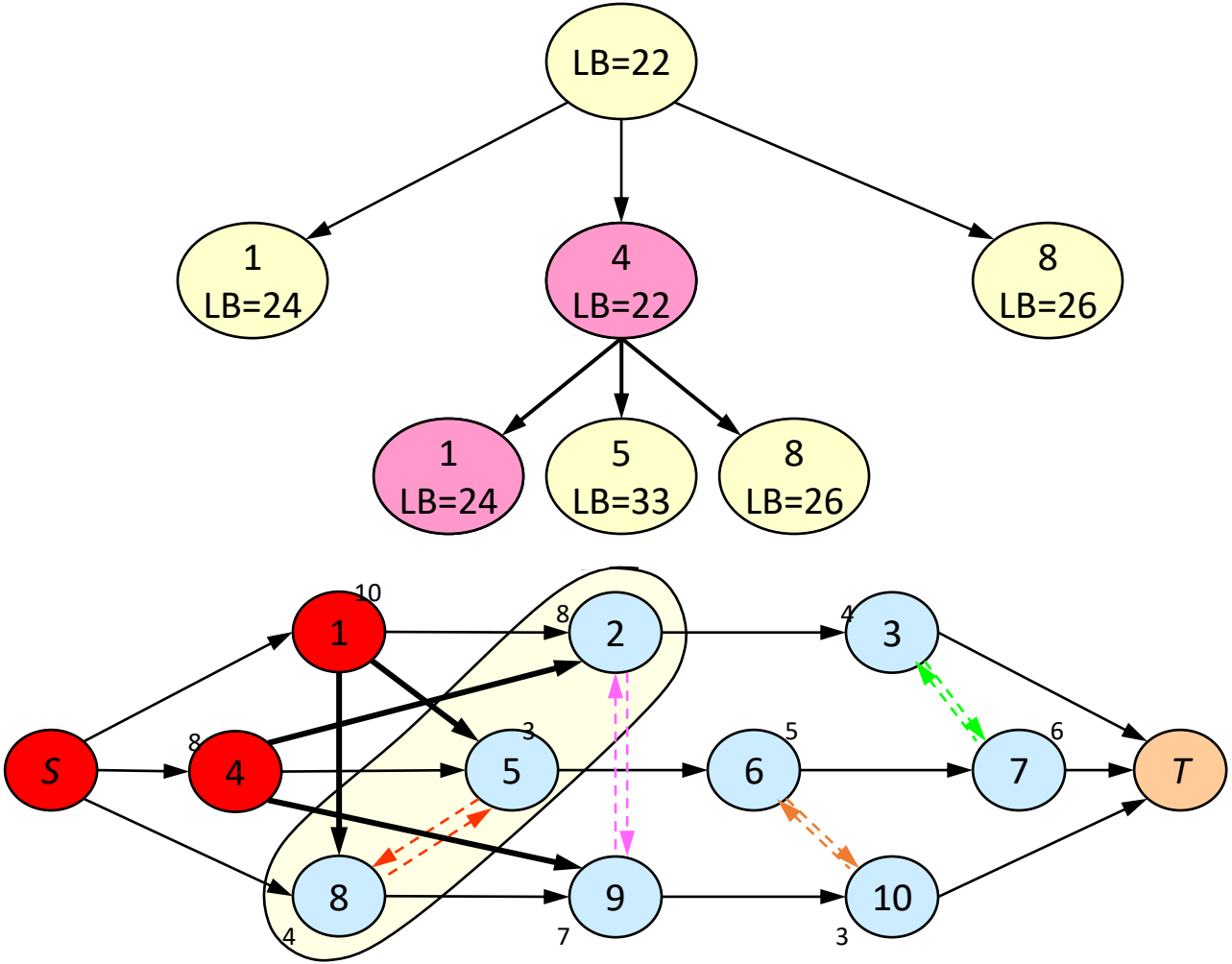

LB=22
1
LB=24
4
LB=22
8
LB=26

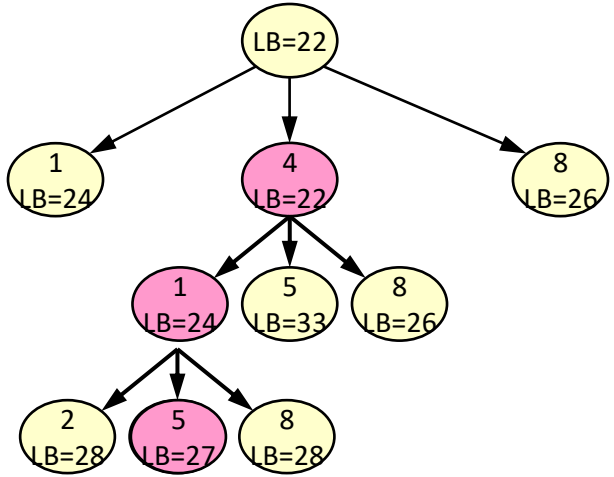

S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
6

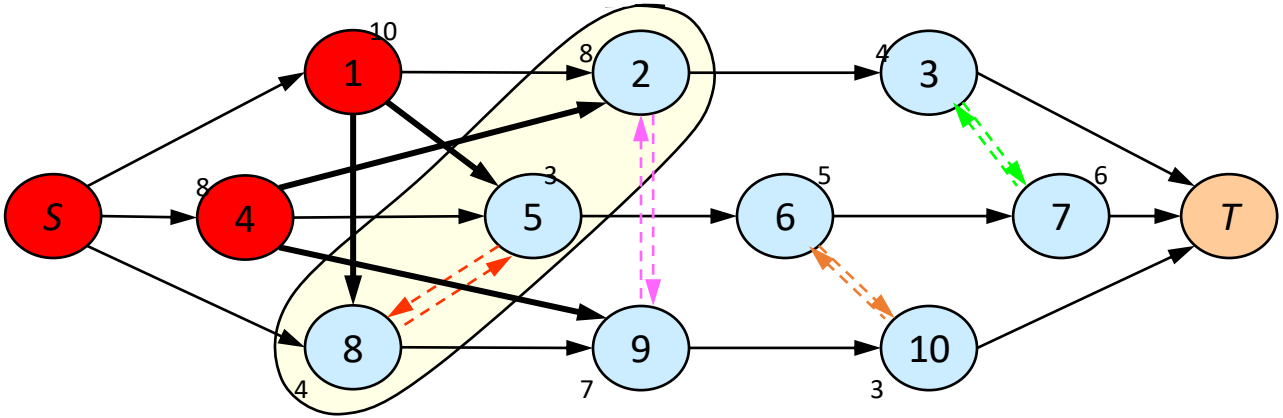

LB=22
1
LB=24
4
LB=22
8
LB=26

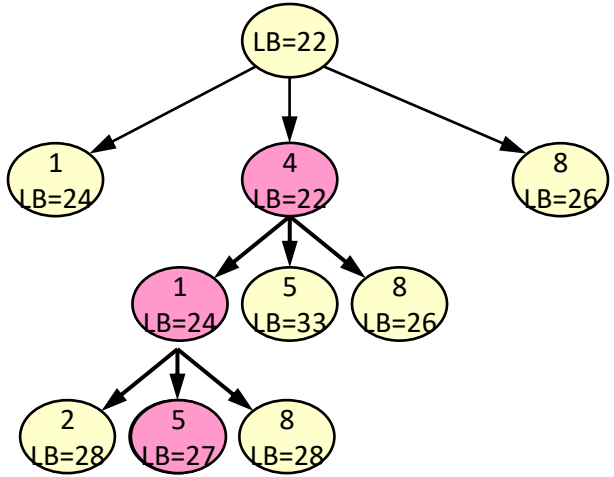

S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
3

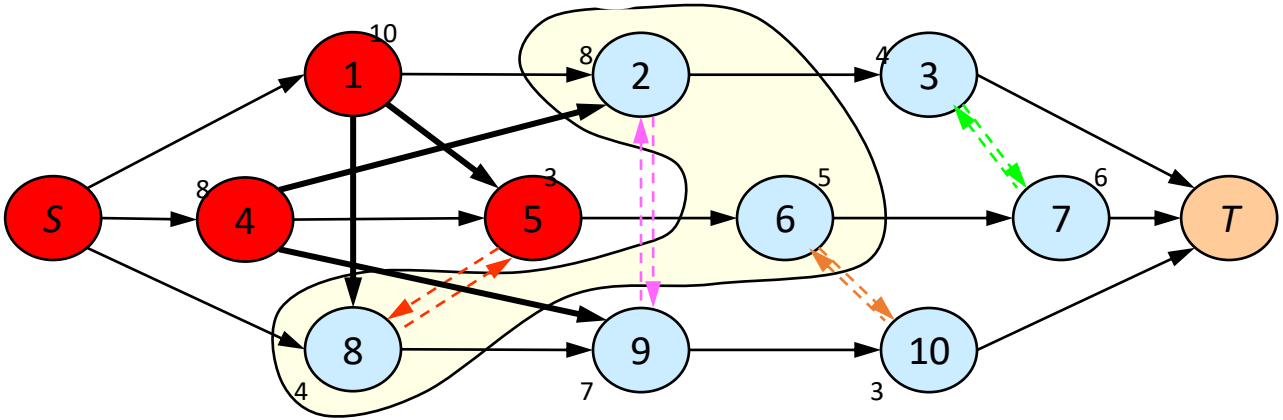

LB=22
1
LB=24
4
LB=22
8
LB=26
1
LB=24
5
LB=33
8
LB=26
S
4
1
2
3
5
6
7
8
9
10
T
10
8
4
3
5
6
4
7
3

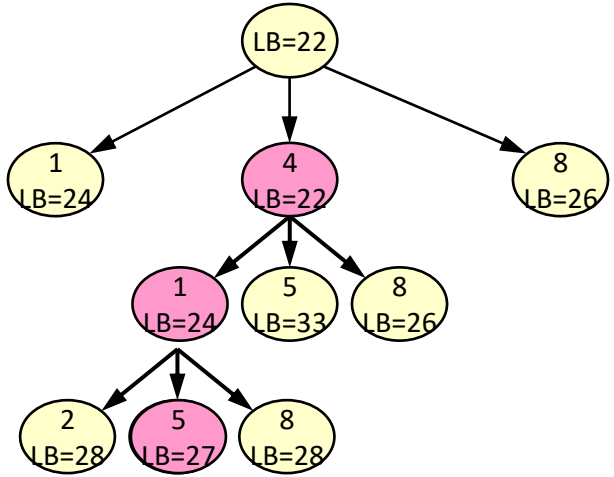

LB=22
1
LB=24
4
LB=22
8
LB=26
1
LB=24
5
LB=33
8
LB=26
S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
6
4
7
3

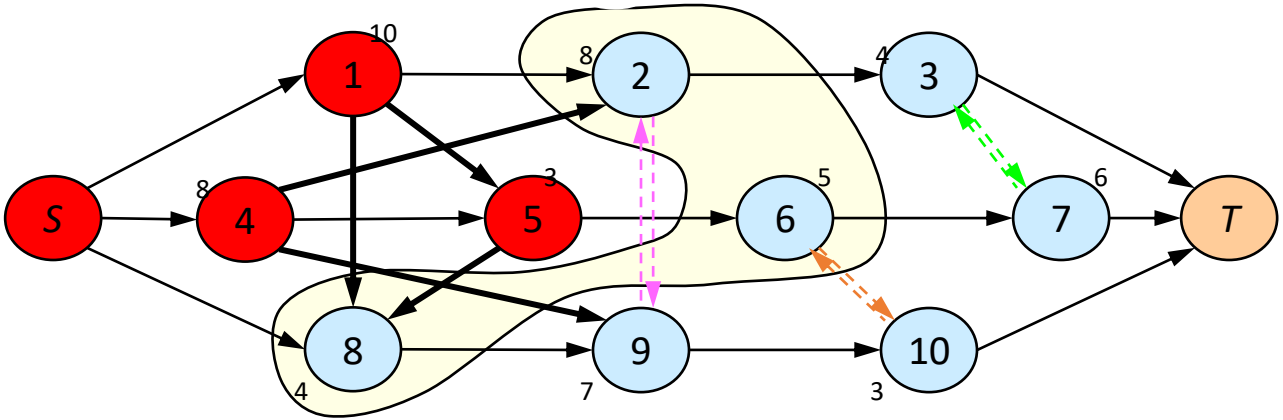

LB=22
1
LB=24
4
LB=22
8
LB=26
1
LB=24
5
LB=33
8
LB=26
2
LB=28
5
LB=27
8
LB=28

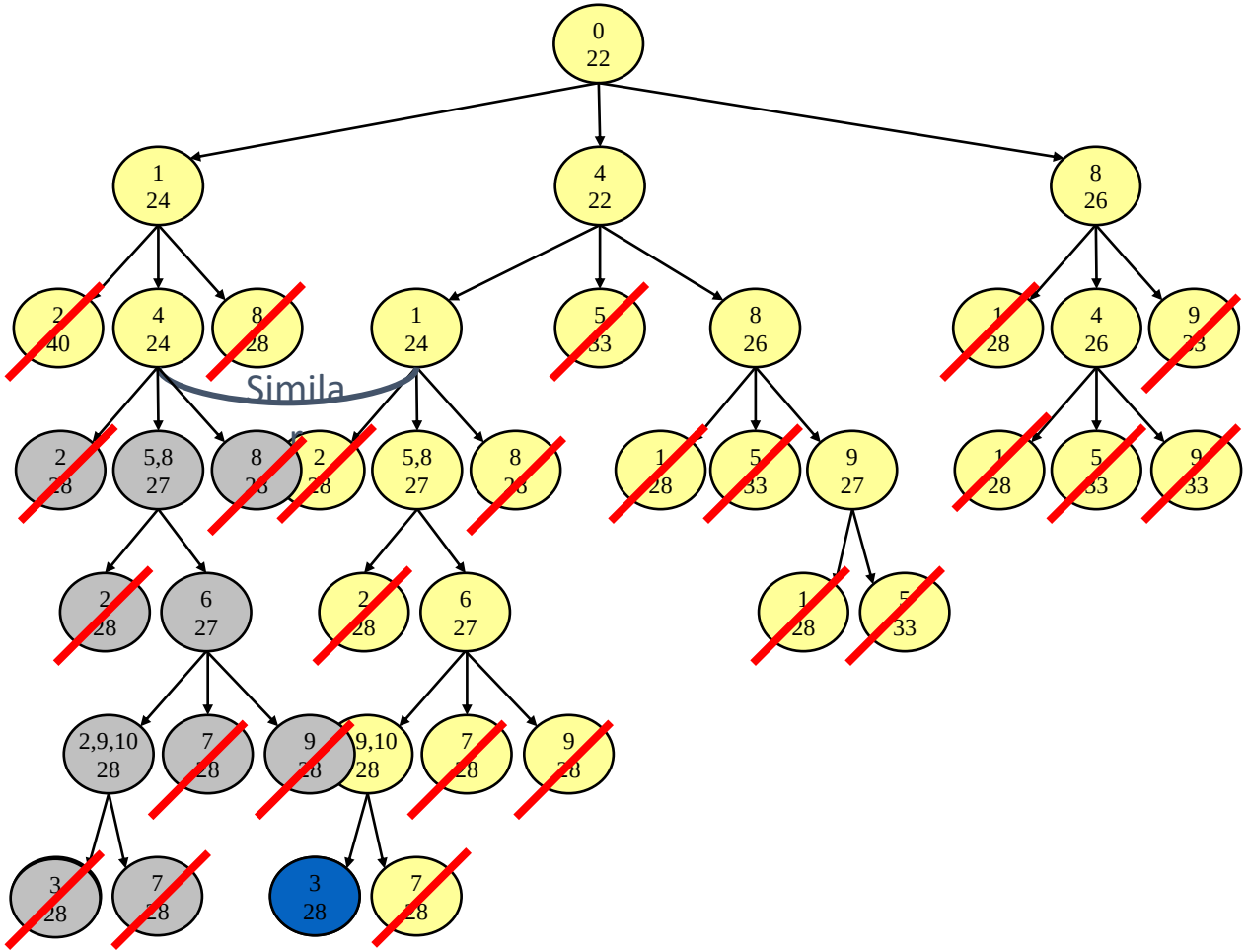

S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
3


LB=22
1
LB=24
4
LB=22
8
LB=26
1
LB=24
5
LB=33
8
LB=26
2
LB=28
5
LB=27
8
LB=28


S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
3


LB=22
1
LB=24
4
LB=22
8
LB=26
1
LB=24
5
LB=33
8
LB=26
2
LB=28
5
LB=27
8
LB=28


S
1
2
3
4
5
6
7
8
9
10
T
10
8
4
8
3
5
6
4
7
3


0
22
1
24
4
22
8
26
2
40
4
24
8
28
1
24
5
33
8
26
1
28
4
26
9
33
Simila
2
28
5,8
27
8
28
2
28
5,8
27
8
28
1
28
5
33
9
27
1
28
5
33
9
33
2
28
6
27
2
28
6
27
1
28
5
33
2,9,10
28
7
28
9
28
9,10
28
7
28
9
28
3
28
7
28
3
28
7
28

Job Shop: Branch & Bound

Advantages of Branch & Bound algorithm:

- Finds an optimal solution

Disadvantages of Branch & Bound algorithm:

- Extremely time-consuming: the number of nodes in a branching tree can be too large.

  Algorithm can construct several feasible schedules and then develop “bad” branches of the tree. The number of the nodes grows exponentially without improving the best solution obtained.

Job Shop: Branch & Bound

$10 \times 10 = 930$

- Fischer and Thompson (1963) posed a small example of the jobs shop problem with 10 jobs and 10 machines.
- The example remained open for more than 25 years.
- It was finally solved by
  - Carlier & Pinson (1989)
  - Applegate & Cook (1991)
  - Brucker, Jurisch & Sievers (1994)

Job shop problem remains one of the most difficult combinatorial problems to date, and always arouses new research interest.

Job Shop: Beam Search

With Beam Search only the most promising nodes at level *k* are selected as nodes to branch from. The remaining nodes at that level are discarded permanently.

The *beam width of the search* is the number of nodes retained in the tree.

The number of nodes selected is the *filter width*.

The choice of the

- Evaluating each node carefully is time-consuming.
- A crude prediction is quick but it may result in discarding good solutions.

The number of nodes selected is the *beam width*.

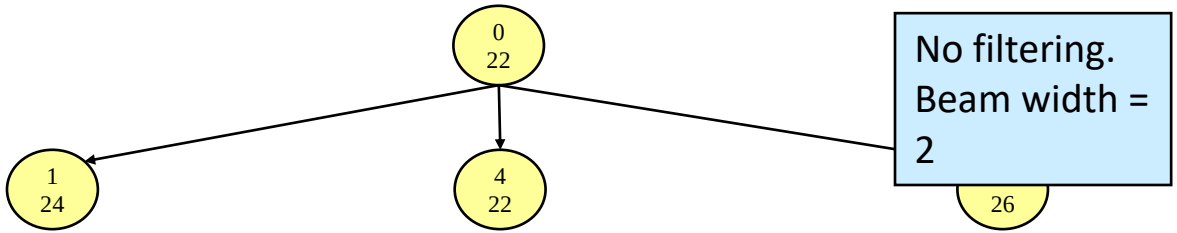
Two-Stage Approach

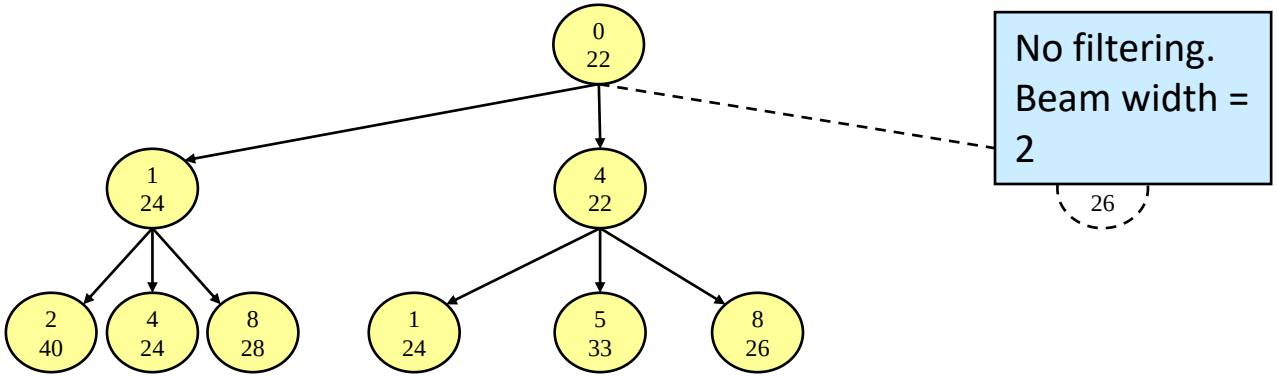
1. For all bodes generated at a particular level of the tree, do a crude evaluation (speed).
2. Select the best nodes for the thorough evaluation (accuracy).
3. After thorough evaluation is done, select the most promising nodes.

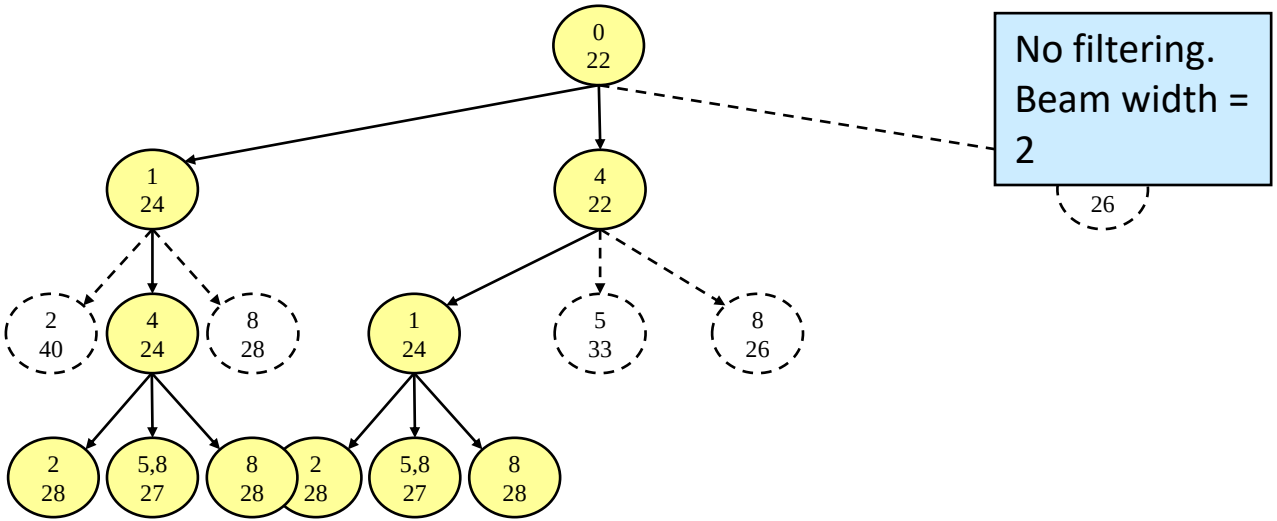
Job Shop: Beam Search

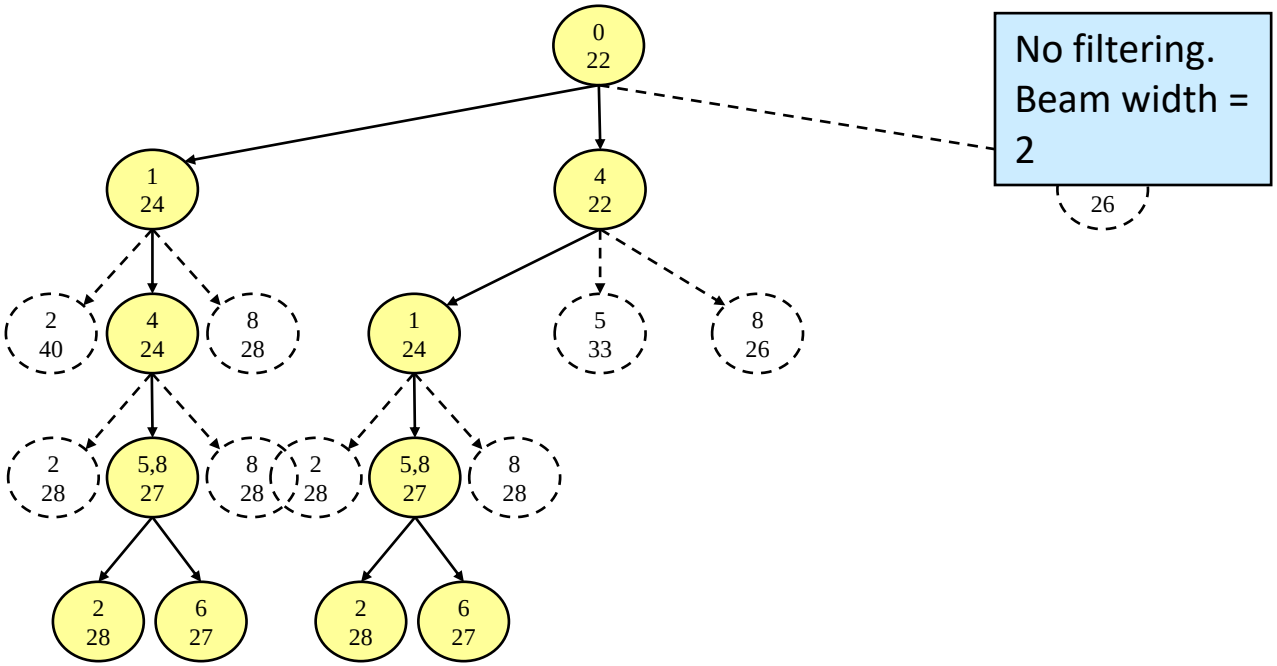
Example of crude prediction: the contribution of the partial schedule to the objective function (the length of the critical path in the graph after some operation has been scheduled).

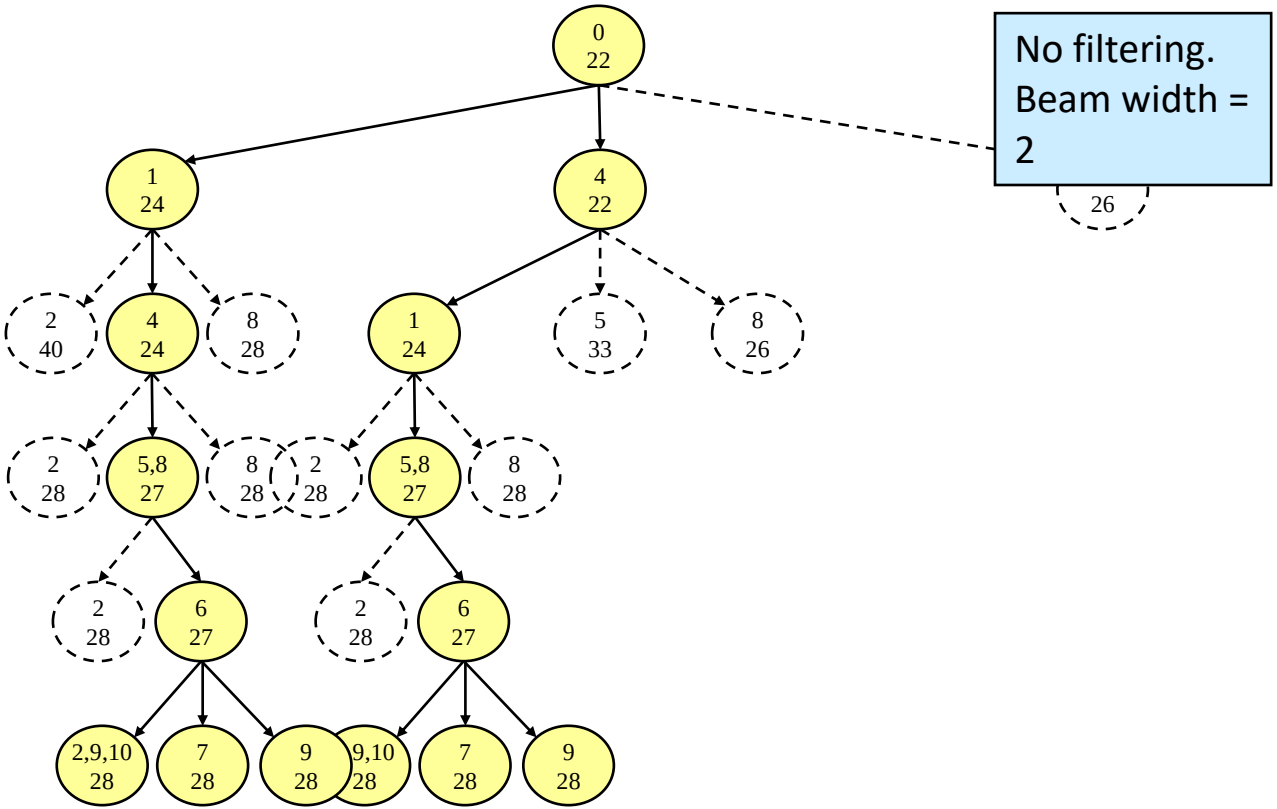
Example of thorough evaluation: all the jobs that have not yet been scheduled are scheduled according to composite dispatching rule.

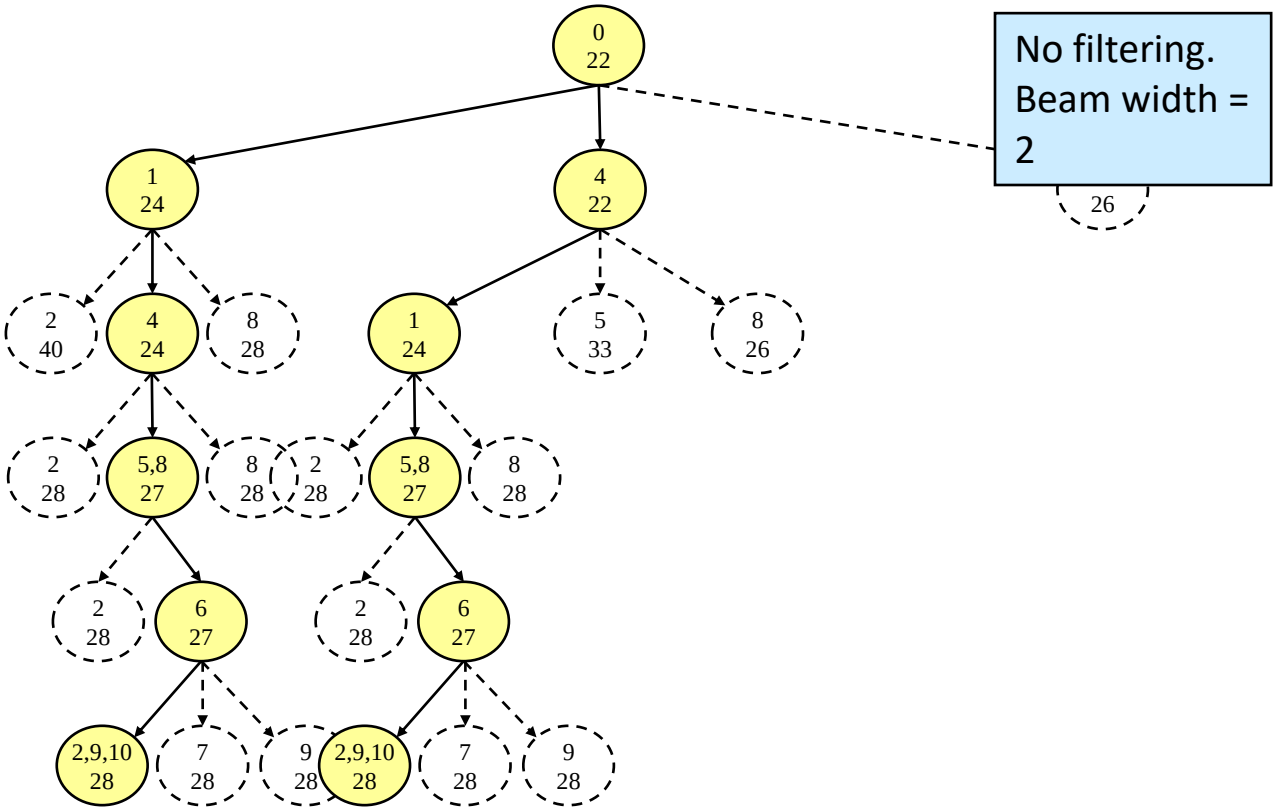

0
22
1
24
4
22
26
No filtering.
Beam width =
2

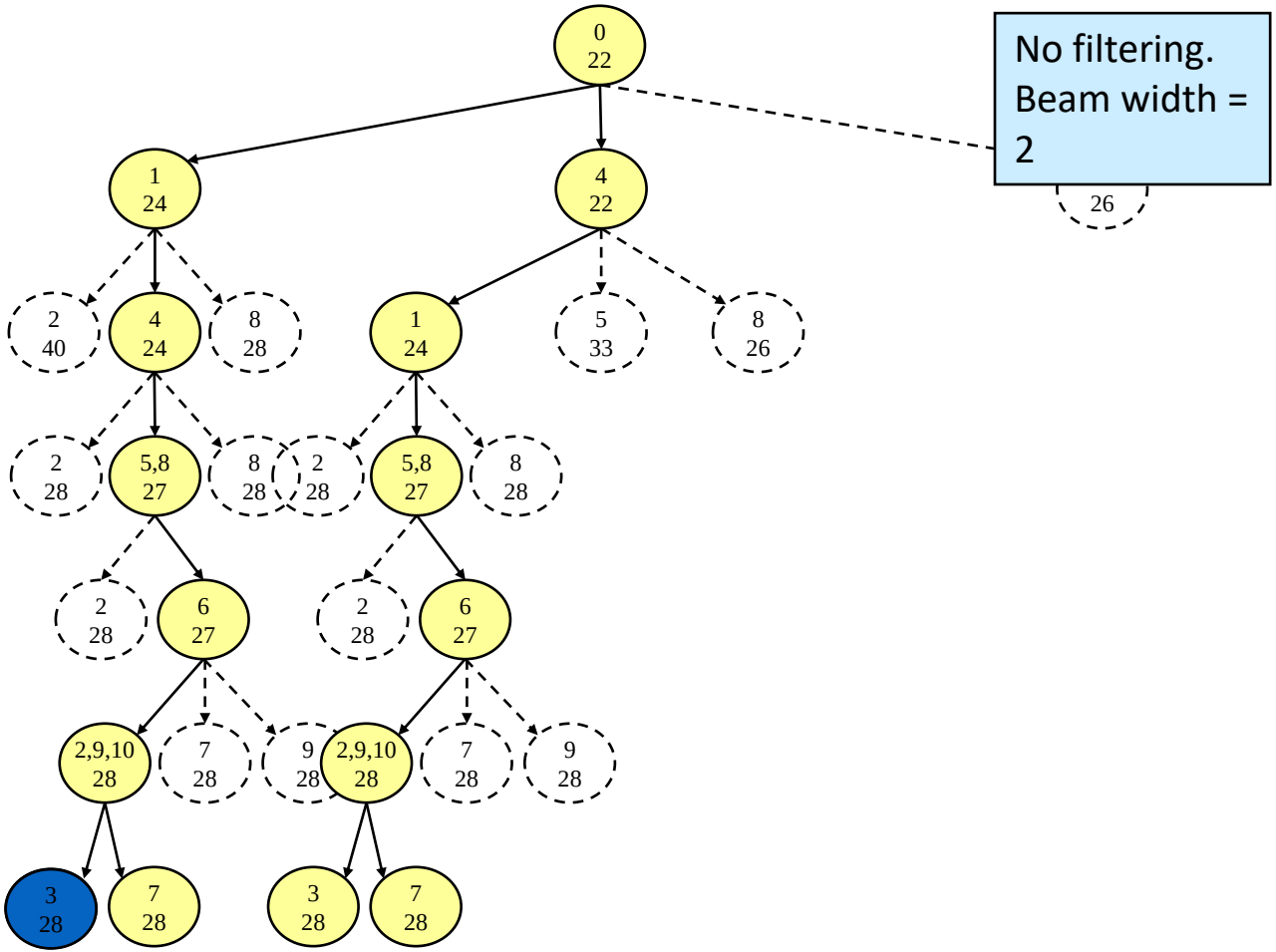

0
22
1
24
4
22
2
40
4
24
8
28
1
24
5
33
8
26
No filtering.
Beam width =
2
26

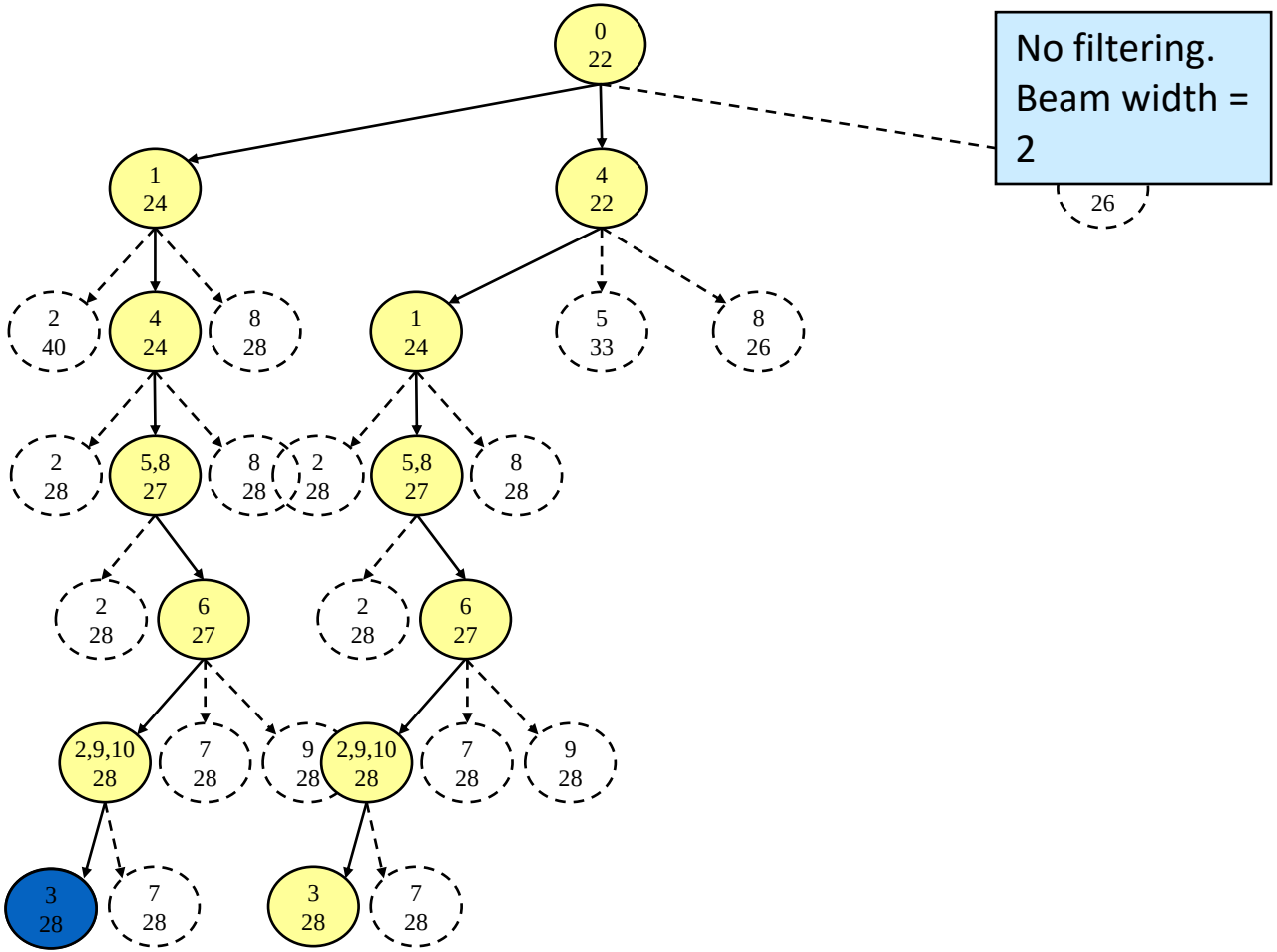

0
22
No filtering.
Beam width =
2
1
24
4
22
26
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28


0
22
1
24
4
22
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28
2
28
6
27
2
28
6
27
No filtering.
Beam width =
2
26


0
22
1
24
4
22
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28
2
28
6
27
2
28
6
27
2,9,10
28
7
28
9
28
9,10
28
7
28
9
28
No filtering.
Beam width =
2
26


0
22
1
24
4
22
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28
2
28
6
27
2
28
6
27
2,9,10
28
7
28
9
28
2,9,10
28
7
28
9
28
No filtering.
Beam width =
2
26


0
22
No filtering.
Beam width =
2
1
24
4
22
26
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28
2
28
6
27
2
28
6
27
2,9,10
28
7
28
9
28
2,9,10
28
7
28
9
28
3
28
7
28
3
28
7
28


0
22
No filtering.
Beam width =
2
1
24
4
22
26
2
40
4
24
8
28
1
24
5
33
8
26
2
28
5,8
27
8
28
2
28
5,8
27
8
28
2
28
6
27
2
28
6
27
2,9,10
28
7
28
9
28
2,9,10
28
7
28
9
28
3
28
7
28
3
28
7
28

Dispatching Rules

A *dispatching rule* is a rule that prioritises all the jobs that are awaiting for processing on a machine.

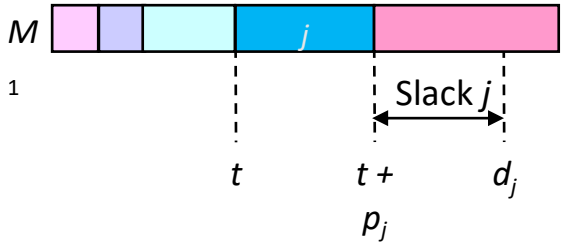
Whenever a machine has been freed, a dispatching rule inspects the waiting jobs and selects the job with the highest priority.

Dispatching Rules

Rules dependent on processing times and weights

Rule		Formal description	Objectives
SPT	*Shortest Processing Time* first	$\uparrow p_j$	ΣC_j
ECT	*Earliest Completion Time* first	$\uparrow S_j + p_j$	ΣC_j
WSPT	*Weighted Shortest Processing Time* first	$\uparrow p_j / w_j$	$\Sigma w_j C_j$
WI	*With Biggest Weight*	$\downarrow w_j$	$\Sigma w_j C_j$
LPT	*Longest Processing Time* first	$\downarrow p_j$	C_{max}

Dispatching Rules



Rules dependent on release and due dates

Rule		Formal description	Objectives
ERD	*Earliest Release Date* first (equivalent to First-Come-First-Served rule)	↑ r_j	Various criteria
EDD	*Earliest Due Date* first	↑ d_j	L_{max}
MS	*Minimum Slack* first (if a machine is freed at time t, the remaining slack of a job $\max\{d_j - p_j - t, 0\}$ is computed and the job with the minimum slack is scheduled next)	↑ $Slack_j$	L_{max}

Dispatching Rules

The basic dispatching rules are of limited use:

— when a complex objective has to be minimised, none of the basic dispatching rules can perform effectively;

— combination of basic dispatching rules can perform significantly better.

Composite Dispatching Rules

— It is a ranking expression that combines a number of basic dispatching rules.

— Each basic rule in the composite dispatching rule has its own scaling parameter that is chosen to properly scale the contribution of the basic rule to the total ranking expression.

Dispatching Rules

Example: Consider $1 \mid\mid \Sigma w_j T_j$, which is NP-hard in the strong sense.

A *heuristic* is needed which provides a reasonably good solution with a reasonable computational effort.

Appropriate basic dispatching rules:

— WSPT (optimal for $\Sigma w_j C_j$),

— EDD, MS (optimal when due dates are loose and spread out)

Dispatching Rules

Composite Dispatching Rule

ATC - Apparent Tardiness Cost

- combines WSPT and MS
- jobs are scheduled one at a time: every time the machine becomes free, a ranking index is computed for each remaining job and the job with the highest ranking index is then selected to be processed next.

Dispatching Rules

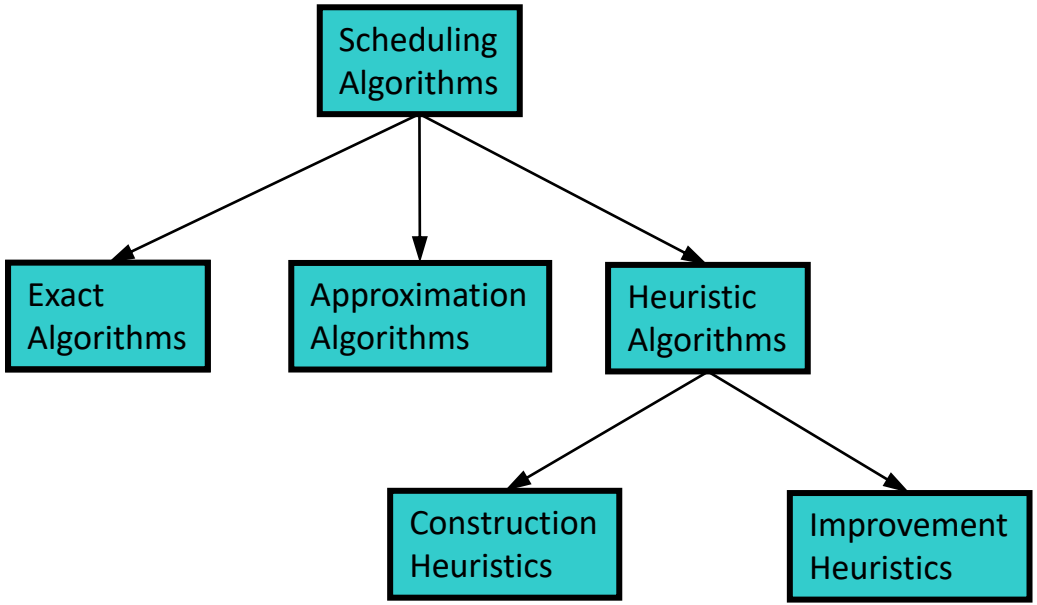
Advantages of dispatching rules:

- Very simple to implement
- Fast
- Can find a reasonably good solution in a relatively short time
- Optimal for special cases

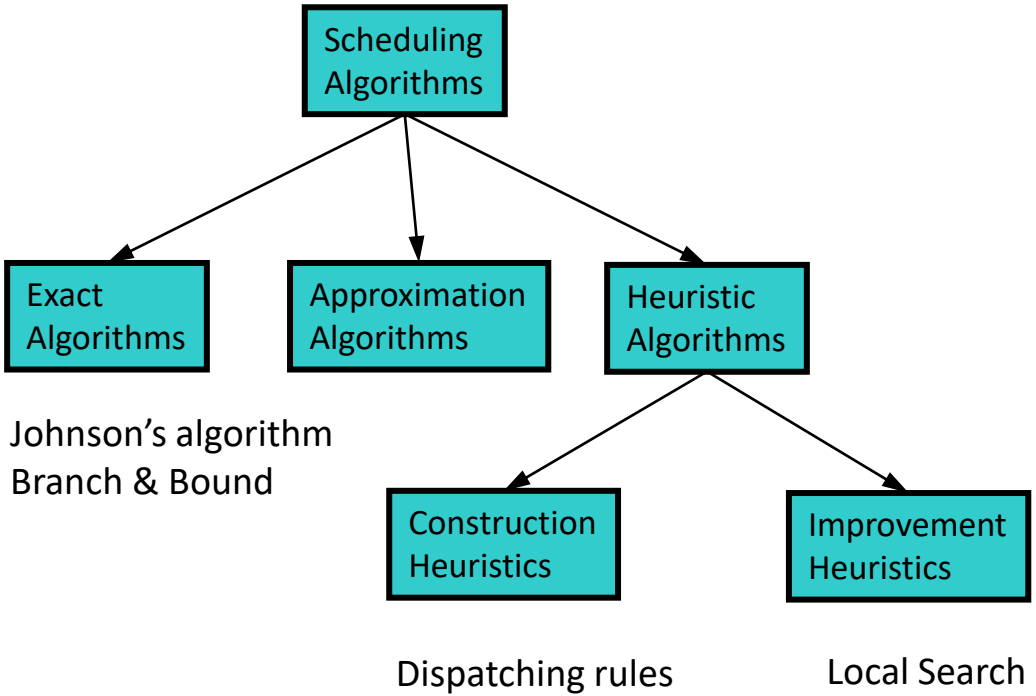
Disadvantages of dispatching rules:

- Limited use in practice
- Can find unpredictably bad solution

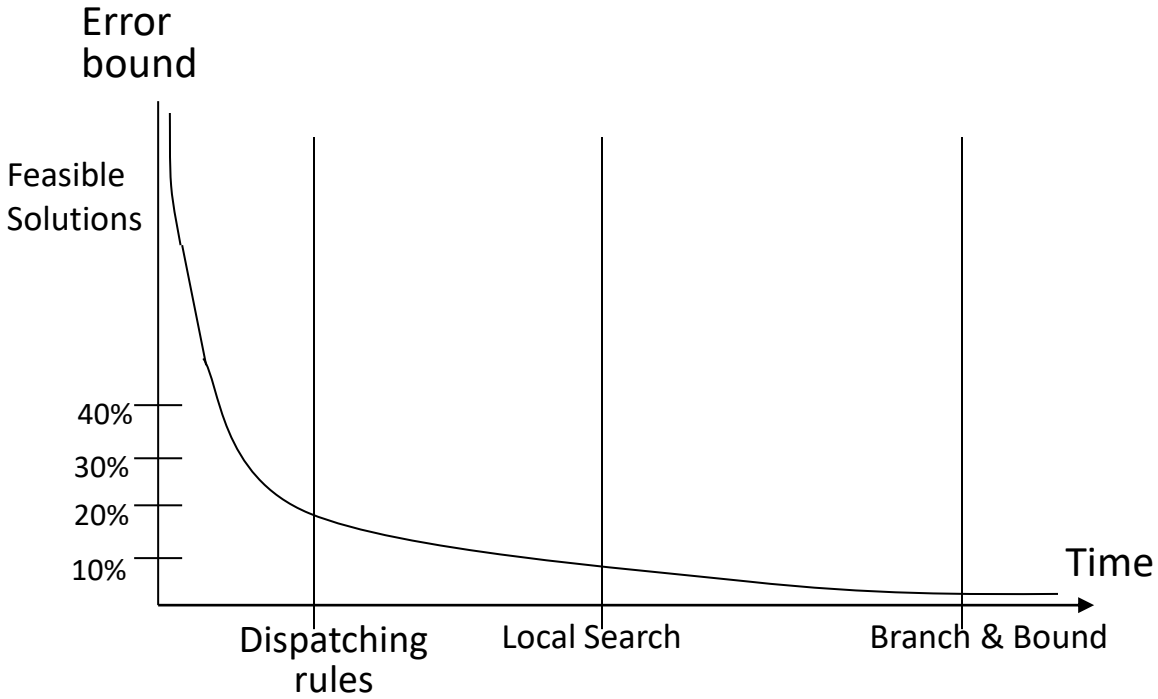
Algorithms Classification



Algorithms Classification



Choosing an Algorithm



Local Search

Local Search is an iterative algorithm that moves from one solution S to another S' according to some neighbourhood structure.

1. Iterative Improvement
2. Threshold Accepting
3. Simulated Annealing
4. Tabu search

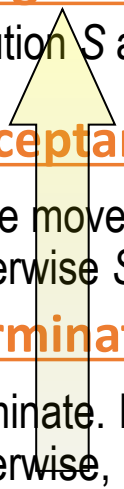
General Local Search Procedure

1. **Initialisation.** Choose an initial schedule S to be the current solution and compute the value of the objective function $F(S)$.

2. **Neighbour Generation.** Select a neighbour S' of the current solution S and compute $F(S')$.

3. **Acceptance Test.** Test whether to accept the move from S to S'. If the move is accepted, then S' replaces S as the current solution; otherwise S is retained as the current solution.

4. **Termination Test.** Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the neighbour generation step.



General Local Search Procedure

1. **Initialisation.**

   Random permutation $(J_1, J_2, \ldots, J_n)$ for single-machine problem or flow-shop, more complicated structures for other models.

2. **Neigh**

   Transpose neighbourhood: (1,2,3,4,5,6,7) → (1,3,2,4,5,6,7)
   Swap neighbourhood: (1,2,3,4,5,6,7) → (1,6,3,4,5, 2,7)
   Insert neighbourhood: (1,2,3,4,5,6,7) → (1,3,4,5,6, 2,7)

   Neighbours may be generated randomly, systematically, or by some combination of the two approaches.

3. **Acceptance Test.**

   Specific for each type of local search algorithm

4. **Termination Test**

   Limited computation time or the number of iterations

General Local Search Procedure

In Step 3, the acceptance rule is usually based on values *F(S)* and *F(S')* of the objective function for schedules *S* and *S'*.

In some algorithms only moves to "better" schedules are accepted (schedule *S'* is better than *S* if *F(S')* < *F(S)*)

In others it may be allowed to move to "worse" schedules.

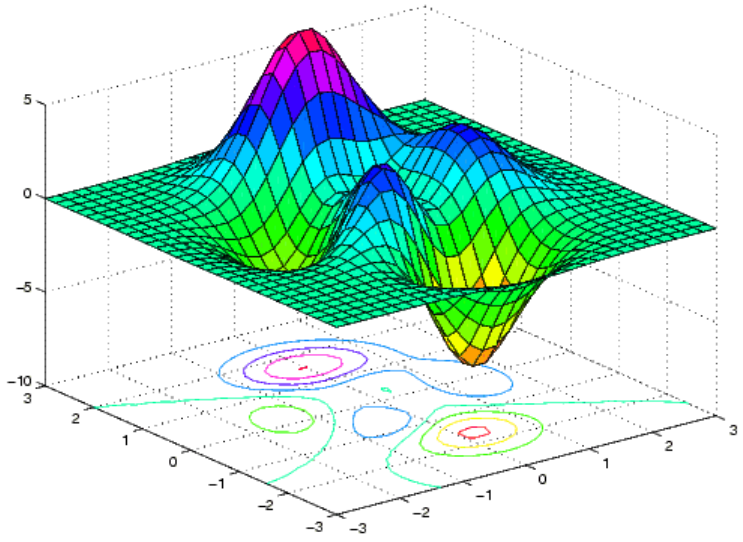
Sometimes "wait and see" approach is adopted.

The algorithm terminates in Step 4 if the computation time exceeds the limit or after completing the established number of iterations or if no further improvements are possible.

1. Iterative Impro

The algorithm can be applied repeatedly starting each time with a different randomly generated initial solution.

1. Initialisa
2. Neighbo
3. Accepta
4. Termina



As described

1. Iterative Improvement

Log Book - List of Schedules.seq

Schedule	Time	C_{max}	T_{max}	$\sum U_j$	$\sum C_j$	$\sum T_j$	$\sum w_j C_j$	$\sum w_j T_j$
Sch 1234	1	37	32	4	100	81	857	632
Sch 1243	1	37	36	4	91	72	705	480
Sch 1324	1	37	31	4	103	84	1003	778
Sch 1342	1	37	35	4	97	78	931	706
Sch 1423	1	37	36	4	85	66	633	408
Sch 1432	1	37	35	4	88	69	779	554
Sch 2134	1	37	32	4	100	81	877	652
Sch 2143	1	37	36	4	91	72	725	500
Sch 2314	1	37	29	4	103	84	1049	824
Sch 2341	1	37	33	4	97	78	985	760
Sch 2413	1	37	36	4	85	66	661	436
Sch 2431	1	37	33	4	88	69	833	608
Sch 3124	1	37	31	4	106	87	1175	950
Sch 3142	1	37	35	4	100	81	1103	878
Sch 3214	1	37	29	4	106	87	1195	970
Sch 3241	1	37	33	4	100	81	1131	906
Sch 3412	1	37	35	4	94	75	1039	814
Sch 3421	1	37	33	4	94	75	1059	834
Sch 4123	1	37	36	3	79	68	569	440
Sch 4132	1	37	35	3	82	71	715	586
Sch 4213	1	37	36	3	79	68	589	460
Sch 4231	1	37	33	3	82	71	761	632
Sch 4312	1	37	35	3	85	74	887	758
Sch 4321	1	37	33	3	85	74	907	778

2. Threshold Accep

Allows to continue the local search even if a local optimum has been obtained

1. **Initialisation** — As described

2. **Neighbour G** — As described

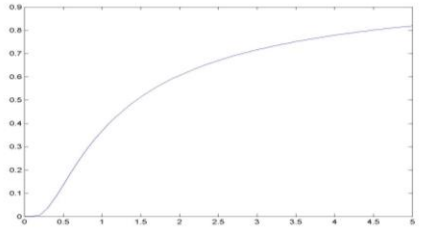
3. **Acceptanc** — Choose S' if $F(S')-F(S)<\alpha$, where $\alpha > 0$ is a threshold value. Usually α is relatively large at the beginning, becomes smaller later on.

4. **Termination** — As described

3. Simulated Annealing

1. **Initialisation.** — As described
2. **Neighbour Gene** — As described
3. **Acceptance Test** — Probabilistic acceptance test
4. **Termination Tes** — As described

3. Simulated Annealing



Probabilistic Acceptance Test:

Determine $\Delta = \boldsymbol{F}(\boldsymbol{S'}) - \boldsymbol{F}(\boldsymbol{S})$.

- If $\Delta \leq 0$, then a move to schedule ***S'*** is always accepted.
- If $\Delta > 0$, then a move to schedule ***S'*** is accepted with probability

$$e^{-\Delta/T},$$

where ***T*** is a parameter called the *temperature*, which changes during the course of the algorithm.

Usually *T* is large in the beginning and then it decreases until it is close to 0 at the final stages.

Different "*cooling schemes*" can be applied. Often $T_k = a^k$, where T_k is the temperature at iteration k and $0<a<1$.

“Cycling” of SA and TA

- It is possible to get back to the solutions already visited

- A simple way to avoid such problems is to store all visited solutions in a list called *tabu list*.
- A new solution can be accepted if it is not contained in the list.

4. Tabu Search (TS)

1. **Initialisation.** As described

2. **Neighbour Generation** As described

3. **Acceptance Test.**
   - A “worse” schedule S’ may be accepted
   - Deterministic acceptance test based on *tabu list*

4. **Termination Test** As described

4. Tabu Search (TS)

Tabu List

- Tabu list stores attributes of the previous few moves.
- It has a fixed number of entries (usually between 5 and 9).
- It is updated each time ***S'*** is accepted:
  - the reverse transformation is entered at the top of tabu list to avoid returning to the same solution (to avoid returning to a local optimum);
  - all other entries are pushed down one position;
  - the bottom entry is deleted.

4. Tabu Search (TS)

Deterministic Acceptance Test:

Determine $\Delta = F(S') - F(S)$.

- If $\Delta<0$ and ***S'*** is "non-tabu", then a move to ***S'*** is always accepted
- If $\Delta<0$ and ***S'*** is "tabu", then a move to ***S'*** may be accepted for a "promising" schedule ***S'*** (if $F(S')$ is less than the objective function value for any other solution obtained before)
- If $\Delta\geq 0$ and ***S'*** is "tabu", then a move to ***S'*** is always rejected.
- If $\Delta \geq 0$ and ***S'*** is "non-tabu", then a "wait and see" approach is adopted: ***S'*** remains as a candidate while the search continues for a neighbour which can be accepted immediately. If no such neighbour is found, a move to the best candidate ***S'*** is made.

4. Tabu Search (TS)

Log Book - List of Schedules.seq

Schedule	Time	C_{max}	T_{max}	$\sum U_j$	$\sum C_j$	$\sum T_j$	$\sum w_j C_j$	$\sum w_j T_j$
Sch 1234	1	37	32	4	100	81	857	632
Sch 1243	1	37	36	4	91	72	705	480
Sch 1324	1	37	31	4	103	84	1003	778
Sch 1342	1	37	35	4	97	78	931	706
Sch 1423	1	37	36	4	85	66	633	408
Sch 1432	1	37	35	4	88	69	779	554
Sch 2134	1	37	32	4	100	81	877	652
Sch 2143	1	37	36	4	91	72	725	500
Sch 2314	1	37	29	4	103	84	1049	824
Sch 2341	1	37	33	4	97	78	985	760
Sch 2413	1	37	36	4	85	66	661	436
Sch 2431	1	37	33	4	88	69	833	608
Sch 3124	1	37	31	4	106	87	1175	950
Sch 3142	1	37	35	4	100	81	1103	878
Sch 3214	1	37	29	4	106	87	1195	970
Sch 3241	1	37	33	4	100	81	1131	906
Sch 3412	1	37	35	4	94	75	1039	814
Sch 3421	1	37	33	4	94	75	1059	834
Sch 4123	1	37	36	3	79	68	569	440
Sch 4132	1	37	35	3	82	71	715	586
Sch 4213	1	37	36	3	79	68	589	460
Sch 4231	1	37	33	3	82	71	761	632
Sch 4312	1	37	35	3	85	74	887	758
Sch 4321	1	37	33	3	85	74	907	778

Conclusions

- Local search algorithms are very generic.
- They have been applied successfully to many industrial problems.
- Performance of local search algorithms depends on construction of neighborhood.
- A method that exploits special structure is usually faster (if one exists).

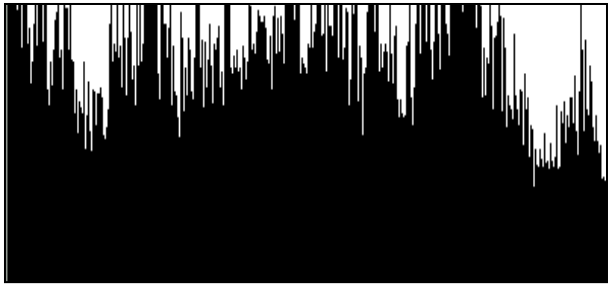
Intensification vs Diversification

- ***Intensification mechanisms*** aim to carefully examine a specific area of the search space in order to find a higher-quality solution. Intensification is often achieved by adopting a greedy approach (the search is directed to the best possible neighbour).
- ***Diversification mechanisms*** aim to explore the search space widely in order to avoid the search stagnation.
- An important issue in the design of a local search method is to achieve a good balance between intensification and diversification.

Local Search Process

B&B, Iterative Improvement

TA



TS

SA