Resume

Flow Shop
- Iterated Greedy
- Tabu Search (block representation and neighborhood pruning)

Job Shop:
- Definition
- Starting times and m-tuple permutation representation
- Disjunctive graph representation [Roy and Sussman, 1964]
- Shifting Bottleneck Heuristic [Adams, Balas and Zawack, 1988]
Generalizations: Time Lags

Generalized time constraints
They can be used to model:

- **Release time:**
  \[ S_0 + r_i \leq S_i \Leftrightarrow d_{0i} = r_i \]

- **Deadlines:**
  \[ S_i + p_i - d_i \leq S_0 \Leftrightarrow d_{i0} = p_i - d_i \]

- **Exact relative timing (perishability constraints):**
  if operation \( j \) must start \( l_{ij} \) after operation \( i \):
  \[ S_i + p_i + l_{ij} \leq S_j \quad \text{and} \quad S_j - (p_i + l_{ij}) \leq S_i \]
  \((l_{ij} = 0 \text{ if no-wait constraint})\)

- **Set up times:**
  \[ S_i + p_i + s_{ij} \leq S_j \quad \text{or} \quad S_j + p_j + s_{ji} \leq S_i \]

- **Machine unavailabilities:**
  - Machine \( M_k \) unavailable in \([a_1, b_1], [a_2, b_2], \ldots, [a_v, b_v]\)
  - Introduce \( v \) artificial operations with \( \lambda = 1, \ldots, v \) with \( \mu_k = M_k \) and:
    \[ p_\lambda = b_\lambda - a_\lambda \]
    \[ r_\lambda = a_\lambda \]
    \[ d_\lambda = b_\lambda \]

- **Minimum lateness objectives:**
  \[ L_{\max} = \max\limits_{j=1}^{N} (C_j - d_j) \Leftrightarrow d_{n_j,n_j+1} = p_{n_j} - d_j \]
Blocking

Arises with limited buffers: after processing, a job remains on the machine until the next machine is freed

- Needed a generalization of the disjunctive graph model
  \[ G = (N, E, A) \] [Mascis, Pacciarelli, 2002]

1. Two non-blocking operations to be processed on the same machine
   \[ S_i + p_i \leq S_j \quad \text{or} \quad S_i + p_i \leq S_i \]

2. Two blocking operations \( i, j \) to be processed on the same machine \( \mu(i) = \mu(j) \)
   \[ S_{MS(j)} \leq S_i \quad \text{or} \quad S_{MS(i)} \leq S_j \]

3. \( i \) is blocking, \( j \) is non-blocking (ideal) and \( i, j \) to be processed on the same machine \( \mu(i) = \mu(j) \).
   \[ S_i + p_i \leq S_j \quad \text{or} \quad S_{MS(j)} \leq S_i \]

- A complete selection \( S \) is consistent if it chooses alternatives from each pair such that the resulting graph does not contain positive cycles.

Example:
- \( O_0, O_1, \ldots, O_{13} \)
- \( M(O_1) = M(O_5) = M(O_9) \)
- \( M(O_2) = M(O_6) = M(O_{10}) \)
- \( M(O_3) = M(O_7) = M(O_{11}) \)

- Length of arcs can be negative
- Multiple occurrences possible: \( ((i, j), (u, v)) \in A \) and \( ((i, j), (h, k)) \in A \)
- The last operation of a job \( j \) is always non-blocking.

Example:
- \( p_a = 4 \)
- \( p_b = 2 \)
- \( p_c = 1 \)
- \( b \) must start at least 9 days after \( a \) has started
- \( c \) must start at least 8 days after \( b \) is finished
- \( c \) must finish within 16 days after \( a \) has started

\[ S_a + 9 \leq S_b \]
\[ S_b + 10 \leq S_c \]
\[ S_c - 15 \leq S_a \]

This leads to an absurd.
In the alternative graph the cycle is positive.
The Makespan still corresponds to the longest path in the graph with the arc selection $G(S)$.

If there are no cycles of length strictly positive it can still be computed efficiently in $O(|N| |E \cup A|)$ by Bellman-Ford (1958) algorithm.

The algorithm iteratively considers all edges in a certain order and updates an array of longest path lengths for each vertex. It stops if a loop over all edges does not yield any update or after $|N|$ iterations over all edges (in which case we know there is a positive cycle).

Possible to maintain incremental updates when changing the selection [Demetrescu Frangioni, Marchetti-Spaccamela, Nanni, 2000].

Heuristics for the Alternative Graph Model

- The search space is highly constrained + detecting positive cycles is costly
- Hence local search methods not very successful
- Rely on the construction paradigm
- Rollout algorithm [Meloni, Pacciarelli, Pranzo, 2004]

Rollout
- **Master process**: grows a partial selection $S^k$: decides the next element to fix based on an heuristic function (selects the one with minimal value)
- **Slave process**: evaluates heuristically the alternative choices. Completes the selection by keeping fixed what passed by the master process and fixing one alternative at a time.

Slave heuristics
- **Avoid Maximum Current Completion time**
  find an arc $(h, k)$ that if selected would increase most the length of the longest path in $G(S^k)$ and select its alternative
  $$\max_{(u,v) \in A} \{l(0, u) + a_{uv} + l(u, n)\}$$

- **Select Most Critical Pair**
  find the pair that, in the worst case, would increase least the length of the longest path in $G(S^k)$ and select the best alternative
  $$\max_{((i,j),(h,k)) \in A} \min\{l(0, u) + a_{hk} + l(k, n), l(0, i) + a_{ij} + l(j, n)\}$$

- **Select Max Sum Pair**
  finds the pair with greatest potential effect on the length of the longest path in $G(S^k)$ and select the best alternative
  $$\max_{((i,j),(h,k)) \in A} \{l(0, u) + a_{hk} + l(k, n) + l(0, i) + a_{ij} + l(j, n)\}$$

Trade off quality vs keeping feasibility
Results depend on the characteristics of the instance.
Outline

1. Job Shop Generalizations

2. Resource Constrained Project Scheduling Model

Resource Constrained Project Scheduling Model

Given:
- activities (jobs) \( j = 1, \ldots, n \)
- renewable resources \( i = 1, \ldots, m \)
- amount of resources available \( R_i \)
- processing times \( p_j \)
- amount of resource used \( r_{ij} \)
- precedence constraints \( j \rightarrow k \)

Further generalizations
- Time dependent resource profile \( R_i(t) \) given by \((t_{i}^{\mu}, R_{i}^{\mu})\) where \( 0 = t_{i}^{1} < t_{i}^{2} < \ldots < t_{i}^{m_{i}} = T \)
- Disjunctive resource, if \( R_k(t) = \{0, 1\} \); cumulative resource, otherwise
- Multiple modes for an activity \( j \)
  - processing time and use of resource depends on its mode \( m \): \( p_{jm}, r_{jkm} \).

Solutions

Task: Find a schedule indicating the starting time of each activity

- All solution methods restrict the search to feasible schedules, \( S, S' \)
- Types of schedules
  - Local left shift (LLS): \( S \rightarrow S' \) with \( S'_j < S_j \) and \( S'_l = S_l \) for all \( l \neq j \).
  - Global left shift (GLS): LLS passing through infeasible schedule
  - Semi active schedule: no LLS possible
  - Active schedule: no GLS possible
  - Non-delay schedule: no GLS and LLS possible even with preemption
- If regular objectives \( \Rightarrow \) exists an optimum which is active

Hence:
- Schedule not given by start times \( S_i \)
  - space too large \( O(T^n) \)
- difficult to check feasibility
- Sequence (list, permutation) of activities \( \pi = (j_1, \ldots, j_n) \)
- \( \pi \) determines the order of activities to be passed to a schedule generation scheme
Assignment 1

- A contractor has to complete \( n \) activities.
- The duration of activity \( j \) is \( p_j \).
- Each activity requires a crew of size \( W_j \).
- The activities are not subject to precedence constraints.
- The contractor has \( W \) workers at his disposal.
- His objective is to complete all \( n \) activities in minimum time.

Assignment 3

- In a basic high-school timetabling problem we are given \( m \) classes \( C_1, \ldots, C_m \).
- \( h \) teachers \( a_1, \ldots, a_h \) and
- \( T \) teaching periods \( t_1, \ldots, t_T \).
- Furthermore, we have lectures \( i = l_1, \ldots, l_n \).
- Associated with each lecture is a unique teacher and a unique class.
- A teacher \( a_i \) may be available only in certain teaching periods.
- The corresponding timetabling problem is to assign the lectures to the teaching periods such that
  - each class has at most one lecture in any time period
  - each teacher has at most one lecture in any time period
  - each teacher has only to teach in time periods where he is available.

Assignment 4

- A set of jobs \( J_1, \ldots, J_g \) are to be processed by auditors \( A_1, \ldots, A_m \).
- Job \( J_l \) consists of \( n_l \) tasks \((l = 1, \ldots, g)\).
- There are precedence constraints \( i_1 \rightarrow i_2 \) between tasks \( i_1, i_2 \) of the same job.
- Each job \( J_l \) has a release time \( r_l \), a due date \( d_l \) and a weight \( w_l \).
- Each task must be processed by exactly one auditor. If task \( i \) is processed by auditor \( A_k \), then its processing time is \( p_{i,k} \).
- Auditor \( A_k \) is available during disjoint time intervals \([s_k^\nu, t_k^\nu]\) \((\nu = 1, \ldots, m)\) with \( t_k^\nu < s_k^{\nu+1} \) for \( \nu = 1, \ldots, m_k − 1 \).
- Furthermore, the total working time of \( A_k \) is bounded from below by \( H_k^− \) and from above by \( H_k^+ \) \((k = 1, \ldots, m)\).
- We have to find an assignment \( \alpha(i) \) for each task \( i = 1, \ldots, n := \sum_{l=1}^{g} n_l \) to an auditor \( A_{\alpha(i)} \) such that
  - each task is processed without preemption in a time window of the assigned auditor
  - the total workload of \( A_k \) is bounded by \( H_k^− \) and \( H_k^+ \) for \( k = 1, \ldots, m \).
  - the precedence constraints are satisfied.
  - all tasks of \( J_l \) do not start before time \( r_l \).
  - the total weighted tardiness \( \sum_{l=1}^{g} w_l r_l \) is minimized.