Scheduling Problems

General Idea of the Problem

- Allocation of Resources
- Allocation of Time Slots
- Constraints
- Optimisation

Definition of the Problem

- $J = \{J_1, \ldots, J_n\}$
- $M = \{M_1, \ldots, M_m\}$
- Schedule – Mapping of jobs to machines and processing times
- The Schedule is subject to feasibility constraints and optimisation objectives.

Schedule Constraints

- Each machine can only process one job at a time.
- Each job can only be processed by one machine at any time.
- Once a machine has started processing a job, it will continue running on that job until the job is finished.

Classification of Problems

- Single-machine problems
- Multi-machine problems
- Single-stage problems
- Multi-stage problems

Other Concepts

- Processing time p_i
- Release dates r_i
- Due dates d_i
- Weights w_i
- Setup times t_{ij}
- Precedence constraints

Classification of Single Stage Multi-Machine Problems

- Parallel Machine Problems
  - Identical parallel machine problems
  - Uniform parallel machine problems
  - Unrelated parallel machine problems

Classification of Multi-Stage Multi-Machine Problems

- Flow Shop Problems
- Open Shop Problems
- Job Shop Problems
- Group Shop Environment

Definitions

- Completion Time C_i is earliest time at which J_i is completely processed.
- Lateness $L_i := C_i - d_i$
- Tardiness $T_i := \max\{C_i - d_i, 0\}$
- Earliness $E_i := \max\{d_i - C_i, 0\}$

Objective Functions

- Maximum Completion Time (makespan)
  - $C_{max} := \max\{C_1, \ldots, C_n\}$
- Sum of the (weighted) completion times
  - $\sum_{i=1}^{n} w_i * C_i$
- Total Weighted Tardiness
  - $\sum_{i=1}^{n} w_i * T_i$

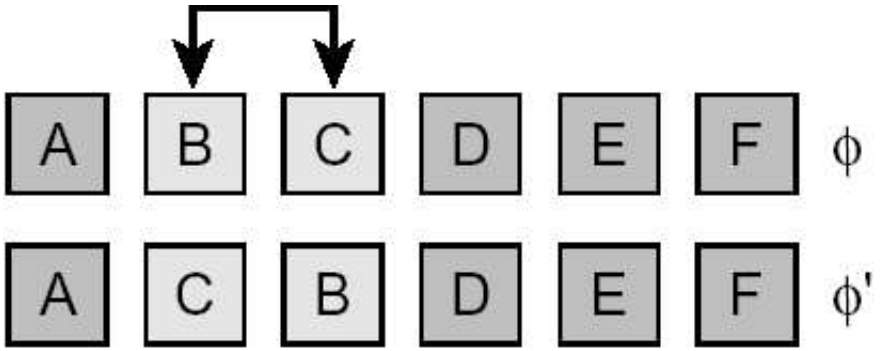
Candidate Solutions

- Permutations or sequences of jobs
  - Appropriate for many single-machine problems and flow shop problems.
- *M* sequences, one for each machine
  - Appropriate for parallel machine problems with release and due dates.

Neighbourhood Relations

- **Transpose neighbourhood** N_t: Two permutations ø, ø' are transpose neighbours if, and only if, one can be obtained from the other by swapping the positions of two adjacent jobs.

Transpose Neighbourhood

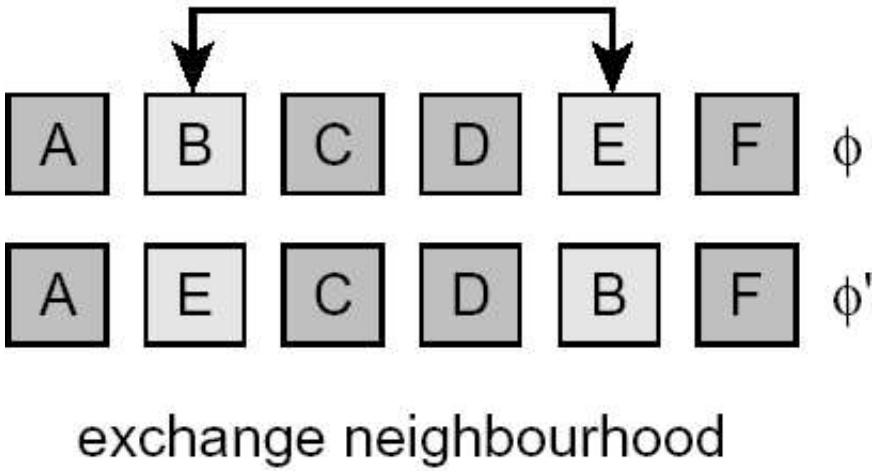


transpose neighbourhood

Neighbourhood Relations

- **Exchange neighbourhood** N_e: Two permutations ø, ø' are 2-exchange neighbours if, and only if, one can be obtained from the other by exchanging two jobs at arbitrary positions.

Exchange Neighbourhood

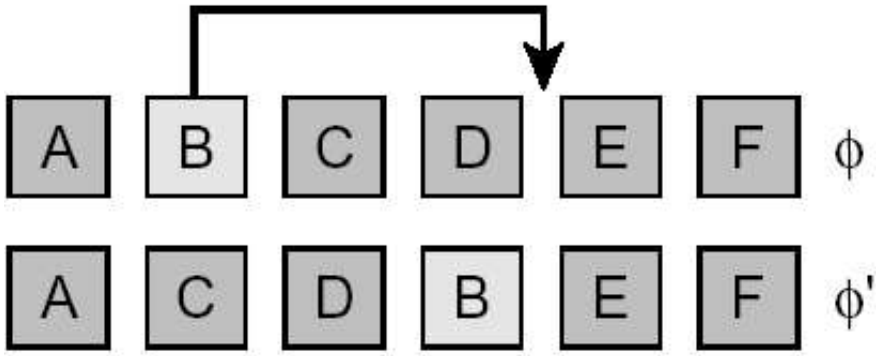


exchange neighbourhood

Neighbourhood Relations

- **Insertion neighbourhood** N_i: Two permutations ø, ø' are insertion neighbours if, and only if, one can be obtained from the other by removing a job from one position and inserting it at another position.

Insertion Neighbourhood



Single-Machine Maximum Lateness Problem

- Given
  - *n* jobs $J_1, \ldots, J_n$
  - *n* respective due dates $d_1, \ldots, d_n$
- Goal
  - Minimise maximum lateness
  - $L_{max} := \max\{L_1, \ldots, L_n\}$

Solution

- Sequence the jobs in non-decreasing order of their due dates.
- Also known as the *earliest due date (EDD)* rule.
- Runs in *O(n * log n)* time.
- Also minimses the maximal tardiness T_{max}

Single Machine Total Weighted Tardiness Problem

- Given
  - *n* jobs that have to be processed on a single machine
  - For each job J_i, a processing time p_i, a weight w_i and a due date d_i.
  - All jobs become available for processing at time zero.
- Goal
  - Find a schedule that that minimses the total weighted tardiness $\sum_{i=1}^{n} w_i * T_i$

Construction Heuristics

- **Earliest due date (EDD).** Jobs are sequenced in non-decreasing order of their due dates d_j.
- **Modified due date (MDD).** Sequence jobs in non-decreasing order of their modified due dates $mdd_j := \max\{C + p_j, d_j\}$, where C is the sum of the processing times that have already been sequenced.

Construction Heuristics cont.

- **Apparent urgency (AU).** Under this rule, jobs are sequenced in non-decreasing order of their apparent urgency $au_j := (w_j / p_j) * e^{-(\max\{dj - Cj, 0\}) / kÞ}$. Here, Þ denotes the average processing time of the remaining jobs, k is a parameter, and $C_i := C + p_i$.

Iterative Best Improvement

- Iterative best improvement methods based on N_e and N_i, $IBI(N_e)$ and $IBI(N_i)$.
- These can be combined in a 2-phase local search algorithm that either performs first $IBI(N_e)$ and then $IBI(N_i)$ or vice versa.

Performance Comparison

Construction Heuristic	*No Local Search*			$+IBI(N_e)$			$+IBI(N_i)$		
	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}
EDD	135	24	0.001	0.62	33	0.140	1.19	38	0.64
MDD	61	24	0.002	0.65	38	0.078	1.31	36	0.77
AU	21	21	0.008	0.92	28	0.040	0.56	42	0.26

Construction Heuristic	$+IBI(N_e) + IBI(N_i)$			$+IBI(N_i) + IBI(N_e)$		
	Δ_{avg}	n_{opt}	t_{avg}	Δ_{avg}	n_{opt}	t_{avg}
EDD	0.24	46	0.20	0.47	48	0.67
MDD	0.40	46	0.14	0.44	42	0.79
AU	0.59	46	0.10	0.21	49	0.27

An ACO Algorithm for the SMTWTP

- The ACS-BSD algorithm was developed by Besten, Stützle, and Dorigo.
- Pheromone values t_{ij} are associated with each assignment of a job J_j to a sequence position i.
- During the construction phase, each ant builds candidate solutions, by iteratively adding jobs that are not yet in the sequence.

ACS-BSD cont.

- The job to be appended next is based on the pheromone values t_{ij} and heuristic values n_{ij}.
- If TF > 0.3 then $n_{ij} := 1/au_j$ else $n_{ij} := 1/mdd_j$.
- Given a current partial sequence of length i -1, for position i the ant selects with probability q the best choice as indicated by the combination of pheromone trails and the heuristic information, while with probability 1 – q, it performs a probabilistically biased exploration.

ACS-BSD cont.

- ACS-BSD uses a candidate list, from which in each construction step the job to be added to the current partial sequence is chosen.
- The candidate list is built by scanning the current incumbent solution and adding all jobs that are found in the incumbent solution but not in the current partial sequence. This process is stopped when the candidate list has reached a maximum length.

ACS-BSD cont.

- ACS-BSD uses a heterogeneous colony of ants, where each of the two 2-phase local search algorithms is applied by one half of the ants.
- There are two forms of pheromone updates.

Iterated Dynasearch

- ILS-CPV developed by Congram, Potts and van de Velde.
- Dynasearch – performs complex iterative improvement steps that are assembled from a set of mutually independent, simple search steps.
- Simple steps are based on the standard exchange neighbourhood, N_e.

Iterated Dynasearch cont.

- Performs iterative best improvement in the neighbourhood consisting of the sequences reachable from the current candidate sequence by any set of independent improving exchange steps.
- ILS-CPV starts from an initial candidate solution generated by the AU construction heuristic.
- Uses dynasearch as its subsidiary local search procedure.

Iterated Dynasearch cont. Perturbation

- The perturbation phase consists of six random exchange steps.
- The resulting sequence is scanned and any adjacent non-late jobs that are not in EDD order are transposed.
- After 100 iterations of ILS, any adjacent non-late jobs in EDD order are transposed with probability 1/3, unless this would result in one of the jobs becoming late.

Iterated Dynasearch cont. Backtracking

- For l ILS iterations each new local minimum is accepted regardless of quality. This corresponds to a random walk phase of the iterated local search process.
- If in these l iterations the incumbent candidate solution, s, has not been improved the random walk phase starts again from s.

Iterated Dynasearch cont. Improvements

- An enhancement of ILS-CPV is in the dynasearch algorithm to use simple search steps based on the insertion neighbourhood N_i, as components for complex dynasearch steps.

Flow Shop Scheduling

- The order in which a job passes through the machines is the same for all jobs.
- All jobs are available at time zero.
- Each operation is to be performed on a specific machine.
- Each machine can process at most one job at a time.
- Each job can be processed by at most one machine at a time.

Flow Shop Scheduling - Buffers

- If the buffers are queues that operate on the first come – first serve principle the jobs pass through all machines in the same order. These are known as *permutation flow shop problems*.
- If changes in the sequence in which jobs are processed are allowed, the flow shop problem becomes much harder.

Permutation Flow Shop Problems (PFSP)

- Capacity of the buffer between machines is unlimited.
- The optimisation objective is to minimise the completion time of the last job.

Formal Definition

- A set of m machines $M_1, \ldots, M_m$
- A set of n jobs $J_1, \ldots, J_n$ where each job consists of m operations $o_{i1}, \ldots, o_{im}$ that have to be performed on machines $M_1, \ldots, M_m$ in that order, with processing time p_{ij} for operation o_{ij}.
- The objective is to find a sequence that minimises the makespan.

Iterated Local Search for the PFSP

- ILS-S-PFSP initializes the search with the NEH heuristic.
- The NEH is an insertion heuristic
- It first computes for each job J_i the sum p_i of the processing times of its operations, and then orders the jobs according to non-increasing p_i.
- Then the jobs are considered in this order for inclusion into a partial sequence; at each step the next job is inserted into the position where it leads to a minimum increase in the makespan.

Iterated Local Search for the PFSP cont.

- ILS-S-PFSP uses a modified iterative first improvement as its local search.
- First a random permutation of the job indices is generated.
- Then, in each step the next index *i* from this permutation is selected and all possibilities for inserting job ø(i) are examined.
- When a an improving sequence is found job ø(i) is inserted at the sequence position that leads to the maximal reduction in makespan.

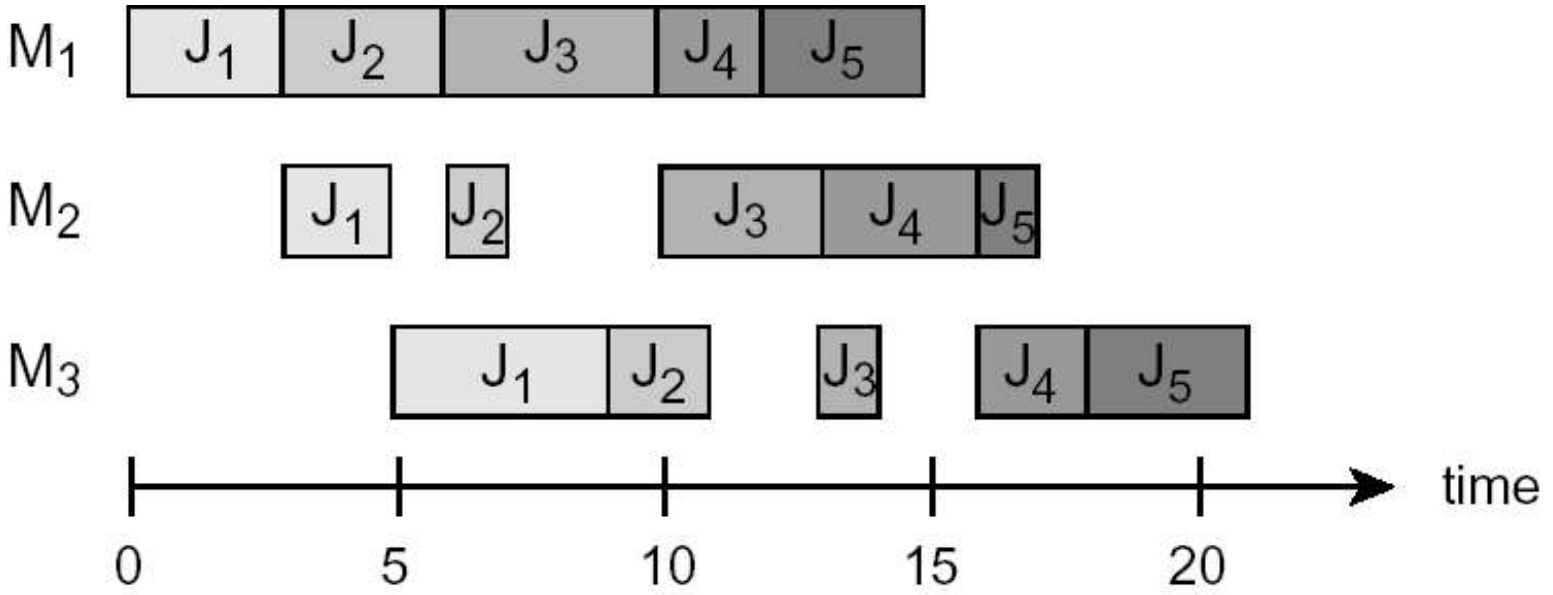
Iterated Local Search for the PFSP cont. Perturbations and Acceptance Criterion

- The perturbation for ILS-S-PFSP consists of two random steps in N_t, followed by one random step in N_e with the additional restriction that $|i-j| <= max\{n/5, 30\}$, where i and j are the positions of the two jobs exchanged in the N_e step, and n is the number of jobs.
- The acceptance criterion used is based on the Metropolis condition with a constant temperature value T := p/15, where p is the average processing time over all operations.

Critical Paths

- In a schedule for the PFSP, some operations cannot be delayed without increasing the makespan of a sequence. Operations for which this is true are called critical, while all other operations are called slack.

Critical Paths



Motivation for Critical Blocks

- Consider a critical path in a candidate sequence $\varnothing$ with K critical blocks. Let $m(B_i)$ be the machine associated with the critical block B_i and let I_i be the set of jobs that are internal to critical block B_i $(1 <= i <= K)$. If $m(B_1) = M_1$, include job $\varnothing(1)$ in I_1, and if $m(B_K) = M_m$, include job $\varnothing(n)$ in I_K. Then no reordering of the jobs in any of the sets I_i can improve the makespan of $\varnothing$.

Tabu Search for PFSP

- TS-NS-PFSP is based on the insertion neighbourhood N_i, but it makes strong use of the block properties in order to restrict the size of the effectively searched local neighbourhoods. The resulting neighbourhood relation is N_i^{NS}.
- Uses the NEH heuristic for generating an initial sequence.
- In each search step, a best-improving non-tabu neighbour of the current sequence is selected. Also uses a complex aspiration criterion.

Tabu Search for PFSP cont. Tabu Status

- If a neighbouring candidate solution is obtained from $ø$ by removing a job at position i and inserting it at position $j > i$, the pair $(ø(i), ø(i+1))$ is added to the tabu list; if $i < j$, then the pair $(ø(i+1), ø(i))$ is added.
- Given a candidate solution $ø'$, a search step that moves a job from position k to position $l > k$ is declared tabu if, and only if, at least one pair of jobs $(ø'(j), ø'(k)), j = k+1, \ldots, l$ is in the tabu list, and vice versa for $l < k$.

Tabu Search for PFSP cont. Local Search

- Builds a set *RM*. The elements of *RM* are the best neighbouring sequences in N_i^{NS} for each job *ø(i)*, $i = 1, \ldots, n$.
- Next a set *RM'* is built which consists of all the neighbours that are not tabu or whose tabu status is overridden by the aspiration criterion.
- Then TS-NS-PFSP selects the best sequence in *RM'* to replace the current candidate solution.

Tabu Search for PFSP cont. Aspiration Criterion

- A schedule $\varnothing'$ is aspired if, and only if, it improves over the best makespan obtained in any iteration directly preceeding or directly succeeding an iteration in which the same makespan as that of the current schedule was achieved.

Tabu Search for PFSP cont. Backtracking

- Each time the incumbent candidate solution is improved, TS-NS-PFSP stores the tabu list *TL* associated with the new incumbent solution as well as its set of admissible neighbours.
- The backtracking mechanism is triggered when for a given number of iterations after search initialisation or after the most recent backtracking step no improvement in the incumbent solution has been found.

Group Shop Problems - Definition

- Set of machines $M := \{M_1, \ldots, M_m\}$
- Set of jobs $J := \{J_1, \ldots, J_n\}$, where each job J_i consists of m operations $o_{i1}, \ldots, o_{im}$ and a processing time for p_{ij} for each operation o_{ij}.
- The operations of each job do not have to be performed in the canonical order $o_{i1}, \ldots, o_{im}$.

Group Shop Problems – Definition cont.

- Precedence constraints are given by a means of a partition of the set $\{o_{i1}, \ldots, o_{im}\}$ in the form of a set of groups $Gi = \{g_{i1}, \ldots, g_{il(i)}\}$ and a canonical total order $g_{i1} < g_{i2} < \ldots < g_{il(i)}$ between those groups.
- All operations of the first group have to be processed before all operations of the second group, etc., but the order in which the operations within the same group are performed is not restricted.

Special Cases of the GSP

- *Job Shop Problem (JSP)* – There exists a total order of all the operations belonging to each job. Each group comprises only one single operation.
- *Open Shop Problem (OSP)* – There are no precedence relations among the operations of each job. For each job all of its operations form one group.
- All of these problems are known to be *NP-hard* with a few special cases having polynomial time solutions.

Tabu Search for Job Shop Scheduling

- Also known as the TS-NS-JSP algorithm.
- Similar to TS-NS-PFSP
- Main ingredients
  - Underlying neighbourhood relation
  - Pruning mechanism for effectively searching this neighbourhood
  - Makes use of critical paths

Tabu Search for Job Shop Scheduling cont. Neighbourhood Relation

- The TS-NS-JSP is based on the transpose neighbourhood N_t restricted to operations that are on the same machine.
- This neighbourhood relation N_t^{NS} exploits the following observations.
  - Reversing the order of two operations on a critical path in a candidate schedule *s* never results in an infeasible candidate solution.
  - Reversing the order of two operations that are not on a critical path cannot lead to a reduction of the makespan.

Tabu Search for Job Shop Scheduling cont. Neighbourhood Relation

- Reversing the order of two internal operations in a block of a critical path cannot lead to a reduction of the makespan.
- Reversing the order of the first two operations on the first machine block or the last two operations on the last block cannot reduce the makespan.

- Based on these observations $N_t^{NS}(s)$ comprises only schedules that are obtained by transpositions of the first two operations and the last two operations of any block except for the first and the last block.

Tabu Search for Job Shop Scheduling cont. Tabu Status

- Tabu status is associated with individual search steps in N_t^{NS} such that after performing a search step, the reverse step is considered tabu for the next *tt* search steps.

Tabu Search for Job Shop Scheduling cont. Search Steps

- Each search step is selected by first generating a critical path and then choosing the best search step from the local N_t^{NS} neighbourhood that is either not tabu or whose tabu status is overridden by an aspiration criterion.
- TS-NS-JSP uses an aspiration criterion that overrides the tabu status of a search step if it leads to an improvement over the best makespan encountered so far.

Tabu Search for Job Shop Scheduling cont. Additional Mechanisms

- Occasionally restore the search process to a previously stored, highly promising candidate solution, also called an *elite solution*. This is triggered whenever a fixed number of iterations has been performed without achieving an improvement in the incumbent solution.
- A cycle detection mechanism stops the local search if it is deemed to be stagnating.