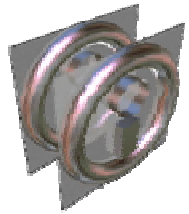
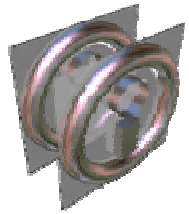
Introduction to production scheduling



Scheduling

- Definition
  - Scheduling deals with the efficient allocation of tasks over resources
  - The general scheduling problem is, given a number of tasks and a number of resources, to establish the dates when each task should be accomplished on each resource

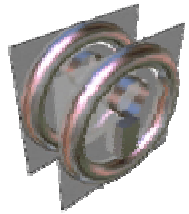
- Some real-life example:
  - Scheduling of the PC room at Swansea University: a number of different courses (tasks) have to be given using the PC room (resource)
    - The efficient use of the PC room is mandatory, otherwise the University will need a second PC room and the fees will increase!



Production Scheduling (I)

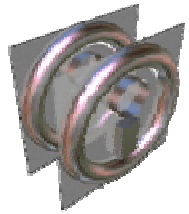
- The MRP tell us the quantities of products to manufacture in every time bucket
  - However, MRP does not make any assumption about the resources (i.e. labour, machines) currently available in the factory
  - E.g. two different components have to be manufactured in the same section. How to schedule them?

- In production scheduling, we have a number of jobs (J_i) to be manufactured over a number of machines (M_j)
  - The production scheduling problem deals with obtaining the date for each job to enter on each machine
  - Not necessarily physical machines, they may be stages (consisting on several machines or labour) in a manufacturing process



Production scheduling (II)

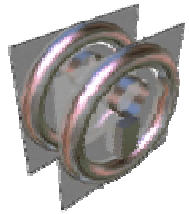
- Jobs have to be manufactured in each machine in a certain order (known as job sequence) during a certain time period (known as processing time)
  - Processing time of job J_i in machine M_j is usually denoted by $t_{i,j}$
- Both order and time period are given by the technological process



Example: computer assembly (I)

- Tomcat Ltd. is a company that assembles computers. Three main steps can be distinguished in this process:
  - Motherboard & microprocessor are installed
  - Peripheral devices are plugged in the motherboard
    - The number and type of devices that have been order by each customer
  - The computer (all its components) are tested
    - This process depends on the number and type of components

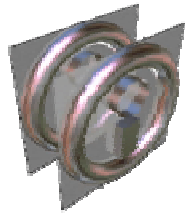
- The order of the steps is given by the technological process and does not depend on the specific order



Example: computer assembly (II)

- The plant in which Tomcat Ltd. assembles the computers is organised in three sections, according to the three main steps:
  - Section 1: Motherboard & microprocessor
  - Section 2: Peripheral devices
  - Section 3: Computer test

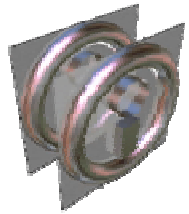
- On each section, a worker is performing the operation
  - Obviously, new orders cannot start until the worker completes the current order
  - Let us assume that an order cannot overtake another order, i.e. the job sequence is the same for all steps



Example: Computer assembly (III)

- Let us assume that we have three orders (computers to manufacture).
  - According to the nature of each order (components, type, etc.), we can have some estimate of the average times (minutes) for each step:

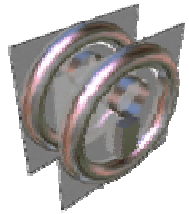
	Motherboard Section	**Plug Devices Section**	**Test Computer Section**
Order #1	2	15	7
Order #2	3	10	5
Order #3	2	12	5



Question #1

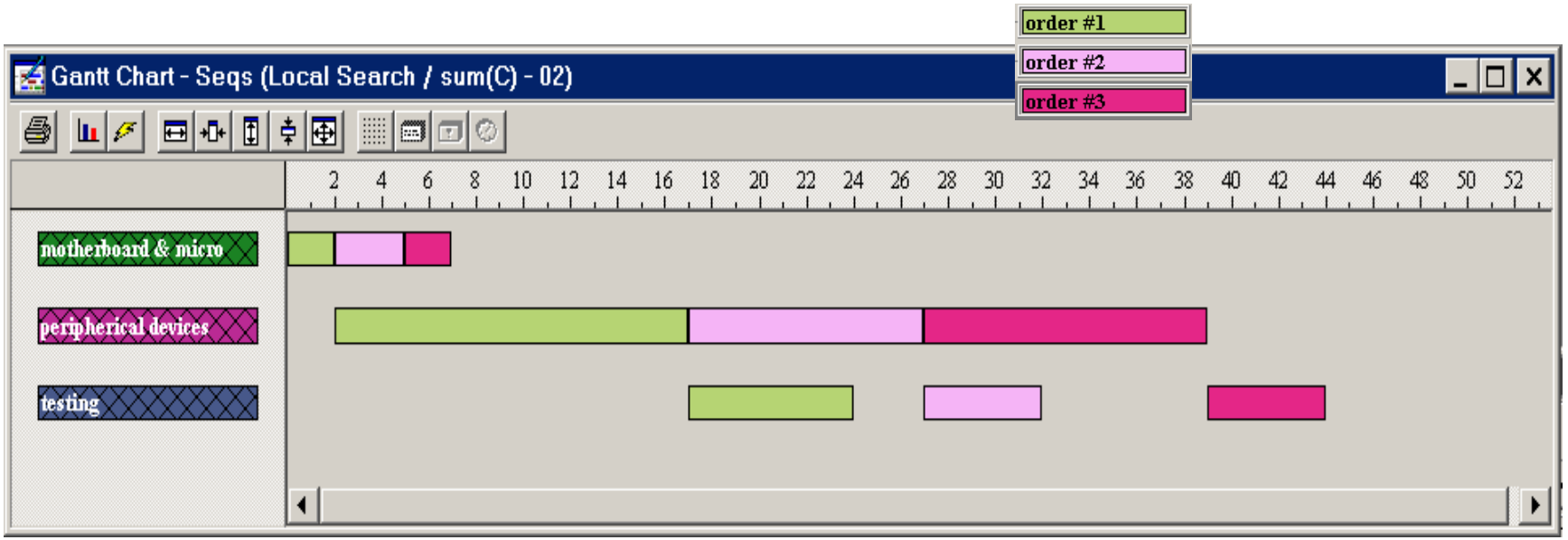
- The objective of the company is to keep the average time to assembly a computer as lowest as possible
  - The aim is to leverage the assembly time regardless the specific type of computer

- Which of the following sequences is the most convenient for the objective of the company:
  - #1, #2, #3?
  - #3, #2, #1?
  - Or is it an irrelevant problem (i.e. does the average time to assembly the computers depend on the sequences?)



Gantt chart

- A representation of an specific solution of a scheduling problem in terms of the machines and jobs

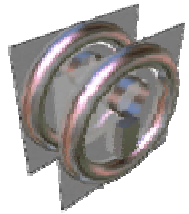


Completion times

- Completion time: $C_{i,j}$
  - Time for which job J_i finishes its processing on machine M_j
- Completion time: C_i
  - Time for which job J_i finishes its processing in the last machine

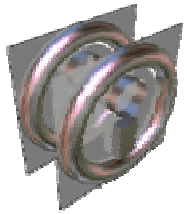
- Question #1 revisited:
  - It is easy to see that, for the case under consideration, the following holds: $C_{[k],j} = \max(C_{[k-1],j} ; C_{[k],j-1}) + t_{[k],j}$

assuming $C_{[k],0} = 0$, and $C_{[0],j} = 0$



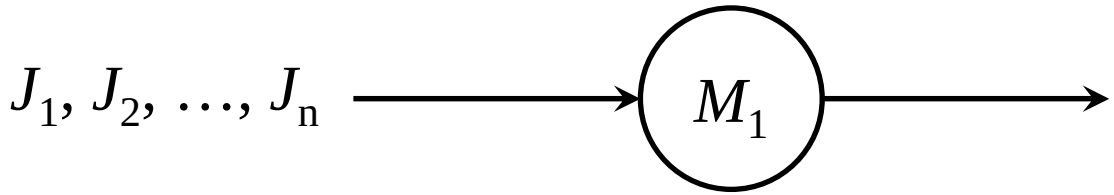
Scheduling environment (I)

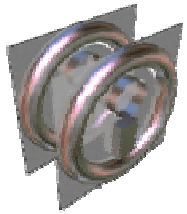
- The environment or framework of a scheduling problem refers to the way the jobs must visit the machines:
  - Single machine
    - Interesting case: bottleneck process: the important issue is scheduling jobs in the bottleneck
  - Flow shops
    - All jobs have the same routing
    - Additionally, most of the times it is consider than the sequence is the same for all machines
  - Job shops
    - Each job has a different route
    - It is one of the most complex case
  - Others (Parallel machines, open shops, etc)



Scheduling environment (I)

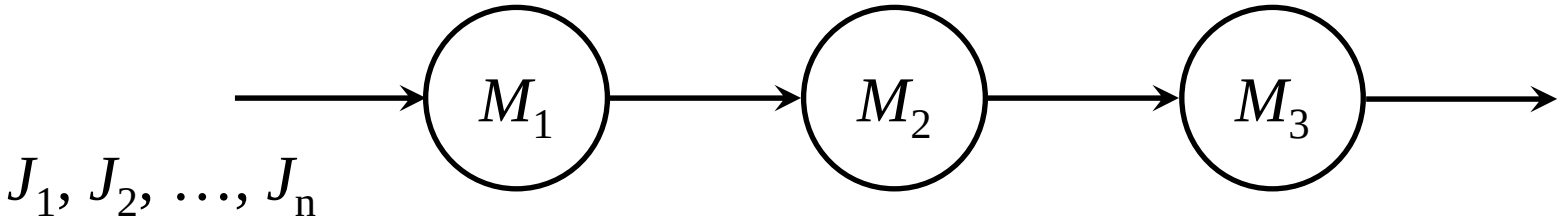
- The environment or framework of a scheduling problem refers to the way the jobs must visit the machines:
  - Single machine
    - Interesting case: bottleneck process: the important issue is scheduling jobs in the bottleneck
    - One may reduce the different steps (sections) in the plant to a single machine



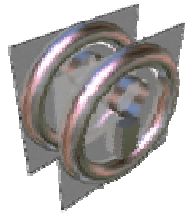


Scheduling environment (II)

- Flow shops
  - All jobs have the same routing
    - E.g. in Tomcat Ltd

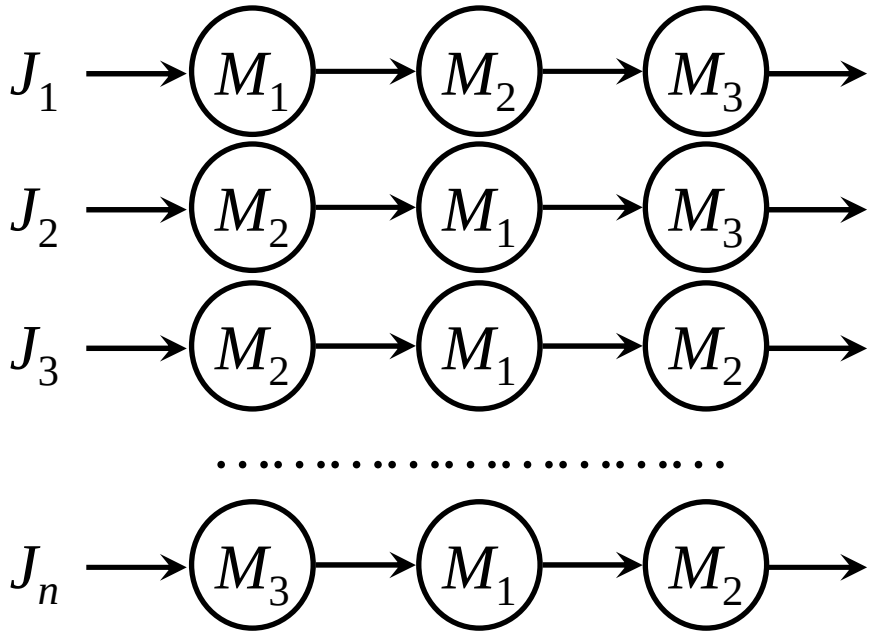


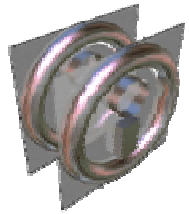
  - Additionally, most of the times it is consider than the sequence is the same for all machines
    - Permutation flow shop



Scheduling environment (III)

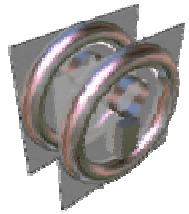
- Job shops
  - Each job has, in general, a different route
  - It is one of the most complex case





Scheduling environment (IV)

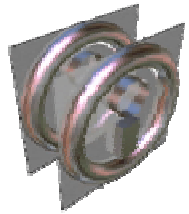
- There may be other (more complex) environments
  - E.g. parallel machines, open shops, etc
- The environment largely determines the difficulty of the scheduling problem
  - Most cases are special cases of the job shop problem
- There may be additional constraints for the scheduling problem:
  - No-wait constraints
  - Pre-emption
  - Precedence constraints
  - ....



Scheduling objectives

- A scheduling objective is a measure to evaluate the quality of certain schedule
  - In real-life situations, there are many (sometimes conflicting) objectives

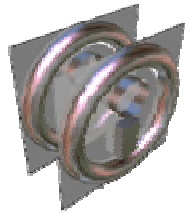
- In general, one can distinguish two types of objectives:
  - Due date related objectives
  - Non-due date related objectives



Due date related objectives (I)

- For this kind of problems, we assume that each job J_i has, in general, a due date d_i and a release date r_i
  - The due date represents the commitment of the company with a customer
  - The release date implies the non availability of raw materials from the beginning

- When each job has a due date, a basic objective is fulfilling this due date
  - Indicator of the service level
  - However, finishing the order as soon as possible (much before the due date) is not a good idea
    - Inventory costs

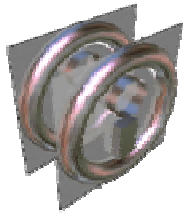


Due date related objectives (II)

- Lateness of a job: $L_i = C_i - d_i$
  - Maximum lateness: $L_{max} = max(L_i)$
    - Serves to bound the maximum period after the due date
  - Average (total) lateness: $\underline{L} = \Sigma L_i / n$
    - Not very useful since L_i can yield negative values

- Tardiness of a job: $T_i = max(0, L_i)$
  - Maximum tardiness $T_{max} = max(T_i)$
  - Average (total) lateness: $\underline{T} = \Sigma T_i / n$

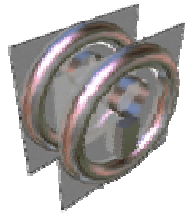
- Earliness of a job: $E_i = max(0, -L_i)$
  - Maximum earliness $E_{max} = max(E_i)$
  - Average (total) earliness: $\underline{E} = \Sigma E_i / n$



Due date related objectives (III)

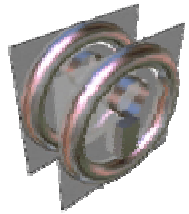
- Rather often, not all jobs (customers) are equally important.
  - Therefore, one can assign a weight w_i to each job representing the relative importance of each job

- Some measures that take into account the different weight of the jobs are:
  - Weighted lateness: $wL = \Sigma w_i L_i$
  - Weighted tardiness: $wT = \Sigma w_i T_i$
  - Weighted earliness: $wE = \Sigma w_i E_i$



Non due date objectives

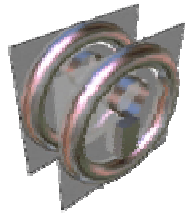
- Average lead time
  - Minimise average (total) completion times: ΣC_i
- Machine utilisation
  - Minimise idle time
    - Time after finishing one job and before starting the next one
  - Minimise makespan
    - $C_{max} = max(C_i)$
    - It can be shown that minimising makespan is equivalent to minimising idle time



Example: Shipyard

- Let us assume that we have three orders (vessels to build and their corresponding due dates).
  - According to the nature of each vessel, we can have some estimate of the average times (months) for each step:

	Step #1	**Step #2**	**Step #3**	**Due dates**
Order #1	2	6	10	30
Order #2	15	10	5	20
Order #3	4	12	8	28

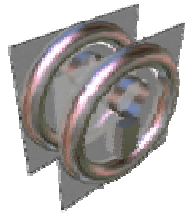


Question #2

- Evaluate the following sequences:
  - #1, #2, #3
  - #3, #1, #2

  with respect to the following objectives:
    - Makespan
    - Minimise total tardiness
    - Minimise maximum tardiness
    - Minimise total earliness
    - Minimise maximum earliness

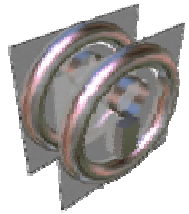
- Are these conflicting objectives?



Scheduling is hard (I)

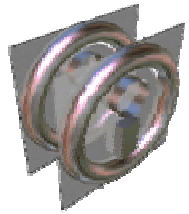
- Except for few simple cases, there is not an easy way to obtain the optimal solution
  - Up to the 70's, scheduling researchers looked for procedures to get the optimal solution of several scheduling problems, usually implemented as computer programs
  - What it was found is that the computation times of these procedures grow very fast with the problem size
    - E.g. the computation time required to solve a scheduling problem on a single machine with 10 jobs was much higher than the double of the time to solve the problem with 5 jobs.
  - These procedures were said to be **NP** (Non Polynomial)

- What is the problem with NP procedures?



Question #3: Example of (silly) NP procedure

- Obviously, a (silly) procedure to obtain the optimal solution of scheduling *n* jobs on a single machine (using any criterion) is to evaluate all possible solutions
- Assume a computer that can evaluate one schedule in 1 sec. Obtain the times required to evaluate all schedules when there are 5, 10, 20, 25 and 50 jobs
- Obtain these times assuming now a computer that can evaluate a schedule in 10^{-10} secs.
- Obtain these times assuming now a computer that can evaluate a schedule in 10^{-20} secs.

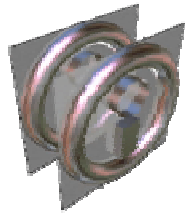


Scheduling is hard (II)

- The problem with NP procedures is that there is always a frontier for which the problems remain (optimally) unsolvable

- Question #3 revisited:
  - One may have a computer able to schedule 50 jobs in a factory in 1 sec.
  - Only if two additional jobs are coming, that computer would take more than 44 minutes to get the solution

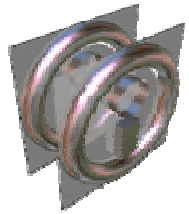
- The opposite of a NP procedure is a P (Polynomial) procedure



Scheduling is hard (III)

- In the 70's, a branch of Mathematics named Complexity Theory was established
  - By using this theory, one can proof whether it exists a P procedure for a given problem, or not.

- Along the decade it was shown that, for most of the scheduling problems, it does not exist a P procedure to solve them
  - For most of the cases, one must try different solutions and get the best one. However, there are two main difficulties for this:
    - The scheduling decision has to be taken in a relatively short period of time (at least as compared to the processing times of the machines)
    - The number of possible solutions may be very big

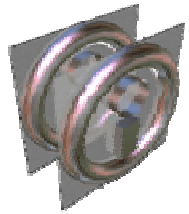


Dispatching rules and approximate procedures

- Since most of the scheduling problems cannot be optimally solved in a reasonable time, the only thing to do is use dispatching rules or approximate solutions

- Dispatching rules
  - Rules of thumb to schedule jobs

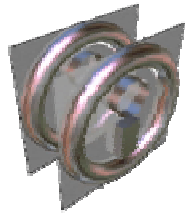
- Approximate procedures
  - Algorithms to improve a given solution (usually obtained from a dispatching rule)



Dispatching rules (I)

- Dispatching rules
  - Rules of thumb to schedule jobs

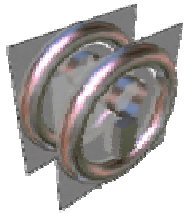
- Some well-known dispatching rules:
  - EDD – Earliest Due Date
    - Sort the jobs in ascending order of their due dates
  - SPT – Shortest Processing Time
    - Sort the jobs in ascending order of their sum of processing times
  - LPT – Longest Processing Time
    - Sort the jobs in descending order of their sum of processing times
  - FCFS (FIFO) – First Come First Served (First In First Out)
    - Sort the jobs in ascending order of their release date



Question #4: Thinking on dispatching rules

- Think on one scheduling problem for which EDD may yield good solutions
  - Hint: is a single-machine scheduling problem

- Build an instance of this problem using LEKIN and compare the results with the results obtained by the other dispatching rules

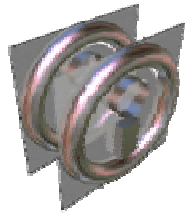


Approximate procedures (I)

- Approximate procedures employ an initial sequence *S* to look for another (better) sequences in what it is called the **neighbourhood** of *S*

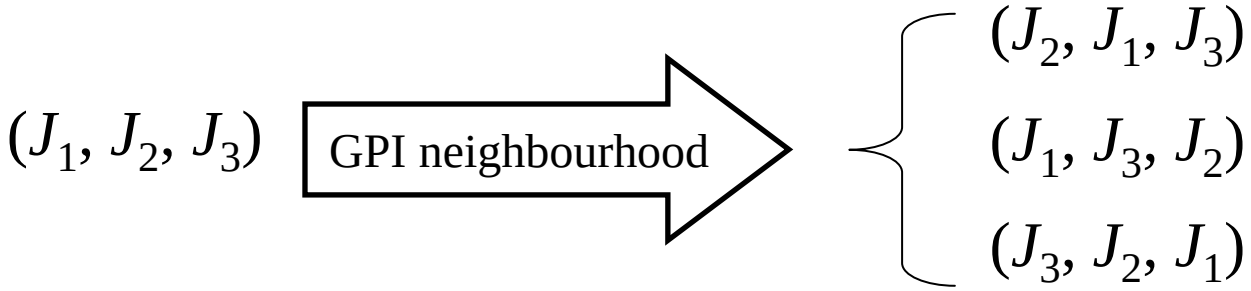
- What is a neighbourhood of a sequence?
  - A consistent way to generate new sequences by changing one (or more) jobs in the current schedule
  - Adjacent Pairwise Interchange (API)
    - Exchange the position of all adjacent jobs

(J_1, J_2, J_3) → API neighbourhood → (J_2, J_1, J_3), (J_1, J_3, J_2)

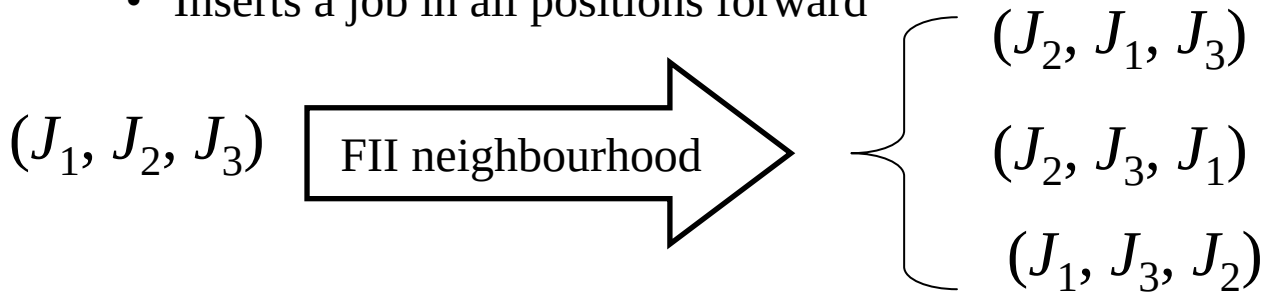


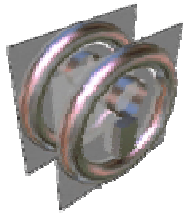
Approximate procedures (II)

- Other useful neighbourhoods:
  - General Pairwise Interchange (GPI):
    - Exchange the position of all adjacent jobs



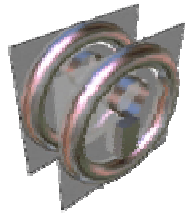
  - Forward Insertion Interchange (FII)
    - Inserts a job in all positions forward





Example of approximate sols.: Greedy Search

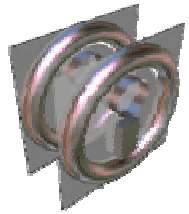
- Step 1: Obtain an initial solution. This is set as the current best solution
- Step 2: Use a neighbourhood to generate new solutions from the current best solution
- Step 3: Take the best one from the solutions generated
- Step 4: If the solution obtained from Step 3 is better than the current best solution, replace it as the new current best solution and go to Step 2.
- Step 5: Stop and print the current best solution



Question #5

- Apply greedy search (adjacent interchange) to the following flow shop scheduling problem with makespan objective:

	M_1	M_2	M_3
J_1	3	8	1
J_2	4	2	5
J_3	3	7	5
J_4	8	7	6
J_5	3	2	1
J_6	2	10	2



Set ups

- For some real cases, resources (machines) cannot process one job after another immediately (set up time)
- From a scheduling viewpoint, there are two types of set up operations
- Sequence-independent setup:
  - E.g. each 100 operations, the machine has to be inspected
  - This are not relevant for scheduling
- Sequence-dependent setup:
  - E.g. each time a machine changes from processing job type A to job type B, it requires 2 min. for tool changing
  - This type of setup is extremely important for scheduling