Design of Scheduling Systems

Contents
1. Architecture of Scheduling Systems
2. Databases, Object Bases, Knowledge Bases
3. Schedule Generation
4. User Interface
5. Generic Systems versus Application-Specific Systems

Architecture of Scheduling Systems

Modules:
1. Databases, Object Bases, Knowledge Bases
2. Schedule Generation
3. User Interface

Databases, Objects, Knowledge Bases

Data Concerning:
- Jobs (Operations)
- Work Centers (Machines)
- Schedules

Design the system properly in order to make transfer of data easy.

Example: Plugging in new algorithm in existing system should be easy.

Databases

Database management subsystem
- custom-made
- commercial

Basic functions include:
- multiple editing
- sorting
- search routines

SQL command:
SELECT fields to be displayed FROM tables containing the fields WHERE conditions hold

Two types of data

Static: do not depend on the schedule
- ordered product quantity
- due date
- release date
- weights of the jobs
- machine data (machine speeds, scheduled maintenance times)
- job and machine dependent (setup time between jobs $j$ and $k$ on machine $i$)

Dynamic: schedule dependent
- starting and completion times of the jobs
- idle times of the machines
- times that a machine is undergoing setups
- sequences in which the jobs are processed on the machines
- number of jobs that are late
- tardiness of the late jobs

Literature:
- Operations Scheduling with Applications in Manufacturing and Services, Michael Pinedo and Xiuli Chao, McGraw Hill, 2000 Chapter 10
Setup times
- static - kept in the table look-up
- dynamic (through formula)

Calendar
- factory holidays
- scheduled machine maintenance
- number of shifts

Data base record for job data type

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>Quantity</th>
<th>Priority</th>
<th>Ready</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IBM</td>
<td>4</td>
<td>160</td>
<td>2</td>
<td>10</td>
<td>200</td>
</tr>
</tbody>
</table>

Job instance

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Name</th>
<th>Type</th>
<th>Quantity</th>
<th>Priority</th>
<th>Ready</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IBM</td>
<td>4</td>
<td>160</td>
<td>2</td>
<td>10</td>
<td>200</td>
</tr>
</tbody>
</table>

Object - instance of Job

create()
copy()
delete()
display()

Class Job

Objects

Knowledge Base

- IF-THEN Rules
  - IF condition THEN action
  - Example
    - IF a job's processing time is longer than a given value
    - THEN the job may be assigned to a fast machine

Certainty factors
- Activity takes 5 days with CF = 0.8

Fuzzy facts
- Activity will take about 5 days

Fuzzy rules
- IF the cement is of good quality
  - THEN activity takes about 5 days.

Schedule Generation

1. Algorithmic approach
   - mathematical formulation: includes objectives and constraints
   - Three phases:
     - preprocessing
     - algorithm
     - postprocessing

2. Knowledge based approach
   - concerned with underlying problem structures that cannot be described in an analytical format
   - Forward chaining
   - Backward chaining
   - Relaxation of constraints

Algorithmic versus knowledge based approach

Algorithmic approach appropriate when:
- the problem allows for a precise mathematical formulation
- there is a little amount of randomness in the environment
- there is a need for optimisation, frequently and in real time
- there are not many exceptions

Not appropriate
- in dynamic environment
- even for moderately big problems can be computationally infeasible

Knowledge based approach appropriate when
- feasible schedules are required

Disadvantage:
- time consuming
**User Interface**

- Interface for the plant or enterprise
  - plant layout interface
  - resource calendar interface
  - routing table interface
- Interface for the schedule generation module
- Interface for the schedule manipulation
  - Gantt chart interface
  - Dispatch list
  - Capacity buckets
  - Input-output diagram

**Gantt chart**

- cascading and propagation effect
- not appropriate for large number of jobs and machines

**Dispatch list**

- does not give a good view of the schedule relative to time

**Capacity bucket**

- shows numerically for each time segment
  - the utilisation of each machine
- good for medium or long-term planning

**Input-output diagram (throughput diagram)**

- Shows cumulatively over time:
  - the total amount of orders received
  - the total amount produced
  - the total amount shipped
- Separate windows display the values of performance measures.

**User interface should allow:**

- Interactive optimization
  - Freezing jobs and reoptimization
  - Creating new schedules by combining different parts from different schedules
- Cascading and propagation effects
  - After a change by the user, the system
    - does feasibility analysis
    - takes care of cascading and propagation effects
    - does internal reoptimization
Generic Systems Versus Application-Specific Systems

**Generic Systems**
- Customisation of general systems may require a lot of programming.
- If the generic system is modular the customisation is easier.
- Usually contain less specific components: interface, data management, standard scheduling algorithms.
- Development is less expensive.
- Existing maintenance

**Application-specific systems**
- When the size of the scheduling problem is too large for a PC
- Cover peculiarity of scheduling environment.
- Own maintenance.
- Schedule manipulation is easier and faster.

Summary
- Gap between the scheduling theory and practice still exists.
- Scheduling systems reduce the time of the scheduling, and produce better schedules.