Dynamic Scheduling

Hana Rudová
http://www.fi.muni.cz/~hanka

Faculty of Informatics, Masaryk University
Brno, Czech Republic

CoreGRID Institute on Resource Management and Scheduling

Istituto di Scienza e Tecnologie dell’Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy
10th June, 2008
Dynamic scheduling

1. Problem description
2. Solving methods
   - Dispatching rules and policies
   - Local search
   - Branch & bound
   - Iterative forward search
3. Applications
   - Scheduling in Grid environment
   - University course timetabling
Static scheduling

Scheduling

- resources/machines vs. activities/jobs
- optimal resource allocation of activities over the time

Classical scheduling = static scheduling

- all resources and all activities are given
- no uncertainty in the behavior of resources and activities
Problem change during the solving process

Examples:

unscheduled resource breakdown
new jobs appear during solving phase
shorter/longer processing time than expected
Dynamic scheduling

- Problem change during the solving process
- Examples:
  - unscheduled resource breakdown
  - new jobs appear during solving phase
  - shorter/longer processing time than expected
Minimal perturbation problem

- New solution with minimal changes/perturbation from original solution
  - solution published
  - changes undesirable wrt. user
  - try to avoid avalanche effect
- Examples: school timetable, flight assignment, ...
Minimal perturbation problem

- New solution with minimal changes/perturbation from original solution
  - solution published
  - changes undesirable wrt. user
  - try to avoid avalanche effect
- Examples: school timetable, flight assignment, ...

- Minimal perturbation problem: \((P, \alpha, C^+, C^-, \delta)\)
  - \(P\) is a problem
  - \(\alpha\) solution of \(P\)
  - \(C^+, C^-\) sets of new and removed constraints
  - \(\delta\) distance function over two solutions

- Solution of minimal perturbation problem
  - solution \(\beta\) of the problem with \(C \setminus C^- \cup C^+\)
    with minimal distance \(\delta(\beta, \alpha)\)
Solving: dynamic problem vs. minimal perturbation problem

Dynamic problem

1. Problem definition
2. Scheduler
3. Solution
4. Objective function

Minimal perturbation problem

1. Problem definition
2. Scheduler
3. Solution
4. Objective function
5. Distance function
1 Problem description

2 Solving methods
   - Dispatching rules and policies
   - Local search
   - Branch & bound
   - Iterative forward search

3 Applications
   - Scheduling in Grid environment
   - University course timetabling
Dispatching rules and policies

Dispatching rules (static problems)
- new job assigned to the resource wrt. rule
- rule = longest job/job with smallest due date/... earlier
- combination of several rules

\[\begin{array}{c}
1 \\
3 \\
6 \\
8 \\
9 \\
\end{array} \]

\[\begin{array}{c}
\uparrow \\
4 \\
\end{array} \]
Dispatching rules and policies

Dispatching rules (static problems)
- new job assigned to the resource wrt. rule
- rule = longest job/job with smallest due date/... earlier
- combination of several rules

```
1 3 6 8 9

```

Policies (dynamic problems)
- again: new job assigned to the resource wrt. policy
- dynamics: new information concerning jobs and machines becomes available continuously
Dispatching rules and policies

Optimal for specific problems
- Earliest Due Date (EDD) first rule optimal for $1||L_{\text{max}}$

Good for problems
- high dynamics
  - subsequently appearing jobs
  - uncertain processing time
  - resource breakdowns
- the need for a prompt response

Not helpful for (complex) optimization
- minimal perturbation problem
- Quality of Service (QoS) requirements
Complex policy

Place jobs to the **Earliest Gap** in a machine where it fits
Complex policy

Place jobs to the **Earliest Gap** in a machine where it fits.

[Diagram showing resource allocation over time with labels M1, M2, M3, M4 indicating machines and time represented on the x-axis.]
Place jobs to the **Earliest Gap** in a *random machine* where it fits.
Place jobs to the **Earliest Gap** in the **best machine** where it fits.
Local search (LS) in scheduling

- Initial schedule: complete, random generation (trivially)
  - example: 10 jobs on one machine
    
    1 3 6 8 7 10 9 2 4 5

- Local changes/repairs of complete inconsistent schedule
  - exchange of two jobs,
  - job moved to different resource, ...
Local search (LS) in scheduling

- **Initial schedule:** complete, random generation (trivially)
  - example: 10 jobs on one machine
    \[1\, 3\, 6\, 8\, 7\, 10\, 9\, 2\, 4\, 5\]

- **Local changes/repairs of complete inconsistent schedule**
  - exchange of two jobs,
  - job moved to different resource, ...

- **Static scheduling:** search goals
  1. improvements in objective function
  2. search for consistent schedule

- **Dynamic scheduling:** extension
  3. solution of original problem as initial schedule
  4. natural extension
  5. good time complexity

- **Minimal perturbation problem:** extension
  6. minimization of the different values (e.g. job assignments)
  7. no optimality guarantees
Tree search

- Constructive search method
  - e.g., subsequent placement of particular activities to schedule
- Search by trivial enumeration
Tree search

- **Constructive search method**
  - e.g., subsequent placement of particular activities to schedule
- **Search by trivial enumeration**

- **Constraint propagation**
  - propagation = inconsistent states and their successors not explored
Branch & bound (BB)

Branch & bound search
- tree search method
- search for optimal solution
- maintained lower bound
- space with quality worse than lower bound pruned
Branch & bound (BB)

Branch & bound search
- tree search method
- search for optimal solution
- maintained lower bound
- space with quality worse than lower bound pruned

Local search
- large scale problems
- no guarantee of solution quality

Branch & bound
- not applicable for large scale problems
- can be used for local optimization
BB: dynamic scheduling

We have: solution of original problem + new changes and want "improve it"

Selection of activity $x$ with improper placement

Application of BB with at most $n$ changes

- reasonable $n$ is rather small
- reasonable search space size
- optimality guarantee within this search space
- solution out of this space not reasonable wrt. $x$ anyway
Iterative forward search (IFS) for scheduling

- Constructive method
- Partial consistent schedules

**Standard forward step**
- selection of activities and their placements if possible

"Backward step"
1. if impossible to assign selected activities without constraint violation
2. finding conflicting activities and removal of their placement
3. assignment of selected activities

**Timetable**

<table>
<thead>
<tr>
<th></th>
<th>7:30a</th>
<th>8:00a</th>
<th>8:30a</th>
<th>9:00a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENGR 126R Lec 1</td>
<td>4, 54, 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue</td>
<td>OLS 252 Lec 1</td>
<td></td>
<td>PHYS 272 Lec 1</td>
<td>0, 17, 0</td>
</tr>
<tr>
<td>Wed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENGR 126R Lec 1</td>
<td>4, 54, 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iterative forward search (IFS) for scheduling

- Constructive method
- Partial consistent schedules

**Standard forward step**
- selection of activities and their placements if possible

"Backward step"

1. if impossible to assign selected activities without constraint violation
2. finding conflicting activities and removal of their placement
3. assignment of selected activities

**Consequence**
- principles of the local search (local changes)
- IFS is not a tree search method

**Important principle**
- placement selection
- conflict statistics (memory for the number of conflicts)
Conflict statistics

- Suppose: selection of placement \( a \) of activity \( A \)
  we must remove placement of \( b \) for activity \( B \), i.e.

\[
[A = a \rightarrow \neg B = b]
\]

- We can remember during computation

\[
A = a \quad \Rightarrow \quad 3 \times \neg B = b, \quad 4 \times \neg B = c \\
2 \times \neg C = a, \quad 120 \times \neg D = d
\]
Conflict statistics

- Suppose: selection of placement $a$ of activity $A$
  we must remove placement of $b$ for activity $B$, i.e.

  \[ A = a \rightarrow \neg B = b \]

- We can remember during computation

  \[
  A = a \quad \Rightarrow \quad 3 \times \neg B = b, \quad 4 \times \neg B = c \\
  2 \times \neg C = a, \quad 120 \times \neg D = d
  \]

- Placement selection
  - selection of placements with smallest conflicts
IFS for dynamic scheduling

Static scheduling: search goals

1. search for placement of all activities
2. optimization of the objective function
   - preferred time assignment, preferred resource allocation, ...

Minimal perturbation problem: extension

3. minimization of the number of different placements
4. original placements preferred
   - during placement selection supported
     minimal difference in the number
     of differently assigned activities
IFS: experiments

- Plot 1: Additional perturbations vs. Input perturbations
- Plot 2: Additional perturbations [%] vs. Input perturbations
1 Problem description

2 Solving methods
   • Dispatching rules and policies
   • Local search
   • Branch & bound
   • Iterative forward search

3 Applications
   • Scheduling in Grid environment
   • University course timetabling
Job Scheduling in Grid environment

- **Distributed heterogeneous resources**
  - clusters, supercomputers, desktops, special instruments
  - autonomy, robustness

- **Type of jobs**
  - high-throughput (parameter studies), parallel, distributed

- **Scheduling on Grids**
  - scheduling of computational jobs to resources
  - scheduling of data transfers and data processing

- **Examples**
  - several clusters over Czech Republic (*META Center*)
  - hundreds clusters over Europe (EGEE)
  - several supercomputers (Deisa)
  - SETI@Home
Grid scheduling with Alea simulator

Problems with QoS requirements

- synthetic problems proposed by the group at ISTI CNR (Italy)
- heterogenous resources (frequency, number of processors)
- dynamic problems
  - jobs arriving over the time
  - known job processing time
  - no resource changes
- deadlines, sw licenses
- resource usage, job slowdown

Grid simulator Alea

- extension of simulation tool GridSim (Java)
- centralized scheduler
- modular system: different problems and algorithms
- static and dynamic scheduling
Local search for dynamic grid scheduling

- **Policies**
  - common use in the theory and practice
  - used for generation of the initial solution

- **Local search**
  - applied to static grid scheduling problems - very slow
  - grid community very sceptical to
    the effective use of local search!
  - local search not yet applied to the dynamic grid scheduling
    - computation of the schedule from scratch too slow

- **Goal**
  - application of local search to dynamic grid scheduling problems
  - generally
    - parallel jobs, QoS, interactive jobs, advance reservation, ...
    - application of advanced AI/OR algorithms

- **Current results**
  - synthetic problems with QoS requirements
  - improvements in optimization results, very good running time
Local search: experiments (deadlines vs. machine usage)
Local search: experiments (deadlines vs. machine usage)
Local search: experiments (run-time)
University course timetabling

- **Timetabling at Purdue University**
  - [http://www.unitime.org](http://www.unitime.org)
- **Large scale decentralized problem**
  - 2 central problems: large lecture rooms, computer laboratories
  - About 70 problems for particular administrative domains
- **State of the project**
  - 2001-2005: 3 projects between Masaryk and Purdue University
  - Spring 2005: first used large lecture room timetabling
  - Autumn 2007: first used system for the whole university
- **Timetabling steps**
  1. Centralized solving of large lecture rooms
  2. Solving of particular departmental problems
  3. Centralized solving of computer laboratories problem
  4. Centralized changes in the timetable
Static timetabling problems

- **Large lecture room problem**
  - 800 courses x 2 meetings, 55 classrooms
  - individual course enrollment for about 28,000 students
  - preferences on time and space, distances between classrooms,...
  - many exception handled

- **Other problems**: both automated solving and data input

- **Algorithm**: Iterative forward search
Minimal perturbation problem

Solving methods

- Iterative forward search to react on user requests
- Branch and bound for local schedule optimization

Suggestions

<table>
<thead>
<tr>
<th>Score</th>
<th>Class</th>
<th>Date</th>
<th>Time</th>
<th>Room</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.2</td>
<td>POL 101 Lec 3</td>
<td>Full Term</td>
<td>TTh 12:00p → TTh 7:30a</td>
<td>BRNG 2280</td>
<td>+11</td>
</tr>
<tr>
<td>31.7</td>
<td>POL 101 Lec 3</td>
<td>Full Term</td>
<td>TTh 12:00p → TTh 10:30a</td>
<td>BRNG 2280</td>
<td>+36 (h+3)</td>
</tr>
<tr>
<td></td>
<td>HIST 342 Lec 1</td>
<td>Full Term</td>
<td>TTh 10:30a → TTh 1:30p</td>
<td>BRNG 2280 → BRNG 2290</td>
<td></td>
</tr>
<tr>
<td>36.6</td>
<td>POL 101 Lec 3</td>
<td>Full Term</td>
<td>TTh 12:00p → TTh 10:30a</td>
<td>BRNG 2280</td>
<td>+36 (h+4)</td>
</tr>
<tr>
<td></td>
<td>HIST 342 Lec 1</td>
<td>Full Term</td>
<td>TTh 10:30a → TTh 7:30a</td>
<td>BRNG 2280</td>
<td></td>
</tr>
<tr>
<td>44.1</td>
<td>POL 101 Lec 3</td>
<td>Full Term</td>
<td>TTh 12:00p → TTh 10:30a</td>
<td>BRNG 2280</td>
<td>+34 (h+2)</td>
</tr>
<tr>
<td></td>
<td>HIST 342 Lec 1</td>
<td>Full Term</td>
<td>TTh 10:30a → TTh 3:00p</td>
<td>BRNG 2280 → BRNG 2290</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OBHR 330 Lec 4</td>
<td>Full Term</td>
<td>TTh 3:00p</td>
<td>BRNG 2290 → LWSN B155</td>
<td></td>
</tr>
</tbody>
</table>

(all 1571 possibilities up to 3 changes were considered, top 4 of 17 suggestions displayed)
Summary

Search algorithms for scheduling problems

Educational timetabling
- constraint satisfaction
- soft constraints
- local search methods

Grid scheduling
- intelligent scheduling algorithms
- simulators

http://www.fi.muni.cz/~hanka/publ/pisa08.pdf

Successor of Purdue solver
- written by Tomáš Müller
- extension of iterative forward search by local search methods
- International Timetabling Competition 2007
  - finalist in all three tracks, winner of two tracks