Task Scheduling

A Lecture in CE Freshman Seminar Series:
Ten Puzzling Problems in Computer Engineering
About This Presentation

This presentation belongs to the lecture series entitled “Ten Puzzling Problems in Computer Engineering,” devised for a ten-week, one-unit, freshman seminar course by Behrooz Parhami, Professor of Computer Engineering at University of California, Santa Barbara. The material can be used freely in teaching and other educational settings. Unauthorized uses, including any use for financial gain, are prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>May 2012</td>
<td>May 2015</td>
<td></td>
<td>May 2016</td>
<td></td>
</tr>
</tbody>
</table>
Mini-Sudoku Puzzle

Complete entries in this chart so that numbers 1-6 appear without repetition in each row, each column and each $2 \times 3$ block.

Standard Sudoku consists of a $9 \times 9$ chart, but this mini version is good for a quick fix.

USA Today carries a daily mini-Sudoku at its site: [http://puzzles.usatoday.com](http://puzzles.usatoday.com)

Sudoku isn’t a math puzzle: We can use the letters A-F, or any other six symbols, instead of the numbers 1-6.
Mini-Sudoku Puzzle: Solution Method

Complete entries in this chart so that letters A-F appear without repetition in each row, each column and each $2 \times 3$ block.

To continue from here, write down all possible choices in the remaining blank boxes and see whether the resulting info leads to more progress.

SuDoKu: abbr. in Japanese for “numbers must be single.” Euler may have invented it; Howard Garns (US) & Wayne Gould (HK) popularized it in modern times.

<table>
<thead>
<tr>
<th></th>
<th>AE</th>
<th>B</th>
<th></th>
<th>D</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td>D</td>
<td>C</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>C</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>E</td>
<td></td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
<td>F</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td></td>
<td></td>
<td>C</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
Sudoku Puzzle: Easy Example

Complete entries in this chart so that numbers 1-9 appear without repetition in each row, each column and each $3 \times 3$ block.

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7</td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Many newspapers carry these puzzles; there are also many collections in book form.

Sudoku puzzles of varying difficulties (easy, medium, hard, evil) are available at [http://www.websudoku.com](http://www.websudoku.com) and several other Web sites, such as USA Today’s site [http://puzzles.usatoday.com](http://puzzles.usatoday.com).
Sudoku Puzzle Solution Method

Strategy 1: Identify a missing number from a row, column, or block; if you can exclude all but one cell for that number, then write it down.

Strategy 2: When you can’t make progress by Strategy 1, write down all candidate numbers in the cells and try to eliminate a number of options via reasoning. For example if xy, xy, xyz are candidates in three cells of a block, then the cell marked xyz must hold z.
Sudoku Puzzle: Hard Example

Complete entries in this chart so that numbers 1-9 appear without repetition in each row, each column and each $3 \times 3$ block.

Hard puzzles typically have fewer entries supplied, with each row, column, or block containing only a few entries.

Hard puzzles may have handles or starting points (5 in the top left block or 9 in center and lower right blocks).
Constructing a (Mini-)Sudoku Puzzle

Begin with a completed puzzle and one by one remove selected entries that can be deduced from the remaining ones.

This will ensure a unique solution, which is a desirable attribute.

**Interesting fact:**
In a $9 \times 9$ Sudoku puzzle, you may need at least 17 initial entries (clues) for the solution to be unique; no one knows whether a 16-clue puzzle with unique solution exists.

http://people.csse.uwa.edu.au/gordon/sudokumin.php (Web page devoted to minimum Sudoku)
Variations on Sudoku

Other sizes (e.g., $6 \times 6$, with $2 \times 3$ blocks; or $16 \times 16$, with $4 \times 4$ blocks)

Combining this 2000s phenomenon with Rubik’s cube of the 1980s . . .
or with the age-old sliding 15 puzzle

Latin square
Task Scheduling Problem

We have a set of tasks

There are some “processors” that can execute tasks

Assign tasks to processors so as to meet certain constraints

A task may fit only some processors
Tasks may have prerequisite tasks
Preemption may be (dis)allowed
Tasks may have deadlines
Shortest schedule may be sought

Numbers in Sudoku puzzle

Cells in Sudoku puzzle can hold numbers

Place numbers in cells while honoring some constraints

Use only numbers 1-9
Some numbers already placed
Different numbers in each row
Different numbers in each column
Different numbers in each block

Virtually all instances of the task scheduling problem are difficult (NP-hard), just like Sudoku
Resource Allocation Problem

We have a set of resources

Numbers in Sudoku puzzle

There are “locations” where resources may be placed

Cells in Sudoku puzzle can hold numbers

Assign resources to locations to meet certain constraints

Place numbers in cells while honoring some constraints

A resource may fit only some locations
Resources must be “easily” accessible
Resource mobility may be (dis)allowed
Resource cost may differ by location
Lowest-cost assignment may be sought

Use only numbers 1-9
Some numbers already placed
Different numbers in each row
Different numbers in each column
Different numbers in each block

Virtually all instances of the resource allocation problem are difficult (NP-hard), just like Sudoku
Scheduling Required CE Courses

Constraints
Prerequisite:  
Solid downward arrow
Corequisite:  
Dashed sideways arrow
Units per quarter: \( \leq 18 \)

May 2016  
Task Scheduling  
BParhani
Constraints

Prerequisite:
Solid downward arrow

Corequisite:
Dashed sideways arrow

Units per quarter: ≤ 18
## Job-Shop Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Task</th>
<th>Machine</th>
<th>Time</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ja</td>
<td>Ta1</td>
<td>M1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Ja</td>
<td>Ta2</td>
<td>M3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb1</td>
<td>M2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb2</td>
<td>M1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb3</td>
<td>M2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Jc</td>
<td>Tc1</td>
<td>M3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Jd</td>
<td>Td1</td>
<td>M1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Jd</td>
<td>Td2</td>
<td>M2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

### Diagram

- **M1**: Blue
- **M2**: Green
- **M3**: Orange
## Schedule Refinement

### Table:

<table>
<thead>
<tr>
<th>Job</th>
<th>Task</th>
<th>Machine</th>
<th>Time</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ja</td>
<td>Ta1</td>
<td>M1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Ja</td>
<td>Ta2</td>
<td>M3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb1</td>
<td>M2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb2</td>
<td>M1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Jb</td>
<td>Tb3</td>
<td>M2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Jc</td>
<td>Tc1</td>
<td>M3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Jd</td>
<td>Td1</td>
<td>M1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Jd</td>
<td>Td2</td>
<td>M2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

### Graph:

- **Tb1**
- **Tb2**
- **Tb3**
- **Ta1**
- **Ta2**
- **Tc1**
- **Td1**
- **Td2**

The graph shows the schedule refinement with different tasks assigned to various machines over time. There is a question mark indicating a switch point. The tasks Ta1, Ta2, Tc1, Td1, Td2 are represented by colored bars, and the machines M1, M2, M3 are color-coded accordingly. The schedule runs from 0 to 14 time units. The table summarizes the jobs, tasks, machines, times, and staff required for each task.
Multiprocessor Scheduling

Task graph with unit-time tasks
Here’s a heuristic known as list scheduling:
1. Find the depth $T_\infty$ of the task graph
2. Take $T_\infty$ as a goal for the running time $T_p$
3. Determine the latest possible start times
4. Assign priorities in order of latest times

$T_\infty = 8$ (execution time goal)
Latest start times: see the layered diagram
Priorities: shown on the diagram in red

When two tasks have the same “latest start time,” a secondary tie-breaking rule is used
Assignment to Processors

Tasks listed in priority order

<table>
<thead>
<tr>
<th>1* 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: 1 2 3 4 6 5 7 8 9 10 11 12 13</td>
</tr>
<tr>
<td>P2: 5 7 9 11</td>
</tr>
<tr>
<td>P3: 8 11</td>
</tr>
</tbody>
</table>

Even in this simple case of unit-time tasks, multiprocessor scheduling remains difficult with as few as 3 processors.
Two Related and Similar Problems

The knapsack problem

- We have storage capacity $W$ and $n$ files of sizes $w_i$ and values $v_i$
- Pick a max-value subset of files that fit in the storage space $W$
- Files cannot be broken into pieces
- Naïve solution: Examine all $2^n$ subsets
- Dynamic programming solution
- Various heuristic aids
- Approx. solutions (say, 90% of optimal)

Off-line game of Tetris

- We have a rectangular bin and a sequence of tetrominos
- Find optimal play to maximize the number of pieces used
- Pieces can only be rotated
- Exponentially many choices

There are many other related and similarly hard problems, some of which don’t even admit efficient approximations