70068 Scheduling and Resource Allocation CW

Ali Abbas

November 26, 2024

1 Question 1

1.1 LCL Proof

The *Least Cost Last* (LCL) rule solves the scheduling problem $1|prec|f_{\max}$ optimally, in $O(n^2)$ time, by constructing an optimal schedule from back to front.

Definitions

- $N = \{1, 2, \ldots, n\}$ is the index set of all jobs
- $L \subseteq N$ is the subset of jobs without successors (which can be scheduled last)
- $p(S) = \sum_{j \in S} p_j$ is the total processing time of a subset S
- $f^*_{\max}(S)$ is the cost of the optimal schedule for subset S
- $f_{\max}$ denotes the maximum (not necessarily optimal) cost across all jobs in a schedule
- $f_j(p(N))$ is a cost function which assigns a penalty/cost depending on job j and the time at which the machine has finished processing, $p(N)$
  - In the notation of the coursework specification $g_j(\cdot) = f_j(\cdot)$
  - In the next section we'll use $g_j(C_j) = T_j = \max(0, C_j - d_j)$ (tardiness)

Proof and Discussion

1. One job in L must be scheduled last. A job can be selected for the final job which minimises $f_j(p(N))$, but it can never result in a cost lower than the cost of the optimal schedule, otherwise that would be the optimal schedule. This is expressed as such:

$$f^*_{\max}(N) \geq \min_{j \in L} f_j(p(N))$$

2. And removing a job $j \in N$ can't increase the optimal cost any more, since the cost of scheduling a job last is ≥ 0, so omitting it cannot result in a more expensive schedule:

$$f^*_{\max}(N) \geq f^*_{\max}(N - \{j\}),\ \forall j \in N$$

3. We can then select a $J_l \in L$ that minimises $f_j(p(N))$:

$$f_l(p(N)) = \min_{j \in L} f_j(p(N))$$

   Which gives us:

$$f^*_{\max}(N) \geq \max\{f_l(p(N)), f^*_{\max}(N - \{l\})\}$$

4. The right hand side of the above inequality is the cost of an optimal schedule, where J_l is processed last, so you can recursively apply the LCL rule to $N - \{J_l\}$ and construct a schedule in reverse order
   - Since J_l is found in $O(n)$ time, then with n, repeated applications of the LCL rule yields the optimal schedule in $O(n^2)$ time

You can also show that LCL is optimal by using the *adjacent pairwise interchange argument*:

- You take an optimal schedule S and assume it isn't an LCL schedule
- Since it's not an LCL schedule, there must exist at least one pair where a lower cost job precedes a higher cost one

- It can then be shown that swapping these jobs yields a lower cost schedule, which is a contradiction since we took an optimal schedule
- So by virtue of the fact that a lower cost job can precede a higher cost one leads to a contradiction
- And so such pairs cannot exist for an optimal schedule, so we can reject the assumption that this isn't an LCL schedule
- And by contradiction then we have that LCL is optimal for $1|prec|f_{\max}$

Small Example

Example Setup:
Jobs: J0, J1, J2, J3, J4
Processing Times: p=[2,3,1,2,3]
Dude Dates: d=[6,5,7,4,9]
Precedence Constraints (DAG): J0 $\rightarrow$ J1, J0 $\rightarrow$ J2, J3 $\rightarrow$ J4
Cost Function: $f_j(C_j) = T_j = max(0, C_j - d_j)$

Iterations We can schedule jobs in reverse order using the *Least Cost Last* (LCL) rule:

0. Available jobs: $V = \{1, 2, 4\}$ (no successors)
   $p(N) = 11$

   $$f_1(p(N)) = \max(0, 11 - 5) = 6, \quad f_2(p(N)) = \max(0, 11 - 4) = 7, \quad f_4(p(N)) = \max(0, 11 - 9) = 2$$

   Select J_4 (minimises $f_j(p(N))$)

   Partial schedule cost: 2

   Updated schedule: [4]

1. Available jobs: $V = \{1, 2, 3\}$ (no successors)
   $p(N) = 8$

   $$f_1(p(N)) = \max(0, 8 - 5) = 3, \quad f_2(p(N)) = \max(0, 8 - 4) = 4, \quad f_3(p(N)) = \max(0, 8 - 7) = 1$$

   Select J_3 (minimises $f_j(p(N))$)

   Partial schedule cost: 2

   Updated schedule: $[3, 4]$

2. Available jobs: $V = \{1, 2\}$ (no successors)
   $p(N) = 6$

   $$f_1(p(N)) = \max(0, 6 - 5) = 1, \quad f_2(p(N)) = \max(0, 6 - 4) = 2$$

   Select J_1 (minimises $f_j(p(N))$)

   Partial schedule cost: 2

   Updated schedule: $[1, 3, 4]$

3. Available jobs: $V = \{2\}$ (no successors)
   $p(N) = 3$

   $$f_2(p(N)) = \max(0, 3 - 4) = 0$$

   Select J_2

   Partial schedule cost: 2

   Updated schedule: $[2, 1, 3, 4]$

4. Available jobs: $V = \{0\}$ (no successors)
   $p(N) = 2$

   $$f_0(p(N)) = \max(0, 2 - 6) = 0$$

   Select J_0

   Partial schedule cost: 2

   Updated schedule: $[0, 2, 1, 3, 4]$

Results

- Final schedule: $[J_0, J_2, J_1, J_3, J_4]$
- Completion times:

$$C_0 = 2,\ C_2 = 3,\ C_1 = 6,\ C_3 = 8,\ C_4 = 11$$

- Tardiness:

$$T_0 = 0,\ T_2 = 0,\ T_1 = 1,\ T_3 = 1,\ T_4 = 2$$

- Maximum cost:

$$f_{\max} = \max(T_0, T_2, T_1, T_3, T_4) = 2$$

1.2 LCL Implementation

These iterations use 0-based indexing.

Iterations

0. Available jobs: $V = \{0, 30\}$ (no successors)
   $p(N) = 170$

   $$f_0(p(N)) = \max(0, 170 - 172) = 0, \quad f_{30}(p(N)) = \max(0, 170 - 269) = 0$$

   Select J_{30} (minimises $f_j(p(N))$)
   Partial schedule cost: 0

   Updated schedule: $[30]$

1. Available jobs: $V = \{0, 1, 10\}$ (no successors)
   $p(N) = 160$

   $$f_0(p(N)) = \max(0, 160 - 172) = 0, \quad f_1(p(N)) = \max(0, 160 - 82) = 78, \quad f_{10}(p(N)) = \max(0, 160 - 253) = 0$$

   Select J_0 (minimises $f_j(p(N))$)
   Partial schedule cost: 0

   Updated schedule: $[0, 30]$

...

4. Available jobs: $V = \{1, 4\}$ (no successors)
   $p(N) = 147$

   $$f_1(p(N)) = \max(0, 147 - 82) = 65, \quad f_4(p(N)) = \max(0, 147 - 93) = 54$$

   Select J_1 (minimises $f_j(p(N))$) Partial schedule cost: 65

   Updated schedule: $[1, 14, 10, 0, 30]$

   This iteration was noteable since it incurred the maximum tardiness incurred for this problem.

...

17. Available jobs: $V = \{17, 18, 20, 5\}$ (no successors)
    $p(N) = 68$

    $$f_{17}(p(N)) = \max(0, 68 - 77) = 0, \quad f_{18}(p(N)) = \max(0, 68 - 88) = 0,$$
    $$f_{20}(p(N)) = \max(0, 68 - 71) = 0, \quad f_5(p(N)) = \max(0, 68 - 71) = 0$$

    Select J_{17} (arbitrarily breaking the tie)
    Partial schedule cost: 65

    Updated schedule: $[17, 16, 15, 13, 28, 12, 27, 26, 25, 24, 23, 11, 4, 1, 14, 10, 0, 30]$

...

30. Available jobs: $V = \{29\}$ (no successors)
    $p(N) = 2$

    $$f_{29}(p(N)) = \max(0, 2 - 329) = 0$$

    Select J_{29} (minimises $f_j(p(N))$)
    Final schedule cost: 65

    Final schedule: $[29, 9, 8, 3, 2, 7, 6, 22, 21, 5, 20, 19, 18, 17, 16, 15, 13, 28, 12, 27, 26, 25, 24, 23, 11, 4, 1, 14, 10, 0, 30]$

Note: the tie breaking doesn't specify any ordering in the current implementation (since the set of available jobs is backed by a hash table).

1.3 LCL Implementation

2 Question 2

2.1 Tabu Search

2.2 Best Schedule