1 2-Machine Scheduling With Costs

Let’s consider new variant of multiprocessor scheduling problem.

- Given set of \( n \) jobs.
- Each job must be run on one of two machines \( (m = 2) \).
- Jobs cannot be split across machines.
- Job \( i \) takes time \( \ell_{i1} \) on machine 1, or \( \ell_{i2} \) on machine 2.
- **Goal**: assign jobs to machines to minimize time at which last job finishes.
- As before, if \( \ell_{i1} = \ell_{i2} \) for all \( i \), problem is same as unweighted case, which is NP optimization problem.
- Again, however, approximation we’ve seen for unweighted case does not work for weighted case!
- Will derive a new approximation algorithm using LP.

2 Formulating an ILP for Scheduling

First, let’s formulate the problem as an ILP.

- For each job \( i \), define indicators \( x_{i1} \) and \( x_{i2} \).
- \( x_{ij} \) is 1 if job \( i \) runs on machine \( j \), 0 otherwise.
- Every job runs somewhere, and no job runs on both machines, so for each job \( i \),
  \[ x_{i1} + x_{i2} = 1. \]
- (We also enforce \( x_{ij} \geq 0 \), as for all LPs. We do not separately enforce \( x_{ij} \leq 1 \), as it follows from the other constraints.)
- Now, we use a trick to minimize the largest finishing time.
- Let \( t \) be a variable representing time at which last job finishes.
Clearly, for $1 \leq j \leq 2$, we have
\[
\sum_i l_{ij} \cdot x_{ij} \leq t
\]
since all the jobs on each machine must finish by time $t$.

- Objective function to be minimized is simply “$t$”.
- In a feasible ILP soln, $x_{ij}$s specify which jobs run on each machine.
- (Technically, an ILP would require variable $t$ to be integer as well, but problem is still hard if we let $t$ be real but require $x$’s to be integer.)

Consider the LP relaxation of this ILP.
- Each $x_{ij}$ is between 0 and 1, and they add to 1 for any job.
- Effectively, LP may split each job across two machines, running part of it on each.
- This is not allowed in our problem, or in ILP.
- How does the “split solution” help us solve the ILP well?
- Will show that LP solution is “almost” integer, then show how to fix up the non-integer part.
- More specifically, we will show that in LP solution, most variables do not need to be rounded!

3 Analysis of LP (Lack of) Rounding

To begin, need one important fact about linear programming.

- Suppose we have a linear programming problem $P$ over a set $X$ of $k$ variables.
- In general, $j$th constraint of $P$ has form
  \[
  \sum_i a_{ij} x_i \text{ op } b_j.
  \]
  where “op” is one of $\leq$, $\geq$, or $=$.
- **Defn**: solution $X$ makes $j$th constraint *tight* if
  \[
  \sum_i a_{ij} x_i = b_j.
  \]
  That is, $X$ satisfies $j$th constraint with equality.
- **Thm**: If any optimal feasible solution $\overline{X}$ exists for $P$, then there exists such an $\overline{X}$ that makes at least $k$ constraints tight.

Why?
• Thm is one version of the **Fundamental Theorem of LP** – an optimal solution can always be found at a vertex of the LP polytope.

• The $k$ tight constraints correspond to the $k$ hyperplanes that have to intersect to form a vertex in $k$-dimensional space.

• NB: an $a = n$ constraint counts as *one* constraint, not two inequalities, since it is represented by just one hyperplane.

So what?

• Suppose $P$ has $c$ total constraints and $k$ total variables.

• Suppose that $k'$ of the constraints have form $x_i \geq 0$. (Clearly, $k' \leq k$ and is usually $= k$.)

• (Remaining $c - k'$ constraints are not of this form.)

• Now consider optimal soln $X$ that makes $k$ constraints of $P$ tight.

• Number of *tight* constraints of the form $x_i = 0$ is then at least

$$k - (c - k').$$

• Conclude that in $X$, at least $k + k' - c$ variables have value 0.

Let’s count vars and constraints in our LP relaxation for scheduling.

• Suppose there are $n$ total jobs to be scheduled.

• For each $1 \leq i \leq n$, we have variables $x_{i1}$ and $x_{i2}$.

• We added one more variable for largest running time $t$.

• Hence, $k = 2n + 1$ total variables.

• Now for the constraints:

  - Two constraints of form

$$\sum_i \ell_{ij} \cdot x_{ij} \leq t.$$  

  - For each $1 \leq i \leq n$, one equality constraint:

$$x_{i1} + x_{i2} = 1.$$  

  - Finally, $k' = 2n + 1$ constraints of form $v \geq 0$.

Hence, a total of $c = 3n + 3$ constraints.

• Conclude that in an optimal solution $X$, at least $(2n+1) + (2n+1) - (3n+3) = n - 1$ variables are 0.

• $t$ cannot feasibly be 0 if there are any jobs of nonzero length.
Hence, \( n - 1 \) of the \( x_{ij} \)'s are zero.

But \( x_{i1} \) and \( x_{i2} \) cannot both be 0.

Conclude that, for at least \( n - 1 \) values of \( i \), one of \( x_{i1} \) and \( x_{i2} \) is 0, and the other is 1.

(Note that we could have dropped constraint \( t \geq 0 \) from the LP, since it is implied by other constraints; this does not change our analysis because it reduces both \( c \) and \( k' \) by one.)

Hence, an optimal solution to the LP relaxation does not split at least \( n - 1 \) of \( n \) jobs, and so leaves at most one job split between machines.

4 The Full Algorithm

- Algorithm LP-MPSCHED takes \( n \) jobs and 2 processors.
- Form ILP as defined previously over variables \( X = \{t, x_{11}, x_{12}, \ldots, x_{n1}, x_{n2}\} \).
- Solve the LP relaxation, obtaining LP optimum \( X \).
- For every integral pair \( x_{i1}, x_{i2} \), assign each job \( i \) to machine 1 if \( x_{i1} = 1 \) or machine 2 if \( x_{i2} = 1 \).
- If no non-integral \( x_{ij} \) exists, we are done. Otherwise, suppose some \( x_{qj} \) is not integral.
- Assign job \( q \) to the machine on which it takes the least time.

Claim: schedule returned by LP-MPSCHED has length at most twice the optimum.

- Pf: Define

\[
T(X) = \max \sum_i \ell_{ij} x_{ij}
\]

to be objective value for LP solution \( X \).

- Let \( X^* \) be ILP optimum; let \( \overline{X} \) be LP optimum; and let \( \hat{X} \) be algorithm’s solution.
- We know that \( T(\overline{X}) \leq T(X^*) \) (LP OPT \( \leq \) ILP OPT).
- Now for upper bound!

- Consider (non-feasible) solution \( X' \) obtained by removing split job \( q \) from \( \overline{X} \) entirely, that is, setting \( \overline{x}_{q1} = \overline{x}_{q2} = 0 \).
- Note that \( T(X') \leq T(\overline{X}) \leq T(X^*) \).
- Now let \( \ell_q = \min(\ell_{q1}, \ell_{q2}) \).
- Any feasible schedule includes job \( q \), so \( \ell_q \leq T(X^*) \).
- Algo’s soln takes solution \( X' \) and adds job \( q \) to its least expensive processor, incurring added cost at most \( \ell_q \) for the solution.
• Hence, we have

\[ T(\tilde{X}) \leq T(X') + \ell_q \leq T(X^*) + T(X^*) = 2T(X^*). \]

Conclude that LP-MPSCHED is a 2-approximation. QED

With some work, can extend this algorithm into a 2-approximation for weighted MP scheduling on an arbitrary number of processors.