Weighted interval scheduling

Robbie Nohra

November 2018

We have a set of n requests.

$$S = \{r_1, \ldots, r_n\} \ .$$

Each request $r_i = (s_i, f_i)$ has a start time s_i and finish time f_i.

Each request i has a corresponding weight w_i.

We want a subset $\mathcal{O}$ of S (possibly all of S) such that

$$\sum_{i:r_i \in \mathcal{O}} w_i$$

is maximized.

The one constraint is that the intervals in the subset must be *disjoint*.

Preliminary work

First order the requests r_i in non-decreasing order of the finish times f_i.

Next define

$$p(j) = \max_i \left\{ i < j : r_i \cap r_j = \emptyset \right\} \ .$$

$r_{p(j)}$ is then the closest disjoint interval to the left of r_j.

Define $S_j = \{r_1, \ldots, r_j\}$.

The first key property that $r_{p(j)}$ possesses is that all intervals between itself and r_j intersect with r_j.

By definition, it is is the *largest* non-intersecting interval to the left of r_j.

This means that if we had an optimal solution $\mathcal{O}_j$ to S_j and we happened to know that $r_j \in \mathcal{O}_j$ we could consider the weight w_j, disregard the sets $\{r_{p(j)+1}, \ldots, r_{j-1}\}$, and iterate.

We are implicitly using the fact here that the intervals were sorted in the first step

The second key property is that if $\mathcal{O}_j$ is an optimal solution for S_j and again we happen to know that $r_j \in \mathcal{O}_j$ then there exists an optimal solution $\mathcal{O}_{p(j)} \subset \mathcal{O}_j$ to $S_{p(j)}$.

The proof follows by contradiction.

Suppose that $r_j \in \mathcal{O}_j$ and there did not exist an optimal solution $\mathcal{O}_{p(j)} \subset \mathcal{O}_j$ to $S_{p(j)}$.

In particular, this would mean that $\mathcal{O}_j \cap S_{p(j)} \subset O_j$ was a suboptimal solution to $S_{p(j)}$.

We could therefore adjoin some collection of intervals $\tau \subset S_{p(j)}$ to $\mathcal{O}_j \cap S_{p(j)}$ and improve the optimality of $\mathcal{O}_j \cap S_{p(j)}$ itself on $S_{p(j)}$.

The contradiction we want to arrive at is that this improves the optimality of $\mathcal{O}_j$ itself. To prove this we just need to show that no interval in τ overlaps with an interval in $\mathcal{O}_j$.

This already holds for $\mathcal{O}_| \cap S_{p(j)}$ by definition of τ

Intervals belonging to an optimal solution are by definition disjoint.

All that is needed to be shown then is that the intervals in τ do not overlap with the inervals in $\mathcal{O}_j \cap S'_{p(j)} = \mathcal{O}_j \cap \{r_{p(j)+1}, \ldots, r_{j-1}\}$.

We can omit r_j since we already know by hypothesis that $r_j \in \mathcal{O}_|$. If r_j is also in τ then an interval in τ will overlap with an interval in $\mathcal{O}_|$, namely r_j with itself. But *every* non-empty interval intersects itself, so that is not really an issue as far as the proof is concerned.

Now, this set looks familiar though. We've already shown above that if $r_j \in \mathcal{O}_j$ then $\{r_{p(j)+1}, \ldots, r_{j-1}\} \not\subset \mathcal{O}_j$.

That is, since $r_j \in \mathcal{O}_j$ by hypothesis, $\mathcal{O}_j \cap \{r_{p(j)+1}, \ldots, r_{j-1}\} = \emptyset$.

The intervals in τ therefore have nothing to intersect with in $\mathcal{O}_j \cap \{r_{p(j)+1}, \ldots, r_{j-1}\}$. The statement that they do not overlap is then vacuously true.

We've therefore arrived at our contradiction. We can adjoin τ to $\mathcal{O}_j$ itself and improve its optimality. But $\mathcal{O}_j$ was by definition optimal.

We have therefore proven by contradiction that if $r_j \in \mathcal{O}_j$ (where $\mathcal{O}_j$ is an optimal solution to S_j) then there must exist an optimal solution $\mathcal{O}_{p(j)} \subset \mathcal{O}_j$ to $S_{p(j)}$.

If $r_j \notin \mathcal{O}_j$ then we can disregard r_j altogether and say that the optimal solution over S_j is equal to the optimal solution over S_{j-1}.

The proof follows directly by contradiction.

The solution

Define $\mathcal{O}_j$ to be the optimal solution over S_j and $\text{opt}(j)$ to be the value of this solution.

For $S_0 = \emptyset$ we define $\text{opt}(0) = 0$

We have determined above that a recursive solution involves dividing the solution at each iteration into cases.

The first case is when $r_j \in \mathcal{O}_j$. In this case we showed that

$$\text{opt}(j) = v_j + \text{opt}(p(j))$$

The second case is when $r_j \notin S_j$. In this case we showed that

$$\text{opt}(j) = \text{opt}(j-1)$$

How do we know if $r_j \in \mathcal{O}_j$?

We can just look at both quantities above and see which one is larger. In that way, we can merge both cases into one by writing

$$\text{opt}(j) = \max\left(v_j + \text{opt}(p(j)), \text{opt}(j-1)\right)$$

Our goal is to compute $\text{opt}(n)$. We can do that now by iterating over this recursion for $j = 1, \ldots, n$.