Computer Science B36 Fall 2017
Scarborough Campus University of Toronto

Comments on proving the correctness of (some) greedy algorithms

Vassos Hadzilacos

In class we proved the correctness of the greedy algorithm for interval scheduling by employing a "greedy-stays-ahead" argument. Specifically, we proved that if the set of non-conflicting jobs constructed by the greedy algorithm listed left-to-right are

$$j_1, j_2, \ldots, j_k$$

and the jobs in an optimal set listed left-to-right are

$$j_1^*, j_2^*, \ldots j_m^*,$$

then $f(j_t) \leq f(j_t^*)$ for every $t = 1, 2, \ldots, k$. Using this, we then argued that the set of jobs produced by the greedy algorithm is, in fact, optimal.

There is an alternative proof of a similar nature that uses what might be called the "promising set" argument. Specifically, we prove that:

Lemma 1 *For every iteration i of the loop, the set of jobs that have been added to the set A at the end of that iteration is a subset of some optimal set.*

Proof. By induction on i. The basis $i = 0$ is obvious, since at the end of the "0-th iteration" of the loop, i.e., just before we enter the loop for the first time, A is empty.

For the induction step, let A_i be the set of jobs in A at the end of iteration i, for some $i \geq 0$. Assume (by induction hypothesis) that A_i is a subset of some optimal set A^* of jobs. Let A_{i+1} be the set of jobs in A at the end of iteration $i+1$. We will prove that A_{i+1} is a subset of some optimal set $\hat{A}$ of jobs. (Note that $\hat{A}$ may be a different optimal set than A^*.) This is obvious if no job is added to A in iteration $i+1$, or if the job j added to A in iteration $i+1$ happens to be in A^*. So, suppose that some job j is added to A in iteration $i+1$ and $j \notin A^*$.

In that case, first we note that A^* contains some job j^* that conflicts with j: for, otherwise, we could add j to A^* and obtain a feasible set with more jobs than A^*, contradicting that A^* is optimal. Second, we note that A^* does not contain two jobs that conflict with j: for, otherwise, both of them would start after all the jobs in A_i finished (because A_i is, by induction hypothesis, a subset of A^*, which is optimal and hence feasible), and one of them, call it j', would finish before j; so the greedy algorithm would have added j', rather than j, to A in iteration $i+1$. So, there is exactly one job j^* in $A^* - A_i$ that conflicts with j. Let $\hat{A}$ be the set obtained from A^* by replacing j^* with j; i.e., $\hat{A} = (A^* - \{j^*\}) \cup \{j\}$. The set $\hat{A}$, (i) is feasible (since j^* is the only job in A^* that conflicts with j); (ii) has the same number of jobs as the optimal set A^*; and (iii) contains $A_i \cup \{j\} = A_{i+1}$ (by construction). So, $\hat{A}$ is an optimal set that contains all the jobs in A_{i+1}, as wanted. □

We now prove that the set A returned by the greedy algorithm is optimal: By the preceding lemma, for some optimal set A^*, $A \subseteq A^*$. We claim that, in fact, $A = A^*$. For, otherwise, there would exist some job $j \in A^* - A$. Since A^* is optimal, j does not conflict with any other job in A^*. Since $A \subseteq A^*$ and $j \notin A$,

j does not conflict with any job in A. So, when j was considered for inclusion in A by the algorithm, it would have be added to it, contradicting that $j \notin A$. So, $A = A^*$, which means that the greedy algorithm returns an optimal set.

The "greedy-stays-ahead" and "promising set" arguments both capture the intuition that a greedy algorithm builds an optimal solution incrementally so that the partial solution constructed at each stage is "optimal so far". In the "greedy-stays-ahead" argument we do induction on the number of "pieces" that make up the optimal solution. In the "promising set" argument we do induction on the number of iterations that the greedy algorithm performs.

The "exchange" (or "switching") argument we saw in the proof of the greedy algorithm for minimum-lateness scheduling has a different flavour. There we prove that the greedy algorithm produces an optimal solution by showing how to transform an arbitrary optimal solution to the one constructed by the greedy algorithm through a series of "exchange (or switching) steps", each of which preserves optimality. This style of proof also uses induction (or its first cousin, the well-ordering principle), but in a different way. We don't do induction on the number of steps through which the algorithm builds its solution. Rather, we do induction on the number of optimality-preserving steps required to gradually "massage" the arbitrarily chosen optimal solution into the solution produced by the greedy algorithm.

The promising set argument bears some resemblance to the exchange argument in that the optimal solution that extends the partial solution may need to be changed in some iterations, and the change involves replacing (i.e., exchanging) some elements of an optimal solution with some elements of the solution constructed by the greedy algorithm. For instance, in the preceding proof, we had to change A^* to $\hat{A}$ by replacing j^* with j.