CSE 101 Class Notes
Greedy Algorithms

April 21, 2009

Overview

Greedy algorithms always choose the intuitively "best" option at each decision point, and (unlike backtracking) do not consider other alternatives. The problem is that while this powerful intuition is powerful, it is often wrong – just as in life, acting in one's immediate best interest is not always the best longer-term strategy. Therefore it is especially important to prove that a greedy algorithm finds the best solution. Luckily, these proofs typically follow a standard form, which we will discuss today.

Example: Event scheduling

Instance: a set of events $E = \{E_i = (s_i, f_i)\}$ where $s_i < f_i$.

Solution form: A subset $S \subset E$.

Constraints: No two events in S intersect.

Objective: Maximize the number of events, $|S|$.

Possible Greedy Algorithms

At each decision, a backtracking algorithm tries each feasible choice and returns the best solution. A greedy algorithm, on the other hand, tries only the (locally) best option, ignoring locally less-appealing choices. The first hard part of coming up with a greedy algorithm is finding the right definition of "locally best" – often a naive approach will fail where a more sophisticated version of local optimality will work.

To show that a greedy algorithm works, we must show that choosing greedily will do no worse than searching exhaustively. This is the second hard part of coming up with a greedy algorithm – many problems have greedy "solutions" that are as appealing and natural as they are wrong.

Example: Job (interval) scheduling

Given a list of jobs with start- and finish-times, maximize the number of non-overlapping jobs performed. We can imagine several definitions of local optimality, choosing at each step the job with:

1. shortest duration
2. fewest conflicts
3. earliest start
4. earliest finish

None of definitions 1-3 works; to see this, we can come up with counterexamples for each. For example, maybe there is a short duration job that intersects with two long duration jobs, one intersecting the first part, the other the second part: e.g., (1,9),(7,11),(10,18). The first algorithm would take the short middle job over the two longer jobs. The first job to start might also be the last to end, so the third algorithm is not always optimal. For the second, consider 3 jobs from (1,3), 5 from (2,6), one from (4,8), 3 from (7,10), one from (9,12) , 5 from (11,15), and 3 from (14,16). The number of conflicts for each category are 7,8,8,4,8,8,7, so the second algorithm would pick one of the middle 3 jobs, from (7,10) first. This gives a schedule with 3 jobs, wheras there is a scheudle with 4 jobs, one from each of the first, third, fifth, and seventh categories.

Algorithm 4 succeeds in these cases, but we need to prove that it works for all cases.

Algorithm

The right heuristic is to always choose the event with the earliest end-time. Here is a backtracking algorithm of `Schedule` that considers both options for this choice:

```
1     Schedule(E)
2         e <- event minimizing finish(e)
3         s1 <- Schedule(E - overlap(e)) + { e }
4         s2 <- Schedule(E - e)
5         if |s1| <  |s2|
6             return s2
7         else
8             return s1
```

The greedy version differs by never considering `s2` on line 4, always returning `s1`.

Correctness

We need to prove that `s1` is always at least as good as `s2`, i.e. $|S_1| \geq |S_2|$.

Consider an event e_i chosen by the greedy algorithm. Let S' be the best schedule not containing e_i, i.e. the best solution violating our greedy heuristic; and let S be the best schedule including e_i, i.e. the best following the heuristic in considering e_i. Our goal is to show that S is at least as good as S'.

Theorem: The greedy schedule is always a largest legal schedule.

> **Lemma:** Let E_i be the event with minimal f_i. Let S' be any legal schedule s.t. $E_i \notin S$. Then there is a legal schedule S s.t. $E_i \in S$ and $|S| \geq |S'|$.
>
> **Proof:** Let $C(E_i)$ be the set of elements that conflict with E_i. Any even $E_j \in C(E_i)$ must have $s_i \leq f_j \leq f_i$. Let $S = S' - C(E_i) \cup \{E_i\}$. Then S, containing E_i, is a legal schedule, since S' has no conflicts, and S does not contain any events in $C(E_i)$.
>
> We still need to show that $|S| \geq |S'|$, i.e. that $|C(E_i)| \leq 1$. Assume that this is not the case, i.e. $\exists E_j, E_k$ with $s_i \leq f_j, f_k \leq f_i$. Then E_j, E_k must conflict, contradicting our assumption that S' is a legal schedule. Therefore $|C(E_i)| \leq 1$, and $|S| \geq |S'|$.

Proof: (by strong induction on $|S|$, the number of events)

Inductive hypothesis: for all n, if there exists a schedule S' with n events, then the transformation above can be used to generate a greedy schedule S with $|S| \geq |S'|$.

Base case: If $|S| = 0, 1$, the algorithm finds either 0 or 1 event.

Inductive case: Assume the greedy algorithm is optimal for $|S| < n$, and let E_i be the first remaining event to finish (i.e. the greedy choice). Let S'_n be the non-greedy optimal schedule making a non-greedy choice instead of E_i, and let S'_{n-1} be its first $n-1$ events. By our lemma, the schedule $S'' = S'_{n-1} - C(E_i) \cup \{E_i\}$ has $|S''| \geq |S'_n|$ and $E_i \in S''$. By our inductive hypothesis, $|S_{n-1}| \geq |S'_{n-1}|$. Also, since S_{n-1} is greedy, the last event in S'_{n-1} finishes no sooner than the last event in S_{n-1}, so $S_{n-1} \cup \{E_i\}$ is a legal solution.

Therefore

$$
\begin{aligned}
|S_n| &= |S_{n-1} \cup \{E_i\}| = |S_{n-1} - C(E_i) \cup \{E_i\}| \\
&\geq |S'_{n-1} - C(E_i) \cup \{E_i\}| = S'' \\
&\geq |S'_n|
\end{aligned}
$$

proving our inductive hypothesis.

Example

```
    (1,3),(2,4),(3,6),(0,6),(5,7),(5,8),(9,10)
S2:         +                 +           +
A:    +     -          (-)
B:                +           -    (-)
S1:   +           +                       +
```

Here `S2`'s first move is non-greedy. The transformation in our proof replaces `(2,4)` with `(1,3)` (line `A`), then performs another greedy transformation (line `B`), yielding an equal-sized schedule `S1`.

Implementation

A naive implementation of the above approach, searching the entire list at each step for the next choice, then for overlapping events, would take $O(n^2)$. An efficient implementation instead sorts the list by finish time, and skips over overlapping events as it traverses this sorted list:

```
Schedule2(E)
    E <- sort_by_finish(E)
    S <- E[1]
    f <- finish(E[1])
    for i = 2 .. n
        if start(E[i]) > f
            S <- S + E[i]
            f <- finish(E[i])
```

General approach

The "modify-the-solution" approach used above can be applied to prove the correctness of many greedy algorithms. In general, we proceed by the following steps:

1. Let d be the first decision point, and let g be the greedy choice at d.
2. Let S' be a solution not choosing g.
3. Show how to transform S' into some S that chooses g, and that is at least as good as S'.
4. Conclude by induction that any S' with a series of non-greedy decisions at $d_1 \ldots d_n$ can be transformed into an equal-or-better greedy solution, and that therefore the greedy algorithm is optimal.