Maximum Independent Set for Intervals by Divide-Prune-and-Conquer

Jack Snoeyink*
Dept. of Computer Science
UNC Chapel Hill

Abstract

As an easy example of divide, prune, and conquer, we give an output-sensitive $O(n \log k)$-time algorithm to compute, for n intervals, a maximum independent set of size k.

1 Introduction

Consider the problem of finding a maximum independent set of n intervals on the real line – that is, a subset of non-overlapping intervals of maximum size, k. If the intervals are sorted by their right endpoints, then an easy linear-time greedy algorithm solves the problem [3]: take the interval I that ends first; skip all intervals that intersect I, which are immediately consecutive in the sorted order; and repeat. On the other hand, if the intervals are not sorted, element uniqueness gives an $\Omega(n \log n)$ lower bound [3].

This note establishes an $O(n \log k)$-time algorithm that finds the same solution as the greedy algorithm. This running time can be called output sensitive, as it depends on this size k of a maximum independent set. It is obtained by the algorithmic technique of divide, prune, and conquer, which I first encountered in output-sensitive algorithms for convex hulls [1, 2].

2 Divide, prune, and conquer

A few divide and conquer algorithms in computational geometry also prune to make sure that they expend effort only on subproblems that are guaranateed to contribute to the solution. A lemma of Edelsbrunner and Shi [2] can show output-sensitive running times for such algorithms; we give a simple proof, based on recursion trees, from Chan [1].

Consider a recursion tree with a cost function c defined on tree nodes such that $c(\nu)$ is the resources spent directly on the call ν, excluding what was spent in ν's recursive subcalls. The cost function c is called α-fading for some constant $\alpha > 0$ if $c(\omega) \leq \alpha c(\nu)$ whenever ω is a child of ν. The cost of the recursion tree is the sum of all node costs; it captures the total amount of resources spent during the execution.

*Partially supported by NSF (grants 0086013 and 0429901) and Darpa/NGA.

Lemma 1 *Let T be a recursion tree with k leaves and an α-fading cost function. If the sum of costs at each level of T is at most C, then the total cost of T is at most $C(\lceil \log_{1/\alpha} k \rceil + \frac{1}{1-\alpha})$.*

Proof. The total cost of the first $\lceil \log_{1/\alpha} k \rceil$ levels is bounded by $C\lceil \log_{1/\alpha} k \rceil$. The remaining costs are bounded by the cost of at most k paths to the leaves that start from level $\log_{1/\alpha} k = -\log_\alpha k$. The cost of the first node on each path is bounded by $C\alpha^{-\log_\alpha k} = C/k$, so the cost of each path is bounded by the geometric series $(C/k)\sum_{i\geq 0} \alpha^i = (C/k)(1-\alpha)^{-1}$. This establishes the lemma. □

3 Computing a maximum-size independent set of intervals

Given a set of intervals, S, we can find the same maximum independent set found by the greedy algorithm without sorting first. The idea is to perform divide and conquer, but to prune subproblems that will not produce a new interval in the solution: find the median of all $2n$ interval endpoints, and partition S into the sets of intervals S_- to the left, S_+ to the right, and $S_|$ spanning the median value. Recursively find $\texttt{MIS}(S_-)$ on the left, and return the coordinate x of the rightmost endpoint in this MIS. Discard any intervals of $S_|$ that span x, then, if $S_|$ is non-empty, find the interval $s \in S_|$ with the leftmost right endpoint. If S_+ is empty, output s; otherwise recursively find $\texttt{MIS}(\{s\} \cup S_+)$. This is detailed in pseudocode as Alg. 1.

Corollary 2 *Function $\texttt{MIS}(S)$ finds a maximum independent set of size k among the n intervals S in $O(n \log k)$ time.*

Proof. Function $\texttt{MIS}(S)$ outputs an interval at every leaf of its recursion tree, so it has at most k leaves. The cost of finding a median and splitting S is $|S|$; intervals are partitioned in the division, so the cost of each level is bounded by $O(n)$. By choosing the median of all endpoints, we guarantee that $|S_-| = |S_+| \leq |S|/2$, since $S_|$ has one endpoint to each side. Since we add at most one segment to S_+ in recursion, the cost is 1/2-fading, and Lemma 1 establishes the bound. □

```
Function MIS(S)outputs MIS & returns the rightmost x coordinate
    find the median of the 2|S| endpoints of |S|.
    partition S into left S_−, crossing S_|, and right S_+ sets.
    if S_− is non-empty
        { x = MIS(S_−); delete from S_| any intervals containing x; }
    delete from S_| all but the segment with leftmost right endpoint, if any.
    if (S_+ is empty)
        if (S_| contains a single segment s)
            { output s; set x = right endpoint of s }
    else x = MIS(S_| ∪ S_+);
return x;
```

Algorithm 1: Function MIS(S) takes a set of intervals S, outputs the intervals of a maximum independent set (MIS) that would be found by the greedy algorithm, and returns the x coordinate of the rightmost point in the MIS.

If we prefer not to implement linear-time median finding for an otherwise simple algorithm, we can choose an interval midpoint at random to serve as the median. The same lemma establishes $O(n \log k)$ expected time.

Corollary 3 *Function* MIS(S)*, modified to choose the midpoint of a random interval as median, finds a k interval MIS among n intervals S in $O(n \log k)$ expected time.*

Proof. The number of leaves and per-level cost bound are as in the previous corollary. When $n = |S|$ elements, the expected maximum of $|S_-|$ and $|S_+|$ after partitioning is

$$\frac{1}{n} \sum_{1 \le i < n} \max\{i, n-i\} \le \frac{2}{n} \sum_{\lfloor n/2 \rfloor \le i < n} i$$
$$\le \frac{1}{n}\big(n(n-1) - \frac{n^2}{4}\big) \le \frac{3}{4}n$$

Thus, the expected time is (3/4)-fading, and Lemma 1 establishes the bound. □

Although for circular-arc graphs a maximum independent set can also be found in linear time after sorting [4, 5], our result cannot be extended. Timothy Chan has pointed out (personal communication) that the element uniqueness lower bound of $\Omega(n \log n)$ applies even to determine if the maximum independent set has $k = 2$ elements: given angles $\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_n \in [0, \pi)$, we could make intervals $(\alpha_i, \alpha_i + \pi)$ and $(\beta_j + \pi, \beta_j + 2\pi)$, which would allow an independent set of 2 intervals iff there exist i and j such that $\alpha_i = \beta_j$.

References

[1] Timothy M. Y. Chan. *Output-sensitive Construction of Convex Hulls.* PhD thesis, Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, 1997.

[2] H. Edelsbrunner and W. Shi. An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem. *SIAM J. Comput.*, 20:259–277, 1991.

[3] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12:459–467, 1982.

[4] Glenn K. Manacher and Terrance A. Mankus. A simple linear time algorithm for finding a maximum independent set of circular arcs using intervals alone. *Networks*, 39(2):68–72, 2002.

[5] S. Masuda and K. Nakajima. An optimal algorithm for finding a maximum independent set of a circular-arc graph. *SIAM J Comput*, 17:41–52, 1988.