Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 39 (2006) 2281 – 2292
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Fast algorithms for finding disjoint subsequences with extremal densities[☆]

Anders Bergkvist[a], Peter Damaschke[b,*]

[a]Department of Molecular Biology, Göteborg University, P.O. Box 462, 40530 Göteborg, Sweden
[b]School of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden

Received 6 July 2005; received in revised form 3 January 2006; accepted 19 January 2006

Abstract

We derive fast algorithms for the following problem: given a set of n points on the real line and two parameters s and p, find s disjoint intervals of maximum total length that contain at most p of the given points. Our main contribution consists of algorithms whose time bounds improve upon a straightforward dynamic programming algorithm, in the relevant case that input size n is much bigger than parameters s and p. These results are achieved by selecting a few candidate intervals that are provably sufficient for building an optimal solution via dynamic programming. As a byproduct of this idea we improve an algorithm for a similar subsequence problem of Chen et al. [Disjoint segments with maximum density, in: International Workshop on Bioinformatics Research and Applications IWBRA 2005, (within ICCS 2005), Lecture Notes in Computer Science, vol. 3515, Springer, Berlin, pp. 845–850]. The problems are motivated by the search for significant patterns in biological data. Finally, we propose several heuristics that further reduce the time complexity in typical instances. One of them leads to an apparently open subsequence sum problem of independent interest.
© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Holes in data; Range prediction; Protein torsion angle; Protein structure prediction; Dynamic programming; Selection algorithms; Time complexity

1. Problem statement and results

Finding large empty regions (big holes) in data sets is interesting in data mining and statistical inference. The problem in its most general form is described by a space X, a family $\mathscr{F}$ of subsets of X, an additive size function that assigns a positive real number to every non-empty set in $\mathscr{F}$, and a finite set D of n data points. The goal is to find the largest set in $\mathscr{F}$ disjoint to D. In a commonly considered case, X is a finite-dimensional real vector space, $\mathscr{F}$ is the family of axis-parallel boxes, and the size of a box is its volume.

It is natural to generalize the problem in two directions: we are interested in several disjoint big holes in X rather than only one largest hole, and we tolerate a limited number of data points in these holes. That is, we introduce two parameters s and p, and look for s disjoint sets with maximum total size which may contain up to p data points.

Already in two dimensions, i.e., for axis-parallel rectangles in the plane, we do not know of complexity results for this parameterized problem. It is even unclear whether the problem is polynomial. However, in the present paper we study the one-dimensional case where $\mathscr{F}$ is the set of intervals contained in a fixed finite-length interval X, and where the size measure is the length of intervals. This one-dimensional case is already interesting for some applications, as we outline in Section 2.

Basically, the one-dimensional problem is easy to solve by dynamic programming, however the question of algorithms being as fast as possible for given n, s, p turns out to be non-trivial. This is the main subject of our paper.

[☆] Part of the results have been presented in preliminary form in the Proceedings of the *16th International Symposium on Algorithms and Computation ISAAC 2005*, Lecture Notes in Computer Science, vol. 3827, Springer, Berlin, pp. 714–723.

* Corresponding author. Tel.: +46 31 772 5405; fax: +46 31 165655.
E-mail addresses: abk@molbio.gu.se (A. Bergkvist), ptr@cs.chalmers.se (P. Damaschke).

0031-3203/$30.00 © 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2006.01.008

Note that an optimal solution always consists of s open intervals with data points at their ends. (The term "open" means that the ends are excluded, i.e., not considered as points of the interval. We also remark that the two outermost intervals of a solution may reach the ends of X.) Therefore we get the following, more convenient formal description. In a sequence of positive real numbers $x_0, x_1, x_2, \dots, x_n$ called *items*, we call any set of consecutive items $x_i, \dots, x_j$ ($i \leqslant j$) an *interval of length* $x_i + \cdots + x_j$. Sometimes we say "item x_i" for "item i with length x_i" if there is no confusion. Obviously, x_i (for $0 < i < n$) means the distance of the ith and $(i+1)$st data point in their natural ordering in X, while x_0 and x_n is the distance of the first and last data point to the left and right end, respectively, of X. Now our problem can be stated as follows:

Disjoint Intervals of Maximum Length (DIMaxL)
Given a sequence $x_0, x_1, x_2, \dots, x_n$ and integers $s \geqslant 1$ and $p \geqslant 0$, find s pairwise disjoint intervals with a total of $s + p$ items and maximum total length.

A batch of remarks is in order here. If $x_i \geqslant 0$ for all i, an optimal solution will always use exactly $s + p$ items, otherwise we could add another item to some interval. Hence, the problem would not change if we wrote "at most $s + p$ items" in the specification. In our formulation with exactly $s + p$ items, the x_i can be arbitrary real numbers, and an optimal solution may have to take some negative items, in order to connect large positive items to intervals. Also, a problem instance does not change if any constant is added to all x_i. On the other hand, an optimal solution may have chains of, say, t incident intervals. In such cases, a chain can be split arbitrarily in t other intervals without changing s, p and the total length. For certain instances, a trivial solution exists and can be verified in $O(n)$ time: if the $s + p$ largest items form (at most) s intervals, this is obviously an optimal solution. In particular, DIMaxL with $p = 0$ is a trivial problem. However, in general the $s + p$ largest items can be scattered in the sequence, and then the optimum is no longer obvious. Parts of an optimal solution may consist of medium-length items that are clustered, and then we have to find the best among many candidates.

We also mention the relation to independent sets in interval graphs. (We assume some background in graph algorithms. However, the reader may skip this paragraph. The statements will not be used further, we just had to make sure that DIMaxL has not already been considered in that context, under a different name.) Given a sequence of numbers, consider the interval graph $G(p)$ with intervals of at most p items as vertices, where two vertices are adjacent if the intervals intersect. We sloppily identify intervals and vertices in $G(p)$. DIMaxL can be rephrased as a special case of finding a maximum weight independent set of s vertices in an interval graph which is given as an interval model. Although Independent Set and related problems in interval graphs and other intersection graphs are well studied (we refer to Ref. [1] for some special results and more pointers to the vast literature), we cannot directly make use of earlier results, since we have to apply the special features of our instances to make our algorithms as fast as possible. In our particular case, every vertex of an interval graph $G(p)$ belongs to at most p consecutive cliques, and the vertex weights are the interval lengths in the interval model. To our best knowledge there is no previous work on Weighted Independent Set in interval graphs with these special restrictions.

Organization of the paper and results: In Section 2, we motivate DIMaxL by statistical inference tasks. Our particular bioinformatics motivation is discussed in more detail, but there may be more, given that our approach is rather general and conceptually simple. Section 3 is the core of the paper: we develop our basic algorithms for DIMaxL. A naive dynamic programming algorithm solves DIMaxL in $O(spn)$ time. By some preprocessing that selects possible intervals for an optimal solution, without missing any candidate, we can achieve a time bound $O(pn + s^2p^3)$. Since we have $s, p \ll n$ in statistically relevant cases, this alternative algorithm is then an improvement. As a byproduct we also improve an algorithm for a similar numerical subsequence problem: define the density of an interval $x_i, \dots, x_j$ to be the ratio $(x_i + \cdots + x_j)/(j - i + 1)$. Let us call the following problem that has been introduced and attacked in Ref. [2].

Disjoint Intervals of Maximum Density (DIMaxD)
Given a sequence $x_1, x_2, \dots, x_n$ and integers $k \geqslant 1$ and $l \geqslant 1$, find k disjoint intervals, each with *at least l* items, so as to maximize the sum of their densities.

(Loosely speaking, our problem DIMaxL is looking for low-density regions in point sets, opposite to DIMaxD.) Chen et al. [2] gave an $O(kln)$ time algorithm, as well as algorithms running in time $O(n)$ and $O(n + l^2)$ in the special cases $k = 2$ and 3, respectively. They explicitly asked if there is an $o(kln)$ algorithm for general k. We give an affirmative answer. We can solve DIMaxD in $O(ln + k^2l^2)$ time, using the same idea of candidate selection followed by dynamic programming on the small candidate set. Actually, this scheme would be applicable to any similar problems where one has to find a set of disjoint subsequences that optimizes some objective function. The main property we need from the problem is that the candidate intervals are of limited length.

In Section 4 we further elaborate on our approach for DIMaxL. For different ranges of parameters s, p we can further reduce the "n-free" term of our time bound. Although these extra improvements are small (third root of a parameter), we find the matter interesting from the algorithmic point of view: better time bounds are achieved by suitable combinations of the two basic algorithms. While the analysis becomes somewhat tricky, implementation is not harder. Basically we treat short and long candidate intervals differently. We also expect that our current bounds are not yet the ultimate ones.

On the other hand, it is not clear whether the pn term can be reduced, too. In Section 5, we propose a few heuristics to