Densities and Partitions

Sharon Curtis
Oxford Brookes University

11th May 2009
**Segment Density**

Elements in the input sequence have

\[
width : \text{Elem} \to \mathbb{R}^+ \quad \text{area} : \text{Elem} \to \mathbb{R}
\]

For a non-empty segment \(\text{seq}[i..j]\),

\[
density(\text{seq}[i..j]) = \frac{\text{area}(\text{seq}[i..j])}{\text{width}(\text{seq}[i..j])} = \frac{\sum_{k=i}^{j} \text{area}(\text{seq}[k])}{\sum_{k=i}^{j} \text{width}(\text{seq}[k])}
\]
Maximum Segment Density Problem

For an input sequence $seq[1..n]$, a maximally dense segment $seq[i..j]$ is one that maximises $density(seq[i..j])$, for $1 \leq i \leq j \leq n$.

...subject to a lower bound constraint $L$ on the width of the segments considered:

$$density(seq[i..j]) \geq L$$

and (optionally) an upper bound constraint $U$ too:

$$density(seq[i..j]) \leq U$$
Densities and DNA Analysis

The maximum segment density problem can be applied to the analysing of DNA sequences, e.g. to search for

- regions of DNA rich in GC pairs; simply count $area = 1$ for a G or C nucleotide and $area = 0$ otherwise

- high frequency of occurrence of certain patterns; set the $area$ equal to the frequency of the pattern within a region of DNA

- regions of DNA with a high density of mutations; this can give information about the shape of the DNA in such regions
History of the MSD problem

In the case that $L = U$, the MSD problem is trivially solvable in $O(n)$, using a “sliding window” technique, where the window is of fixed width.

There have been various attempts at algorithms for MSD with more than linear complexity; a recent one being the $O(n \log L)$ algorithm of Lin, Jiang & Chao [LJC02].

However, Chung & Lu [CL04] produced an $O(n)$ algorithm, and so did Goldwasser, Kao & Lu [GKL05].

Shin Cheng-Mu and I have been looking at these last two algorithms to better mathematically formalise algorithm design patterns for this and similar problems.


**Sliding Window Technique for Segment Problems**

A sliding window technique from right-to-left is based on an ordinary design step for segment optimisation problems: every segment is a prefix of a suffix.

The idea is to note a best prefix of each suffix of the input, and then output a best of those to return an overall best segment.

To make this strategy efficient, a window is used. A window is a prefix of the currently-being-considered suffix of the input:
A best prefix for the currently-being-considered suffix of the input can be found by only considering prefixes of the window (if the structure of the problem permits).

With each step, the window is updated: its leading edge moves along by one element, and its trailing edge moves (maybe) some places along to the left. But the edges of the window always move forwards, never backwards.

If the window can be updated in (amortized) constant time and the best segment obtained from it in constant time, then the overall algorithm is linear.
Sliding Window Technique for Maximum Segment Density

The [CL04] and [GKL05] solutions to the MSD problem both use a similar windowing technique.

However, one difference is that the best prefix returned from the window is not necessarily a best prefix of the portion of the input considered so far.

Also we can say a little more about the structure of the window: some elements (lhs) must always be included in a returned prefix as there is a lower bound \( L \) on segment width.

The focus is on the right-hand side of the window: how do we know what prefix of rhs gives us a maximally dense prefix of the whole window?
Not the densest prefix of $rhs$ - because the $density$ function is not monotonic! It depends on the $lhs$ too.

Obtaining a maximally dense prefix of the whole window depends crucially on an interesting kind of partition...
Decreasing Right-Skew Partitions

A *decreasing right-skew partition* (DRSP) is a partition of a sequence into non-empty segments, such that:

- The segments of the partition are of strictly decreasing density.
- Every segment in the partition is *right-skew*. A segment \( \text{seq}[i..j] \) is right-skew iff for every \( i < j \leq k \), \( \text{density}(\text{seq}[i..j - 1]) \leq \text{density}(\text{seq}[j..k]) \).
Densities and Partitions

Such a partition always exists, and is unique.

$DRSP$s can be constructed from a sequence by repeatedly taking the longest right-skew prefix ($lrsp$).

Alternatively - neat bit! - $DRSP$s can be constructed from a sequence by repeatedly taking the longest highest-density prefix ($lhdp$), because $lhdp = lrsp$.

Alternatively, $DRSP$s can be constructed from a sequence by repeatedly taking the longest lowest-density suffix ($llds$).

Alternatively, $DRSP$s can be constructed from a sequence by repeatedly taking the longest right-skew suffix ($lrss$).

So how does knowing the $DRSP$ help with $MSD$?
**Decreasing Right-Skew Partitions and MSD**

Consider the *DRSP* of the right-hand side of the window, e.g.

At least one maximally dense prefix of the whole window will terminate at one of the boundaries of the *DRSP* segments.

From the right-hand side of the window, a partition segment is repeatedly removed, until this would no longer increase the density of the whole window, or until we run out of right-hand side of window.

[Interesting: the dominance relationships between segments do not form a typical monotonicity condition such as that used in dynamic programming.]

\[^1\) a \leq \text{kind of increase not a strict}<\]
[GKL05] base their algorithm on $DRSP$s and pre-process the input sequence to get all necessary $DRSP$ information in advance.

[CL04] base their algorithm on repeatedly taking longest lowest-density suffixes and updating their partition structure as they process the input, but at heart, the algorithms are the same!

The key to efficiency rests on maintaining an data structure to store the $DRSP$ for the right half of the window, and efficient updating of the structure. This involves doing both the following in amortized $O(1)$ time:

- adding single elements to the $DRSP$ data structure as the leading edge of the right-hand side of the window moves left;

- chopping off sections at the trailing edge of the window if the upper bound on segment width $U$ is exceeded.
Properties of Density

Not many properties of the density function are required, the success of these algorithms rests entirely on

- Densities of segments being quickly calculated from stored or accumulated values (in this case, sums of areas and widths of segments)

- The density property: for any sequences of elements $x$ and $y$,

  \[ d \ x \oplus d \ (x \oplus y) \equiv d \ (x \oplus y) \oplus d \ y \equiv d \ x \oplus d \ y \]

  where $d = \text{density}$, and $\oplus$ could be $<$, $\leq$, $>$ or $\geq$. 
References

