Partitioning Number Sequences into Optimal Subsequences

J. Zobel
Department of Computer Science, RMIT University
GPO Box 2476V, Melbourne 3001, Australia
jz@cs.rmit.edu.au

P. Dart
Department of Computer Science and Software Engineering
The University of Melbourne, Parkville 3052, Australia
philip@cs.mu.oz.au

Abstract

We consider how to partition finite sequences of positive numbers into subsequences such that each resulting subsequence has a sum of at least a given minimum. Given several different optimality criteria, based on maximising the number of subsequences and minimising their variance in size, we develop and analyse a series of algorithms that yield optimal solutions.

Keywords: algorithms, analysis of algorithms, computational complexity.

1 Introduction

Consider a finite sequence of positive numbers that is to be partitioned into subsequences as follows. Each subsequence must contain numbers whose sum is at least L, where L is fixed; the variation between the sums is to be kept small; and the sums should be close to L. That is, the criteria for the partitioning are that the number of sums should be maximised and the variance should be minimised.

The problem of partitioning arose from our work in information retrieval, where it can be desirable to divide documents consisting of small units of text (such as sentences or paragraphs) into larger units of consistent length, for retrieval or presentation [2, 5]. First, enforcing a minimum length helps ensure that there is sufficient content to allow accurate matching of queries and documents. Second, when units of text significantly vary in length, estimation of their relevance to a query is inaccurate, to it is desirable to reduce the variance. Partitioning can also be used for division of irregularly-sized data such as web pages into packets for network transmission, where there is a trade-off between response time and total cost of transmission.

Similar problems have been described by other authors. Larmore and Hirschberg have considered how to break scrolls into pages whose lengths must fall within a certain range [4]. In contrast to partitioning, any successful pagination is satisfactory. Knuth and Plass have comprehensively discussed the problems presented by breaking paragraphs into lines with acceptable inter-word spacing [3], a far more complex problem than partitioning, for not only can the size of the items in a line be varied within certain limits, but the items can be divided with a hyphen between two lines. Another related problem is bin packing, in which items are to be placed in the minimum number of bins of some fixed maximum capacity [1]; however, packing does not have a constraint on the order of items.

2 Partitioning

Partitioning can be described formally as follows. Given a *source* sequence, that is, a sequence of positive numbers

$$W = \langle w_1, \ldots, w_n \rangle ,$$

we wish to coalesce adjacent numbers in the sequence to form a *target* sequence, that is, a new sequence of positive numbers

$$T = \langle t_1, \ldots, t_m \rangle .$$

A target is such that each $t_i \geq L$, where L is a fixed lower bound satisfying $\sum_{j=1}^{n} w_j \geq L > 0$, and the target's elements are defined by

$$t_i = \sum_{j=k_{i-1}+1}^{k_i} w_j ,$$

where $k_{i-1} < k_i$, $k_0 = 0$ and $k_m = n$. Thus the ith subsequence consists of numbers $w_{k_{i-1}+1}, \ldots, w_{k_i}$ and has sum t_i.

In addition, we wish to choose $k_1, \ldots, k_{m-1}$ to minimise the average distance between each t_i and L. A suitable function for minimising the average distance between t_i and L is the variance from L given by

$$\mathit{variance}(T) = \frac{1}{m}\sum_{i=1}^{m}(t_i - L)^2 \,.$$

Before we describe our algorithms for finding the optimal target, we consider some of the difficulties presented by partitioning. One is that local optimality does not imply global optimality. For example, consider a source

$$\langle 10, 1, 9, 2, 8, 3, 7, 4\rangle \,,$$

where L is 10. Two targets are $\langle 10, 10, 10, 14\rangle$ and $\langle 11, 11, 11, 11\rangle$. The first target is locally better at the left-hand end, but the second is optimal. Thus an algorithm for finding an optimal target cannot proceed by accruing numbers in the source and choosing the optimal partial target.

Another difficulty presented by partitioning is that the target with lowest variance may not be the longest possible. An example source with this property is

$$\langle 10, 5, 5, 9, 9, 5, 5, 9, 9, 5, 5, 9, 9, 5, 5, 9, 9, 5, 5, 9, 9, 5, 5, 10\rangle \,.$$

where L is 10. Two targets are

$$\langle 10, 10, 18, 10, 18, 10, 18, 10, 18, 10, 18, 10, 10\rangle$$

and

$$\langle 15, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15\rangle \,.$$

The latter is shorter, but has lower variance. However, it is difficult to construct such sources, and we believe them to be very rare. Note that if the source is shortened by omission of a $\langle 5, 5, 9, 9\rangle$ subsequence, the longer target has the lower variance; that is, the source given is the shortest (of that pattern) that results in lower variance for the shorter target.

3 Optimising for variance

The optimum target, with regard to any optimality criteria, can be found exhaustively as follows. Any given target can be written as

$$T = \langle w_1 + \ldots + w_{k_1}, w_{k_1+1} + \ldots + w_{k_2}, \ldots, w_{k_{m-1}+1} + \ldots + w_{k_m}\rangle \,.$$

From this form it can be seen that all possible targets can be found by enumerating combinations of sums, and that there are 2^{n-1} different combinations; note, however, that some of these combinations can contain some $t_i < L$ and are therefore not targets.

```
1. Set $S_1 \leftarrow \{\langle w_1 \rangle\}$.
2. For each $j$ from 2 to $n$,
    (a) Create an empty set $S_j$.
    (b) For each $l \in S_{j-1}$,
        i. Add $l \cdot w_j$ to $S_j$.
        ii. Add $l + w_j$ to $S_j$.
3. Set $S \leftarrow \{l \in S_n | v \geq L$ for each value $v$ in $l\}$.
4. Choose from $S$ the sequence $l \in S$ with minimum variance.
```

Figure 1: *Algorithm A—naive optimisation for variance.*

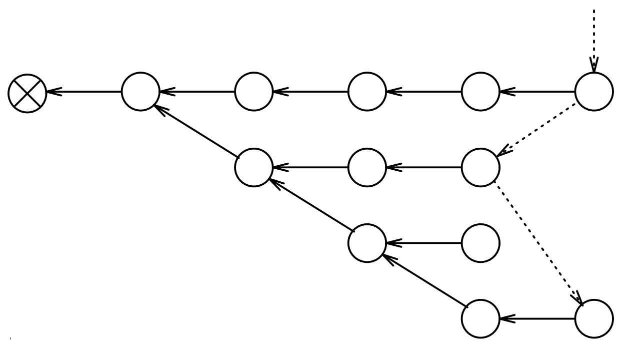

Figure 2: *Linked list "fan" structure for storing sequences.*

We use the following notation. Suppose $l = \langle v_1, \ldots, v_k \rangle$ is a sequence and v is a value. Then a *composition* $\langle v_1, \ldots, v_k + v \rangle$ is denoted by $l + v$; an *extension* $\langle v_1, \ldots, v_k, v \rangle$ is denoted by $l \cdot v$; the length k of l is denoted by *len*(l); and the last value v_k in l is denoted by *last*(l).

The set S of targets can be enumerated by the method shown in Figure 1. Input are n and a source $\langle w_1, \ldots, w_n \rangle$. We assume, as in all the algorithms in this paper, that $\sum_{j=1}^{n} w_j \geq L$. That is, we assume there is a valid target.

A simple way to implement Algorithm A (and Algorithms B and C, described later) is to have a linked structure as follows. Each t_k value requires a node with fields containing: its value; k; the sum $\sum_{i=1}^{k} (t_i - L)^2$; a pointer to t_{k-1}; and a pointer to the last t value in another sequence. The inter-sequence pointers give a linked list joining the ends of all the candidate sequences together. Candidate sequences can be reconstructed by following intra-sequence pointers from the end of the sequence to the start. To simplify implementation, a sentinel node should be placed before w_1. This "fan" structure is illustrated in Figure 2.

In Figure 2, the dashed lines are the inter-sequence pointers and the crossed node is the sentinel. One sequence is unreachable, because there is no pointer to the last node; although not possible with Algorithm A, the other algorithms can create such a structure.

Using the fan structure, finding the optimal sequence—the sequence with the lowest variance—is straightforward: the information stored in the last node in each sequence can be used to derive the variance, and the last nodes are connected by the inter-sequence pointers. The fan structure itself is desirable because any leading subsequence that is common to several sequences is stored once only.

Assuming the fan implementation, we can derive the following properties.

Theorem 1 Algorithm A has space and time complexity of $O(2^n)$.

Proof At step 2b of Algorithm A, two sequences are added to S_j for each sequence in S_{j-1}, so that $|S_j| = 2|S_{j-1}|$. Given the fan implementation, the additional cost in space at each j is linear in the current number of sequences. The result follows. $\square$

Theorem 2 For any source W and the set S yielded by application of Algorithm A to W, the target in S with the lowest variance will be optimal.

Proof The result follows from the observation that all targets are in S. $\square$

We now show how Algorithm A can be modified to yield the more efficient Algorithm B, described below. The inefficiencies in Algorithm A can be addressed using the following propositions.

Proposition 1 Consider S_j, for any j where $1 \leq j \leq n$, derived by Algorithm A. For any sequence $l = \langle t_1, \ldots, t_{k-1}, t_k \rangle \in S_j$ such that $t_{k-1} < L$, no sequence in S_n derived from l, by composition or extension, is a target.

Proof Straightforward. $\square$

Thus subsequences that cannot lead to targets can be progressively eliminated, by modifying step 2(b)i of Algorithm A such that $l \cdot w_j$ is added to S_j only if $\mathit{last}(l) \geq L$. It can be seen by induction that, for any j and any sequence $\langle t_1, \ldots, t_k \rangle \in S_j$, each $t_i \geq L$ for $1 \leq i < k$. Step 3 can then be modified to keep only the sequences whose last element is at least L.

Proposition 2 Consider two targets derived by Algorithm A,

$$\begin{aligned} T &= \langle t_1, \ldots, t_k + w_j + v, t'_1, \ldots, t'_{k'} \rangle \in S \text{ and} \\ T' &= \langle t_1, \ldots, t_k, w_j + v, t'_1, \ldots, t'_{k'} \rangle \in S\,. \end{aligned}$$

Then the variance of T' is less than the variance of T.

Proof Straightforward. $\square$

```
1. Set $S_1 \leftarrow \{\langle w_1 \rangle\}$.
2. For each $j$ from 2 to $n$,
    (a) Create an empty set $S_j$.
    (b) For each $l \in S_{j-1}$,
        i. If $last(l) < L$ or $w_j < L$, add $l + w_j$ to $S_j$.
        ii. If $last(l) \geq L$, add $l \cdot w_j$ to $S_j$.
3. Set $S \leftarrow \{l \in S_n | last(l) \geq L\}$.
4. Choose from $S$ the sequence $l \in S$ with minimum variance.
```

Figure 3: *Algorithm B—optimisation for variance.*

It follows that T—or, if $w_j \geq L$, even its leading subsequence $l = \langle t_1, \ldots, t_k + w_j \rangle$—need never be created, since for any target derived from l there will be a target with lower variance derived from $\langle t_1, \ldots, t_k, w_j \rangle$. Sequences such as l can be avoided by not applying composition when $last(l) \geq L$ and $w_j \geq L$.

The algorithm can now be restated as Algorithm B, shown in Figure 3, and again the fan implementation should be used. As for Algorithm A, the sequence in S with the lowest variance will be the optimal target. The complexity is as follows.

Theorem 3 Algorithm B has space and time complexity of $O(2^n)$.

Proof The result follows from the observation that in the worst case Algorithm B generates the same sequences as Algorithm A, and never generates more sequences. □

Although the upper bound on the cost of Algorithm B is the same as that of Algorithm A, Algorithm B is in general cheaper, particularly when the source contains elements that are at least L. Nonetheless, for some sources Algorithm B has cost $O(2^n)$.

4 Optimising for length and variance

We can use length and variance constraints to refine Algorithm B in accordance with the following proposition, to yield an Algorithm C that produces targets of maximum length, and of these targets, chooses the target with minimum variance.

Proposition 3 Consider two sequences

$$l = \langle t_1, \ldots, t_k \rangle \in S_{j-1} \text{ and}$$
$$l' = \langle t'_1, \ldots, t'_{k'} \rangle \in S_{j-1}\,,$$

where $t_k \geq L$ and $t'_{k'} \geq L$.

1. For any target
$$T = \langle t_1, \ldots, t_k, w_j + v, u_1, \ldots, u_p \rangle$$
that contains l as a leading subsequence, there is a target
$$T' = \langle t'_1, \ldots, t'_{k'}, w_j + v, u_1, \ldots, u_p \rangle$$
that contains l' as a leading subsequence.
2. If $k < k'$ then T will be shorter than T'.
3. If $k = k'$ and $\mathit{variance}(l) > \mathit{variance}(l')$, then $\mathit{variance}(T) > \mathit{variance}(T')$.
4. If $k = k'$ and $\mathit{variance}(l) = \mathit{variance}(l')$, then $\mathit{variance}(T) = \mathit{variance}(T')$.

Proof Straightforward. □

Thus, when choosing sequences in S_{j-1} for extension, only the sequence with maximal length and minimal variance need be considered at step 2(b)ii.

Similarly, step 2(b)i of Algorithm B can be refined as follows. If $l \in S_{j-1}$, $\mathit{last}(l) \geq L$, and $w_j \geq L$, then, as discussed above, for each target T that can be derived from $l + w_j$, a target T' with lower variance can be derived from $l \cdot w_j$. Moreover, T will be shorter than T'. Furthermore, suppose that S_{j-1} contains two sequences $l = \langle t_1, \ldots, t_k \rangle$ and $l' = \langle t'_1, \ldots, t'_{k'} \rangle$, where $t_k \geq L$, $t'_{k'} \geq L$, and $k < k'$. From the above proposition, for each target that can be derived from l, a longer target can be derived from l', so we do not need to consider composition to l.

Using the refinements, an algorithm that optimises for length and variance can be stated as in Figure 4.

Theorem 4 Consider the set S returned by Algorithm C.

1. Each sequence in S is a target of maximal length.
2. The target of maximal length and, for that length, minimal variance, is in S.

Proof The result follows from Proposition 3 and from case analysis of the algorithm. □

To analyse the complexity of Algorithm C we first need two lemmas.

Lemma 1 In Algorithm C, $|S_j| \leq |S_{j-1}| + 1$.

Proof In step 2e of Algorithm C, at most one sequence is added to S_j for each sequence in S_{j-1}, and only one sequence is added to S_j at step 2d. No other steps add sequences. □

Lemma 2 In Algorithm C, if two adjacent source elements w_{j-1} and w_j are both at least L, the set S_j has just one element.

```
1. Set $S_1 \leftarrow \{\langle w_1 \rangle\}$.
2. For each $j$ from 2 to $n$,
    (a) Create an empty set $S_j$.
    (b) Set $M \leftarrow \{l \in S_{j-1} | last(l) \geq L\}$.
    (c) Let $K$ be the length of the longest sequence in $M$.
    (d) If $M' = \{l \in M | len(l) = K\}$ is not empty, choose an arbitrary sequence $l \in M'$
        such that there is no sequence $l' \in M'$ with lower variance, and add $l \cdot w_j$ to $S_j$.
    (e) For each $l \in S_{j-1}$,
        i. If $last(l) < L$ add $l + w_j$ to $S_j$.
        ii. If $last(l) \geq L$ and $len(l) = K$ and $w_j < L$ then add $l + w_j$ to $S_j$.
3. Set $S \leftarrow \{l \in S_n | last(l) \geq L\}$.
4. Choose from $S$ the sequence $l \in S$ with minimum variance.
```

Figure 4: *Algorithm C—optimisation for length and variance.*

Proof Neither option of step 2e applies, and step 2d adds exactly one candidate. □

We say that a sequence is *minor* if it does not contain two adjacent elements of at least L.

Theorem 5 Algorithm C has time complexity of $O(ns)$, where s is the length of the longest minor subsequence in the input source.

Proof The result follows from Lemmas 1 and 2. □

Algorithm C should be implemented with the following modifications to the fan structure. At step 2d, a new node for w_j must be created, and must point to the node for the last element in l; the node for this element must be marked as referenced. At step 2e, each value $last(l)$ can be replaced by $last(l) + w_j$, except for the single value of $last(l)$ that was marked as referenced, in which case a node for $last(l) + w_j$ must be created, with a pointer to $last(l)$'s predecessor.

Theorem 6 Algorithm C has space complexity of $O(n)$.

Proof Using the fan structure, at most two nodes are added for each element in the source. □

5 Optimising for length alone

Algorithm D, shown in Figure 5, is linear in time and space and emits the solution as it proceeds. The variables *prev* and *curr* refer to the previous and current target sequence entries respectively.

```
1. Set $prev \leftarrow undefined$ and $curr \leftarrow w_1$.
2. For each $j$ from 2 to $n$,
    (a) If $curr \geq L$, emit $prev$ if it is defined, set $prev \leftarrow curr$, and set $curr \leftarrow w_j$.
    (b) Otherwise, if $w_j > prev$ then set $prev \leftarrow prev + curr$ and set $curr \leftarrow w_j$.
    (c) Otherwise, set $curr \leftarrow curr + w_j$.
3. If $curr < L$ then set $prev \leftarrow prev + curr$ and emit $prev$. Otherwise, emit $prev$ followed
   by $curr$.
```

Figure 5: *Algorithm D—optimisation for length.*

Theorem 7 Algorithm D has $O(n)$ time complexity, requires $O(n)$ space to store the solution but only $O(1)$ working space.

Proof The algorithm requires at most four comparisons and one sum for each element of the source, and only *curr* and *prev* are stored. □

Moreover, Algorithm D emits partial solutions as it proceeds, whereas the other algorithms do not emit any solution until the whole source has been inspected.

Theorem 8 Algorithm D always generated targets of the maximum length.

Proof Step 2 generates leading subsequences in which, including the values of *prev* and *curr*, all but the last element is at least L. Using induction on j, it can be shown that, for a source of any length n, the partial target generated by step 2 is of the longest possible length, and that if $curr < L$ then *curr* is at least as large as the last element in any partial target of the same length. That the target generated by Algorithm D is of maximal length follows immediately. □

However, the targets computed by Algorithm D do not necessarily have minimal variance for their length. We believe that there is no linear time algorithm that yields such targets, because optimisation of variance cannot be determined locally.

6 Summary

We have described four algorithms for computing an optimal partitioning of a source into a target, for three different optimality criteria. The first two algorithms, A and B, minimise variance, at a complexity of $O(2^n)$ for a source of length n; we believe it unlikely that there is an algorithm that minimises variance at lower complexity. The third algorithm, C, minimises variance at maximum target length and has space complexity of $O(n)$ and time complexity of $O(ns)$, where s is the length of the longest minor subsequence in the source. Thus the worst case is a source for which all elements are less than L, giving a complexity of $O(n^2)$—a marked improvement over the first two algorithms. Given our observation that almost all targets of minimum variance are of maximum length, it follows that Algorithm C is an excellent heuristic method for minimising variance alone. The final algorithm, D, maximises target length and has space and time complexity of $O(n)$. In the application of interest to us—dividing long documents into pages—there were sources of over ten thousand elements. In experiments with a large collection of documents represented as sequences of elements, Algorithms A and B were impossible to use because of the space requirements, while Algorithm C was acceptable but occasionally slow. Algorithm D took less than a second and yielded no solutions with unacceptable variance, but in some documents the variance was much greater than in the solutions produced by the other algorithms.

Acknowledgements

We wish to thank Alistair Moffat, James Thom, and Nick Wormald. This work was supported by the Australian Research Council and the Collaborative Information Technology Research Institute in Melbourne, Australia.

References

[1] M.R. Garey and D.S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979.

[2] M. Kaszkiel, J. Zobel, and R. Sacks-Davis. Efficient passage ranking for document databases. *ACM Transactions on Information Systems*, 17(4):406–439, October 1999.

[3] D.E. Knuth and M.F. Plass. Breaking paragraphs into lines. *Software—Practice and Experience*, 11:1119–1184, 1981.

[4] L.L. Larmore and D.S. Hirschberg. Efficient optimal pagination of scrolls. *Communications of the ACM*, 28(8):854–856, 1985.

[5] J. Zobel, A. Moffat, R. Wilkinson, and R. Sacks-Davis. Efficient retrieval of partial documents. *Information Processing & Management*, 31(3):361–377, 1995.