Algorithmic Complexity:
Polynomially Subquadratic Algorithms for Min-Sum Convolution Problems

My research in computer science deals with min-sum (aka tropical) convolution problems.

**Definition 1 (Min-Sum Convolution)** Let $a := \{a_n\}_{n=0}^N$ and $b := \{b_n\}_{n=0}^N$ be real valued $N+1$ dimensional vectors. The min-sum convolution $c := \{c_n\}_{n=0}^{2N}$ is given by

$$c_n := \min_{i+j=n} a_i + b_j.$$  

It is generally unrealistic from a computational perspective to work with arbitrary real vectors. Thus for concreteness, before discussing general problems, past work, or solution strategies, I will define the specific min-sum convolution problem on which I am currently working in conjunction with Leonard Schulman (Caltech):

**Problem 1 (Linearly Bounded Min-Sum Convolution)** Let $a := \{a_n\}_{n=0}^N$ and $b := \{b_n\}_{n=0}^N$ be $N+1$ dimensional vectors with the restriction that for all $n$, $a_n$ and $b_n$ are (not necessarily distinct) integers in the interval $[0, O(N)]$. Define the min-sum convolution $c := \{c_n\}_{n=0}^{2N}$ by

$$c_n := \min_{i+j=n} a_i + b_j.$$  

Find some $\delta > 0$ and an algorithm of expected time complexity $O(N^{2-\delta})$ to determine $c$.

I will denote the input vectors by $a$ and $b$ and their min-sum convolution by $c$ for the remainder of the document. Naively this problem can be solved with $\Theta(N^2)$ computations, using the following pseudo code:

```
1  For (n from 0 to 2N)
2      Let $c_n := a_0 + b_k$
3  For (k = 1 to n)
4      $c_n = \min(c_n, a_k + b_{n-k})$
```

The $\Theta(N^2)$ complexity is immediate from the algorithm as is its correctness, however it is by no means optimal.

According to [1], the fastest known algorithm to date is in [3], with expected running time $O(N^2/2^{\Omega(\sqrt{\log N})})$. More succinctly, this is substantially faster than the naive $N^2$ complexity, but not quite polynomially faster. We began investigating this problem in the wake of a recent breakthrough on a special case, when $a$ and $b$ are (weakly) increasing. In [1], the authors establish a
deterministic algorithm with worst case running time polynomially faster than \( O(N^2) \) and a randomized algorithm with slightly faster expected running time than that. For concreteness, both algorithms in [1] run in time between \( N^{1.8} \) and \( N^{1.9} \) up to constants.

Monotonicity is a strong assumption on input vectors and is not easily disentangled from the algorithms of [1], however there are heuristic reasons that Leonard and I are optimistic that our more general problem is feasible. To see this, note that the naive algorithm requires \( \Theta(N^2) \) computations, but only \( \Theta(N) \) are directly necessary for min-sum convolution. In other words, if someone kindly handed me a list of \( 2N + 1 \) pairs \( \{(i_n, j_n)\}_{n=0}^{2N} \) such that \( i_n + j_n = n \) and \( a_{i_n} + b_{j_n} \) is minimal in the set \( \{a_i + b_j\}_{i+j=n} \), then I could calculate the min-sum convolution in time \( \Theta(N) \) because it could simply be

\[
\{a_{i_n} + b_{j_n}\}_{n=0}^{2N}.
\]

Of course, the catch is that I do not know exactly where to look for the \( a \) and \( b \) components that will actually show up in the min-sum convolution. But it is not a complete mystery. For instance the minimum of \( a \) plus the minimum of \( b \) will have to show up somewhere in the min-sum convolution. Conversely, I will almost never have to add the maximum of \( a \) with the maximum of \( b \).

To find a solution to our min-sum problem, we are following the intuition underlying the field of additive combinatorics ([2] is an excellent reference). Imagine that \( a \) and \( b \) are (generally two different) permutations on the set \([0, N]\). This is a strong assumption, but it is a good guide for general cases and would be an interesting case to solve in itself. Define

\[
I := \{i \in [0, N] : a_i \leq N^{1/2}\} \quad \text{and} \quad J := \{j \in [0, N] : b_j \leq N^{1/2}\}
\]

and the sum set

\[
I + J := \{i + j : i \in I, j \in J\}.
\]

If an index \( n \) lies in \( I + J \), (i.e. there exists \( i \in I, j \in J \) such that \( n = i + j \)) then I know immediately

\[
c_n \leq a_i + b_j \leq 2N^{1/2}
\]

and therefore, if \( a_{i_n} + b_{j_n} = c_n \), then either \( a_{i_n} \leq 2N^{1/2} \) or \( b_{j_n} \leq 2N^{1/2} \) (or both). We generally expect that because \( I \) and \( J \) have at least \( N^{1/2} \) elements, the sum set \( I + J \) should have about \( |I||J| \), i.e. \( \Theta(N) \), elements. In fact if \( a \) and \( b \) are independently chosen randomly from permutations (or even from linearly bounded vectors), with high probability it will be the case that \( |I+J| = \Theta(N) \). If this is the case, then by doing only \( 4N \) computations, namely adding the lowest \( 2N^{1/2} \) values in \( a \) with the lowest \( 2N^{1/2} \) values in \( b \), calculate \( \Theta(N) \) components of \( c \). If I could continue finding components of \( c \) at that rate (which is far too good to hope for), I could calculate \( c \) in \( O(N) \) time (which is far better than we are aiming for).

Certainly there is no guarantee that \( |I + J| \) is anywhere close to \( 2N + 1 \). For instance, if \( a \) and \( b \) were both sorted, \( I \) and \( J \) would simply be continuous
intervals and their sum set would have only $|I| + |J| - 1$ elements. Alternatively, if every index in $I$ and $J$ both consist of the multiples of $\sqrt{N}$ (assume $N$ here is a square for simplicity), their sumset will again have $|I| + |J| - 1$ elements.

As a field, additive combinatorics is essentially one big collage of answers to the question, "If $I + J$ is that are a lot smaller than $|I||J|$, what can we say about the structure of $I$, the structure of $J$, and the relationship between them. So, while it would be inaccurate to say that I only need to add the lowest components in $a$ with the lowest components in $b$ (in some appropriately quantified sense), it might possible to say that either we save time this way, or else the low values have to be so arithmetically structured that we can divide the problem up recursively into smaller (again, in some appropriately quantified sense) min-sum convolution problems.

In short, Leonard and my overarching goal is to find an algorithm to recursively break down a min-sum convolution problem into "smaller" problems until they can either be efficiently handled by dealing only with the lowest components. If we are able to tackle our main problem, we hope to consider generalizing to

- $a$ and $b$ that are in a polynomial range (with respect to $N$) and
- $a$ and $b$ indexed by elements of other abelian groups, interpreting the definition $c_n = \min_{i+j=n}$ in terms of addition in the chosen group.

That said, both Leonard and I would be thrilled just to find a polynomially subquadratic algorithm for our current problem.

References

