Necklaces, Convolutions, and $X + Y$

David Bremner[1⋆], Timothy M. Chan[2⋆], Erik D. Demaine[3⋆⋆], Jeff Erickson[4], Ferran Hurtado[5], John Iacono[6⋆⋆], Stefan Langerman[7], and Perouz Taslakian[8]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada, bremner@unb.ca
[2] School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, tmchan@uwaterloo.ca.
[3] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, edemaine@mit.edu.
[4] Computer Science Department, University of Illinois, Urbana-Champaign, IL, USA, jeffe@cs.uiuc.edu
[5] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, Ferran.Hurtado@upc.es
[6] Department of Computer and Information Science, Polytechnic University, Brooklyn, NY, USA, http://john.poly.edu.
[7] Chercheur qualifié du FNRS, Départment d'Informatique, Université Libre de Bruxelles, Brussels, Belgium, stefan.langerman@ulb.ac.be.
[8] School of Computer Science, McGill University, Montréal, Québec, Canada, perouz@cs.mcgill.ca

Abstract. We give subquadratic algorithms that, given two necklaces each with n beads at arbitrary positions, compute the optimal rotation of the necklaces to best align the beads. Here alignment is measured according to the ℓ_p norm of the vector of distances between pairs of beads from opposite necklaces in the best perfect matching. We show surprisingly different results for $p = 1$, $p = 2$, and $p = \infty$. For $p = 2$, we reduce the problem to standard convolution, while for $p = \infty$ and $p = 1$, we reduce the problem to $(\min, +)$ convolution and $(\text{median}, +)$ convolution. Then we solve the latter two convolution problems in subquadratic time, which are interesting results in their own right. These results shed some light on the classic sorting $X + Y$ problem, because the convolutions can be viewed as computing order statistics on the antidiagonals of the $X + Y$ matrix. All of our algorithms run in $o(n^2)$ time, whereas the obvious algorithms for these problems run in $\Theta(n^2)$ time.

1 Introduction

How should we rotate two necklaces, each with n beads at different locations, to best align the beads? More precisely, each necklace is represented by a set of n points on the unit-circumference circle, and the goal is to find rotations of the

⋆ Supported by NSERC.
⋆⋆ Supported in part by NSF grants CCF-0430849 and OISE-0334653 and by an Alfred P. Sloan Fellowship.

necklaces, and a perfect matching between the beads of the two necklaces, that minimizes some norm of the circular distances between matched beads. In particular, the ℓ_1 norm minimizes the average absolute circular distance between matched beads, the ℓ_2 norm minimizes the average squared circular distance between matched beads, and the ℓ_∞ norm minimizes the maximum circular distance between matched beads. The ℓ_1 version of this necklace alignment problem was introduced by Toussaint [29] in the context of comparing rhythms in computational music theory, with possible applications to rhythm phylogeny [14,30].

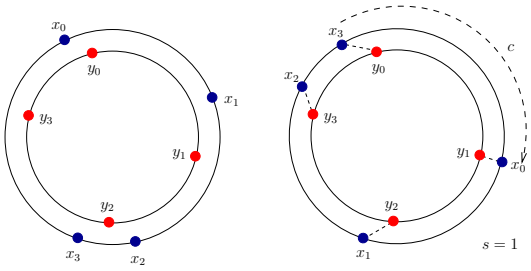

Fig. 1. An example of necklace alignment: the input (left) and one possible output (right).

Toussaint [29] gave a simple $O(n^2)$-time algorithm for ℓ_1 necklace alignment, and asked whether the problem could be solved in $o(n^2)$ time. In this paper, we solve this open problem by giving $o(n^2)$-time algorithms for ℓ_1, ℓ_2, and ℓ_∞ necklace alignment, in both the standard real RAM model of computation and the less realistic nonuniform linear decision tree model of computation.

Our approach is based on reducing the necklace alignment problem to another important problem, convolution, for which we also obtain improved algorithms. The $(+,\cdot)$ convolution of two vectors $\mathbf{x} = \langle x_0, x_1, \ldots, x_{n-1}\rangle$ and $\mathbf{y} = \langle y_0, y_1, \ldots, y_{n-1}\rangle$, is the vector $\mathbf{x} * \mathbf{y} = \langle z_0, z_1, \ldots, z_{n-1}\rangle$ where $z_k = \sum_{i=0}^{k} x_i \cdot y_{k-i}$. While any $(\oplus,\odot)$ convolution with specified addition and multiplication operators (here denoted $\mathbf{x} \overset{\odot}{*}_{\oplus} \mathbf{y}$) can be computed in $O(n^2)$ time, the $(+,\cdot)$ convolution can be computed in $O(n \lg n)$ time using the Fast Fourier Transform [10,21,22], because the Fourier transform converts convolution into elementwise multiplication. Indeed, fast $(+,\cdot)$ convolution was one of the early breakthroughs in algorithms, with applications to polynomial and integer multiplication [3], batch polynomial evaluation [11, Problem 30-5], 3SUM [15,1], string matching [17,23,9], matrix multiplication [7], and even juggling [5].

As we show in Theorems 1, 3, and 11, respectively, ℓ_2 necklace alignment reduces to standard $(+,\cdot)$ convolution, ℓ_∞ necklace alignment reduces to $(\min,+)$ [and $(\max,+)$] convolution, and ℓ_1 necklace alignment reduces to $(\text{median},+)$ convolution (whose kth entry is $\text{median}_{i=0}^{k}(x_i + y_{k-i})$). The $(\min,+)$ convolution problem has appeared frequently in the literature, already appearing in Bellman's early work on dynamic programming in the early 1960s [2,16,24,25,26,28]. Its name varies among "minimum convolution", "min-sum convolution", "inf-convolution", "infimal convolution", and "epigraphical sum".[9] To date, however,

[9] "Tropical convolution" would also make sense, by direct analogy with tropical geometry, but we have never seen this terminology used in print.

no worst-case $o(n^2)$-time algorithms for this convolution, or the more complex (median, $+$) convolution, has been obtained. In this paper, we develop worst-case $o(n^2)$-time algorithms for (min, $+$) and (median, $+$) convolution, in the real RAM and the nonuniform linear decision tree.

Necklace alignment problem. More formally, in the *necklace alignment problem*, the input is a number p representing the ℓ_p norm, and two sorted vectors of n real numbers, $\mathbf{x} = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ and $\mathbf{y} = \langle y_0, y_1, \ldots, y_{n-1} \rangle$, representing the two necklaces. See Figure 1. Canonically, we assume that each number x_i and y_i is in the range $[0, 1)$, representing a point on the unit-circumference circle (parameterized clockwise from some fixed point).

The optimization problem involves two parameters. The first parameter, the *offset* $c \in [0, 1)$, is the clockwise rotation angle of the first necklace relative to the second necklace. The second parameter, the *shift* $s \in \{0, 1, \ldots, n\}$, defines the perfect matching between beads: bead i of the first necklace matches with bead $(i + s) \bmod n$ of the second necklace. (Here we use the property that an optimal perfect matching between the beads does not cross itself.)

The goal of the ℓ_p necklace alignment problem is to find the offset $c \in [0, 1)$ and the shift $s \in \{0, 1, \ldots, n\}$ that minimize $\sum_{i=0}^{n-1} \left|x_i - y_{(i+s) \bmod n} + c\right|^p$ or, in the case $p = \infty$, that minimize $\max_{i=0}^{n-1} \left|x_i - y_{(i+s) \bmod n} + c\right|$.

Although not obvious from the definition, the ℓ_1, ℓ_2, and ℓ_∞ necklace alignment problems all have trivial $O(n^2)$ solutions. In each case, as we show, the optimal offset c can be computed in linear time for a given shift value s (sometimes even independent of s). The optimization problem is thus effectively over just $s \in \{0, 1, \ldots, n\}$, and the objective costs $O(n)$ time to compute for each s, giving an $O(n^2)$-time algorithm.

Related work. Although necklaces are studied throughout mathematics, mainly in combinatorial settings, we are not aware of any work on the necklace alignment problem before Toussaint [29]. He introduced ℓ_1 necklace alignment, calling it the *cyclic swap-distance* or *necklace swap-distance* problem, with a restriction that the beads lie at integer coordinates. Colannino et al. [8] consider some different distance measures between two sets of points on the real line in which the matching does not have to match every point. They do not, however, consider alignment under such distance measures.

The only subquadratic results for (min, $+$) convolution concern two special cases. First, the (min, $+$) convolution of two convex sequences or functions can be trivially computed in $O(n)$ time by a simple merge, which is the same as computing the Minkowski sum of two convex polygons [26]. This special case is already used in image processing and computer vision [16,24]. Second, Bussieck et al. [4] proved that the (min, $+$) convolution of two *randomly permuted* sequences can be computed in $O(n \lg n)$ expected time. Our results are the first to improve the worst-case running time for (min, $+$) convolution.

Connections to $X + Y$. The necklace alignment problems, and their corresponding convolution problems, are also intrinsically connected to problems on

$X + Y$ matrices. Given two lists of n numbers, $X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ and $Y = \langle y_0, y_1, \ldots, y_{n-1} \rangle$, $X + Y$ is the matrix of all pairwise sums, whose (i, j)th entry is $x_i + y_j$. A classic unsolved problem [12] is whether the entries of $X+Y$ can be sorted in $o(n^2 \lg n)$ time. Fredman [19] showed that $O(n^2)$ comparisons suffice in the nonuniform linear decision tree model, but it remains open whether this can be converted into an $O(n^2)$-time algorithm in the real RAM model. Steiger and Streinu [27] gave a simple algorithm that takes $O(n^2 \log n)$ time while using only $O(n^2)$ comparisons.

The $(\min, +)$ convolution is equivalent to finding the minimum element in each antidiagonal of the $X + Y$ matrix, and similarly the $(\max, +)$ convolution finds the maximum element in each antidiagonal. We show that ℓ_∞ necklace alignment is equivalent to finding the antidiagonal of $X + Y$ with the smallest *range* (the maximum element minus the minimum element). The $(\text{median}, +)$ convolution is equivalent to finding the median element in each antidiagonal of the $X + Y$ matrix. We show that ℓ_1 necklace alignment is equivalent to finding the antidiagonal of $X + Y$ with the smallest *median cost* (the total distance between each element and the median of the elements). Given the apparent difficulty in sorting $X+Y$, it seems natural to believe that the minimum, maximum, and median elements of every antidiagonal cannot be found, and that the corresponding objectives cannot be minimized, any faster than $O(n^2)$ total time. Figure 2 shows a sample $X + Y$ matrix with the maximum element in each antidiagonal marked, with no apparent structure. Nonetheless, we show that $o(n^2)$ algorithms are possible.

Our results. In the standard real RAM model, we give subquadratic algorithms for the ℓ_1, ℓ_2, and ℓ_∞ necklace alignment problems, and for the $(\min, +)$ and $(\text{median}, +)$ convolution problems, using techniques of Chan [6]. Despite the roughly logarithmic factor improvements for ℓ_1 and ℓ_∞, these results do not use word-level bit tricks of word-RAM fame.

1. $O(n \lg n)$-time algorithm on the real RAM for ℓ_2 necklace alignment (Section 2).
2. $O(n^2/\lg n)$-time algorithm on the real RAM for ℓ_∞ necklace alignment and $(\min, +)$ convolution (Section 3).
3. $O(n^2(\lg \lg n)^2/\lg n)$-time algorithm on the real RAM for ℓ_1 necklace alignment and $(\text{median}, +)$ convolution (Section 4).

In the nonuniform linear decision tree model, we give faster algorithms for the ℓ_1 and ℓ_∞ necklace alignment problems, using techniques of Fredman [19,20]:

4. $O(n\sqrt{n})$-time algorithm in the nonuniform linear decision tree model for ℓ_∞ necklace alignment and $(\min, +)$ convolution (Section 3).
5. $O(n\sqrt{n \lg n})$-time algorithm in the nonuniform linear decision tree model for ℓ_1 necklace alignment and $(\text{median}, +)$ convolution (Section 4).

(Although we state our results here in terms of $(\min, +)$ and $(\text{median}, +)$ convolution, our results discuss $-$ instead of $+$ for synergy with necklace alignment.)

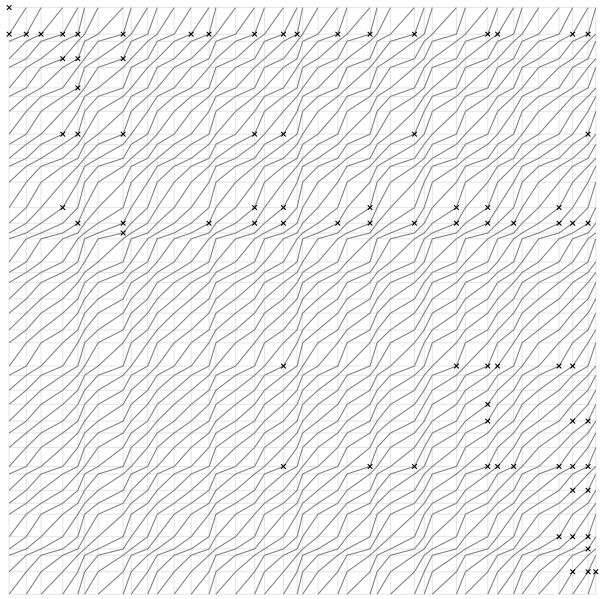

Fig. 2. An $X + Y$ matrix. Each polygonal line denotes an antidiagonal of the matrix, with a point at coordinates (x, y) denoting the value $x + y$ for $x \in X$ and $y \in Y$. An $\times$ denotes the maximum element in each antidiagonal.

2 ℓ_2 Necklace Alignment and $(+, \cdot)$ Convolution

In this section, we show how ℓ_2 necklace alignment reduces to standard convolution, leading to an $O(n \lg n)$-time algorithm.

Theorem 1. *The ℓ_2 necklace alignment problem can be solved in $O(n \lg n)$ time on a real RAM.*

Proof. The objective $\sum_{i=0}^{n-1} \left(x_i - y_{(i+s) \bmod n} + c\right)^2$ expands algebraically to

$$\begin{aligned}
&\sum_{i=0}^{n-1} \left(x_i^2 + 2cx_i + c^2\right) + \sum_{i=0}^{n-1} \left(y_{(i+s) \bmod n}^2 - 2cy_{(i+s) \bmod n}\right) - 2\sum_{i=0}^{n-1} x_i y_{(i+s) \bmod n} \\
&= \left[\sum_{i=0}^{n-1} \left(x_i^2 + y_i^2\right) + 2c\sum_{i=0}^{n-1} \left(x_i - y_i\right) + nc^2\right] - 2\sum_{i=0}^{n-1} x_i y_{(i+s) \bmod n}.
\end{aligned}$$

The first term depends solely on the inputs and the variable c, while the second term depends solely on the inputs and the variable s. Thus the two terms can be optimized separately. The first term can be optimized in $O(n)$ time by solving for when the derivative, which is linear in c, is zero. The second term can be

computed, for each $s \in \{0, 1, \ldots, n-1\}$, in $O(n \lg n)$ time using $(+, \cdot)$ convolution (and therefore optimized in the same time). Specifically, define the vectors

$$\mathbf{x}' = \langle x_0, x_1, \ldots, x_{n-1}; \underbrace{0, 0, \ldots, 0}_{n} \rangle; \quad \mathbf{y}' = \langle y_{n-1}, y_{n-2}, \ldots, y_0; y_{n-1}, y_{n-2}, \ldots, y_0 \rangle.$$

Then, for $s' \in \{0, 1, \ldots, n-1\}$, the $(n+s')$th entry of the convolution $\mathbf{x}' * \mathbf{y}'$ is $\sum_{i=0}^{n+s'} x'_i y'_{n+s'-i} = \sum_{i=0}^{n-1} x_i y_{(i-s'-1) \bmod n}$, which is the desired entry if we let $s' = n-1-s$. We can compute the entire convolution in $O(n \lg n)$ time using the Fast Fourier Transform. □

3 ℓ_∞ Necklace Alignment and $(\min, +)$ Convolution

First we show the relation between ℓ_∞ necklace alignment and $(\min, +)$ convolution. We need the following basic fact:

Fact 2. *For any vector* $\mathbf{z} = \langle z_0, z_1, \ldots, z_{n-1} \rangle$*, the minimum value of* $\max_{i=0}^{n-1} |z_i + c|$ *is* $\frac{1}{2}\left(\max_{i=0}^{n-1} z_i - \min_{i=0}^{n-1} z_i\right)$*, which is achieved when* $c = -\frac{1}{2}\left(\min_{i=0}^{n-1} z_i + \max_{i=0}^{n-1} z_i\right)$.

Instead of using $(\min, +)$ convolution directly, we use two equivalent forms, $(\min, -)$ and $(\max, -)$ convolution:

Theorem 3. *The* ℓ_∞ *necklace alignment problem can be reduced in* $O(n)$ *time to one* $(\min, -)$ *convolution and one* $(\max, -)$ *convolution.*

Proof. For two necklaces $\mathbf{x}$ and $\mathbf{y}$, we apply the $(\min, -)$ convolution to the following vectors:

$$\mathbf{x}' = \langle x_0, x_1, \ldots, x_{n-1}; \underbrace{\infty, \ldots, \infty}_{n} \rangle; \quad \mathbf{y}' = \langle y_{n-1}, y_{n-2}, \ldots, y_0; y_{n-1}, y_{n-2}, \ldots, y_0 \rangle.$$

Then, for $s' \in \{0, 1, \ldots, n-1\}$, the $(n+s')$th entry of $\mathbf{x}' \underset{\min}{\bar{*}} \mathbf{y}'$ is $\min_{i=0}^{n+s'}(x'_i - y'_{n+s'-i}) = \min_{i=0}^{n-1}(x_i - y_{(i-s'-1) \bmod n})$, which is $\min_{i=0}^{n-1}(x_i - y_{(i+s) \bmod n})$ if we let $s' = n-1-s$. By symmetry, we can compute the $(\max, -)$ convolution $\mathbf{x}'' \underset{\max}{\bar{*}} \mathbf{y}'$, where $\mathbf{x}''$ has $-\infty$'s in place of ∞'s, and use it to compute $\max_{i=0}^{n-1}(x_i - y_{(i+s) \bmod n})$ for each $s \in \{0, 1, \ldots, n-1\}$. Applying Fact 2, we can therefore minimize $\max_{i=0}^{n-1} |x_i - y_{(i+s) \bmod n} + c|$ over c, for each $s \in \{0, 1, \ldots, n-1\}$. By brute force, we can minimize over s as well using $O(n)$ additional comparisons and time. □

For our nonuniform linear decision tree results, we use the main theorem of Fredman's work on sorting $X + Y$:

Theorem 4. [19] *For any fixed set* Γ *of permutations of* N *elements, there is a comparison tree of depth* $O(N + \lg |\Gamma|)$ *that sorts any sequence whose rank permutation belongs to* Γ.

Theorem 5. *The* $(\min, -)$ *convolution of two vectors of length* n *can be computed in* $O(n\sqrt{n})$ *time in the nonuniform linear decision tree model.*

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ denote the two vectors of length n, and let $\mathbf{x} \underset{\min}{\bar{*}} \mathbf{y}$ denote their $(\min, -)$ convolution, whose kth entry is $\min_{i=0}^{k} (x_i - y_{k-i})$.

First we sort the set $D = \{x_i - x_j, y_i - y_j : |i-j| \le d\}$ of pairwise differences between nearby x_i's and nearby y_i's, where $d \le n$ is a value to be determined later. This set D has $N = O(nd)$ elements. The possible sorted orders of D correspond to cells in the arrangement of hyperplanes in $\mathbb{R}^{2n}$ induced by all $\binom{N}{2}$ possible comparisons between elements in the set, and this hyperplane arrangement has $O(N^{4n})$ cells. By Theorem 4, there is a comparison tree sorting D of depth $O(N + n \lg N) = O(nd + n \lg n)$.

The comparisons we make to sort D enable us to compare $x_i - y_{k-i}$ versus $x_j - y_{k-j}$ for free, provided $|i-j| \le d$, because $x_i - y_{k-i} < x_j - y_{k-j}$ precisely if $x_i - x_j < y_{k-i} - y_{k-j}$. Thus, in particular, we can compute $M_k(\lambda) = \min\{x_i - y_{k-i} : i = \lambda, \lambda+1, \ldots, \min\{\lambda + d, n\} - 1\}$ for free (using the outcomes of the comparisons we have already made).

We can rewrite the kth entry $\min_{i=0}^{k}(x_i - y_{k-i})$ of $\mathbf{x} \underset{\min}{\bar{*}} \mathbf{y}$ as $\min\{M_k(0), M_k(d), M_k(2d), \ldots, M_k(\lceil k/d \rceil d)\}$, and thus we can compute it in $O(k/d) = O(n/d)$ comparisons between differences. Therefore all n entries can be computed in $O(nd + n \lg n + n^2/d)$ total time.

This asymptotic running time is minimized when $nd = \Theta(n^2/d)$, i.e., when $d^2 = \Theta(n)$. Substituting $d = \sqrt{n}$, we obtain a running time of $O(n\sqrt{n})$ in the nonuniform linear decision tree model. □

Combining Theorems 3 and 5, we obtain the following result:

Corollary 6. *The* ℓ_∞ *necklace alignment problem can be solved in* $O(n\sqrt{n})$ *time in the nonuniform linear decision tree model.*

Our results on the real RAM use the following geometric lemma from Chan's work on all-pairs shortest paths:

Lemma 7. [6, Lemma 2.1] *Given* n *points* $p_1, p_2, \ldots, p_n$ *in* d *dimensions, each colored either red or blue, we can find the* P *pairs* (p_i, p_j) *for which* p_i *is red,* p_j *is blue, and* p_i *dominates* p_j *(i.e., for all* k*, the* k*th coordinate of* p_i *is at least the* k*th coordinate of* p_j*), in* $2^{O(d)} n^{1+\varepsilon} + O(P)$ *time for arbitrarily small* $\varepsilon > 0$.

Theorem 8. *The* $(\min, -)$ *convolution of two vectors of length* n *can be computed in* $O(n^2/\lg n)$ *time on a real RAM.*

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ denote the two vectors of length n, and let $\mathbf{x} \underset{\max}{\bar{*}} \mathbf{y}$ denote their $(\max, -)$ convolution. (Symmetrically, we can compute the $(\min, -)$ convolution.) For each $\delta \in \{0, 1, \ldots, d-1\}$, for each $i \in \{0, d, 2d, \ldots, \lfloor n/d \rfloor d\}$, and for each $j \in \{0, 1, \ldots, n-1\}$, we define the d-dimensional points

$$p_{\delta,i} = (x_{i+\delta} - x_i, x_{i+\delta} - x_{i+1}, \ldots, x_{i+\delta} - x_{i+d-1}),$$
$$q_{\delta,j} = (y_{j-\delta} - y_i, y_{j-\delta} - y_{i-1}, \ldots, y_{j-\delta} - y_{j-d-1}).$$

(To handle boundary cases, define $x_i = \infty$ and $y_j = -\infty$ for indices i, j outside $[0, n-1]$.) For each $\delta \in \{0, 1, \ldots, d-1\}$, we apply Lemma 7 to the set of red points $\{p_{\delta,i} : i = 0, d, 2d, \ldots, \lfloor n/d \rfloor d\}$ and the set of blue points $\{q_{\delta,j} : j = 0, 1, \ldots, n-1\}$, to obtain all dominating pairs $(p_{\delta,i}, q_{\delta,j})$.

Point $p_{\delta,i}$ dominates $q_{\delta,j}$ precisely if $x_{i+\delta} - x_{i+\delta'} \geq y_{j-\delta} - y_{j-\delta'}$ for all $\delta' \in \{0, 1, \ldots, d-1\}$ (ignoring the indices outside $[0, n-1]$). By re-arranging terms, this condition is equivalent to $x_{i+\delta} - y_{j-\delta} \geq x_{i+\delta'} - y_{j-\delta'}$ for all $\delta' \in \{0, 1, \ldots, d-1\}$. If we substitute $j = k - i$, we obtain that $(p_{\delta,i}, q_{\delta,k-i})$ is a dominating pair precisely if $x_{i+\delta} - y_{k-i-\delta} = \max_{\delta'=1}^{d-1}(x_{i+\delta'} - y_{k-i-\delta'})$. Thus, the set of dominating pairs gives us the maximum $M_k(i) = \max\{x_i - y_{k-i}, x_{i+1} - y_{k-i+1}, \ldots, x_{\min\{i+d,n\}-1} - y_{\min\{k-i+d,n\}-1}\}$ for each i divisible by d and for each k. Also, there can be at most $O(n^2/d)$ such pairs for all i, j, δ, because there are $O(n/d)$ choices for i and $O(n)$ choices for j, and if $(p_{\delta,i}, q_{\delta,j})$ is a dominating pair, then $(p_{\delta',i}, q_{\delta',j})$ cannot be a dominating pair for any $\delta' \neq \delta$. (Here we assume that the max is achieved uniquely, which can be arranged by standard perturbation techniques or by breaking ties consistently [6].) Hence, the running time of the d executions of Lemma 7 is $d2^{O(d)}n^{1+\varepsilon} + O(n^2/d)$ time, which is $O(n^2/\lg n)$ if we choose $d = \alpha \lg n$ for a sufficiently small constant $\alpha > 0$. We can rewrite the kth entry $\max_{i=0}^{k}(x_i - y_{k-i})$ of $\mathbf{x} \underset{\max}{\bar{*}} \mathbf{y}$ as $\max\{M_k(0), M_k(d), M_k(2d), \ldots, M_k(\lceil k/d \rceil d)\}$, and thus we can compute it in $O(k/d) = O(n/d)$ time. Thus all n entries can be computed in $O(n^2/d) = O(n^2/\lg n)$ time on a real RAM. □

Combining Theorems 3 and 8, we obtain the following result:

Corollary 9. *The ℓ_∞ necklace alignment problem can be solved in $O(n^2/\lg n)$ time on a real RAM.*

This approach likely cannot be improved beyond $O(n^2/\lg n)$. Such an improvement would require an improvement to Lemma 7, which would in turn improve the fastest known algorithm for all-pairs shortest paths in dense graphs, the $O(n^3/\lg n)$-time algorithm of [6].

4 ℓ_1 Necklace Alignment and (median, +) Convolution

First we show the relation between ℓ_1 necklace alignment and (median, +) convolution. We need the following basic fact:

Fact 10. *For any vector $\mathbf{z} = \langle z_0, z_1, \ldots, z_{n-1} \rangle$, $\sum_{i=0}^{n-1} |z_i + c|$ is minimized when $c = -\operatorname{median}_{i=0}^{n-1} z_i$.*

Instead of using (median, +) convolution directly, we use the equivalent form, (median, −) convolution:

Theorem 11. *The ℓ_1 necklace alignment problem can be reduced in $O(n)$ time to one* (median, −) *convolution.*

Proof. For two necklaces $\mathbf{x}$ and $\mathbf{y}$, we apply the (median, $-$) convolution to the following vectors, as in the proof of Theorem 3:

$$\mathbf{x}' = \langle x_0, x_0, x_1, x_1, \ldots, x_{n-1}, x_{n-1}; \underbrace{\infty, -\infty, \infty, -\infty, \ldots, \infty, -\infty}_{2n} \rangle,$$

$$\mathbf{y}' = \langle y_{n-1}, y_{n-1}, y_{n-2}, y_{n-2}, \ldots, y_0, y_0; y_{n-1}, y_{n-1}, y_{n-2}, y_{n-2}, \ldots, y_0, y_0 \rangle.$$

Then, for $s' \in \{0, 1, \ldots, n-1\}$, the $2(n+s')+1$st entry of $\mathbf{x}' \underset{\text{med}}{\bar{*}} \mathbf{y}'$ is $\text{median}_{i=0}^{2(n+s')+1}(x'_i - y'_{2(n+s')+1-i}) = \text{median}_{i=0}^{n-1}(x_i - y_{(i-s'-1) \bmod n})$, which is $\text{median}_{i=0}^{n-1}(x_i - y_{(i+s) \bmod n})$ if we let $s' = n - 1 - s$. Applying Fact 10, we can therefore minimize $\text{median}_{i=0}^{n-1} |x_i - y_{(i+s) \bmod n} + c|$ over c, for each $s \in \{0, 1, \ldots, n-1\}$. By brute force, we can minimize over s as well using $O(n)$ additional comparisons and time. □

Our results for (median, $-$) convolution use the following result of Frederickson and Johnson:

Theorem 12. [18] *The median element of the union of k sorted lists, each of length n, can be computed in $O(k \lg n)$ time and comparisons.*

We begin with our results for the nonuniform linear decision tree model:

Theorem 13. *The* (median, $-$) *convolution of two vectors of length n can be computed in $O(n\sqrt{n \lg n})$ time in the nonuniform linear decision tree model.*

Proof. As in the proof of Theorem 6, we sort the set $D = \{x_i - x_j, y_i - y_j : |i - j| \le d\}$ of pairwise differences between nearby x_i's and nearby y_i's, where $d \le n$ is a value to be determined later. By Theorem 4, this step requires $O(nd + n \lg n)$ comparisons between differences. These comparisons enable us to compare $x_i - y_{k-i}$ versus $x_j - y_{k-j}$ for free, provided $|i - j| \le d$, because $x_i - y_{k-i} < x_j - y_{k-j}$ precisely if $x_i - x_j < y_{k-i} - y_{k-j}$. In particular, we can sort each list $L_k(\lambda) = \langle x_i - y_{k-i} : i = \lambda, \lambda + 1, \ldots, \min\{\lambda + d, n\} - 1 \rangle$ for free. By Theorem 12, we can compute the median of $L_k(0) \cup L_k(d) \cup L_k(2d) \cup \cdots \cup L_k(\lceil k/d \rceil d)$, i.e., $\text{median}_{i=0}^{k}(x_i - y_{k-i})$, in $O((k/d) \lg d) = O((n/d) \lg d)$ comparisons. Also, in the same asymptotic number of comparisons, we can binary search to find where the median fits in each of the $L_k(\lambda)$ lists, and therefore which differences are smaller and which differences are larger than the median. This median is the kth entry of $\mathbf{x} \underset{\text{med}}{\bar{*}} \mathbf{y}$. Therefore, we can compute all n entries of $\mathbf{x} \underset{\text{med}}{\bar{*}} \mathbf{y}$ in $O(nd + n \lg n + (n^2/d) \lg d)$ comparisons. This asymptotic running time is minimized when $nd = \Theta((n^2/d) \lg d)$, i.e., when $d^2 / \lg d = \Theta(n)$. Substituting $d = \sqrt{n \lg n}$, we obtain a running time of $O(n\sqrt{n \lg n})$ in the nonuniform linear decision tree model. □

Combining Theorems 11 and 13, we obtain the following result:

Corollary 14. *The ℓ_1 necklace alignment problem can be solved in $O(n\sqrt{n \lg n})$ time in the nonuniform linear decision tree model.*

Now we turn to the analogous results for the real RAM:

Theorem 15. *The* (median, $-$) *convolution of two vectors of length n can be computed in $O(n^2(\lg\lg n)^2/\lg n)$ time on a real RAM.*

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ denote the two vectors of length n, and let $\mathbf{x} \underset{\text{med}}{\bar{*}} \mathbf{y}$ denote their (median, $-$) convolution. For each permutation π on the set $\{0, 1, \ldots, d-1\}$, for each $i \in \{0, d, 2d, \ldots, \lfloor n/d \rfloor d\}$, and for each $j \in \{0, 1, \ldots, n-1\}$, we define the $(d-1)$-dimensional points

$$p_{\pi,i} = (x_{i+\pi(0)} - x_{i+\pi(1)}, x_{i+\pi(1)} - x_{i+\pi(2)}, \ldots, x_{i+\pi(d-2)} - x_{i+\pi(d-1)}),$$
$$q_{\pi,j} = (y_{j-\pi(0)} - y_{j-\pi(1)}, y_{j-\pi(1)} - y_{j-\pi(2)}, \ldots, y_{j-\pi(d-2)} - y_{j-\pi(d-1)}),$$

(To handle boundary cases, define $x_i = \infty$ and $y_j = -\infty$ for indices i, j outside $[0, n-1]$.) For each permutation π, we apply Lemma 7 to the set of red points $\{p_{\pi,i} : i = 0, d, 2d, \ldots, \lfloor n/d \rfloor d\}$ and the set of blue points $\{q_{\pi,j} : j = 0, 1, \ldots, n-1\}$, to obtain all dominating pairs $(p_{\pi,i}, q_{\pi,j})$.

Point $p_{\pi,i}$ dominates $q_{\pi,j}$ precisely if $x_{i+\pi(\delta)} - x_{i+\pi(\delta+1)} \geq y_{j-\pi(\delta)} - y_{j-\pi(\delta+1)}$ for all $\delta \in \{0, 1, \ldots, d-2\}$ (ignoring the indices outside $[0, n-1]$). By rearranging terms, this condition is equivalent to $x_{i+\pi(\delta)} - y_{j-\pi(\delta)} \geq x_{i+\pi(\delta+1)} - y_{j-\pi(\delta+1)}$ for all $\delta \in \{0, 1, \ldots, d-2\}$, i.e., π is a sorting permutation of $\langle x_i - y_j, x_{i+1} - y_{j-1}, \ldots, x_{i+d-1} - y_{j-d+1} \rangle$. If we substitute $j = k - i$, we obtain that $(p_{\pi,i}, q_{\pi,k-i})$ is a dominating pair precisely if π is a sorting permutation of the list $L_k(i) = \langle x_i - y_{k-i}, x_{i+1} - y_{k-i+1}, \ldots, x_{\min\{i+d,n\}-1} - y_{\min\{k-i+d,n\}-1} \rangle$. Thus, the set of dominating pairs gives us the sorted order of $L_k(i)$ for each i divisible by d and for each k. Also, there can be at most $O(n^2/d)$ total dominating pairs $(p_{\pi,i}, q_{\pi,j})$ over all i, j, π, because there are $O(n/d)$ choices for i and $O(n)$ choices for j, and if $(p_{\pi,i}, q_{\pi,j})$ is a dominating pair, then $(p_{\pi',i}, q_{\pi',j})$ cannot be a dominating pair for any permutation $\pi' \neq \pi$. (Here we assume that the sorted order is unique, which can be arranged by standard perturbation techniques or by breaking ties consistently [6].) Hence, the running time of the $d!$ executions of Lemma 7 is $d!\, 2^{O(d)} n^{1+\varepsilon} + O(n^2/d)$ time, which is $O(n^2 \lg\lg n / \lg n)$ if we choose $d = \alpha \lg n / \lg\lg n$ for a sufficiently small constant $\alpha > 0$. By Theorem 12, we can compute the median of $L_k(0) \cup L_k(d) \cup L_k(2d) \cup \cdots \cup L_k(\lceil k/d \rceil d)$, i.e., $\text{median}_{i=0}^{k}(x_i - y_{k-i})$, in $O((k/d)\lg d) = O((n/d)\lg d)$ comparisons. Also, in the same asymptotic number of comparisons, we can binary search to find where the median fits in each of the $L_k(\lambda)$ lists, and therefore which differences are smaller and which differences are larger than the median. This median is the kth entry of $\mathbf{x} \underset{\text{med}}{\bar{*}} \mathbf{y}$. Therefore all n entries can be computed in $O(n^2(\lg d)/d) = O(n^2(\lg\lg n)^2/\lg n)$ time on a real RAM. □

Combining Theorems 11 and 15, we obtain the following result:

Corollary 16. *The ℓ_1 necklace alignment problem can be solved in $O(n^2(\lg\lg n)^2/\lg n)$ time on a real RAM.*

As before, this approach likely cannot be improved beyond $O(n^2/\lg n)$, because such an improvement would require an improvement to Lemma 7, which would in turn improve the fastest known algorithm for all-pairs shortest paths in dense graphs [6]. In contrast to (median, +) convolution, (mean, +) convolution is trivial to compute in linear time.

5 Conclusion

The convolution problems we consider here have connections to many classic problems, and it would be interesting to explore whether the structural information extracted by our algorithms could be used to devise faster algorithms for these classic problems. For example, does the antidiagonal information of the $X + Y$ matrix lead to a $o(n^2 \lg n)$-time algorithm for sorting $X + Y$? We believe that any further improvements to our convolution algorithms would require progress and/or have interesting implications on all-pairs shortest paths [6].

Our $(\min, -)$-convolution algorithms give subquadratic algorithms for *polyhedral 3SUM*: given three lists, $A = \langle a_0, a_1, \ldots, a_{n-1}\rangle$, $B = \langle b_0, b_1, \ldots, b_{n-1}\rangle$, and $C = \langle c_0, c_1, \ldots, c_{2n-2}\rangle$, such that $a_i + b_j \le c_{i+j}$ for all $0 \le i, j < n$, decide whether $a_i + b_j = c_{i+j}$ for any $0 \le i, j < n$. This problem is a special case of 3SUM, and this special case has an $\Omega(n^2)$ lower bound in the 3-linear decision tree model [15]. Our results solve polyhedral 3SUM in $O(n^2/\lg n)$ time in the 4-linear decision tree model, and in $O(n\sqrt{n})$ time in the nonuniform 4-linear decision tree model, solving an open problem of Erickson [13]. Can these algorithms be extended to solve 3SUM in subquadratic time in the (nonuniform) decision tree model?

Acknowledgments. This work was initiated at the 20th Bellairs Winter Workshop on Computational Geometry held January 28–February 4, 2005. We thank the other participants of that workshop—Greg Aloupis, Justin Colannino, Mirela Damian-Iordache, Vida Dujmović, Francisco Gomez-Martin, Danny Krizanc, Erin McLeish, Henk Meijer, Patrick Morin, Mark Overmars, Suneeta Ramaswami, David Rappaport, Diane Souvaine, Ileana Streinu, David Wood, Godfried Toussaint, Remco Veltkamp, and Sue Whitesides—for helpful discussions and contributing to a fun and creative atmosphere. We particularly thank the organizer, Godfried Toussaint, for posing the problem to us.

References

1. I. Baran, E. D. Demaine, and M. Pătraşcu. Subquadratic algorithms for 3SUM. In *Proc. 9th Worksh. Algorithms & Data Structures*, LNCS 3608, pp. 409–421, 2005.
2. R. Bellman and W. Karush. Mathematical programming and the maximum transform. *J. SIAM*, 10(3):550–567, 1962.
3. D. J. Bernstein. Fast multiplication and its applications. In J. Buhler and P. Stevenhagen, eds., *Algorithmic Number Theory*. Cambridge University Press. To appear.
4. M. Bussieck, H. Hassler, G. J. Woeginger, and U. T. Zimmermann. Fast algorithms for the maximum convolution problem. *Oper. Res. Lett.*, 15(3):133–141, 1994.
5. J. Cardinal, S. Kremer, and S. Langerman. Juggling with pattern matching. *Theory Comput. Syst.*, 39(3):425–437, 2006.
6. T. M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. In *Proc. 9th Worksh. Algorithms & Data Structures*, LNCS 3608, pp. 318–324, 2005.

7. H. Cohn, R. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic algorithms for matrix multiplication. In *Proc. 46th IEEE Symp. Found. Computer Science*, pp. 379–388, 2005.
8. J. Colannino, M. Damian, F. Hurtado, J. Iacono, H. Meijer, S. Ramaswami, and G. Toussaint. An $O(n \log n)$-time algorithm for the restriction scaffold assignment. *J. Comput. Biol.*, 13(4):979–989, 2006.
9. R. Cole and R. Hariharan. Verifying candidate matches in sparse and wildcard matching. In *Proc. 34th ACM Symp. Theory of Computing*, pp. 592–601, 2002.
10. J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
11. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.
12. E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke. Problem 41: Sorting $X + Y$ (pairwise sums). In *The Open Problems Project*. http://cs.smith.edu/~orourke/TOPP/P41.html.
13. E. D. Demaine and J. O'Rourke. Open problems from CCCG 2005. In *Proc. 18th Canadian Conference on Computational Geometry*, 2006.
14. J. M. Díaz-Báñez, G. Farigu, F. Gómez, D. Rappaport, and G. T. Toussaint. El compás flamenco: A phylogenetic analysis. In *Proc. BRIDGES: Mathematical Connections in Art, Music, and Science*, pp. 61–70, 2004.
15. J. Erickson. Lower bounds for linear satisfiability problems. *Chic. J. Theoret. Comput. Sci.*, 1999.
16. P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. TR2004-1963, Faculty of Computing and Information Science, Cornell Univ.
17. M. J. Fischer and M. S. Paterson. String-matching and other products. In *Complexity of computation*, Proc. SIAM-AMS Applied Math. Symp., 1973, pp. 113–125.
18. G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. System Sci.*, 24(2):197–208, 1982.
19. M. L. Fredman. How good is the information theory bound in sorting? *Theoret. Comput. Sci.*, 1(4):355–361, 1976.
20. M. L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976.
21. C. F. Gauss. *Werke*, vol. 3. Königlichen Gesellschaft der Wissenschaften, 1866.
22. M. T. Heideman, D. H. Johnson, and C. S. Burrus. Gauss and the history of the fast Fourier transform. *Arch. Hist. Exact Sci.*, 34(3):265–277, 1985.
23. P. Indyk. Faster algorithms for string matching problems: Matching the convolution bound. In *Proc. 39th Symp. Found. Computer Science*, pp. 166–173, 1998.
24. P. Maragos. Differential morphology. In S. Mitra and G. Sicuranza, eds., *Nonlinear Image Processing*, ch. 10, pp. 289–329. Academic Press, 2000.
25. J.-J. Moreau. Inf-convolution, sous-additivité, convexité des fonctions numériques. *J. Math. Pures Appl. (9)*, 49:109–154, 1970.
26. R. T. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1970.
27. W. L. Steiger and I. Streinu. A pseudo-algorithmic separation of lines from pseudolines. *Infor. Process. Lett.*, 53(5):295–299, 1995.
28. T. Strömberg. The operation of infimal convolution. *Dissertationes Math. (Rozprawy Mat.)*, 352:58, 1996.
29. G. Toussaint. The geometry of musical rhythm. In *Proc. Japan Conference on Discrete and Computational Geometry*, LNCS 3742, pp. 198–212, 2004.
30. G. T. Toussaint. A comparison of rhythmic similarity measures. In *Proc. 5th International Conference on Music Information Retrieval*, pp. 242–245, 2004.