Hamming distance completeness

Originally published in:
Leibniz International Proceedings in Informatics (LIPIcs) 128, https://doi.org/10.4230/LIPIcs.CPM.2019.14

Hamming Distance Completeness

Karim Labib
Google Zürich, Switzerland[1]
karimlabib@google.com

Przemysław Uznański
Institute of Computer Science, University of Wrocław, Poland[1]
puznanski@cs.uni.wroc.pl

Daniel Wolleb-Graf
Department of Computer Science, ETH Zürich, Switzerland
daniel.graf@inf.ethz.ch

Abstract

We show, given a binary integer function $\diamond$ that is piecewise polynomial, that $(+, \diamond)$ vector products are equivalent under one-to-polylog reductions to the computation of the Hamming distance. Examples include the dominance and ℓ_{2p+1} distances for constant p. Our results imply equivalence (up to polylog factors) between the complexity of computing All Pairs Hamming Distance, All Pairs ℓ_{2p+1} Distance and Dominance Matrix Product, and equivalence between Hamming Distance Pattern Matching, ℓ_{2p+1} Pattern Matching and Less-Than Pattern Matching. The resulting algorithms for ℓ_{2p+1} Pattern Matching and All Pairs ℓ_{2p+1}, for $2p+1 = 3, 5, 7, \ldots$ are likely to be optimal, given lack of progress in improving upper bounds for Hamming distance in the past 30 years. While reductions between selected pairs of products were presented in the past, our work is the first to generalize them to a general class of functions, showing that a wide class of "intermediate" complexity problems are in fact equivalent.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases fine grained complexity, approximate pattern matching, matrix products

Digital Object Identifier 10.4230/LIPIcs.CPM.2019.14

Related Version A full version of the paper is available at [15], `https://arxiv.org/abs/1711.03887`.

1 Introduction

In the last few decades, many classical algorithmic problems received new attention when formulated as algebraic problems. In pattern matching, instead of looking for occurrences of a pattern as a substring of a text, we can define a similarity score between two strings and ask for this score between the pattern $\mathbf{P}$ of length m and every m-substring of the text $\mathbf{T}$ of length $n \geq m$. For example, scores of Hamming distance or L_1 distance between numerical strings generalize the classical pattern matching: the total score for a given alignment is zero iff the pattern occurs exactly there in the text. One can go in this framework even further, and consider a function that is not a metric, e.g. LessThanPatternMatching which outputs the number of coordinates for which the pattern element is no larger than the corresponding text element. However, all those problems share a common additive structure, where for an input pattern $\mathbf{P}$ and text $\mathbf{T}$, the score vector $\mathbf{O}$ is such that $\mathbf{O}[i] = \sum_j \mathbf{P}[j] \diamond \mathbf{T}[i+j]$ for some binary function $\diamond$.

Just as those pattern matching generalizations are based on *convolution*, there is a family of problems based on *matrix multiplication*, varying in flavour according to the vector product used. There, we are given two matrices A and B representing n vectors of dimension

[1] Most of the work was done while the authors were affiliated with ETH Zürich.

Table 1 Summary of different score functions and the corresponding problems. $\mathbf{1}[\varphi]$ is 1 iff φ and 0 otherwise.

Name	Score function	Pattern Matching problem	Matrix Product problem
Hamming	$\mathbf{1}[x \neq y]$	$\mathbf{O}[i] = \lvert\{j : \mathbf{P}[j] \neq \mathbf{T}[i+j]\}\rvert$	$O[i][j] = \lvert\{k : \mathbf{A}_i[k] \neq \mathbf{B}_j[k]\}\rvert$
Dominance	$\mathbf{1}[x \leq y]$	$\mathbf{O}[i] = \lvert\{j : \mathbf{P}[j] \leq \mathbf{T}[i+j]\}\rvert$	$O[i][j] = \lvert\{k : \mathbf{A}_i[k] \leq \mathbf{B}_j[k]\}\rvert$
Threshold	$\mathbf{1}[\lvert x-y\rvert \geq \delta]$	$\mathbf{O}[i] = \lvert\{j : \lvert\mathbf{P}[j] - \mathbf{T}[i+j]\rvert > \delta\}\rvert$	$O[i][j] = \lvert\{k : \lvert\mathbf{A}_i[k] - \mathbf{B}_j[k]\rvert > \delta\}\rvert$
ℓ_1 distance	$\lvert x-y\rvert$	$\mathbf{O}[i] = \sum_j \lvert\mathbf{P}[j] - \mathbf{T}[i+j]\rvert$	$O[i][j] = \sum_{k=1}^{n} \lvert\mathbf{A}_i[k] - \mathbf{B}_j[k]\rvert$
ℓ_2 distance	$(x-y)^2$	$\mathbf{O}[i] = \sum_j (\mathbf{P}[j] - \mathbf{T}[i+j])^2$	$O[i][j] = \sum_{k=1}^{n} (\mathbf{A}_i[k] - \mathbf{B}_j[k])^2$

d, and the output is the matrix $O[i][j] = \sum_k A[i][k] \diamond B[k][j]$. This is equivalent to the computation of all pairwise $(+, \diamond)$-vector products for two vector families, the so called MatrixProduct problems.

In both of those worlds, the complexity is spanned between *easy* and *hard* cases. An easy case is observed for e.g. $(+, \times)$ products, which have an upper bound of $\mathcal{O}(n \log n)$ for convolution (the classical Discrete Fast Fourier Transform algorithm) or $\mathcal{O}(n^\omega)$ (where $\omega < 2.373$ c.f. Le Gall [20]) for matrix multiplication. A hard case is considered to be respectively either near quadratic time or near cubic time problems. In the world of $(+, \diamond)$ vector products, we have not observed problems of the hard type, and instead they are either easy, or admit some intermediate complexity.[†] For many pattern matching generalizations there are independently achieved algorithms of identical complexity $\mathcal{O}(n\sqrt{m \log m})$. Similarly, for many MatrixProduct problems, the best algorithms are of complexity $\mathcal{O}(n^{(\omega+3)/2})$ or similar. Why are so many different problems of essentially the same complexity?

Our contribution

We show that there is a shared source of hardness to those problems. That is, we show that for a wide class of $(+, \diamond)$ products, the corresponding problems are of (almost) equivalent hardness. This class includes not only products like Hamming distance or Dominance, but in fact any piecewise polynomial function of two variables (for appropriate definition of piecewise polynomiality, c.f. Definition 3) excluding certain degenerate forms (e.g. polynomials). Thus, we show that we should not expect the problems based on $(+, \diamond)$ products to be significantly harder to compute than e.g. ones based on Hamming distance (given reasonable restrictions on $\diamond$). The reduction applies both to the Pattern Matching setting and to the Matrix Product setting alike. We refer to Table 1 for a summary of considered problems, to Table 2 for a summary of existing upper bounds, and to Figure 1 for a summary of the old and new reductions.

Yuster [32] improved the exponent of DominanceMatrixProduct (when $d = n$) from $(3+\omega)/2 \leq 2.6865$ to $\rho \leq 2.6834$, where ρ satisfies $\rho = \omega(1, 4-\rho, 1)$, and $\omega(a, b, c)$ is the exponent of fast multiplication of rectangular matrices $n^a \times n^b$ with $n^b \times n^c$. By our reduction, this improvement applies to all other MatrixProduct- problems considered here. Similarly, the tradeoff achieved for one problem (e.g. HammingDistanceMatrixProduct) between vectors of dimension d and the exponent (c.f. [25] and [14]) applies by our results to all the other MatrixProduct problems considered here. Looking at the sparsity of the input, the tradeoff between the number of relevant entries in the input and the runtime (c.f. Vassilevska [26], Vassilevska et. al. [28] and Duan and Pettie [10]) applies to all of the mentioned problems. (See Section 2 for precise upper bounds.)

[†] To observe good candidates for *hard* problems, we have to go beyond $(+, \diamond)$ products, and consider either $(\min, +)$ convolution (c.f. [9, 19]) or $(\min, +)$ matrix product (c.f. [30]).

Table 2 Overview of the known results and of how we abbreviate the corresponding problem names.

Name	Pattern Matching problem			Matrix Product problem		
Hamming	HAMPM	$\mathcal{O}(n\sqrt{m \log m})$	[1]	HAMPROD	$\mathcal{O}(n^{(\omega+3)/2})$	[25]
Dominance	LESSTHANPM	$\mathcal{O}(n\sqrt{m \log m})$	[2]	DOMPROD	$\mathcal{O}(n^{\rho})$	[32]
Threshold	THRPM	$\mathcal{O}(n\sqrt{m \log m})$	[5]	THRPROD	$\mathcal{O}(n^{(\omega+3)/2})$	[17]
ℓ_1 distance	L_1PM	$\mathcal{O}(n\sqrt{m \log m})$	[8, 3]	L_1PROD	$\mathcal{O}(n^{(\omega+3)/2})$	[17]
ℓ_2 distance	L_2PM	$\mathcal{O}(n \log m)$	[23]	L_2PROD	$\mathcal{O}(n^{\omega})$	[17]

We thus observe that there is a shared barrier in a broad class of problems and one is unlikely to improve upon existing upper bounds without some significant breakthrough. For both pattern matching problems and geometric problems we consider here, existing runtimes come from a tradeoff between the number of buckets and the size of these buckets. Without a novel technique, this runtime is unlikely to be improved. Similarly, any lower bound proof for one of the listed problems would immediately apply to every other problem.

Further applications

We provide the following further applications of our reductions.

- Since our reductions preserve structural properties of inputs like *size of Run Length Encoding*, in [13] they were used to establish equivalence between running time of HAMMINGDISTANCEPATTERNMATCHING and L_1PATTERNMATCHING on instances with bounded Run Length Encoding.
- In Censor-Hillel et al. [6] authors analyze the complexity of sparse matrix multiplication under the restricted bandwidth all-to-all communication model (the so called CONGESTED CLIQUE model). We note that their analysis immediately implies bounds to computation of Hamming Distance Matrix Product (and thus other matrix products as well) by presented, in the full version of this paper [15], bi-directional reductions to and from sparse matrix multiplication.
- Consider the problem of Image Template Matching, where one is given as an input a two-dimensional text T and a pattern P of dimensions $n \times n$ and $m \times m$ respectively. The goal is to compute the dissimilarity score between P and all $m \times m$-subsquares of T. Atallah in [4] gives an $\widetilde{\mathcal{O}}(n^2 m)$ running time algorithm for the L_1 version of this problem (so called *Sum of the Absolute Value* difference measure). We note, that by our reductions, an equivalence between the L_1, Hamming, Dominance and Threshold versions of this problem is established.

2 Related work

We now list different pattern matching problems that differ in their underlying score functions. The **HammingDistancePatternMatching**‡ was studied by Abrahamson [1], **LessThanPatternMatching** was introduced by Amir and Farach [2], **L_1PatternMatching** was studied first by Lipsky [21] and later in conference publications by Clifford, Clifford and Iliopoulous [8] and Amir, Lipsky, Porat and Umanski [3] (although the 2-dimensional version

‡ Also known as Pattern Matching with Mismatches.

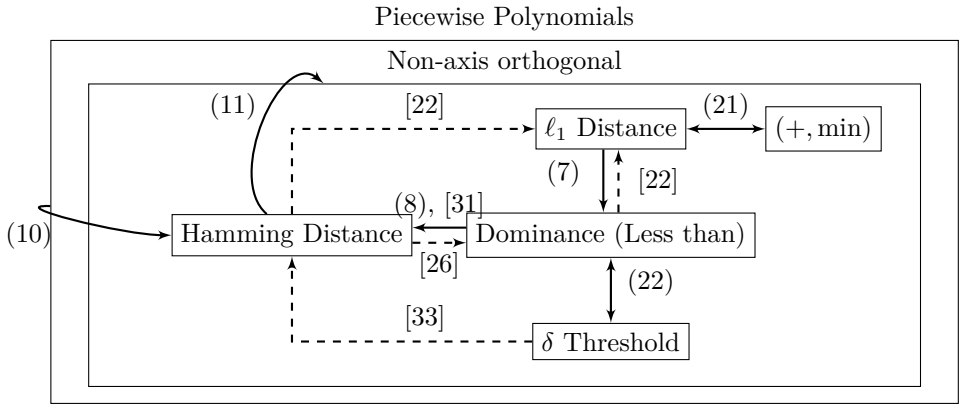

Figure 1 Existing (dashed line) and new (solid line) reductions between problems, together with problem classes.

of this problem was studied earlier by Atallah [4]) and **ThresholdPatternMatching** was studied by Atallah and Duket [5]. For all those problems, the currently best known algorithms run in time $\mathcal{O}(n\sqrt{m \log m})$ using similar techniques: high/low frequency, bucketing and convolution.

For $\mathbf{L_2}$**PatternMatching**, since $\mathbf{O}[i] = \sum_j \mathbf{P}[j]^2 + \sum_j \mathbf{T}[i+j]^2 - 2\sum_j \mathbf{P}[j]\mathbf{T}[i+j]$, the dominating term in the computation arises from computing a single convolution in time $\mathcal{O}(n \log m)$ via Fast Fourier Transform (FFT) as observed by Lipsky and Porat [23]. This approach extends to L_{2p}PM, with running time $\mathcal{O}(p^2 n \log m)$.

On the side of reductions, only little was known. Lipsky and Porat [22] showed that both HamPM and LessThanPM reduce to L_1PM showing that the latter problem is no easier than the former problems. The question of whether e.g. HamPM could be substantially easier than L_1PM remained open. The first non-trivial reduction (although not stated as a lower-bound type result) was provided by Zhang and Atallah [33], where they showed that ThrPM with threshold δ reduces to $\mathcal{O}(\log \delta)$ instances of HamPM (see Figure 1).

In computational geometry, a classical problem is to process a set of n points given in d-dimensional space. One can consider e.g. the metric space and ask for a pair of closest or farthest points. Some of those problems in low-dimensions (i.e. $d = \operatorname{poly} \log n$) exhibit a structure that allows for solutions almost linear in n for some metrics (see Williams [29]). However, in high-dimensional data, the situation is usually dire, as the so called *curse of dimensionality* kicks in (c.f. [18] and [16]) and for processing such spaces usually the fastest known approach is to compute all pairwise distances [17].

Those problems come in two flavours, AllPairs- where the input is a single matrix A and the goal is to compute all corresponding pairwise vector products between the rows of A, and MatrixProduct, where the input are two matrices A and B, and the goal is to compute products between rows of A and columns of B. Those two formulations are in fact equivalent by the folklore reduction.

DominanceMatrixProduct was introduced by Matoušek [24], where he provided a solution working in time $\mathcal{O}(n^{(\omega+3)/2}) \subseteq \mathcal{O}(n^{2.687})$. Vassilevska [26] and Vassilevska et al. [28] considered the dominance product on sparse inputs where we denote by m_1 and m_2 the number of entries in A and B, respectively that contribute to the score. They obtain a bound

of $\mathcal{O}(\min(n^\omega + \sqrt{m_1 m_2} \cdot n^{\frac{\omega-1}{2}}, n^2 + (m_1 m_2)^{\frac{\omega-2}{\omega-\alpha-1}} n^{\frac{2-\alpha\omega}{\omega-\alpha-1}}))$.[§] Duan and Pettie [10] simplified this analysis. For $d \ll n$, Vassilevska and Williams [27] and [26] gave an algorithm with a time bound of $\mathcal{O}(n^{\frac{2\omega-\omega\alpha-2}{\omega-\alpha-1}} d^{\frac{2\omega-4}{\omega-\alpha-1}} + n^{2+o(1)})$. Yuster [32] improved the bound of the case $d = n$ to $\mathcal{O}(n^\rho)$, where ρ is a solution to $\rho = \omega(1, 4-\rho, 1)$. The bound $\rho \leq 2.6834$ is provided. Recently, Gold and Sharir [14] presented an updated analysis of the time vs. dimension tradeoff using newer bounds on rectangular matrix multiplication. For $d = n$, this gives a running time of $\mathcal{O}(n^{2.6598})$.

AllPairsL$_1$Distance was considered by Indyk et al. [17], with an $\mathcal{O}(n^{(\omega+3)/2})$ algorithm for the case when $d = n$. Although not stated as such, one algorithm presented in [17] can be adapted to computing **ThresholdMatrixProducts** in time $\mathcal{O}(n^{(3+\omega)/2})$. [26] introduced **(+, max)-MatrixProduct** where the score matrix is $O[i][j] = \sum_{k=1}^{n} \max(\mathbf{A}_i[k], \mathbf{B}_j[k])$ and presented a bucketing solution with running time $\mathcal{O}(n^{(\omega+3)/2})$. The algorithm follows in spirit the L$_1$Prod algorithm from [17] with a tweaked score contribution. **ClosestHammingPair** was considered by Min et al. [25], where as an intermediate step the **AllPairsHamming-Distance** is computed (without actually naming the problem). Inspired by the reduction from Hamming distance to L_1 in [22], they utilized the L$_1$Prod algorithm from [17]. This resulted in a $\mathcal{O}(n^{(\omega+3)/2})$ time algorithm when $d = n$. They also utilized rectangular matrix multiplication bounds to provide a tradeoff in the complexity when $d \ll n$. Writing their upper bound in a general form, the complexity is $\mathcal{O}(n^{1+\omega(1,s,1)/2}\sqrt{d})$ where $d = n^s$. Given the improved bounds for rectangular matrix multiplication by Le Gall [11] and subsequently by Le Gall and Urrutia [12], the bounds from [25] are stronger. **AllPairsL$_2$Distance** as observed by Indyk et al. [17] reduces to a single matrix multiplication and thus admits a running time of $\mathcal{O}(n^\omega)$. Similarly, L$_{2p}$Prod admits a running time of $\mathcal{O}(p^2 n^\omega)$.

We observe that L_2 is an "easy" score function. For every other score function mentioned, all solutions presented use a bucketing or a high/low frequency technique to decompose the problem into ones solvable by convolution (for Pattern Matching problems) or matrix multiplication (for All Pairs problems). We refer to Tables 1 and 2 for a summary. There are several related problems that use the aforementioned problems as subroutines. **Weighted Pattern Matching** in the most general setting asks for $\mathbf{O}[i] = \sum_j w(P[j], T[i+j])$ for some weight function w. In [23] Lipsky and Porat presented a simple $\mathcal{O}(|\Sigma| n \log m)$ algorithm. **Pattern Matching with Wildcards** admits a simple determinsitic $\mathcal{O}(n \log m)$ solution via weighted L$_2$PM, as shown by Clifford and Clifford [7]. **Closest L_∞ Pair** was considered by Indyk et al. [17] where the presented algorithm uses binary search on top of ThrProd (implicitly, the intermediate problem they solve is not named). The total running time is $\mathcal{O}(n^{(\omega+3)/2} \log D)$, where D is the diameter of the input point set.

3 Preliminaries

The problems discussed previously have at their core the computation of a $(+, \diamond)$ vector product, that is $\sum_i x_i \diamond y_i$ for some binary function $\diamond$. Formally, for vectors $\mathbf{A}, \mathbf{B}$ and matrices $\mathcal{A}, \mathcal{B}$, we denote the $(+, \diamond)$ vector product as $\textsc{vprod}(\diamond, \mathbf{A}, \mathbf{B}) \stackrel{\text{def}}{=} \sum_i \mathbf{A}[i] \diamond \mathbf{B}[i]$, the $(+, \diamond)$ convolution as $\textsc{conv}(\diamond, \mathbf{A}, \mathbf{B}) = \mathbf{C}$ such that $\mathbf{C}[k] = \sum_{i+j=k} \mathbf{A}[i] \diamond \mathbf{B}[j]$ and the $(+, \diamond)$ matrix product as $\textsc{mprod}(\diamond, \mathcal{A}, \mathcal{B}) = \mathcal{C}$ such that $\mathcal{C}[i, j] = \sum_k \mathcal{A}[i, k] \diamond \mathcal{B}[k, j]$.

We define $\mathbf{1}[\varphi]$ to be 1 iff φ and 0 otherwise. Since $\mathsf{Ham}(x, y) \stackrel{\text{def}}{=} \mathbf{1}[x \neq y]$, then $\textsc{vprod}(\mathsf{Ham}, \mathbf{X}, \mathbf{Y})$ is the Hamming Distance between $\mathbf{X}$ and $\mathbf{Y}$, HamPM is essentially $\textsc{conv}(\mathsf{Ham}, \mathbf{X}, \mathbf{Y}^R)$, and HamProd between vectors $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_n\}$ is $\textsc{mprod}\left(\mathsf{Ham}, \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^T, \begin{bmatrix} Y_1 & \cdots & Y_n \end{bmatrix}\right)$.

[§] $\alpha = \sup\{0 \leq r \leq 1 : \omega(1, r, 1) = 2 + o(1)\} \geq 0.31389$.

We now shift our attention to the relations between the binary functions.

▶ **Definition 1.** *We say that $\diamond$ reduces preserving linearity to instances of $\square_1, \ldots, \square_K$ (for some positive integer K), if there are functions $f_1, \ldots, f_K$ and $g_1, \ldots, g_K$ and coefficients $\alpha_1, \ldots, \alpha_K$, such that for any x, y:*¶

$$x \diamond y = \sum_i \alpha_i \cdot \Big(f_i(x) \ \square_i \ g_i(y) \Big).$$

A one-to-many reduction from $\diamond$ to $\square$ is also a one-to-many reduction from $(+, \diamond)$ vector product/convolution/matrix multiplication to $(+, \square)$ vector product/convolution/matrix multiplication. Indeed, given Definition 1, we have for any vectors $\mathbf{A}, \mathbf{B}$ and matrices $\mathcal{A}, \mathcal{B}$: $\textsc{vprod}(\diamond, \mathbf{A}, \mathbf{B}) = \sum_i \alpha_i \cdot \textsc{vprod}(\square_i, f_i(\mathbf{A}), g_i(\mathbf{B}))$, $\textsc{conv}(\diamond, \mathbf{A}, \mathbf{B}) = \sum_i \alpha_i \cdot \textsc{conv}(\square_i, f_i(\mathbf{A}), g_i(\mathbf{B}))$ and $\textsc{mprod}(\diamond, \mathcal{A}, \mathcal{B}) = \sum_i \alpha_i \cdot \textsc{mprod}(\square_i, f_i(\mathcal{A}), g_i(\mathcal{B}))$, where $f(\mathbf{A})$ and $f(\mathcal{A})$ denotes a coordinate-wise application of f to vector $\mathbf{A}$ and matrix $\mathcal{A}$, respectively.

4 Main results

▶ Remark 2. We assume that all input values and coefficients are integers bounded in absolute value by M. All reductions presented use standard arithmetic operations and require thus $\operatorname{poly}\log M$ computation time.

▶ **Definition 3.** *For integers A, B, C and polynomial $P(x, y)$ we say that the function $P(x, y) \cdot \mathbf{1}[Ax + By + C > 0]$ is* halfplane polynomial. *We call a sum of halfplane polynomial functions a* piecewise polynomial. *We say that a function is* axis-orthogonal piecewise polynomial, *if it is piecewise polynomial and for every i, $A_i = 0$ or $B_i = 0$.*

Observe that $\mathsf{Ham}(x, y) = \mathbf{1}[x > y] + \mathbf{1}[x < y]$, $\max(x, y) = x \cdot \mathbf{1}[x \geq y] + y \cdot \mathbf{1}[x < y]$, $|x - y|^{2p+1} = (x - y)^{2p+1} \cdot \mathbf{1}[x > y] + (y - x)^{2p+1} \cdot \mathbf{1}[x < y]$, and $\mathsf{Thr}_\delta(x, y) \stackrel{\text{def}}{=} \mathbf{1}[|x - y| \geq \delta] = \mathbf{1}[x \leq y - \delta] + \mathbf{1}[x \geq y + \delta]$.

▶ **Theorem 4.** *Let $\diamond$ be a piecewise polynomial of constant degree and $\operatorname{poly}\log n$ number of summands.*

- *If $\diamond$ is axis orthogonal, then $\diamond$ is "easy": $(+, \diamond)$ convolution takes $\widetilde{O}(n)$ time, $(+, \diamond)$ matrix multiplication takes $\widetilde{O}(n^\omega)$ time.*
- *Otherwise, $\diamond$ is* Hamming distance complete*: under one-to-polylog reductions, on inputs bounded in absolute value by $\operatorname{poly}(n)$, $(+, \diamond)$ product is equivalent to Hamming distance, $(+, \diamond)$ convolution is equivalent to* HamPM *and $(+, \diamond)$ matrix multiplication is equivalent to* HamProd.

Theorem 4 "hard" case follows from two technical results presented in Section 6, Theorem 10 and Theorem 11. The "easy" case is resolved by Lemma 13.

▶ **Corollary 5.** *The following problems are equivalent under one-to-polylog reductions:* HamPM, LessThanPM, L_{2p+1}PM *for a constant integer $p \geq 0$,* ThrPM *and $(+, \max)$-*Convolution.

▶ **Corollary 6.** *The following problems are equivalent under one-to-polylog reductions:* HamProd, DomProd, L_{2p+1}Prod *for a constant integer $p \geq 0$,* ThrProd *and $(+, \max)$-*MatrixProduct.

¶ For the sake of simplicity, we are omitting in the definition the post-processing function necessary e.g. $(\,\cdot\,)^{1/p}$ for L_p norms.

5 Warm-up

We start by showing a reduction from ℓ_1 distance to $\mathcal{O}(\log^2 M)$ instances of Hamming distance, with an intermediate step of $\mathcal{O}(\log M)$ instances of dominance. Note that the reduction from dominance to $\mathcal{O}(\log M)$ Hamming distances follows from adapting reductions from [31] to our setting, and reduction from ℓ_1 to $\mathcal{O}(\log M)$ dominance relations follows as a natural adaptation of the same technique. However, since they serve as a nice overview of techniques used in our main result, and already have a nontrivial consequence (e.g. collapsing hardness of L_1PM and HamPM), we present them separately.

Scaling

Observe that for many "natural" functions $\diamond$ and integers x, y, $x \diamond y$ is approximated by $\lfloor x/2 \rfloor \diamond \lfloor y/2 \rfloor$ (up to some fixed multiplicative factor). This allows us to unwind $x \diamond y$ into a weighted sum of $\mathcal{O}(\log(\max(|x|, |y|))$ corrective terms. For example, if for some constant C, integers $x, y \geq 0$ and some corrective function ξ: $x \diamond y = C \cdot (\lfloor x/2 \rfloor \diamond \lfloor y/2 \rfloor) + \xi(x, y)$ then naturally $x \diamond y = 0 \diamond 0 + \sum_{i \geq 0} C^i \cdot \xi(\lfloor x/2^i \rfloor, \lfloor y/2^i \rfloor)$.

Sparsity

We consider a generalized version of the input with special "ignore" marks $\star$ as possible elements. Those elements of the input never contribute to the final score of the $(+, \diamond)$ product. Formally, we operate on $\mathbb{Z} + \{\star\}$, with special arithmetic rules (unless stated otherwise):

- for any *single argument* function: $f(\star) = \star$,
- for any *double argument* function: $g(\star, \star) = g(\star, y) = g(x, \star) = 0$.[‖]

The goal of this formalism is twofold. The first one is to handle sparse inputs formally (i.e. vectors with $\mathcal{O}(n^{1-\varepsilon})$ relevant entries). The second one is that such "ignore" marks coupled with filtering (defined below) allow us to split the input based on properties of its values. We note that these "ignore" marks do not increase the computational complexity of Hamming distance (see Lemma 9 in the Appendix).

Filtering

We define the following functions:

$$\texttt{even}(x) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } x \text{ is even} \\ \star & \text{otherwise} \end{cases} \qquad \texttt{odd}(x) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } x \text{ is odd} \\ \star & \text{otherwise} \end{cases}$$

Those functions, when applied to a vector or a matrix, allows us to filter values according to parity, e.g. for $\mathbf{A} = [1, 2, 3, 4]$ one gets $\texttt{even}(\mathbf{A}) = [\star, 2, \star, 4]$.

We now give two reductions that illustrate the usefulness of these techniques. Both reductions are illustrated in Appendix C (Figures 2 and 3). In the following theorems, recall that M is the largest possible integer input.

▶ **Theorem 7.** *The L_1 distance reduces to $\mathcal{O}(\log M)$ instances of dominance.*

[‖] We have to keep in mind that whether a function is a single or double argument is context dependent: e.g. writing: $\mathbf{1}[x \neq y] = 1 - \mathbf{1}[x = y]$, we have to treat 1 as a function of x and y as well.

Proof. Since L_1 distance is shift-invariant, i.e. $|(x+\Delta)-(y+\Delta)| = |x-y|$ for any Δ, we can assume that $0 \le x, y < M$ for some $M = \text{poly}(n)$. Observe that for $x, y \ge 0$, $|x-y| = 2 \cdot \Big|\lfloor x/2 \rfloor - \lfloor y/2 \rfloor\Big| + \eta(x,y)$, where, denoting $\mathsf{Dom}(x,y) \stackrel{\text{def}}{=} \mathbf{1}[x \le y]$,

$$\begin{aligned}\eta(x,y) &= \mathbf{1}[(x \text{ is odd}) \wedge (y \text{ is even}) \wedge (x \ge y)] - \mathbf{1}[(x \text{ is even}) \wedge (y \text{ is odd}) \wedge (x \ge y)] \\ &\quad + \mathbf{1}[(y \text{ is odd}) \wedge (x \text{ is even}) \wedge (y \ge x)] - \mathbf{1}[(y \text{ is even}) \wedge (x \text{ is odd}) \wedge (y \ge x)] \\ &= \mathsf{Dom}(\mathtt{odd}(-x), \mathtt{even}(-y)) - \mathsf{Dom}(\mathtt{even}(-x), \mathtt{odd}(-y)) \\ &\quad + \mathsf{Dom}(\mathtt{even}(x), \mathtt{odd}(y)) - \mathsf{Dom}(\mathtt{odd}(x), \mathtt{even}(y)).\end{aligned}$$

By unwinding, we get $|x-y| = \sum_{i=0}^{\log M} 2^i \cdot \eta(\lfloor x/2^i \rfloor, \lfloor y/2^i \rfloor)$ which completes the reduction. ◀

▶ **Theorem 8.** *Dominance reduces to $\mathcal{O}(\log M)$ instances of Hamming distance and multiplication.*

Proof. Since dominance is shift-invariant, w.l.o.g. we assume that $0 \le x, y < M$ for some $M = \text{poly}(n)$. Observe the following recurrence relation, for $x, y \ge 0$:

$$\begin{aligned}\mathsf{Dom}(x,y) &= \mathsf{Dom}(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor) - \mathbf{1}[(x \text{ is odd}) \wedge (x = y+1)] \\ &= \mathsf{Dom}(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor) - \mathbf{1}[x \text{ is odd}] + \mathbf{1}[x \text{ is odd}] \cdot \mathsf{Ham}(x, y+1)\end{aligned}$$

By unwinding, we get:

$$\mathsf{Dom}(x,y) = 1 - \sum_{i=0}^{\log M} \mathbf{1}\big[\lfloor x/2^i \rfloor \text{ is odd}\big] + \sum_{i=0}^{\log M} \mathbf{1}\big[\lfloor x/2^i \rfloor \text{ is odd}\big] \cdot \mathsf{Ham}(\lfloor x/2^i \rfloor, \lfloor y/2^i \rfloor + 1).$$

Using filtering notation, this becomes

$$\mathsf{Dom}(x,y) \quad = \quad \underbrace{1 - \sum_{i=0}^{\log M} \mathbf{1}\big[\lfloor x/2^i \rfloor \text{ is odd}\big]}_{(*)} + \underbrace{\sum_{i=0}^{\log M} \mathsf{Ham}(\mathtt{odd}(\lfloor x/2^i \rfloor), \lfloor y/2^i \rfloor + 1)}_{(**)}$$

Now observe, that $(*)$ is purely a function of x. If x is guaranteed to be an integer, then evaluating it as part of an operator (i.e. inside convolution or matrix-multiplication) is trivial. As y is never mapped to $\star$ in $(**)$, treating $(*)$ as a single argument function suffices.

The second term $(**)$ uses our filtering function and the convention that Ham evaluates to 0 if at least one of its inputs is $\star$. Thus $(**)$ is a sum of $\mathcal{O}(\log n)$ Hamming distances on inputs from $\mathbb{Z} \cup \{\star\}$. By Lemma 9, each of those reduces to two instances of Hamming distance on inputs from $\mathbb{Z}$. ◀

▶ **Lemma 9.** *Hamming distance in $\mathbb{N} + \{\star\}$ reduces preserving linearity to two instances of Hamming distance in $\mathbb{N}$.*

▶ Remark. In general, we have to take into account that both $x, y \in \mathbb{Z} \cup \{\star\}$. Thus, we have to treat term $(*)$ as a function of *both* x and y, that is evaluating to 0 if $x = \star$ or $y = \star$. In general, $(*)$ reduces to evaluating, after the reduction step, some polynomial $Q(x', y') = f(x')$ (where y' might be $\star$) with $f(x') = 1 - \sum_{i=0}^{\log M} \mathbf{1}\big[\lfloor x'/2^i \rfloor \text{ is odd}\big]$. By Lemma 13, $f(x')$ can be resolved in the time of a regular convolution or matrix multiplication and thus the computation time for $(*)$ is dominated by $(**)$, that is HamPM and HamProd, respectively.

6 Hamming distance completeness

The goal of this section is proving Theorem 4. We achieve this by showing two separate reductions, one from *all piecewise polynomial functions* to Hamming distance and one from Hamming distance to all non axis-orthogonal piecewise polynomials.

▶ **Theorem 10.** *If $\diamond$ is a piecewise polynomial of degree d with c summands then it reduces to $\mathcal{O}(c \cdot d^2 \cdot \log^{d+1} M)$ instances of Hamming distance. The reduction works even if we allow "ignore" symbols.*

▶ **Theorem 11.** *If $\diamond$ is a piecewise polynomial of degree d but is not axis-orthogonal piecewise polynomial, then Hamming distance reduces to $\mathcal{O}(d^2)$ instances of $\diamond$ and multiplication.*

We devote the rest of this section to the proofs of both Theorem 10 and Theorem 11.

To prove Theorem 10, we consider every summand separately. We show that summands with "simple" conditions (that depend on only one argument) are no harder than simple multiplication. Every other summand with conditional term $\mathbf{1}[A_i x + B_i y + C_i > 0]$ reduces under linear transformations of its arguments to $\mathbf{1}[x < y]$. It is thus enough to consider terms of the form $x^a y^b \mathbf{1}[x < y]$. We decompose such terms recursively into a sum of: terms with smaller values ($x/2, y/2$ instead of x, y), terms of smaller degree, and terms with a conditional term of a simpler form of $\mathbf{1}[x = y]$. Exhaustively applying this decomposition leaves us with a polylog number of terms of the form $w(x) \cdot \mathbf{1}[x = y]$, with which we deal separately (those decompose into a logarithmic number of regular Hamming distances).

▶ **Lemma 12.** *For an integer weight function w, the character weighted matches, that is $w(x) \cdot \mathbf{1}[x = y]$, reduce to $\mathcal{O}(\log M)$ instances of Hamming distance and multiplication.*

▶ **Lemma 13.** *An axis-orthogonal piecewise polynomial $\diamond$ of c summands of degree d reduces to $\mathcal{O}(d^2 c)$ multiplications.*

▶ **Lemma 14.** *Given integers $a, b \geq 0$, the binary function $x^a y^b \cdot \mathbf{1}[x < y]$ reduces to $\mathcal{O}(\log^{a+b+1} M)$ instances of Hamming distance and multiplication.*

Proof. Denote $\mathrm{MDom}_{a,b}(x, y) = x^a y^b \cdot \mathbf{1}[x < y]$, $\mathrm{MEq}_a(x, y) = x^a \cdot \mathbf{1}[x = y]$. First, we argue that w.l.o.g. $x, y \geq 0$. Indeed, observe that $\mathrm{MDom}_{a,b}(x+\Delta, y+\Delta) = (x+\Delta)^a (y+\Delta)^b \cdot \mathbf{1}[x < y]$, thus for large enough Δ, the computation of $\mathrm{MDom}_{a,b}$ on inputs of arbitrary sign reduces to at most $(a+1)(b+1)$ instances of MDom on non-negative inputs. Thus we assume that $0 \leq x, y \leq M$ for some $M = \mathrm{poly}(n)$.

We proceed with the following decomposition, where $u = \lfloor \frac{x}{2} \rfloor$ and $v = \lfloor \frac{y}{2} \rfloor$.

$$\begin{aligned} \mathrm{MDom}_{a,b}(x, y) &= (2u)^a (2v)^b \cdot \mathbf{1}[u < v] && (*) \\ &+ (2u)^a (2v)^b \cdot \Big(\mathbf{1}[x < y] - \mathbf{1}[u < v] \Big) && (**) \\ &+ \big(x^a y^b - (2u)^a (2v)^b \big) \cdot \mathbf{1}[x < y] && (***) \end{aligned}$$

Intuitively, we are representing the term $\mathrm{MDom}_{a,b}(x, y)$ with simpler terms: (*) represents rounding down x, y to even numbers $2u, 2v$, (**) is the corrective term for the indicator part and (***) is the corrective term for the monomial part. Simplifying those terms separately, we have

$$\begin{aligned} (*) &= 2^{a+b} \cdot \mathrm{MDom}_{a,b}(u, v), \\ (**) &= x^a (y-1)^b \cdot \mathbf{1}[\mathtt{even}(x) = \mathtt{odd}(y) - 1] = \mathrm{MEq}_{a+b}(\mathtt{even}(x), \mathtt{odd}(y) - 1), \\ (***) &= P_{a,b}(x, y) \cdot \mathbf{1}[\mathtt{odd}(x) < \mathtt{even}(y)] + Q_{a,b}(x, y) \\ &\quad \cdot \mathbf{1}[\mathtt{even}(x) < \mathtt{odd}(y)] + R_{a,b}(x, y) \cdot \mathbf{1}[\mathtt{odd}(x) < \mathtt{odd}(y)], \end{aligned}$$

where $P_{a,b}(x,y) = (x^a y^b - (x-1)^a y^b)$, $Q_{a,b}(x,y) = (x^a y^b - x^a (y-1)^b)$ and $R_{a,b}(x,y) = (x^a y^b - (x-1)^a (y-1)^b)$.

All in all, our recursion decomposes $\mathrm{MDom}_{a,b}(x,y)$ into several terms – either with the inputs reduced by a factor of 2, the test for dominance replaced with a test for equality, or to monomials of smaller degree (observe that each of $P_{a,b}(x,y)$, $Q_{a,b}(x,y)$ and $R_{a,b}(x,y)$ is of degree at most $a+b-1$). Let $T(a,b,m)$ denote the number of instances of Hamming distance that a single instance of $\mathrm{MDom}_{a,b}$, with inputs bounded in value by 2^m, is reduced to. Since by Lemma 12, MEq_{a+b} reduces to $\mathcal{O}(m \cdot (a+b))$ instances of Hamming distance, there is

$$T(a,b,m) \leq \mathcal{O}(m \cdot (a+b)) + T(a,b,m-1) + \sum_{\substack{0 \leq i \leq a \\ 0 \leq j \leq b \\ (i,j) \neq (a,b)}} 3T(i,j,m),$$

which is satisfied (for some constant C) by $T(a,b,m) \leq C \cdot m \cdot (a+b) \cdot \binom{a+b+m}{a,b,m} \cdot 4^a \cdot 4^b$. For fixed values a, b this is $\mathcal{O}(\log^{a+b+1} M)$. ◀

Proof of Theorem 10. Consider an arbitrary piecewise-polynomial binary function $\diamond$. Consider its summand $P(x,y) \cdot \mathbf{1}[Ax + By + C > 0]$. If $A = 0$ or $B = 0$ then it reduces to a binary function of degenerated form $P(x,y) \cdot \mathbf{1}[Ax + C > 0]$ which in turn reduces to $\mathcal{O}(d^2)$ multiplications by Lemma 13.

Otherwise, if $A \neq 0$ and $B \neq 0$, then there is a one-to-one linear input reduction, $u = -Ax$ and $v = By + C$, that reduces from $(-Ax)^i (By + C)^j \cdot \mathbf{1}[Ax + By + C > 0]$ to $u^i v^j \cdot \mathbf{1}[u < v]$. Note that any polynomial of degree a and b over x and y is a linear combination of $(-Ax)^i (By + C)^j$ for $0 \leq i \leq a$ and $0 \leq j \leq b$.

By applying those reductions to each summand and applying Lemma 14 to each monomial of the summand, we reach the claimed bound. ◀

To prove Theorem 11, we need the following technical Lemma:

▶ **Lemma 15.** *Consider a family of distinct lines $\Lambda = \{\lambda_i\}_{i=1}^{|\Lambda|}$, $\lambda_i = \{x, y : A_i x + B_i y + C_i = 0\}$ for integers A_i, B_i, C_i such that $|A_i|, |B_i|, |C_i| \leq M$. If there is at least one $\lambda \in \Lambda$ that is not axis-orthogonal, then there exists $\lambda_i \in \Lambda$ and $\alpha, \beta, \gamma, \delta$ such that:*

- *for any line λ_j that is not parallel to λ_i, the set $\{(\alpha x + \gamma, \beta y + \delta) : x, y \in [0 \dots N]\}$ lies* on the same side *of λ_j,*
- *for any line λ_j that is parallel to λ_i, the sets $\{(\alpha x + \gamma, \beta y + \delta) : x > y\}$ and $\{(\alpha x + \gamma, \beta y + \delta) : x < y\}$ are separated by λ_j.*

Moreover, $|\alpha|, |\beta|, |\gamma|, |\delta| \leq \mathrm{poly}(M, N)$.

Proof. Pick λ_i that is not axis-orthogonal, that is $A_i, B_i \neq 0$.

Let us denote the grid $\mathcal{G} = \{(\alpha x + \gamma, \beta y + \delta) : x, y \in [0 \dots N]\}$. To guarantee that main diagonal of $\mathcal{G}$ lies on λ_i, we need to have $\alpha = B_i \cdot k$ and $\beta = -A_i \cdot k$ for some nonzero integer k, and select values of γ, δ accordingly so that $(\gamma, \delta) \in \lambda_i$.

For non-parallel λ_i, λ_j, the coordinates of intersection point are:

$$x_{i,j} = -\begin{vmatrix} C_i & B_i \\ C_j & B_j \end{vmatrix} / \begin{vmatrix} A_i & B_i \\ A_j & B_j \end{vmatrix} \qquad y_{i,j} = -\begin{vmatrix} A_i & C_i \\ A_j & C_j \end{vmatrix} / \begin{vmatrix} A_i & B_i \\ A_j & B_j \end{vmatrix}.$$

To guarantee that whole $\mathcal{G}$ lies on the same side of λ_j, it is enough to make sure that all 4 corners are on the same side. However, we observe that iff e.g. corners (γ, δ) and $(\alpha N + \gamma, \delta)$ are separated by λ_j, it means that for some $r \in [0,1]$ lines λ_j and $A_i(x - r\alpha N) + B_i y + C_i = 0$

(that is λ_i shifted in x by $+r\alpha N$) intersect on point with $x = \delta$. To satisfy the first condition of the lemma, it is enough if every point of the convex closure of $\mathcal{G}$ has x coordinate with absolute value at least $2M^2 + |\alpha|MN$, since that is larger than any possible intersection point as described above (condition (a)). Similarly for y coordinate it should be at least $2M^2 + |\beta|MN$ (condition (b)).

Take λ_j parallel to λ_i, that is they differ only on value of C. We first make sure that all such λ_j fall between lines $\{(\alpha t + \gamma, \beta(t+1) + \delta) : t \in \mathbb{R}\}$ and λ_i or λ_i and $\{(\alpha(t+1) + \gamma, \beta t + \delta) : t \in \mathbb{R}\}$ (those lines are λ_i "shifted" one step up or down in the grid), by making sure α and β are large enough in absolute value. Indeed, it is enough to have $|\alpha A_i| = |\beta B_i| > 2M$ being largest possible difference between two values of C. It is enough to select $k = 3M$, and $\alpha = 3MB_i, \beta = -3MA_i$.

We then select γ and δ as smallest in absolute value points of λ_i such that conditions (a) and (b) are satisfied. ◀

Proof of Theorem 11. Let us take the binary function $x \diamond y = \sum_i P_i(x,y) \cdot \mathbf{1}[A_i x + B_i y + C_i > 0]$ as in the theorem statement, assuming it is of the simplest form (no redundant terms and minimal number of summands possible). We construct a reduction from Hamming distance to $\diamond$ by a series of intermediate operators.

Let d be the highest degree of any $P_1, P_2, \ldots$. Consider all the lines being borders of regions, that is $\lambda_i = \{(u,v) : A_i u + B_i v + C = 0\}$ (as elements of the continuous Euclidean plane).

We now apply Lemma 15, with $N = 3dM + 2d$. Consider $F(x,y) \stackrel{\text{def}}{=} (\alpha x + \gamma) \diamond (\beta y + \delta)$. Limited to $x, y \in [0 \ldots N]$, $F(x,y)$ is piecewise linear of a much simpler form:

$$F(x,y) = Q_>(x,y) \cdot \mathbf{1}[x > y] + Q_=(x,y) \cdot \mathbf{1}[x = y] + Q_<(x,y) \cdot \mathbf{1}[x < y]$$

for $Q_>, Q_=, Q_<$ being polynomials of degree at most d, and $Q_< \not\equiv Q_>$. Let D_x, D_y be the operators of discrete differentiation, that is $D_x F(x,y) \stackrel{\text{def}}{=} F(x+1,y) - F(x,y)$, $D_y F(x,y) \stackrel{\text{def}}{=} F(x,y+1) - F(x,y)$. There are integers $0 \le a, b \le d$ such that $D_x^a D_y^b (Q_<(x,y) - Q_>(x,y)) \equiv c$ for some constant $c \neq 0$. Thus if we consider the function:

$$G(x,y) \stackrel{\text{def}}{=} \frac{1}{c} \cdot D_x^a D_y^b (F(x,y) - Q_>(x,y)),$$

it has the following properties on $x, y \in [0 \ldots N - d]$: for $y - x > d$: $G(x,y) = 1$, and for $y - x < -d$: $G(x,y) = 0$. We observe that for $x, y \in [0 \ldots M]$, there is $\mathsf{Dom}(x,y) = G(3d \cdot x, 3d \cdot y + d)$. All in all, Ham reduces to $\mathcal{O}(d^2)$ instances of $\diamond$ and a single evaluation of a fixed polynomial $Q_>(x,y)$, which reduces to $\mathcal{O}(d^2)$ multiplications. ◀

7 Conclusion

There are several immediate applications of Theorem 10 and Theorem 11. The first one is that the improvement to DomProd from [32] translates to other MatrixProduct problems:

▶ **Corollary 16.** DomProd, L_1Prod, L_{2p+1}Prod, ThrProd, HamProd *and* $(+, \min)$-MatrixProduct *are solvable in time* $\widetilde{\mathcal{O}}(n^\rho)$*, where* $\rho \le 2.6834$ *is a solution to* $\rho = \omega(1, 4 - \rho, 1)$*.*

Observe that the reductions we presented map $\star$ to $\star$. Thus, e.g. by [26],[28] and [10], we immediately get that all considered MatrixProduct problems are of the same complexity

even on sparse inputs, up to a poly $\log n$ multiplicative term and additive term of time it takes to compute (classical) sparse matrix product of relevant matrices (which is a simpler problem).

▶ **Corollary 17.** *Consider sparse inputs where we denote by m_1 and m_2 the number of entries in A and B that contribute to the score, where A and B are matrices of n vectors of dimension n.* DomProd*,* L_1Prod*,* L_{2p+1}Prod*,* ThrProd*,* HamProd *and* $(+, \min)$-MatrixProduct *are solvable in time* $\widetilde{\mathcal{O}}(\min(n^\omega + \sqrt{m_1 m_2} \cdot n^{\frac{\omega-1}{2}}, n^2 + (m_1 m_2)^{\frac{\omega-2}{\omega-\alpha-1}} n^{\frac{2-\alpha\omega}{\omega-\alpha-1}}))$.

Since our reductions preserve the dimension of the problems, any tradeoff between $d \ll n$ and the running time translates to all other problems as well, with a poly $\log n$ multiplicative term and a $\widetilde{\mathcal{O}}(n^\omega)$ additive term. One can improve the running time of the algorithm presented in [25] using the trick of batch-processing via rectangular matrix multiplication in [32], as done for Dominance Product in [14], to obtain the following time complexity:

▶ **Corollary 18.** *For n vectors of dimension $d = n^s$ for $0 \leq s \leq 1$,* DomProd*,* L_1Prod*,* L_{2p+1}Prod*,* ThrProd*,* HamProd *and* $(+, \min)$-MatrixProduct *are solvable in time* $\widetilde{\mathcal{O}}(n^{\rho(s)})$ *where* $\rho(s) = \inf\{x : 2 \leq x \leq 3 \text{ and } \omega(1, 2+2s-x, 1) \geq x\}$*. In particular, for* $d = \mathcal{O}(n^{\alpha/2}) \supseteq \mathcal{O}(n^{0.156945})$ *all those problems are solvable in time* $\widetilde{\mathcal{O}}(n^2)$*.*

Similarly, one can look into the relation between sparsity and running time for pattern matching problems. Here, we obtain the following result:

▶ **Theorem 19.** *For a text of length n and a pattern of length m, $n \geq m$, with s_t and s_p relevant entries, respectively, the running time of* HamPM*,* LessThanPM*,* ThrPM *and* L_{2p+1}PM *is* $\widetilde{\mathcal{O}}(\sqrt{n s_t s_p} + n)$*.*

We present the following application of the scaling/filtering framework: weighted mismatches. We distinguish between *position weighted mismatches* and *character weighted mismatches*. In the pattern matching setting, the former asks for $\mathbf{O}[i] = \sum_{j:\mathbf{P}[j] \neq \mathbf{T}[i+j]} w(j)$, whereas the latter asks for $\mathbf{O}[i] = \sum_{j:\mathbf{P}[j] \neq \mathbf{T}[i+j]} w(\mathbf{P}[j])$, for some given weight function $w : \mathbb{Z} \to \mathbb{Z}$. We see that character weighted mismatches are expressible by a function $w(x) \cdot \mathbf{1}[x \neq y]$ and get by Lemma 12 that Hamming Distance Pattern Matching with Character Weights is no harder than HamPM (up to a $\log n$ factor). For position weights, we present the following:

▶ **Theorem 20.** *Hamming Distance Pattern Matching with Position Weights reduces to $\mathcal{O}(\log n)$ instances of* HamPM*.*

While it is no surprise that for example the technique of [32] can be applied to other MatrixProduct problems, it is a nice side effect of our reduction that it can be applied "automatically" without looking deeper into the structure of any of the different MatrixProduct problems involved. The reductions presented signify that regardless of whether we are looking for improved upper bounds, or new lower bounds, it is enough to concentrate on a single score function from the whole class of equivalent functions. In our opinion, Hamming distance is the "cleanest" score function, since it is the simplest – it assumes no arithmetic underlying structure of the alphabet (unlike e.g. L_1 distance) and not even an ordering of the alphabet.

References

1 Karl R. Abrahamson. Generalized String Matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.

2 Amihood Amir and Martin Farach. Efficient matching of nonrectangular shapes. *Annals of Mathematics and Artificial Intelligence*, 4(3):211–224, September 1991. doi:10.1007/BF01531057.

3 Amihood Amir, Ohad Lipsky, Ely Porat, and Julia Umanski. Approximate Matching in the L_1 Metric. In *CPM*, pages 91–103, 2005. doi:10.1007/11496656_9.

4 Mikhail J. Atallah. Faster image template matching in the sum of the absolute value of differences measure. *IEEE Trans. Image Processing*, 10(4):659–663, 2001. doi:10.1109/83.913600.

5 Mikhail J. Atallah and Timothy W. Duket. Pattern matching in the Hamming distance with thresholds. *Inf. Process. Lett.*, 111(14):674–677, 2011. doi:10.1016/j.ipl.2011.04.004.

6 Keren Censor-Hillel, Dean Leitersdorf, and Elia Turner. Sparse Matrix Multiplication and Triangle Listing in the Congested Clique Model. In *OPODIS 2018*, pages 4:1–4:17, 2018. doi:10.4230/LIPIcs.OPODIS.2018.4.

7 Peter Clifford and Raphaël Clifford. Simple deterministic wildcard matching. *Inf. Process. Lett.*, 101(2):53–54, 2007. doi:10.1016/j.ipl.2006.08.002.

8 Peter Clifford, Raphaël Clifford, and Costas S. Iliopoulos. Faster Algorithms for δ,γ-Matching and Related Problems. In *CPM*, pages 68–78, 2005. doi:10.1007/11496656_7.

9 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On Problems Equivalent to (min, +)-Convolution. In *ICALP*, pages 22:1–22:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.22.

10 Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *SODA*, pages 384–391, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496813.

11 François Le Gall. Faster Algorithms for Rectangular Matrix Multiplication. In *FOCS*, pages 514–523, 2012. doi:10.1109/FOCS.2012.80.

12 Francois Le Gall and Florent Urrutia. Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In *SODA*, pages 1029–1046, 2018. doi:10.1137/1.9781611975031.67.

13 Paweł Gawrychowski and Przemysław Uznański. Towards Unified Approximate Pattern Matching for Hamming and L_1 Distance. In *ICALP 2018*, pages 62:1–62:13, 2018. doi:10.4230/LIPIcs.ICALP.2018.62.

14 Omer Gold and Micha Sharir. Dominance Product and High-Dimensional Closest Pair under L_∞. In *ISAAC*, pages 39:1–39:12, 2017. doi:10.4230/LIPIcs.ISAAC.2017.39.

15 Daniel Graf, Karim Labib, and Przemysław Uznański. Hamming distance completeness and sparse matrix multiplication. *CoRR*, abs/1711.03887, 2017. arXiv:1711.03887.

16 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

17 Piotr Indyk, Moshe Lewenstein, Ohad Lipsky, and Ely Porat. Closest Pair Problems in Very High Dimensions. In *ICALP*, pages 782–792, 2004. doi:10.1007/978-3-540-27836-8_66.

18 Eamonn J. Keogh and Abdullah Mueen. Curse of Dimensionality. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 314–315. Springer, 2017. doi:10.1007/978-1-4899-7687-1_192.

19 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *ICALP 2017*, pages 21:1–21:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.21.

20 François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *ISSAC*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

21 Ohad Lipsky. Eficient Distance Computations. Master's thesis, Bar-Ilan University. Department of Mathematics and Computer Science., 2003.

22 Ohad Lipsky and Ely Porat. L_1 pattern matching lower bound. *Inf. Process. Lett.*, 105(4):141–143, 2008. doi:10.1016/j.ipl.2007.08.011.

23 Ohad Lipsky and Ely Porat. Approximate Pattern Matching with the L_1, L_2 and L_∞ Metrics. *Algorithmica*, 60(2):335–348, 2011. doi:10.1007/s00453-009-9345-9.

24 Jiří Matoušek. Computing Dominances in E^n (Short Communication). *Inf. Process. Lett.*, 38(5):277–278, June 1991. doi:10.1016/0020-0190(91)90071-O.

25 Kerui Min, Ming-Yang Kao, and Hong Zhu. The Closest Pair Problem under the Hamming Metric. In *COCOON*, pages 205–214, 2009. doi:10.1007/978-3-642-02882-3_21.

26 Virginia Vassilevska. *Efficient algorithms for path problems in weighted graphs*. PhD thesis, Carnegie Mellon University, 2008.

27 Virginia Vassilevska and Ryan Williams. Finding a Maximum Weight Triangle in $n^{3-\delta}$ Time, with Applications. In *STOC*, pages 225–231, 2006. doi:10.1145/1132516.1132550.

28 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All Pairs Bottleneck Paths and Max-Min Matrix Products in Truly Subcubic Time. *Theory of Computing*, 5(1):173–189, 2009. doi:10.4086/toc.2009.v005a009.

29 Ryan Williams. On the Difference Between Closest, Furthest, and Orthogonal Pairs: Nearly-Linear vs Barely-Subquadratic Complexity. In *SODA*, pages 1207–1215, 2018. doi:10.1137/1.9781611975031.78.

30 Virginia Vassilevska Williams and Ryan Williams. Subcubic Equivalences between Path, Matrix and Triangle Problems. In *FOCS 2010*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

31 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

32 Raphael Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In *SODA*, pages 950–957, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496873.

33 Peng Zhang and Mikhail J. Atallah. On approximate pattern matching with thresholds. *Inf. Process. Lett.*, 123:21–26, 2017. doi:10.1016/j.ipl.2017.03.001.

A Omitted proofs from Section 6

▶ **Lemma 12.** *For an integer weight function w, the character weighted matches, that is $w(x) \cdot \mathbf{1}[x = y]$, reduce to $\mathcal{O}(\log M)$ instances of Hamming distance and multiplication.*

Proof. Let M be the upper bound on all values of w in the considered domain of inputs. Given two integers x, y, we observe the following equality:

$$w(x) \cdot [x = y] \quad = \quad \sum_{i=0}^{\log M} 2^i \cdot \mathbf{1}[w_i(x) = w_i(y)]$$

where the filtering function w_i is defined based on w:

$$w_i(x) = \begin{cases} x & i\text{-th bit of } w(x) \text{ is } 1 \\ \star & \text{otherwise.} \end{cases}$$

Observing that $\mathbf{1}[x = y] = 1 - \mathsf{Ham}(x, y)$ finishes the proof. ◀

▶ **Lemma 13.** *An axis-orthogonal piecewise polynomial $\diamond$ of c summands of degree d reduces to $\mathcal{O}(d^2 c)$ multiplications.*

Proof. Given an axis orthogonal piecewise polynomial $F(x, y) = \sum_{i=1}^{c} P_i(x, y) \cdot \mathbf{1}[A_i x + B_i y + C_i > 0]$ of degree d. Consider summand $P_i(x, y)\mathbf{1}[A_i x + C_i > 0]$ (w.l.o.g. we assume that $B_i = 0$ and $A_i \neq 0$). Consider a monomial of $P_i(x, y)$, e.g. $x^a y^b$. Define $x' = x^a$ iff $A_i x + C_i > 0$ and $x' = \star$ otherwise, and $y' = y^b$. Then $x^a y^b \cdot \mathbf{1}[A_i x + C_i > 0] = x' y'$. ◀

Axis orthogonal piecewise polynomial $\diamond$ are no harder than multiplication in e.g. vector convolution or matrix multiplication. By Theorem 13 it reduces to multiplication in $\mathbb{Z} \cup \{\star\}$, which in turn reduces to multiplication in $\mathbb{Z}$. Indeed, it is enough to consider a map $\mathbb{Z} \cup \{\star\} \to \mathbb{Z}$ that is identity on $\mathbb{Z}$ and maps $\star \to 0$.

▶ **Theorem 19.** *For a text of length n and a pattern of length m, $n \geq m$, with s_t and s_p relevant entries, respectively, the running time of* HamPM, LessThanPM, ThrPM *and* L_{2p+1}PM *is* $\widetilde{\mathcal{O}}(\sqrt{ns_ts_p} + n)$.

Proof. Consider LessThanPM. The proof follows the non-sparse case. W.l.o.g. all the $2s_p$ actual entries are distinct integers (if it is not so, they can be made so using small $\varepsilon > 0$ shifts and then re-arranged back into integers preserving order). The s_p relevant entries of the pattern are sorted and partitioned into k buckets $B_1, \ldots, B_k$ so that B_1 gets s_p/k smallest elements, B_2 following s_p/k smallest elements, etc. We get inter-bucket contribution for bucket B_i from convolution of P_i^R with T_i, where P_i, T_i are binary strings such that $P_i[j] = 1$ iff $P[j] \in B_1 \cup \ldots \cup B_i$ and $T_i[j] = 1$ iff $T[j] \in B_i$. This in a total takes $\mathcal{O}(kn \log m)$ time for all k buckets. Intra-bucket contributions are captured in a brute force manner in $\mathcal{O}(s_ts_p/k)$ where each relevant text element is compared with at most s_p/k elements in its corresponding bucket. Choosing k to be $\max(1, \sqrt{(s_ts_p)/(n \log m)})$ gives the time bound of $\widetilde{\mathcal{O}}(n + \sqrt{ns_ts_p})$. ◀

▶ **Theorem 20.** *Hamming Distance Pattern Matching with Position Weights reduces to* $\mathcal{O}(\log n)$ *instances of* HamPM.

Proof. We solve $\mathcal{O}(\log n)$ instances of HamPM with filtering involved. This is done by constructing different pattern strings where $\mathbf{P}_i$ is defined as follows:

$$\mathbf{P}_i[j] = \begin{cases} P[j] & i\text{-th bit of } w(j) \text{ is } 1 \\ \star & \text{otherwise.} \end{cases}$$

Let $\mathbf{O}_i$ be the result vector of HamPM between text $\mathbf{T}$ and pattern $\mathbf{P}_i$. The final result vector, $\mathbf{O}$, for the Hamming distance pattern matching with position weights can be computed such that $\mathbf{O}[k] = \sum_{i=0}^{\lceil \log W \rceil} 2^i \cdot \mathbf{O}_i[k]$ where W is the maximum position weight. Given our assumption that $W = \mathrm{poly}(n)$, the result follows. ◀

What remains is to show that one can get rid of $\star$ when e.g. computing Hamming distance. We show this in the pattern matching setting for simplicity. However this can be easily extended for matrix multiplication problems as well.

▶ **Lemma 9.** *Hamming distance in $\mathbb{N} + \{\star\}$ reduces preserving linearity to two instances of Hamming distance in $\mathbb{N}$.*

Proof. Let $x, y \in \mathbb{N} + \{\star\}$. To compute $\mathsf{Ham}(x, y)$, we first use mapping that puts $\star$ into separate integer, and then apply correction that fixes distances between $\star$.

For the first instance:

$$f(t) = \begin{cases} 0 & \text{if } t = \star \\ t + 1 & \text{otherwise} \end{cases}$$

As for the second instance:

$$g(t) = \begin{cases} 0 & \text{if } t = \star \\ 1 & \text{otherwise} \end{cases}$$

Observe that $\mathsf{Ham}(x, y) = \mathsf{Ham}(f(x), f(y)) - \mathsf{Ham}(g(x), g(y))$. ◀

B Supplementary reductions

▶ **Theorem 21.** *L_1 distance reduces to* min *and multiplications.* min *reduces to L_1 and multiplications.*

Proof. $\min(x, y) = x/2 + y/2 - |x - y|/2$ and $|x - y| = x + y - \min(x, y)$. ◀

▶ **Lemma 22.** *Dominance and δ-threshold are equivalent.*

Proof. Since both dominance and threshold are shift-invariant, we assume $0 \leq x, y \leq M$ for some M bounded by $\text{poly}(n)$. Dominance reduces to one instance of threshold as $\mathsf{Dom}(x, y) = \mathsf{Thr}_\delta(x + \delta, y)$ for any $\delta > M$. Threshold reduces to two instances of dominance as $\mathsf{Thr}_\delta(x, y) = \mathsf{Dom}(y + \delta, x) + \mathsf{Dom}(x + \delta, y)$ for $\delta > 0$. ◀

Remark: Thus, the result from [33] is implied by combining Theorem 8 with Lemma 22.

▶ **Lemma 23** ([26])**.** *Hamming distance reduces to 2 instances of dominance.*

Proof. $\mathsf{Ham}(x, y) = \mathsf{Dom}(x + 1, y) + \mathsf{Dom}(-x + 1, -y)$. ◀

▶ **Lemma 24** ([22])**.** *Dominance reduces to 2 instances of L_1, Hamming distance reduces to 3 instances of L_1.*

Proof. $\mathsf{Dom}(x, y) = |x - (y + 1)|/2 - |x - y|/2 + 1/2$ and $\mathsf{Ham}(x, y) = 1 + |x - y| - |x - (y + 1)|/2 - |(x + 1) - y|/2$ ◀

C Example reductions

Figures 2 and 3 illustrate our reductions from Theorems 7 and 8, respectively.

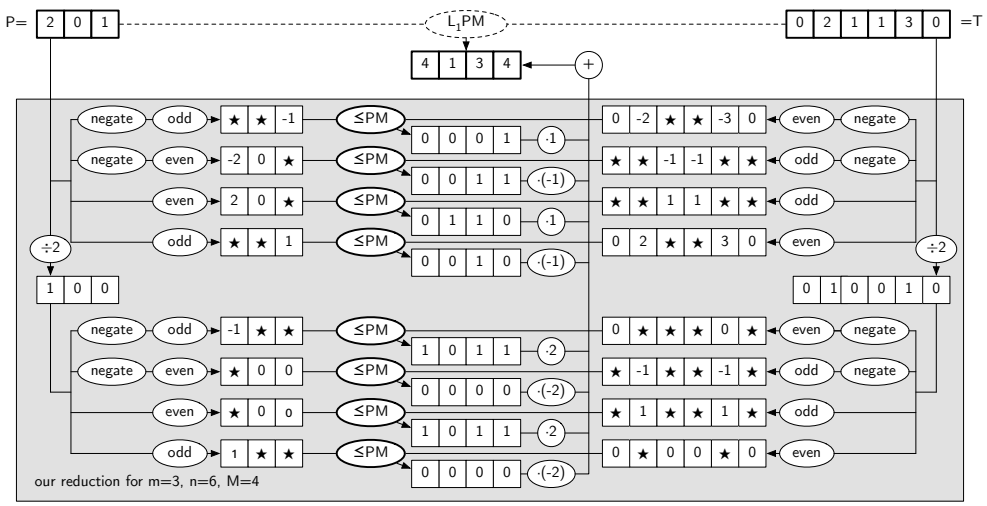

Figure 2 Our reduction from L1PatternMatching to LessThanPatternMatching in Theorem 7 instantiated for a pattern P of length $m = 3$ and a text T of length $n = 6$ over an alphabet of integers $\{0, 1, 2, 3\}$, so $M = 2^2 = 4$.

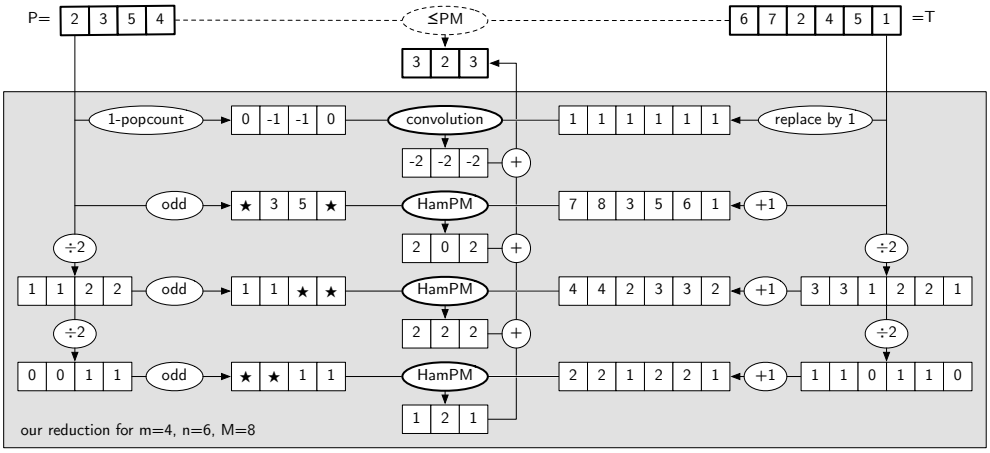

Figure 3 Our reduction from LessThanPatternMatching to HammingDistancePatternMatching in Theorem 8 instantiated for a pattern P of length $m = 4$ and a text T of length $n = 6$ over an alphabet of integers $\{0, 1, 2, 3, 4, 5, 6, 7\}$, so $M = 2^3 = 8$.