

# On Computing Centroids According to the $p$ -Norms of Hamming Distance Vectors

**Jiehua Chen**

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Poland  
jiehua.chen2@gmail.com

**Danny Hermelin**

Department of Industrial Engineering and Management, Ben-Gurion University of the Negev,  
Beer Sheva, Israel  
hermelin@bgu.ac.il

**Manuel Sorge**

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Poland  
manuel.sorge@mimuw.edu.pl

---

## Abstract

In this paper we consider the  $p$ -NORM HAMMING CENTROID problem which asks to determine whether some given strings have a centroid with a bound on the  $p$ -norm of its Hamming distances to the strings. Specifically, given a set  $S$  of strings and a real  $k$ , we consider the problem of determining whether there exists a string  $s^*$  with  $(\sum_{s \in S} d^p(s^*, s))^{1/p} \leq k$ , where  $d(\cdot)$  denotes the Hamming distance metric. This problem has important applications in data clustering and multi-winner committee elections, and is a generalization of the well-known polynomial-time solvable CONSENSUS STRING ( $p = 1$ ) problem, as well as the NP-hard CLOSEST STRING ( $p = \infty$ ) problem.

Our main result shows that the problem is NP-hard for all fixed rational  $p > 1$ , closing the gap for all rational values of  $p$  between 1 and  $\infty$ . Under standard complexity assumptions the reduction also implies that the problem has no  $2^{o(n+m)}$ -time or  $2^{o(k \frac{p}{p+1})}$ -time algorithm, where  $m$  denotes the number of input strings and  $n$  denotes the length of each string, for any fixed  $p > 1$ . The first bound matches a straightforward brute-force algorithm. The second bound is tight in the sense that for each fixed  $\varepsilon > 0$ , we provide a  $2^{k \frac{p}{p+1} + \varepsilon}$ -time algorithm. In the last part of the paper, we complement our hardness result by presenting a fixed-parameter algorithm and a factor-2 approximation algorithm for the problem.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational complexity and cryptography; Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms; Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Facility location and clustering; Theory of computation  $\rightarrow$  Lower bounds and information complexity; Theory of computation  $\rightarrow$  Design and analysis of algorithms; Mathematics of computing  $\rightarrow$  Combinatorial optimization


**Keywords and phrases** Strings, Clustering, Multiwinner Election, Hamming Distance

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2019.28


**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1807.06469>.

**Funding** Research started while both JC and MS were with Ben-Gurion University of the Negev, Beer-Sheva, Israel. This work is supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, the Israel Science Foundation (grant no. 551145/14), and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement numbers 677651 (JC) and 714704 (MS).



 © Jiehua Chen, Danny Hermelin, and Manuel Sorge;  
licensed under Creative Commons License CC-BY  
27th Annual European Symposium on Algorithms (ESA 2019).

Editors: Michael A. Bender, Ola Svensson, and Grzegorz Herman; Article No. 28; pp. 28:1–28:16  
Leibniz International Proceedings in Informatics

 **LIPICs** Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**1 Introduction**

The *Hamming distance between two strings* of equal length is the number of positions at which the corresponding symbols in the strings differ. In other words, it measures the number of substitutions of symbols required to change one string into the other, or the number of errors that could have transformed one string into the other. This is perhaps the most fundamental string metric known, named after Richard Hamming who introduced the concept in 1950 [24].

While Hamming distance has a variety of applications in a plethora of different domains, a common usage for it appears when clustering data of various sorts. Here, one typically wishes to cluster the data into groups that are centered around some centroid, where the notion of centroid varies from application to application. Two prominent examples in this context are: **Consensus String**, where the centroid has a bound on the sum of its (Hamming) distance to all strings, and

**Closest String**, where the centroid has a bound on the maximum distance to all strings.

In functional analysis terms, these two problems can be formalized using the  $p$ -norms of the Hamming distance vectors associated with the clusters. That is, if  $S \subseteq \{0, 1\}^n$  is a cluster and  $s^* \in \{0, 1\}^n$  is its centroid, then the  $p$ -norm of the corresponding Hamming distance vector is defined by

$$\|(s^*, S)\|_p := \left( \sum_{s \in S} d^p(s^*, s) \right)^{1/p},$$

where  $d(s^*, s) = |\{i: s^*[i] \neq s[i], 1 \leq i \leq n\}|$  denotes the Hamming distance between  $s^*$  and  $s$ . Using this notation, we can formulate **CONSENSUS STRING** as the problem of finding a centroid  $s^*$  with a bound on  $\|(s^*, S)\|_1$  for a given set  $S$  of strings, while **CLOSEST STRING** can be formulated as the problem of finding a centroid  $s^*$  with a bound on  $\|(s^*, S)\|_\infty$ .

The following cluster  $S$  with 5 strings, each of length 7, shows that for different  $p$ , we indeed obtain different optimal centroids. For each  $p \in \{1, 2, \infty\}$ , string  $s_p^*$  is an optimal  $p$ -norm centroid but it is not an optimal  $q$ -norm centroid, where  $q \in \{1, 2, \infty\} \setminus \{p\}$ . Moreover, one can verify that  $s_2^*$  is the only optimal 2-norm centroid and no optimal  $\infty$ -norm centroid is an optimal 2-norm centroid.

$S :$		$p$		
	centroid	$\ \cdot\ _1$	$\ \cdot\ _2$	$\ \cdot\ _\infty$
1111 111	$s_1^* = 0000\ 000$	14	$\sqrt{68}$	7
1111 000	$s_2^* = 0011\ 000$	16	$\sqrt{56}$	5
0000 100	$s_\infty^* = 0011\ 001$	17	$\sqrt{61}$	4
0000 010				
0000 001				

The notion of  $p$ -norms for distance vectors is very common in many different research fields [33, 30, 21, 34, 20, 2, 27, 3, 17, 39]. In cluster analysis of data mining and machine learning, one main goal is to partition  $m$  observations (i.e.,  $m$  real vectors of the same dimension) into  $K$  groups so that the sum of “discrepancies” between each observation and its nearest center is minimized. Here, two highly prominent clustering methods are  $K$ -means [32] and  $K$ -medians [25, 4] clustering, each using a slightly different notion of discrepancy measure. The first method aims to minimize the *sum of squared Euclidean distances* between each observation and the “mean” of its respective group. In other words, it minimizes the squared 2-norm of the Euclidean-distance vector.  $K$ -medians, on the other hand, uses the 1-norm instead of the squared 2-norm to define the discrepancy to the mean. Thus, instead of calculating the mean for each group to determine its centroid, one calculates the median.

In committee elections from social choice theory [17, 39, 35, 16], the  $p$ -norm is used to analyze how well a possible committee represents the voter's choices. In a fundamental approval-based procedure to select a  $t$ -person committee from  $n$  candidates, each voter either approves or disapproves each of the candidates, which can be expressed as a binary string of length  $n$ . An optimal committee is a length- $n$  binary string containing exactly  $t$  ones and which minimizes the  $p$ -norm of the vector of the Hamming distances to each voter's preference string [39].

### Problem Definition, Notation, and Conventions

Since the Hamming distance is frequently used in various applications, e.g., in computational biology [36], information theory, coding theory and cryptography [24, 11, 37], in social choice [26, 1] and since the notion of  $p$ -norm is very prominent in clustering tools [38, 6, 30, 40] and preference aggregation rules [1, 5, 35], where often  $p = 1, 2, \infty$  but also other values of  $p$  are used, it is natural to consider computational problems associated with the  $p$ -norm of the Hamming distance metric. This is the main purpose of this paper. Specifically, we consider the following problem:

$p$ -NORM HAMMING CENTROID ( $p$ -HDC)

**Input:** A set  $S$  of strings  $s_1, \dots, s_m \in \{0, 1\}^n$  and a real  $k$ .

**Question:** Is there a string  $s^* \in \{0, 1\}^n$  such that  $\|(s^*, S)\|_p \leq k$ ?

Throughout, we will call a string  $s^*$  as above a *solution*. Note that there is nothing special about using the binary alphabet in the definition above, but for ease of presentation we use it throughout the paper. When  $p = 1$ , our  $p$ -HDC problem is precisely the CONSENSUS STRING problem, and when  $p = \infty$  it becomes the CLOSEST STRING problem.

In the following, we list some notation and conventions that we use. By  $p$ -distance we mean the  $p^{\text{th}}$ -power of the Hamming distance. For each natural number  $t$  by  $[t]$  we denote the set  $\{1, 2, \dots, t\}$ . Unless stated otherwise, by *strings* we mean binary strings over alphabet  $\{0, 1\}$ . Given a string  $s$ , we use  $|s|$  to denote the length of this string. For two binary strings  $s$  and  $s'$ , let  $s \circ s'$  denote the concatenation of  $s$  and  $s'$ . By  $s[j]$  we denote the  $j$ th value or the value in the  $j^{\text{th}}$  character of string  $s$ . By  $\bar{s} = (1 - s[j])_{j \in [|s|]}$  we denote the complement of the (binary) string  $s$ . Given two integers  $j, j' \in \{1, 2, \dots, |s|\}$  with  $j \leq j'$ , we write  $s|_j^{j'}$  for the substring  $s[j]s[j+1] \cdots s[j']$ . Given a number  $\ell$ , we use  $\mathbf{0}_\ell$  and  $\mathbf{1}_\ell$  to denote the length- $\ell$  all-zero string and the length- $\ell$  all-one string, respectively.

### Our Contributions

Our main result is a tight running time bound on the  $p$ -HDC problem for all fixed rationals  $p > 1$ . Specifically, we show that the problem is NP-hard and can be solved in  $2^{k^{p/(p+1)+\varepsilon}} \cdot |I|^{O(1)}$  time for arbitrary small  $\varepsilon > 0$  where  $|I|$  denotes the size of the instance, but cannot be solved in  $2^{o(k^{p/(p+1)})}$  time unless the Exponential Time Hypothesis (ETH) [12] fails. The lower bounds are given in Theorem 3 and Proposition 6 and the upper bound in Theorem 10. While the upper bound in this result is not very difficult, the lower bound uses an intricate construction and some delicate arguments to prove its correctness. In particular, the construction extensively utilizes the fact that since  $p > 1$ , the  $p$ -norm of Hamming distances is convex and always admits a second derivative. We believe that this kind of technique is of interest on its own. As another consequence of the hardness construction, we also obtain a  $2^{o(n+m)}$  running time lower bound assuming ETH, which gives evidence that the trivial brute-force  $2^n \cdot |I|$ -time algorithm for the problem cannot be substantially

improved. Moreover, the lower bounds also hold when we constrain the solution string to have a prescribed number of ones. That is, we also show hardness for the committee election problem mentioned above (Corollary 7).

In the final part of the paper we present two more algorithms for  $p$ -HDC. First, we provide an  $m^{O(m^2)} \cdot |I|^{O(1)}$  time algorithm (see Theorem 13), by first formulating the problem as a so-called Combinatorial  $n$ -fold Integer Program, and then applying the algorithm developed by Knop et al. [28]. Second, we show that the problem can be approximated in polynomial time within a factor of 2, using an extension of the well known 2-approximation algorithm for CLOSEST STRING (see Proposition 14).

## Related Work

The NP-complete CLOSEST STRING [18, 29] problem (aka. MINIMUM RADIUS) is a special case of  $p$ -HDC with  $p = \infty$ . It seems, however, difficult to adapt this hardness reduction to achieve our hardness results for every fixed rational  $p$  (see also the beginning of Section 2 for some more discussion). CLOSEST STRING has been studied extensively under the lens of parameterized complexity and approximation algorithmics. The first fixed-parameter algorithm for parameter  $k$ , the maximum Hamming distance bound, was given by Gram et al. [22], runs in  $O(k^k \cdot km + mn)$  time where  $m$  and  $n$  denote the number and the length of input strings, respectively. This algorithm works for arbitrary alphabet  $\Sigma$ . For small alphabets  $\Sigma$ , there are algorithms with  $O(mn + n \cdot |\Sigma|^{O(k)})$  running time [31, 9]. Both types of running time are tight under the ETH [12, Theorem 14.17]. For arbitrary alphabet  $\Sigma$ , Knop et al. [28] gave an algorithm with  $m^{O(m^2)} \cdot \log n$  running time based on so-called  $n$ -fold integer programming. As for approximability, CLOSEST STRING admits a PTAS with running time  $O(n^{O(\epsilon^{-2})})$  [31] but no EPTAS unless  $\text{FPT} = \text{W}[1]$  [13].

Our problem falls into the general framework of convex optimization with binary variables. If the input and the output are allowed to have fractional values, then the underlying convex optimization problem, called  $L_p$ -NORM CONVEX MINIMIZATION, can be solved in polynomial time for each fixed value  $p \leq 2$  [34, Chapter 6.3.2]. This convex optimization problem is a special variant of the in general NP-hard  $L_p$  SUBSPACE APPROXIMATION problem [14, 23]. This problem has as input  $m$  points  $s_1, \dots, s_m$  in  $\mathbb{R}^n$  and an integer  $k'$ , and asks to find a subspace  $H$  of  $\mathbb{R}^n$  of dimension  $k'$  that minimizes the following  $\sum_{i=1}^m (\text{dist}^p(H, a_i))^{1/p}$ , where  $\text{dist}(H, a_i)$  is the minimum Euclidean distance between  $a_i$  and any point in  $H$ . For  $k = 0$ ,  $L_p$  SUBSPACE APPROXIMATION is equivalent to the  $L_p$ -NORM CONVEX MINIMIZATION.

For  $p \in \{2, 3\}$ , maximizing (instead of minimizing) the  $p$ -norm reduces to MIRKIN MINIMIZATION in consensus clustering with input and output restricted to two-clusters, which was shown to be NP-hard [15] under Turing reductions. Recently, Chen et al. [7] showed that the simple  $2^n$ -time algorithm by brute-force searching all possible outcome solutions is essentially tight under ETH. They also provided some efficient algorithms and showed that the problem admits an FPTAS using a simple rounding technique.

## 2 NP-hardness for the $p$ -Norm of Hamming Distance Vectors

We now show that  $p$ -HDC is NP-hard for each fixed rational number  $p > 1$  (Theorem 3 and Proposition 6) and that algorithms with running time  $2^{o(n+m)}$  or  $2^{o(k^{p/(p+1)})}$  would contradict the ETH. We reduce from the NP-hard 3-COLORING problem [19] in which, given an undirected graph  $G = (V, E)$ , we ask whether there is a *proper vertex coloring*  $\text{col}: V \rightarrow \{0, 1, 2\}$ , that is, no two *adjacent* vertices receive the same color.

The first challenge we need to overcome when designing the reduction is to produce some regularity in the solution string: Given  $\hat{n} \in \mathbb{N}$ , in Lemma 1, we show how to construct a set of strings to enforce a solution string to have exactly  $\hat{n}$  ones which only occur in the columns of some specific range. This allows us later on to build gadgets that have several discrete states. Indeed, after controlling the overall number of ones in the solution in this way, we can allocate three columns (one for each color) for each vertex  $v$  in  $G$  and build a gadget for  $v$  such that this gadget induces minimum  $p$ -distance to the solution if and only if there is exactly 1 one in the solution in the columns allocated for  $v$ . This column determines the color for  $v$ . Then, for each edge, we will introduce an edge gadget consisting of six strings which induce minimum  $p$ -distance in the solution if and only if they are “covered” by the ones in the solution exactly twice, corresponding to different colors.

In general, the design of gadgets for  $p$ -HDC is quite different from the known NP-hard case CLOSEST STRING ( $p = \infty$ ) [18, 29]: In CLOSEST STRING every optimal solution  $s^*$  must regard the “worst” possible input string while in our case  $s^*$  can escape such constraints by distributing some of its Hamming distance from the “worst” to other strings.

In the remainder of this section, let  $a$  and  $b$  be two fixed integers such that  $a$  and  $b$  are coprime,  $a > b$ , and  $p = a/b > 1$ . To better capture the Hamming distance, we introduce the notion of the *Hamming set* of two strings  $s$  and  $s'$  of equal length  $n$ , which consists of the indices of the columns at which both strings differ:  $\text{hs}(s, s') = \{j \in [n] \mid s[j] \neq s'[j]\}$ .

As mentioned, we first show how to construct a set of strings to enforce some structure on the optimal solution, that is, a binary string with minimum sum of the  $p$ -distances.

► **Lemma 1** ( $\star^1$ ). *Let  $p > 1$  be a fixed rational number, and  $a$  and  $b$  be two coprime fixed integers with  $p = a/b$ . Let  $S$  consist of one string  $\mathbf{1}_{(2^b+1)\hat{n}}\mathbf{0}_{\hat{n}}$  and  $2^{a-b}$  copies of string  $\mathbf{0}_{(2^b+2)\hat{n}}$ , where  $\hat{n}$  is a positive integer. For each string  $s^* \in \{0, 1\}^{(2^b+2)\hat{n}}$ , the following holds.*

- (1) *If  $d(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) = \hat{n}$  and  $\text{hs}(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \subseteq [(2^b+1)\hat{n}]$ , then  $\|(s^*, S)\|_p^p = (2^a + 2^{a-b}) \cdot \hat{n}^p$ .*
- (2) *If  $d(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \neq \hat{n}$  or  $\text{hs}(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \not\subseteq [(2^b+1)\hat{n}]$ , then  $\|(s^*, S)\|_p^p > (2^a + 2^{a-b}) \cdot \hat{n}^p$ .*

To show Lemma 1 we crucially use the fact that  $p > 1$ . In contrast, if  $p = 1$ , then taking the majority value in each column yields an optimal solution, and thus it is impossible to force every optimal solution to have a certain number of ones without at the same time specifying in which precise columns these ones should occur.

In the reduction we make heavy use of specific pairs of strings whose Hamming distances to an arbitrary string always sum up to some lower bound. They will enforce local structure in some columns of the solution, while being somewhat immune to changes elsewhere. As a tool in the reduction we derive the following lower bound on the  $p$ -distance of an arbitrary string to a pair of strings which are quite far from each other, in terms of Hamming distances.

► **Lemma 2** ( $\star$ ). *Let  $s_1$  and  $s_2$  be two strings of the same length  $R$  such that the Hamming distance between  $s_1$  and  $s_2$  is  $d(s_1, s_2) = 2L$ . For each rational  $p > 1$  and each length- $R$  string  $\hat{s}$  the following holds.*

- (1)  $d^p(\hat{s}, s_1) + d^p(\hat{s}, s_2) \geq 2 \cdot L^p$ .
- (2) *If  $d(\hat{s}, s_1) = d(\hat{s}, s_2) = L$ , then  $d^p(\hat{s}, s_1) + d^p(\hat{s}, s_2) = 2 \cdot L^p$ .*
- (3) *If  $d(\hat{s}, s_1) \neq L$  or  $d(\hat{s}, s_2) \neq L$ , then  $d^p(\hat{s}, s_1) + d^p(\hat{s}, s_2) > 2 \cdot L^p$ .*

Using Lemmas 1 and 2, we can show NP-hardness of  $p$ -HDC for each fixed rational  $p > 1$ . For better readability, we will first show hardness for the case with multiple identical strings (Theorem 3) and then extend the construction to also include the case where no two strings are the same (Proposition 6).

<sup>1</sup> Proofs for results marked by  $\star$  can be found in [8].

► **Theorem 3.** For each fixed rational number  $p > 1$ ,  $p$ -HDC (with possibly multiple identical strings) is NP-hard.

**Proof.** First of all, let  $a$  and  $b$  be two fixed coprime integers such that  $p = a/b$ . To show the hardness result, we reduce from the NP-hard 3-COLORING problem [19] defined above. Let  $G = (V, E)$  be an instance of 3-COLORING. Let  $n$  be the number of vertices in  $G$  and  $m$  the number of edges. Denote  $V = \{v_1, v_2, \dots, v_n\}$  and  $E = \{e_1, e_2, \dots, e_m\}$ .

*Construction.* We introduce three groups of strings of length  $(2^b + 2) \cdot \hat{n}$  each, where  $\hat{n} = n + m$ . The first group ensures that each optimal solution string must have exactly  $\hat{n}$  ones which appear in the first  $3\hat{n}$  columns (using Lemma 1), the second group ensures that an optimal solution enforces that each vertex has exactly one of the three colors, and the third group, combined with the second group, ensures that no two adjacent vertices obtain the same color.

**Group 1.** Construct one string  $\mathbf{1}_{(2^b+1)\hat{n}} \circ \mathbf{0}_{\hat{n}}$  and  $2^{a-b}$  copies of the same string  $\mathbf{0}_{(2^b+2)\hat{n}}$ .

**Group 2.** This group consists of one pair of strings for each vertex. Each pair consists of two strings which are mostly complements to each other. This ensures that the Hamming distance to the solution induced by a pair is somewhat homogeneous, regardless where exactly the ones in the solution occur. However, in each pair there are three columns, corresponding to the vertex, which will skew the pairs of Hamming distances in a way to induce minimum  $p$ -distances only if the solution has exactly 1 one in these three columns. Formally, for each vertex  $v_i \in V$ , let  $u_i$  be a string of length  $3\hat{n}$  which has exactly 3 ones in the columns  $3i - 2, 3i - 1, 3i$ , and let  $\bar{u}_i$  be the complement of  $u_i$ . Deriving from  $u_i$ , we construct two *vertex strings*  $s_i$  and  $r_i$  with  $s_i := u_i \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{0} \circ \mathbf{1}_{\hat{n}-1}$  and  $r_i := \bar{u}_i \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{1} \circ \mathbf{0}_{\hat{n}-1}$ . Note that both strings  $s_i$  and  $r_i$  have all zeros in the columns  $\{3\hat{n}, \dots, (2^b + 1)\hat{n}\}$  such that  $d(s_i, r_i) = 4\hat{n}$ .

For an illustration, the strings  $s_2$  and  $r_2$ , which correspond to the vertex  $v_2$ , are as follows:  
 $s_2 = \mathbf{000111} \circ \mathbf{0}_{3\hat{n}-6} \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{0} \circ \mathbf{1}_{\hat{n}-1}$ ,  $r_2 = \mathbf{111000} \circ \mathbf{1}_{3\hat{n}-6} \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{1} \circ \mathbf{0}_{\hat{n}-1}$ .

**Group 3.** We now use three pairs of strings for each edge to ensure relatively homogeneous distributions of Hamming distances to the solution and then skew them. This time, we aim to skew distances to the solution so that their corresponding  $p$ -distances are minimum only if the solution distributes exactly three ones (corresponding to the colors) over three special regions: two corresponding to the endpoints of the edge and one extra dummy region.

Formally, for each edge  $e_j \in E$  let  $e_j^{(0)}, e_j^{(1)}$ , and  $e_j^{(2)}$  denote three strings, each of length  $3\hat{n}$ , that ensure that the edge and both of its endpoints each have a distinct color:

$$\forall \ell \in \{1, 2, \dots, \hat{n}\}: e_j^{(0)}[3\ell - 2, 3\ell - 1, 3\ell] := \begin{cases} 100, & 1 \leq \ell \leq n \text{ with } v_\ell \in e_j, \text{ or } \ell = j + n, \\ 000, & \text{otherwise.} \end{cases}$$

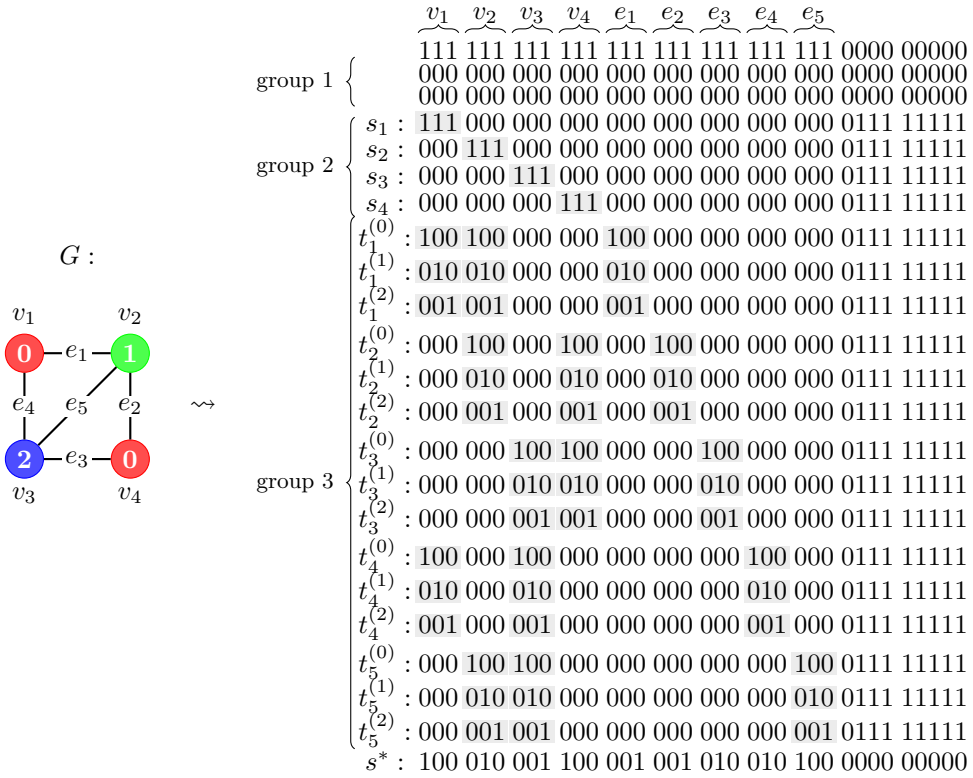
$$e_j^{(1)}[3\ell - 2, 3\ell - 1, 3\ell] := \begin{cases} 010, & 1 \leq \ell \leq n \text{ with } v_\ell \in e_j, \text{ or } \ell = j + n, \\ 000, & \text{otherwise.} \end{cases}$$

$$e_j^{(2)}[3\ell - 2, 3\ell - 1, 3\ell] := \begin{cases} 001, & 1 \leq \ell \leq n \text{ with } v_\ell \in e_j, \text{ or } \ell = j + n, \\ 000, & \text{otherwise.} \end{cases}$$

Now, we construct the following six *edge strings* for edge  $e_j$ :

$$\forall z \in \{0, 1, 2\}: t_j^{(z)} := e_j^{(z)} \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{0} \circ \mathbf{1}_{\hat{n}-1} \text{ and } w_j^{(z)} := \bar{e}_j^{(z)} \circ \mathbf{0}_{(2^b-2)\hat{n}} \circ \mathbf{1} \circ \mathbf{0}_{\hat{n}-1}.$$

Just as for group 2, the two strings  $t_j^{(z)}$  and  $w_j^{(z)}$  have all zeros in the columns  $\{3\hat{n}, \dots, (2^b + 1)\hat{n}\}$  such that  $d(t_j^{(z)}, w_j^{(z)}) = 4\hat{n}$ .



**Figure 1** Illustration of the reduction used in Theorem 3. The left figure depicts a graph  $G$  that admits a proper vertex coloring  $\text{col}$  (see the labels on the vertices). For instance, vertex  $v_1$  has color 0, i.e.,  $\text{col}(v_1) = 0$ . The right figure shows the crucial part of an instance of  $p$ -HDC with  $p = 2$  (i.e.,  $a = 2$  and  $b = 1$ ) that we will construct according to the proof for Theorem 3. For each pair of constructed strings we only show the first one. A solution string  $s^*$  corresponding to the coloring  $\text{col}$  is depicted at the bottom of the right figure.

For an example, assume that  $a = 3$ ,  $b = 2$ ,  $n = 3$ , and  $m = 2$ , and there is an edge of the form  $e_2 = \{v_1, v_3\}$ . Then, the two triples of strings that we construct for  $e_2$  have each length  $(2^b + 2)(n + m) = 30$  and are

$$\begin{aligned}
 t_j^{(0)} &= 100\ 000\ 100\ 000\ 100\ 0000000000\ 01111, & w_j^{(0)} &= 011\ 111\ 011\ 111\ 011\ 0000000000\ 10000, \\
 t_j^{(1)} &= 010\ 000\ 010\ 000\ 010\ 0000000000\ 01111, & w_j^{(1)} &= 101\ 111\ 101\ 111\ 101\ 0000000000\ 10000, \\
 t_j^{(2)} &= 001\ 000\ 001\ 000\ 001\ 0000000000\ 01111, & w_j^{(2)} &= 110\ 111\ 110\ 111\ 110\ 0000000000\ 10000.
 \end{aligned}$$

Summarizing, the instance  $I'$  of  $p$ -HDC consists of the following strings, each of length  $(2^b + 2)\hat{n} = (2^b + 2)(n + m)$ :

- (1) Add the  $2^{a-b} + 1$  strings in group 1 to  $I'$ .
- (2) For each vertex  $v_i \in V$ , add the vertex strings  $s_i$  and  $r_i$  to  $I'$ .
- (3) For each edge  $e_j \in E$ , add two triples  $t_j^{(0)}, t_j^{(1)}, t_j^{(2)}$  and  $w_j^{(0)}, w_j^{(1)}, w_j^{(2)}$  to  $I'$ .

See Figure 1 for an illustration.

Finally, we define  $k$  such that  $k^p = (2^a + 2^{a-b}) \cdot \hat{n}^p + 2(n + 3m) \cdot (2\hat{n})^p$ . This completes the construction, which can clearly be done in polynomial time.

*Correctness of the construction.* Before we show the correctness of our construction, we define a notion and make an observation. Let  $s$  and  $s'$  be two strings of equal length. We say that  $s$  covers  $s'$  exactly once if there is exactly one integer  $\ell \in \{1, 2, \dots, |s|\}$  with  $s[\ell] = s'[\ell] = 1$ .

▷ Claim 4 (★). Let  $s^*$  and  $s$  be two strings, both of length  $4\hat{n}$ , such that

(i)  $s^*$  has exactly  $\hat{n}$  ones and each of them is in the first  $3\hat{n}$  columns, and

(ii) in  $s$ , the first  $3\hat{n}$  columns have exactly 3 ones and the last  $\hat{n}$  columns are  $0 \circ \mathbf{1}_{\hat{n}-1}$ .

Then, if  $s^*$  covers  $s$  exactly once, then  $d^p(s^*, s) + d^p(s^*, \bar{s}) = 2 \cdot (2\hat{n})^p$ ; else  $d^p(s^*, s) + d^p(s^*, \bar{s}) > 2 \cdot (2\hat{n})^p$ .

We show that  $G$  has a proper 3-coloring if and only if there is a string  $s^*$  such that the sum of the  $p$ -distances from  $s^*$  to all strings in  $I'$  is at most  $k^p = (2^a + 2^{a-b}) \cdot \hat{n}^p + 2(n+3m) \cdot (2\hat{n})^p$ .

For the “if” direction, let  $s^*$  be a string which has a sum of  $p$ -distances of at most  $k^p$  to all strings in  $I'$ . Before we define a coloring for the vertices and show that it is proper we observe several properties of the solution string  $s^*$ .

By Lemma 2(1), the sum of  $p$ -distances to all strings from group 2 and group 3 is at least  $2 \cdot (2\hat{n})^p \cdot (n+3m)$  since these groups consist of  $n+3m$  pairs of strings, and the strings in each of these pairs have Hamming distance exactly  $4\hat{n}$  to each other. By the definition of  $k$ , the sum of  $p$ -distances from  $s^*$  to the first of group of strings is thus at most  $(2^a + 2^{a-b}) \cdot \hat{n}^p$ . Hence, by the contra-positive of Lemma 1(2), the solution string  $s^*$  has exactly  $\hat{n}$  ones, which all appear in the first  $(2^b + 1)\hat{n}$  columns, i.e.,  $d(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) = \hat{n}$  and  $\text{hs}(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \subseteq [(2^b + 1)\hat{n}]$ . By Lemma 1(1), this implies that

$$\sum_{s \in \text{group 1}} d^p(s^*, s) = (2^a + 2^{a-b}) \cdot \hat{n}^p. \quad (1)$$

Next, we claim that the ones in the solution  $s^*$  indeed all appear in the first  $3\hat{n}$  columns, i.e.,  $\text{hs}(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \subseteq [3\hat{n}]$ . Suppose, for the sake of contradiction, that solution  $s^*$  contains  $x$  ones which appear in columns ranging from  $3\hat{n} + 1$  to  $(2^b + 1)\hat{n}$  with  $x > 0$ . Consider an arbitrary pair of strings  $s_i$  and  $r_i$  from group 2 or an arbitrary pair of strings  $t_i^{(z)}$  and  $w_i^{(z)}$  from group 3; for the sake of readability, represent them by  $s$  and  $s'$ . By construction, strings  $s$  and  $s'$  have Hamming distance exactly  $4\hat{n}$  to each other, but have all zeros in the columns between  $3\hat{n} + 1$  and  $(2^b + 1)\hat{n}$ . Since  $x > 0$ , by the triangle inequality of Hamming distances, it follows that at least one string from the pair,  $s$  or  $s'$ , has Hamming distance more than  $2\hat{n}$  from  $s^*$ . However, by Lemma 2(3), this means that the sum of  $p$ -distances from  $s^*$  to  $\{s, s'\}$  exceeds  $2 \cdot (2\hat{n})^p$ . Since there are in total  $n + 3m$  such pairs in groups 2 and 3, the sum of  $p$ -distances from  $s^*$  to these groups exceeds  $2(n + 3m) \cdot (2\hat{n})^p$ , a contradiction to equation (1) and the defined bound  $k$ . Thus, indeed, it holds that

$$d(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) = \hat{n} \text{ and } \text{hs}(s^*, \mathbf{0}_{(2^b+2)\hat{n}}) \subseteq [3\hat{n}]. \quad (2)$$

This implies that, when determining the  $p$ -distance of  $s^*$  to the strings from group 2 and group 3, we can ignore, the values in the columns ranging from  $3\hat{n} + 1$  to  $(2^b + 1)\hat{n}$ , in each string, which includes the solution  $s^*$ , because  $s^*$  also has only zeros in these columns. We will hence from now on treat these columns as if they do not exist. In this way, we obtain strings of length  $4\hat{n}$ . Again, consider an arbitrary pair of strings  $s_i$  and  $r_i$  from group 2 (resp. an arbitrary pair of strings  $t_i^{(z)}$  and  $w_i^{(z)}$  from group 3), and represent them by  $s$  and  $s'$ . Since we ignore columns  $3\hat{n} + 1$  to  $(2^b + 1)\hat{n}$ , string  $s'$  is the complement of  $s$ . By construction, the Hamming distance between  $s$  and  $s'$  is exactly  $4\hat{n}$ . Using Claim 4 on  $s^*$ ,  $s$ ,  $s'$ , the sum of  $p$ -distances from  $s^*$  to the pair  $\{s, s'\}$  is indeed equal to  $2 \cdot (2\hat{n})^p$ . By the same claim, it follows that  $s^*$  covers each string  $s_i$  (resp.  $t_j^{(z)}$ ) from group 2 (resp. group 3) exactly once.

Having this property, we are ready to color the vertices. Let  $\text{col}: V \rightarrow \{0, 1, 2\}$  be a mapping defined as follows. For each  $v_i \in V$ , set  $\text{col}(v_i) = z$  where  $z \in \{0, 1, 2\}$  such that  $s^*[3i - 2 + z] = 1$ . Note that, since  $s^*$  covers  $s_i$  exactly once and since  $s_i$  has exactly three



ones in the columns  $3i - 2$ ,  $3i - 1$ , and  $3i$ , there is indeed such a  $z$  with  $\text{col}(v_i)$ . We claim that  $\text{col}$  is a proper coloring for  $G$ . Suppose, towards a contradiction, that there is an edge  $e_j = \{v_i, v_{i'}\} \in E$  such that  $v_i$  and  $v_{i'}$  have the same color from  $\text{col}$ , say  $z \in \{0, 1, 2\}$ . By the definition of  $\text{col}$ , this means that  $s^*[3i - 2 + z] = s^*[3i' - 2 + z] = 1$ . However, by the definition of the string  $t_j^{(z)}$  which corresponds to the edge  $e_j$ , we also have that  $t_j^{(z)}[3i - 2 + z] = t_j^{(z)}[3i' - 2 + z] = 1$ . This implies that  $t_j^{(z)}$  is not covered by  $s^*$  exactly once – a contradiction to our reasoning above that  $s^*$  covers each string from the third group exactly once.

For the “only if” direction, let  $\text{col}: V \rightarrow \{0, 1, 2\}$  be a proper coloring for  $G$ . For an edge  $e \in E$  with two endpoints  $v_i, v_{i'}$ , let  $\text{col}(e) = \{\text{col}(v_i), \text{col}(v_{i'})\}$ . We claim that string  $s^*$ , defined as follows, has the desired bound on the sum of the  $p$ -distances to all strings of  $I'$ .

$$\forall i \in \{1, 2, \dots, n\}: s^*[3i - 2, 3i - 1, 3i] := \begin{cases} 100, & \text{col}(v_i) = 0, \\ 010, & \text{col}(v_i) = 1, \\ 001, & \text{col}(v_i) = 2. \end{cases}$$

$$\forall j \in \{n + 1, n + 2, \dots, \hat{n}\}: s^*[3j - 2, 3j - 1, 3j] := \begin{cases} 100, & \text{col}(e_j) = \{1, 2\}, \\ 010, & \text{col}(e_j) = \{0, 2\}, \\ 001, & \text{col}(e_j) = \{0, 1\}. \end{cases}$$

$$s^* \Big|_{3\hat{n}+1}^{(2^b+2)\hat{n}} := \mathbf{0}_{\hat{n}}.$$

First of all, since  $\text{col}$  is a proper coloring,  $s^*$  is well defined in all  $(2^b + 2)\hat{n}$  columns. Moreover, it has exactly  $n$  ones in the first  $3n$  columns and exactly  $m$  ones in the next  $3m$  columns, and all zeros in the remaining columns. Thus, by Lemma 1(2), the sum of the  $p$ -distances from  $s^*$  to the first group of strings is  $(2^a + 2^{a-b}) \cdot \hat{n}^p$ .

Now, we focus on strings from group 2 and group 3. Since the solution  $s^*$  and each string in these groups have only zeros in the columns between  $3\hat{n} + 1$  and  $(2^b + 1)\hat{n}$ , we can simply ignore the values in these columns and assume from now on that the strings have length  $4\hat{n}$ . Moreover, for each  $i \in [n]$ , the pair  $s_i$  and  $r_i$  can be considered as complement to each other. Thus, for each string  $s_i$  from group 2,  $s^*$  and  $s_i$  fulfill the properties stated in Claim 4. Moreover, by definition,  $s^*$  covers  $s_i$  exactly once. Thus, by the same claim, we have that the sum of the  $p$ -distances from  $s^*$  to all strings in group 2 is  $n \cdot 2 \cdot (2\hat{n})^p$ .

Analogously, consider a string  $t_j^{(z)}$  from group 3,  $j \in \{1, 2, \dots, m\}$  and  $z \in \{0, 1, 2\}$ . Recall that  $t_j^{(z)}$  corresponds to the edge  $e_j$ , and let  $v_i$  and  $v_{i'}$  be the two endpoints of edge  $e_j$ . We claim that  $s^*$  covers  $t_j^{(z)}$  exactly once. Observe that  $t_j^{(z)}$  has exactly 3 ones in the first  $3\hat{n}$  columns, namely at columns  $3i - 2 + z$ ,  $3i' - 2 + z$ , and  $3n + 3j - 2 + z$ . To prove that  $s^*$  covers  $t_j^{(z)}$  exactly once, it suffices to show that  $s^*$  has 1 one in exactly one of these three columns. To show this, we consider the substrings  $t_j^{(z)} \Big|_{3n+3j-2}^{3n+3j}$  and  $s^* \Big|_{3n+3j-2}^{3n+3j}$ .

**Case 1:**  $s^* \Big|_{3n+3j-2}^{3n+3j} = t_j^{(z)} \Big|_{3n+3j-2}^{3n+3j}$ . By the definition of  $s^*$ , this implies that  $s^*[3n + 3j - 2 + z] = 1$  and  $\text{col}(e_j) = \{0, 1, 2\} \setminus \{z\}$ . We claim that  $s^*[3i - 2 + z] = s^*[3i' - 2 + z] = 0$ . By the definition of  $s^*$  regarding the columns that correspond to the endpoint  $v_i$  of edge  $e_j$ , we have that  $s^*[3i - 2 + \text{col}(v_i)] = 1$  while  $s^*[3i - 2 + z] = 0$  (since  $z \notin \text{col}(e_j) = \{\text{col}(v_i), \text{col}(v_{i'})\}$ ). Analogously, by the definition of  $s^*$  regarding the columns that correspond to the other endpoint  $v_{i'}$  of edge  $e_j$ , we have that  $s^*[3i' - 2 + \text{col}(v_{i'})] = 1$  while  $s^*[3i' - 2 + z] = 0$  (since  $z \notin \text{col}(e_j) = \{\text{col}(v_i), \text{col}(v_{i'})\}$ ). Thus,  $3n + 3j - z$  is the only column in which both  $s^*$  and  $t_j^{(z)}$  have 1 one, implying that  $s^*$  covers  $t_j^{(z)}$  exactly once.

**Case 2:**  $s^*|_{3n+3j-2}^{3n+3j} \neq t_j^{(z)}|_{3n+3j-2}^{3n+3j}$ . This means that  $s^*[3n+3j-2+z] = 0$  and that  $z \in \text{col}(e_j)$ . To show that  $s^*$  covers  $t_j^{(z)}$  exactly once in this case, it suffices to show that either  $s^*[3i-2+z] = 1$  and  $s^*[3i'-2+z] = 0$ , or  $s^*[3i-2+z] = 0$  and  $s^*[3i'-2+z] = 1$ .

- Assume that  $s^*[3i-2+z] = 1$ . Then, by the definition of  $s^*$  regarding the columns that correspond to the endpoint  $v_i$  of edge  $e_j$ , this means that  $\text{col}(v_i) = z$ . Since  $\text{col}$  is a proper coloring, it follows that  $\text{col}(v_{i'}) \neq z$ . Thus, again by the definition of  $s^*$  regarding the columns that correspond to the other endpoint  $v_{i'}$  of edge  $e_j$ , it follows that  $s^*[3i'-2+z] = 0$ .
- Assume that  $s^*[3i-2+z] = 0$ . Then, by the definition of  $s^*$  regarding the columns that correspond to the endpoint  $v_i$  of edge  $e_j$ , we have  $\text{col}(v_i) \neq z$ . Since  $z \in \text{col}(e_j)$  and  $\text{col}$  is a proper coloring, the other endpoint  $v_{i'}$  of edge  $e_j$  must have color  $\text{col}(v_{i'}) = z$ . Again, by the definition of  $s^*$  regarding the columns that correspond  $v_{i'}$ , it follows that  $s^*[3i'-2+z] = 1$ .

We have just shown that  $s^*$  covers  $t_j^{(z)}$  exactly once. Since  $s^*$  and  $t_j^{(z)}$  fulfill the property stated in Claim 4, it follows from the same claim that the sum of  $p$ -distances from  $s^*$  to  $t_j^{(z)}$  and to  $w_j^{(z)}$  is  $2 \cdot (2\hat{n})^p$ . There are  $3m$  pairs in this group. So, the sum of the  $p$ -distances from  $s^*$  to all strings of this group is  $3m \cdot 2 \cdot (2\hat{n})^p$ .

In total, the sum of the  $p$ -distances from  $s^*$  to all strings of  $I'$  is  $(2^a + 2^{a-b}) \cdot \hat{n}^p + 2 \cdot (2\hat{n})^p \cdot (n + 3m) = k^p$ , as required. ◀

Our NP-hardness reduction implies the following running time lower bounds [12].

► **Corollary 5** (★). *For each fixed rational number  $p > 1$ , unless the ETH fails, no  $2^{o(\hat{n}+\hat{m})} \cdot |I'|^{O(1)}$ -time or  $2^{o(k^{p/(p+1)})} \cdot |I'|^{O(1)}$ -time algorithm exists that decides every given instance  $I'$  of  $p$ -HDC where  $\hat{n}$  is the length of the input strings,  $\hat{m}$  is the number of input strings, and  $k$  is the  $p$ -norm bound.*

Using a slight modification of the construction, we can show that our results are not idiosyncratic to instances which contain some strings multiple times. (Recall that the gadget from Lemma 1 in the construction contains  $2^{a-b}$  copies of the all-zero string.) The basic idea is to append an identity matrix to the strings we need to distinguish, and then to show using a slightly more involved analysis that the gadgets still work in the same way.

► **Proposition 6** (★). *Theorem 3 and Corollary 5 hold even if all input strings are distinct.*

Let  $p$ -NORM APPROVAL COMMITTEE be the variant of  $p$ -HDC in which we additionally get  $t \in \mathbb{N}$  as an input and require the number of ones in the solution string  $s^*$  to be exactly  $t$  [39]. Note that in the proof of Theorem 3 we have first shown that each solution string contains exactly  $\hat{n}$  ones. Thus, the reduction works in the same way for  $p$ -NORM APPROVAL COMMITTEE when we specify  $t = \hat{n}$  in the constructed instance. We hence obtain the following.

► **Corollary 7**. *For each fixed rational  $p > 1$ ,  $p$ -NORM APPROVAL COMMITTEE is NP-hard and admits no algorithm running in  $2^{o(\hat{n}+\hat{m})} \cdot |I'|^{O(1)}$ -time or in  $2^{o(k^{p/(p+1)})} \cdot |I'|^{O(1)}$ -time unless the ETH fails, where  $\hat{n}$  is the number of candidates,  $\hat{m}$  is the number of voters, and  $k$  is the  $p$ -norm bound.*

### 3 Algorithmic Results

We now turn to our positive results. In Section 3.1 we provide an efficient algorithm when the objective value  $k$  is small. In Section 3.2, we derive an integer convex programming formulation to obtain an efficient algorithm for instances where the number  $m$  of input strings is small. Finally, we give a simple 2-approximation in Section 3.3.

### 3.1 A Subexponential-Time Algorithm

In this section, we present an algorithm with running time  $2^{k^{p/(p+1)+\epsilon}} \cdot |I|^{O(1)}$  for any  $\epsilon > 0$  and input instance  $I$  with distance bound  $k$ . By the lower bound result given in Corollary 5, we know that under ETH, the running time of the obtained algorithm is tight.

The algorithm is built on two subcases, distinguishing on a relation between the number  $m$  of input strings and the distance bound  $k$ . In each subcase we use a distinct algorithm that runs in subexponential time when restricted to that subcase. To start with, a dynamic programming algorithm which keeps track of the achievable vector of Hamming distances to each input string after columns 1 to  $j \leq n$  has running time  $O(n \cdot k^m)$ .

► **Lemma 8** ( $\star$ ).  *$p$ -HDC can be solved in  $O(n \cdot k^m)$  time and space, where  $m$  and  $n$  are the number and the length of the input strings, respectively, and  $k$  is the  $p$ -norm distance bound.*

The dynamic program given in Lemma 8 is efficient if there is a small number  $m$  of input strings only. In particular, if  $m$  satisfies  $m \leq \frac{k^{p/(p+1)}}{\log k}$ , then we immediately obtain an  $O(n \cdot 2^{k^{p/(p+1)}})$ -time algorithm. Otherwise, we can use Lemma 9. The algorithm behind Lemma 9 is based on a different but related idea as the fixed-parameter algorithm for CLOSEST STRING given by Gramm et al. [22]: We use data reduction to shrink the length of the strings by  $k^p$ , observe that one of the input strings must be close to a solution with bound  $k$  if it exists, and then find the solution by a search tree.

► **Lemma 9**.  *$p$ -HDC can be solved in  $O(nm^2 \cdot k^{\frac{p \cdot k}{\sqrt[p]{m}}})$  time, where  $m$  and  $n$  are the number and the length of the input strings, respectively, and  $k$  is the  $p$ -norm distance bound.*

**Proof.** Let  $I = (S, k)$  be an instance of  $p$ -HDC with  $S = (s_1, \dots, s_m)$  being the input strings of length  $n$  and  $k$  being the  $p$ -norm distance bound. To show the statement, we first observe that if a column is an all-zero (resp. an all-one) column, then we can simply assume that an optimal solution will also have zero (resp. one) in this column as our objective function is convex. By preprocessing all columns that are either an all-zero or an all-one vector, we obtain an equivalent instance, where each column has at least a zero and at least a one. Thus, for each column, no matter which value a solution has at this column, it will always induce Hamming distance of at least one to some input string. Consequently, if there are more than  $k^p$  columns remaining, then we can simply answer “no” as any string will have cost more than  $k$  to the input. Otherwise, there remain at most  $k^p$  columns.

If  $I$  is a yes-instance, meaning that there is a solution  $s^*$  for  $I$  with  $\|(s^*, S)\|_p \leq k$ , then there is an input string  $s^{**} \in S$  whose Hamming distance satisfies  $d(s^{**}, s^*) \leq \sqrt[p]{\frac{k^p}{m}} = \frac{k}{\sqrt[p]{m}}$ . Thus, we iterate over all input strings in  $S$ , assuming in each iteration that the current string is the aforementioned  $s^{**}$ . For each string  $s_i$  that we assume to be the aforementioned  $s^{**}$ , we go over all strings  $\hat{s}$  that differ from  $s_i$  by  $k'$  columns with  $k' \leq \frac{k}{\sqrt[p]{m}}$ . We check whether  $\|(\hat{s}, S)\|_p \leq k$ . We answer “no” if for each input string  $s_i \in S$ , no length- $n$  string  $\hat{s}$  with  $d(s_i, \hat{s}) \leq \frac{k}{\sqrt[p]{m}}$  exists which satisfies  $\|(\hat{s}, S)\|_p \leq k$ .

It remains to show the running-time bound. Observe that the preprocessing for all-zero and all-one columns can be done in  $O(nm)$  time. After that, for each of the  $m$  input strings  $s_i$ , we search all strings of Hamming distance at most  $k' \leq \frac{k}{\sqrt[p]{m}}$  to  $s_i$ , and there are  $O(n^{\frac{k}{\sqrt[p]{m}}})$  such strings. For each of them, we compute the objective function, which can be accomplished in  $O(nm)$  time. As already reasoned, after the preprocessing,  $n$  is upper-bounded by  $k^p$ . Thus, the overall running time bound is  $O(nm + nm^2 \cdot n^{\frac{k}{\sqrt[p]{m}}}) = O(nm^2 \cdot k^{\frac{p \cdot k}{\sqrt[p]{m}}})$ , as claimed. ◀

Combining Lemma 8 with Lemma 9, we obtain a subexponential algorithm with respect to  $k$ .

► **Theorem 10.** *For each fixed positive value  $\varepsilon > 0$ ,  $p$ -HDC can be solved in  $O(nm^2 \cdot 2^{k^{p/(p+1)+\varepsilon}})$  time, where  $n$  and  $m$  denote the length and the number of input strings, and  $k$  is the  $p$ -norm distance bound with  $p > 1$ .*

**Proof.** Let  $I = (S, k)$  be an instance of  $p$ -HDC with  $S = (s_1, \dots, s_m)$  being the input strings of length  $n$  and  $k$  being the  $p$ -norm distance bound. As already discussed, to solve our problem we distinguish between two cases, depending on whether  $m \leq \frac{k^{p/(p+1)}}{\log k}$  holds.

If  $m \leq \frac{k^{p/(p+1)}}{\log k}$ , then  $k^m \leq k^{\frac{k^{p/(p+1)}}{\log k}} \leq 2^{k^{p/(p+1)}}$ . In this case, we use the dynamic programming approach given in the proof of Lemma 8, which has the desired running time  $O(n \cdot k^m) = O(n \cdot 2^{k^{p/(p+1)}})$ .

Otherwise,  $m > \frac{k^{p/(p+1)}}{\log k}$ , meaning that  $\frac{p \cdot k \cdot \log k}{\sqrt[p]{m}} < p \cdot k \cdot \log k / \sqrt[p]{\frac{k^{p/(p+1)}}{\log k}} = p \cdot k^{p/(p+1)} \cdot (\log k)^{(p+1)/p}$ . For each fixed positive  $\varepsilon \in \mathbb{R}$  there exists  $k_0 = k_0(p, \varepsilon) \in \mathbb{R}$  such that, for each  $k \geq k_0$ , we have  $p \cdot (\log k)^{(p+1)/p} < k^\varepsilon$ . If  $k < k_0$ , then the algorithm in the proof of Lemma 9 runs in  $O(nm^2)$  time. Otherwise  $k \geq k_0$ , which implies  $\frac{p \cdot k \cdot \log k}{\sqrt[p]{m}} < k^{p/(p+1)+\varepsilon}$ . Thus, the algorithm given in the proof of Lemma 9 has a running time of  $O(nm^2 \cdot k^{\frac{p \cdot k}{\sqrt[p]{m}}}) = O(nm^2 \cdot 2^{\frac{p \cdot k \cdot \log k}{\sqrt[p]{m}}}) = O(nm^2 \cdot 2^{k^{p/(p+1)+\varepsilon}})$ .

Altogether we presented an algorithm which has the desired running time bound. ◀

### 3.2 A Fixed-Parameter Algorithm for the Number of Input Strings

In this section, we show that minimizing the sum of the  $p$ -distances is fixed-parameter tractable for the number  $m$  of input strings. The idea is to formulate our problem as a combinatorial  $n$ -fold integer program (CnIP) with  $O(2^m)$  variables and  $O(m)$  constraints. We then apply the following simplified result of Knop et al. [28]:

► **Proposition 11** ([28, Theorem 3]). *Let  $E \in \mathbb{Z}^{(r+1) \times t}$  be a matrix such that the last row equals  $(1, 1, \dots, 1) \in \mathbb{Z}^t$ . Let  $b \in \mathbb{Z}^{r+1}$ ,  $\ell, u \in \mathbb{Z}^t$ , and let  $f: \mathbb{R}^t \rightarrow \mathbb{R}$  be a separable convex function given by an evaluation oracle<sup>2</sup>. Then, there is an algorithm that solves<sup>3</sup>  $P := \min\{f(x) \mid Ex = b \wedge \ell \leq x \leq u \wedge x \in \mathbb{Z}^t\}$  in  $t^{O(r)} \cdot ((1 + \|E\|_\infty) \cdot r)^{O(r^2)} \cdot L + T$  time, where  $L$  is the total bit-length of  $b, \ell, u$ , and the oracle of  $f$ , and  $T$  is the time that an algorithm needs to solve the continuous relaxation of  $P$ .*

To get a useful running time bound from Proposition 11, we need a bounded number of variables. To do this, we group columns in the input strings with the same “type” together and introduce an integer variable for each column type. To this end, given a set  $S = \{s_1, \dots, s_m\}$  of length- $n$  strings, we say that two columns  $j, j' \in [n]$  have the *same type* if for each  $i \in [m]$  it holds that  $s_i[j] = s_i[j']$ . The *type* of column  $j$  is its equivalence class in the same-type relation. Thus, each type is represented by a vector in  $\{0, 1\}^m$ . Let  $n'$  denote the number of different (column) types in  $S$ . Then,  $n' \leq \min(2^m, n)$ . Enumerate the  $n'$  column types as  $t_1, \dots, t_{n'}$ . Below we identify a column type with its index for easier notation. Using this, we can encode the set  $S$  succinctly by introducing a constant  $e(j)$  for each column type  $j \in [n']$  that denotes the number of columns with type  $j$ . Given a solution string  $s^*$ , we can also encode this string  $s^*$  via an integer vector  $x \in \{0, 1, \dots, n\}^{n'}$ , where for each type  $j \in [n']$  we define  $x[j]$  as the number of ones in the solution  $s^*$  whose corresponding columns are of type  $j$ . Note that this encodes all essential information in a solution, since the actual

<sup>2</sup> A function is separable convex if it is the sum of univariate convex functions.

<sup>3</sup> The algorithm correctly reports either a minimizer  $x \in P$  or that  $P$  is infeasible or unbounded.

order of the columns is not important (see Example 12). Vice versa, each integer vector in  $x \in \{0, 1, \dots, n\}^{n'}$  satisfying  $0 \leq x[j] \leq e(j)$  for each  $j \in [n']$  yields a length- $n$  binary string  $s^*(x)$ ; it remains to add constraints and a suitable objective function to ensure that  $s^*(x)$  has minimum sum of  $p$ -distances to the input strings.

► **Example 12.** For an illustration, let  $S = \{0000, 0001, 1110\}$ . The set  $S$  has two different column types, represented by  $(0, 0, 1)^T$ , call it type 1, and  $(0, 1, 0)^T$ , call it type 2. There are three columns of type 1 and one column of type 2. The solution 0110 for  $S$  can be encoded by two variables  $x[1] = 2$  and  $x[2] = 0$ .

We next introduce  $m$  variables  $y \in \{0, 1, \dots, n\}^m$  that shall be equal to the Hamming distances of each input string  $s_i$ ,  $i \in [m]$ , to the solution  $s^*(x)$  selected by  $x$ . To achieve this, we need a formula specifying the Hamming distance between the two strings  $s_i$  and  $s^*(x)$ , and this formula needs to be linear in  $x$ . This can be achieved as follows; for the sake of simplicity, we let  $s_i[j] = 1$  if the column of type  $j$  has one in the  $i^{\text{th}}$  row and  $s_i[j] = 0$  if it has zero in the  $i^{\text{th}}$  row:  $d(s_i, s^*(x)) = \sum_{j=1}^{n'} (s_i[j] \cdot (e(j) - x[j]) + (1 - s_i[j]) \cdot x[j]) = \sum_{j=1}^{n'} (e(j) \cdot s_i[j] + (1 - 2s_i[j]) \cdot x[j]) = w_i + \sum_{j=1}^{n'} x[j] \cdot (1 - 2s_i[j])$ , where we define  $w_i := \sum_{j=1}^{n'} e(j) \cdot s_i[j]$ , which denotes the number of ones in string  $s_i$ .

We can now formulate an appropriate CnIP. The variables are  $x \in \mathbb{R}^{n'}$ ,  $y \in \mathbb{R}^m$ , and a dummy variable  $z \in \mathbb{Z}$ . The bounds  $\ell, u$  for the variables are defined such that

- (1) for each  $j \in [n']$  it holds that  $0 \leq x[j] \leq e(j)$ ,
- (2) for each  $i \in [m]$  it holds that  $0 \leq y[i] \leq n$ , and
- (3) there is virtually no constraint on  $z$ , that is,  $-n' \cdot n + mn \leq z \leq n' \cdot n + mn$ .

The objective function is defined as  $f(x, y, z) = \sum_{i=1}^m y[i]^p$  which is clearly separable convex over the domain specified by  $\ell$  and  $u$ . Finally, the constraint system  $Et = b$ , where  $t^T = (x^T y^T z)$  is defined such that the first  $m$  constraints are  $\sum_{j=1}^{n'} (x[j] \cdot (1 - 2s_i[j])) - y[i] = -w_i$ , for each  $i \in [m]$ , and the last constraint is  $\sum_{j=1}^{n'} x[j] + \sum_{i=1}^m y[i] + z = 0$  (note that this constraint can always be fulfilled by setting  $z$  accordingly).

By the above reasoning, an instance of  $p$ -HDC is a yes-instance if and only if  $\min\{f(x) \mid Et = b \wedge \ell \leq t \leq u \wedge t \in \mathbb{Z}^{n'+n+1}\}$  is at most  $k^p$ . Plugging in the running time of Proposition 11, and using a polynomial-time algorithm for the continuous relaxation of the CnIP above [10], we obtain the following.

► **Theorem 13.**  $p$ -NORM HAMMING CENTROID can be solved in  $m^{O(m^2)} \cdot (n \cdot m)^{O(1)}$  time.

### 3.3 A Factor-2 Approximation

It is known that by taking an input string that minimizes the largest Hamming distance over all input strings, CLOSEST STRING can be approximated within factor 2. Indeed, using a similar idea, we show that the minimization version of our  $p$ -HDC problem can also be approximated within factor 2. More specifically, we show that an input string which has minimum  $p$ -norm to all other input strings is a 2-approximate solution.

► **Proposition 14** (\*). *The minimization variant of  $p$ -HDC can be approximated within factor 2 in polynomial time.*

## 4 Conclusion and Outlook

We analyzed the complexity of  $p$ -NORM HAMMING CENTROID for all fixed rational values  $p$  between  $p = 1$  and  $p = \infty$ . We believe that the running time bounds established in this paper, of essentially  $2^{\Theta(k^{\frac{p}{p+1}})} \cdot (nm)^{O(1)}$ , connect the extreme points  $p = 1$  and  $p = \infty$  in

a very satisfying way. We did not consider the non-norm case of  $0 < p < 1$ , as it does not fit our clustering motivation very well. But this non-convex case might be of independent interest, and may be the subject of future work. Furthermore, we have focused here on the case where  $p$  is a fixed constant. It would also be interesting to study the case where  $p$  is part of the input.

An interesting generalization of CLOSEST STRING is CLOSEST SUBSTRING in which we seek a string  $s^*$  of a certain specified length such that each of the input strings has a substring which is close to  $s^*$  (see, e.g., Ma and Sun [31]). It would be interesting to see how our results carry over to this and other similar variants. Finally, the fact that the simple 2-factor approximation for CLOSEST STRING carries over to  $p$ -HDC may imply that there are similar connections for approximation algorithms. This warrants further investigation into whether  $p$ -HDC admits a PTAS.

---

### References

- 1 Georgios Amanatidis, Nathanaël Barrot, Jérôme Lang, Evangelos Markakis, and Bernard Ries. Multiple referenda and multiwinner elections using hamming distances: Complexity and manipulability. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pages 715–723, 2015.
- 2 Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *Journal of Algorithms*, 52(2):120–133, 2004. doi:10.1016/j.jalgor.2004.02.003.
- 3 Gleb Beliakov, Humberto Bustince Sola, and Tomasa Calvo. *A Practical Guide to Averaging Functions*, volume 329 of *Studies in Fuzziness and Soft Computing*. Springer, 2016.
- 4 Paul S. Bradley, Olvi L. Mangasarian, and W. Nick Street. Clustering via concave minimization. In *Proceedings of Advances in Neural Information Processing Systems 9 (NIPS '96)*, pages 368–374, 1996.
- 5 Steven J. Brams, D. Marc Kilgour, and M. Remzi Sanver. A minimax procedure for negotiating multilateral treaties. In Rudolf Avenhaus and I. William Zartman, editors, *Diplomacy Games: Formal Models and International Negotiations*, pages 265–282. Springer, 2007. doi:10.1007/978-3-540-68304-9\_14.
- 6 Margaret L. Brandeau and Samuel S. Chiu. Parametric facility location on a tree network with an  $L_p$ -norm cost function. *Transportation Science*, 22(1):59–69, 1988.
- 7 Jiehua Chen, Danny Hermelin, and Manuel Sorge. A note on clustering aggregation. Technical report, arXiv:1807.08949, 2018. arXiv:1807.08949.
- 8 Jiehua Chen, Danny Hermelin, and Manuel Sorge. On computing centroids according to the  $p$ -norms of hamming distance vectors. Technical report, arXiv:1807.06469, 2018. arXiv:1807.06469.
- 9 Zhi-Zhong Chen, Bin Ma, and Lusheng Wang. A three-string approach to the closest string problem. *Journal of Computer and System Sciences*, 78(1):164–178, 2012. doi:10.1016/j.jcss.2011.01.003.
- 10 Sergei Chubanov. A polynomial-time descent method for separable convex optimization problems with linear constraints. *SIAM Journal on Optimization*, 26(1):856–889, 2016. doi:10.1137/14098524X.
- 11 Gérard D. Cohen, Iiro S. Honkala, Simon Litsyn, and Antoine Lobstein. *Covering Codes*, volume 54. North-Holland, 1997.
- 12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 13 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Lower bounds for approximation schemes for closest string. In *Proceedings of the 15th Scandinavian Workshop on Algorithm Theory (SWAT '16)*, volume 53 of *LIPICs*, pages 12:1–12:10. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

- 14 Amit Deshpande, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '11)*, pages 482–496, 2011.
- 15 Martin Dörnfelder, Jiong Guo, Christian Komusiewicz, and Mathias Weller. On the parameterized complexity of consensus clustering. *Theoretical Computer Science*, 542:71–82, 2014. doi:10.1016/j.tcs.2014.05.002.
- 16 Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. Committee scoring rules: Axiomatic characterization and hierarchy. *ACM Transactions on Economics and Computation*, 7(1):3:1–3:39, 2019.
- 17 Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. Multiwinner rules on paths from  $k$ -Borda to Chamberlin-Courant. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*, pages 192–198. AAAI Press, 2017.
- 18 Moti Frances and Ami Litman. On covering problems of codes. *Theory of Computing Systems*, 30:113–119, 1997.
- 19 Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- 20 Dennis C. Ghiglia and Louis A. Romero. Minimum  $l_p$ -norm two-dimensional phase unwrapping. *Journal of the Optical Society of America A*, 13(10):1999–2013, 1996.
- 21 René Gonin and Arthur H. Money. *Nonlinear  $L_p$ -norm Estimation*. Marcel Dekker, Inc., 1989.
- 22 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for Closest String and related problems. *Algorithmica*, 37(1):25–42, 2003.
- 23 Venkatesan Guruswami, Prasad Raghavendra, Rishi Saket, and Yi Wu. Bypassing UGC from some optimal geometric inapproximability results. *ACM Transactions on Algorithms*, 12(1):6:1–6:25, 2016.
- 24 Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2), 1950.
- 25 Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
- 26 D. Marc Kilgour. Approval balloting for multi-winner elections. In J.-F. Laslier and M.R. Sanver, editors, *Handbook on Approval Voting, Studies in Choice and Welfare*, chapter 6, pages 105–124. Springer, 2010.
- 27 Marius Kloft, Ulf Brefeld, Sören Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate  $l_p$ -norm multiple kernel learning. In *Proceedings of Advances in Neural Information Processing Systems 22 (NIPS '09)*, pages 997–1005, 2009.
- 28 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial  $n$ -fold integer programming and applications. Technical report, arXiv:1705.08657, 2017. arXiv:1705.08657.
- 29 J. Kevin Lanctôt, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Information and Computation*, 185(1):41–55, 2003.
- 30 Robert F. Love, J. James G. Morris, and George O. Wesolowsky. *Facilities Location: Models & Methods*. North-Holland, 1988.
- 31 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substrings problems. *SIAM Journal on Computing*, 39(4):1432–1443, 2009.
- 32 J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5<sup>th</sup> Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- 33 A. H. Money, J. F. Affleck-Graves, M. L. Hart, and G. D. I. Barr. The linear regression model:  $l_p$  norm estimation and the choice of  $p$ . *Journal of Communications in Statistics—Simulation and Computation*, 11(1):89–109, 1982.
- 34 Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Series: Studies in Applied and Numerical Mathematics. Society for Industrial and Applied Mathematics, 1994.
- 35 Fanny Pascual, Krzysztof Rzdca, and Piotr Skowron. Collective schedules: Scheduling meets computational social choice. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 18)*, pages 667–675, 2018. URL: <http://dl.acm.org/citation.cfm?id=3237482>.

## 28:16 On Computing Centroids According to the $p$ -Norms of Hamming Distance Vectors

- 36 Pavel A. Pevzner. *Computational Molecular Biology—An Algorithmic Approach*. MIT Press, 2000.
- 37 Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.
- 38 Douglas R. Shier and Perino M. Dearing. Optimal locations for a class of nonlinear, single-facility location problems on a network. *Operations Research*, 31(2):292–303, 1983.
- 39 Shankar Sivarajan. A generalization of the minisum and minimax voting methods. *SIAM Undergraduate Research Online*, 11, 2018. doi:10.1137/16S014870.
- 40 Wen-Jun Zeng, Hing-Cheung So, and Abdelhak M. Zoubir. An  $\ell_p$ -norm minimization approach to time delay estimation in impulsive noise. *Digital Signal Processing*, 23(4):1247–1254, 2013.