Equivalent Continuous Formulation of General Hashing Problem

Shengnan Wang, Chunguang Li, *Senior Member, IEEE*, and Hui-Liang Shen

***Abstract*—Hashing-based approximate nearest neighbors search has attracted broad research interest, due to its low computational cost and fast retrieval speed. The hashing technique maps the data points into binary codes and, meanwhile, preserves the similarity in the original space. Generally, we need to solve a discrete optimization problem to learn the binary codes and hash functions, which is NP-hard. In the literature, most hashing methods choose to solve a relaxed problem by discarding the discrete constraints. However, such a relaxation scheme will cause large quantization error, which makes the learned binary codes less effective. In this paper, we present an equivalent continuous formulation of the discrete hashing problem. Specifically, we show that the discrete hashing problem can be transformed into a continuous optimization problem without any relaxations, while the transformed continuous optimization problem has the same optimal solutions and the same optimal value as the original discrete hashing problem. After transformation, the continuous optimization methods can be applied. We devise the algorithms based on the idea of DC (difference of convex functions) programming to solve this problem. The proposed continuous hashing scheme can be easily applied to the existing hashing models, including both supervised and unsupervised hashing. We evaluate the proposed method on several benchmarks and the results show the superiority of the proposed method compared with the state-of-the-art hashing methods.**

***Index Terms*—Continuous hashing, DC programming, large-scale image retrieval, learning to hash (LH).**

I. Introduction

Recently, hashing-based approximate nearest neighbors search has attracted substantial attention in various fields, including large-scale image retrieval, machine learning, and computer vision [1]–[11]. Hashing method encodes high-dimensional data points into compact binary codes, and meanwhile maps similar data points to adjacent binary hash codes to preserve the similarity in the original space. Using the binary codes to represent the massive data, nearest neighbor search can be easily accomplished with a constant time complexity.

Hashing methods can be mainly divided into data-independent hashing [2], [15], [16] and data-dependent hashing [6], [10], [12], [31], [36]. One of the most popular data-independent hashing methods is locality sensitive hashing (LSH) [23], which generates the hash functions by random projections. Though LSH can map similar samples to similar codes with high probability, it usually requires long bits to achieve good precision. Such a drawback restricts its application.

In recent years, the learning-based data-dependent hashing methods have become more popular than the data-independent hashing methods, for the benefit that they can effectively and efficiently learn very compact binary codes of massive data. Different from the data-independent hashing, data-dependent hashing methods learn binary codes and hash functions from data, so they are also called learning to hash (LH) methods. Representative LH approaches include spectral hashing [7], iterative quantization (ITQ) [6], supervised discrete hashing (SDH) [12], structure sensitive hashing [14], and hashing with angular reconstructive embeddings [47]. More recently, Do *et al.* [20], Liong *et al.* [21], and Liu *et al.* [22] used deep networks to learn binary hash codes and achieved promising performance. He *et al.* [24] proposed a novel hashing scheme which can significantly save the training cost and meanwhile preserve the semantic similarity of the original data. Leng *et al.* [26], Wang *et al.* [27], and Liu *et al.* [28] proposed distributed hashing methods to deal with the large-scale image retrieval problems. The LH methods can be further divided into unsupervised hashing [5], [17], [23], [25], [29], [48]–[50] and supervised hashing [10], [12], [19], [30], [34], [37], [53]. The main difference between unsupervised hashing and supervised hashing is whether the label information is available for learning hash functions.

In general, the data-dependent hashing methods (including both unsupervised and supervised hashing) need to solve a discrete optimization problem to learn hash functions and hash codes, which is NP-hard. In the existing literature, most hashing methods choose to relax the problem by dropping the discrete constraints [7], [38]. Such a scheme brings much convenience and greatly simplifies the original problem. However, the relaxed problem can only obtain an approximate solution, which is proved to be less effective and of low quality, especially when the codes have long bits. In this situation, some discrete optimization methods [12], [17], [18] are

Manuscript received May 30, 2018; revised October 19, 2018; accepted January 16, 2019. Date of publication January 30, 2019; date of current version August 4, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61571392 and Grant 61631003, and in part by the National Program for Special Support of Eminent Professionals. This paper was recommended by Associate Editor X. Li. *(Corresponding author: Chunguang Li.)*

The authors are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China, and also with the Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking, Zhejiang University, Hangzhou 310027, China (e-mail: cgli@zju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2019.2894020.

Digital Object Identifier 10.1109/TCYB.2019.2894020

proposed, which learn the binary codes without relaxations. Shen *et al.* [12] proposed a discrete supervised hashing method named SDH, and Luo *et al.* [13] further gave a robust version of SDH which can effectively suppress the influence of unreliable binary codes and potentially noisily labeled samples. SDH and its variants solve the hashing problem using the discrete cyclic coordinate descent, which is not time efficient and can only be applied to the specific hashing model having the form of binary quadratic program. In addition, SDH is not suitable for solving the hashing problem with bits uncorrelation and balance constraints, which are widely used in hashing literature to make the binary codes efficient. Liu *et al.* [17] proposed a discrete graph hashing method for unsupervised hashing. However, this method needs to solve expensive optimization subproblems at each iteration and the proposed discrete optimization framework in [17] can only be applied to the graph hashing model. Shen *et al.* [35] gave a discrete optimization method termed discrete proximal linearized minimization, which uses the projected gradient method to solve the discrete binary codes learning problem. Generally, the projected gradient method is suitable for solving the convex optimization problem. However, the discrete hashing problem is not a convex optimization problem, since the projection set is discrete. So the convergence of the projected gradient iteration is difficult to be guaranteed. Attouch *et al.* [40] claimed that if the objective function has Lipschitz continuous gradient in the whole real space $\mathbb{R}^n$ and some other conditions are satisfied, then the projected gradient method can converge to a critical point. However, for most hashing problems, we can only guarantee the objective function has Lipschitz continuous gradient in a finite region. In addition, even these conditions are satisfied, the projected gradient method can only converge to a critical point while the quality of the critical point is not guaranteed. Since the discrete hashing problem usually has a mass of critical points, the performance of the convergence point may be of low quality.

In this paper, to overcome the above-mentioned drawbacks, we propose a method that transforms the discrete binary code learning problem into an equivalent continuous optimization problem without any relaxations. The main contributions of this paper are summarized as follows.

1) We present an equivalent continuous formulation of the discrete hashing problem. We show that the discrete hashing problem can be transformed into a continuous optimization problem without any relaxations. The transformed continuous optimization problem has the same optimal solutions and the same optimal value as the original discrete optimization problem.
2) After transformation, we need to deal with a continuous nonconvex optimization problem. We devise the algorithms based on the idea of DC programming to solve the problem for obtaining binary codes and hash functions.
3) The proposed scheme can be applied to the existing discrete hashing models, including both supervised hashing and unsupervised hashing as well as hashing with bits uncorrelation and balance constraints. We apply the scheme to solve the large-scale image retrieval problems, and perform numerical simulations on representative retrieval benchmarks to show the superiority of the proposed method, compared with the state-of-the-art methods.

The rest of this paper is organized as follows. Section II introduces the general hashing problem and reformulates the discrete hashing problem as an equivalent continuous optimization problem. In Section III, we propose the algorithms based on the idea of DC programming to solve the hashing problems. In Section IV, we evaluate the proposed methods on several benchmarks with comparison to the state-of-the-art methods. Finally, we draw the conclusion in Section V.

Notation: In this paper, we denote a matrix by a capital letter, and a vector by a lowercase letter. We use **1** to denote the vector with all components one. We use I to denote an identity matrix. For a vector x, we use x^{T} to denote the transpose of x and we use $||x||$ to denote its l_2-norm. For a matrix X, $||X||$ denotes its Frobenius norm and $\mathrm{tr}(X)$ denotes the trace of X. We use $\mathrm{sign}(\cdot)$ to denote the element-wise sign function.

II. Continuous Formulation of Hashing Problem

A. General Hashing Problem

Let $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d\times n}$ denote the data set of n samples. The aim of hashing is to encode each high-dimensional data point x into a compact binary code $b \in \{-1, 1\}^c$. Generally, such a task is finished by solving the following binary code learning problem:

$$\begin{aligned} \min_B \quad & \mathcal{L}(B) \\ \text{s.t.} \quad & B \in \{-1, 1\}^{c\times n} \end{aligned} \tag{1}$$

where $B = [b_1, b_2, \dots, b_n]$ are the binary codes of the whole data and $\mathcal{L}(B)$ is the objective function which is determined by the specific hashing model.

The above problem is a discrete optimization problem, which is NP-hard. Traditional hashing methods relax the problem by discarding the discrete constraints. As stated before, such a manner will bring large accumulated quantization error and make the binary codes less effective.

B. Continuous Formulation of Discrete Hashing

The relaxed method directly removes the discrete constraints, which makes the relaxed problem quite different from the original problem. In other words, the optimal solution of the relaxed problem is usually different from that of the original problem. Completely ignoring the discrete constraints is inadvisable. In this section, we present an equivalent continuous formulation of the discrete binary codes learning problem. We show that the discrete binary codes learning problem can be transformed into a continuous optimization problem without any relaxations, and the transformed continuous optimization problem has the same optimal solutions and the same optimal value as the original problem. Though usually the transformed continuous optimization problem is a nonconvex optimization problem and it is still NP-hard to obtain its optimal solution, after transformation, we can apply continuous optimization methods to solve the discrete hashing problem to obtain high-quality solutions. As we can see, in the literature, there are more efficient methods that can be used for continuous optimization, while the methods for discrete optimization are relatively rare and the performance is usually not satisfactory.

In this section, we consider a relatively simple case: the hashing problem without bits uncorrelation and balance constraints, that is, problem (1). To reformulate problem (1) in a continuous optimization form, we need the following lemma.

Lemma 1 [41]: Suppose that f is Lipschitz continuous on $\mathcal{X}$ with constant L. Let φ be a non-negative function defined on $\mathcal{X}$ and $S := \{x \in \mathcal{X} : \varphi(x) = 0\}$. If $d(x, S) \le \varphi(x)$ for all $x \in \mathcal{X}$, then for all $\gamma \ge L$, the two problems

$$\inf\{f(x) : x \in S\} \text{ and } \inf\{f(x) + \gamma\varphi(x) : x \in \mathcal{X}\}$$

are equivalent, that is, the two problems have the same optimal solutions and the same optimal value. Here

$$d(x, S) := \inf_{z \in S} ||x - z||$$

denotes the distance from a point x to the set S.

Compared with the hashing problem having the bits uncorrelation and balance constraints, problem (1) is relatively simple, and the process of transforming problem (1) into a continuous optimization problem is also relatively easy. In the following, we present the specific transformation method.

Let

$$\mathcal{B}_1 = \left\{B \in \mathbb{R}^{c\times n} : B \in [-1, 1]^{c\times n}\right\} \tag{2}$$

and

$$\mathcal{B}_2 = \left\{B \in \mathbb{R}^{c\times n} : B \in \{-1, 1\}^{c\times n}\right\}.$$

For all $x \in \mathcal{B}_1$, we have that

$$\begin{aligned} d^2(x, \mathcal{B}_2) &= \min\left\{\sum_{k=1}^{c}\sum_{i=1}^{n}\left(x_{k,i} - y_{k,i}\right)^2 : y \in \mathcal{B}_2\right\} \\ &= \sum_{k=1}^{c}\sum_{i=1}^{n}\left(1 - |x_{k,i}|\right)^2. \end{aligned} \tag{3}$$

Thus,

$$\begin{aligned} d(B, \mathcal{B}_2) &= \sqrt{\sum_{k=1}^{c}\sum_{i=1}^{n}(1 - |B_{k,i}|)^2} \\ &\le \sum_{k=1}^{c}\sum_{i=1}^{n}(1 - |B_{k,i}|) \\ &\le \sum_{k=1}^{c}\sum_{i=1}^{n}(1 - B_{k,i}^2) \\ &= cn - \mathrm{tr}(BB^{\mathrm{T}}) = \varphi_1(B) \end{aligned} \tag{4}$$

for all $B \in \mathcal{B}_1$. It is easy to verify that $\varphi_1(B) \ge 0$ and $B \in \mathcal{B}_2$ iff $\varphi_1(B) = 0$, for all $B \in \mathcal{B}_1$.

According to Lemma 1, we have that problem (1) is equivalent to the following continuous optimization problem:

$$\min_{B\in\mathcal{B}_1} \quad \mathcal{L}(B) + \gamma\varphi_1(B) \tag{5}$$

for all $\gamma \ge L$, where L is the Lipschitz constant of $\mathcal{L}(B)$ on $\mathcal{B}_1$.

This scheme can be applied to most existing hashing methods, including both supervised and unsupervised hashing. For example, Shen *et al.* [12] proposed a supervised hashing model which learns binary codes by optimizing

$$\begin{aligned} \min_{B,W} \quad & \sum_{i=1}^{n} ||y_i - W^{\mathrm{T}}b_i||^2 + \nu||W||^2 \\ \text{s.t.} \quad & b_i \in \{-1, 1\}^c \end{aligned} \tag{6}$$

where $y_i \in \{0, 1\}^r$ is the label of data x_i, $W \in \mathbb{R}^{c\times r}$ is the classification matrix, and ν is the regularization parameter. The above problem (6) can be rewritten as the matrix form

$$\begin{aligned} \min_{B,W} \quad & ||Y - W^{\mathrm{T}}B||^2 + \nu||W||^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{c\times n} \end{aligned} \tag{7}$$

where $Y = [y_1, \ldots, y_n] \in \mathbb{R}^{r\times n}$ is the label matrix. Using transformation approach shown above, the continuous equivalent of (7) can be represented as

$$\begin{aligned} \min_{B,W} \quad & ||Y - W^{\mathrm{T}}B||^2 + \nu||W||^2 + \gamma\varphi_1(B) \\ \text{s.t.} \quad & B \in [-1, 1]^{c\times n}. \end{aligned} \tag{8}$$

The unsupervised hashing problem can also be equivalently formulated as a continuous optimization problem, for example, the unsupervised graph hashing [7], which learns binary codes by solving the following problem:

$$\begin{aligned} \min_{B} \quad & \mathcal{L}(B) = \mathrm{tr}(B(D - M)B^{\mathrm{T}}) \\ \text{s.t.} \quad & B \in \{-1, 1\}^{c\times n} \end{aligned} \tag{9}$$

where $M_{n\times n}$ is the affinity matrix with $M_{i,j} = \exp(-||x_i - x_j||^2/\epsilon)$ and D is a diagonal $n \times n$ matrix with $D_{i,i} = \sum_j M_{i,j}$. Here, the parameter $\epsilon > 0$ defines the distance in $\mathbb{R}^d$ which corresponds to similar items. Note that the time complexity to compute M is $\mathcal{O}(n^2)$, which is unacceptable in large-scale applications. In the real application, we will construct an anchor-based affinity matrix [5] $M = ZZ^{\mathrm{T}}$, where $Z \in \mathbb{R}^{n\times m}$ is a truncated similarity matrix. The time complexity to construct M by Z is only $\mathcal{O}(nm)$ with m anchors ($m \ll n$).

Applying the proposed continuous transformation method, we give the continuous formulation of problem (9)

$$\min_{B\in\mathcal{B}_1} \ \mathrm{tr}\left(B(D - M)B^{\mathrm{T}}\right) + \gamma\varphi_1(B). \tag{10}$$

Remark 1: The term $\varphi_1(B)$ can be seen as the penalty which is used to eliminate the quantization error and force the optimal solution of the transformed problem to be in the original domain $\mathcal{B}_2$. The condition $\gamma \ge L$ is only a sufficient condition but not necessary. Usually, a smaller γ is enough to guarantee the equivalence of problem (5) and problem (1), especially when B is large scale and correspondingly the Lipschitz constant L is usually very large. Though a large γ can always ensure that the optimal solution of (5) is the same as that of (1), since $\varphi_1(B)$ is concave, large γ will also add the difficulty of optimizing (5). In the experiment section, we will show that a moderate γ is enough to guarantee the quantization error to be zero.

C. Continuous Formulation of Hashing With Bits Uncorrelation and Balance Constraints

To make the binary codes efficient, most hashing methods will add the bits uncorrelation and balance constraints. With these constraints, the hashing problem becomes

$$\begin{aligned}\min_{B}\ & \mathcal{L}(B)\\ \text{s.t. } & B \in \{-1,1\}^{c\times n}\\ & B\mathbf{1} = 0\\ & BB^{\mathrm{T}} = nI. \end{aligned}\tag{11}$$

From the last section, we see that, by introducing $\varphi_1(B)$, the discrete hashing problem (1) can be transformed into a continuous optimization problem. However, $\varphi_1(B)$ can only transform the discrete domain $\mathcal{B}_2$ into the continuous domain $\mathcal{B}_1$, but it cannot handle the additional bits uncorrelation and balance constraints. In this section, we will show that by constructing suitable $\varphi(B)$, the bits uncorrelation and balance constraints can also be eliminated, and the above problem (11) can be transformed into an equivalent continuous unconstrained optimization problem. Meanwhile, the transformed problem still has the same optimal solutions and the same optimal value as the original problem (11). Since the two constraints make the problem more complex, it is difficult to directly construct $\varphi(B)$ like the last section. In the following, we give the method that constructs $\varphi(B)$ step by step, and one can see that due to the existence of the uncorrelation and balance constraints, $\varphi(B)$ is complex and quite different from $\varphi_1(B)$.

Let

$$\begin{aligned}\mathcal{V} &= \left\{B \in \mathbb{R}^{c\times n} : B \in \{-1,1\}^{c\times n}, B\mathbf{1} = 0\right\}\\ \mathcal{W} &= \left\{B \in \mathbb{R}^{c\times n} : B \in \{-1,1\}^{c\times n}, B\mathbf{1} = 0, BB^{\mathrm{T}} = nI\right\}.\end{aligned}$$

From Lemma 1, we see that the key point to construct $\varphi(B)$ is to compute the upper bound of $d(B,\mathcal{W})$ for all $B \in \mathcal{B}_1$. In the following, we give the upper bound of $d(B,\mathcal{W})$, $\forall B \in \mathcal{B}_1$ step by step.

Lemma 2: For all $B \in \mathcal{B}_2$

$$d(B,\mathcal{V}) \le ||B\mathbf{1}||.$$

Proof: Let $\mathcal{X} := \{x \in \mathbb{R}^n : x \in \{-1,1\}^n\}$ and let $\mathcal{Y} := \{y \in \mathbb{R}^n : y \in \{-1,1\}^n, y^{\mathrm{T}}\mathbf{1} = 0\}$. Let a^+ denote the number of the components of x equaling to 1, and a^- be the number of the components of x equaling to -1. Without loss of generality, we assume $a^+ \ge a^-$. Then one has that for all $x \in \mathcal{X}$

$$\begin{aligned} d^2(x,\mathcal{Y}) &= 4\left(a^+ - \frac{n}{2}\right)\\ &= 4\left(a^+ - \frac{a^+ + a^-}{2}\right)\\ &= 2\left(a^+ - a^-\right)\\ &\le \left(a^+ - a^-\right)^2 = \left(x'\mathbf{1}\right)^2. \end{aligned}$$

The inequality holds because $a^+ - a^- = 0$ or $a^+ - a^- \ge 2$. Therefore, for all $B \in \mathcal{B}_2$

$$d(B,\mathcal{V}) \le ||B\mathbf{1}||.\tag{12}$$

Lemma 2 is proved. ■

Let $\varphi_2(B) = ||B\mathbf{1}||$. It is easy to see that $\varphi_2(B) = 0$ iff $B \in \mathcal{V}$, for all $B \in \mathcal{B}_2$, and $\varphi_2(B)$ is Lipschitz continuous on $\mathcal{B}_1$.

Lemma 3: Let τ_1 be the Lipschitz constant of $\varphi_2(B)$ on $\mathcal{B}_1$, then one has that

$$d(B,\mathcal{V}) \le (1+\tau_1)\varphi_1(B) + \varphi_2(B), \forall B \in \mathcal{B}_1.$$

Proof: Since $\varphi_2(B)$ is Lipschitz continuous on $\mathcal{B}_1$ with constant τ_1, we have that

$$\varphi_2(B_1) \le \varphi_2(B_2) + \tau_1 d(B_1,B_2), \forall B_1, B_2 \in \mathcal{B}_1.$$

Let $B \in \mathcal{B}_1$ and let $B' \in \mathcal{B}_2$ be the projection of B on $\mathcal{B}_2$. Then we have

$$\begin{aligned} d(B,\mathcal{V}) &\le d(B,\mathcal{B}_2) + d\left(B',\mathcal{V}\right)\\ &\le d(B,\mathcal{B}_2) + \varphi_2\left(B'\right)\\ &\le (1+\tau_1)d\left(B,B'\right) + \varphi_2(B)\\ &= (1+\tau_1)d(B,\mathcal{B}_2) + \varphi_2(B)\\ &\le (1+\tau_1)\varphi_1(B) + \varphi_2(B). \end{aligned}$$

The lemma is proved. ■

Let $\zeta(B) = (1+\tau_1)\varphi_1(B) + \varphi_2(B)$, then one has that for all $B \in \mathcal{B}_1$, $d(B,\mathcal{V}) \le \zeta(B)$ and $\zeta(B) = 0$ iff $B \in \mathcal{V}$.

Next, we give an upper bound of $d(B,\mathcal{W})$, $\forall B \in \mathcal{V}$.

Lemma 4: For all $B \in \mathcal{V}$, one has that

$$d(B,\mathcal{W}) \le \varsigma\left(||BB^{\mathrm{T}}|| - n\sqrt{c}\right)$$

where $\varsigma = n^{(3/2)}c^{(3/2)}$.

Proof: It is easy to see that

$$d(B,\mathcal{W}) = \min\{||B-H||_F : H \in \mathcal{W}\} \le 2\sqrt{cn}$$

for all $B \in \mathcal{V}$. Let x_1 and x_2 be two vectors from $\mathcal{D} := \{x \in \mathbb{R}^n : x \in \{-1,1\}^n, x^{\mathrm{T}}\mathbf{1} = 0\}$. If $x_1^{\mathrm{T}}x_2 \ne 0$, then it is easy to see that $|x_1^{\mathrm{T}}x_2| \ge 2$. Therefore, for $B \in \mathcal{V}$, if $B \notin \mathcal{W}$, we have that $BB^{\mathrm{T}} \ne nI$ and

$$||BB^{\mathrm{T}}||_F^2 - n^2c \ge 4.$$

Thus, for all $B \in \mathcal{V}$, we have that

$$d(B,\mathcal{W}) \le \sqrt{cn}\left(||BB^{\mathrm{T}}||_F^2 - n^2c\right)/2.\tag{13}$$

Note that

$$||BB^{\mathrm{T}}||_F^2 - n^2c = \left(||BB^{\mathrm{T}}|| + n\sqrt{c}\right) * \left(||BB^{\mathrm{T}}|| - n\sqrt{c}\right)$$

and

$$||BB^{\mathrm{T}}|| + n\sqrt{c} \le nc + n\sqrt{c} \le 2nc$$

then

$$||BB^{\mathrm{T}}||_F^2 - n^2c \le 2nc * \left(||BB^{\mathrm{T}}|| - n\sqrt{c}\right).$$

Therefore,

$$d(B,\mathcal{W}) \le n^{\frac{3}{2}}c^{\frac{3}{2}}\left(||BB^{\mathrm{T}}|| - n\sqrt{c}\right).$$

Lemma 4 is proved. ■

Let $\varphi_3(B) = ||BB^{\mathrm{T}}|| - n\sqrt{c}$. It is easy to verify that $\varphi_3(B) = 0$ iff $B \in \mathcal{W}$, for $B \in \mathcal{V}$.

Lemma 5: Let τ_2 be the Lipschitz constant of $\varsigma\varphi_3(B)$ on $\mathcal{B}_1$, then for all $B \in \mathcal{B}_1$, one has that

$$\begin{aligned} d(B, \mathcal{W}) &\leq (1+\tau_2)\zeta(B) + \varsigma\varphi_3(B) \\ &= (1+\tau_2)[(1+\tau_1)\varphi_1(B) + \varphi_2(B)] + \varsigma\varphi_3(B). \end{aligned}$$

The proof of Lemma 5 is the same to the proof of Lemma 3, so we omit it.

Let $\varphi(B) = (1+\tau_2)[(1+\tau_1)\varphi_1(B)+\varphi_2(B)] + \varsigma\varphi_3(B)$. One has that

$$d(B, \mathcal{W}) \leq \varphi(B)$$

and $\varphi(B) = 0$ iff $B \in \mathcal{W}$, for all $B \in \mathcal{B}_1$. According to Lemma 1, we derive that problem (11) is equivalent to the following unconstrained continuous optimization problem:

$$\begin{aligned} \min_B \;& \mathcal{L}(B) + \gamma\varphi(B) \\ &= \mathcal{L}(B) + \gamma(1+\tau_2)[(1+\tau_1)\varphi_1(B) + \varphi_2(B)] \\ &\quad + \varsigma\gamma\varphi_3(B) \\ \text{s.t.} \;& B \in [-1, 1]^{c\times n}. \end{aligned} \tag{14}$$

For convenience, we can represent problem (14) by

$$\begin{aligned} \min_B \;& \mathcal{L}(B) + \eta_1\varphi_1(B) + \eta_2\varphi_2(B) + \eta_3\varphi_3(B) \\ \text{s.t.} \;& B \in [-1, 1]^{c\times n} \end{aligned} \tag{15}$$

where η_1, η_2 and η_3 are the parameters. Compared with (5), the bits uncorrelation and balance constraints are eliminated by introducing $\varphi_2(B)$ and $\varphi_3(B)$.

Also, we can apply the above scheme to existing hashing models with bits uncorrelation and balance constraints. First, we consider the supervised case. We consider that the objective function $\mathcal{L}(B)$ in (11) has the same formulation as the objective function in (7), that is, $\mathcal{L}(B) = ||Y - W^{\mathrm{T}}B||^2 + \nu||W||^2$. Then the continuous formulation of the supervised hashing with bits uncorrelation and balance constraints can be represented as

$$\begin{aligned} \min_{B\in\mathcal{B}_1} \;& ||Y - W^{\mathrm{T}}B||^2 + \nu||W||^2 + \eta_1\varphi_1(B) + \eta_2\varphi_2(B) \\ &+ \eta_3\varphi_3(B). \end{aligned} \tag{16}$$

Similarly, in the unsupervised case $\mathcal{L}(B) = \mathrm{tr}(B(D-M)B^{\mathrm{T}})$, the continuous formulation can be represented as

$$\min_{B\in\mathcal{B}_1} \mathrm{tr}\big(B(D-M)B^{\mathrm{T}}\big) + \eta_1\varphi_1(B) + \eta_2\varphi_2(B) + \eta_3\varphi_3(B). \tag{17}$$

Remark 2: In the hashing problem (11), the bits uncorrelation and balance constraints are introduced to make the code length short (more efficient). From (15), we see that the terms $\eta_2\varphi_2(B)$ and $\eta_3\varphi_3(B)$ force the variable to satisfy the bits uncorrelation and balance constraints. The larger the parameters η_2, η_3 are, the more the two constraints are satisfied. Different from the traditional constrained optimization problems in which the constraints should be completely satisfied, in the hashing problems, the bits uncorrelation and balance constraints are added to make the binary codes efficient, which are not treated as hard constraints that cannot be violated. In the hashing problem, we preferentially minimize the objective function to make the binary codes effective. Shen *et al.* [12] showed that excessively emphasizing the two constraints will reduce the effectiveness of the solution. In Section IV, we will also show that large η_2, η_3 are not conducive to getting good binary codes. In real applications, in order to achieve a tradeoff between effectiveness and efficiency, we can tune the parameters η_2, η_3 with cross-validation techniques.

III. Algorithms for Solving Equivalent Continuous Hashing Problems

In the above section, we have transformed the discrete hashing problems into continuous optimization problems, then many continuous optimization methods can be used to obtain the binary codes and hash functions. In this section, we propose some algorithms based on the idea of DC programming to solve the above continuous optimization problems. First, we introduce some preliminaries of DC programming.

A. Brief Introduction of DC Programming

DC programming [42] is very suitable for solving smooth or nonsmooth nonconvex continuous optimization problem. The main idea of DC programming is to decompose the objective function $f(x)$ as

$$\min f(x) = g(x) - h(x), x \in \mathbb{R}^d$$

where both g and h are convex functions. For a constrained optimization problem

$$\min f(x) = g(x) - h(x), x \in \mathcal{C} \tag{18}$$

where $\mathcal{C} \subseteq \mathbb{R}^d$ is a convex set, by introducing the indicator function, the above problem can be formulated as the standard DC form

$$\min I_{\mathcal{C}}(x) + g(x) - h(x), x \in \mathbb{R}^d$$

where

$$I_{\mathcal{C}}(x) = \begin{cases} 0, & x \in \mathcal{C} \\ \infty, & \text{else.} \end{cases}$$

DC programming solves the above problems by DC algorithm (DCA), which quite often gives global solutions. DCA is proved to be more robust and more efficient than other related standard methods, especially in the large-scale setting [42]–[45]. The generic DCA scheme for solving (18) can be formulated as Algorithm 1. Here, $\partial h(x)$ denotes the subgradient of $h(x)$ defined as

$$\partial h(x) = \Big\{ y \in \mathbb{R}^d : h(w) - h(x) \geq \langle w - x, y\rangle, \forall w \in \textbf{dom}\ h \Big\}.$$

If $h(x)$ is differentiable, then $\partial h(x)$ is just the gradient of $h(x)$.

For a detailed introduction of DC programming and DCA, one can refer to [43] and [46].

B. Algorithm for Solving Problem (8)

Like [12], we optimize problem (8) by alternatively updating the variables W and B. With fixed B, the variable W is updated by

$$W = \big(BB^{\mathrm{T}} + \nu I\big)^{-1} BY^{\mathrm{T}}. \tag{19}$$

```
Algorithm 1 DCA
Initialization
    Let $x^0 \in \mathcal{C}$ be a starting point, and set $k = 0$.
Repeat
    • Choose $y^k \in \partial h(x^k)$;
    • Choose
      $x^{k+1} \in \arg\min\{g(x) - [h(x^k) + \langle (x - x^k), y^k \rangle]; x \in \mathcal{C}\}$;
    • $k = k + 1$;
Until Convergence condition holds.
```

```
Algorithm 2 CSH
Initialization
    Input training data $\{X, Y\}$; classification matrix $W$; code
    length $c$; parameter $\gamma$. Initialize the starting point $B^0$ by the
    sign of random Gaussian matrix. Set $k = 0$.
Repeat
    Compute $A^k = 2(1 + \gamma)B^k$;
    Update $B$ by $B^{k+1} = \mathcal{P}_{\mathcal{B}_1}[(2WW^{\mathrm{T}} + 2I)^{-1}(2WY + A^k)]$;
    $k = k + 1$;
Until converge or reach maximum iterations.
Output $B = \mathrm{sign}(B)$;
```

With fixed W, the variable B is updated by optimizing

$$\begin{aligned}\min_{B} \quad & ||Y - W^{\mathrm{T}}B||^2 + \gamma\varphi_1(B) \\ \text{s.t.} \quad & B \in [-1, 1]^{c\times n}.\end{aligned} \tag{20}$$

Note that the objective function of (20) is nonconvex with respect to B. We aim to solve problem (20) by DC programming. According to the last section, the key step to apply DC programming is to decompose the objective function of problem (20) into the form of the difference of convex functions like (18). We rewrite problem (20) as

$$\begin{aligned}\min_{B} \quad & G(B) - H(B) \\ \text{s.t.} \quad & B \in [-1, 1]^{c\times n}\end{aligned} \tag{21}$$

where $G(B) = ||Y - W^{\mathrm{T}}B||^2 + \mathrm{tr}(BB^{\mathrm{T}})$, and $H(B) = \mathrm{tr}(BB^{\mathrm{T}}) + \gamma(\mathrm{tr}(BB^{\mathrm{T}}) - cn)$. According to the generic DCA scheme shown in Algorithm 1, to solve problem (21), at each iteration k, we need to compute

$$A^k \in \partial H\left(B^k\right) = 2(1 + \gamma)B^k. \tag{22}$$

Then, we compute B^{k+1} by solving the convex programming

$$\min_{B\in\mathcal{B}_1} ||Y - W^{\mathrm{T}}B||^2 + \mathrm{tr}\left(BB^{\mathrm{T}}\right) - \mathrm{tr}\left(B^{\mathrm{T}}A^k\right) \tag{23}$$

which has the analytical solution

$$B^{k+1} = \mathcal{P}_{\mathcal{B}_1}\left[\left(2WW^{\mathrm{T}} + 2I\right)^{-1}\left(2WY + A^k\right)\right]. \tag{24}$$

We summarize the procedure of updating B in Algorithm 2. In practice, we only need to alternatively update W and B $2 \sim 5$ times.

After the binary codes are obtained by the algorithm, we then use them to generate a hash function so that a novel query $x \in \mathbb{R}^d$ can be efficiently encoded into a binary code. We adopt the widely used nonlinear hash function having the form [12], [51]

$$F(x) = \mathrm{sign}\left(P^{\mathrm{T}}\phi(x)\right) \tag{25}$$

where $\phi(x) = [\exp(-||x - a_1||^2/\sigma), \dots, \exp(-||x - a_m||^2/\sigma)]^{\mathrm{T}}$ is an m-dimensional column vector obtained using the RBF kernel mapping and $P \in \mathbb{R}^{m\times c}$ is a projection matrix that projects $\phi(x)$ onto the low-dimensional space. The vectors $a_1, \dots, a_m$ are m anchor points randomly selected from the training samples and σ is the kernel width. When the binary codes B are obtained, the hash function is learned by minimizing

$$\min_{P} ||B - P^{\mathrm{T}}\phi(X)||^2.$$

It is easy to see that the optimal solution is

$$P = \left(\phi(X)\phi(X)^{\mathrm{T}}\right)^{-1}\phi(X)B^{\mathrm{T}}. \tag{26}$$

Hash functions having the similar formulations as (25) are widely used in the literature, such as SDH [12], SGH [48], and KSH [10]. Such hash functions are demonstrated to have better performance compared with linear hash functions, especially when the data is linearly inseparable [10].

C. Algorithm for Solving Problem (10)

To apply the DC programming, we also rewrite problem (10) as

$$\begin{aligned}\min_{B} \quad & G(B) - H(B) \\ \text{s.t.} \quad & B \in [-1, 1]^{c\times n}\end{aligned} \tag{27}$$

where

$$G(B) = \mathrm{tr}\left(B(D - M)B^{\mathrm{T}}\right)$$

and

$$H(B) = \gamma\left(\mathrm{tr}\left(BB^{\mathrm{T}}\right) - cn\right).$$

Here, we adopt the anchor-based affinity matrix $M = ZZ^{\mathrm{T}}$ with the truncated similarity matrix Z as stated before. To optimize (27), according to DCA, at each iteration k, we compute

$$A^k \in \partial H\left(B^k\right) = 2\gamma B^k. \tag{28}$$

Then, we compute B^{k+1} by solving the convex programming

$$\min_{B\in\mathcal{B}_1} J(B) = \mathrm{tr}\left(B(D - M)B^{\mathrm{T}}\right) - \mathrm{tr}\left(B^{\mathrm{T}}A^k\right). \tag{29}$$

We see that the above problem has the analytical solution

$$B^{k+1} = \mathcal{P}_{\mathcal{B}_1}\left[A^k * (D - M)^{-1}/2\right]. \tag{30}$$

However, the time complexity for computing $(D - M)^{-1}$ is $\mathcal{O}(n^3)$, which is unacceptable in large-scale application. Instead, we use the projected gradient descent method to solve the subproblem (29), avoiding unacceptable computational complexity. The gradient of $J(B)$ is

$$\nabla J(B) = 2B(D - M) - A^k. \tag{31}$$

```
Algorithm 3 CUH
Initialization
    Input training data $X$, code length $c$, parameter $\gamma$. Construct the matrices $M$ and $D$ using the training data and anchor points. Initialize the starting point $B^0$ by the sign of random Gaussian matrix. Set $k = 0$.
Repeat
    Compute $A^k = 2\gamma B^k$;
    Compute the gradient $\nabla J(B^k) = 2B^k(D - M) - A^k$;
    Update $B^{k+1} = \mathcal{P}_{\mathcal{B}_1}[B^k - \lambda \nabla J(B^k)]$;
    $k = k + 1$;
Until converge or reach maximum iterations.
Output $B = \text{sign}(B)$;
```

The optimization procedure is summarized in Algorithm 3.

Here, λ is a constant stepsize. At each iteration in Algorithm 3, we only need to update variable B once using the projected gradient descent.

After the binary codes are obtained, we can compute the projection matrix P using (26) to form the hash function $F(x)$ by (25).

D. Algorithm for Solving Problem (15)

As stated before, the bits uncorrelation and balance constraints are widely used in the previous hashing methods. In this section, we aim to solve problem (15) by DC programming. The term $\mathcal{L}(B)$ in (15) depends on the specific hashing model. For the supervised hashing (16), $\mathcal{L}(B) = ||Y - W^{\mathrm{T}}B||^2$, and for the unsupervised graph hashing (17), $\mathcal{L}(B) = \text{tr}(B(D - M)B^{\mathrm{T}})$. We know that the key point to apply DC programming is to decompose the objective function of (15) into the form of the difference of convex functions. Note that $\varphi_1(B)$ is concave, and $\varphi_2(B)$ and $\varphi_3(B)$ are convex, so we only need to decompose $\mathcal{L}(B)$ by

$$\mathcal{L}(B) = \mathcal{L}_1(B) - \mathcal{L}_2(B)$$

with convex $\mathcal{L}_1(B)$ and $\mathcal{L}_2(B)$. In the literature, most existing hashing models have convex objective functions, including the above-mentioned two hashing models. In such cases, we can directly set $\mathcal{L}_1(B) = \mathcal{L}(B)$ and $\mathcal{L}_2(B) = 0$.

Now, we rewrite problem (15) as

$$\min_{B\in\mathcal{B}_1} G(B) - H(B) \tag{32}$$

where

$$\begin{aligned} G(B) &= \mathcal{L}_1(B) + \eta_2\varphi_2(B) + \eta_3\varphi_3(B) \\ &= \mathcal{L}_1(B) + \eta_2||B\mathbf{1}|| + \eta_3\left(||BB^{\mathrm{T}}|| - n\sqrt{c}\right) \end{aligned}$$

and

$$\begin{aligned} H(B) &= \mathcal{L}_2(B) - \eta_1\varphi_1(B) \\ &= \mathcal{L}_2(B) + \eta_1\left(\text{tr}\left(BB^{\mathrm{T}}\right) - cn\right). \end{aligned}$$

Obviously, both $G(B)$ and $H(B)$ are convex. Next, we devise the algorithm based on DCA to solve problem (32). At each iteration k, we compute

$$A^k \in \partial H\left(B^k\right) = \nabla\mathcal{L}_2(B) + 2\eta_1 B^k. \tag{33}$$

```
Algorithm 4 CCH
Initialization
    Input training data $X$, code length $c$, parameters $\eta_1, \eta_2, \eta_3$. Initialize the starting point $B^0$ by the sign of random Gaussian matrix. Set $k = 0$.
Repeat
    Compute $A^k = \nabla\mathcal{L}_2(B) + 2\eta_1 B^k$;
    Compute the gradient $\nabla J(B^k)$ by (35) with $B = B_k$;
    Update $B^{k+1} = \mathcal{P}_{\mathcal{B}_1}[B^k - \lambda \nabla J(B^k)]$;
    $k = k + 1$;
Until converge or reach maximum iterations.
Output $B = \text{sign}(B)$;
```

Then, we obtain B^{k+1} by solving the convex programming

$$\min_{B\in\mathcal{B}_1} J(B) = G(B) - \text{tr}\left(B^{\mathrm{T}}A^k\right). \tag{34}$$

It is not easy to find an analytical solution of problem (34). Similar to problem (29), we can solve this problem using projected gradient descent method. The gradient of $J(B)$ is computed as

$$\nabla J(B) = \nabla\mathcal{L}_1(B) + \eta_2\nabla\varphi_2(B) + \eta_3\nabla\varphi_3(B) - A^k \tag{35}$$

where

$$\begin{aligned} \nabla\varphi_2(B) &= B * \mathbf{1}\mathbf{1}^{\mathrm{T}}/||B\mathbf{1}|| \\ \nabla\varphi_3(B) &= 2BB^{\mathrm{T}}B/||BB|| \end{aligned}$$

and $\nabla\mathcal{L}_1(B)$ is determined by the specific hashing model. The optimization procedure is summarized in Algorithm 4.

After obtaining the binary codes, we compute the projection matrix P by (26) and form the hash function $F(x)$ by (25).

Remark 3: The continuous constrained hashing (CCH) algorithm is used for solving the discrete hashing problem with bits uncorrelation and balance constraints, that is, problem (11). Problem (11) is a general model and CCH is a general algorithm framework. The CCH algorithm can be used for solving both the supervised and unsupervised hashing problems. For clarity, in this paper, we respectively, use CCH-S and CCH-U to represent the CCH algorithm for solving the supervised hashing problem (16) and the unsupervised hashing problem (17).

IV. Experiments

In this section, we use numerical experiments to show the performance of the proposed algorithms. We evaluate our method on three benchmark datasets: 1) CIFAR-10 [12]; 2) MNIST [34]; and 3) NUS-WIDE [5]. For each dataset, we randomly select 1000 data points from the training data as the anchor points. The baselines for comparison include the supervised hashing: SDH [12], FSDH [33], COSDISH [30], CCA-ITQ [6], KSH [10], and LFH [39]; and the unsupervised hashing: IMH [32], AGH [5] SGH [48], OCH [36], and PCA-ITQ [6]. For each method, we use the same parameters setting provided by the corresponding papers. We use CCH-U and CCH-S to, respectively, denote the CCH algorithm in the unsupervised and supervised cases as stated before. In the following

TABLE I
MAP AND PRECISION OF TOP 500 RETURNS ON CIFAR-10

Methods	MAP						Precision@500					
	8bits	16bits	32bits	64its	128bits	256bits	8bits	16bits	32bits	64its	128bits	256bits
KSH	0.2252	0.2257	0.3023	0.3234	0.3211	0.3492	0.3024	0.3220	0.3948	0.4237	0.4391	0.4428
SDH	0.5819	0.6163	0.6226	0.6462	0.6751	0.6774	0.5216	**0.5565**	0.5704	0.5773	0.5935	0.6047
FSDH	0.4551	0.6028	0.6214	0.6346	0.6624	0.6630	0.4166	0.5493	0.5788	0.5845	0.5875	0.6034
COSDISH	0.5200	0.5805	0.6219	0.6468	0.6563	0.6595	0.4451	0.5122	0.5467	0.5676	0.5756	0.5774
CCA-ITQ	0.2939	0.3504	0.3742	0.3909	0.4003	0.4036	0.3943	0.4434	0.4732	0.4891	0.4985	0.5021
LFH	0.3114	0.4133	0.5495	0.6259	0.6258	0.6484	0.2557	0.3643	0.4694	0.5413	0.5391	0.5663
CSH	**0.5846**	**0.6270**	**0.6590**	**0.6688**	**0.6899**	**0.6944**	**0.5220**	0.5460	**0.5983**	**0.5984**	**0.6139**	**0.6221**
CCH-S	**0.6062**	**0.6286**	**0.6624**	**0.6810**	**0.6954**	**0.6990**	**0.5427**	**0.5613**	**0.6045**	**0.6027**	**0.6155**	**0.6240**
IMH	0.1480	0.1566	0.1785	0.1826	0.1840	0.1877	0.1934	0.2136	0.2226	0.2349	0.2456	0.2436
AGH	0.1397	0.1468	0.1419	0.1410	0.1372	0.1320	0.1910	0.2287	0.2385	0.2644	0.2667	0.2589
SGH	0.1393	0.1473	0.1611	0.1677	0.1786	0.1824	0.1877	0.2226	0.2391	0.2533	0.2734	0.2830
PCA-ITQ	0.1490	0.1559	0.1647	0.1683	0.1769	0.1817	0.1916	0.2229	0.2395	0.2633	**0.2851**	0.2928
OCH	0.1417	0.1491	0.1645	0.1699	0.1765	0.1792	0.1807	0.2076	0.2419	0.2614	0.2746	0.2832
CUH	**0.1522**	**0.1655**	**0.1792**	**0.1874**	**0.1992**	**0.2021**	**0.1976**	**0.2316**	**0.2429**	**0.2724**	0.2772	**0.2946**
CCH-U	**0.1526**	**0.1660**	**0.1874**	**0.1902**	**0.1996**	**0.2047**	**0.2013**	**0.2368**	**0.2552**	**0.2744**	0.2832	**0.2977**

TABLE II
TRAINING TIME WITH DIFFERENT CODE SIZES ON CIFAR-10

Methods	Training time (s)					
	8bits	16bits	32bits	64its	128bits	256bits
KSH	161.2506	317.6609	892.5041	1.6621e+03	2.7033e+03	5.5102e+03
SDH	6.0604	14.4796	40.2144	62.2184	143.8140	357.4322
FSDH	6.9068	8.0636	10.2519	11.1950	13.0693	14.0730
COSDISH	4.4520	9.6953	24.1832	94.6709	400.5899	2.0872e+03
CCA-ITQ	1.2256	1.4973	6.2665	7.9809	8.8673	13.6286
LFH	1.5904	2.4527	4.4915	8.2955	16.4675	33.6897
CSH	3.4402	4.7697	5.6417	6.4909	7.8666	12.2046
CCH-S	4.2274	5.4568	5.7529	7.2741	10.8102	17.2031
IMH	14.0779	16.0084	16.6630	17.7004	21.5291	28.9154
AGH	20.4192	21.3838	21.1865	21.2277	22.1057	22.6049
SGH	8.4745	11.5111	33.3866	65.2935	117.2659	256.3173
PCA-ITQ	0.9253	1.4337	2.7473	3.5643	4.4034	6.6472
OCH	3.1133	8.7909	9.2521	10.5146	17.0130	23.5606
CUH	0.9626	1.7781	2.6453	4.6081	8.5370	15.4904
CCH-U	0.4559	0.9900	2.8481	5.2874	9.8194	16.7856

examples, we empirically set $\lambda = 5, \gamma = 1, \eta_1 = 2, \eta_2 = 10$, and $\eta_3 = 10$, and we set the maximum iteration number $t = 20$.

All the results below are averaged over 50 independent runs.

A. CIFAR-10: Test on Tiny Natural Images

In this example, we choose the dataset CIFAR-10 to test the performance of the proposed method. CIFAR-10 consists of 60K labeled images evenly divided in ten classes. Each image is represented by a 512-D GIST descriptor extracted from a 32×32 color image. We randomly select 1000 points as queries and use the rest as the training set. Since each data point from CIFAR-10 has the label information, both supervised and unsupervised hashing algorithms can be applied. The ground truth neighbors of each query are determined based on the label agreement.

For evaluation, we report the results in terms of mean average precision (MAP) and mean precision of the top 500 retrieved neighbors based on Hamming ranking in Table I with the code length varying from 8 bits to 256 bits.

From Table I, we see that the proposed continuous supervised hashing (CSH), continuous unsupervised hashing (CUH), and CCH algorithms outperform the other hashing methods in most cases. We also see that CCH-S and CCH-U, respectively, achieve better results than CSH and CUH, which implies that the bits uncorrelation and balance constraints can improve the quality of the binary codes.

We further report the training time consumption of the hashing methods in Table II. The results show that our algorithms are time efficient. The training time of the proposed methods is much less than that of KSH, SDH, SGH, and COSDISH, and is close to that of IMH, AGH, PCA-ITQ, OCH, CCA-ITQ, LFH, and FSDH.

Next, we investigate the sensitivity of the parameters η_2 and η_3. Fig. 1 shows the MAP results of CCH-U and CCH-S on CIFAR-10 with 64 bits with respect to different values of η_2 and η_3. We can see that the moderate η_2, η_3 can help to improve the retrieval accuracy, while the large η_2, η_3 will reduce the retrieval accuracy on the contrary, which demonstrates that excessively emphasizing the two constraints will reduce the quality of the solution.

The parameter γ in CSH and CUH and the parameter η_1 in CCH-S and CCH-U mainly affect the quantization error. We plot the curves of the quantization error with respect to the parameters γ, η_1 with 64 bits in Fig. 2. From the figure, we see

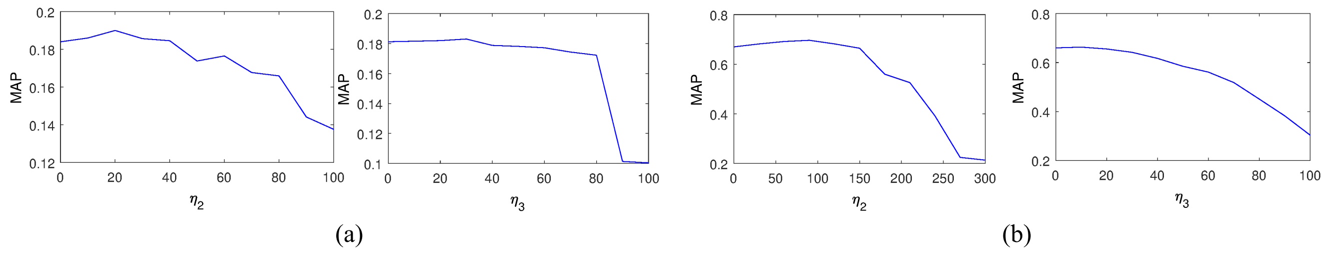

Fig. 1. MAP results with respect to parameters η_2, η_3 with 64 bits on CIFAR-10. (a) CCH-U. (b) CCH-S.

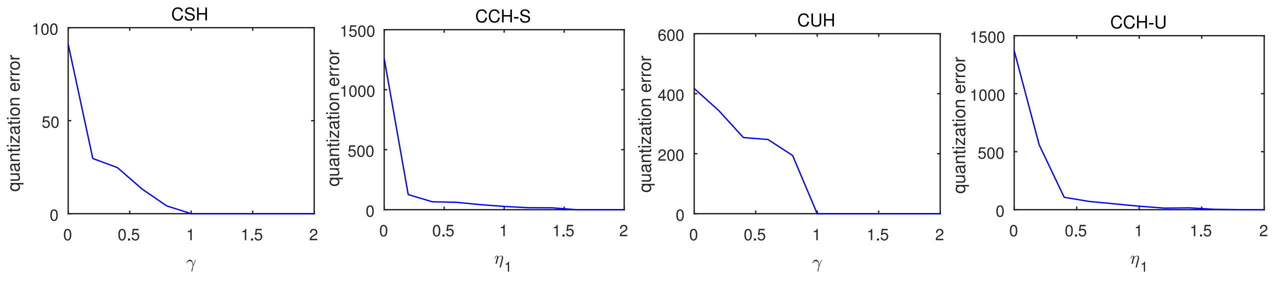

Fig. 2. Impact of γ and η_1 on the quantization error with 64 bits.

TABLE III
ACCURACY ON MNIST UNDER CLASS-WISE SPLITTING PROTOCOL

Methods	MAP						Precision@500					
	8bits	16bits	32bits	64its	128bits	256bits	8bits	16bits	32bits	64its	128bits	256bits
KSH	0.5082	0.5341	0.5815	0.6116	0.6027	0.6176	0.5973	0.6308	0.6528	0.6668	0.6818	0.7443
SDH	0.4624	0.5590	0.5310	0.5302	0.5427	0.5402	0.3444	0.5314	0.5310	0.5429	0.5685	0.6342
FSDH	0.4766	0.5076	0.5302	0.5399	0.5503	0.5463	0.3276	0.4632	0.5418	0.5663	0.5836	0.5823
COSDISH	0.4278	0.4691	0.4852	0.4927	0.5119	0.5003	0.4483	0.5366	0.5865	0.5745	0.6099	0.6114
CCA-ITQ	0.5824	0.6027	0.6171	**0.6297**	**0.6338**	0.6338	**0.6902**	0.7385	0.7617	0.7646	0.7768	0.7783
LFH	0.4506	0.4204	0.5102	0.5091	0.4951	0.4989	0.4306	0.4932	0.6529	0.6635	0.6852	0.6807
CSH	**0.5990**	**0.6615**	**0.6577**	0.6283	0.6295	**0.6342**	0.6482	**0.7506**	**0.7841**	**0.7829**	**0.8000**	**0.7940**
CCH-S	**0.6028**	**0.6601**	**0.6624**	**0.6410**	**0.6354**	**0.6390**	**0.6954**	**0.7537**	**0.7812**	**0.7811**	**0.7818**	**0.8142**

that moderate γ, η_1 are enough to guarantee the quantization error to be very small, and when $\gamma = 1$ and $\eta_1 = 2$, the quantization error is very close to zero. The results shown in the figure are in line with the statement in Remark 1.

B. MNIST: Evaluation Using the Class-Wise Splitting Protocol [52]

Recently, a class-wise splitting protocol [52] was proposed for evaluating supervised hashing, and it is demonstrated that the class-wise splitting protocol can better capture desirable properties of supervised hashing schemes, compared with the traditional protocols. So, in this example, we use this protocol to further test the performance of the proposed method on supervised hashing. The dataset for test is MNIST, which consists of 70 000 images of handwritten digits from "0" to "9." Each of the images is represented by a 784-D vector. The whole dataset is partitioned into a test set with 1000 samples and a training set with all remaining samples. The class-wise splitting protocol requires the hashing method to use disjoint set of classes for training and testing. Like [52], we separate the training set based on class splits into two disjoint sets, namely train70 and train30, where train70 contains 70% of the classes and train30 contains the remaining 30% of the classes. Also, we separate the testing set into test70 and test30 using the same method, and the classes in test70 are the same as those in train70. In this example, the set train70 is used for training. The set train30 is the database for retrieval and the set test30 is the query set. The set test70 is not used. We show the results in terms of MAP and the mean precision of the top 500 retrieved neighbors in Table III. The results show that the proposed methods outperform the other supervised hashing methods for comparison in most cases in terms of the class-wise splitting protocol.

C. NUS-WIDE: Accuracy on Large-Scale Dataset

In this example, we test the performance of the proposed method on the large-scale dataset NUS-WIDE, which consists of around 270 000 Web images associated with 81 ground truth concept labels and each image contains multiple semantic labels. Each image is represented by a 500-D bag-of-words feature. Any two images sharing at least one label are seen as true neighbors. We collect 21 most frequent labels and randomly select 100 images as queries for each label. We use the remaining images as the training set. Note that it is hard to separate the dataset based on class splits without overlap since each sample in NUS-WIDE belongs to several classes [54]. So, in this example, we use the same traditional evaluation protocol as Example 1. Since the dataset of this

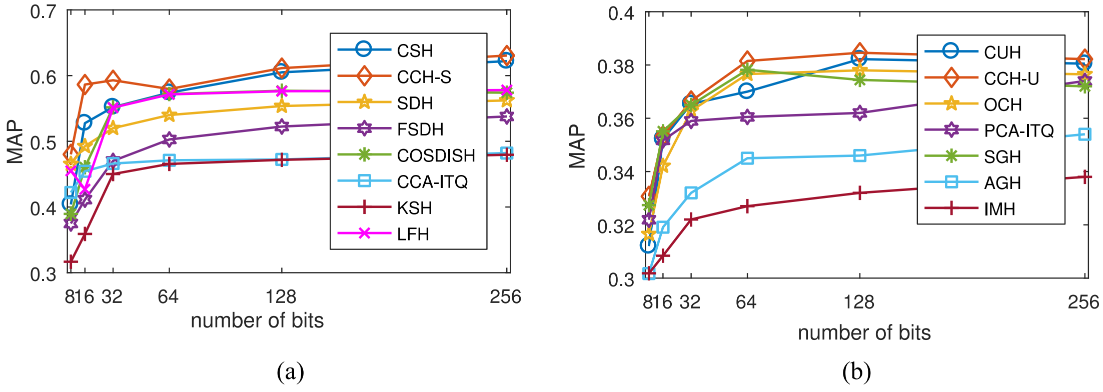

Fig. 3. Compared top-1000 MAP results on NUS-WIDE with code lengths from 32 to 256 bits. (a) Supervised hashing and (b) Unsupervised hashing.

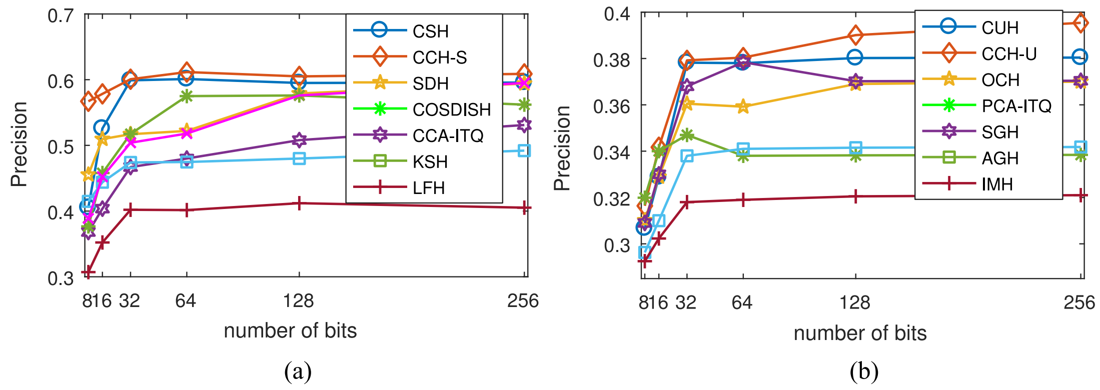

Fig. 4. Precision of top 500 returns on NUS-WIDE with code lengths from 32 to 256 bits. (a) Supervised hashing and (b) unsupervised hashing.

example is large-scale and it is very slow to compute MAP of the whole dataset, we only return top 1000 retrieved samples and calculate the MAP of the top 1000 returns. The results in terms of top-1000 MAP and mean precision of top 500 retrieved neighbors are, respectively, shown in Figs. 3 and 4. The results shown in the figures demonstrate the superiorities of the proposed methods.

V. Conclusion

In this paper, we gave an equivalent continuous formulation of the general discrete hashing problem. Specifically, we showed that the discrete hashing problem can be transformed into a continuous optimization problem without any relaxations, and the continuous optimization problem has the same optimal solutions and optimal value as the original problem. After transformation, we devised algorithms based on the idea of DC programming and DCA to solve the continuous optimization problem. We showed that such a scheme can be applied to both supervised and unsupervised hashing models. Experiments showed the superiority of the proposed method in terms of accuracy and time efficiency.

References

[1] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[2] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2143–2157, Dec. 2009.

[3] L. Liu, M. Yu, and L. Shao, "Multiview alignment hashing for efficient image search," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 956–966, Mar. 2015.

[4] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[5] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.

[6] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[7] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 21, 2009, pp. 1753–1760.

[8] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 907–919, Feb. 2016.

[9] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. Shen, "Sparse hashing for fast multimedia search," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, pp. 1–9, 2013.

[10] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2074–2081.

[11] J. Song *et al.*, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.

[12] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 37–45.

[13] Y. Luo *et al.*, "Robust discrete code modeling for supervised hashing," *Pattern Recognit.*, vol. 75, pp. 128–135, Mar. 2018.

[14] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[15] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annu. ACM Symp. Comput. Geometry*, 2004, pp. 253–262.
[16] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, 2008.
[17] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.
[18] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li, "Robust discrete spectral hashing for large-scale image semantic indexing," *IEEE Trans. Big Data*, vol. 1, no. 4, pp. 162–171, Dec. 2015.
[19] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 353–360.
[20] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 219–234.
[21] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, 2015, pp. 2475–2483.
[22] Y. Liu *et al.*, "Deep self-taught hashing for image retrieval," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2822781.
[23] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[24] S. He *et al.*, "Learning binary codes with local and inner data structure," *Neurocomputing*, vol. 282, pp. 32–41, Mar. 2018.
[25] F. Shen *et al.*, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3034–3044, Dec. 2018, doi: 10.1109/TPAMI.2018.2789887.
[26] C. Leng, J. Wu, J. Cheng, X. Zhang, and H. Lu, "Hashing for distributed data," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1642–1650.
[27] S. Wang, C. Li, and H.-L. Shen, "Distributed graph hashing," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2816791.
[28] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5324–5336, Nov. 2017.
[29] W. Kong and W.-J. Li, "Isotropic hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1646–1654.
[30] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1230–1236.
[31] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," in *Proc. ACM Conf. Knowl. Disc. Data Min.*, 2012, pp. 498–506.
[32] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1562–1569.
[33] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.
[34] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.
[35] F. Shen *et al.*, "A fast optimization method for general binary code learning," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5610–5621, Dec. 2016.
[36] H. Liu, R. Ji, Y. Wu, and F. Huang, "Ordinal constrained binary code learning for nearest neighbor search," in *Proc. AAAI*, 2017, pp. 2238–2244.
[37] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 1, 2014, pp. 2156–2162.
[38] Y. Weiss, R. Fergus, and A. Torralba, "Multidimensional spectral hashing," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 340–353.
[39] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, "Supervised hashing with latent factor models," in *Proc. ACM Conf. Inf. Retriev.*, 2014, pp. 173–182.
[40] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods," *Math. Programming*, vol. 137, nos. 1–2, pp. 91–129, 2013.
[41] H. A. L. Thi, T. P. Dinh, and H. V. Ngai, "Exact penalty and error bounds in DC programming," *J. Glob. Optim*, vol. 52, no. 3, pp. 509–535, 2012.
[42] L. T. H. An and P. D. Tao, "The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems," *Ann. Oper. Res.*, vol. 133, nos. 1–4, pp. 23–46, 2005.
[43] T. P. Dinh and H. A. L. Thi, "Convex analysis approach to DC programming: Theory, algorithms and applications," *Acta Math. Vietnamica*, vol. 22, no. 1, pp. 289–355, 1997.
[44] L. T. H. An and P. D. Tao, "Large-scale molecular optimization from distance matrices by a D.C. optimization approach," *SIAM J. Optim.*, vol. 14, no. 1, pp. 77–114, 2003.
[45] P. D. Tao and L. T. H. An, "A D.C. optimization algorithm for solving the trust-region subproblem," *SIAM J. Optim.*, vol. 8, no. 2, pp. 476–505, 1998.
[46] R. Horst and N. V. Thoai, "DC programming: Overview," *J. Optim. Theory Appl.*, vol. 103, no. 1, pp. 1–43, 1999.
[47] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, "Hashing with angular reconstructive embeddings," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 545–555, Feb. 2018.
[48] Q.-Y. Jiang and W.-J. Li, "Scalable graph hashing with feature transformation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 2248–2254.
[49] L. Zhu, J. Shen, L. Xie, and Z. Cheng, "Unsupervised topic hypergraph hashing for efficient mobile image retrieval," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3941–3954, Nov. 2017.
[50] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2548–2558, Nov. 2016.
[51] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. 23rd Adv. Neural Inf. Process. Syst.*, vol. 22, 2009, pp. 1042–1050.
[52] A. Sablayrolles, M. Douze, H. Jégou, and N. Usunier, "How should we evaluate supervised hashing?" in *Proc. IEEE Conf. Acoust. Speech Signal Process. (ICASSP)*, 2017, pp. 1732–1736.
[53] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, Dec. 2016, pp. 1711–1717.
[54] C. Zhang and W.-S. Zheng, "Semi-supervised multi-view discrete hashing for fast image search," *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2604–2617, Jun. 2017.

Shengnan Wang received the B.S. degree in information science and electronic engineering from Zhejiang University, Hangzhou, China, in 2014, where he is currently pursuing the Ph.D. degree with the College of Information Science and Electronic Engineering.

His current research interests include machine learning and optimization.

Chunguang Li (M'14–SM'14) received the M.S. degree in pattern recognition and intelligent systems and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China, Chengdu, China, in 2002 and 2004, respectively.

He is currently a Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His current research interests include statistical signal processing, machine learning, and quantitative finance.

Hui-Liang Shen received the B.Eng. and Ph.D. degrees in electronic engineering from Zhejiang University, Hangzhou, China, in 1996 and 2002, respectively.

He was a Research Associate and a Research Fellow with the Hong Kong Polytechnic University, Hong Kong, from 2001 to 2005. He is currently a Professor with the College of Information Science and Electronic Engineering, Zhejiang University. His current research interests include multispectral color imaging, image processing, and 3-D computer vision.