Hash Bit Selection for Nearest Neighbor Search

Xianglong Liu, *Member, IEEE*, Junfeng He, *Member, IEEE*, and Shih-Fu Chang, *Fellow, IEEE*

***Abstract*— To overcome the barrier of storage and computation when dealing with gigantic-scale data sets, compact hashing has been studied extensively to approximate the nearest neighbor search. Despite the recent advances, critical design issues remain open in how to select the right features, hashing algorithms, and/or parameter settings. In this paper, we address these by posing an optimal hash bit selection problem, in which an optimal subset of hash bits are selected from a pool of candidate bits generated by different features, algorithms, or parameters. Inspired by the optimization criteria used in existing hashing algorithms, we adopt the bit reliability and their complementarity as the selection criteria that can be carefully tailored for hashing performance in different tasks. Then, the bit selection solution is discovered by finding the best tradeoff between search accuracy and time using a modified dynamic programming method. To further reduce the computational complexity, we employ the pairwise relationship among hash bits to approximate the high-order independence property, and formulate it as an efficient quadratic programming method that is theoretically equivalent to the normalized dominant set problem in a vertex- and edge-weighted graph. Extensive large-scale experiments have been conducted under several important application scenarios of hash techniques, where our bit selection framework can achieve superior performance over both the naive selection methods and the state-of-the-art hashing algorithms, with significant accuracy gains ranging from 10% to 50%, relatively.**

***Index Terms*— Nearest neighbor search, hash bit selection, bit reliability, bit complementarity, normalized dominant set, locality-sensitive hashing.**

I. Introduction

NEAREST neighbor (NN) search is a fundamental issue in many applications such as multimedia search, stero-vision, machine learning, and biomedical pattern matching. It has attracted great attention in the past decades [1]–[11]. Though tree based NN search methods (*e.g.*, k-D tree [1]) have gained popularity in the past decades, they suffered from the severe performance degeneration in many cases for high-dimensional features (*e.g.*, SIFT-based bag-of-words feature, GIST, deep learning based feature, etc.) [12]. Instead, as one of the commonly-used approaches, hash based methods have shown nice theoretic guarantee properties and significant empirical success in many applications [6], [13]–[22].

Locality-Sensitive Hashing (LSH) [13] is one of the most well-known hash based NN search methods. To guarantee the search accuracy, the basic LSH method tries to embed similar data in original similarity metrics like l_p-norm ($p \in (0, 2]$) into similar codes by thresholding the random projections [14]. However, since the projections are generated independently and randomly, usually long hash codes are required to meet the desired performance, which increase computation and memory consumption in practice. To generate compact, yet informative binary codes, various types of hashing algorithms have been further proposed following LSH [9], [17]–[20], [23]–[26], [26]–[33].

Though the past decade has witnessed the rapid development of hashing research, however, designing a hashing algorithm for specific scenarios still requires lots of efforts. Even tailoring a hashing algorithm for different datasets usually requires significant efforts to discover the best parameter settings, partially due to the varying difficulty of the nearest neighbor search on different data [34]. Since there are a variety of hashing algorithms in hands, an obvious question is whether we can directly choose the most desirable subset of hash functions (or bits in binary form) from different sources generated by existing hashing algorithms. This is analogous to the well-known feature selection problem that aims at selecting the optimal subset of features from an existing feature pool [35]. Therefore, we similarly name such problem hash bit selection, which aims at selecting the most informative bits from a pool of hash bits.

Targeting the NN search task, we adopt the bit reliability and their complementarity as the selection criteria, motivated by the optimization approach used in existing compact hashing algorithms. These two important criteria can be carefully tailored for good hashing performance in any specific task. We then propose similarity preservation of each bit and the independence among them for the general hashing problems. Considering the overall (high-order) bit independence property, we achieve the selection of the most informative hash bits by finding the best tradeoff between search accuracy and time using a dynamic programming method with certain additional techniques to achieve speedup. To further reduce the computational complexity, we employ the pairwise relationship to approximate the high-order bit independence, and formulate the selection problem as a quadratic programming problem

Manuscript received November 9, 2015; revised June 21, 2016, September 29, 2016, and March 1, 2017; accepted March 28, 2017. Date of publication April 19, 2017; date of current version August 21, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61402026 and in part by the Foundation of State Key Laboratory of Software Development Environment under Grant 2015ZX-04. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wen Gao. *(Corresponding author: Junfeng He.)*

X. Liu is with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 10091, China (e-mail: xlliu@nlsde.buaa.edu.cn).

J. He is with Google, Mountain View, CA 94043 USA (e-mail: junfenghe@google.com).

S.-F. Chang is with the Department of Electrical Engineering, Columbia University, New York City, NY 10027 USA (e-mail: shih.fu.chang@columbia.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes the proof details of Theorem 1. Contact xlliu@nlsde.buaa.edu.cn and junfenghe@google.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2695895

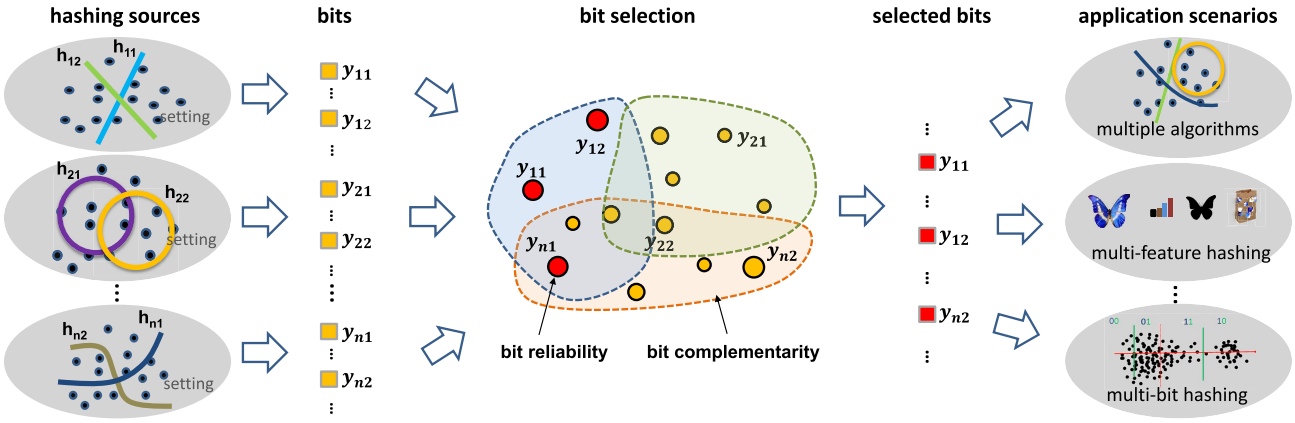


Fig. 1. The proposed unified bit selection for various hashing scenarios, where the candidate bits can be generated from multiple sources.

that can be solved efficiently using the replicator dynamics. The optimal subset is proved to be equivalent to the normalized dominant set in a vertex- and edge-weighted graph.

Figure 1 demonstrates the framework of the proposed bit selection which is generic for a wide range of scenarios and hashing sources. Specifically, it serves as a unified framework that can support various important scenarios (*e.g.*, hashing with multiple features, multiple hashing algorithms, multiple bit hashing, etc.) using different types of hashing algorithms (linear, nonlinear, multi-bit, multi-feature, etc.) with different feature spaces, parameter settings, etc., and naturally supports different search schemes including Hamming distance ranking and hash table lookup. Under this framework, we first propose two useful selection criteria: the bit reliability and complementarity. According to these selection criteria, we respectively design two selection algorithms via dynamic and quadratic programming with respect to different orders of the bit complementarity. Since the dynamic programming based solution, considering the high order correlations among bits, is quite time-consuming, the quadratic based one is further designed to speed up the computation by reducing the higher-order complementarity to the pairwise case.

The whole paper extends upon a previous conference publication [36] with additional exploration on the bit selection criteria and algorithms, detailed discussions from different point of views, and expanded experimental results. The remaining sections are organized as follows. Section III introduces bit selection framework and criteria. In Section IV we first present a straightforward selection method via dynamic programming, considering the high-order bit complementarity. To further reduce the computation, in Section V we approximate the bit complementarity based on the pairwise mutual information and thus reformulate the selection as an normalized dominant set problem that can be efficiently solved by quadratic programming. We conduct comprehensive experiments to demonstrate the superiority of the bit selection framework in Section VI.

II. Related Work

Owing to the attractive performance, in the past decade hashing technique has been widely used in a variety of applications including large-scale visual search [30], [31], machine learning [7] and recommendation [37]. The pioneering work Locality-Sensitive Hashing (LSH) hashes similar data into the similar (usually binary) codes, and achieves fast search using Hamming distance ranking or hash table lookup by exploiting the efficient binary operations. Based on the concept of LSH, a bunch of hashing studies have been proposed to learn informative hash functions that can achieve satisfying performance using compact hash codes. To pursue compact, yet informative binary codes, various types of hashing algorithms have been proposed following LSH, such as unsupervised [20], [24], [25], (semi-)supervised [17], [26], nonlinear (kernelized [26], [27], spherical [28] and clustering [19]), multiple features [29], [30], multiple labels [31], multiple bits [18], [32], [38], [39] and deep learning [40], [41].

Among these methods, the most typical solution is first proposing an optimization objective such as the neighbor dissimilarity [15], [42], [43], reconstruction error [9], [16], [41], ranking loss [44], [45], distance bias [46], and quantization error [19], [24], [25], and then designing efficient optimization algorithms that can find the desirable hash functions. In most cases, these criteria simultaneously take both the quality of hash functions and their complementarity into account [15], [17], [24], [31], [42]. As previous research show, the complementarity among hash functions is important for compact binary codes generation [15], [18], [42]. In practice, the entropy is a natural measurement of the complementarity among hash functions. However, its computation involves the joint probability distributions, which together with the discrete constraints dramatically makes the optimization of hash functions rather difficult. Therefore, research like [15], [24] and [25] employed the orthogonality on hash functions to surrogate the bit complementarity. Besides, [28] and [19] respectively forced the balanced partitions to achieve the independence between the nonlinear hash functions. The sequential manner is another alternative to achieve bit complementarity heuristically [17], [31].

Besides the emerging hashing algorithms, the bit selection is a promising solution targeting the specific scenario. In the literature, there are very few works regarding the bit selection problem. The most related work [47] greedily selects bits preserving maximum margins for specific semantics. But it

is only suitable to scenarios like semantic search, where sequentially estimating the averaged margin is computationally expensive, and the independence between bits, benefiting the compact hash codes [15], [23], is not considered explicitly. Recently, [8] proposed an a globally optimized bit selection that exploits the bit correlation based on mutual information minimization and achieved encouraging performance in fast visual search.

The bit selection finds the most desirable bits from a large pool of candidates for the nearest neighbor search task, which to some extent works as the traditional feature selection does. The feature selection finds a small subset of features to minimize redundancy and maximize relevance to the target (usually class labels in classification) [35]. In the past decades, there are many studies proposed that mostly focus on selecting relevant features that are highly relevant to the class labels [48]–[50]. By eliminating the noisy, redundant, and irrelevant features based on different relevance definition, the feature selection can faithfully improve the learning performance in the classification task.

However, from the aspect of the motivation, targeting task, and problem formulation, the bit selection is quite different from feature selection. The bit selection is proposed to deal with the problem that how to exploit the existing hashing algorithms for the nearest neighbor search in an easy and flexible way, instead of designing a new specific algorithm which requires much effort. The different motivations and the targeting tasks make the selection criteria are quite different, rather than the relevance to the class label in feature selection [49], [50]. Besides, the bit selection pursues a desired number of hash bits from the candidate binary codes, which prevents from directly employing the selection solutions in the feature selection.

III. Bit Selection

During the past decade, quite a number of hashing algorithms have been proposed for different scenarios, which can be adopted to generate a large pool of over-complete hash bits (or functions in binary hashing) for n data points $Z = \{\mathbf{z}_i, i = 1, \ldots, n\}$. Here each $\mathbf{z}_i$ is encoded by L bits generated by specified hashing algorithms with different features, parameter settings, etc. We denote the L bits of heterogeneous types by an index set $\mathcal{V} = \{1, \ldots, L\}$, and represent the i-th bits of all n points by $\mathbf{y}_i \in \{-1, 1\}^n$. The goal of bit selection is to discover a small bit subset (of size l) $\mathcal{S} \subset \mathcal{V}$, which can achieve satisfying NN search performance using short hash codes.

A. Selection Criteria

In the literature, two properties have been proven critical for compact hash codes in the task of nearest neighbor search: **bit reliability** and **bit complementarity** [15], [23]. Bit reliability considers the capability that the embedded binary codes can retain the original distances in Hamming space, and meanwhile bits complementarity measures the independence among them, which together lead to short, yet discriminative codes. Moreover, [8], [23], [51] pointed out that the bit complementarity helps to pursue large entropy among bits and thus allows fast search using hash tables. Therefore, for nearest neighbor search intuitively we prefer the bits of high bit reliability and mutually complementary in the bit section. Note that both selection criteria are general for hashing scenarios, and there are many options that can be flexibly tailored for different objectives.

1) Bit Reliability: To obtain good hash codes guaranteeing search accuracy, hashing algorithms should preserve similarities between data points, *i.e.*, similar points are supposed to share similar codes with small hamming distances [15], [23]. The similarity should be adaptively defined based on the predefined objectives, *e.g.*, for Euclidean neighbor search, we can define the similarity based on ℓ_2 distances; while for image search, similarity based on label consistency might be more appropriate.

Without loss of generality, we introduce the typical definition of the similarities $S = (s_{ij}) \in \mathbb{R}^{n \times n}$ based on ℓ_2 distances. Formally, each nonzero entry s_{ij} is the similarity between the nearest neighbors $\mathbf{z}_i$ and $\mathbf{z}_j$:

$$S_{ij} = \begin{cases} e^{-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\sigma^2}}, & \mathbf{z}_j \in \mathcal{N}(\mathbf{z}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $\mathcal{N}(\mathbf{z}_i)$ is the nearest neighbor set of $\mathbf{z}_i$.

For the high-quality hash bit, if $\mathbf{z}_j \in \mathcal{N}(\mathbf{z}_i)$, then the higher the similarity S_{ij} between $\mathbf{z}_j$ and $\mathbf{z}_i$, the larger the probability that the k-th hash bits of $\mathbf{z}_j$ and $\mathbf{z}_i$ satisfies $\mathbf{y}_{kj} = \mathbf{y}_{ki}$. Then the similarity preservation for k-th bit can be defined following the spectral embedding loss [15]:

$$\pi_k = -\sum_{ij} S_{ij} \|\mathbf{y}_{kj} - \mathbf{y}_{ki}\|^2, \tag{2}$$

To further make the weight positive and sensitive to the capability of neighbor preservation, in practice we use the exponential form with a parameter $\gamma > 0$:

$$\pi_k = e^{-\gamma \mathbf{y}_k \mathcal{L} \mathbf{y}_k^T}, \tag{3}$$

where $\mathcal{L} = \text{diag}(S\mathbf{1}) - S$ is the Laplacian matrix.

2) Bit Complementarity: Since hash-based nearest neighbor search retrieves a number of points in the hitted buckets (*i.e.*, those buckets with similar hash bits as the query) and usually further rerank them using raw features, the computational complexity will highly depend on the number of samples in the selected hash buckets. Perfectly balanced buckets will distribute samples evenly, and thus reduce the searching time on average.

Note the fact that both the worst case and the average case of time complexity in hash-based nearest neighbor search can be minimized if the buckets are perfectly balanced, when the entropy among all hash bits is maximized. Similar discussion in prior research indicated that maximizing entropy among hash bits not only provides the most compact and least redundant hash code, but also perfectly balances all the hash buckets [23]. Therefore, to achieve the desired performance with less hash bits, it requires the entropy of the selected hash bits to be maximized to guarantee the compact/irredudant hash codes. Fortunately, in the bit selection framework, all

candidate hash bits are available for the estimation of the probability distributions before computing the entropy, which largely reduces the optimization difficulty.

We represent the chosen bits by the l dimensional binary random vector $Y_{\mathcal{S}}$ corresponding to the selected bits $\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_l}$ in $\mathcal{S}$. It is not difficult to see that if the entropy $H(Y_{\mathcal{S}})$ is maximized, there would be no redundancy among selected bits and hence the bits are most compact. If all the bits $\mathbf{y}_{i_k}, i_k \in \mathcal{S}$ are independent for any choice $\mathcal{S} \subset \mathcal{V}$, we will have $H(\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_l}) = H(\mathbf{y}_{i_1}) + H(\mathbf{y}_{i_2}) + \ldots + H(\mathbf{y}_{i_l})$. However, in practice for most of the cases, the candidate hash bits are not guaranteed to be independent. Some of them may have dependency among each other. In this case, the computation of the entropy $H(Y_{\mathcal{S}})$ involves higher-order relationships among the candidate bits. For the sequential selection, the natural way to compute the entropy is applying the chain rule:

$$H(\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_k}) = H(\mathbf{y}_{i_1}) + H(\mathbf{y}_{i_2}/\mathbf{y}_{i_k}) + \ldots + H(\mathbf{y}_{i_k}/\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_k-1}).$$

Algorithm 1 Bit Selection via Dynamic Programming.

Initialize $\mathcal{S} = \emptyset$, $\mathcal{V} = \{1, \ldots, L\}$, $\mathcal{F}_{0,j} = 0$, $\mathcal{T}_{0,j} = \emptyset$, $j = 0, \ldots, l$;
for $i = 1, \ldots, L$ **do**
  for $j = 1, \ldots, l$ **do**
    Compute conditional entropy $H_{i|\mathcal{T}_{i-1,j-1}} = H(\mathbf{y}_i/Y_{\mathcal{T}_{i-1,j-1}})$;
    Compute $v_i = \pi_i + \alpha H_{i|\mathcal{T}_{i-1,j-1}}$;
    if $\mathcal{F}_{i-1,j} \geq \mathcal{F}_{i-1,j-1} + v_i$ **then**
      Update $\mathcal{F}_{i,j} = \mathcal{F}_{i-1,j}$;
      Update $\mathcal{T}_{i,j} = \mathcal{T}_{i-1,j}$;
    else
      Update $\mathcal{F}_{i,j} = \mathcal{F}_{i-1,j-1} + v_i$;
      Update $\mathcal{T}_{i,j} = \mathcal{T}_{i-1,j-1} \cup i$;
    end if
  end for
end for
Return bits corresponding to $\mathcal{S} = \mathcal{T}_{L,l}$.

IV. Bit Selection Via Dynamic Programming

In the last section, we have introduced two important criteria for hash bit selection, among which the bit reliability and complementarity correspondingly improve the quality of the hash codes from the individual and overall points of view. With the desired number of hash bits (*e.g.*, l in this paper) fixed, maximizing their entropy will impose strong complementarity on them, and thus results in discriminative and balanced hash codes. Next we will present a simple model that incorporated both selection criteria and an naive solution using dynamic programming.

A. Formulation

The bit reliability and their complementarity together can provide an appropriate tradeoff between the accuracy and the time cost of the nearest neighbor search. A good subset of hash bits should possess higher reliability and complementarity simultaneously.

Specifically, given L candidates, the problem of selecting an optimal subset of l bits for nearest neighbor search can be formulated as finding the best tradeoff between bit reliability and complementarity (meanwhile between search accuracy and search time) via an optimization framework linearly combining both selection criteria:

$$\begin{aligned} &\max_{\mathcal{S}} \ \sum\nolimits_{i \in \mathcal{S}} \pi_i + \alpha H(Y_{\mathcal{S}}) \\ &\text{s.t. } |\mathcal{S}| = l, \mathcal{S} \subseteq \mathcal{V} \end{aligned} \tag{4}$$

where $|\mathcal{S}|$ means the cardinality of set $\mathcal{S}$. $\alpha > 0$ is a parameter to control the tradeoff between the two criteria (or between search accuracy and search time).

B. Optimization

The optimization in equation (4) is quite difficult, because the term $H(Y)$ potentially involves higher-order relationships among bits. However, if all the candidate bits $\mathbf{y}_i, i = 1, \ldots, L$ are independent, we will have $H(\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_l}) = H(\mathbf{y}_{i_1}) + H(\mathbf{y}_{i_2}) + \ldots + H(\mathbf{y}_{i_l})$ for any $\mathcal{S}$. Subsequently, the problem can be rewritten as:

$$\begin{aligned} &\max_{\mathcal{S}} \ \sum\nolimits_{i \in \mathcal{S}} \pi_i + \alpha H(\mathbf{y}_i) \\ &\text{s.t. } |\mathcal{S}| = l, \mathcal{S} \subseteq \mathcal{V} \end{aligned} \tag{5}$$

which can be solved efficiently using a greedy algorithm (or forward selection procedure): we sequentially select the bit with the highest value of $\pi_i + \alpha H(\mathbf{y}_i)$ among the remaining ones, until we get l bits. The problem also can be regarded as a degenerated Knapsack problem with all item weights equal to 1 and the capacity of the bag constrained to be l.

In most cases, the candidate bits are generated by different ways, and usually contain higher-order correlations among them. Therefore, the problem (4) is very difficult to solve efficiently. Fortunately, we can decompose the computation of the bit entropy according to (4), where $H(Y_{\mathcal{S}})$ is increased by $H(\mathbf{y}_{i_k}/\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_{k-1}})$ when $\mathcal{S}$ is expanded from $\{i_1, \ldots, i_{k-1}\}$ to $\{i_1, \ldots, i_k\}$ $(k \leq l)$.

Motivated by this observation, we provide a simple approximate solution, and the maximum objective value $\mathcal{F}_{i,j}$ when selecting the j-th bit into from the subset $\mathcal{S}$ from i candidates, where $\mathcal{F}_{i,j}$ is defined as follows:

$$\begin{aligned} &(1)\ \mathcal{F}_{0,j} = 0, \quad 0 \leq j \leq l \\ &(2)\ \mathcal{F}_{i,j} = \max\{\mathcal{F}_{i-1,j}, \mathcal{F}_{i-1,j-1} + v_i\}\ell \end{aligned} \tag{6}$$

where v_i is the value of $v_i = \pi_i + \alpha H(\mathbf{y}_i/\mathcal{S})$ when adding the i-th bit into $\mathcal{S}$. This is quite similar to Knapsack problem for general cases, with the value of each added item equal to the sum of its bit reliability and the entropy increase, all item weights equal to 1, and the capacity of the bag constrained to be l. Therefore, we modify the dynamic programming algorithm of Knapsack problem, where the updating condition depends on $\mathcal{F}_{i-1,j} \geq \mathcal{F}_{i-1,j-1} + v_i$, *i.e.*, whether adding the bit will increase the objective value $\mathcal{F}$.

Algorithm 1 lists the detailed procedures as a simple approximating solution for problem (4). Unlike the general

dynamic programming, Algorithm 1 can not guarantee the global optimum anymore. However, it is shown to be very fast and effective in experiments.

C. Speedup

In Algorithm 1, the bottleneck of the time complexity mainly lies on the computation of the conditional entropy $H_{i|\mathcal{T}_{i-1,j-1}} = H(\mathbf{y}_i/Y_{\mathcal{T}_{i-1,j-1}})$, where $\mathcal{T}_{i-1,j-1}$ denotes the selected bit subset containing $j-1$ selected bits from $i-1$ candidate ones. This operation would take very long time when the size $|\mathcal{T}_{i-1,j-1}|$ of the subset becomes large.

We note that for any set $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and any bit $\mathbf{y}_i$,

$$H(\mathbf{y}_i/Y_{\mathcal{S}_2}) \leq H(\mathbf{y}_i/Y_{\mathcal{S}_1}). \tag{7}$$

This motivates us to build a set $\mathcal{O}$, whose each element is a subset of $\mathcal{T}_{i-1,j-1}$, and then employ $\min_{\mathcal{Q}\in\mathcal{O}} H(\mathbf{y}_i/Y_{\mathcal{Q}})$ as a good upper bound for $H(\mathbf{y}_i/Y_{\mathcal{T}_{i-1,j-1}})$.

In practice, $\mathcal{O}$ contains a great number of subsets of $\mathcal{T}_{i-1,j-1}$. To reduce the complexity, we can only consider subsets with no more than τ elements. τ actually determines the order of the relationship among the candidate bits, and is usually set to 2 or 3 for efficiency. However, computing their entropy online is still time-consuming. We further introduce a set Θ containing all sets with no more than τ elements from $\mathcal{V}$. Therefore, $\mathcal{O} \subseteq \Theta$, *i.e.*, if any set $\mathcal{Q} \in \mathcal{O}$, then $\mathcal{Q} \in \Theta$. To speed up our algorithm using $\min_{\mathcal{Q}\in\mathcal{O}} H(\mathbf{y}_i/Y_{\mathcal{Q}})$ as a fast approximation of $H(\mathbf{y}_i/Y_{\mathcal{T}_{i-1,j-1}})$, we only need build Θ and precompute all $H(\mathbf{y}_i/Y_{\mathcal{Q}})$ for each $i \in \mathcal{V}$ and $\mathcal{Q} \in \Theta$ aforehand. At the online selection stage, $H(\mathbf{y}_i/Y_{\mathcal{Q}})$ for $\mathcal{Q} \in \mathcal{O}$ can be computed fast by looking up the precomputed values.

V. Bit Selection Via Normalized Dominant Set

We have presented a simple dynamic programming algorithm for bit selection taking both similarity preservation and complementarity into account. However, due to the high-order relationships involved in the computation of the entropy, the algorithm even with speedup techniques still consumes much time, especially when using a large τ. In this section, we will first give a pairwise approximation for the entropy computation, then present a simple efficient quadratic programming algorithm for bit selection, and finally theoretically explain the intuition with a concept named normalized dominant set.

A. Pairwise Approximation

If the entropy $H(Y_{\mathcal{S}})$ is maximized in (4), there would be no redundancy among selected bits and hence the bits are most compact. Actually note the fact that

$$H(Y_{\mathcal{S}}) = \sum_{i\in\mathcal{S}} H(\mathbf{y}_i) - I(\mathbf{y}_1, \ldots, \mathbf{y}_i, \ldots) \tag{8}$$

where $I(\mathbf{y}_1, \ldots, \mathbf{y}_i, \ldots)$ is the mutual information among the bits of $Y_{\mathcal{S}}$. The above equation indicates that maximizing $H(Y_{\mathcal{S}})$ not only requires every bit itself to be the most informative, but also makes sure there are no redundancy among $\mathbf{y}_i$, $i \in \mathcal{S}$, which would lead to the most compact bits. In the above equation, we can easily observe that if $\mathbf{y}_i$ is highly correlated with other bits in $\mathcal{S}$, $I(\mathbf{y}_1, \ldots, \mathbf{y}_i, \ldots)$ would be increased, therefore $H(Y_{\mathcal{S}})$ would be decreased. $H(y_i)$ will be maximized if the bits itself are balanced, *i.e.*, 50% chance to be +1 or -1. Since most existing hashing methods already generated balanced bits [24], [26], [52], [53], guaranteeing the informativeness of the selected bits, to maximize $H(Y_{\mathcal{S}})$ we can place more importance on the minimizing the second part characterized by the mutual information $I(\mathbf{y}_1, \ldots, \mathbf{y}_i, \ldots)$ among all candidate bits.

Considering higher-order independence among more than two hash bits hardly improves the search quality [28]. This fact will be also validated by our experimental observation in Section VI. Therefore, for efficiency we approximately measure the independence among bits using their pairwise relationships. The smaller mutual information between i-th and j-th bits indicates a larger independence between them, and thus we define the pairwise bit complementarity between i-th and j-th bits based on their mutual information $I(\mathbf{y}_i, \mathbf{y}_j)$:

$$a_{ij} = e^{-\lambda I(\mathbf{y}_i, \mathbf{y}_j)}, \tag{9}$$

where $\lambda > 0$, and an exponential form is again adopted for positive and sensitive measurement.

Note that $a_{ij} = a_{ji}$, which means $A = (a_{ij})$ serves as a symmetric complementarity matrix between all bits. Moreover since each bit is self-dependent, all the elements on the main diagonal of A are zeros.

B. Formulation

The bit selection problem mainly aims to find bits that not only preserve the similarity of data points, but also show strong uncorrelation. Formally, with the pairwise approximation to the mutual information among all candidates, we can define an affinity matrix $\hat{A}$ incorporating both bit similarity preservation and their complementarity:

$$\hat{A} = \Pi A \Pi, \tag{10}$$

where $\Pi = \mathrm{diag}(\pi)$. The affinity matrix $\hat{A}$, judging the cohesiveness between bits, should be non-negative, symmetric, and monotonic with respect to bit quality and their complementarity. Specifically, for any two bits i and j in $\mathcal{V}$, $\hat{A}_{ij} \geq 0$, $\hat{A}_{ij} = \hat{A}_{ji}$, and $\hat{A}_{ij}$ should monotonically increase with respect to π_i, π_j and a_{ij}, due to the fact that each desired bit is supposed to strongly connect to others, in terms of similarity preservation and mutual independence.

Based on the affinity matrix, the bit selection problem can be straightforwardly reformulated as a quadratic programming with binary constraints, rather than the dynamic programming based on conditional entropy in last section:

$$\begin{aligned} &\max_{\mathbf{x}} \ \frac{1}{2}\mathbf{x}^T \hat{A}\mathbf{x} \\ &\text{s.t.} \ \ \mathbf{x} \in \{0,1\}^{L\times 1}, \ |\mathbf{x}|_0 = l \end{aligned} \tag{11}$$

The objective function $\frac{1}{2}\mathbf{x}^T\hat{A}\mathbf{x}$ measures the cohesiveness among the selected bits indicated by the binary vector $\mathbf{x}$: $x_i = 1$ $(i = 1, 2, \ldots, L)$ means the i-th bit is selected.

The optimal $\mathbf{x}^*$ should maximize the cohesiveness among its corresponding bit subset of desired size l.

Here, the way to incorporate the two important properties of compact hash codes is different from the linear combination used in dynamic programming algorithm in last section. In fact, the definition consisting of symmetric production surprisingly possesses good nature behind. In Section V-D, we will disclose its physical meaning.

C. Optimization

Directly optimizing Problem (11) is quite difficult (NP hard) due to the discrete constraints on $\mathbf{x}$. Fortunately, motivated by previous research on subset selection [54], it can be approximately solved by relaxing the binary $\mathbf{x}$ to a non-negative real-valued one. Each element of $\mathbf{x}$, with a continuous value in [0, 1] instead of the discrete one in {0, 1}, characterizes the importance of the desired bit.

If we define the support of $\mathbf{x}$ as $\sigma(\mathbf{x}) = \{i \in \mathcal{V} : x_i \neq 0\}$, then the desired bit subset corresponds to the elements in the support with largest values. This turns to a quadratic programming with continuous constraints on $\mathbf{x}$:

$$\begin{aligned} &\max_{\mathbf{x}} \ \frac{1}{2}\mathbf{x}^T \hat{A}\mathbf{x} \\ &\text{s.t. } \mathbf{x} \in \Delta \end{aligned} \tag{12}$$

where

$$\Delta = \{\mathbf{x} \in \mathbb{R}^L : \mathbf{x} \geq 0 \text{ and } \mathbf{1}^T\mathbf{x} = 1\}, \tag{13}$$

To find (local) solutions of a quadratic programming problem, a straightforward and powerful way is the so-called replicator dynamics [54], arising in evolutionary game theory. In the replicator dynamics, we first initialize $\mathbf{x}(0) = L^{-1}\mathbf{1}$, and then perform the iteration in an efficient way as follows:

$$x_i(t+1) = x_i(t)\frac{(\hat{A}\mathbf{x}(t))_i}{\mathbf{x}(t)^T\hat{A}\mathbf{x}(t)}, \ i = 1, \ldots, L. \tag{14}$$

The above dynamics enjoy a sound property that the simplex Δ is invariant to the sequential updating. Moreover, it has been proven that with symmetric, nonnegative $\hat{A}$, the objective function will strictly increase, and its asymptotically stable points correspond to strict local solutions.

By solving (12) using the replicator dynamics, usually we can get l bits when $l \ll L$. However, in practice it is possible that $|\sigma(\mathbf{x}^*)| < l$ happens. In Algorithm 2 we lists the an effective strategy that sequentially solves similar problems to (12) with respect to the remaining bit set and removes the selected bits from the candidate pool. Note that the algorithm is also feasible for large-scale problems using other efficient methods like graph shift, because usually A can be very sparse.

D. Normalized Dominant Set

We have formulated the proposed bit selection as a quadratic programming problem considering the pairwise bit complementarity. Next, we present a theoretical analysis that reveals the nature of the selection.

For pairwise relations, the pooled bits can be represented by a vertex-weighted and undirected edge-weighted graph [54], [55]: $G = (\mathcal{V}, E, A, \pi)$, where $\mathcal{V} = \{1, \ldots, L\}$ is the vertex set corresponding to the L pooled bits with weights $\pi = [\pi_1, \ldots, \pi_L]^T$, characterizing the bit quality, and $E \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set with weights $A = (a_{ij})$. Each $a_{ij} : (i, j) \in E \to \mathbb{R}_+$, a positive weight corresponding to the edge between vertex i and j, reflecting the pairwise complementarity between bits.

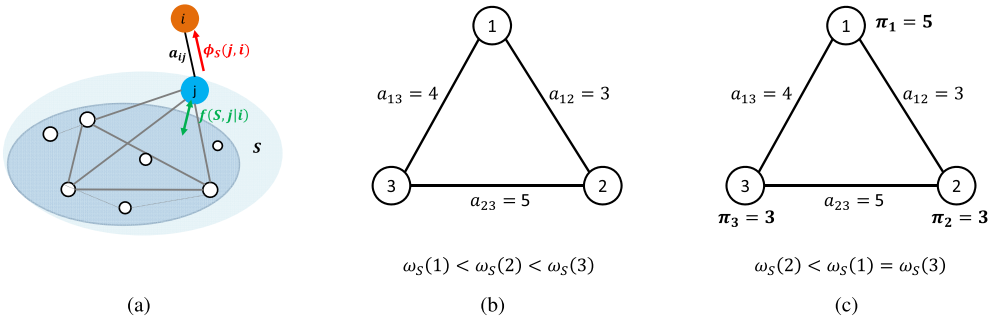
Algorithm 2 Bit Selection via Normalized Dominant Set.

```
Initialize S = ∅, V = {1, ..., L};
while |S| ≤ l do
    Find the support σ of local optima x* by solving problem (12) with respect to bits in V;
    if |S ∪ σ| ≥ l then
        Find σ̂ ⊆ σ containing l − |S| vertices with largest scores x*_σ̂;
        Terminate with S = S ∪ σ̂.
    else
        Update S = S ∪ σ, V = V \ σ.
    end if
end while
Return bits corresponding to S.
```

With the graph representation, the bit selection over the large pool is intuitively equivalent to finding a bit subset possessing high vertex and edge weights. This is similar to the popular dense subgraph discovery, and also can be regarded as a dominant set discovery problem [54]. However, since we concern the vertex- and edge-weighted graph G, in the literature there are very few work regarding the dominant set discover on such graph. Fortunately, we show that we can introduce the normalized dominant set as the dense subgraph on such graph for bit selection, and prove that the local optima of a quadratic programming defined in (12) corresponds to the normalized dominant set.

The desired bit subset should have high vertex and edge weights inside, *i.e.*, high internal homogeneity, and high inhomogeneity between vertices inside and those outside. Following the dominant set research [54], [55], we introduce a discriminative weight assignment to each vertex, invigorating both vertex and edge weights and meanwhile with respect to the homogeneity among the whole set. Based on the induced weights, we can rank all candidate bits and find the most dominant ones.

1) Relative Internal Homogeneity: Suppose we have $\mathcal{S} \subseteq \mathcal{V}$ as a nonempty subset of vertices and $j \in \mathcal{S}$. We can characterize the induced vertex weights from the predefined vertex and edge weights, by measuring the connection from any vertex $j \in \mathcal{S}$ to $i \notin \mathcal{S}$. It should take both the internal homogeneity in $\mathcal{S}$ and the vertex weights of j and i into account. We first define a function $f(\mathcal{S}, j|i)$ for the relative internal homogeneity between j and other vertices in $\mathcal{S}$ with respect to i. It should satisfy three basic properties: (1) **Non-negativity** f should be non-negative for $\mathcal{S}$, j, and i; (2) **Symmetry** f should be symmetric with respect to all vertices in $\mathcal{S}$; and (3) **Monotonicity** f should be monotonic with respect to each of its arguments.



(a) (b) (c)

Fig. 2. (a) An illustration of relative connection defined in (16). The induced vertex weight of vertex i with regard to $\mathcal{S} = \{1, 2, 3\}$ for two types of graph: (b) Edge-weighted graph [54]: $w_{\mathcal{S}}(1) = 10 < w_{\mathcal{S}}(2) = 16 < w_{\mathcal{S}}(3) = 18$; (c) Vertex-weighted Edge-weighted graph: $w_{\mathcal{S}}(2) = \frac{8}{9} < w_{\mathcal{S}}(1) = \frac{4}{3} = w_{\mathcal{S}s}(3) = \frac{4}{3}$.

Positivity and symmetry are simple consequences of the internal homogeneity definition. Monotonicity is a reasonable constraint from the following aspects: f should be monotonically increasing with respect to $\mathcal{S}$ and j, because the increase of vertex weighs and edge weights in S will lead to higher internal homogeneity between j and $\mathcal{S}$; f should be monotonically decreasing with respect to vertex i, due to the intuition that a large weight on i has strong attraction to $j \in \mathcal{S}$, and thus indicates relatively small internal homogeneity between j and $\mathcal{S}$.

With these properties, there are many choices for f, and in this paper we consider the following form:

$$f(\mathcal{S}, j|i) = \frac{\pi_i^{-1}}{\sum_{k\in\mathcal{S}} \pi_k^{-1}} \sum_{k\in\mathcal{S}} a_{jk}. \tag{15}$$

Note that when all elements of π are identical, f will degenerate to the average weight between j and $\mathcal{S}$ adopted in dominant set in [54].

2) Relative External Connection: Now we can define the connection between vertex $j \in \mathcal{S}$ and $i \notin \mathcal{S}$ motivated by the transition rate in Markov jump processes:

$$\phi_{\mathcal{S}}(j, i) = \frac{\pi_j}{\pi_i}(a_{ji} - f(\mathcal{S}, j|i)). \tag{16}$$

$\phi_{\mathcal{S}}(j, i)$ measures the relative external connection from vertex $j \in \mathcal{S}$ to $i \notin \mathcal{S}$, considering not only the vertex- and edge weights between vertex j and i, but also the internal homogeneity of j in $\mathcal{S}$.

The connection strength is determined by both $(a_{ji} - f(\mathcal{S}, j|i))$, the homogeneity between j and i eliminating the internal homogeneity of i in $\mathcal{S}$, and the vertex weight ratio $\frac{\pi_j}{\pi_i}$. If π_j is small compared to π_i, vertex j will contribute less to vertex i. See the illustrative example in Figure 2 (a). Note $\phi_{\mathcal{S}}(j, i)$ can be either positive or negative.

3) Induced Vertex Weights: Now we formalize the induced vertex weights in a recursive way:

Definition 1: Let $\mathcal{S} \subseteq \mathcal{V}$ be a nonempty vertex subset and $i \in \mathcal{S}$. The induced vertex weight of i with regard to $\mathcal{S}$ is

$$w_{\mathcal{S}}(i) = \begin{cases} \frac{1}{\pi_i^2}, & \text{if } |\mathcal{S}| = 1 \\ \sum_{j\in\mathcal{S}\setminus\{i\}} \phi_{\mathcal{S}\setminus\{i\}}(j, i) w_{\mathcal{S}\setminus\{i\}}(j), & \text{otherwise.} \end{cases}$$

The total weight of $\mathcal{S}$ is defined to be: $W(\mathcal{S}) = \sum_{i\in\mathcal{S}} w_{\mathcal{S}}(i)$. The induced vertex weight $w_{\mathcal{S}}(i)$ serves as a measure of the relative overall connections between vertex i and the remaining vertices in $\mathcal{S}$. Therefore it can be naturally regarded as a rank score for each vertex, normalized by the vertex weights in the computation.

An example comparing our induced vertex weight and that of the dominant set [54] is shown in Figure 2. As we can see, our induced vertex weights, aggregating both vertex and edge weights, alter the final rank of vertices, and assign high scores to vertices with large vertex weights.

We formally introduce the definition of the normalized dominant set in a vertex- and edge-weighted graph based on induced vertex weights:

Definition 2: A nonempty subset of vertices $\mathcal{S} \subseteq \mathcal{V}$ such that $W(\mathcal{T}) > 0$ for any nonempty $\mathcal{T} \subseteq \mathcal{S}$, is said to be normalized dominant if: (1) $w_{\mathcal{S}}(i) > 0$, for all $i \in \mathcal{S}$; (2) $w_{\mathcal{S}\cup\{i\}}(i) \leq 0$, for all $i \notin \mathcal{S}$.

In above definition, the first condition premisses the strong connections among vertices in the normalized dominant set, while the second forces external inhomogeneity.

4) Local Optima of Quadratic Programming: Now we can reveal the nature of the proposed quadratic programming in (12) by establishing its intrinsic connections with the normalized dominant set.

Theorem 1: If $\mathbf{x}^*$ is a strict local solution to Problem 12 with $\hat{A} = \Pi A \Pi$, and $\Pi = \text{diag}(\pi)$, then its support $\sigma = \sigma(\mathbf{x})$ is the normalized dominant set of graph $G = (\mathcal{V}, E, A, \pi)$, provided that $w_{\sigma\cup\{i\}}(i) \neq 0$ for all $i \notin \sigma$.

See the supplementary materials for the proof. Theorem 1 indicates that the non-zero elements of the local optima $\mathbf{x}^*$ of program (12) correspond to the normalized dominant set $\mathcal{S}$. Therefore, Algorithm 2 actually selects the normalized dominant set from the graph with the remaining bit subset in each iteration, until l bits are selected.

E. Related Point of Views

1) Game Dynamics: The above theoretical analysis indicates that discovering the most dominant bits from the pool may be formulated as a non-cooperative game, where the dominant set turns out to be informally equivalent to a classical equilibrium in evolutionary game theory.

Reference [56] pioneered evolutionary game theory by applying the principles and techniques of game theory to studying the evolution of animals, where randomly drawn pairs of individuals repeat a symmetric two-player game. The players act according to inherited behavioral patterns, instead of behaving rationally with complete knowledge of the game in traditional game theory. In evolutionary game theory the evolutionary stable strategy (essentially a Nash equilibrium) guarantees that once it is adopted by a population, the natural selection alone is sufficient to prevent alternative (mutant) strategies from invading successfully.

For our bit selection, the dominant bits discovery can be formulated as the following two-player game [57]. Given the candidate bits $\mathcal{V}$ and a matrix of affinities $\hat{A}$ between the bits in $\mathcal{V}$. Two players with complete knowledge of the game play by simultaneously selecting a bit in $\mathcal{V}$. After both have shown their choice, each player receives a payoff proportional to the affinity with respect to the elements chosen by the two players.

Clearly, within this game it is better for each player to pick a bit that is strongly supported (*i.e.*, high bit reliability and complementarity) by those that the adversary probably choose. Supposing the bits in $\mathcal{V}$ consist of two parts: a cohesive group with high mutual affinity (*i.e.*, internal homogeneity) and the remaining inhomogeneous bits that usually give inferior support to all candidates, the optimal choice for a player will be selecting elements belonging to the cohesive group rather than the inhomogeneous ones. The bit group corresponds to the evolutionary stable strategy of the non-cooperative game, namely the competition process over time will eventually drive those weakly supported bits to extinction, and while preserve the dominant ones that own strong support from compatible edges inside and competitive pressure to the others outside [58]. This informally establishes the connections between our bit selection formulation in (12) and the algorithm using replicator dynamics from the perspective of game theory.

2) Random Walk: The evolutionary replicator dynamics can be interpreted as a diffusion process [59]. We can interpret the pooled hash bits as a new graph $\hat{G} = (\mathcal{V}, E, \hat{A})$, consisting of L candidate bits in $\mathcal{V}$ as the vertices, and edges E that link vertices with the edge weights $\hat{A}$ considering both bit reliability and independence.

The diffusion process as a random walk spreads through the entire graph according to the edge weights. By rewriting the dynamics of (14) in matrix form:

$$\mathbf{x}(t+1) = \mathbf{x}(t) \odot \hat{A}\mathbf{x}(t), \tag{17}$$

and $\mathbf{x}(t+1) = \frac{\mathbf{x}(t+1)}{\|\mathbf{x}(t+1)\|_1}$, where $\odot$ is the Hadamard matrix product. The iterative process highly relates to random walk. Instead of tracking where it visited, the diffusion process mainly concerns the probability distribution $\mathbf{x}$ over all candidate bits after reaching the convergence.

The process starts with an initialization $\mathbf{x}(0) = L^{-1}\mathbf{1}$ which lies on the invariant simplex Δ under the replicator dynamics formulation in (14). Consequently, each iteration of the replicator dynamics consists of a update and a normalization part, *i.e.*, first spreading the similarities on the graph and then normalizing the newly obtained matrix to guarantee the simplex invariance.

TABLE I
COMPARISON OF DIFFERENT BIT SELECTION METHODS

	BIT RELIABILITY	BIT COMPLEMENTARITY
RANDOM	-	-
GREEDY	✓	-
DOMSET	-	2-ORDER
DYNAMIC	✓	τ-ORDER ($\tau \geq 2$)
NDOMSET	✓	2-ORDER

Here $\hat{A}$ can be regarded as a generalized transition matrix defining the probabilities for walking from one bit to neighboring ones. Although not a valid transition matrix, it encodes the same global information as the commonly used one, and is therefore also applicable for diffusion. The higher bit reliability and independence between any two bits with respect to others indicate stronger transition between them and thus larger distribution probabilities on them. This is consistent with our theoretical analysis based on the normalized dominant set.

VI. EXPERIMENTS

In our experiments, we will evaluate bit selection over several state-of-the-arts *basic hashing algorithms*. Besides, we will also investigate the performance under diverse useful scenarios: *hashing with multiple features, mixed multiple hashing algorithms, and multiple bit hashing*.

In the literature, several naive solutions in terms of metric and semantic neighbor search on several datasets can be adopted for bit selection. The straightforward method is the random way ("**Random**") without considering either of the aforementioned properties. Previous research has attempted to take the similarity preservation ("**Greedy**") into account using greedy selection method [47], [60] for specific semantics. To deal with bits correlations, the dominant set method ("**DomSet**"), which is first used to find the most dense subgraph in the literature [54], [55], can be adopted to select the most uncorrelated subset. The greedy selection method and the dominant set respectively consider similarity preservation and complementarity between bits (the edges in the graph). To simultaneously consider both bit quality and their correlations in bit selection, we propose a dynamic programming method (*i.e.*, Algorithm 1, named "**Dynamic**") and a fast, approximated quadratic programming method using normalized dominant set (*i.e.*, Algorithm 2, named "**NDomSet**"). Table I shows the comparison of different bit selection methods.

We adopt a number of state-of-the-arts hashing methods of different types in bit generation, such as Locality Sensitive Hashing (LSH) [14], Kernelized Locality Sensitive Hashing (KLSH) [27], PCA-based Hashing (PCAH) and its variation with random rotation (PCAR) [24], Iterative Quantization (ITQ) [24], Random Maximum Margin Hashing (RMMH) [61] and Spherical Hashing (SPH) [28].

- **LSH**: LSH generates Gaussian random projection vectors and preserves the locality with high probability.
- **KLSH**: KLSH constructs randomized locality-sensitive functions with arbitrary kernel functions. We feed it the Gaussian RBF kernel $k(\mathbf{z}_1, \mathbf{z}_2) = \exp(-\eta\|\mathbf{z}_1 - \mathbf{z}_2\|^2)$

TABLE II
COMPARISON OF DIFFERENT HASHING ALGORITHMS

	LINEAR/NONLINEAR	RANDOM/OPTIMIZED
LSH	LINEAR	RANDOM
PCAH	LINEAR	OPTIMIZED
PCAR	LINEAR	RANDOM+OPTIMIZED
KLSH	NONLINEAR	RANDOM
RMMH	NONLINEAR	RANDOM+OPTIMIZED
ITQ	LINEAR	OPTIMIZED
SPH	NONLINEAR	OPTIMIZED

and 300 support samples. The kernel parameter η is tuned to an appropriate value on each dataset.

- **PCAH** and **PCAR**: PCAH selects the top principal directions as the hashing projection vectors, while PCAR randomly rotates the projections along these directions. Both methods guarantee the orthogonality of the projection vectors, and meanwhile preserve the most variance of the training data.
- **ITQ**: Instead of the random rotation in PCAR, ITQ iteratively finds the data rotation in a subspace to minimize the binary quantization error.
- **RMMH**: RMMH boosts the scattering of the data by purely random splits of the data, and adopts large margin classifiers to learn hash functions with good generalization performances.
- **SPH**: This algorithm iteratively finds the balanced and independent partitions, which maps spatially coherent data points into a binary code.

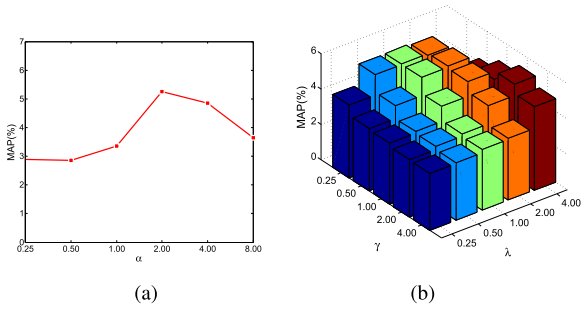
These methods covers linear/nonlinear (kernel or manifold) hashing and random/optimized hashing (see Table II). To evaluate bit selection performance for hashing with multiple features, we employ the recent multiple feature hashing (MFH) [29] and multiple feature kernel hashing (MFKH) [30] for comparison. In addition, double bit algorithms [55], [62] are employed in the scenario using multiple bit hashing.

A. Datasets and Protocol

We conduct experiments on two popular tasks including metric and semantic neighbor search. The former selects the nearest neighbors according to their distances in certain metric space (*e.g.*, Euclidean space), while the latter treats database points sharing the same semantic labels with the query as the groundtruth. We respectively employ several widely-used datasets for both tasks:

- **Metric neighbor search**: GIST-1M: A set of one million 960-D GIST descriptors and SIFT-1M: A set of one million 128-D SIFT descriptors [63].
- **Semantic neighbor search**: CIFAR-10: It contains 60K 32×32 color images of 10 classes and 6K images in each class, and NUS-WIDE: It comprises over 269K images with 81 concept tags, of which we consider 25 most frequent tags [64].

For each dataset, we respectively construct two subsets: a training set with 10,000 random samples and a testing set with 3,000 queries. On the training set, we can compute the Laplacian matrix using 100 nearest neighbors for each training sample. On GIST-1M and SIFT-1M for metric neighbor search, the groundtruth is defined by the top ‰5 nearest neighbors based on Euclidean distances. All experiments are conducted on a workstation with Intel Xeon CPU E5645@2.40GHz and 24GB memory, and the results reported in this paper are averaged over 10 runs. For parameter sensitivity, we respectively evaluate the dynamic and normalized dominant set methods on different datasets with respect to different parameters (*i.e.*, α, λ and γ) varying in $\{\frac{1}{4}, \frac{1}{2}, 1, 2, 4\}$. We investigate the parameter sensitivity over several datasets, and show that on GIST-1M in Figure 3. The experimental results indicate a similar observation that usually $\alpha = 2$, $\lambda = 2$ and $\gamma = 0.25$ can promise good performance in most cases, and therefore, we roughly share the same parameter settings in all experiments.



(a) (b)

Fig. 3. The parameter sensitivity when selecting 32 bits on GIST-1M. (a) MAP wrt α. (b) MAP wrt λ and γ.

We adopt two common search methods used in the literature: (1) Hamming distance ranking: all points in the database are ranked according to the Hamming distances from the query, and the top ranked are treated as the search results. (2) Hash table lookup: an indexing table is constructed using the binary codes, and points falling within certain Hamming radius (usually choose 2) from the query codes are returned as search results.

B. Results and Discussion

1) Bit Selection Over Basic Hashing Algorithms: Although most hashing algorithms learn hash bits according to their heuristical criteria, their solutions usually beyond the desired or optimal ones due to the problem relaxation and complicated scenarios [20]. Our bit selection serves as a generic framework that selects the most informative and complementary bits generated by different hashing algorithms for specific scenarios, which can be directly tailored by adopting different selection criteria.

We evaluate the bit selection over six basic hashing algorithms respectively on GIST-1M in terms of ℓ_2 metric (Euclidean) neighbor search. Each hashing algorithm is applied to generating 500 hash bits on GIST-1M as a bit pool. Table III compares performances of different bit selection methods using varying numbers of hash bits. It lists the mean average precision (MAP), a overall measure of the effectiveness in terms of both recall and precision.

TABLE III

MAP (%) OF BIT SELECTION OVER DIFFERENT HASHING ALGORITHMS USING 32 - 128 BITS ON GIST-1M

L =500		l =32	l =64	l =128
LSH	RANDOM	3.83±0.13	6.88±0.24	11.15±0.33
	GREEDY	3.19±0.22	5.18±0.17	8.84±0.15
	DOMSET	1.94±0.06	4.13±0.13	8.31±0.17
	DYNAMIC	4.71±0.28	7.82±0.31	11.81±0.32
	NDOMSET	**5.14**±0.11	**8.22**±0.20	**12.07**±0.19
	DYNAMIC-3	5.21	8.17	12.00
PCAR	RANDOM	4.24±0.38	6.98±0.43	11.81±0.20
	GREEDY	3.41±0.13	5.59±0.25	9.48±0.20
	DOMSET	1.97±0.15	4.21±0.24	8.87±0.27
	DYNAMIC	5.15±0.23	8.55±0.20	12.98±0.34
	NDOMSET	**5.48**±0.30	**8.93**±0.33	**13.28**±0.14
	DYNAMIC-3	**4.94**	8.04	12.51
KLSH	RANDOM	6.14±0.38	10.44±0.37	16.76±0.47
	GREEDY	3.72±0.17	7.29±0.43	13.75±0.70
	DOMSET	4.38±0.23	8.77±0.29	15.57±0.30
	DYNAMIC	6.62±0.18	11.09±0.17	17.00±0.36
	NDOMSET	**7.00**±0.19	**11.22**±0.35	**17.03**±0.57
	DYNAMIC-3	**7.03**	11.35	17.02
RMMH	RANDOM	5.81±0.26	10.75±0.25	16.53±0.43
	GREEDY	5.74±0.48	9.56±0.43	15.65±0.59
	DOMSET	3.78±0.22	7.72±0.38	13.97±0.49
	DYNAMIC	**7.19**±0.32	11.65±0.37	16.88±0.33
	NDOMSET	7.06±0.26	**11.78**±0.24	**17.56**±0.41
	DYNAMIC-3	**7.45**	11.87	17.22
ITQ	RANDOM	4.38±0.20	7.28±0.31	11.34±0.17
	GREEDY	3.74±0.14	5.96±0.24	9.37±0.24
	DOMSET	4.21±0.23	7.49±0.28	11.89±0.23
	DYNAMIC	4.94±0.20	7.88±0.19	11.66±0.16
	NDOMSET	**5.20**±0.24	**8.30**±0.17	**12.14**±0.21
	DYNAMIC-3	4.17	6.81	10.63
SPH	RANDOM	6.05±0.33	9.65±0.66	15.67±0.62
	GREEDY	4.39±0.28	9.87±0.50	16.64±0.42
	DOMSET	3.23±0.14	6.25±0.34	11.26±0.37
	DYNAMIC	7.12±0.09	11.39±0.22	16.20±0.26
	NDOMSET	**7.13**±0.24	**11.71**±0.39	**17.07**±0.38
	DYNAMIC-3	**8.01**	**12.54**	**18.10**

From the table, we first observe that the performances of all methods improve when using more bits. The baseline methods can hardly discover the best bit subset by considering either bit reliability or bit independence separately. Instead, our proposed methods, especially "NDomSet", complementarily and efficiently combine the selection criteria to enhance the compactness and informativeness of the hash codes, and subsequently achieve the best performances.

Since both LSH and RMMH generate their hash functions independently, "Random" over them can be regarded as an equivalent to the original hashing algorithms. From this point of view, by comparing to the performance of "Random" we can conclude that both the proposed "Dynamic" and "NDomSet" can achieve significant performance gains over the basic hashing algorithms (*e.g.*, 34.20% using 32 bits and 19.48% using 64 bits over LSH), especially when selecting a small number of bits. Note that the selection performance depend on the bit quality. Selecting more hash bits will increase the probability that large redundancy exists among the selected hash bits, and thus the performance gain over the basic hashing algorithm will be decreased. This is consistent with our experimental results, where the performance gain of selecting 32 or 64 bits is higher than that of selecting 128 bits in Table III.

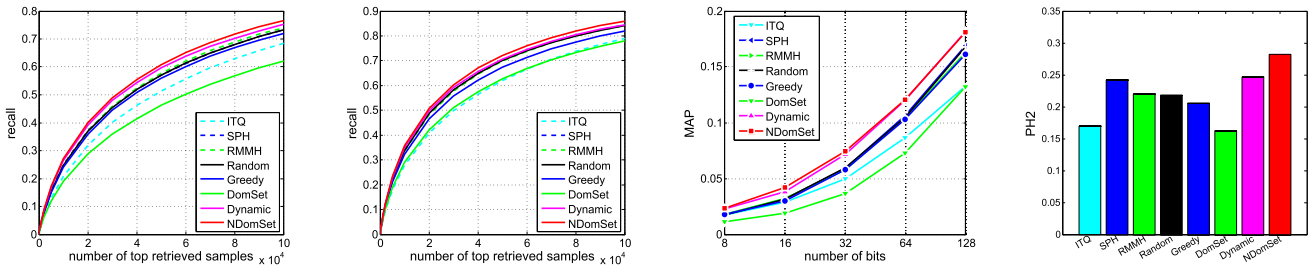
Furthermore, if we compare the selection performances over ITQ and SPH, we can find that "DomSet" performs better over ITQ than "Random", and while "Greedy" works better over SPH. This indicates that the orthogonality constraints on hash functions in ITQ are not enough for learning independent bits, while for SPH that already takes the bit relations into account when learning hash functions, its bit reliability should be further enhanced. This observation can help us find the starting point to accordingly improve the existing hashing algorithms.

2) Bit Selection Over Multiple Hashing Algorithms: With features of a low dimension it usually fails to generate long hash codes using PCA [24], [42], landmark [18] or cluster [19] based hashing algorithms. Moreover, sometimes the bit quality decreases dramatically due to the information concentration in many algorithms [15], [42]. Besides a single hashing algorithm evaluated, the proposed bit selection framework can also work well with multiple hashing algorithms. By building a large bit pool using multiple hashing algorithms, bit selection elegantly and complementarily chooses any desired number of heterogeneous bits, and hopefully achieves performance gains over existing hashing algorithms. This is quite beneficial to the practical and successful applications of existing hashing algorithms.

To evaluate the bit selection performance, in this scenario, we first employ the SIFT-1M with low-dimensional (128) SIFT features [63], with which the baselines hashing algorithms PCAH, PCAR and ITQ can only learn a very limited number (at most 128) of hash functions. When longer hash codes (say 196 in our experiment) are desired, these basic hashing algorithms cannot meet the requirement directly. However, with our bit selection framework, we can build a larger bit pool, *e.g.*, 384 bits, of which PCAH, PCAR, and ITQ respectively generate 128 ones.

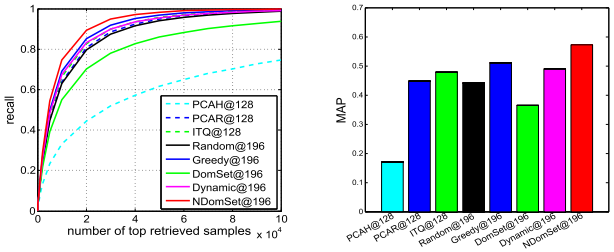
In Figure 5 we compare the performances of all selection methods picking 196 bits and the basic hashing algorithms with the largest number of bits they can learn. The results show that bit selection methods like "Greedy", "Dynamic" and "NDomSet" using 196 bits outperform the three basic hashing algorithms with its best performance, and moreover they also achieve higher recall and MAP than the other selection methods including "Random" and "DomSet" with the same settings. This observation indicates the bit quality plays a critical role in the selection over multiple sources. Note that though "Greedy" gets a close and satisfying performance to "Dynamic", but "NDomSet" obtains even more significant performance gains over all baselines, including the basic hashing algorithms ITQ, PCAR, and PCAH with the maximum number of hash bits.

To further evaluate the selection over multiple algorithms on GIST-1M, we build a large bit pool with 600 bits, of which 200 are respectively generated by ITQ, SPH and RMMH with 960-D GIST features. Then all bit selection methods are performed on this pool and compared with original hashing algorithms using its top bits on GIST-1M. In Figure 4, we report several performance statistics, *i.e.*, the recall, MAP and hamming lookup precision within radius 2 (PH2). In this case, both "Greedy" and "DomSet" are defeated by "Random",



(a) (b) (c) (d)

Fig. 4. Performance comparison of bit selection methods over multiple hashing algorithms on GIST-1M. (a) Recall @ 64 bits. (b) Recall @ 128 bits. (c) MAP @ 8-128 bits. (d) PH2 @ 32 bits.



(a) (b)

Fig. 5. Performance comparison of bit selection methods and the original hashing algorithms over multiple hashing methods on SIFT-1M. (a) Recall @ 196 bits. (b) MAP.

which indicates that neither the similarity preservation nor the bit complementarity alone is sufficient for high quality bit selection. Our bit selection methods, incorporating both criteria simultaneously, faithfully boost the performance over the basic hashing algorithms, and meanwhile surpasses all selection baselines.

3) Bit Selection With Multiple Features: Real-world data are often described by different features. Hashing with multiple features can adaptively incorporate different representations to explore informative hashing functions [29], [30]. Bit selection efficiently tackles the problem by simply picking bits produced by different hashing algorithms with different features, instead of solving the hard optimization problems involving multiple features, which can largely save the efforts on designing multi-view hashing algorithms. We evaluate the nearest neighbor search with multiple features on CIFAR-10 and NUS-WIDE, in terms of semantic neighbor search.

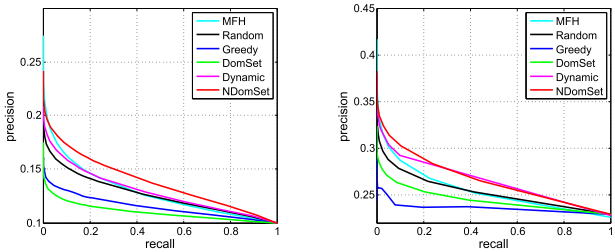
For CIFAR-10, each image is represented by a 384-D GIST feature and a 300-D SIFT BoW of 8×8 patches with a 4 space overlap; while for NUS-WIDE the provided 128-D wavelet texture and 225-D block-wise color moments are directly used as features. Then on each dataset, 250 hash bits are generated respectively for each type of feature using LSH, and mixed together as a 500 bit pool for bit selection.

Besides the naive bit selection methods, we also compare our bit selection methods with the state-of-the-art multiple feature hashing algorithms MFH [29] and MFKH [30]. MAP performance on both datasets is reported in Table IV. All bit selection methods improve their performance when using longer hash codes. But the original multiple feature hashing

TABLE IV

MAP (%) USING 32 AND 64 BITS ON CIFAR-10 AND NUS-WIDE

L =500	CIFAR-10		NUS-WIDE	
	l =32	l =64	l =32	l =64
MFH [29]	13.44±0.09	12.74±0.04	25.28±0.37	25.91±0.37
MFKH [30]	**15.10**±0.17	14.94±0.10	25.12±0.33	24.84±0.42
RANDOM	13.30±0.74	14.32±0.42	25.69±0.51	26.34±0.56
GREEDY	12.17±0.42	12.92±0.41	23.91±0.37	24.16±0.36
DOMSET	11.19±0.06	11.97±0.06	24.66±0.49	25.66±0.51
DYNAMIC	13.81±0.28	14.48±0.21	26.48±0.51	27.17±0.43
NDOMSET	14.80±0.34	**15.64**±0.35	**27.17**±0.23	**27.74**±0.16

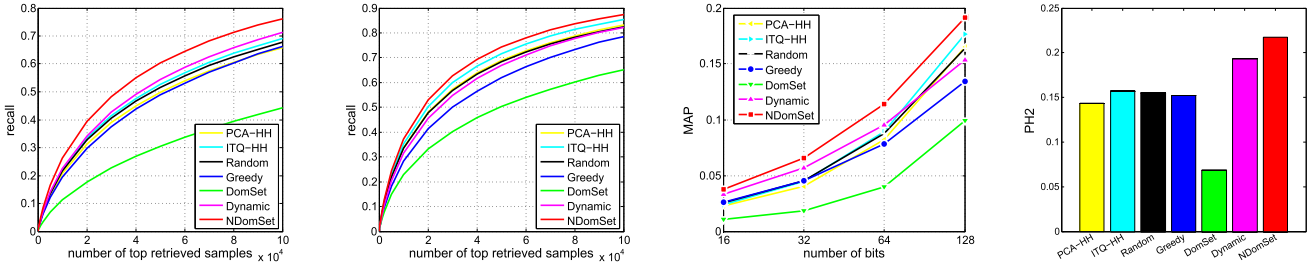


(a) (b)

Fig. 6. Semantic neighbors search performances on CIAFR-10 and NUS-WIDE using multiple features. (a) P-R @ 32 bits, CIAFR-10. (b) P-R @ 32 bits, NUS-WIDE.

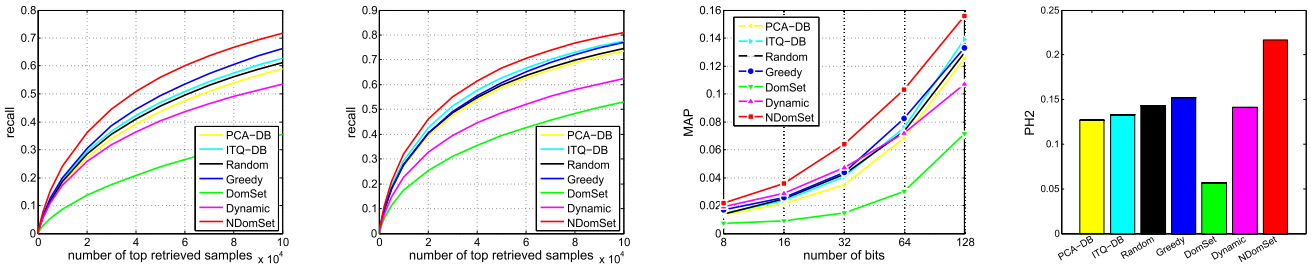
algorithms MFH and MFKH perform worse with more hash bits, partially due to the orthogonal constraints. This phenomenon has also been observed and discussed in many prior research [17], [18], which to some degree reflects the difficulty of designing hashing algorithm for specific scenarios. With the help of our bit selection framework, we can clearly conclude that both "Dynamic" and "NDomSet" can easily achieve satisfying performance over any existing hashing algorithm, and here over LSH they gain remarkable superiority, *e.g.*, "NDomSet" gets up to 22.76% performance gain on CIFAR-10 and 7.48% one on NUS-WIDE over MFH. We also show the precision-recall (P-R) curves using 32 bits on both datasets in Figure 6, where the observation that "NDomSet" owns the largest areas under its P-R curves further confirms our conclusion.

4) Bit Selection Over Multiple Bit Hashing: Most of state-of-the-art hashing algorithms generate one bit by first projecting data into a real value and then simply thresholding (or quantizing) it to a binary value. This might result in unexpected quantization loss of similarity preservation, because the threshold that typically lies in the dense region probably



(a) (b) (c) (d)

Fig. 7. Performance comparison of bit selection methods over hierarchical hashing on GIST-1M. (a) R @ 64 bits. (b) R @ 128 bits. (c) MAP. (d) PH2 @ 32 bits.



(a) (b) (c) (d)

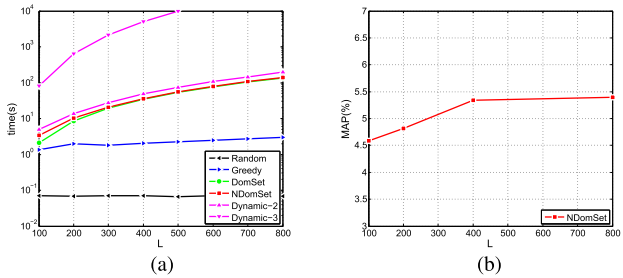
Fig. 8. Performance comparison of bit selection methods over double bit quantization on GIST-1M. (a) R @ 64 bits. (b) R @ 128 bits. (c) MAP. (d) PH2 @ 32 bits.

partitions the nearby neighbor points into different bits. Several work have attempted to alleviate the problem by generating multiple bits per projection [18], [62], [65]. Since the quality of bits per projection varies, especially for the hierarchical hashing process, it is reasonable that we can select the most informative bits for different scenarios.

To evaluate our bit selection framework over multiple bit hashing for the general nearest neighbor search, we employ the two popular multi-bit algorithms: hierarchical hashing (HH) proposed in [55] and double bit quantization (DB) proposed by [38], respectively with equal probability thresholding on PCAR (PCAR-HH and PCAR-DB) and ITQ (ITQ-HH and ITQ-DB). Each of them serves as a basic hashing source that provides a desired number of hash bits.

A 500 bit pool is first built with 250 bits generated by PCAR-HH and the rest bits by ITQ-HH on GIST-1M. Figure 7 shows the results comparing PCAR-HH, ITQ-HH and different bit selection methods over the bit pool. In Figure 7 (a)-(c) both the recall and MAP of all methods increase dramatically when using more bits, and our "Dynamic" and "NDomSet" consistently outperform both the hierarchical hashing and selection baselines in all cases. Their significant performance improvements certainly support the conclusion that our bit selection framework can help recognize the bits of high quality. Note that when selecting more hash bits (more than 64), it is can be observed that the MAP performance of "NDomSet" increases much faster than "Dynamic". This means that beside the time efficiency our quadratic formulation based on normalized dominant set also possesses better generalization ability than the dynamic programming algorithm.

Figure 8 shows the results comparing PCAR-DB, ITQ-DB and different selection methods over the 500 bit pool generated similarly on GIST-1M. Compared with greedy selection, "Random" and "DomSet" give the worst performance, which indicates that the quality of hash bits generated by DB varies widely and thus most information might be contained in a very small bit subset. In this experiment, though "Dynamic" gives a fair performance in terms of MAP and PH2, our "NDomSet" ranks first with a large margin compared to the best competitors "Greedy" and ITQ-DB in all cases, *e.g.*, up to 45.66% and 56.76% MAP gaps respectively, and 51.47% and 62.86% PH2 gaps respectively using 32 bits. Previous work report that double bit quantization improves the hashing performance [18], [38], and this observation leads to the conclusion that our bit selection can further improve performances over DB by elegantly examining the most dominant bit subset.



(a) (b)

Fig. 9. The effect of the bit pool size when selecting 32 bits on GIST-1M. (a) time cost wrt L. (b) MAP wrt L.

5) Dynamic Programming vs. Normalized Dominant Set: Figure 9(a) plots the time cost of each selection method on bit pools of different size from 100 to 8000. "Dynamic-3" takes much more time than that using second-order ("Dynamic-2") especially when working on a large bit pool. This is because that computing the entropy of more than 2 bits will bring expensive computational cost mainly due to the combinatorial explosion. Moreover, by comparing the time cost of

"NDomSet", "DomSet" and "Greedy", it is easy to conclude that estimating the mutual information between bits contributes the most to the computational cost. Even so, our "NDomSet", consuming less time compared to "Dynamic", demonstrates its efficiency and effectiveness over a number of basic hashing algorithms. Besides, we also investigate the effect of the bit pool size on the selection performance in Figure 9(b). From the figure we can see that when using a large bit pool, it is more possible that NDomSet method can select more informative hash bits, and thus can achieve better performance.

The results listed in Table III further investigate the performance of "Dynamic" with respect to different orders. In most cases, the selection methods via dynamic programming, considering bit relationships of second ("Dynamic") or third order ("Dynamic-3"), outperform all baseline methods. Moreover, "Dynamic-3" performs slightly better than "Dynamic" over certain hashing algorithms, which means that considering the high-order mutual information among bits can improve the search accuracy to some extent, yet at the expensive computational cost (see Figure 9). In all cases, we can find that our "NDomSet" is able to obtain best performance with much less computation. The experimental results under the scenarios of bit selection over multiple hashing algorithms (Figure 4 and 5), with multiple features (Figure 6 and Table IV) and over multiple bit hashing (Figure 7 and 8) further validate our conclusion that although "Dynamic" can achieve better performance than the baselines, "NDomSet" usually obtains even more significant performance gains.

VII. Conclusions

For generic hashing problems under different scenarios, this paper proposed a novel solution named hash bit selection, which supports different hashing algorithms using different features, settings, etc. Motivated by the optimization objectives used in existing compact hashing algorithms, it adopted similarity preservation of each bit and the independence among them as the selection criteria for compact, yet discriminative hash codes. We presented two related selection methods via dynamic programming and quadratic programming, incorporating bit reliability and complementarity. Based on high-order bit relationships, the former method found the tradeoff between search accuracy and efficiency with speedup techniques by linearly combining the selection criteria. By approximating the bit complementarity based on the pairwise mutual relations, the quadratic programming method further speeds up the selection through a stable distribution discovery, solved efficiently using the replicator dynamics with a natural meanings equivalent to the normalized dominant set in a vertex- and edge-weighted graph. Our extensive experiments for several important hashing scenarios proved the effectiveness and efficiency of our bit selection framework, and also demonstrated its practical meaning and future prospect in many areas.

References

[1] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[2] J. He *et al.*, "Mobile product search with bag of hash bits and boundary reranking," in *Proc. CVPR*, 2012, pp. 3005–3012.

[3] L. Chen, D. Xu, I. W.-H. Tsang, and X. Li, "Spectral embedded hashing for scalable image retrieval," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1180–1190, Jul. 2014.

[4] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashin," in *Proc. IEEE CVPR*, Jun. 2014, pp. 1–2.

[5] J. Song, Y. Yang, X. Li, Z. Huang, and Y. Yang, "Robust hashing with local models for approximate similarity search," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1225–1236, Jul. 2014.

[6] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proc. IEEE CVPR*, Jun. 2015, pp. 37–45.

[7] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Bourdev, and R. Fergus, "Web scale photo hash clustering on a single machine," in *Proc. IEEE CVPR*, Jun. 2015, pp. 19–27.

[8] L.-Y. Duan, J. Lin, Z. Wang, T. Huang, and W. Gao, "Weighted component hashing of binary aggregated descriptors for fast visual search," *IEEE Trans. Multimedia*, vol. 17, no. 6, pp. 828–842, Jun. 2015.

[9] Z. Wang, L. Duan, J. Yuan, T. Huang, and W. Gao, "To project more or to quantize more: Minimize reconstruction bias for learning compact binary codes," in *Proc. IJCAI*, 2016, pp. 2181–2188.

[10] 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.

[11] L. Liu, F. Shen, Y. Shen, X. Liu, and L. Shao, "Deep sketch hashing: Fast free-hand sketch-based image retrieval," in *Proc. IEEE CVPR*, Mar. 2017, p. 1.

[12] J. E. Goodman, J. O'Rourke, and P. Indyk, *Nearest Neighbors High-Dimensional Spaces, Handbook Discrete Computer Geometry*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2004.

[13] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. STOC*, 1998, pp. 604–613.

[14] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. SCG*, 2004, pp. 253–262.

[15] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.

[16] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. NIPS*, 2009, pp. 1042–1050.

[17] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in *Proc. ICML*, 2010, pp. 1127–1134.

[18] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

[19] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. CVPR*, Jun. 2013, pp. 2938–2945.

[20] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. NIPS*, 2014, pp. 3419–3427.

[21] Y. Xia, K. He, P. Kohli, and J. Sun, "Sparse projections for high-dimensional binary codes," in *Proc. IEEE CVPR*, Jun. 2015, pp. 3332–3339.

[22] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4514–4524, Oct. 2016.

[23] J. He, R. Radhakrishnan, S.-F. Chang, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time," in *Proc. CVPR*, Jun. 2011, pp. 753–760.

[24] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. CVPR*, Jun. 2011, pp. 817–824.

[25] M. Norouzi and D. J. Fleet, "Cartesian K-means," in *Proc. CVPR*, 2013, pp. 3017–3024.

[26] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, Jun. 2012, pp. 2074–2081.

[27] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. ICCV*, Sep. 2009, pp. 2130–2137.

[28] J. P. Heo, Y. Lee, J. He, S.-F. Chang, and S. E. Yoon, "Spherical hashing," in *Proc. CVPR*, 2012, pp. 2957–2964.

[29] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *Proc. ACM MM*, 2011, pp. 423–432.

[30] X. Liu, J. He, and B. Lang, "Multiple feature kernel hashing for large-scale visual search," *Pattern Recognit.*, vol. 47, no. 2, pp. 748–757, Feb. 2014.

[31] X. Liu, Y. Mu, B. Lang, and S.-F. Chang, "Mixed image-keyword query adaptive hashing over multilabel images," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 2, pp. 22-1–22-21, Feb. 2014.
[32] C. Deng, H. Deng, X. Liu, and Y. Yuan, "Adaptive multi-bit quantization for hashing," *Neurocomputing*, vol. 151, pp. 319–326, Mar. 2015.
[33] 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.
[34] J. He, S. Kumar, and S.-F. Chang, "On the difficulty of nearest neighbor search," in *Proc. ICML*, Edinburgh, Scotland, 2012, pp. 1127–1134.
[35] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 143–159, Jan. 2002.
[36] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: A unified solution for selection problems in hashing," in *Proc. CVPR*, 2013, pp. 1570–1577.
[37] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proc. ACM SIGIR*, 2016, pp. 1–10.
[38] Y. Lee, J. P. Heo, and S. E. Yoon, "Quadra-embedding: Binary code embedding with low quantization error," in *Proc. ACCV*, 2012, pp. 214–227.
[39] Z. Wang, L.-Y. Duan, J. Lin, X. Wang, T. Huang, and W. Gao, "Hamming compatible quantization for hashing," in *Proc. AAAI*, 2015, pp. 2298–2304.
[40] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE CVPR*, Jan. 2015, pp. 3270–3278.
[41] V. Erin 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. (CVPR)*, 2015, pp. 2475–2483.
[42] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. CVPR*, 2010, pp. 3424–3431.
[43] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.
[44] J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in *Proc. ACM MM*, 2013, pp. 133–142.
[45] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang, "Learning hash codes with listwise supervision," in *Proc. ICCV*, 2013, pp. 3032–3039.
[46] Z. Wang, L.-Y. Duan, T. Huang, and W. Gao, "Affinity preserving quantization for hashing: A vector quantization approach to learning compact binary codes," in *Proc. AAAI*, 2016, pp. 1102–1108.
[47] Y. Mu, X. Chen, X. Liu, T.-S. Chua, and S. Yan, "Multimedia semantics-aware query-adaptive hashing with bits reconfigurability," in *Proc. IJMIR*, 2012, pp. 1–12.
[48] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003.
[49] X. He, D. Cai, and P. Niyogi, *Laplacian Score for Feature Selection*. Cambridge, MA, USA: MIT Press, 2005, pp. 1–8.
[50] Y. Sun, S. Todorovic, and S. Goodison, "Local-learning-based feature selection for high-dimensional data analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1610–1626, Sep. 2010.
[51] X. Liu, C. Deng, Y. Mu, and Z. Li, "Boosting complementary hash tables for fast nearest neighbor search," in *Proc. AAAI*, 2017, 4183–4189.
[52] J. He, W. Liu, and S.-F. Chang, "Scalable similarity search with optimized kernel hashing," in *Proc. ACM SIGKDD*, 2010, pp. 1129–1138.
[53] 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.
[54] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 167–172, Jan. 2007.
[55] H. Liu and S. Yan, "Robust graph mode seeking by graph shift," in *Proc. ICML*, 2010, pp. 671–678.
[56] J. M. Smith, *Evolution Theory Games*. Cambridge, U.K.: Cambridge Univ. Press, 1982.
[57] M. Pelillo, "What is a cluster? perspectives from game theory," in *Proc. PASCAL*, 2009, p. 712.
[58] A. Torsello, S. R. Bulo, and M. Pelillo, "Grouping with asymmetric affinities: A game-theoretic perspective," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2006, pp. 292–299.
[59] M. Donoser and H. Bischof, "Diffusion processes for retrieval revisited," in *Proc. CVPR*, 2013, pp. 1320–1327.
[60] Y. Mu, X. Chen, T.-S. Chua, and S. Yan, "Learning reconfigurable hashing for diverse semantics," in *Proc. ACM ICMR*, 2011, p. 7.
[61] A. Joly and O. Buisson, "Random maximum margin hashing," in *Proc. IEEE CVPR*, Jun. 2011, pp. 873–880.
[62] W. Kong and W.-J. Li, "Double-bit quantization for hashing," in *Proc. AAAI*, 2012, p. 5.
[63] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[64] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-WIDE: A real-world Web image database from National University of Singapore," in *Proc. CIVR*, 2009, p. 48.
[65] C. Xiong, W. Chen, G. Chen, D. Johnson, and J. J. Corso, "Adaptive quantization for hashing: An information-based approach to learning binary codes," in *Proc. SDM*, 2014, pp. 172–180.

Xianglong Liu (M'12) received the B.S. and Ph.D. degrees in computer science from Beihang University, Beijing, in 2008 and 2014, respectively. From 2011 to 2012, he visited the Digital Video and Multimedia Laboratory, Columbia University, as a joint Ph.D. Student. He is currently an Associated Professor with Beihang University. His research interests include machine learning, computer vision, and multimedia information retrieval.

Junfeng He (M'08) received the B.S. and M.S. degrees from Tsinghua University and the Ph.D. degree from Columbia University. He is currently with Google, where he is involved in research topics about on-device deep learning models, HCI, and computer vision. His research interests include image/video indexing and search, computer vision, machine learning, ranking, and HCI.

Shih-Fu Chang (F'04) was the Chairman of the Columbia Electrical Engineering Department from 2007 to 2010 and an Advisor for several companies and research institutes. He has been a Senior Vice Dean of the Columbia Engineering School since 2012. He is currently the Richard Dicker Professor and the Director of the Digital Video and Multimedia Laboratory, Columbia University. He is also an active Researcher, leading development of innovative technologies for multimedia information extraction and retrieval, while contributing to fundamental advances of the fields of machine learning, computer vision, and signal processing. Recognized by many paper awards and citation impacts, his scholarly work set trends in several important areas, such as content-based visual search, compressed-domain video manipulation, image authentication, large-scale high-dimensional data indexing, and semantic video search. He co-led the ADVENT university-industry research consortium with participation of more than 25 industry sponsors. His research has been broadly supported by government agencies as well as many industry sponsors. He is a fellow of the American Association for the Advancement of Science. He received the IEEE Signal Processing Society Technical Achievement Award, the ACM SIG Multimedia Technical Achievement Award, the IEEE Kiyo Tomiyasu Award, Service Recognition Awards from the IEEE and the ACM, and the Great Teacher Award from the Society of Columbia Graduates. He served as the Editor-in-Chief of the *IEEE Signal Processing Magazine* from 2006 to 2008.