Hash Bit Selection: a Unified Solution for Selection Problems in Hashing

Xianglong Liu[†] Junfeng He[‡*] Bo Lang[†] Shih-Fu Chang[‡]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]Department of Electrical Engineering, Columbia University, New York, NY, USA
[*]Facebook, 1601 Willow Rd, Menlo Park, CA, USA
{xlliu, langbo}@nlsde.buaa.edu.cn jh2700@columbia.edu sfchang@ee.columbia.edu

Abstract

Recent years have witnessed the active development of hashing techniques for nearest neighbor search over big datasets. However, to apply hashing techniques successfully, there are several important issues remaining open in selecting features, hashing algorithms, parameter settings, kernels, etc. In this work, we unify all these selection problems into a hash bit selection framework, i.e., selecting the most informative hash bits from a pool of candidate bits generated by different types of hashing methods using different feature spaces and/or parameter settings, etc. We represent the bit pool as a vertex- and edge-weighted graph with the candidate bits as vertices. The vertex weight represents the bit quality in terms of similarity preservation, and the edge weight reflects independence (non-redundancy) between bits. Then we formulate the bit selection problem as quadratic programming on the graph, and solve it efficiently by replicator dynamics. Moreover, a theoretical study is provided to reveal a very interesting insight: the selected bits actually are the normalized dominant set of the candidate bit graph. We conducted extensive large-scale experiments for three important application scenarios of hash techniques, i.e., hashing with multiple features, multiple hashing algorithms, and multiple bit hashing. We demonstrate that our bit selection approach can achieve superior performance over both naive selection methods and state-of-the-art hashing methods under each scenario, with significant accuracy gains ranging from 10% to 50% relatively.

1. Introduction

The explosive growth of the vision data motivates the recent studies on hashing based nearest neighbor search. Locality-Sensitive Hashing (LSH) [3] as one of the most well-known methods was first proposed to establish the hashing paradigm. It produces binary codes by randomly projecting data and thresholding the projections, and can achieve fast search in constant or sub-linear time. To guarantee the performance, LSH embeds similar data in original similarity metrics like l_p-norm ($p \in (0, 2]$) into similar codes in Hamming space. However, long LSH codes are often desired to achieve a satisfactory performance, since its hashing functions are independently and randomly generated. To pursue compact, yet informative binary codes, various types of hashing methods have been proposed following LSH, such as unsupervised [4,6], (semi-)supervised [16, 20], kernelized [10], spherical [7], multiple features [18], and multiple bits [11, 14].

Despite the aforementioned progress, it still requires lots of effort to design or tailor a hashing method that can work well for each specific data set and query scenario, partially due to the varying difficulty of nearest neighbor search as revealed in the theoretical analysis in [5]. Inspired by the well-known feature selection problem that aims at selecting the optimal subset of features from an existing feature pool, an analogous question is whether we can directly choose the most desirable subset of hash bits from different bit sources, targeting the specific scenario. This brings us the so-called bit selection problem, which aims at selecting good bits from a pool of hash bits generated by different hashing algorithms with varied settings, different feature spaces, etc. The bit selection process is compatible with any hashing algorithm (linear, kernelized, spherical, multi-bit, etc.) with different parameter settings or feature types.

The bit selection serves as a unified framework for various scenarios including (but not limited to): (1) **hashing with multiple features**: since visual data are often described by different visual descriptors, hashing with multiple features can incorporate different representations to explore the informative hashing functions. Bit selection efficiently tackles the problem by picking bits produced by different hashing algorithm with different features, instead of solving the hard optimization problem involving multiple features [18]; (2) **multiple hashing methods**: with features of a low dimension it usually fails to generate long hash

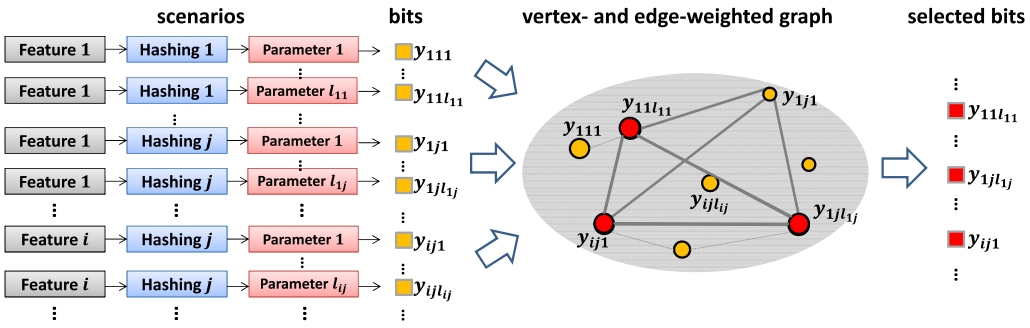


Figure 1. The proposed unified bit selection using normalized dominant set.

codes using PCA [4, 19] and landmark [14] based hashing methods. Moreover sometimes the bit quality decreases dramatically [19, 21]. By building a large bit pool using multiple hashing methods, bit selection elegantly and complementarily chooses a desired number of heterogeneous bits, and hopefully achieves major performance gains over state-of-the-art hashing methods; (3) **multiple bit hashing**: several works have attempted to improve the hashing performance by generating multiple bits per projection [11, 14]. Since the quality of bits varies, especially using the hierarchical hashing process, it is reasonable that we can select the most informative bits from them.

In the literature, not much work has been reported regarding the bit selection problem except [15], which greedily selects bits preserving maximum margins under certain metrics. However, sequentially estimating the averaged margin leads to a high computational cost. Furthermore, the independence between bits, which can benefit the compact hash codes [6, 21], is not considered explicitly.

In this paper, we propose a unified bit selection method that simultaneously considers both similarity preservation and independence between bits to guarantee good hashing performance. Our paper has the following contributions:

1. We first propose a generic bit selection that unifies various important scenarios (*e.g.*, hashing with multiple features, multiple hashing methods, multiple bit hashing, etc.) using hashing techniques. The bit selection supports different types of hashing methods using different feature spaces, parameter settings, etc.

2. We consider two important criteria carefully tailored for hashing performance, *i.e.*, similarity preservation and independence between bits, and represent the bit pool as a vertex- and edge-weighted graph. The bit selection problem turns into the discovery of the normalized dominant set in the bit graph.

3. We formulate the problem as a quadratic programming, and solve it efficiently using the replicator dynamics. A theoretical study shows the nature of the solution based on the normalized dominant set.

Figure 1 demonstrates the proposed bit selection method generic for a wide range of scenarios. Our extensive empirical study on several large-scale benchmarks highlights the benefits of our method under various useful scenarios, with significant performance gains over several naive selection methods and state-of-the-art hashing methods.

2. Bit Selection

2.1. Problem Definition

Suppose there is a large pool of over-complete hash bits for n data points $Z = \{\mathbf{z}_i, i = 1, \ldots, n\}$, where $\mathbf{z}_i$ is encoded by L bits generated by various hashing methods with different features, parameter settings, etc. Denoting the L types of bits with index set $V = \{1, \ldots, L\}$, the i-th bits ($i = 1, \ldots, L$) for all n points can be represented as $\mathbf{y}_i \in \{-1, 1\}^{1\times n}$. The goal of bit selection is to exploit a small bit subset (of size l) $\mathcal{S} \subset V$, which not only reduces the search and storage cost with short lengths, but also achieves good performance with strong discriminative power.

2.2. Selection Criteria

In the literature, two properties have been proved critical for compact hash codes: **similarity preservation** and **independence** [6, 21]. Similarity preservation means that the embedded binary codes can retain the original distances in Hamming space, while keeping bits independent avoids redundancy among them and leads to short, yet discriminative codes. Moreover, as discussed in [6] the independence also helps to achieve large entropy among bits and thus allowing for fast search speeds. Therefore intuitively bits that are not only capable of preserving similarity but also mutually independent should be selected. Note that besides the proposed selection criteria, there are other options that can be flexibly tailored for different objectives.

2.2.1 Similarity Preservation

To obtain good hash codes guaranteeing search accuracy, hashing methods should preserve similarities between data points, namely, similar points are supposed to have similar hash codes with small hamming distances [6, 21]. The similarity can be adaptively tailored for different objectives, eg., for ℓ_2 (Euclidean) similarity preservation of LSH we can define it based on ℓ_2 neighbors; while for hashing with multiple features, similarity based on label consistency might be more appropriate.

In our paper, we give a typical definition of the similarities $S \in \mathbb{R}^{n\times n}$, whose entry S_{ij} is the similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$, considering whether $\mathbf{z}_j$ belongs to the nearest neighbor set $\mathcal{N}(\mathbf{z}_i)$ of $\mathbf{z}_i$:

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

Then the similarity preservation for i-th bit, based on the spectral embedding loss [21], can be defined as:

$$\pi_i = \exp\left(-\gamma \mathbf{y}_i \mathcal{L} \mathbf{y}_i^T\right), \tag{2}$$

where the parameter $\gamma > 0$, and $\mathcal{L} = \text{diag}(S\mathbf{1}) - S$ is the Laplacian matrix.

2.2.2 Mutual Independence

Previous research shows that balance and independence of hashing functions are important for generating compact binary codes [21]. [6] argued that minimizing mutual information criterion provides the most compact and least redundant hash code. However, considering higher-order independence among more than two hash bits hardly improves the search quality [7]. Therefore, we approximately measure the independence using pair-wise relationships between hash bits.

Suppose the distribution for i-th bit is $p(b_i), b_i \in \{-1, 1\}$, and the joint distribution for i-th and j-th bits is $p(b_i, b_j)$, then the independence between i-th and j-th bits based on their mutual information is defined with a $\lambda > 0$:

$$a_{ij} = \exp\left[-\lambda \sum_{b_i, b_j} p(b_i, b_j) \log \frac{p(b_i, b_j)}{p(b_i)p(b_j)}\right]. \tag{3}$$

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

3. Formulation and Optimization

In this section, we intuitively formulate the bit selection problem as a quadratic programming with binary constraints, and then propose an efficient solution by relaxing the discrete constraints.

3.1. Formulation

We want to select bits that not only preserve the similarity of data points, but also are uncorrelated. Formally, with an affinity matrix $\hat{A}$ incorporating both bit similarity preservation and their independence, the problem of finding a good subset of size l from L bits can be formulated as:

$$\begin{array}{ll} \max & \frac{1}{2}\mathbf{x}^T \hat{A} \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \Omega \end{array} \tag{4}$$

where

$$\Omega = \{\mathbf{x} \in \{0, 1\}^L : |\mathbf{x}|_0 = l\} \tag{5}$$

In the above formulation, 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 $\mathbf{x}$, where if x_i $(i = 1, 2, \ldots, L)$ is 1, the i-th bit is selected; otherwise, it is rejected. The optimal $\mathbf{x}^*$ should maximize the cohesiveness among its corresponding bit subset of desired size l.

The affinity matrix $\hat{A}$, judging the cohesiveness between bits, should be non-negative, symmetric, and monotonic with respect to both bit similarity preservation and their independence. Specifically, for any two bits i and j in V, $\hat{A}_{ij} \geq 0$, $\hat{A}_{ij} = \hat{A}_{ji}$, and $\hat{A}_{ij}$ should be monotonically increasing with respect to π_i, π_j and a_{ij}, due to the fact that in the desired bit subset any bit strongly connects to the others in terms of similarity preservation and mutual independence. Therefore, a possible definition is

$$\hat{A} = \Pi A \Pi, \tag{6}$$

where $\Pi = \text{diag}(\pi)$. In Section 4, we will disclose the physical meaning of the definition.

3.2. Optimization

The optimization of problem (4) is quite difficult due to the discrete constraints on $\mathbf{x}$. However, motivated by previous research on subset selection [12, 17], it can be approximately solved by relaxing the binary $\mathbf{x}$ to a non-negative real-valued one. Each element of $\mathbf{x}$ expresses the importance of association with the desired bit subset. Therefore, if define the support of $\mathbf{x}$ as $\sigma(\mathbf{x}) = \{i \in V : x_i \neq 0\}$, 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 $\mathbf{x}$:

$$\begin{array}{ll} \max & \frac{1}{2}\mathbf{x}^T \hat{A} \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \Delta \end{array} \tag{7}$$

where

$$\Delta = \{\mathbf{x} \in \mathbb{R}^L : \mathbf{x} \geq 0 \text{ and } \mathbf{1}^T \mathbf{x} = 1\} \tag{8}$$

with $\hat{A}$ defined in (6).

A straightforward and powerful way to find (local) solutions of a quadratic programming problem is the so-called

Algorithm 1 Bit Selection.

Initialize $S = \emptyset$, $V = \{1, \ldots, L\}$;
while $|S| \leq l$ **do**
  Find the support σ of local optima $\mathbf{x}^*$ by solving problem (7) with respect to bits in V;
  if $|S \cup \sigma| \geq l$ **then**
    Find $\hat{\sigma} \subseteq \sigma$ containing $l - |S|$ vertices with largest scores $\mathbf{x}^*_{\hat{\sigma}}$;
    Terminate with $S = S \cup \hat{\sigma}$.
  else
    Update $S = S \cup \sigma$, $V = V \setminus \sigma$.
  end if
end while
Return bits corresponding to S.

replicator dynamics [12, 17], arising in evolutionary game theory. Given an initialization of $\mathbf{x}(0)$ (we use $\frac{1}{L}\mathbf{1}$), the iteration can be performed efficiently in the following form:

$$x_i(t+1) = x_i(t)\frac{(\hat{A}\mathbf{x}(t))_i}{\mathbf{x}(t)^T\hat{A}\mathbf{x}(t)},\ i = 1, \ldots, L. \quad (9)$$

Under these dynamics, the simplex Δ is invariant. Moreover, it has been proven that with symmetric $\hat{A}$ whose entries are nonnegative, the objective function will strictly increase, and its asymptotically stable points correspond to strict local solutions [12].

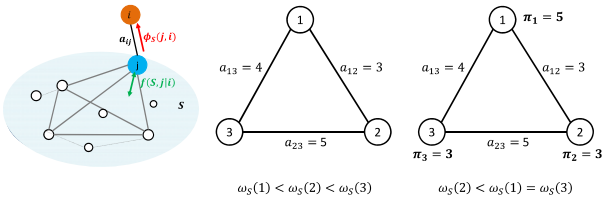
Usually we can get l bits by solving (7) once in real-world situations where $l \ll L$. However, it is possible that $|\sigma(\mathbf{x}^*)| < l$ happens. An effective strategy in Algorithm 1 is to iteratively solve similar problems to (7) with respect to the remaining bit set by removing bits already selected in the previous iteration. The algorithm is feasible for large-scale problems, since A can be sparse, and efficient methods like graph shift [12] can be adopted for fast computation.

4. Theoretic Analysis

We have presented a simple but efficient algorithm for bit selection considering both similarity preservation and independence. Next, we give a theoretic explanation of our intuition with an concept named normalized dominant set.

4.1. Graph Representation

Motivated by recent research on modelling correlations based on graph [12, 17], we first represent the pooled bits as a vertex-weighted and undirected edge-weighted graph: $G = (V, E, A, \pi)$, where $V = \{1, \ldots, L\}$ is the vertex set corresponding to the L types of pooled bits with weights $\pi = [\pi_1, \ldots, \pi_L]^T$, and $E \subseteq V \times V$ 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. The vertex and edge weights correspond to the provided selection criteria: π represents the capabilities of similarity preservation, and A reflects the independence between bits.



(a) (b) (c)

Figure 2. (a) An illustration of relative connection defined in (10). The induced vertex weight of vertex i with regard to $S = \{1, 2, 3\}$ for two types of graph: (b) Edge-weighted graph [17]: $w_S(1) = 10 < w_S(2) = 16 < w_S(3) = 18$; (c) Vertex-weighted Edge-weighted graph: $w_S(2) = \frac{8}{9} < w_S(1) = \frac{4}{3} = w_S(3) = \frac{4}{3}$.

Based on the graph representation, the bit selection can be regarded as a dense subgraph discovery in the vertex- and edge-weighted graph G. The dominant set is proposed to tackle such a problem in the edge-weighted graph [13, 17]. However, in the literature there are very few works that uncover the dense subgraph on a vertex- and edge-weighted graph. In our work, we introduce the concept of normalized dominant set on such graph for bit selection.

4.2. Normalized Dominant Set

Corresponding to the most desirable bits, the normalized dominant set should have high vertex and edge weights inside, *i.e.*, high internal homogeneity. Following prior successful research on dominant set [12, 17], we start with an idea that both vertex and edge weights will induce a discriminative assignment of weights on the vertices, with respect to the homogeneity among the whole vertex set.

Let $S \subseteq V$ be a nonempty subset of vertices and $j \in S$. We attempt to characterize the induced vertex weights by measuring the connection from any vertex $j \in S$ to $i \notin S$. It should take both the internal homogeneity in S and the vertex weights of j and i into account. Motivated by the transition rate in Markov Jump [1], we introduce the external connection from vertex $j \in S$ to $i \notin S$:

$$\phi_S(j, i) = \frac{\pi_j}{\pi_i}(a_{ji} - f(S, j|i)), \quad (10)$$

where $f(S, j|i) = \frac{\pi_i^{-1}}{\sum_{k\in S}\pi_k^{-1}}\sum_{k\in S} a_{jk}$ is the relative internal homogeneity between j and other vertices in S with respect to i. When all elements of π are identical, f will degenerate to the average weight between j and S, namely only the edge weights have effect on the homogeneity [17]. The connection strength is determined by both the external homogeneity $(a_{ji} - f(S, j|i))$ 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).

Following the idea of [17], we formalize the induced vertex weight of i with regard to S in a recursive way:

$$w_S(i) = \begin{cases} \frac{1}{\pi_i^2}, & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S\setminus\{i\}}(j,i) w_{S\setminus\{i\}}(j), & \text{otherwise.} \end{cases} \tag{11}$$

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

Based on the induced vertex weights, we define the normalized dominant set in a vertex- and edge-weighted graph, similar to the dominant set in [17]:

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

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

Figure 2 shows an example comparing our induced vertex weight and that of the dominant set [17]. 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.

4.3. Local Optima of the Quadratic Programming

We establish the intrinsic connections between the normalized dominant set and the local optima of quadratic programming in (7) by the following theorem.

Theorem 1 *If $\mathbf{x}^*$ is a strict local solution of program (7) with $\hat{A} = \Pi A \Pi$, where $\Pi = diag(\pi)$, then its support $\sigma = \sigma(\mathbf{x})$ is the normalized dominant set of graph $G = (V, E, A, \pi)$, provided that $w_{\sigma\cup\{i\}}(i) \neq 0$ for all $i \notin \sigma$.*

Here, the if-and-only-if statement holds with the definition of the weighted characteristic vector [17]. See detailed proofs in the supplementary material.

Theorem 1 tells that the non-zero elements of the local optima $\mathbf{x}^*$ of program (7) form the normalized dominant set S. Moreover, their values correspond to the induced vertex weights of vertices in S, normalized by $W(S)$. Therefore, in Algorithm 1 each iteration selects the normalized dominant set from the graph with the remaining bit subset, until l bits are selected. Note that in (2) when $\gamma \to 0$, $\hat{A} \to A$, which means that the normalized dominant set problem, corresponding to problem (7), degenerates to the dominant set problem [17].

5. Experiments

In this section we will evaluate the bit selection for diverse useful scenarios: *hashing with multiple features, mixed multiple hashing methods, and multiple bit hashing*. The proposed bit selection method (**NDomSet**) will be compared with several naive solutions in terms of metric and semantic neighbor search on several datasets. The straightforward selection 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 [15]. To deal with bits correlations, the dominant set method (**DomSet**), which is first used to find the most dense subgraph in the literature [12, 17], can be adopted to select the most uncorrelated subset. The greedy selection method and the dominant set respectively consider similarity preservation and independence between bits (the edges in the graph). As we will show in the experiments, although greedy selection and dominant set, respectively considering one property, might obtain a performance gain on certain datasets, they usually fail because of the ignorance of the other property. Based on the graph, the proposed bit selection using normalized dominant set discovers the good bit subsets by complementarily incorporating both properties.

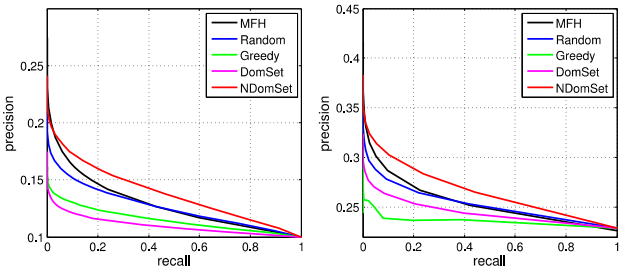
In our experiments, a number of state-of-the-arts hashing methods such as Locality Sensitive Hashing (LSH) [3], PCA-based Hashing (PCAH) and its variation with random rotation (PCAR) [4], Iterative Quantization (ITQ) [4], Spherical Hashing (SPH) [7] and Random Maximum Margin Hashing (RMMH) [9] are involved in bit generation. These methods covers linear/nonlinear hashing and random/optimized hashing. To evaluate bit selection performance for hashing with multiple features, we employ the recent multiple feature hashing (MFH) proposed in [18] for comparison. In addition, double bit methods [11, 12] are employed in the scenario using multiple bit hashing.

5.1. Datasets and Protocol

We conduct experiments on several datasets for both metric and semantic neighbor search. The former selects the nearest neighbors according to their distances in metric space, while the latter treats database points sharing the same semantic labels as the groundtruth. We give a brief description of three large-scale datasets that have been widely used in hashing research:

- **GIST-1M**: A set of one million 960-D GIST descriptors [8].
- **CIFAR-10**[1]: It contains 60K 32×32 color images of 10 classes and 6K images in each class.

[1] http://www.cs.utoronto.ca/∼kriz/cifar.html



(a) P-R @ 32 bits, CIAFR-10 (b) P-R @ 32 bits, NUS-WIDE

Figure 3. Semantic neighbors search performances on CIAFR-10 and NUS-WIDE using multiple features.

Table 1. MAP (%) using 32 and 64 bits on CIFAR-10 and NUS-WIDE.

500 BITS	CIFAR-10		NUS-WIDE	
	32	64	32	64
MFH [18]	13.44±0.09	12.74±0.04	25.28±0.37	25.91±0.37
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
NDOMSET	**14.80**±0.34	**15.64**±0.35	**27.17**±0.23	**27.74**±0.16

- **NUS-WIDE**: It comprises over 269K images with 81 concept tags, of which we consider 25 most frequent tags [2].

For each dataset, we respectively construct two subsets: a training set of 10,000 random samples and a testing set of 1,000 random samples. The training set is used to obtain the Laplacian matrix, where 100 nearest neighbors are considered for each training sample. GIST-1M is adopted for evaluation on metric neighbor search, whose groundtruth is defined by the top ‰5 nearest neighbors computed by the exhaustive linear scan based on Euclidean distances, while the remaining two datasets are for semantic neighbor search.

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 5 runs. For parameter (γ and λ) sensitivity, our experiments indicate that the proposed bit selection is relatively robust. Thus we roughly tuned the parameters on a small set, and all selection baselines share the same parameter settings.

5.2. Results and Discussion

5.2.1 Bit selection with multiple features

An attractive advantage for bit selection is that multiple features can be adopted to build the bit pool, which can save lots of effort on designing multi-view hashing algorithms. We evaluate such performance on CIFAR-10 and NUS-WIDE in terms of semantic neighbor search. In CIFAR-10, each image is represented by a 384-D GIST feature and a 300-D bag of visual words quantized from dense SIFT features of 8×8 patches with a 4 space overlap; while in NUS-WIDE the provided 128-D wavelet texture and 225-D block-wise color moments are directly used as features. On each dataset, 250 hash bits are generated respectively for each type of feature using LSH, and then these bits are mixed together to form a 500 bit pool for bit selection.

Baselines. Besides the naive bit selection methods (Random, Greedy, and DomSet), we also compare our bit selection to the state-of-the-art multiple feature hashing method **MFH** [18], which learns the compact hashing functions by individually preserving the local structure of each feature and globally considering that of all features.

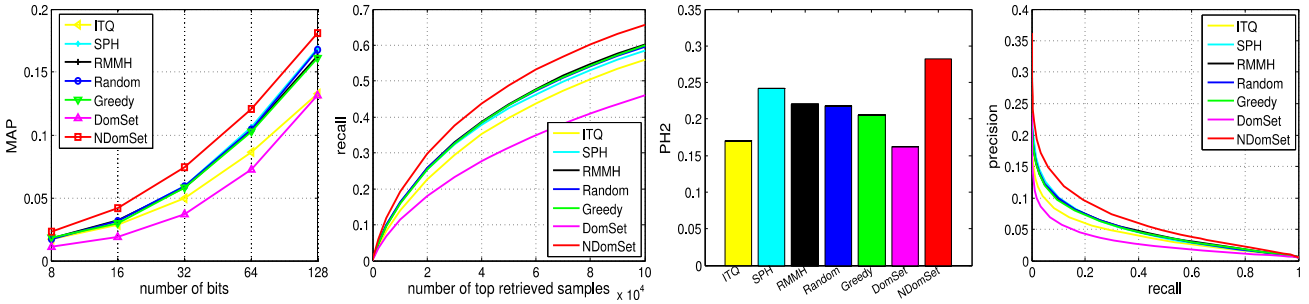
The mean average precision (MAP) of bit selection methods and MFH on both CIFAR-10 and NUS-WIDE are presented in Table 1. Performances of all methods improve when using longer hash codes except MFH, and clearly NDomSet achieves the highest performance in all cases with remarkable superiority (up to 22.76% performance gain on CIFAR-10 and 7.48% on NUS-WIDE over MFH). The MAP decrease of MFH when using more bits is partially due to the orthogonal constraint, which has been observed and discussed in [20]. We also compare precision-recall (P-R) performances using 32 bits on both datasets in Figure 3. Greedy performs better than DomSet on CIFAR-10, which means the similarity preservation is more desirable than independence for LSH bits generated from two feature types of CIFAR-10. The opposite case occurs on NUS-WIDE dataset. However, both selection methods fail to compete with NDomSet. NDomSet over LSH bits achieves better performance than both MFH and other bit selection methods, with the largest areas under the P-R curves.

5.2.2 Bit selection over multiple hashing

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. Our bit selection gives a generic framework that can gather the most informative and complementary bits generated by different methods for specific scenarios, which directly meet the provided criteria (similarity preservation and independence in our paper) .

Baselines. The learned reconfigurable hashing (**LRH**) [15] is the only published work studying bit selection problem. Similar to Greedy, it heuristically finds bits preserving maximum margins, but at the cost of high computations.

We first evaluate the bit selection over two basic hashing methods (LSH and PCAR) respectively on GIST-1M in terms of ℓ_2 metric (Euclidean) neighbor search. Each hashing method is applied to generate 500 bits on GIST-1M as a bit pool. Table 2 lists the MAP using varying numbers of hash bits, comparing different bit selection methods. From



(a) MAP @ 8-128 bits (b) R @ 32 bits (c) PH2 @ 32 bits (d) P-R @ 32 bits

Figure 4. Performance comparison of bit selection methods over multiple hashing algorithms on GIST-1M.

the table, two main observations can be obtained: (1) The performances of all methods improve when using more hash bits; (2) NDomSet works well over both hashing methods and consistently outperforms all baselines significantly. The results indicate that only considering one from similarity preservation and mutual independence will be difficult for bit selection methods to explore the best bit subset on the dataset, while our method can complementarily combine both properties and thus achieves the best performance. LSH generates hashing functions independently, and thus Random over it is equivalent to the original hashing method. It can be noted that NDomSet attains significant performance gains over LSH (or Random): 34.20% using 32 bits, and 19.48% using 64 bits.

The proposed bit selection can also work well with multiple hashing methods, besides being compatible with multiple features. A bunch of hashing methods have been developed in the literature, however, these methods are usually prevented from being widely used, due to the limited hash code length, stemming from the feature dimension [4, 19] or the landmark number [14]. Therefore, it becomes very beneficial to select the required number of bits using bit selection over multiple hashing methods. To evaluate the selection performance, we build a large bit pool with 600 bits, of which 200 are respectively generated by ITQ, SPH, and RMMH with 960-D GIST feature. Then all bit selection methods are performed on this pool and compared with original hashing methods using its top bits on GIST-1M.

In Figure 4, we report several performance statistics based on the Euclidean groundtruth. We show the recall, hamming lookup precision within radius 2 (PH2), and P-R curves using 32 bits, and investigate the MAP tendency with 8-128 hash bits. First note that in this experiment similarity preservation is more crucial than independence for the selection on mixed hash bits: in Figure 4 (a) Greedy achieves a close performance to that of RMMH, and a superior one to DomSet. It means that to a certain extent the candidate bits from the three hashing methods are individually of high quality and mutually independent. Even so, our bit selection method, incorporating both criteria, can faithfully boost the performance over the three hashing methods, and meanwhile surpasses all other selection baselines.

Table 2. MAP (%) of bit selection over different hashing algorithms using 32 - 128 bits on GIST-1M.

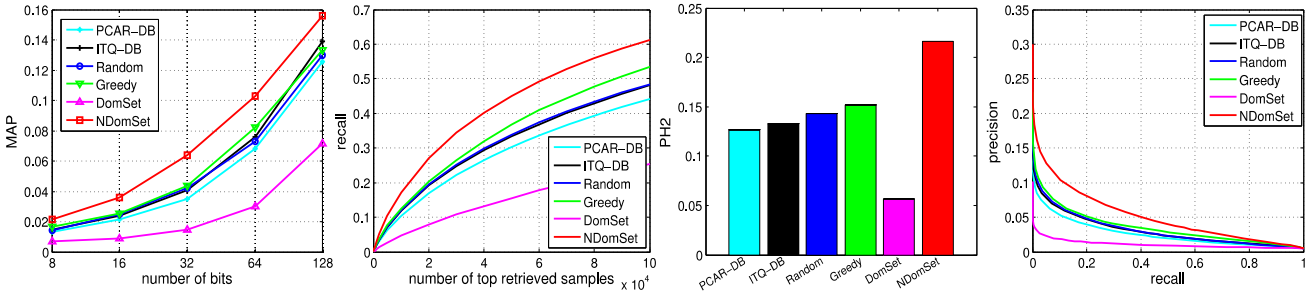
500 BITS		32	64	128
LSH	LRH [15]	3.71±0.19	6.93±0.21	11.27±0.16
	RANDOM	3.83±0.13	6.88±0.24	11.15±0.33
	GREEDY	3.19±0.22	5.18±0.17	8.84±0.15
	DOMSET	1.94±0.06	4.13±0.13	8.31±0.17
	NDOMSET	**5.14**±0.11	**8.22**±0.20	**12.07**±0.19
PCAR	LRH [15]	3.89±0.20	7.02±0.24	11.53±0.23
	RANDOM	4.24±0.38	6.98±0.43	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
	NDOMSET	**5.48**±0.30	**8.93**±0.33	**13.28**±0.14

5.2.3 Bit selection over multiple bit hashing

Most of state-of-the-art hashing methods generate a single bit by thresholding once along each projection vector. This might result in unexpected loss of similarity preservation, because the threshold, typically lying in the dense region, partitions neighbor points close to the threshold to different bits. Several works have attempted to improve the hashing performance by generating multiple bits per projection [11, 14]. 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.

Baselines. We employ bit quantization (DB) [11] with equal probability thresholding on PCAR (**PCAR-DB**) and ITQ (**ITQ-DB**) as our multiple bit hashing baselines.

A 500 bit pool is first built with 250 bits generated by PCAR-DB and the rest bits by ITQ-DB on GIST-1M. Figure 5 shows the results comparing PCAR-DB, ITQ-DB, and different bit selection methods. In Figure 5 (a) the MAP increases when using more bits, and NDomSet consistently achieves the best performance. Compared with greedy selection, Random and DomSet give the worst performance, which indicates that the quality of hash bits generated



(a) MAP @ 8-128 bits (b) R @ 32 bits (c) PH2 @ 32 bits (d) P-R @ 32 bits

Figure 5. Performance comparison of bit selection methods over multiple bit hashing on GIST-1M.

by DB varies widely and thus most information might be contained in a very small bit subset. Previous works report that double bit quantization improves the hashing performance [11, 14]. Here our bit selection method ranks first with a large margin compared to the best competitors greedy selection and ITQ-DB in terms of MAP (up to 45.66% and 56.76% gaps respectively), recall, PH2 (up to 51.47% and 62.86% gaps respectively) and P-R curves using 32 bits. This fact leads to the conclusion that our bit selection method can further improve performances over DB by elegantly examining the most dominant bit subset.

6. Conclusion

We proposed a unified bit selection method supporting various scenarios. It was formulated as a quadratic programming that simultaneously considers both similarity preservation and independence between bits. By representing the large bit pool as a vertex- and edge-weighted graph, the desired selected bit subset corresponds to the introduced normalized dominant set, and can be solved efficiently by a quadratic programming. Comprehensive results over large-scale benchmarks are considerably encouraging: for several scenarios the proposed bit selection method significantly outperforms the state-of-the-art hashing methods.

7. Acknowledgement

This work is supported in part by National Major Project of China (2010ZX01042-002-001-00) and the SKLSDE Foundation (SKLSDE-2011ZX-01).

References

[1] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time markov chains. In *CONCUR*, pages 146–161. Springer-Verlag, 1999.

[2] 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 *CIVR*, 2009.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, 2004.

[4] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[5] J. He, S. Kumar, and S.-F. Chang. On the difficulty of nearest neighbor search. In *ICML*, 2012.

[6] J. He, R. Radhakrishnan, S.-F. Chang, and C. Bauer. Compact hashing with joint optimization of search accuracy and time. In *CVPR*, 2011.

[7] J. Heo, Y. Lee, J. He, S.-F. Chang, and S. Yoon. Spherical hashing. In *CVPR*, 2012.

[8] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1):117–128, 2011.

[9] A. Joly and O. Buisson. Random Maximum Margin Hashing. In *CVPR*. IEEE, 2011.

[10] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[11] Y. Lee, J. Heo, and S. Yoon. Quadra-embedding: Binary code embedding with low quantization error. In *ACCV*, 2012.

[12] H. Liu and S. Yan. Robust graph mode seeking by graph shift. In *ICML*, 2010.

[13] S. Liu, H. Liu, L. J. Latecki, S. Yan, C. Xu, and H. Lu. Size adaptive selection of most informative features. In *AAAI*, 2011.

[14] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

[15] Y. Mu, X. Chen, X. Liu, T.-S. Chua, and S. Yan. Multimedia semantics-aware query-adaptive hashing with bits reconfigurability. *IJMIR*, pages 1–12, 2012.

[16] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[17] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE TPAMI*, 29(1):167–172, 2007.

[18] J. Song, Y. Yang, Z. Huang, H. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, 2011.

[19] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[20] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, 2010.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.