1. Proof of Theorem 1

To prove the theorem in the paper, we first prove the following proposition and lemmas.

The definition of the normalized dominant set can be characterized and represented in terms of determinants. For $S \subseteq V$, we denote by $A_S$ the submatrix of $A$ formed by the rows and the columns indexed by the elements of $S$. Then we define the matrix $B_S$:

$$B_S = \begin{pmatrix} 0 & (\pi^{-1})^T \end{pmatrix},$$

and the matrix $J B_S$:

$$J B_S = \begin{pmatrix} 0 & (\pi^{-1})^T \end{pmatrix},$$

where $S = \{i_1, \ldots, i_m\}$ with $i_1 < \ldots < i_m$ and $A_S^i$ is the $i$-th column of $A_S$.

**Proposition 1** Let $S = \{i_1, \ldots, i_m\} \subseteq V$ be a nonempty subset of vertices and assume $i_1 < \ldots < i_m$ without loss of generality. Then for any $i_h \in S$,

$$w_S(i_h) = (-1)^m \det(J B_S),$$

and,

$$W_S = (-1)^m \det(B_S).$$

**Proof** First we prove (3) holds for $m = 1$.

1. For $m = 1$,

$$J B_{(i)} = \begin{pmatrix} 0 & \pi_i^{-1} \end{pmatrix},$$

It is easy to verify that $w(j)(i) = \frac{1}{\pi_i} = -\det(J B_{(i)})$.

2. For $m > 1$ and $S = \{i_1, \ldots, i_m\}$ with $i_1 < \ldots < i_m$,
According to the recursive definition of $w_S(i_h)$, we can conclude that (3) holds for $m \geq 1$.

For (4), since $\det(B_S) = \sum_{i_h \in S} \det(hB_S)$, then $W(S) = (-1)^m \det(B_S)$.

When using the following fact
\[
\begin{bmatrix}
\pi_i^{-1} \\
\pi_i^{-1} (A_S^b)^T
\end{bmatrix} = 0,
\]
we give an alternative way to compute $w_S(i)$:
\[
w_S(i) = \sum_{j \in S \setminus \{i\}} \frac{1}{\pi_i}(\pi_i \pi_j a_{ij} - \pi_h \pi_j a_{hj})w_S(j)(i) \tag{5}
\]
with $h$ as an arbitrary element of $S \setminus \{i\}$, and $|S| > 1$.

**Lemma 1** With $\hat{A} = \Pi \Lambda$, where $\Pi = \text{diag}(\pi)$, the KKT equality conditions in (7) in the paper hold, if and only if
\[
B_\sigma[-\lambda, \pi_i x_i, \ldots, \pi_i x_i] = [1, 0, \ldots, 0]^T, \tag{6}
\]
where $\lambda$ is a real constant number, $\sigma = \sigma(x) = \{i_1, \ldots, i_h\}$ with $i_1 < \ldots < i_h$, and the matrix $B_\sigma$:
\[
B_\sigma = \begin{pmatrix}
0 & \pi_i^{-1} & \cdots & \pi_{i_h}^{-1} \\
\pi_i^{-1} & a_{i_1 i_1} & \cdots & a_{i_h i_h} \\
\vdots & \vdots & \ddots & \vdots \\
\pi_i^{-1} & a_{i_1 i_h} & \cdots & a_{i_h i_h}
\end{pmatrix}.
\]

**Proof** $x \in \Delta$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for problem (7), if there exist $n+1$ real constants (Lagrange multipliers) $\mu_1, \ldots, \mu_L$, and $\lambda$, with $\mu_i \geq 0$ for all $i = 1 \ldots L$, such that for all $i = 1 \ldots L$:
\[
(\hat{A}x)_i - \lambda + \mu_i = 0; \quad x_i \mu_i = 0.
\]
Because of the nonnegativity of both $x_i$ and $\mu_i$, it can be restated as follows:
\[
(\hat{A}x)_i \begin{cases}
\lambda, & \text{if } i \in \sigma(x); \\
\leq \lambda, & \text{otherwise}
\end{cases}
\]
with some real constant $\lambda = x^T \hat{A} x$.

For $\sigma = \sigma(x) = \{i_1, \ldots, i_m\}$ with $i_1 < \ldots < i_m$.
\[
B_\sigma = \begin{pmatrix}
0 & \pi_i^{-1} & \cdots & \pi_{i_m}^{-1} \\
\pi_i^{-1} & a_{i_1 i_1} & \cdots & a_{i_m i_m} \\
\vdots & \vdots & \ddots & \vdots \\
\pi_i^{-1} & a_{i_1 i_m} & \cdots & a_{i_m i_m}
\end{pmatrix}.
\]
Then $B_\sigma[-\lambda, \pi_i x_i, \ldots, \pi_i x_i] = [1, 0, \ldots, 0]^T$ is equivalent to:
\[
\begin{cases}
\sum_{h=1}^{m} x_{ih} = 1; \\
A_\sigma[\pi_i x_i, \ldots, \pi_i x_i] = \lambda[\pi_i^{-1}, \ldots, \pi_{i_m}^{-1}]
\end{cases}
\]

namely, using $\Pi = \text{diag}(\pi)$
\[
\begin{cases}
1^T x = 1; \\
(\Pi \Lambda \Pi x)_i = \lambda, \ i \in \sigma(x)
\end{cases}
\]

By setting $A = \Pi \Lambda$, we prove Lemma 1.

**Lemma 2** Let $\sigma = \sigma(x)$ be the support of a vector $x \in \Delta$. Then, $x$ satisfies the KKT equality conditions in (7) in the paper if and only if
\[
x_i = \begin{cases}
w_{\sigma,i}(i), & \text{if } i \in \sigma; \\
0, & \text{otherwise.}
\end{cases} \tag{7}
\]

Moreover,
\[
\frac{w_{\sigma,i}(j)(j)}{W(\sigma)} = \frac{1}{\pi_j} [(\hat{A} x)_j - (\hat{A} x)_{i_h}] = -\frac{1}{\pi_j} \mu_j \tag{8}
\]
for all $i \in \sigma$ and $j \notin \sigma$, where the $\mu_j$ are the (nonnegative) Lagrange multipliers of program (7).

**Proof** For (6) which is equivalent to the KKT conditions $\hat{A} x_i = \lambda$, if $i \in \sigma(x)$, it can be treated as a linear equation problem with unknowns $\lambda$ and $x_i, i \in \sigma$. Since $\det(B_\sigma) \neq 0$, the problem has an unique solution where the support denoted by $\sigma = \{i_1, \ldots, i_m\}$ without loss of generality. Using Cramer’s rule, we can get
\[
\pi_{i_h} x_{i_h} = \frac{\pi_{i_h} \det(hB_\sigma)}{\det(B_\sigma)}.
\]

Then according to Lemma 1, we have
\[
x_{i_h} = \frac{(-1)^m w_{\sigma,i}(i_h)}{(-1)^m W(\sigma)} = \frac{w_{\sigma,i}(i_h)}{W(\sigma)}.
\]
for any $1 \leq h \leq m$. Therefore, $x = x^\sigma$.

Using Equation (5) in the paper, we obtain:
\[
\frac{w_{\sigma,j}(j)}{W(\sigma)} = \frac{1}{\pi_j} \sum_{i_h \in \sigma} \pi_{i_h} a_{i_h j} - \pi_{i_h} x_{i_h} a_{i_h j} w_{\sigma,i}(i_h) \frac{W(\sigma)}{W(\sigma)}
\]
\[
= \frac{1}{\pi_j} \sum_{i_h \in \sigma} [\pi_{i_h} a_{i_h j} - \pi_{i_h} x_{i_h} a_{i_h j}] x^\sigma_{i_h}
\]
\[
= \frac{1}{\pi_j} [(\hat{A} x)_j - (\hat{A} x)_{i_h}].
\]

We have the fact $(\hat{A} x)_j - (\hat{A} x)_{i} = -\mu_j$ for all $i \in \sigma$ and $j \notin \sigma$, and $\pi_j > 0$ for all $j$. Then we can conclude that
\[
\frac{w_{\sigma,j}(j)}{W(\sigma)} = \frac{1}{\pi_j} [(\hat{A} x)_j - (\hat{A} x)_{i}] = -\frac{1}{\pi_j} \mu_j \leq 0.
\]
for all $i \in \sigma$ and $j \notin \sigma$, where the $\mu_j$ are the (nonnegative) Lagrange multipliers of quadratic programming problem in the paper.
Table 1. MAP (%) of bit selection over different hashing algorithms using 32-128 bits on GIST-1M.

<table>
<thead>
<tr>
<th>Baselines</th>
<th>500 BITS</th>
<th>32 BITS</th>
<th>64 BITS</th>
<th>128 BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMMH</td>
<td>5.85</td>
<td>10.86</td>
<td>16.78</td>
<td></td>
</tr>
<tr>
<td>LRH</td>
<td>5.81</td>
<td>10.75</td>
<td>16.53</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>5.74</td>
<td>9.56</td>
<td>15.65</td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td>3.78</td>
<td>7.72</td>
<td>13.97</td>
<td></td>
</tr>
<tr>
<td>DomSet</td>
<td>7.06</td>
<td>11.78</td>
<td>17.56</td>
<td></td>
</tr>
<tr>
<td>NDomSet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Proof Using Lemma 1 and 2, the proof can be completed following that of [7].

2. More experimental results

2.1. Bit selection over basic hashing method

Baselines. We still use the learned reconfigurable hashing (LRH) [6] and other naive selection methods as our baselines. However, here we report more results over the state-of-the-art hashing method RMMH [3].

The results are shown in Table 1. As we can observe, the proposed selection (NDomset) attains the best performance in all cases in terms of MAP with 32, 64 and 128 bits. Furthermore, RMMH generates each hash bit independently, and thus randomly selecting \( l \) bits acts like generating \( l \) bits using RMMH directly. By comparing results using Random and other bit selection methods, we can conclude that our bit selection method improves the performance of RMMH most. Although LRH can obtain performance gains when using long hash bits, it fails to compete with NDomSet in terms of both accuracy and speed (Roughly, NDomSet is more than five times faster than it).

2.2. Bit selection over multiple hashing methods

Baselines. We consider the scenario using multiple hashing methods, where our baselines include some hashing methods with the longest codes they can generate.

The scenario derives from the fact that a number of hashing methods can only generate hash codes of limited length, due to the limited number of the feature dimension [1, 8] or the landmark number [5]. Using multiple hashing methods, we can generate any desired number of hash bits, and meanwhile our bit selection can select the most desired ones from the pool of bits generated by multiple hashing methods. With 128-D SIFT features [2], hashing methods like PCAH, PCAR and ITQ can generate at most 128 bits. Therefore to use longer hash codes, say 196 in our experiment, we build a large bit pool with 384 bits, of which each 128 bits are generated respectively by PCAH, PCAR, and ITQ [1]. Then we compare all selection methods picking 196 bits and basic hashing methods with 128 bits.

Figure 2 show the comparison performances in terms of P-R curves and MAP. As we can see, bit selection methods like Greedy and NDomSet using 196 bits outperform the three basic hashing methods using their longest hash codes individually (i.e., 128 bits), and moreover they achieve better performances than the native selection methods including Random and DomSet using the same number of hash bits. This observation demonstrates the benefits of a good bit selection under this specific scenario. Note that our bit selection obtains the most significant performance gains in all cases.

2.3. Bit selection over multiple bit hashing

Baselines. We employ another double bit method using hierarchical hashing (HH) proposed in [4]. Similar to DB in
the paper, we generate bits by HH over PCAR (PCAR-HH) and ITQ (ITQ-HH).

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 2 shows the results comparing PCAR-HH, ITQ-HH, and different bit selection methods. In Figure 2 (a) the MAP of NDomSet increases dramatically when using more bits, and is consistently superior to both the double bit hashing and selection baselines. Figure 2 (c)-(d) plot the recall, PH2 and P-R curves using 32 bits. In all cases, significant performance improvements are observed, which certainly supports our conclusion in the paper that our bit selection method can recognize the bits of good quality and further improve performances over multi-bit hashing algorithms.

References


