Materialized Views Selection in Distributed Networks

Zhenhua Huang[1,2], Zhenqi Zhao[2], Jiujun Cheng[2]

[1] School of Computer Science, South China Normal University, China

[2] School of Electronics and Information, Tongji University, China

jukiehuang@163.com, zhenqizhao0714@gmail.com, chengjj@tongji.edu.cn

Abstract

Materialized views selection has recently received increasing attention in the database community. Although several greedy and heuristic algorithms for materialized views selection have already been proposed in centralized environments, their quality can not be guaranteed well in realistic distributed environments. Motivated by the above, under the constraint of maintenance and communication costs, this paper proposes two efficient algorithms in SPA (Super-Peer Architecture) distributed networks and uses the map/reduce distributed computation model to quickly find the optimal materialized views. For the first algorithm SMVSA, we first heuristically constructs the initial set of materialized views and then adjusts the set of materialized views based on simulated annealing. While for the second algorithm SMVMST, we reduce the materialized views selection to the problem of producing the minimum Steiner tree and obtain the approximate optimal set of materialized views. The detailed theoretical analyses and extensive experiments demonstrate that our proposed algorithms are both efficient and effective.

Keywords: Materialized views, Map/reduce, Distributed networks, Simulated annealing, Minimum Steiner tree

1 Introduction

Materialized views selection has recently received overwhelming attention in the database community [1]. Given a multi-dimensional dataset, the materialized views selection problem is to find an optimal set of materialized views under a maintenance cost constraint (such as maintenance time or disk space) for the purpose of minimizing the total query processing cost for a given set of queries. It is easy to see that given a k-dimensional dataset, it totally has 2^k-1 materialized views. And it becomes extremely difficult to obtain the optimal set of materialized views as k increases [2].

Recently, various techniques have been proposed to improve the efficiency and quality of materialized views selection. Aouiche et al. [3] presented a framework for materialized views selection that exploits a data mining technique in order to determine clusters of similar queries. Gou et al. [4] considered the problem of materialized view selection under a disk-space constraint and proposed an innovative competitive A* algorithm [5]. The authors showed that it is just the distinctive topological structure of the dependent lattice that makes the A* search a very competitive strategy for this problem. Kumar et al. [6] designed an approach for selecting materialized views using the genetic algorithm [7]. The proposed approach computed the top-k views from a multidimensional lattice by exploring and exploiting the search space containing all possible views. Vishwanath et al. [8] applied the advanced concepts of frequent rule mining [9] of the data mining approach to select and maintain materialized views, which can decrease the query response time greatly. Jogekar et al. [10] presented a framework for selecting best materialized views so as to achieve the effective combination of good query response time, low query processing cost and low view maintenance cost in a specified storage space constraint. The parameters in the proposed framework included query frequency cost, query storage cost and query processing cost. Yu et al. [11] proposed a novel approach to the materialized view selection problem based on an adaptive genetic algorithm. The authors established a cost model that integrates the query, maintenance and storage costs to evaluate the performance of approaches and measure the fitness of an individual in the genetic algorithm, and introduced the adjustable factors for crossover probability and mutation probability, allowing the genetic algorithm to run quickly and avoid premature convergence. Kumar and Arun [12] made an attempt to select optimal materialized sets of views, which can significantly reduce the query response time. The authors presented a honey bee mating optimization [13] based view selection algorithm (HBMOVSA) to select Top-K views, from amongst all possible views, in a multidimensional lattice. Yao et al. [14] presented an improved and effective algorithm for materialized view selection. This algorithm considered the effect on the overall space and cost by adding candidate

*Corresponding Author: Zhenhua Huang; E-mail: jukiehuang@163.com

DOI: 10.3966/160792642019032002011

materialized views and reducing the views, as well as optimized the addition and deletion of candidate materialized views by selecting a lower cost for selecting views.

To the best of our knowledge, the existing algorithms at least have two drawbacks: (i) these algorithms are mostly proposed in centralized environments, and can not efficiently extend for realistic distributed applications. (ii) The accuracy of existing algorithms can not be guaranteed well since they may easily be trapped in a local optimum. Especially when the allowed maintenance cost is relatively small, the accuracy of these algorithms is extremely low.

In order to eliminate the above two drawbacks, in this paper, we focus on improving the efficiency and accuracy of materialized views selection under the cost constraint of maintenance and communication in SPA (Super-Peer Architecture) distributed networks [15]. The constraint of disk space is not taken into consideration in our paper since hard disks are relatively cheap at present and the response time is more important for users. Based on the map/reduce distributed computation model, two efficient algorithms are proposed to quickly find the optimal materialized views for a given set of queries. The first algorithm named SMVSA (Selecting Materialized Views based on Simulated Annealing) implements the process of materialized views selection via two phases: (i) heuristically constructing the initial set of materialized views based on the individual maximum benefit [16]; and (ii) adjusting the set of materialized views based on simulated annealing [17]. While the second algorithm named SMVMST (Selecting Materialized Views based on Minimum Steiner Tree) is an approximate approach which reduces the materialized views selection to the problem of producing the minimum Steiner tree [18] and gets the approximate optimal set of materialized views. Our two algorithms all have the polynomial time complexity. The detailed theoretical analyses and extensive experiments show that our proposed algorithms are both efficient and effective. In particular, our SMVMST algorithm can efficiently balance the time cost and accuracy, and has good extendibility.

2 Problem Description

Without loss of generality, we let the SPA distributed network H include m storage nodes $N_s^{(1)},\ldots, N_s^{(m)}$, and the query and computation node in H be N_q. Assume that for a given k-dimensional dataset, 2^k-1 materialized views $MV=\{mv_1,\ldots, mv_{2^k-1}\}$ are distributedly stored on $N_s^{(1)},\ldots, N_s^{(m)}$, and the set of queries $\{q_1, \ldots, q_n\}$ are submitted on N_q. Figure 1 shows the lattice of materialized views L^4 for a 4-dimensional dataset $R(ABCD)$. And for a given set of queries $Q=\{q_1, q_2, q_3\}$, Figure 2 shows an example of using L^4 to answer Q. In Figure 2, we can see that (*i*) q_1 can be answered by *ABCD*, *ABC*, *BCD* and *BC*; (*ii*) q_2 can be answered by *ABCD*, *ABD*, *ACD*, *AD*, *A* and *D*; (*iii*) q_3 can be answered by *ABCD* and *BCD*.

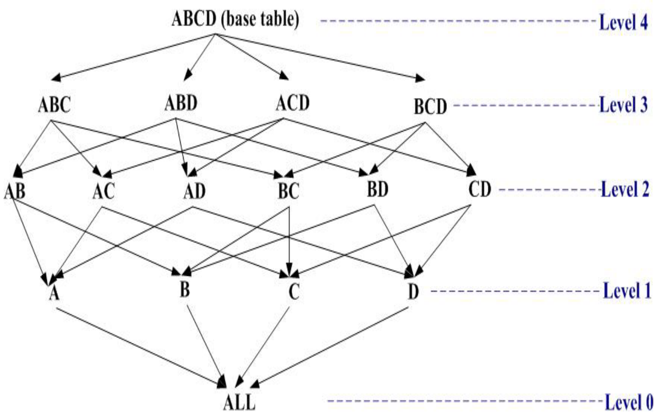


Figure 1. The lattice of materialized views

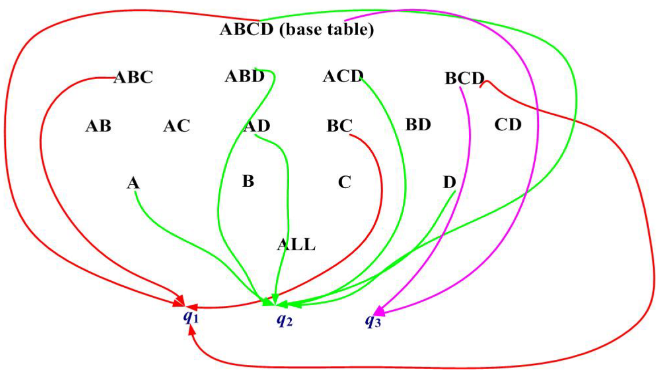


Figure 2. An example of using L^4 to answer Q

It is not difficult to see that the set of 2^k-1 materialized views $MV=\{mv_1,\ldots, mv_{2^k-1}\}$ and the set of queries $Q=\{q_1, \ldots, q_n\}$ form a directed weighted graph $G=(N, E, W)$: $N=MV\cup Q$; $E=\{<mv_1\rightarrow mv_2>|mv_1, mv_2\in MV$ and mv_2 can be updated by $mv_1\}\cup\{<mv\rightarrow q>|mv\in MV, q\in Q$ and q can be answered by $mv\}$; W includes the following parts:

— four weights on a vertex $\lambda\in N$:

∍ sc_λ: the initial data scan cost;

∍ qf_λ: the query frequency;

∍ uf_λ: the update frequency;

∍ cf_λ: the communication frequency.

It is easy to see that for each $\lambda\in Q$, $sc_\lambda=uf_\lambda=cf_\lambda=0$.

— two weights on an edge $<mv_1\rightarrow mv_2>\in E$ (mv_1, $mv_2\in MV$):

∍ $ucost(mv_1, mv_2)$: the updating cost of mv_2 using mv_1;

∍ $ccost(mv_1, mv_2)$: the communication cost of mv_1 from the storage node including mv_1 to the storage node including mv_2.

— two weights on an edge $<mv_1\rightarrow q>\in E$ ($mv_1\in MV$ and $q\in Q$):

∍ *qcost*(mv_1, q): the query processing cost of q using mv_1;

∍ *ccost*(mv_1, q): the communication cost of mv_1 from the storage node including mv_1 to the query and computation node including mv_2.

Based on the directed weighted graph G, for a query q and a selected materialized view mv, we define the function $\psi(mv, q)$ as the sum of the query processing costs associated with edges on the shortest path from v to mv plus the initial data scan cost of the vertex mv, sc_{mv}. That is,

$$\psi(mv, q) = \min\{q\cos t(mv, q)\} + sc_{mv} \qquad \textbf{(1)}$$

Note that if the view mv can not answer the query q, then we use the raw dataset instead of mv. In a similar fashion, we define two cost functions:

(i) For two materialized view mv_1 and mv_2, we define the function $\phi(mv_1, mv_2)$ as the sum of the maintenance costs associated with the edges on the shortest path from mv_1 to mv_2. That is,

$$\phi(mv_1, mv_2) = \min\{u\cos t(mv_1, mv_2)\} \qquad \textbf{(2)}$$

(ii) For two nodes α, β in N, we define the function $\varpi(\alpha, \beta)$ as the sum of the communication costs associated with the edges on the shortest path from α to β. That is,

$$\varpi(\alpha, \beta) = \min\{c\cos t(\alpha, \beta)\} \qquad \textbf{(3)}$$

Then we can define the problem of materialized views selection under the cost constraint of maintenance and communication.

Problem definition. Given an above-mentioned directed weighted graph G=(N, E, W), the problem of materialized views selection under the cost constraint of maintenance and communication is to select a set of views $\Delta \subseteq MV$ that minimizes $\Theta(\Delta, G, Q)$, where

$$\Theta(\Delta, G, Q) = \sum_{q \in Q} qf_q \cdot \psi(\Delta, q) \qquad \textbf{(4)}$$

under the constraint that $\Omega(\Delta, G) \leq userCost$, where $\Omega(\Delta)$, the total maintenance and communication cost can be defined as

$$\Omega(\Delta, G) = \sum_{mv \in \Delta} uf_{mv} \cdot \phi(\Delta, mv) + \sum_{\alpha \in \Delta \cup Q} cf_\alpha \cdot \varpi(\Delta, \alpha) \qquad \textbf{(5)}$$

Here, $\psi(\Delta, q)$ denotes the minimum cost of answering a query $q(\in Q)$ in the presence of the set of materialized views Δ; $\phi(\Delta, mv)$ is the minimum cost of maintaining a materialized view mv in the presence of the set of materialized views Δ; and $\varpi(\Delta, \alpha)$ is the minimum communication cost from N_{mv} to N_α where N_{mv} is the distributed node including $mv \in \Delta$ and N_α is the distributed node including α.

3 The Processing Framework of Our Solution

For a k-dimensional dataset $\mathfrak{I}$ and a set of queries Q= $\{q_1, \ldots, q_n\}$, obtaining the optimal set of materialized views $MV^{opt} \subseteq MV = \{mv_1, \ldots, mv_{2^k-1}\}$ is well-known NP- complete problem [19]. Hence in this paper, we propose two efficient algorithms SMVSA and SMVMST to quickly find the optimal materialized views. The detailed implementation of SMVSA and SMVMST can be seen in the next section. In this section, we present the processing framework that supports SMVSA and SMVMST based on the map/reduce distributed computation model. Figure 3 shows the data and control flows for the processing framework of our solution.

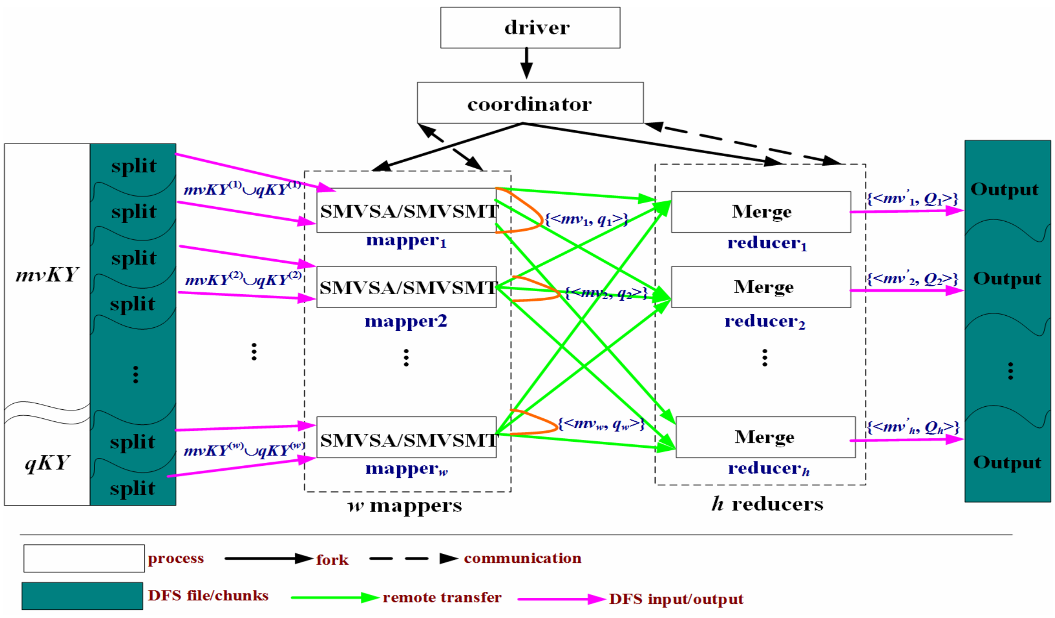


Figure 3. The data and control flow for the processing framework of our solution

The pseudo-code of processing framework can be shown in Algorithm 1.

Algorithm 1. The processing framework

```
Input:  the materialized views set $MV=\{mv_1,\ldots,mv_{2^k-1}\}$, the set of queries $Q=\{q_1, \ldots, q_n\}$,
        and the costconstraint of maintenance and communication $userCost$;
Output: the optimal set of materialized views $MV^{opt}$.
Begin
1.  $mvKY \leftarrow \varnothing$;
2.  $qKY \leftarrow \varnothing$;
3.  For each materialized view $mv_i \in MV$ ($1 \le i \le 2^k$-1) do
4.      $mvKY \leftarrow mvKY \cup \{<$'mv'$+i, mv_i>\}$;
5.  For each query $q_i \in Q$ ($1 \le i \le n$) do
6.      $qKY \leftarrow qKY \cup \{<$'q'$+i, q_i>\}$;
7.  Partition $mvKY$ into $w$ parts $mvKY^{(1)},\ldots, mvKY^{(w)}$;
    /* $w$ is an user parameter */
8.  Partition $qKY$ into $w$ parts $qKY^{(1)},\ldots, qKY^{(w)}$;
9.  For $i$=1 to $w$ do
10.     $mvqKY^{(i)} \leftarrow mvKY^{(i)} \cup qKY^{(i)}$;
11.     $\{<mv_x, q_z> \mid <$'mv'$+x, mv_x> \in mvKY^{(i)}$ and $<$'q'$+z, q_z> \in qKY^{(i)}\} \leftarrow$ **map**($mvqKY^{(i)}$);
        /* $q_z$ can be answered by $mv_x$ */
12. Let the partition function $f$ equal ($i$ mod $r$);
    /* $r$ is the number of computers used to execute the reduce function */
13. $MV^{opt} \leftarrow \varnothing$;
14. For $j$=1 to $h$ do  /* parallel processing */
15.     $\{<mv'_x, Q'> \mid <$'mv'$+x, mv_x> \in mvKY^{(i)}$ and $Q'=\{q_z \mid <$'q'$+z, q_z> \in qKY^{(i)}\}\} \leftarrow$ **reduce**($\{<mv, q>\}$);
        /* each query in $Q'$ can be answered by $mv'_x$ */
16.     $MV^{opt} \leftarrow MV^{opt} \cup \{<mv_x', Q'> \mid Q' \neq \varnothing\}$;
17. Return $MV^{opt}$.
End
```

In Algorithm 1 (The processing framework), under the map/reduce distributed computation model, we first create two key-value sets *mvKY* and *qKY* which are related with materialized views and queries respectively. In *mvKY*, each pair of key-value consists of a materialized view and its identifier. While in *qKY*, each pair of key-value consists of a query and its identifier (Lines 3-6). Then according to the number *w* of mappers in distributed networks, the algorithm partitions *mvKY* and *qKY* into *w* parts respectively: $mvKY^{(1)},\ldots, mvKY^{(w)}$, and $qKY^{(1)},\ldots, qKY^{(w)}$ (Lines 7-8). For each $mvKY^{(i)}$ and $qKY^{(i)}$ ($1 \le i \le w$), the algorithm calls the map function to realize the efficient optimization of materialized views selection (Lines 10-11). Note that the main task of map function is to implement the algorithms SMVSA or SMVMST to obtain the optimal materialized views from $mvKY^{(i)}$ for $qKY^{(i)}$ (see Section 4). Moreover, based on the partition function *f*, reducers receives the corresponding intermediary key-value set from mappers, and call the reduce function to merges the queries and produces the set *Q'* for the same materialized view *mv'* (Line 15). Finally, the algorithm removes those materialized views which are not used to answer any query, and returns the remaining ones to users (Lines 16-17).

The map and reduce functions can be shown in Algorithm 2 and Algorithm 3.

Algorithm 2. the map function

```
Input:  the key-value set $mvKY$={<the identifier of materialized view, the materialized view>},
        the key-value set $qKY$={<the identifier of query, the query>};
Output: the key-value set $intKY$.
Begin
1.  $intKY \leftarrow \varnothing$;
2.  $Cost^{(mp)} \leftarrow userCost/w$;
    /* $userCost$ is the cost constraint, $w$ is the number of mappers */
3.  $R \leftarrow$ the root materialized view which can answer all queries in $qKY$;
4.  If $\Omega(R) > Cost^{(mp)}$ then Return NULL;
    /* $\Omega(R)$ is the time cost of maintenance and communication of $R$ and can be seen in Formula (5) */
5.  Else
6.      $MV^{(mp)} \leftarrow \{mv \mid$<the identifier of $mv$, $mv> \in mvKY\}$;
7.      $Q^{(mp)} \leftarrow \{q \mid$<the identifier of $q$, $q> \in qKY\}$;
8.      Construct the directed weighted graph $G=(N, E, W)$ based on $MV^{(mp)}$ and $Q^{(mp)}$;
        /* see Section 2 for details */
9.      $intKY \leftarrow$ **SMVSA**($G$) or $intKY \leftarrow$ **SMVMST**($G$);
10. Return $intKY$.
End
```

In the above algorithm (Algorithm 1), the map function takes the key-value set $mvKY \cup qKY$ as the input parameter and first obtains the root materialized view *R* (Line 3). And then the map function estimates whether or not the time cost $\Omega(R)$ of maintenance and communication of *R* is greater than *userCost*/*w*. If $\Omega(R) > userCost/w$, then we can easily determine that the maintenance and communication cost of any set of materialized views will be greater than the value of *userCost*/*w*, and therefore the map function can not implement the optimization of materialized views selection and returns the value of NULL (Line 4). Else, the map function constructs the directed weighted graph *G*, and uses the algorithms SMVSA or SMVMST to realize the efficient optimization of materialized views selection (Lines 8-9).

Algorithm 3. the reduce function

```
Input:  the key-value set $intKY$={<the materialized view, the query>};
Output: the key-value set $optKY$.
Begin
1.  $MV^{(re)} \leftarrow \{mv | <mv, q> \in intKY\}$;
2.  $optKY \leftarrow \emptyset$;
3.  For each $mv' \in MV^{(re)}$ do
4.      $Q^{(mv')} \leftarrow \{q | <mv', q> \in intKY\}$;
5.      $optKY \leftarrow optKY \cup \{<mv', Q^{(mv')}>\}$
6.  Return $optKY$.
End
```

In Algorithm 3, the reduce function takes the key-value set *intKY* as the input parameter and first gets the set of materialized views $MV^{(re)}$ including all materialized views from *intKY* (Line 1). Then the reduce function merges the queries and generates $Q^{(mv')}$ for the same materialized view *mv'* (Lines 3-5).

4 Quick Selection of The Optimal Materialized Views

In this section, based on the processing framework presented in Section 3, we give two efficient algorithms SMVSA and SMVMST to select the optimal materialized views for the given set of queries.

4.1 The SMVSA Algorithm

The SMVSA algorithm implements the process of materialized views selection through two phases: (i) heuristically constructing the initial set of materialized views based on the individual maximum benefit, and (ii) adjusting the set of materialized views based on simulated annealing.

Algorithm 4 shows the pseudo-code of SMVSA.

Algorithm 4. SMVSA

```
Input: the directed weighted graph $G=(N, E, W)$;
Output: the key-value set $intKY$.
Begin
1.  $Q \leftarrow$ the set of queries from $G$;
2.  $MV \leftarrow$ the set of materialized views from $G$;
3.  $\wp \leftarrow$ the root materialized view in $G$;
4.  $intKY \leftarrow \emptyset$;
5.  $useMV \leftarrow \{R\}$;
6.  For each $q \in Q$ do  $intKY \leftarrow intKY \cup \{<R, q>\}$;
    /* the first phase: Lines 7-13*/
7.  For each $q \in Q$ do
8.      $List^{(q)} \leftarrow$ the list of materialized views which can answer $q$;
9.      Sort $List^{(q)}$ according to $\psi(mv, q)$ in an ascending order;
        /* $\psi(mv, q)$ can be seen in Formula (1) */
10.     Visit $List^{(q)}$ and obtain the first materialized view $mv^{(q)}$ from $List^{(q)}$ satisfying:
        $\Omega(useMV \cup \{mv^{(q)}\}, G) \le Cost^{(mp)}$;
11.     $useMV \leftarrow useMV \cup \{mv^{(q)}\}$;
12.     $intKY \leftarrow intKY \cup \{<mv^{(q)}, q>\} - \{<\wp, q>\}$;
13. If $\neg\exists q \in Q$, $<R, q> \in intKY$ then $useMV \leftarrow useMV - \{R\}$;
    /* the second phase: Lines 14-36 */
14. Let $T_{start}$, $T_{min}$ be the initial temperature and the minimum temperature respectively;
15. While $T_{start} > T_{min}$ do
16.     For $i$=1 to $\rho$ do
        /*$\rho$ is the number of inner loop iterations */
17.         $intKY' \leftarrow intKY$;
18.         $G' \leftarrow G$;
20.         $useMV' \leftarrow useMV$;
21.         Randomly choose $s=\lceil |intKY|/l \rceil$ pairs from $intKY$ and form the set $S=\{<mv_x, q_x> | 1 \le x \le s\}$;
            /* $l_1$ is an user parameter */
22.         For $x$=1 to $\lfloor s/2 \rfloor$ do
23.             If $q_x$ can be answered by $mv_{s-x}$ and $q_{s-x}$ can be answered by $mv_x$ then
24.                 $intKY' \leftarrow intKY' \cup \{<mv_x, q_{s-x}>, <mv_{s-x}, q_x>\} - \{<mv_x, q_x>, <mv_{s-x}, q_{s-x}>\}$;
25.         Randomly choose $r=\lceil |intKY'|/l_2 \rceil$ pairs from $intKY'$ and form the set $R=\{<mv_z, q_z> | 1 \le z \le r\}$;
            /* $l_2$ is an user parameter */
26.         For $z$=1 to $r$ do
27.             $mv' \leftarrow$ the materialized view in $MV$-$useMV$ which has the minimum cost $\psi(mv', q_z)$;
28.             $intKY' \leftarrow intKY' \cup \{<mv', q_z>\} - \{<mv_z, q_z>\}$;
29.             $useMV' \leftarrow useMV' \cup \{mv'\}$;
30.         If $\neg\exists q \in Q$, $<mv_z, q> \in intKY$ then $useMV' \leftarrow useMV' - \{mv_z\}$;
31.             Update $G'$ based on the redirection between materialized views and queries;
32.         If $\Omega(useMV', G') \le Cost^{(mp)}$ then
33.             $\eta \leftarrow \Theta(useMV', G', Q) - \Theta(useMV, G, Q)$;
                /* $\Theta(useMV, G, Q)$ can be seen in Formula (4)*/
34.             If $\eta < 0$ then $intKY \leftarrow intKY'$;
                $useMV \leftarrow useMV'$;
                $G \leftarrow G'$;
35.             Else if $e^{\frac{-\eta}{T_{start}}}$ > $random$ (0, 1) then $intKY \leftarrow intKY'$; $useMV \leftarrow useMV'$; $G \leftarrow G'$;
36.     $T_{start} \leftarrow T_{start} \times \vartheta$;
        /* $\vartheta \in (0, 1)$ is an user parameter */
37. Return $intKY$;
End
```

In Algorithm 4, SMVSA initially allots the root materialized view R to all queries in Q and constructs the key-value set $intKY=\{<\wp, q> | q \in Q\}$ (Line 6). The first phase of optimization involves Lines 7-13. In this phase, for each query q, SMVSA obtains the materialized view $mv^{(q)}$ which can substitute R to answer q and satisfies that the query processing costs from $mv^{(q)}$ to q is minimum and the cost constraint of maintenance and communication is satisfied. Note that if none of queries is answered by R, then we can safely remove it from the used set *useMV* of materialized views. The second phase of optimization involves Lines 14-36. The core of this phase is generating the new key-value set *intKY'* based on the old one *intKY*. Under the constraint of maintenance and communication cost, SMVSA generates *intKY'* through two ways: the inner exchange of *useMV* (Lines 21-24) and the outer exchange between *useMV* and *MV-useMV* (Lines 25-30). Whether to replace *intKY* with *intKY'* or not is determined by the metropolis rule (Lines 33-35).

It is not difficult to see that the SMVSA algorithm has the polynomial time complexity, which is shown in Theorem 1.

Theorem 1. Assume there exists a set of materialized views $MV=\{mv_1,\ldots, mv_\delta\}$ and a set of queries $Q=\{q_1,\ldots, q_\mu\}$, then the time cost of the SMVSA algorithm equals:

$$O(\mu\cdot\delta\cdot\log\delta+\gamma\cdot\rho\cdot\mu),$$

where γ and ρ are the numbers of outer and inner loop iterations (Lines 15 and 16), respectively.

Proof. The time cost of the SMVSA algorithm mainly includes two parts: (*i*) the time cost of the first phase, i.e., Lines 7-13, and (*ii*) the time cost of the second phase, i.e., Lines 14-36. For the first phase, in order to heuristically construct the initial set of materialized views, SMVSA needs $O(\mu\cdot\delta\cdot\log\delta)$ time cost. For the second phase, in order to adjust the set of materialized views, SMVSA further needs $O(\gamma\cdot\rho\cdot(\mu/2+\mu+\mu)) = O(\gamma\cdot\rho\cdot\mu)$ time cost. Hence, the time cost of the SMVSA algorithm is $O(\mu\cdot\delta\cdot\log\delta+\gamma\cdot\rho\cdot\mu)$.

Therefore, Theorem 1 holds.

4.2 The SMVMST Algorithm

Although the above algorithm SMVSA can efficiently implement the materialized views selection, it can not commendably guarantee the optimization accuracy of materialized views selection, which can be seen in Section 5 (Experimental Evaluation). Hence, in this subsection, we propose the SMVMST algorithm which can commendably guarantee the optimization accuracy. And from Section 5 (Experimental Evaluation), we can find that the SMVMST algorithm efficiently balance the time cost and accuracy, and has good extendibility.

Algorithm 5 shows the pseudo-code of SMVMST.

Algorithm 5. SMVMST

```
Input: the directed weighted graph $G=(N, E, W)$;
Output: the key-value set $intKY$.
Begin
1.  $G'\leftarrow(N', E', W')$ which satisfies:
      (1) $N'=N=MV\cup Q$;
      (2) $E'=E$;
      (3) $W'(node\in N)=\begin{cases} qf_{node}\cdot sc_{node} & if\ node\in MV \\ 0 & if\ node\in Q \end{cases}$;
      (4) $W'(<\alpha\rightarrow\beta>\in E)=$
          $\begin{cases} 0 & if\ \alpha,\beta\in MV \\ qf_\alpha\cdot q\cos t(\alpha,\beta) & if\ \alpha\in MV\ and\ \beta\in Q \end{cases}$;
        /* the meanings of $qf_{node}$, $sc_{node}$ and $qcost(\alpha, \beta)$
           can be seen in Secction 2 */
2.  Produce the minimum steiner tree $msTree$ on $G'$
      using the IWD algorithm;
      /* the IWD (Intelligent Water Drops) algorithm
         was proposed in [18] */
3.  $intKY\leftarrow\varnothing$;
4.  $useMV\leftarrow\varnothing$;
5.  For each $q$ in $msTree$ do
6.    $intKY\leftarrow intKY\cup\{<mv, q>|q$ is answered by $mv$
        in $msTree\}$;
7.    $useMV\leftarrow useMV\cup\{mv\}$;
8.  If $\Omega(useMV, G)\leq Cost^{(mp)}$ then Return $intKY$;
9.  Else
10.   While $\Omega(useMV, G)\leq Cost^{(mp)}$ do
11.     $mv^{(max)}\leftarrow$the materialized view which has the
          maximum cost in $useMV$;
12.     $Q'\leftarrow\{q|q$ is answered by $mv^{(max)}$ in $msTree\}$;
13.     For each $q\in Q'$ do
14.       If $\exists mv\in useMV$, $mv$ can answer $q$ then
            $intKY\leftarrow intKY\cup\{<mv, q>\}-\{<mv^{(max)}, q>\}$;
15.       Else
16.         $mv'\leftarrow$the materialized view which has the
              minimum cost in $MV-useMV$ and can
              answer $q$;
17.         $intKY\leftarrow intKY\cup\{<mv', q>\}-\{<mv^{(max)}, q>\}$;
18.         $useMV\leftarrow useMV\cup\{mv'\}$;
19.     $useMV\leftarrow useMV-\{mv^{(max)}\}$;
20. Return $intKY$;
End
```

Since the directed weighted graph G is unsuited for the minimum Steiner tree algorithms, in Algorithm 5, in order to correctly obtain the minimum Steiner tree *msTree*, SMVMST first constructs a new directed weighted graph G' based on G (Line 1). In G', the sets of vertexes and edges are the same as the ones in G, and their difference is the weight of graph. G' has the weight values about the initial data scan cost and the query processing costs, ignoring the weight values about the maintenance and communication cost. Then based on G', SMVMST utilizes the IWD (Intelligent Water Drops) algorithm to efficiently produce the minimum Steiner tree *msTree* (Line 2). If the maintenance and communication cost of materialized views obtained from *msTree* exceeds $Cost^{(mp)}$, SMVMST adjusts the set *useMV* of materialized views and the key-value set *intKY*. The basic idea of adjustment is to exchange the materialized views to obatain the smaller maintenance and communication cost (Lines 11-19). Once the maintenance and communication cost is not greater than the constraint $Cost^{(mp)}$ through adjustment, SMVMST stops and returns the final *intKY*.

SMVMST has the polynomial time complexity, which is shown in Theorem 2.

Theorem 2. Assume there exists a set of materialized views $MV=\{mv_1,\ldots, mv_\delta\}$ and a set of queries $Q=\{q_1,\ldots, q_\mu\}$, then the time cost of the SMVMST algorithm equals:

$$O(\textstyle\sum_{i=1}^{\log^\delta} C_{\log^\delta}^{i}\cdot i+(\delta+\mu)^{e/(e-1)}+(\delta+\mu)\cdot\log(\delta+\mu)+\delta\cdot\mu\cdot l)\approx$$

$$O(\textstyle\sum_{i=1}^{\log^\delta} C_{\log^\delta}^{i}\cdot i+(\delta+\mu)^{1.58}+(\delta+\mu)\cdot\log(\delta+\mu)+\delta\cdot\mu\cdot l),$$

where l is the number of loop iterations (Line 10) and e is the base of natural logarithm.

Proof. The time cost of SMVMST mainly includes three parts:

(i) $O(\sum_{i=1}^{\log^\delta} C_{\log^\delta}^{i}\cdot i+\delta\cdot\mu)$: the time cost is need to construct the new directed weighted graph $G'(N', E', W')$ based on $\mathbf{G}(N, E, W)$.

(ii) $O(\delta\cdot\mu+(\delta+\mu)^{e/(e-1)}+(\delta+\mu)\cdot\log(\delta+\mu))$: the time cost is need to produce the minimum Steiner tree *msTree* on *G'* based on the IWD algorithm.

(iii) $O(\delta\cdot\mu\cdot l)$: the time cost is need to adjust the set *useMV* of materialized views and the key-value set *intKY*.

Therefore, Theorem 2 holds.

Below, Theorem 3 theoretically proves that the lower bound of SMVMST equals $(e-1)/e\approx0.63$. That is, the query processing cost produced by SMVMST is at most $e/(e-1)\approx$ 1.58 of that produced by the exact optimal algorithm that traverses exponential combinations of materialized views.

Theorem 3. Assume there exists a set of materialized views $MV=\{mv_1,\ldots, mv_\delta\}$ and a set of queries $Q=\{q_1,\ldots, q_\mu\}$ which form a directed weighted graph G. And we further assume the exact optimal algorithm selects w materialized views $MV^{(opt)}=\{mv_1^{opt},\ldots, mv_w^{opt}\}$, and SMVMST selects r materialized views $MV^{(our)}=\{mv_1^{our},\ldots, mv_r^{our}\}$. Then we can have:

$$\Theta(MV^{(our)}, G, Q)/\Theta(MV^{(opt)}, G, Q)\le e/(e-1)\approx1.58. \quad \textbf{(6)}$$

Proof. Without loss of generality, we assume the order that the exact optimal algorithm selects w materialized views is $mv_1^{opt},\ldots, mv_w^{opt}$, and for each materialized view mv_i^{opt} $(1\le i\le w)$, it needs the query processing cost $optc_i$. Meanwhile, we assume the order that our SMVMST algorithm selects r materialized views is mv_1^{our}, …, mv_r^{our}, and for each materialized view mv_i^{our} $(1\le i\le r)$, it needs the query processing cost $ourc_i$. Then we can have: $\Theta(MV^{(our)}, G, Q)=\sum_{i=1}^{w} optc_i$ and $\Theta(MV^{(opt)}, G, Q)=\sum_{i=1}^{r} ourc_i$.

For each pair of materialized views $mv_i^{opt}\in MV^{(opt)}$ and $mv_x^{our}\in MV^{(our)}$, we let $cost_{ix}$ be the common part of mv_x^{our}'s cost and mv_i^{opt}'s cost contributing to the query processing. Then the following four inequalities can be satisfied:

(1) $\sum_{i=1}^{w} c_{ix}\ge ourc_x$;

(2) $\forall i\in[1, w]$, $optc_i\ge ourc_1$;

(3) $\forall i\in[1, w]$, $optc_i-cost_{i1}\ge ourc_2$;

(4) $\forall$i$\in$[1, w], $\mathrm{optc_i-cost_{i1}-cost_{i2}-\ldots-cost_{ix}\ge ourc_x}$.

Then according to the above four inequalities we can get the following r inequalities.

(1) $\Theta(MV^{(opt)}, G, Q)\ge w\cdot ourc_1$;

(2) $\Theta(\mathrm{MV^{(opt)}, G, Q})\ge \mathrm{w\cdot ourc_2+ourc_1}$;

…

r) $\Theta(MV^{(opt)}, G, Q)\ge w\cdot ourc_r+ourc_{r-1}+\ldots+ourc_2+ourc_1$.

We can easily get that $\Theta(MV^{(opt)}, G, Q)$ is not less than any one of $w\cdot ourc_1$, $w\cdot ourc_2+ourc_1,\ldots, w\cdot ourc_r+ourc_{r-1}+\ldots+ourc_2+ourc_1$, and when these r values are equal, $\Theta(MV^{(opt)}, G, Q)$ will achieve the least value. Hence we can have:

$$\Theta(MV^{(opt)}, G, Q)\ge w\cdot\left(\frac{w}{w-1}\right)^{r-1}\cdot ourc_r .$$

On the other hand, when these r values are equal, $\forall x\in[1, r-1]$, $w\cdot ourc_{x+1}-(w-1)\cdot ourc_x=0$. That is $ourc_i=w/(w-1)\cdot ourc_{x+1}$. Then we can have:

$$\Theta(MV^{(our)}, G, Q)=\textstyle\sum_{x=0}^{r-1}\left(\frac{w}{w-1}\right)^{x}\cdot ourc_r$$

Hence we can have:

$$\Theta(MV^{(our)}, G, Q)/\Theta(MV^{(opt)}, G, Q)$$

$$\le k\cdot\textstyle\sum_{y=0}^{\partial_2-1}\left(\frac{\partial_1}{\partial_1-1}\right)^{y-\partial_2+1}$$

$$\le w\cdot\left(1+\frac{w-1}{w}+\ldots+\left(\frac{w-1}{w}\right)^{r-1}\right)$$

$$\le\frac{1}{1-\left(\frac{w-1}{w}\right)^{r}}.$$

Thereby when w and r approach $+\infty$, $\frac{1}{1-\left(\frac{w-1}{w}\right)^{r}}$ will approach $\frac{e}{e-1}\approx1.58$ and we can see that Theorem 3 holds.

5 Experimental Evaluation

In this section, we experimentally evaluate the optimization accuracy and the time cost of our algorithms SMVSA and SMVMST.

5.1 Experimental Setting

In our experiments, the running environment is a three-layer SPA distributed network, which consists of 40 PCs. Each PC has a quad-core i5-3450 CPU, 4G memory, 500G hard drive, and CentOS Linux 6.4 operating system.

The computation node contains a cluster consisting of 8 PCs, in which a PC is selected as the control computer (Master). These 8 PCs constitutes a Hadoop platform whose version number is 1.0.3. The remaining two layers include 32 distributed storage nodes, and each node has one PC.

In our experiments, we produce 100 queries on the computation node, and 2^{14}=16384 materialized views on the storage nodes. Thereby, each storage node has 512 materialized views on average.

There are 4 algorithms compared with our algorithms SMVSA and SMVMST: (*i*) OPTIMAL, the algorithm traverses exponential combinations of materialized views to obtain the exact optimal solution; (*ii*) ARM [8], the algorithm selects materialized views based on the advanced concepts of frequent mining; (*iii*) AGA [11], the algorithm selects materialized views based on the adaptive genetic algorithm; (*iv*) HBMOVSA [12], the algorithm selects materialized views based on the honey bee mating optimization. In the experiments, we extend the above four compared algorithms and integrate them into the map/reduce distributed computation model.

Each class of experiments is divided into three groups:

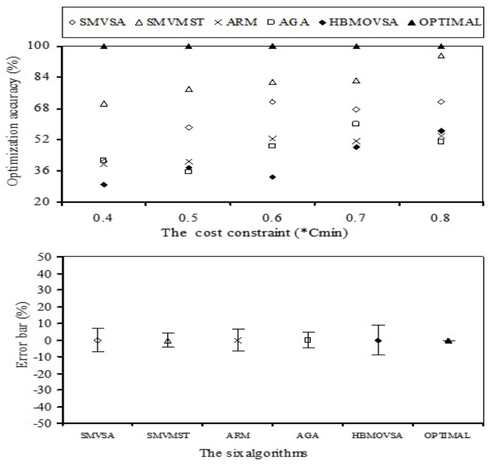
(1) The number *nq* of queries on the computation node is fixed to 100; the number *nmv* of materialized views on each storage node is fixed to 512; the cost constraint *ccont* of maintenance and communication varies in the range [$0.4C_{min}$, $0.8C_{min}$], where C_{min} is the maintenance cost which allows all *nmv* materialized views can be selected.

(2) The number *nq* of queries on the computation node is fixed to 100; the cost constraint *ccont* of maintenance and communication is fixed to $0.6C_{min}$; the number *nmv* of materialized views on each storage node varies in the range [16, 256].

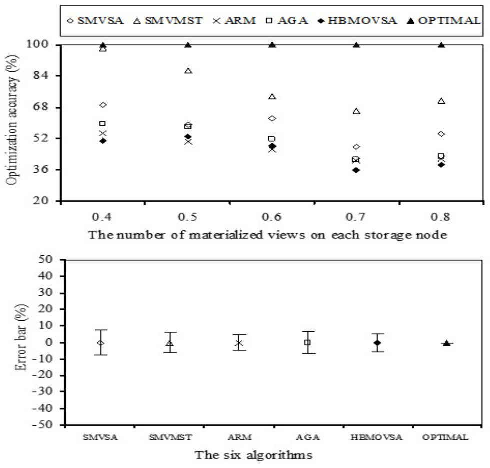
(3) The number *nmv* of materialized views on each storage node is fixed to 512; the cost constraint *ccont* of maintenance and communication is fixed to $0.6C_{min}$; the number *nq* of queries on the computation node varies in the range [50, 90].

5.2 Optimization Accuracy Evaluation for The Algorithms

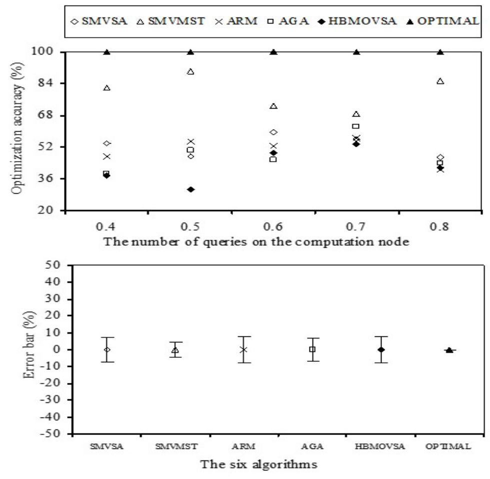
In this subsection, we experimentally evaluate the optimization accuracy of the above six algorithms. Figure 4 shows the results of experiments for these algorithms.



(a) the first group of experiments



(b) the second group of experiments



(c) the third group of experiments

Figure 4. Optimization accuracy evaluation for six algorithms

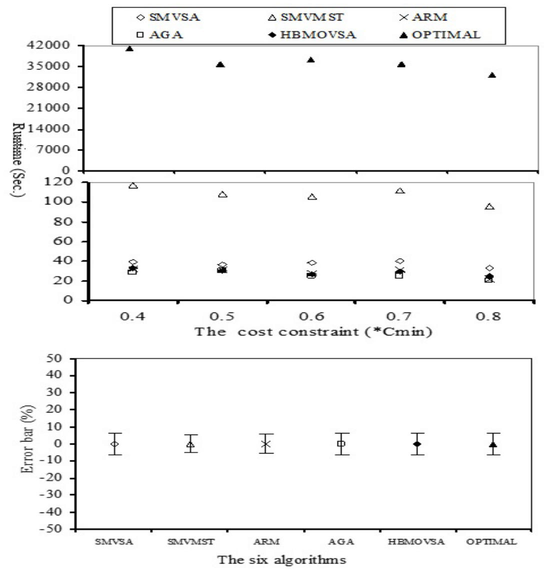
In Figure 4, we take OPTIMAL as the baseline because the set of materialized views selected by OPTIMAL is exactly optimal. That is, the query processing cost of OPTIMAL is least among 6 compared algorithms. And we let the optimization accuracy of OPTIMAL is 100%.

From Figure 4, we can observe that the optimization accuracy of SMVMST approaches the one of OPTIMAL, and the optimization accuracies of SMVSA, ARM, AGA and HBMOVSA are smaller than the one of SMVMST. This is mainly because the algorithms SMVSA, ARM, AGA and HBMOVSA easily fall into the problem of local optimum, and can not select the better set of materialized views. Furthermore, we can see that SMVSA outperforms ARM, AGA and HBMOVSA in most cases. For instance: (i) In Figure 4(a), when the cost constraint of maintenance and communication equals $0.8C_{min}$, the optimization accuracies of SMVMST and SMVSA are equal to 95.2% and 71.5% respectively, while the optimization accuracies of ARM, AGA and HBMOVSA are equal to 53.9%, 50.8% and 56.5% respectively. (ii) In Figure 4(b), when the number of materialized views on each storage node is equal to 16, the optimization accuracies of SMVMST and SMVSA are 98.2% and 69.4% respectively, while the optimization accuracies of ARM, AGA and HBMOVSA are equal to 54.8%, 59.5% and 50.6% respectively. (iii) In Figure 4(c), as the number of queries on the computation node equals 90, the optimization accuracies of SMVMST and SMVSA are equal to 85.4% and 46.8% respectively, while the optimization accuracies of ARM, AGA and HBMOVSA are 40.5%, 43.6% and 41.7% respectively.

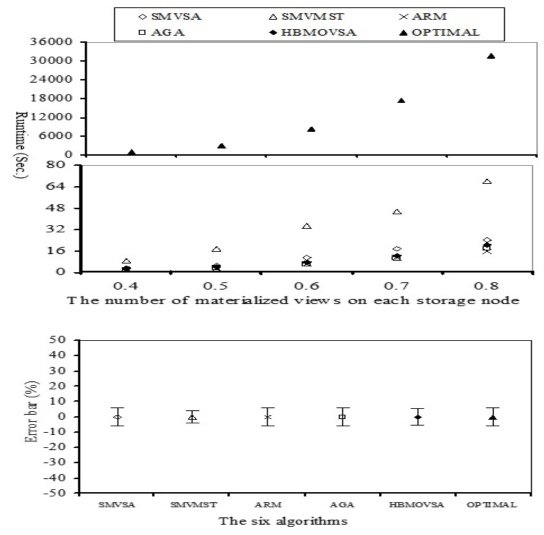
5.3 Runtime Evaluation for The Algorithms

In this subsection, we experimentally evaluate the runtime of six compared algorithms. Figure 5 shows the results of experiments for these algorithms.

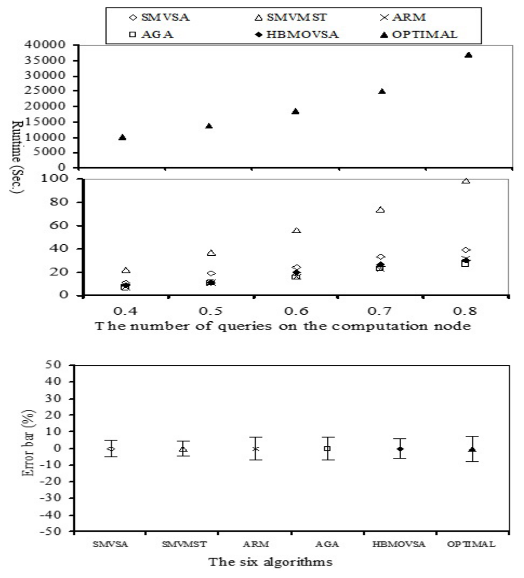
Although in Figure 4, the optimization accuracy of OPTIMAL is slightly higher than the one of SMVMST. While in Figure 5, we can observe that the runtime of OPTIMAL is huge in each experimental setting. The main reason is: (1) in order to obtain the exact optimal set of materialized views, OPTIMAL must traverse all possible combinations of materialized views, and needs the exponential time cost to finish the whole process of selection; (2) while SMVMST does not need to traverse all possible combinations of materialized views, and only has polynomial time cost to return the approximate optimal set of materialized views. Moreover, from Figure 5, we can also observe that the runtime of SMVMST is slight longer than the ones of other four algorithms SMVSA, ARM, AGA and HBMOVSA. For instance: (i) In Figure 5(a), as the cost constraint of maintenance and communication equals $0.4C_{min}$, the runtime of OPTIMAL is 41082.6 seconds, while the runtimes of SMVMST, SMVSA,



(a) The first group of experiments



(b) The second group of experiments



(c) The third group of experiments

Figure 5. Runtime evaluation for six algorithms

ARM, AGA and HBMOVSA are 117.3 seconds, 39.2 seconds, 32.6 seconds, 28.4 seconds and 33.5 seconds respectively. (ii) In Figure 5(b), as the number of materialized views on each storage node equals 256, the runtime of OPTIMAL is 31638.4 seconds, while the runtimes of SMVMST, SMVSA, ARM, AGA and HBMOVSA are 68.4 seconds, 24.2 seconds, 16.1 seconds, 17.5 seconds and 20.5 seconds respectively. (iii) In Figure 5(c), when the number of queries on the computation node equals 90, the runtime of OPTIMAL is 36817.6 seconds, while the runtimes of SMVMST, SMVSA, ARM, AGA and HBMOVSA are 98.4 seconds, 39.6 seconds, 31.6 seconds, 26.5 seconds and 30.1 seconds respectively.

Hence, from the experimental result in Figures 4 and Figure 5, we can draw the conclusions:

(1) OPTIMAL has the best optimization accuracy, but it needs huge runtime.

(2) SMVSA, ARM, AGA and HBMOVSA can quickly select the set of materialized views, but they cannot commendably guarantee the optimization accuracy.

(3) SMVMST can efficiently balance the optimization accuracy and runtime, and has good extendibility.

6 Conclusion and Future Work

Materialized views selection has recently received increasing attention in the database community. The research and implementation of materialized views selection in SPA distributed networks under the cost constraint of maintenance and communication is of great meaning. Aiming at the problem that the existing algorithms of materialized views selection cannot efficiently guarantee the optimization accuracy, we propose two efficient algorithms in SPA distributed networks based on the map/reduce distributed computation model to quickly select the optimal materialized views. The first algorithm SMVSA first heuristically constructs the initial set of materialized views and then adjusts the set of materialized views based on simulated annealing. While the second algorithm SMVMST reduces the materialized views selection to the problem of producing the minimum Steiner tree, and obtains the approximate optimal set of materialized views. The detailed theoretical analyses and extensive experiments demonstrate that our proposed algorithms are both efficient and effective.

Future work will focus on designing more exact cost evaluation model, improving the optimization accuracy of our two algorithms, and on more experimentation.

Acknowledgments

This work is supported by the Shanghai Rising-Star Program (No. 15QA1403900), the National Natural Science Foundation of China (No. 61772366), the Natural Science Foundation of Shanghai (No. 17ZR1445900), the Fok Ying-Tong Education Foundation (142002) and the Fundamental Research Funds for the Central Universities.

References

[1] A. Roukh, L. Bellatreche, A. Boukorca, S. Bouarar, Eco-DMW: Eco-design Methodology for Data Warehouses, *DOLAP*, Melbourne, Australia, 2015, pp. 1-10.

[2] Y. Zhang, S. Zhang, The Power Big Data Applications for Intelligent Community in Smart Grid, *Journal of Internet Technology*, Vol. 17, No. 7, pp. 1491-1500, July, 2016.

[3] K. Aouiche, P. E. Jouve, J. Darmont, Clustering-based Materialized View Selection in Data Warehouses, *ADBIS*, Thessaloniki, Greece, 2006, pp. 81-95.

[4] G. Gou, J. X. Yu, H. Lu, A* Search: An Efficient and Flexible Approach to Materialized View Selection, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 36, No. 3, pp. 411-425, August, 2006.

[5] X. Ji, L. Liu, P. Zhao, D. P. Wang, A-Star Algorithm Based on-Demand Routing Protocol for Hierarchical LEO/MEO Satellite Network, *Big Data*, Santa Clara, CA, 2015, pp. 1545-1549.

[6] T. V. V. Kumar, S. Kumar, Materialized View Selection Using Genetic Algorithm, *Contemporary Computing*, Vol. 306, pp. 225-237, August, 2012.

[7] M. Šetinc, M. Gradišar, L. Tomat, Optimization of a Highway Project Planning Using a Modified Genetic Algorithm, *Optimization*, Vol. 64, No. 3, pp. 687-707, March, 2015.

[8] P. R. Vishwanath, S. Reddy, An Association Rule Mining for Materialized View Selection and View Maintenance, *International Journal of Computer Applications*, Vol. 109, No. 5, pp. 15-20, May, 2015.

[9] C. W. Lin, W. Gan, P. Fournier-Viger, T. P. Hong, V. S. Tseng, Weighted Frequent Itemset Mining over Uncertain Databases, *Applied Intelligence*, Vol. 44, No. 1, pp. 232-250, January, 2016.

[10] R. N. Jogekar, A. Mohd, Design and Implementation of Algorithms for Materialized View Selection and Maintenance in Data Warehousing Environment, *International Journal of Emerging Technology and Advanced Engineering*, Vol. 3, No. 9, pp. 2250-2459, September, 2013.

[11] D. Yu, W. Dou, Z. Zhu, J. Wang, Materialized View Selection Based on Adaptive Genetic Algorithm and Its Implementation with Apache Hive, *International Journal of Computational Intelligence Systems*, Vol. 8, No. 6, pp. 1091-1102, June, 2015.

[12] T. V. V. Kumar, B. Arun, Materialized View Selection Using HBMO, *International Journal of System Assurance Engineering and Management*, Vol. 8, No. 1, pp. 379-392, January, 2017.

[13] A. Zaretalab, M. Teimouri, S. T. A. Niaki, M. Sharifi, An Efficient Memory-based Electromagnetism-like Mechanism

for the Redundancy Allocation Problem, *Applied Soft Computing*, Vol. 38, No. C, pp. 423-436, September, 2015.

[14] D. Yao, A. Abulizi, R. Hou, An Improved Algorithm of Materialized View Selection within the Confinement of Space, *BDCloud*, Dalian, China, 2015, pp. 310-313.

[15] Z. Huang, S. J. E, J. Zhang, B. Zhang, Z. Ji, Pairwise Learning to Recommend with Both Users' and Items' Contextual Information, *IET Communications*, Vol. 10, No. 16, pp. 2084-2090, March, 2016.

[16] Z. Huang, J. Wang, B. Zhang, Efficient Optimization for L-extSKY Recommendations, *Tehnicki vjesnik/Technical Gazette*, Vol. 22, No. 5, pp. 1099-1106, October, 2015.

[17] L. Kang, X. Zhu, A Simulated Annealing Algorithm for First Train Transfer Problem in Urban Railway Networks, *Applied Mathematical Modelling*, 419-435, January, 2016.

[18] J. Byrka, F. Grandoni, T. Rothvoss, L. Sanità, Steiner Tree Approximation via Iterative Randomized Rounding, *Journal of the ACM*, Vol. 60, No. 1, pp. 102-110, January, 2013.

[19] M. C. Hung, M. L. Huang, D. L. Yang, N. L. Hsueh, Efficient Approaches for Materialized Views Selection in a Data Warehouse, *Information Sciences*, Vol. 177, No. 6, pp. 1333-1348, March, 2007.

Biographies

Zhenhua Huang received his Ph.D. degree in computer science from Fudan University China, in 2008. He is currently a professor in the School of Computer Science at South China Normal University. His research interests include big data analysis and konwledge discovery. Since 2004, he has published over 70 papers.

Zhenqi Zhao received the B.S. degree in the College of Electronics and Information Engineering at the Tongji University, 2015. He is currently working toward the master's degree with Computer Science. His main research interests include database and machine learning.

Jiujun Cheng received his Ph.D. degree from Beijing University of Posts and Telecommunications in 2006. He is presently a professor of Tongji University, Shanghai, China. He has over 40 publications including conference and journal papers. His research interests are in mobile computing and cloud computing.