Deterministic View Selection For Data-Analysis Queries: Properties and Algorithms

Rong Huang[1], Rada Chirkova[2], and Yahya Fathi[1]

[1] Operations Research Program, NC State University, Raleigh, NC 27695, {rhuang,fathi}@ncsu.edu
[2] Computer Science Department, NC State University, Raleigh, NC 27695, chirkova@csc.ncsu.edu

Abstract. Materialized views are widely used in many data-intensive systems to accelerate the processing of complex data-analysis queries. In this paper we study the following *deterministic view selection problem* (*DVS*): Given a collection of queries on a given star-schema data warehouse, and a storage limit on the total size of the views that we may materialize, select a collection of views to materialize so as to minimize the total evaluation cost of the given queries. We characterize the structural properties of potentially beneficial views in relation to the given queries in the context of an integer programming (IP) model of this problem. We propose a procedure to effectively prune the search space of potentially beneficial views and reduce the size of the corresponding IP model, which enable us to obtain optimal solutions of realistic-size instances of problem DVS. We also present two heuristic methods to further reduce the search space of views to efficiently obtain competitive inexact solutions for large-size instances of the problem. We evaluate the effectiveness of our proposed techniques through a comprehensive computational study. We also present a computational experiment to compare our methods to those proposed in Asgharzadeh et al. [3,4]. We show that our approaches compare favorably with those in [3,4]. In addition, our proposed techniques in this paper are complementary to those in [3,4] and can be employed either separately or in conjunction with these earlier techniques.

1 Introduction

Data-analysis queries are widely used in data-intensive systems, such as commercial or scientific data warehouses, which store vast collections of data, whose scale tends to grow massively over time. Answering typical data-analysis queries in such systems may involve heavy use of summarization of large volumes of stored data [8, 9], and thus tends to be complex and time consuming. *Materialized views*, that is precomputed and stored extra relations, are commonly used to reduce the evaluation costs of data-analysis queries in relational data intensive systems. Intuitively, a materialized view would improve the efficiency of evaluating a query when the view relation represents the result of (perhaps time-consuming) precomputation of some subexpression of the query of interest;

please see [20] and references therein. As such, materialized views with grouping and aggregation may be especially attractive for evaluating data-analysis queries, because the relations for such views store in compact form the results of (typically expensive) preprocessing of large amounts of data.

Consider an illustration. Large retailer companies, such as Sears in the USA, maintain significant-size databases storing information about ongoing point-of-sale transactions (Pos). For accounting, reporting, and business-intelligence purposes, Sears database system undergoes periodic runs of data-analysis queries on the stored information, including the queries for automatically or manually generated daily, weekly, or monthly summary reports. For instance, the query workload contains the following SQL queries `Q1` and `Q2`. `Q1` asks for the total sales per item category per customer type in December 2011. `Q2` asks for the maximum sales per item category per store in December after 2005. [3]

```
Q1: SELECT itemCategory, custType, SUM (amount)
    FROM Pos WHERE month = ‘December’ AND year = 2011
    GROUP BY itemCategory, custType;

Q2: SELECT itemCategory, storeID, MAX (amount)
    FROM Pos WHERE month = ‘December’ AND year > 2005
    GROUP BY itemCategory, storeID;
```

The following view `V`, which stores total sales (`V1amt`) and maximum sales (`V2amt`) per item category per customer type per store per month per year, can be used to provide the exact answers to both `Q1` and `Q2`.

```
V: SELECT itemCategory, custType, storeID, SUM (amount) AS V1amt,
          MAX (amount) AS V2amt
   FROM Pos GROUP BY itemCategory, custType, storeID, month, year;
```

The cost of evaluation of queries `Q1` and `Q2` can be reduced significantly by using view `V`, as it avoids accessing a large amount of irrelevant data in the base relation (`Pos`).

Ideally in the data-analysis setting, in order to maximize the efficiency of query processing, all the “beneficial views” would be pre-computed and stored (materialized). However, the amount of storage space and computational constraints limit the beneficial views that can be materialized. Naturally, the problem of selecting an appropriate collection of materialized views has to be addressed in the context of the objectives and limitations of that setting. This problem is commonly known as the *View Selection Problem.* In recent years a number of researchers have addressed the subject and developed exact and inexact methods for solving the view-selection problem in a deterministic environment where all queries are assumed to be known and given in advance. See,

[3] To greatly simplify this example, we assume that the data in the Sears database are stored in a single relation `Pos`, with the attributes as named in the query `Q1`. The approach that we propose in this paper is applicable to the practical setting where one or more stored relations form the *star schema* [9].

for instance, [3–5, 12, 15]. In this paper we build on the work of Asgharzadeh et al. [3, 4] who solve the problem under a space limit on the materialized views, and propose new techniques for solving the problem which are more effective than the existing methods. The techniques we propose in this paper for reducing the search space of views are complementary to those proposed in [3, 4] and can be employed either separately or in conjunction with those in [3, 4].

The specific contribution of this paper is as follows:

1. We study the structure and properties of views and queries in the integer-programming model for the view-selection problem. We develop an algorithm that effectively reduces the search space of potentially beneficial views in this IP model, and obtain a smaller IP model whose solution is guaranteed to be optimal for the original problem.
2. We present a computational experiment on our smaller IP model, and discuss the scalability of this model. The size of the IP model is significantly reduced, so that for realistic-size instances of the problem this IP model can be solved efficiently by a commercial IP solver, such as CPLEX [16].
3. We define the cost-benefit ratio of each view which is a measure of *effectiveness* of the view with respect to the given collection of queries. We conduct a theoretical analysis of the properties of the cost-benefit ratio. Based on these properties, we develop two heuristic methods to further reduce the size of the resulting IP models to manageable levels, although we can no longer guarantee that the resulting solution is optimal for the original problem.
4. We present a computational experiment to evaluate the effectiveness of the proposed heuristic methods. The size of the models are significantly reduced to a manageable level even for large-size instances, so that these models can be solved by CPLEX to provide close optimal solution for the original problem. We compare the performance of our proposed heuristic methods with the heuristic algorithms in [3, 4].

The remainder of this paper is organized as follows. We review related work in Section 1.1. In Section 2, we discuss the formulation and settings for the deterministic view-selection problem, and introduce the IP model. In Section 3, we discuss the properties of the queries and views. In Section 4 we propose (exact) approaches to reduce the size of the IP model, while maintaining that the resulting optimal solution of the IP model is also optimal for the original problem. Section 5 contains the computational results for the exact approaches. In Section 6 we propose (heuristic) methods for further reducing the size of resulting IP model, but we no longer guarantee optimality. In Section 7 we conduct a computational experiment for the heuristic methods and report our findings. Finally, Section 8 contains a few concluding remarks.

1.1 Related work

The problem of view selection has been studied in literature with various objectives and constraints for a number of years (see [23] for a survey and the design goals of the problem). One line of the past research consider the view-selection under a deterministic environment where all queries are assumed to be known and given in advance. Numerous algorithms are proposed for this problem (see [14] for a survey). Significant work has also been done on index selection in such settings, both on its own and alongside view selection, please see [1, 2, 6, 7, 10]. Notable work including [13, 15] considers greedy algorithms for efficiently selecting views (with or without) indexes in a generalization of the OLAP setting. Unfortunately, the paper [18] disproves the strong performance bounds of these algorithms, by showing that the underlying approach of [15] cannot provide the stated worst-case performance ratios unless P=NP.

Considerable work including [3–5, 21, 25] has been done in the open literature that employ integer linear programming (ILP) models to obtain the optimal selection of derived data for query processing. In particular, a line of past work [3–5] has focused on formal approaches to selection of views (with or without indexes) to minimize the cost of query processing under storage-space constraint. The results of that work are scalable to realistically large numbers of queries and views, and compare beneficially with several other approaches in the literature including [2, 15, 17], please see [3–5] for the details. In this paper our proposed techniques and IP models are complementary to those proposed in [3, 4] (see Section 2.2 for an overview), and can be employed either separately or in conjunction with these earlier techniques.

2 Background

In this section, we define the scope of the view-selection problem that we consider, that is, the type of the database, queries and views. We also briefly review the models and algorithms in [3,4], based on which, we build our proposed models and methods.

2.1 Problem specification

We consider a star-schema data warehouse [8, 9] with a single fact table and several dimension tables. We assume that all the views to be materialized are defined, with grouping and aggregation but without selection, on the relation (which we call the *raw-data view*) that is the result of the "star-schema join" [19] of all the relations in the schema. We can show formally that for each query posed on the original database, the query can be rewritten equivalently into a query posed on the raw-data view. Using this formal result, in the remainder of the paper we assume that all the queries in the workloads that we consider are posed on the relevant raw-data view. In this context, we consider the evaluation costs of answering unnested select-project-join queries with grouping and aggregation using unindexed materialized views, such that each query can be evaluated using just one view and no other data. (This setting is the same as in [4, 15, 17, 22, 25].) A query q can be answered using a view v only if the set of grouping attributes of v is a superset of the set of attributes in the `GROUP BY` clause of q and of those attributes in the WHERE clause of q that are compared with constants. We use v to represent both a view and the collection of grouping attributes for that view, and we use q to represent both a query and the collection of attributes in the GROUP BY clause of that query plus those attributes in the WHERE clause of the query that are compared with constants. It follows that query q can be answered by view v if and only if $q \subseteq v$. To evaluate a query using a given view (if this view can indeed be used to answer the query) we have to scan all rows of the view. Hence the corresponding evaluation cost is equal to the size of the view itself; similar cost calculation is used in [4, 15, 17, 22, 25]. One way to estimate the view sizes in practice, as suggested in the literature, is by getting a relatively small-size sample of the raw-data view and by then evaluating the view definitions on that table, with a subsequent scaleup of the sizes of the resulting relations. We use a_i to denote the size of each view v_i in the problem input. We also use the parameter d_{ij} to denote the evaluation cost of answering query q_j using view v_i. It follows that for each query q_j we have $d_{ij} = a_i$ if $q_j \subseteq v_i$, and we set $d_{ij} = +\infty$ otherwise, implying that q_j cannot be answered by view v_i.

We consider the following problem, which we call the Deterministic View-Selection (DVS) problem: Given a collection Q of queries on a given star-schema data warehouse $\mathcal{D}$, and a storage limit b on the total size of the views that we may materialize, select a collection of views to materialize so as to minimize the total evaluation cost of the given queries.

The *search space of views* that we consider for a given problem DVS is the *view lattice* introduced by Harinarayan et al. in [15], which includes all the views defined on the raw-data table, such that each view has aggregation on all the attributes aggregated in the input queries. In the view lattice, each node represents a view, and a directed edge from node v_1 to node v_2 implies that v_1 is a parent of v_2, that is, v_2 can be obtained from v_1 by aggregating over one attribute of v_1. We illustrate it by the following example.

Example 1. Given a database with four attributes a, b, c and d, we assume that the input query set Q consists of seven queries $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, where $q_1 = \{b\}$, $q_2 = \{c\}$, $q_3 = \{a, b\}$, $q_4 = \{a, c\}$, $q_5 = \{b, c\}$, $q_6 = \{b, d\}$ and $q_7 = \{c, d\}$. The view lattice is shown in Figure 1. The space requirement in the number of bytes for each view in the lattice is given next to its corresponding node. In this instance, we assume the total space limit $b = 30$. Our objective is to minimize the cost of answering Q by materializing a set of views S with the total size less than or equal to 30.

2.2 An integer programming model

Asgharzadeh et al. [3, 4] propose an integer programming (IP) model for solving the deterministic view-selection problem as we defined in Section 2.1. This IP model has a key role in our discussions below, hence, for completeness, we present it here. Let V denote the search space of views defined in Section 2.1 for a given problem DVS with a query set Q. Let I and J denote the set of subscripts

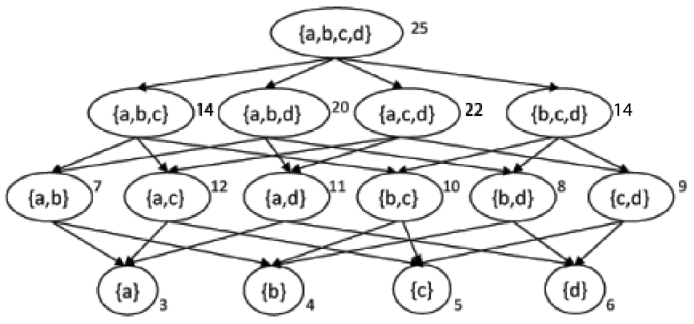

Fig. 1. View lattice for Example 1, with view sizes shown as number of bytes

associated with V and Q, respectively. Define the decision variables x_i and z_{ij} for all $j \in J$ and for all $i \in I$, as follows:

$$x_i = \begin{cases} 1 & \text{if view } v_i \text{ is materialized} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \\ 0 & \text{otherwise} \end{cases}$$

The problem DVS can now be formulated as the following integer programming (IP) model.

$$(IP1) \quad \text{minimize} \quad \sum_{j \in J} \sum_{i \in I} d_{ij} z_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i \in I} z_{ij} = 1 \qquad \forall j \in J \tag{2}$$

$$z_{ij} \leq x_i \qquad \forall j \in J, \forall i \in I \tag{3}$$

$$\sum_{i \in I} a_i x_i \leq b \tag{4}$$

$$\textit{All variables are binary} \tag{5}$$

Constraint (2) states that each query is answered by exactly one view; constraint (3) guarantees that a query can be answered by a view only if the view is materialized. Constraint (4) limits the storage space for the views to be materialized.

Asgharzadeh et al. [3, 4] propose to reduce the size of the search space of views (that is, prune the search space of views) from the view lattice V to a smaller subset based on the following two observations:

Observation 1 *A view v is not a candidate to be selected in the optimal collection of views, and hence it can be removed from the search space of views, if it contains at least one attribute that is not in any of the queries it can answer*

Observation 2 *A view v is not a candidate to be selected in the optimal collection of views, and hence it can be removed from the search space of views, if it is not equal to any query in the given query set Q, and its size is greater than or equal to the total size of queries it can answer.*

As mentioned in Asgharzadeh et al. [3, 4], these observations allow us to remove a certain number of views at the outset, thus reducing the size of the corresponding model $IP1$. We refer to this smaller model as $IP1'$. Of course, we can still guarantee that the optimal solution of the model $IP1'$ is also optimal for the original problem DVS. This, in turn, allows us to solve larger instances of the problem and obtain the corresponding optimal solutions.

Through a comprehensive computational study, Asgharzadeh et al. [3,4] show the effectiveness of this approach in solving relatively large instances of problem DVS. They also propose several heuristic techniques to further reduce the search space of views and the size of the corresponding IP model, hence allowing even larger instances of the problem to be solved in this manner, albeit they can no longer guarantee that the resulting solution is optimal for the original problem.

In this paper we study the structural relationships between the given query set Q and each view v in the view lattice V, and use this relationship to propose effective techniques for further pruning the search space of views and reducing the size of the corresponding IP model. Our proposed techniques for pruning the search space of views are complementary to those proposed in Asgharzadeh et al. [3,4] and can be employed either separately or in conjunction with these earlier techniques. We evaluate the effectiveness of our proposed techniques through a comprehensive computational study, and report our findings.

3 Properties of views and queries

In this section we study the structural properties of the views in relation to a given query set Q in the context of the problem DVS. These properties form the basis upon which we subsequently devise appropriate strategies to prune the search space of views.

3.1 Maximum benefit

Given a problem DVS with a query workload Q, for each view v in the view lattice V, let $Q(v)$ denote the set of queries in Q that v can answer, that is, $Q(v) = \{q \in Q : q \subseteq v\}$.

Definition 1. *For each subset Q' of $Q(v)$, we define the* benefit *of view v over Q' as the amount of space that we can save by materializing view v instead of materializing all queries in Q'. We refer to this benefit as $d(v, Q')$.*

From this definition, it follows that the benefit $d(v, Q')$ is equal to the difference between the total size of queries in Q' and the size of v, that is,

$$d(v, Q') = \sum_{q \in Q'} S(q) - S(v) \tag{6}$$

where $S(\cdot)$ denotes the size of a view (or query).

Definition 2. *Given the input query set Q, for each view v in V, the* maximum benefit *of v over Q is defined as the amount of space that we can save by materializing view v instead of materializing* all *input queries that v can answer, that is, $d(v, Q(v))$.*

From this definition, Observation 2 in Section 2.2 can also be expressed as follows: *A view v is not a candidate to be selected in the optimal collection of views, and hence it can be removed from the search space of views, if it is not equal to any query in the given query set Q, and the maximum benefit of v over Q is non-positive.*

We now make the following observation based on the relationship between the maximum benefit of each view in the view lattice.

Observation 3 *Given a view v in V, if there exists a view v' in V such that $v' \subset v$ and the maximum benefit of v' over Q is greater than or equal to that of v over Q, that is, if $d(v', Q(v')) \geq d(v, Q(v))$, then there exists an optimal solution for problem DVS in which v is not materialized.*

Proof. Assume that there exists a view v' that satisfies the associated condition. By definition, $d(v', Q(v')) = \sum_{q \in Q(v')} S(q) - S(v')$ and $d(v, Q(v)) = \sum_{q \in Q(v)} S(q) - S(v)$. Since $d(v', Q(v')) \geq d(v, Q(v))$, we have that

$$\sum_{q \in Q(v')} S(q) - S(v') \geq \sum_{q \in Q(v)} S(q) - S(v)$$

Hence,

$$S(v) \geq S(v') + \sum_{q \in Q(v)} S(q) - \sum_{q \in Q(v')} S(q)$$

Since $v' \subset v$, we have that $\{q \in Q : q \subseteq v'\} \subseteq \{q \in Q : q \subseteq v\}$. This indicates that $\sum_{q \in Q(v)} S(q) - \sum_{q \in Q(v')} S(q) = \sum_{q \in Q(v) \setminus Q(v')} S(q) \geq 0$. Thus, we have that

$$S(v) \geq S(v') + \sum_{q \in Q(v) \setminus Q(v')} S(q) \tag{7}$$

If v is materialized in an optimal solution for problem DVS, we can replace it by materializing the view v' and the view set V', which is the set of views corresponding to the queries in the set difference of $Q(v)$ and $Q(v')$ (that is, $Q(v) \setminus Q(v')$). According to Equation 7, the size of view v is greater than or equal to the total size of the new views that we materialize. Thus, the space limit is not violated. All the queries that are assigned to v can be answered by v' or by some view in V'. The overall cost of the new solution is no higher than the previous one. Thus, the new solution remains optimal for the problem DVS. The result follows.

Example 1 (Continued). *Compare the views $v' = \{a, c\}$ and $v = \{a, c, d\}$ in V. The view $\{a, c\}$ answers queries $\{c\}$ and $\{a, c\}$ in Q. The view $\{a, c, d\}$ answers queries $\{c\}$, $\{a, c\}$, and $\{c, d\}$ in Q. We have that $d(v', Q(v')) = 5 + 12 - 12 = 5$ and $d(v, Q(v)) = 5 + 12 + 9 - 22 = 4$. Observe that v' is a subset of v, and that $d(v', Q(v')) > d(v, Q(v))$. Thus, instead of materializing view $\{a, c, d\}$, we can materialize views $\{a, c\}$ and $\{c, d\}$, and then use view $\{a, c\}$ to answer the queries $\{c\}$ and $\{a, c\}$, and use view $\{c, d\}$ to answer query $\{c, d\}$. The new cost of answering the queries $\{c\}$, $\{a, c\}$, and $\{c, d\}$ does not exceed the original cost. Thus, view $\{a, c, d\}$ can be eliminated from the search space of views.*

In Section 4, we build on this observation to further reduce the search space of views for the problem DVS.

3.2 Cost-benefit ratio

In this section we introduce a measure of effectiveness associated with each view with respect to a set of queries in the context of the problem DVS. Later in Section 6 we use this measure to devise effective inexact (heuristic) methods for solving the problem.

First, we define the "extra cost" of view v over a collection of queries Q', for each view v and for each subset Q' of $Q(v)$.

Definition 3. *For every view v and for every subset Q' of $Q(v)$, the* extra cost *of view v over the query set Q' is defined as the difference between the cost of answering the queries in Q' using view v and the cost of answering these queries using their respective equivalent views (that is, using views $v = q$, for all $q \in Q'$). We refer to this extra cost as $c(v, Q')$. Equivalently we have that*

$$c(v, Q') = \sum_{q \in Q'} \big(S(v) - S(q)\big) \tag{8}$$

where $S(\cdot)$ refers to the size of the view (or query).

As introduced in Section 3.1, for each view v and for each subset Q' of $Q(v)$, we have already defined $d(v, Q')$, the *benefit* of view v over the query set Q', as the amount of space saved by materializing the view v instead of all the queries in Q'. Note that if a view v is not equal to any query in Q and v is selected for answering a query set Q' in the optimal solution, then the benefit of v over Q' must be positive, that is, $d(v, Q') > 0$. (Otherwise, it is obviously feasible and less costly to answer these queries using their respective equivalent views.) For each view v in V, we refer to any subset Q' of $Q(v)$ with positive benefit value, that is, $d(v, Q') > 0$, as a *Positive Subset of* $Q(v)$.

We define a subset $\widetilde{V}$ of V by excluding from V, i) each view that is equal to some query in Q, and ii) each view that has a non-positive value of maximum benefit over Q, that is,

$$\widetilde{V} = \{v \in V : v \notin Q \text{ and } d(v, Q(v)) > 0\} \tag{9}$$

The following definition is only for views in the set $\widetilde{V}$.

Definition 4. *For each view v in $\widetilde{V}$ and for any positive subset Q' of $Q(v)$, we define the* cost-benefit ratio *of view v with respect to Q' (or simply, the cost-benefit ratio of view v over Q') as the ratio of the* extra cost *over the* benefit *of view v over Q' as defined above. We denote this ratio by $r(v, Q')$. From Definitions 1 and 3, we have that*

$$r(v, Q') = \frac{c(v, Q')}{d(v, Q')} = \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)} \tag{10}$$

The cost-benefit ratio of a view v over a query set Q' measures the extra cost incurred when we use view v to answer the queries in Q' per unit space that we save by materializing v instead of the queries in Q'.

The cost-benefit ratio as defined in Definition 4 is always well defined and non-negative. This follows from the fact that the numerator of this ratio is non-negative since for each query $q \in Q'$ we have $q \subseteq v$ and thus $S(v) - S(q) \geq 0$, and that the denominator is strictly positive since this ratio is defined only for positive subsets Q' of $Q(v)$.

The cost-benefit ratio of view v with respect to the query set Q' is an indicator of the overall value of the view v in answering the queries in Q', in term of both its "cost" and its "benefit". If the cost-benefit ratio $r(v, Q')$ is relatively small, e.g. close to 0, it implies that we pay relatively smaller "extra cost" (increased response time) for utilizing view v to answer the queries in Q' with a relatively larger "benefit" (disk space saved) obtained by materializing the view v instead of the queries in Q'. It follows that the materialization of view v is expected to be *valuable*, that is, v is favored to be selected in the collection of optimal views. If the cost-benefit ratio $r(v, Q')$ is relatively large, it indicates that the materialization of view v may not bring as much "benefit" but a large amount of "penalty" as "extra cost". Thus, the view v is not favored to be materialized.

Minimum cost-benefit ratio From the above discussion, the view with a lower cost-benefit ratio is likely to be more valuable for the problem DVS. In this subsection, we study the properties of the cost-benefit ratio $r(v, Q')$ as a function of Q', and show that this function achieves a non-negative minimum value. We then discuss an efficient procedure for obtaining this minimum value.

Given the input query set Q and the view set V for the problem DVS, for each view $v \in \widetilde{V}$ (that is, $v \notin Q$ and $d(v, Q(v)) > 0$), and for each positive subset Q' of the query set $Q(v)$ (that is $d(v, Q') > 0$), we can compute $r(v, Q')$, the cost-benefit ratio of the view v over Q'. We denote the set of all the positive subsets of $Q(v)$ as $\mathcal{PS}(Q(v))$.

Definition 5. *The* minimum cost-benefit ratio *of v over Q, denoted by $r_{min}(v, Q)$, is the minimum value of the cost-benefit ratios among all the positive subsets of $Q(v)$. Equivalently,*

$$r_{min}(v, Q) = \min_{Q' \in \mathcal{PS}(Q(v))} r(v, Q') = \min_{Q' \in \mathcal{PS}(Q(v))} \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)} \tag{11}$$

Observation 4 *Given the input query set Q and the view set V in the problem DVS, for every view $v \in \widetilde{V}$ (that is, $v \in V$, $v \notin Q$, and $d(v, Q(v)) > 0$), the minimum cost-benefit ratio $r_{min}(v, Q)$ exists and $r_{min}(v, Q) \geq 0$.*

Proof. The existence of $r_{min}(v, Q)$ follows directly from the fact that the set $\mathcal{PS}(Q(v))$ is finite, and that $r(v, Q')$ is well-defined for every $Q' \in \mathcal{PS}(Q(v))$. Since all the cost-benefit ratios are non-negative, we have that $r_{min}(v, Q) \geq 0$.

In order to determine the minimum cost-benefit ratio of each view v over the query set Q, according to its definition in Equation (11), we need to determine the cost-benefit ratio for every positive subset Q' of $Q(v)$. The computational requirement of this work is $O(2^{|Q(v)|})$. The following discussion allows us to reduce this computational requirement significantly.

For each view v, let N_v denote the number of queries in Q that v can answer. Equivalently, $N_v = |Q(v)|$. We sort the queries in the set $Q(v)$ in a non-increasing order of their sizes, that is, $S(v) \geq S(q_{(1)}) \geq S(q_{(2)}) \geq \cdots \geq S(q_{(N_v)})$; of course $Q(v) = \{q_{(1)}, q_{(2)}, \ldots, q_{(N_v)}\}$. Define $Q_{(n)}(v)$ as the collection of n largest queries in $Q(v)$, that is, $Q_{(n)}(v) = \{q_{(1)}, q_{(2)}, \ldots, q_{(n)}\}$, for $n = 1, 2, \ldots, N_v$. We define n_v as the smallest number such that $Q_{(n_v)}(v)$ is a positive subset of $Q(v)$. The following lemma follows directly from the definitions.

Lemma 1. *If $1 \leq n < n_v$, $Q_{(n)} \notin \mathcal{PS}(Q(v))$; If $n_v \leq n \leq N_v$, $Q_{(n)} \in \mathcal{PS}(Q(v))$.*

Proof. Since n_v is the smallest number such that $d(v, Q_{(n_v)}(v)) > 0$, we only need to show that $Q_{(n)}$ is a positive subset of $Q(v)$, for all $n_v \leq n \leq N_v$. If $n \geq n_v$, then we have that

$$d(v, Q_{(n)}) = \sum_{j=1}^{n} S(q_{(j)}) - S(v) \geq \sum_{j=1}^{n_v} S(q_{(j)}) - S(v) = d(v, Q_{(n_v)}) > 0$$

The inequality follows from the fact that $q_{(1)}$ through $q_{(n)}$ are sequenced in a non-decreasing order of their size.

We now have the following proposition.

Proposition 5. *Given the input query set Q and the view set V in the problem DVS, for each view v in $\widetilde{V}$, we have that*

$$r_{min}(v, Q) = \min_{n_v \leq n \leq N_v} r(v, Q_{(n)}(v)) = \min_{n_v \leq n \leq N_v} \frac{\sum_{j=1}^{n} \big(S(v) - S(q_{(j)})\big)}{\sum_{j=1}^{n} S(q_{(j)}) - S(v)} \qquad (12)$$

Proof. By Observation 4, there exists a query set $Q' \in \mathcal{PS}(Q(v))$ such that $r_{min}(v, Q) = r(v, Q')$. Let N be the number of queries in Q', that is, $N = |Q'|$. The total size of the N queries in Q' is less than or equal to the total size of the queries in $Q_{(N)}(v)$ (the N largest queries in $Q(v)$), or equivalently, $\sum_{q \in Q'} S(q) \leq \sum_{j=1}^{N} S(q_{(j)})$. It follows that $\sum_{q \in Q'} \big(S(v) - S(q)\big) \geq \sum_{j=1}^{N} \big(S(v) - S(q_{(j)})\big) \geq 0$, and $0 < \sum_{q\in Q'} S(q) - S(v) \leq \sum_{j=1}^{N} S(q_{(j)}) - S(v)$. In other words, the "extra cost" of v over Q' is greater than or equal to that of v over $Q_{(N)}(v)$, and the "benefit" of v over Q' is positive and no more than that of v over $Q_{(N)}(v)$. Thus, we have that

$$r(v, Q') = \frac{\sum_{q\in Q'} \big(S(v) - S(q)\big)}{\sum_{q\in Q'} S(q) - S(v)} \geq \frac{\sum_{j=1}^{N} \big(S(v) - S(q_{(j)})\big)}{\sum_{j=1}^{N} S(q_{(j)}) - S(v)} = r(v, Q_{(N)}(v)) \geq 0$$

In addition, from Lemma 1, we have that $n_v \leq N \leq N_v$. It follows that

$$r_{min}(v, Q) = r(v, Q') \geq r(v, Q_{(N)}(v)) \geq \min_{n_v \leq n \leq N_v} r(v, Q_{(n)}(v)) \geq r_{min}(v, Q)$$

Thus, $r_{min}(v, Q) = \min_{n_v \leq n \leq N_v} r(v, Q_{(n)}(v))$.

By this proposition, the minimum cost-benefit ratio of v over Q can be obtained by evaluating the cost-benefit ratio of v over the query set $Q_{(n)}(v)$, for $n_v \leq n \leq N_v$. Hence, the computational requirement of evaluating the minimum cost-benefit ratio can be reduced from $O(2^{|Q(v)|})$ to $O\big(|Q(v)|\big)$. We can further improve the efficiency of this evaluation by the following observation.

Proposition 6. *For each view v such that $N_v - n_v \geq 2$, consider the cost-benefit ratio of view v over $Q_{(n)}(v)$, that is, $\big\{r(v, Q_{(n)}(v)) : n_v \leq n \leq N_v\big\}$. For all n, $n_v \leq n \leq N_v - 2$, if $0 \leq r(v, Q_{(n)}(v)) < r(v, Q_{(n+1)}(v))$, then $r(v, Q_{(n+1)}(v)) < r(v, Q_{(n+2)}(v))$.*

Proof. Assume $n_v \leq n \leq N_v - 2$ and $0 \leq r(v, Q_{(n)}(v)) < r(v, Q_{(n+1)}(v))$. We have that

$$\begin{aligned}
& r(v, Q_{(n)}(v)) - r(v, Q_{(n+1)}(v)) \\
=& \frac{\sum\limits_{j=1}^{n} \big(S(v) - S(q_{(j)})\big)}{\sum\limits_{j=1}^{n} S(q_{(j)}) - S(v)} - \frac{\sum\limits_{j=1}^{n+1} \big(S(v) - S(q_{(j)})\big)}{\sum\limits_{j=1}^{n+1} S(q_{(j)}) - S(v)} \\
=& \frac{\Big(\sum\limits_{j=1}^{n} \big(S(v) - S(q_{(j)})\big)\Big)\Big(\sum\limits_{j=1}^{n+1} S(q_{(j)}) - S(v)\Big) - \Big(\sum\limits_{j=1}^{n+1} \big(S(v) - S(q_{(j)})\big)\Big)\Big(\sum\limits_{j=1}^{n} S(q_{(j)}) - S(v)\Big)}{\Big(\sum\limits_{j=1}^{n} S(q_{(j)}) - S(v)\Big)\Big(\sum\limits_{j=1}^{n+1} S(q_{(j)}) - S(v)\Big)} \\
=& \frac{S(v)}{d(v, Q_{(n)}(v)) d(v, Q_{(n+1)}(v))}\Big(S(v) - \sum_{j=1}^{n} S(q_{(j)}) + (n-1) S(q_{(n+1)})\Big)
\end{aligned}$$

Or equivalently,

$$r(v, Q_{(n)}(v)) - r(v, Q_{(n+1)}(v)) = \frac{S(v)}{d(v, Q_{(n)}(v)) d(v, Q_{(n+1)}(v))} T_n \tag{13}$$

where $T_n = \Big(S(v) - \sum_{j=1}^{n} S(q_{(j)}) + (n-1)S(q_{(n+1)})\Big)$.

It follows that by the same argument we have that

$$r(v, Q_{(n+1)}(v)) - r(v, Q_{(n+2)}(v)) = \frac{S(v)}{d(v, Q_{(n+1)}(v))d(v, Q_{(n+2)}(v))} T_{n+1} \quad (14)$$

Since $r(v, Q_{(n)}(v)) < r(v, Q_{(n+1)}(v))$, we have that $r(v, Q_{(n)}(v)) - r(v, Q_{(n+1)}(v)) < 0$. By Equation (13), $T_n < 0$. It follows that $T_{n+1} = T_n - n\big(S(q_{(n+1)}) - S(q_{(n+2)})\big) < 0$. Hence, by Equation (14), we have that $r(v, Q_{(n+1)}(v)) - r(v, Q_{(n+2)}(v)) < 0$.

Proposition 6 implies that once the function $r(v, Q_{(n)}(v))$ increases as we increase n, it will no longer decrease. Hence, we obtain the following corollary.

Corollary 1. *The function $r(v, Q_{(n)}(v))$ is a unimodal function of n for $n_v \leq n \leq N_v$. In other words, the function $r(v, Q_{(n)}(v))$ must be in one of the following three patterns:*

i) $r(v, Q_{(n)}(v))$ is a non-increasing function of n, for $n_v \leq n \leq N_v$;
ii) $r(v, Q_{(n)}(v))$ is a non-decreasing function of n, for $n_v \leq n \leq N_v$;
iii) there exists $\bar{n}$ ($n_v \leq \bar{n} \leq N_v$) such that $r(v, Q_{(n)}(v))$ is a non-increasing function of n for $n_v \leq n \leq \bar{n}$ and it is a non-decreasing function for all $\bar{n} \leq n \leq N_v$.

We can now compute the minimum cost-benefit ratio of view v over the query set Q by computing the cost-benefit ratio $r(v, Q_{(n)}(v))$ from $n = n_v$ until the minimum n, denoted by $\bar{n}$, such that $0 \leq r(v, Q_{(\bar{n})}(v)) < r(v, Q_{(\bar{n}+1)}(v))$. If there exists such a value $\bar{n}$, then $r_{min}(v, Q) = r(v, Q_{(\bar{n})}(v))$. Otherwise, $r_{min}(v, Q) = r(v, Q(v))$.

Example 1 (Continued). *Consider the views $v_1 = \{a, b, c\}$ and $v_2 = \{b, c, d\}$ in the view set $\widetilde{V}$. We can compute $r_{min}(\{a, b, c\}, Q)$ as follows.*

$$r(\{a,b,c\}, \big\{\{a,c\},\{b,c\}\big\}) = \frac{(14-12)+(14-10)}{12+10-14} = 0.75$$
$$r(\{a,b,c\}, \big\{\{a,c\},\{b,c\},\{a,b\}\big\}) = \frac{10}{16} = 0.867 > 0.75$$

Thus, $r_{min}(\{a,b,c\}, Q) = 0.75$. We can apply the same approach to obtain that $r_{min}(\{b,c,d\}, Q) = 1.333$. The values of the function $r(v, Q_{(n)}(v))$ on different values of n for $v = v_1$ and for $v = v_2$ are shown in Figure 2. We observe that $r(v_1, Q_{(n)}(v_1))$ is an increasing function of n for $2 \leq n \leq 5$, and $r(v_2, Q_{(n)}(v_2))$ decreases for $2 \leq n \leq 3$, and increases for $3 \leq n \leq 5$.

The properties of a view discussed above can help us in reducing the search space of views in two significant ways. Firstly, Observation 3 in Section 3.1, along with Observations 1 and 2 stated in Section 2.2, allow us to reduce the search space of views for a given problem DVS while we can still guarantee that

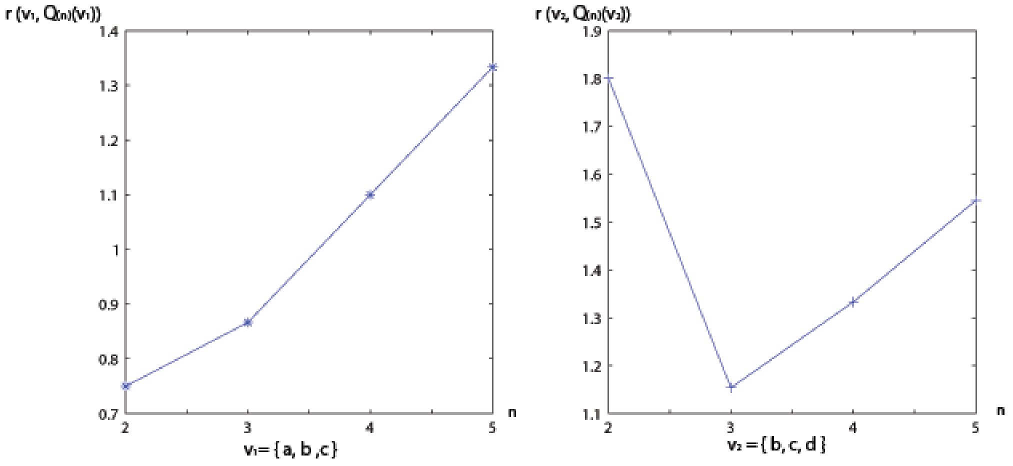

Fig. 2. The minimum cost-benefit ratios in Example 1

the reduced search space contains at least one optimal solution. We discuss this subject in Section 4 below. Secondly, Observation 4 and Propositions 5 and 6 allow us to further reduce the search space of views by keeping only those views that are likely to be effective in answering the given collection of queries. This, in turns, allows us to solve larger instances of the problem, although we can no longer guarantee that the resulting solution is optimal for the original problem. We discuss this subject in Section 6.

4 Solving the problem

As mentioned earlier, Asgharzadeh et al. [3,4] employ the results stated in Observations 1 and 2 in the context of a given problem DVS to remove certain views from the search space, thus reducing the size of the corresponding IP model $IP1$. In this section we propose to further reduce the size of the IP model by employing the results stated in Observation 3 and combining these results with those of Observations 1 and 2. Through a computational study in the next section we show empirical evidence of the effectiveness of this approach in solving relatively large instances of the problem DVS.

4.1 The reduction procedure

We reduce the search space of views, and thus the size of the corresponding IP model, by removing every view that satisfies the stated condition in at least one of the observations 1, 2, or 3. We refer to every such view as a *dominated view.* Note that the conditions stated in these observations are independent of each other, hence the results of these reductions do not depend on the order of their application. Thus, as we go through the collection of views in the set V in order to identify the dominated views, for each view we test for the conditions stated in all three observations before moving to the next view. A detailed description of the *reduction procedure* that we have devised to implement theses reductions is given below. The computational complexity of this procedure is $O(|V|^2) = O(4^K)$, where K is the number of attributes in the database.

We use the binary representation of the subscript number for each view to indicate the set of attributes for the view. For instance, in a database with 4 attributes a, b, c and d, the binary number "1001" represents the set of attributes $\{a, d\}$. Hence, the subscript number of view $\{a, d\}$ is 9 $(= 2^3 + 2^0)$. We can now conduct the reductions on views by iterating over all the subscript numbers of the corresponding views in V. The pseudocode for this procedure is given in the next page.

Algorithm 1 The reduction procedure

```
Input: the given query set $Q$ and the search space of views $V$
Output: the reduced search space of views $\overline{V}$
 1: Sort the elements of $V$ in an increasing order of the subscript numbers
 2: $\overline{V} \leftarrow \emptyset$
 3: for $i \leftarrow 1$ to $|V|$ do
 4:    $v \leftarrow i^{th}$ element of $V$
 5:    $f \leftarrow 1$
 6:    $a \leftarrow |\cup_{q \in Q(v)} q|$
 7:    $d(v, Q(v)) \leftarrow \sum_{q \in Q(v)} S(q) - S(v)$
 8:    if $|v| \neq a$ or $(v \notin Q$ and $d(v, Q(v)) \leq 0)$ then
 9:       $f \leftarrow 0$
10:    else
11:       for $j \leftarrow 1$ to $|\overline{V}|$ do
12:          $\tilde{v} \leftarrow j^{th}$ element of $\overline{V}$
13:          if $\tilde{v} \subset v$ then
14:             if $d(\tilde{v}, Q(\tilde{v})) \geq d(v, Q(v))$ then
15:                $f \leftarrow 0$
16:                break
17:             end if
18:          end if
19:       end for
20:    end if
21:    if $f = 1$ then
22:       $\overline{V} \leftarrow \overline{V} \cup \{v\}$
23:    end if
24: end for
```

In this procedure, we iterate through all the views in the view set V in the increasing order of their subscript numbers, one view in each iteration (lines 1-4). Note that if a view v answers a query (view) v', the subscript number of v is larger than the subscript number of v'. Thus, this ordering guarantees that when conducting the reductions regarding a view v, all the views that are subsets of v have already been considered in this procedure. The last view we consider in this procedure is the raw-data view. In each iteration, we use a binary variable f to indicate if view v can be eliminate. At the beginning of each iteration, we set $f = 1$ (line 5). In each iteration, we first conduct the reduction based on Observations 1 and 2 (lines 6-9). More specifically, for each view v, we compare the number of attributes in v with that in the union set of the queries in Q that v can answer. We also compare the size of view v with the total size of the queries in Q that v can answer. Note that the statements "$|v| \neq a$" and "$v \notin Q$ **and** $d(v, Q) \leq 0$" in line 8 indicate that v can be eliminated by the results of Observation 1 and by the results of Observation 2, respectively. Then we conduct the reduction based on Observation 3 by iterating through each view $\tilde{v}$ that has been selected as a candidate view and can be answered by v (lines 11-19). For each view $\tilde{v}$, we compare the maximum benefit of v over Q ($d(v, Q(v))$) with that of $\tilde{v}$ over Q ($d(\tilde{v}, Q(\tilde{v}))$). The statement "$d(\tilde{v}, Q(\tilde{v})) \geq d(v, Q(v))$" indicates the view v can be eliminated. At the end of each iteration, if the view is not eliminated by any of the results of Observations 1-3, we add it into the view set $\overline{V}$ (lines 21-23). At the beginning of the procedure, the view set $\overline{V}$ is an empty set. As a result, at the end of this procedure, $\overline{V}$ is the new search space of views.

4.2 Smaller IP model

The structure of this model is similar to model $IP1$ that we introduced in Section 2.2, except that we use the reduced search space of views $\overline{V}$ instead of V. More specifically, for each query $q_j \in Q$ we define $\overline{V_j} = \{v_i \in \overline{V} : v_i \supseteq q_j\}$. We also define the associated sets of subscripts for $\overline{V}$ and $\overline{V_j}$ as $\overline{I}$ and $\overline{I_j}$, respectively. Using this notation we now define the smaller IP model that we refer to as model $IP2$ as follows.

$$
\begin{aligned}
(IP2) \quad \text{minimize} \quad & \sum_{j \in J} \sum_{i \in \overline{I}} d_{ij} z_{ij} && (15) \\
\text{subject to} \quad & \sum_{i \in \overline{I_j}} z_{ij} = 1 \qquad \forall j \in J && (16) \\
& z_{ij} \leq x_i \qquad \forall j \in J, \forall i \in \overline{I_j} && (17) \\
& \sum_{i \in \overline{I}} a_i x_i \leq b && (18) \\
& \textit{All variables are binary} && (19)
\end{aligned}
$$

Based on the above observations, an optimal solution for model $IP2$ is guaranteed to be optimal for $IP1$. Hence, it provides an optimal solution for the original problem.

Obviously the size of model $IP2$ in terms of the number of variables and constraints, is potentially smaller than that of model $IP1$. The magnitude of the difference in size, however, depends on the specifies of each instance. In Section 5 we compare the sizes of these two models on an empirical basis and show that the difference in the sizes of the two models can be significant.

5 Experimental results with model $IP2$

In this section, we present the results of a computational experiment with the approach introduced in Section 4 for solving the problem DVS. Our objectives are (i) examining the effectiveness of our approach in reducing the search space of views proposed in Section 4, and (ii) evaluating the scalability of the model $IP2$ introduced in Section 4 and its effectiveness in solving relatively large instances of the problem DVS. We construct a collection of instances of the problem DVS with varying sizes using a number of datasets generated via the TPC-H benchmark [24]. All of our algorithms are implemented in C++ and all the experiments are carried out on a 2.66GHz Intel 2 Quad processor with 3.25 GB RAM running Windows XP Professional. We use CPLEX 11 [16] to solve the integer programming models. We observe that the search spaces of views are significantly reduced in $IP2$ compared with models $IP1$ and $IP1'$ introduced in Sections 2.2, which, in turn, allows us to use $IP2$ to solve larger instances of the problem DVS. More specifically, our experimental results show that:

- The search space of the views in model $IP2$ is significantly smaller than that of models $IP1$ and $IP1'$. The magnitude of this difference depends on the data set in each specific instance.
- Size of the model $IP2$, as measured by the number of variables and constraints, is sufficiently small to allow its use in solving relatively larger (realistic-size) instances of the problem within reasonable execution time. Of course, the resulting solution is guaranteed to be optimal for the original problem DVS.
- In some larger (realistic-size) instances of problem DVS, however, the size of the corresponding model $IP2$ is too large for current "state of art" IP solver. In the next section we propose alternative (scalable) approaches for solving such (larger) instances of the problem.

5.1 Constructing the instances

The input parameters for an instance of the problem DVS are a database $\mathcal{D}$, a query set Q, and the space limit b. In this section, we use two different datasets based on the TPC-H benchmark [24] – a 13-attribute dataset, and a 17-attribute dataset – to construct the collection of instances in our experiments. The same datasets are used in [3–5].

For each instance we generate the queries in the query set randomly. More specifically, given a database fact table (that is, stored relation) $\mathcal{D}$ with K attributes, and given two integers t_0 and $t_1 \in [1, K-1]$, we construct each query q with t attributes from the database $\mathcal{D}$, where $t \in [t_0, t_1]$. In order to randomly construct such a query q, we first determine the number of attributes in q by randomly generating an integer t between t_0 and t_1. Then, we randomly choose t distinct integers $a_1, \ldots, a_t$ from $\{1, 2, \ldots, K\}$ as the attributes of q. Then, $\{a_1, \ldots, a_t\}$ uniquely defines query q over the database $\mathcal{D}$.

Our preliminary experiments showed that the computational requirements of solving model $IP2$ in each instance depend on the relative magnitude of the storage space limits as compared with the size of the queries. In this section for all instances based on the TPC-H datasets we choose the storage space limit b equal to one fifth of the sum of the sizes of the queries in Q, that is $b = 0.2 * \sum_{q \in Q} S(q)$. Our empirical observations show that typically at this value of b the computational requirements of solving the model $IP2$ are relatively high as compared with these requirements at other values of b.

5.2 Reduction in the size of search space

We compare the size of the models $IP1$, $IP1'$ and $IP2$ for several randomly generated instances of the problem DVS. More specifically, we constructed 20 instances for the 13-attribute TPC-H dataset in such a way that for each instance the number of attribute of each query is a random number between 1 and 12. (These queries are randomly chosen from the entire view lattice.) The number of queries for each instance ranges from 20 to 100. We also construct 20 instances for the 17-attribute TPC-H dataset. For the instances 1 to 5 in this group, the number of attributes of each query is a random number between 1 and 16, (that is, each query is chosen from the entire view lattice). For the each of the remaining instances, all of the queries are from certain levels of the view lattice. For the instances 6 to 10, the number of attributes in each query is a random number between 1 and 8, (that is, each query is chosen from the bottom levels of the view lattice). For the instances 11 to 15, the number of attributes of each query is a random number between 5 and 12, (that is, each query is chosen from the middle levels of the view lattice). Finally for the instances 16 to 20, the number of attributes of each query is a random number between 9 and 16, (that is, each query is chosen from the top levels of the view lattice). We refer to the query types of those instances as "bottom-level", "middle-level", and "top-level", respectively. Within the collection of instances for each query type, the number of queries ranges from 20 to 100. For each instance, we report the number of views in models $IP1$, $IP1'$ and $IP2$. For the 20 instances on the 17-attribute dataset, these values are in Table 1. For each instance, we also report the number of variables and constraints in each model. (We do not construct model $IP1$ since [4] has already shown that $IP1'$ is much more effective than $IP1$.) In Table 2 we report the corresponding execution times. More specifically we report the time that it takes to build each model, the time that it takes to solve each model by the CPLEX IP solver, and the total execution time of each model for the instances over the 17-attribute dataset. The pattern of results for the 20 instances over the 13-attribute dataset are similar to those of the 17-attribute dataset. For brevity we do not include them here. We make the following observations.

Table 1. Comparison of the number of views and the sizes in the models $IP1$, $IP1'$ and $IP2$, for instances over the 17-attribute TPC-H dataset

instance	query type	number of queries	number of views			number of variables		number of constraints	
			$IP1$	$IP1'$	$IP2$	$IP1'$	$IP2$	$IP1'$	$IP2$
1	random	20	115,200	125	100	962	784	858	705
2		40	118,420	1,920	915	15,567	8,228	13,688	7,354
3		60	127,760	3,724	1,796	41,949	22,646	38,286	20,911
4		80	127,544	7,070	4,589	86,895	60,702	79,906	56,194
5		100	128,960	16,159	7,710	216,950	118,653	200,892	111,044
6	bottom-level	20	98,744	1,071	1,004	7,793	7,426	6,743	6,443
7		40	125,576	7,409	3,192	81,768	39,027	74,400	35,876
8		60	130,076	21,678	8,048	330,371	138,305	308,754	130,318
9		80	130,916	35,927	14,248	570,725	271,489	534,879	257,322
10		100	130,984	44,229	17,937	817,183	396,196	773,055	378,360
11	middle-level	20	15,415	301	301	1,642	1,642	1,362	1,362
12		40	27,665	1,507	1,504	9,036	9,024	7,570	7,561
13		60	38,220	3,787	3,614	26,085	25,289	22,359	21,736
14		80	34,045	4,351	4,305	29,795	29,596	25,525	25,372
15		100	42,507	7,304	7,278	55,638	55,530	48,435	48,353
16	top-level	20	1,027	89	89	378	378	310	310
17		40	1,500	234	234	1,077	1,077	884	884
18		60	1,733	316	316	1,543	1,543	1,288	1,288
19		80	3,758	669	669	3,584	3,584	2,996	2,996
20		100	3,145	687	687	3,731	3,731	3,145	3,145

Table 2. Comparison of the computing times of the models $IP1'$ and $IP2$, for instances over the 17-attribute TPC-H dataset

instance	query type	number of queries	time to build model (sec.)		time to solve model (sec.)		total time (sec.)	
			$IP1'$	$IP2$	$IP1'$	$IP2$	$IP1'$	$IP2$
1	random	20	0.30	0.30	0.14	0.11	0.44	0.41
2		40	0.41	0.39	2.86	1.56	3.27	1.95
3		60	0.66	0.64	7.47	3.47	8.13	4.11
4		80	0.81	1.13	31.77	17.17	32.58	18.30
5		100	1.33	2.05	116.05	44.15	117.38	46.20
6	bottom-level	20	0.41	0.50	1.58	1.72	1.99	2.22
7		40	0.70	0.80	104.28	32.53	104.98	33.33
8		60	1.56	2.28	>30min	385.14	>30min	387.42
9		80	3.19	6.19	out of memory	>30min	out of memory	>30min
10		100	3.11	9.42	out of memory	>30min	out of memory	>30min
11	middle-level	20	0.25	0.25	0.42	0.42	0.67	0.67
12		40	0.31	0.34	4.44	3.89	4.75	4.23
13		60	0.45	0.67	22.42	21.05	22.87	21.72
14		80	0.52	0.80	103.19	93.26	103.71	94.06
15		100	0.61	1.39	>30min	>30min	>30min	>30min
16	top-level	20	0.25	0.22	0.11	0.05	0.36	0.27
17		40	0.30	0.27	2.56	2.28	2.86	2.55
18		60	0.31	0.33	1.08	1.06	1.39	1.39
19		80	0.34	0.38	1.63	1.62	1.97	2.00
20		100	0.41	0.44	3.01	3.14	3.42	3.58

1. In every instance the number of views in the search space for models $IP1'$ and $IP2$ are significantly smaller than the corresponding number in model $IP1$. This reduction is relatively more significant for the instances with a small ratio of the number of queries over the total number of views in the dataset. More specifically, when we have a larger number of queries, the ratio of the number of queries over the total number of views increases (since the total number of views for a 17-attribute dataset is constant and equal to 131072), and the magnitude of associated reductions in the number of views in the search space decreases. We make a similar observation on the instances over the 13-attribute dataset.
2. Comparing the number of views in $IP1'$ and $IP2$, we note that for the instances where the queries are from the entire view lattice (instances 1 to 5) or from the bottom levels of the view lattice (instances 6 to 10), the size of $IP2$ is significantly smaller than that of $IP1'$. It is also observed that for these instances the size of $IP2$, when expressed by the total number of variables and constraints, is much smaller than that of $IP1'$. In addition, the time to solve $IP2$ is also significantly smaller than that for $IP1'$. On the other hand, for the instances where the queries are from the middle levels of the view lattice (instances 11 to 15) the reduction in the number of views in the search space from model $IP1'$ to $IP2$ is not as significant, resulting in no significant difference in the size and the solving time between the corresponding models $IP1'$ and $IP2$. Moreover, for the instances from the top levels of the view lattice (instances 16 to 20), we observe no reduction in the number of views in the search space from model $IP1'$ to $IP2$. These observations are consistent with our expectations as evident from the nature of Observation 3.
3. It is observed that for all instances the time to build the model $IP2$ is relatively larger than that for $IP1'$. This increase in time is much more significant for instances with large number of queries. Comparing the time to build and the time to solve the models $IP1'$ and $IP2$, however, we observe that the decrease in the time to solve the model $IP2$ compared with model $IP1'$ is much more significant than the increase in the time to build the model $IP2$ compared with model $IP1'$. In general it is also observed that the total time for $IP2$ is relatively smaller than that for $IP1'$. This reduction in time is more significant for instances where the queries are from the entire view lattice (instances 1 to 5) or from the bottom levels of the view lattice (instances 6 to 10).

5.3 Scalability of the model $IP2$

In order to evaluate the scalability of our approach we attempt to solve larger instances of the problem DVS by model $IP2$.

In Table 2 we observe that for instances 6 to 10, where the queries are from the bottom levels of the view lattice, the total execution time to solve $IP2$ ranges from 2.22 seconds to 387.42 seconds where the number of queries in each instance is no more than 60. However, we could not solve instances 9 and 10 within the assumed time limit 30 minutes.

As reported in Table 2 for instances 11 to 15 where queries are from the middle levels of the view lattice over the 17-attribute dataset, we observe that we could solve all the instances whose number of queries is no more than 80. The total execution time for $IP2$ ranges from 0.67 seconds to 94.06 seconds. However, when we increase the size of the query set further to 100, the solver fails to provide an optimal solution for model $IP2$ within our time limit of 30 minutes.

For instances where the queries are from the entire view lattice over the 17-attribute dataset (instances 1 to 5) that we report in Table 2, we note that the total execution time for the model $IP2$ ranges from 0.41 seconds to 46.20 seconds. We also note that this time increases as we increase the number of queries in the query set.

In addition, as observed in Table 2 we could solve within 5 seconds all the instances where the queries are from the top levels of the view lattice over the 17-attribute dataset (instances 16 to 20).

In order to further observe the execution time for solving larger instances of the problem where queries are from the entire view lattice, and for instances where queries are from top levels of the lattice, we constructed several instances with even larger number of queries over the 17-attribute dataset as reported in Table 3. Note that for these instances we do not construct the corresponding model $IP1'$, since we have already observed that model $IP2$ is much more effective than model $IP1'$ for these types of instances.

Table 3. Scalability of $IP2$ for instances over the 17-attribute TPC-H dataset

instance	query type	number of queries	number of views	time to build $IP2$ (sec.)	time to solve $IP2$ (sec.)	total time (sec.)
21	random	220	23256	13.00	304.00	317.00
22		240	23921	14.25	374.22	388.47
23		260	25922	15.31	out of memory	out of memory
24	top-level	320	2673	0.91	559.58	560.49
25		340	2703	1.08	814.27	815.35
26		360	2930	1.20	>30min	>30min

In Table 3, we observe that for the instances where queries are randomly chosen from the entire view lattice we could solve the instance where the number of queries is no more than 240. When we increase the number of queries to 260, CPLEX fails to obtain an optimal solution as the computer ran out of memory. For the instances where queries are from the top levels of the view lattice, we could not solve the problem within 30 minutes when we increase the number of queries to 360. In Section 6 we develop algorithms to further reduce the size of the model to solve larger instances of the problem DVS.

6 Heuristic methods

As stated earlier, Observation 4 and Propositions 5-6 allow us to further reduce the size of the search space of views by keeping only those views that are likely to be effective in answering the given collection of queries in the context of the problem DVS. In this section we propose several strategies for carrying out this task, leading to two distinct procedures that we refer to as heuristic methods I and II, respectively. We start the section by discussing the relationship between the cost-benefit ratio and the model $IP2$. Subsequently, we build on the discussion to derive two distinct heuristic methods for reducing the search space of views and the size of the corresponding IP model.

6.1 Cost-benefit ratio and the model $IP2$

In this subsection, we discuss the relationship between the cost-benefit ratio and the integer programming model $IP2$, and provide an intuitive justification as to why a view with relatively low cost-benefit ratio is more favorable to be selected in the context of the problem DVS.

We denote by $J(v)$ the set of subscripts for the queries in $Q(v)$. As introduced in Section 2.1, if view v_i can answer query q_j, then the cost of answering q_j using v_i is the size of v_i. Equivalently, we have that $d_{ij} = a_i = S(v_i)$. We now show that the objective function of the model $IP2$ can be interpreted as the sum of the *extra costs* of answering the queries using the materialized views instead of the queries themselves, as defined in Equation 8, plus a constant. To this end, we rewrite the objective function (15) of $IP2$ as follows.

$$\begin{aligned}
\sum_{j\in J}\sum_{i\in \overline{I_j}} d_{ij}z_{ij} &= \sum_{j\in J}\sum_{i\in \overline{I_j}} S(v_i)z_{ij} && (20)\\
&= \sum_{j\in J}\sum_{i\in \overline{I_j}} S(v_i)z_{ij} - \sum_{j\in J} S(q_j) + \sum_{j\in J} S(q_j) && (21)\\
&= \sum_{j\in J}\sum_{i\in \overline{I_j}} S(v_i)z_{ij} - \sum_{j\in J} S(q_j)\sum_{i\in \overline{I_j}} z_{ij} + \sum_{j\in J} S(q_j) && (22)\\
&= \sum_{j\in J}\sum_{i\in \overline{I_j}} z_{ij}\Big(S(v_i) - S(q_j)\Big) + \sum_{j\in J} S(q_j) && (23)\\
&= \sum_{i\in \overline{I}} x_i\Bigg(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Bigg) + \sum_{j\in J} S(q_j) && (24)
\end{aligned}$$

For each view v_i, we define the *extra cost* of using view v_i to answer query $q_j \in Q(v_i)$ as $\Big(S(v_i) - S(q_j)\Big)$. Using the binary variables z_{ij} as defined in the IP model, it follows that $\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)$ is the extra cost of using view v_i to answer all queries that are assigned to it, and the term

$\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i)-S(q_j)\Big)\Big)$ is the "total extra cost" of answering the queries using the materialized views, instead of using the queries themselves.

Similarly, we define the *benefit* of using view v_i to answer a collection of queries $Q' \subseteq Q(v_i)$ as the amount of disk space saved by materializing this view instead of all queries in Q'. It follows that in the context of the IP model, the benefit associated with view v_i is $\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, and the total benefit associated with a given solution is $\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$. By expanding and rewriting the terms in constraint (18) we note that

$$\begin{aligned} b-\sum_{i\in\overline{I}} a_i x_i &= \sum_{j\in J} S(q_j) - \sum_{i\in\overline{I}} a_i x_i + b - \sum_{j\in J} S(q_j) && (25)\\ &= \sum_{j\in J} S(q_j)\sum_{i\in\overline{I_j}} x_i z_{ij} - \sum_{i\in\overline{I}} a_i x_i - \Big(\sum_{j\in J} S(q_j) - b\Big) && (26)\\ &= \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) - \Big(\sum_{j\in J} S(q_j) - b\Big) && (27)\end{aligned}$$

It follows that constraint (18) can be rewritten as

$$\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) \geq \sum_{j\in J} S(q_j) - b \qquad (28)$$

In other words, the storage-limit constraint (18) could be interpreted as the "benefit constraint", that is, the "total benefit" obtained by materializing the views instead of all the queries, that is, $\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, should be greater than or equal to a given constant $\sum_{j\in J} S(q_j) - b$.

We can now rewrite the model $IP2$ as follows.

$$\begin{aligned} (IP2') \quad \min \quad & \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i)-S(q_j)\Big)\Big) + \sum_{j\in J} S(q_j) && (29)\\ s.t. \quad & \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) \geq \sum_{j\in J} S(q_j) - b && (30)\\ & \sum_{i\in\overline{I_j}} z_{ij} = 1 \qquad \forall j\in J && (31)\\ & z_{ij} \leq x_i \qquad \forall j\in J, \forall i\in\overline{I_j} && (32)\\ & \textit{All variables are binary} && (33)\end{aligned}$$

Note that the structure of model $IP2'$ is similar to the well-known 0-1 min-knapsack problem, where $\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Big)$ corresponds to the "value" of i^{th} item, and $\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$ corresponds to the "weight" of the i^{th} item. Of course, in this case the presence of z_{ij} variables and their relationship with the variables x_i (that is, Constraint (32)) are distinct characteristics that distinguish this model from the conventional knapsack problem.

Dantzig in [11] proposed a greedy approximation algorithm to solve the knapsack problem. It first sorts the items in non-decreasing order of their "value" per unit of "weight". It then proceeds to select the items in that order until the knapsack constraint is satisfied. In our problem DVS, the "cost" per unit of "weight" for each view is $\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Big)/\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, which is a function of z_{ij} rather than a constant. For each solution of the problem DVS, the "cost" per unit of "weight" for each view v is equivalent to the cost-benefit ratio of v over the query set it answers in the solution. We can potentially use this property to devise a greedy algorithm for the problem DVS by first sorting the views in non-decreasing order of their minimum cost-benefit ratios, and then proceed to select the views in the order as needed.

Alternatively, we can employ this property to limit the search space of views in the context of the model $IP2$ that we introduced in Section 4.2, hence reducing the size of this IP model and improving its scalability. To this end we can select only those views with relatively low cost-benefit ratio with respect to the query set Q, and discard all other views. Albeit by doing so we can no longer guarantee that the optimal solution of the resulting IP model would be optimal for the original view selection problem, but we can potentially solve much larger instance of the problem DVS. We discuss several strategies for carrying out this task in the next few subsections. In Section 7 we present the results of a computational experiment to evaluate the effectiveness of these strategies.

6.2 Heuristic method I: single threshold strategy

In this heuristic approach we start with the search space $\overline{V}$ in model $IP2$ introduced in Section 4.2, and remove from this set every view whose minimum cost-benefit ratio is above a certain threshold γ. For practical reasons we exempt from this process every view that is equal to one of the queries in the given query set Q (we keep these views in the search space at all times). The minimum cost-benefit ratio of each view in the set $\{v : v \in \overline{V}, v \notin Q\}$ can be obtained as discussed in Section 3.2. In order to guarantee feasibility, we also keep in the search space the view with the least number of attributes that can answer all the input queries. It is easy to show that this view, denoted by v_{max}, is the union of all the input queries, that is $v_{max} = \cup_{q\in Q}\, q$. This results in a reduced search space that we refer to as $\overline{V}^I(\gamma)$. We have

$$\overline{V}^I(\gamma) = \{v_{max}\} \cup \{\, v : v \in Q\} \cup \{\, v : v \in \overline{V}, v \notin Q, r_{min}(v, Q) \leq \gamma\}. \qquad (34)$$

Correspondingly, for each query $q_j \in Q$, we define its associated reduced set of views as $\overline{V}^I_j(\gamma) = \{v \in \overline{V}^I(\gamma) : v \supseteq q_j\}$.

We now construct an integer programming model associated with the parameter γ which is similar to model $IP2$ in Section 4.2 except that we use $\overline{V}^I(\gamma)$

instead of $\overline{V}$ as the search space of views, and the associated reduced view set $\overline{V}^I_j(\gamma)$ instead of $\overline{V}_j$ for each query $q_j \in Q$ (with corresponding subscript sets $\overline{I}^I(\gamma)$ and $\overline{I}^I_j(\gamma)$ instead of $\overline{I}$ and $\overline{I}_j$ in model $IP2$, respectively). We refer to this model as the *IP model with reduced search space of views associated with* γ, or $IPRv(\gamma)$ for short. We make the following observation.

Observation 7 *Given $\gamma_1 > \gamma_2 > 0$, we have that $\overline{V}^I(\gamma_1) \supseteq \overline{V}^I(\gamma_2)$, and $\overline{V}^I_j(\gamma_1) \supseteq \overline{V}^I_j(\gamma_2)$, for each query $q_j \in Q$.*

Proof. Since $\gamma_1 > \gamma_2 > 0$, it follows that for each view $v \in \{v : v \in \overline{V}, v \notin Q, r_{min}(v, Q) \leq \gamma_2\}$, we have $v \in \{v : v \in \overline{V}, v \notin Q, r_{min}(v, Q) \leq \gamma_1\}$. Thus, $\overline{V}^I(\gamma_1) \supseteq \overline{V}^I(\gamma_2)$. For each query $q_j \in Q$, if $v \in \overline{V}^I_j(\gamma_2)$, then $v \supseteq q$ and $\overline{V}^I(\gamma_2) \subseteq \overline{V}^I(\gamma_1)$. Thus, $v \in \overline{V}^I_j(\gamma_1)$. It follows that $\overline{V}^I_j(\gamma_1) \supseteq \overline{V}^I_j(\gamma_2)$.

Let $Objv(\cdot)$ denote the optimal value of the model $(\cdot)$. We obtain the following corollary.

Corollary 2. *Given $\gamma_1 \geq \gamma_2 > 0$, $Objv\big(IPRv(\gamma_1)\big) \leq Objv\big(IPRv(\gamma_2)\big)$.*

The proof of Corollary 2 is obtained directly from Observation 7.

For any finite threshold γ, we can no longer guarantee that the optimal solution of the associated model $IPRv(\gamma)$ is optimal for the original problem. But this model is potentially smaller, and thus it is easier to solve. Obviously, the effectiveness of this strategy depends on the value of the parameter γ. Larger values of γ result in potentially larger sets $\overline{V}^I(\gamma)$. Thus the corresponding IP model $IPRv(\gamma)$ would also be larger. Solving such a larger IP model is likely to result in larger execution time and memory requirements, but its optimal solution is potentially better (smaller). Indeed at $\gamma = +\infty$, we have $\overline{V}^I(\gamma) = \overline{V}$ and the resulting model $IPRv(\gamma)$ would be identical with model $IP2$ (hence its optimal solution is guaranteed to be optimal for the original problem). But smaller values of γ are more likely to result in smaller IP models, thus the corresponding execution time and memory requirements may be more manageable for larger instances of the problem DVS. Of course it is also true that smaller value of γ could result in removing potentially beneficial views from the search space, thus the optimal solution of the corresponding IP model might not be as effective. We carry out a computational experiment to study the impact of the parameter γ on both the solvability of the corresponding model $IPRv(\gamma)$ and the quality of its optimal solution, on an empirical basis. We report our observations in Section 7. Based on these observations, we also propose a strategy for selecting an appropriate value for γ in each instance.

6.3 Heuristic method II: two-threshold strategy

In principle, this heuristic method is similar to method I discussed in Section 6.2, where we limit the search space of views to a promising subset of $\overline{V}$. But we now further reduce the size of the IP model by limiting the choice of view-query relationships as well.

To do so, given the reduced search space of views $\overline{V}^{I}(\gamma)$ obtained by method I, for each view v in $\overline{V}^{I}(\gamma)$, we remove every view-query relationship in which the corresponding cost-benefit ratio is above a given threshold θ. More specifically, given a second threshold θ, for each view $v \in \overline{V}^{I}(\gamma)$, let $N_{v,\theta}$ be the largest integer such that the cost-benefit ratio of view v over the set of $N_{v,\theta}$ largest input queries is less than or equal to θ, (that is, $r(v, Q_{(N_{v,\theta})}) \leq \theta$). Since the cost-benefit ratio $r(v, Q_{(n)}(v))$ is a unimodal function with respect to n, it follows that such a value of $N_{v,\theta}$ is well-defined, and for all $n > N_{v,\theta}$ we have $r(v, Q_{(n)}(v)) > \theta$. For each view v, we now keep only the view-query relationships between view v and the $N_{v,\theta}$ largest input queries that v can answer. All other view-query relationships for view v are discarded. We always choose the values of $\theta \geq \gamma > 0$, so as to guarantee that for each view v in $\overline{V}^{I}(\gamma)$, the collection of queries that result in the minimum ratio for that view remain in the view-query relationships that we keep. In addition, similar to method I, in order to guarantee feasibility, we always keep all choices of the view-query relationship associated with the view v_{max}. We build on this idea to develop the heuristic method II, and formulate an integer programming model that we denote by $IPRvq(\gamma, \theta)$ for the problem DVS.

Hence, given the reduced search space of views $\overline{V}^{I}(\gamma)$ and the second threshold θ, for each query q, we only keep in the search space of views associated with q (that is, the candidate views for answering q) the view v_{max} and each view $v \in \overline{V}^{I}(\gamma)$ if query q is one of the largest $N_{v,\theta}$ queries that v can answer. For each view $v \in \overline{V}^{I}(\gamma)$ we define $Q_\theta(v)$ as the set of the $N_{v,\theta}$ largest input queries that view v can answer, or equivalently,

$$Q_\theta(v) = \{q \in Q(v) : q \text{ is the } n^{th} \text{ largest query in } Q(v), \text{ for } 1 \leq n \leq N_{v,\theta}\}$$

We can now define the reduced search space of views associated with query q_j, denoted by $\overline{V}_j^{II}(\gamma, \theta)$, as follows.

$$\overline{V}_j^{II}(\gamma, \theta) = \{v \in \overline{V}_j^{I}(\gamma) : q_j \in Q_\theta(v)\} \cup \{\, v_{max}\}.$$

We illustrate this method by the following example.

Example 1 (Continued). *Given $\gamma = 1.2$, we obtain that the views $v_1 = \{a, b, c\}$ and $v_2 = \{b, c, d\}$ are included in the reduced view set $\overline{V}^{I}(\gamma)$. If we set the threshold $\theta = 1.3$, then we obtain that $N_{v_1,\theta} = 4$ and $N_{v_2,\theta} = 3$. In other words, for view v_1 we only keep the relationship between v_1 and the largest 4 queries in $Q(v_1)$. For view v_2 we only keep the relationship between v_2 and the largest 3 queries in $Q(v_2)$. The choices of the relationships are shown in Figure 3. We also make reduction on the choices of view-query relationship associated with all the other views in $\overline{V}^{I}(\gamma)$, and we obtain the corresponding reduced view set for each query as listed below. Recall that $q_1 = \{b\}$, $q_2 = \{c\}$, $q_3 = \{a, b\}$, $q_4 = \{a, c\}$, $q_5 = \{b, c\}$, $q_6 = \{b, d\}$ and $q_7 = \{c, d\}$.*

$$\begin{aligned}
\overline{V}_1^{II}(\gamma,\theta) &= \big\{\{b\},\{a,b\},\{b,c\},\{b,d\},\{a,b,c,d\}\big\}\\
\overline{V}_2^{II}(\gamma,\theta) &= \big\{\{c\},\{b,c\},\{c,d\},\{a,b,c\},\{a,b,c,d\}\big\}\\
\overline{V}_3^{II}(\gamma,\theta) &= \big\{\{a,b\},\{a,b,c\},\{a,b,c,d\}\big\}\\
\overline{V}_4^{II}(\gamma,\theta) &= \big\{\{a,c\},\{a,b,c\},\{a,b,c,d\}\big\}\\
\overline{V}_5^{II}(\gamma,\theta) &= \big\{\{b,c\},\{a,b,c\},\{b,c,d\},\{a,b,c,d\}\big\}\\
\overline{V}_6^{II}(\gamma,\theta) &= \big\{\{b,d\},\{b,c,d\},\{a,b,c,d\}\big\}\\
\overline{V}_7^{II}(\gamma,\theta) &= \big\{\{c,d\},\{b,c,d\},\{a,b,c,d\}\big\}.
\end{aligned}$$

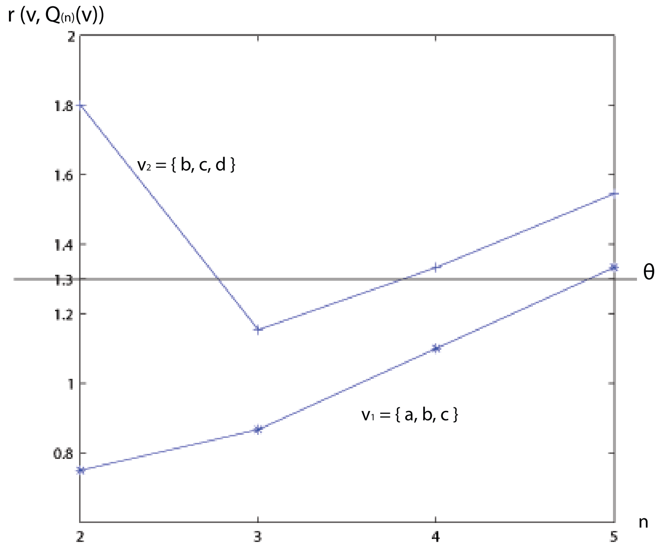

Fig. 3. The choices of view-query relationships in Example 1

Given the thresholds γ and θ, we can now define the corresponding integer programming model that we refer to as $IPRvq(\gamma,\theta)$ by using the set $\overline{V}_j^{II}(\gamma,\theta)$ for each $q_j \in Q$, in place of $\overline{V}_j^{I}(\gamma)$ in model $IPRv(\gamma)$. The following observation and corollary show that the size of the model $IPRvq(\gamma,\theta)$ is smaller than the model $IPRv(\gamma)$. As we increase the value of θ with fixed γ, the objective value of the corresponding model $IPRvq(\gamma,\theta)$ will either decrease or remain the same. As we increase the value of γ with fixed θ, the objective value of the corresponding model $IPRvq(\gamma,\theta)$ will either decrease or remain the same.

Lemma 2. *If $\theta_1 > \theta_2 > 0$, then for each view v we have $N_{v,\theta_1} \geq N_{v,\theta_2}$ and $Q_{\theta_1}(v) \supseteq Q_{\theta_2}(v)$.*

Proof. For $N_{v,\theta_1} < n \leq |Q(v)|$, we have $r(v, Q_{(n)}(v)) > \theta_1 > \theta_2$. Thus, $N_{v,\theta_2} \leq N_{v,\theta_1}$. It follows that $Q_{\theta_1}(v) \supseteq Q_{\theta_2}(v)$.

Observation 8

i) Given $\theta \geq \gamma > 0$, for each query $q_j \in Q$, $\overline{V}^I_j(\gamma) \supseteq \overline{V}^{II}_j(\gamma, \theta)$. If $\theta = +\infty$, $\overline{V}^I_j(\gamma) = \overline{V}^{II}_j(\gamma, \theta)$.

ii) Given $\theta_1 > \theta_2 \geq \gamma > 0$, for each query $q_j \in Q$ we have $\overline{V}^{II}_j(\gamma, \theta_1) \supseteq \overline{V}^{II}_j(\gamma, \theta_2)$.

iii) Given $\theta \geq \gamma_1 > \gamma_2 > 0$, for each query $q_j \in Q$ we have $\overline{V}^{II}_j(\gamma_1, \theta) \supseteq \overline{V}^{II}_j(\gamma_2, \theta)$.

Proof. i) This result can be obtained directly from the definition of $\overline{V}^{II}_j(\gamma, \theta)$.

ii) If $v \in \overline{V}^{II}_j(\gamma, \theta_2)$ and $v \neq v_{max}$, then by definition we obtain that $v \in \overline{V}^I(\gamma)$ and $q \in Q_{\theta_2}(v)$. By Lemma 2, we have $q \in Q_{\theta_1}(v)$. Thus, $v \in \overline{V}^{II}_j(\gamma, \theta_1)$. It follows that $\overline{V}^{II}_j(\gamma, \theta_1) \supseteq \overline{V}^{II}_j(\gamma, \theta_2)$.

iii) If $v \in \overline{V}^{II}_j(\gamma_2, \theta)$ and $v \neq v_{max}$, then by definition we obtain that $v \in \overline{V}^I(\gamma_2)$ and $q \in Q_\theta(v)$. Since $\gamma_1 > \gamma_2 > 0$, by Observation 7, $v \in \overline{V}^I(\gamma_2) \subseteq \overline{V}^I(\gamma_1)$. Thus, $v \in \overline{V}^{II}_j(\gamma_1, \theta)$. It follows that $\overline{V}^{II}_j(\gamma_1, \theta) \supseteq \overline{V}^{II}_j(\gamma_2, \theta)$.

Corollary 3.

i) Given $\theta \geq \gamma > 0$, $Objv\big(IPRv(\gamma)\big) \leq Objv\big(IPRvq(\gamma, \theta)\big)$. If $\theta = +\infty$, the models $IPRv(\gamma)$ and $IPRvq(\gamma, \theta)$ are equivalent.
ii) Given $\theta_1 > \theta_2 \geq \gamma > 0$, $Objv\big(IPRvq(\gamma, \theta_1)\big) \leq Objv\big(IPRvq(\gamma, \theta_2)\big)$.
iii) Given $\theta \geq \gamma_1 > \gamma_2 > 0$, $Objv\big(IPRvq(\gamma_1, \theta)\big) \leq Objv\big(IPRvq(\gamma_2, \theta)\big)$.

Proof. i)-iii) follows directly from Observation 8.

7 Experimental results

In this section, we present the results of a computational experiment with the heuristic methods that we proposed above. We evaluate the performance of these methods by discussing the quality of solution and the efficiency of the reduction in the size of the corresponding IP models. The results also show that our heuristic methods outperform other heuristic methods in terms of both the execution time and the quality of solutions obtained. More specifically, our experimental results show that:

– The size of the resulting models obtained by the heuristic methods I and II is significantly smaller than that of $IP2$, allowing us to obtain optimal or near optimal solution for relatively large instances of the problem within reasonable time and memory limits.

- The magnitude of the size reduction and the quality of resulting solution depends on the values of the parameters (γ, θ), as well as on the specific characteristics of each instance.
- On average, the heuristic methods I and II require less execution time than the heuristic methods proposed in [3, 4], and the resulting solutions are significantly better (i.e., have lower costs).

7.1 Experiments with heuristic method I

In this subsection we evaluate the effectiveness of the single-threshold strategy in terms of its computational requirements as well as the quality of solutions it obtained. We also study the impact of the threshold γ on the quality of the solution obtained by model $IPRv(\gamma)$.

To examine the effect of the parameter γ on the performance of model $IPRv(\gamma)$, for several instances of the problem DVS we solve the model $IPRv(\gamma)$ with different values of γ. Here we report our observations for one instance of the problem. The pattern of observations for other instances is similar to this instance. The instance that we report in Table 4 and Figure 4 contains 125 queries constructed over the 17-attribute TPC-H dataset. The number of attributes of each query is a random number between 5 and 12, (that is, the query is randomly generated from the middle levels of the view lattice). We continue to set the storage limit equal to one-fifth of the sum of the sizes of the queries. In Table 4, for each value of γ we present the number of views in the search space of model $IPRv(\gamma)$, as well as the number of variables and the number of constraints in each model. Note that when $\gamma = +\infty$, the resulting model $IPRv(\infty)$ is identical with the model $IP2$. We evaluate the quality of the solutions with respect to difference values of γ by calculating the *gap* as the ratio of difference between the optimal values of the models $IPRv(\gamma)$ and $IP2$ over the optimal value of $IP2$ (expressed as a percentage). In Table 4 and Figure 4 we present these results, as well as the total execution time of each model $IPRv(\gamma)$.

From Table 4 we observe that the number of views in model $IPRv(\gamma)$ increases as we increase the value of γ from 0.2 to 1. The size of the corresponding model, expressed by the numbers of variables and constraints, is thus monotonically increasing as expected. But even for $\gamma = 1$, where the gap reduces to 0.0%, the size of the resulting model $IPRv(\gamma)$ is still significantly smaller than the model $IP2$, or equivalently, the model $IPRv(\infty)$.

From Figure 4 we observe that the value of the cost of answering the queries in this instance obtained via model $IPRv(\gamma)$ (as measured by the value of *gap*) is a non-increasing function of γ as expected. It is also observed that the total execution time generally increases as we increase the threshold γ. The total execution time of solving this instance via model $IP2$ is around 1.5 hours, which is far beyond the assumed time limit 30 min. However, for all $\gamma \leq 1$ the size of the model $IPRv(\gamma)$ is dramatically reduced, resulting in a significant reduction of the total execution time of the model. Notably at the largest value of γ in this experiment ($\gamma = 1$) the execution time remains below 20 minutes. In addition, when the threshold $\gamma = 1.0$, the gap is equal to 0, indicating that the optimal

Table 4. The number of views and the size of the model $IPRv(\gamma)$ with different values of γ

Threshold (γ)	number of views in $IPRv(\gamma)$	number of variables in $IPRv(\gamma)$	number of constraints in $IPRv(\gamma)$	total execution time (sec.)	gap (%)
0.2	127	496	495	0.3	5.75%
0.3	146	690	670	0.4	5.08%
0.4	241	2,534	2,419	16.2	1.64%
0.5	450	6,908	6,584	392.6	1.30%
0.6	742	11,550	10,934	629.8	0.84%
0.7	1,139	16,944	15,931	760.8	0.46%
0.8	1,507	21,129	19,748	883.3	0.29%
0.9	2,108	26,709	24,727	1184.4	0.00%
1.0	2,475	30,301	27,952	979.1	0.00%
∞	9,524	76,328	66,930	5427.4	0.00%

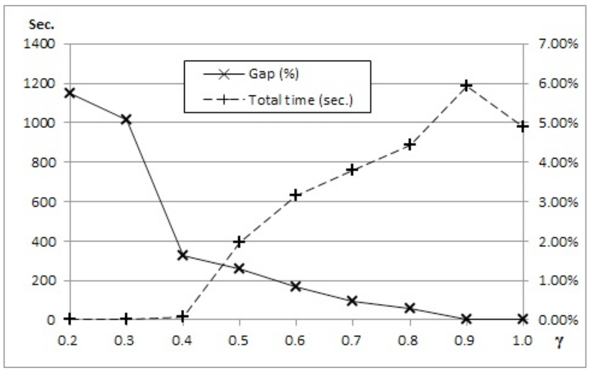

Fig. 4. The results of solving an instance using model $IPRv(\gamma)$ with different values of γ

solution obtained by the respective model $IPRv(\gamma)$ is also an optimal solution for the original problem. Hence, the heuristic method I can be a relatively effective technique in solving the problem DVS regarding both the quality of solution and efficiency of reduction.

To further evaluate the efficiency of the model, we solve 12 instances with 4 different query types (presented in Section 5) over the 17-attribute TPC-H dataset using the model $IPRv(\gamma)$ for $\gamma = 1$. For each query type, we solve the 3 largest instances presented in Section 5. For each instance we increase the value of the parameter γ in the model $IPRv(\gamma)$ from 0.1 to 0.2 to 0.3, $\ldots$, until the cost obtained by model $IPRv(\gamma)$ is within 0.2% difference of the cost obtained by model $IP2$. We refer to $\gamma_{0.2}$ as the smallest value of γ which has such property. (For instances which could not be solved by model $IP2$ within time and memory limits, we refer to $\gamma_{0.2}$ as the smallest value of γ such that the cost obtained from model $IPRv(\gamma)$ is within 0.2% of lower bound of model $IP2$ obtained by CPLEX.) In Table 5, for each instance we compare the number of views in the search space, and the total execution time for the corresponding models $IP2$ and $IPRv(\gamma_{0.2})$.

Table 5. Comparison of the number of views in the search space, the total execution time, and the optimal value in models $IP2$ and $IPRv(\gamma_{0.2})$

ins-tance	query type	number of queries	$\gamma_{0.2}$	number of views		total time (sec.)	
				$IP2$	$IPRv(\gamma_{0.2})$	$IP2$	$IPRv(\gamma_{0.2})$
1	random	220	0.9	23,256	1,774	317.0	80.0
2		240	0.8	18,783	1,450	388.5	65.1
3		260	1.0	25,922	3,947	out of memory	669.9
4	bottom-level	60	1.3	8,048	384	387.4	7.1
5		80	1.2	14,248	619	>30min	107.0
6		100	1.2	17,937	2,092	>30min	1538.3
7	middle-level	60	1.1	3,614	940	21.7	9.5
8		80	0.7	4,305	592	93.8	8.7
9		100	0.9	7,278	1,706	>30min	171.9
10	top-level	320	0.4	2,673	1,123	736.5	334.4
11		340	0.4	2,703	1,237	815.4	367.3
12		360	0.4	2,930	1,204	>30min	1662.5

From the results of Table 5 we observe that the value of $\gamma_{0.2}$ is relatively small, and ranges from 0.4 to 1.3. This confirms our statement that the view with a relatively small value of cost-benefit ratio is more likely to be selected for answering the queries. For every instance the number of views in the search space of model $IPRv(\gamma_{0.2})$ is significantly smaller than model $IP2$. This results in a significant reduction of the total execution time of model $IPRv(\gamma_{0.2})$ compared with the model $IP2$. In addition, for the instances which cannot be solved by model $IP2$ due to memory shortage or time limitation (the instances 3, 5, 6, 9, and 12), we can solve them using model $IPRv(\gamma)$ with acceptable qualities (i.e., within 0.2% of the corresponding lower bound) within the time and memory limits. These results show that the model $IPRv(\gamma)$ allows us to solve larger instances of the problem DVS, and obtain an optimal or near optimal solution more efficiently.

We now study the effect of the space limit and the values of the parameter γ on the performance of model $IPRv(\gamma)$. To this end, we solve the instance in Table 4 under different space limit constraints using model $IPRv(\gamma)$ with different values of γ. More specifically, we assume the space limit is a fraction β of the sum of the sizes of the input queries. We solve the instance for $\beta = 0.1$, 0.2, …, 0.6. In Table 6, for each space limit we present the smallest value of γ such that the cost obtained by the resulting model $IPRv(\gamma)$ is optimal for the corresponding original problem.

Table 6. The impact of space limit parameter β on the choosing of γ

β	0.1	0.2	0.3	0.4	0.5	0.6
min. γ with optimal cost	1.5	1.3	0.7	0.6	0.4	0.3

As observed in Table 6, when we increase the space limit (or β), the minimum value of γ with optimal cost decreases. Generally speaking, when the space limit is relatively large, the optimal solution is more likely to comprise of only the views with small value of minimum cost-benefit ratio.

In general, for solving the problem DVS using model $IPRv(\gamma)$, it seems reasonable to increase the value of γ as much as possible while the corresponding execution time is acceptable.

7.2 Experiments with the heuristic method II

In this subsection, we evaluate the efficiency of heuristic method II in solving the problem DVS that we described in Section 6.3. We also examine the impact of the two parameters (γ, θ) in the model $IPRvq(\gamma, \theta)$ on the quality of solutions.

To evaluate the efficiency of the heuristic method II, we solve the 12 instances of Table 5 using the model $IPRvq(\gamma, \theta)$. More specifically, for each instance we set $\gamma = \theta = \gamma_{0.2}$, where the value of $\gamma_{0.2}$ is presented in Table 5. Subsequently, we increase the value of the parameter θ from $\gamma_{0.2}$ to $(\gamma_{0.2} + 0.1)$ to $(\gamma_{0.2} + 0.2)$, …, until the cost obtained by model $IPRv(\gamma_{0.2}, \theta)$ is within 0.2% difference of the cost obtained by model $IP2$. We refer to $\theta_{0.2}$ as the smallest value of θ that has such property. We compare the number of variables, the number of constraints, and the total execution time of the models $IPRv(\gamma_{0.2})$ and $IPRvq(\gamma_{0.2}, \theta_{0.2})$. The results are shown in Table 7.

From the results of Table 7 we observe that the value of $\theta_{0.2}$ is relatively close to the value of $\gamma_{0.2}$. For every instance in Table 7 the number of variables and the number of constraints in model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ are smaller than those in model $IPRv(\gamma_{0.2})$. This reduction is much more significant for the instances of

Table 7. Comparison of the number of variables, the number of constraints, and the total execution time in models $IPv(\gamma_{0.2})$ and $IPRvq(\gamma_{0.2}, \theta_{0.2})$

instance	query type	number of queries	parameters $(\gamma_{0.2}, \theta_{0.2})$	number of variables		number of constraints		total time (sec.)	
				$IPRv$	$IPRvq$	$IPRv$	$IPRvq$	$IPRv$	$IPRvq$
1	random	220	(0.9,0.9)	61649	17322	60096	15769	80.0	53.0
2		240	(0.8,0.8)	50999	15119	49790	13910	65.1	46.4
3		260	(1.0,1.0)	143467	46237	139781	42551	669.9	316.2
4	bottom-level	60	(1.3,1.7)	7021	3566	6698	3243	7.1	5.6
5		80	(1.2,1.6)	10891	5448	10353	4910	107.0	61.4
6		100	(1.2,1.5)	62458	24282	60467	22291	1538.3	741.7
7	middle-level	60	(1.1,1.1)	9007	7719	8128	6840	9.5	9.3
8		80	(0.7,0.7)	6662	4646	6151	4135	8.7	7.5
9		100	(0.9,0.9)	19067	15647	17462	14042	171.9	146.3
10	top-level	320	(0.4,0.4)	12386	11237	11584	10435	334.4	298.9
11		340	(0.4,0.4)	13489	12505	12593	11609	367.3	362.8
12		360	(0.4,0.4)	13895	12721	13052	11878	1662.5	1657.4

"random" and "bottom-level" queries (instances 1-6). It follows that for these instances the total execution time to solve model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ is significantly shorter than the time to solve model $IPRv(\gamma_{0.2})$.

To further examine the impact of the two parameters (γ, θ) on the effectiveness of the heuristic method II, we solve the model $IPRvq(\gamma, \theta)$ with different values of γ and θ for a number of instances. The instance that we report in Figure 5 is constructed over the 17-attribute TPC-H dataset. It contains 200 queries which are randomly chosen from the middle levels of the view lattice. The number of attributes in each query ranges from 5 to 12. We continue to set the storage limit as one-fifth of the sum of the sizes of the queries. We solve this instance using model $IPRvq(\gamma, \theta)$ for $\gamma = 0.3, 0.4, \ldots, 0.6$, and $\theta = \gamma, \gamma + 0.1, \ldots, 0.6$. To compare the quality of solutions, we also solve the instance using the model $IPRv(\gamma)$, or equivalently, the model $IPRvq(\gamma, \infty)$. The optimal value of each model is presented in Figure 5. Each line represents the results with respect to the same value of γ. All the missing point in this Figure indicates that we cannot solve the corresponding model within the time and memory limits.

In Figure 5 we observe that for each fixed value of γ the optimal cost obtained by model $IPRvq(\gamma, \theta)$ monotonically decreases as we increase the value of θ. Similarly this optimal cost is also a non-increasing function of γ with fixed value of θ. Both of these observations are of course consistent with our expectations for obvious reasons. The lowest cost among the models for different values of the parameters in Figure 5 is achieved when $\gamma = \theta = 0.6$. When we attempted to solve this instance via heuristic method I, we were able to solve the model $IPRv(\gamma)$ only for $\gamma = 0.3$. For larger values of γ (i.e., $\gamma \geq 0.4$) we were not able to solve the corresponding IP model via CPLEX due to memory and time limitations. The cost obtained by model $IPRvq(0.6, 0.6)$ is significantly smaller than the cost obtained by the model $IPRv(0.3)$.

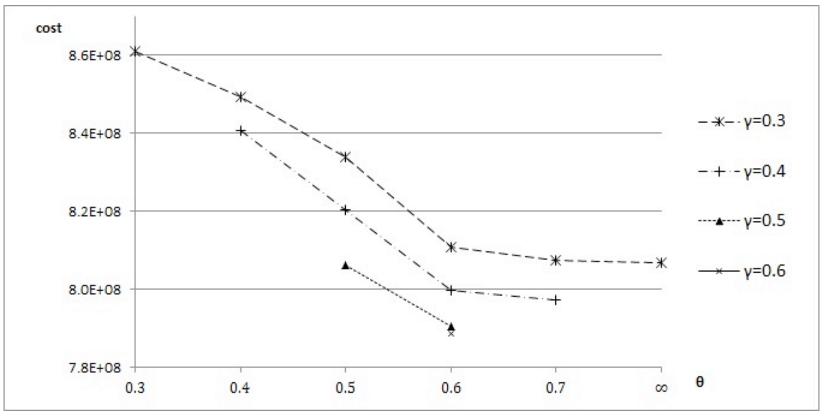

Fig. 5. The results of solving an instance using model $IPRvq(\gamma, \theta)$ with different values of γ and θ

When we compare the impact of the parameters γ and θ on the quality of the solution, the impact of parameter γ appears to be relatively more significant than the impact of parameter θ in obtaining a relatively low cost of query processing. This suggests that perhaps a good strategy for determining the value of γ and θ would be to increase the value of γ as much as possible while the resulting model is still solvable with regard to the memory and time requirements. Note that given the parameter $\gamma = \gamma_0$, the model $IPRvq(\gamma, \theta)$ of the smallest size over all the possible values of θ is the one with $\theta = \gamma = \gamma_0$. As a result, we first increase the value of γ as much as possible while the resulting model $IPRv(\gamma)$ is solvable. Subsequently, we continue to increase the value of γ as long as $IPRvq(\gamma, \gamma)$ is solvable. Then we fix the value of γ and increase θ until the resulting model $IPRvq(\gamma, \theta)$ is solvable within the memory and time limits. This strategy allows us to find an appropriate set of values for the parameters γ and θ that results in a lowest objective function value within the available computational resources.

7.3 Comparison with the inexact model $IPV(s)$

To further evaluate our heuristic methods, in this section we carry out a computational experiment to compare the effectiveness in terms of execution time and quality of solution of our heuristic methods with the inexact method proposed in [3,4] for solving the problem DVS. We first compare the execution time while requiring the same quality of solution over each method for a collection of instances for which we have the optimal value. Subsequently, we compare the quality of solution obtained, in the manner that obtain the lowest cost, by each method within time and memory limits over a second collection of instances for which the optimal value cannot be obtained via the corresponding model $IP2$.

The results of [3,4] show that their methods outperform other heuristic methods including [2,15,17]. The basic idea of the inexact method proposed in [3,4] is to limit the search space of views only to those views that are "relatively close" ancestors of the given queries. More specifically, Asgharzadeh et al. [3] define the set V_s as the reduced search space of views where each view in V_s is the union of exactly 1, or 2, ..., or s queries in the given query set Q. We apply this view reduction algorithm on the search space of views in the model $IP2$ that we introduced in Section 4.2. Given the parameter s, we refer to the resulting inexact model as model $IPV(s)$.

We first compare the effectiveness of our models $IPRv(\gamma)$ and $IPRvq(\gamma,\theta)$ with model $IPV(s)$ in terms of execution time over a collection of instances. This consists of all the instances in Tables 5 and 7 for which we have the optimal value via the corresponding model $IP2$ (i.e., instances 1, 2, 4, 7, 8, 10, and 11 in Tables 5 and 7). We solve this collection of instances using model $IPV(s)$. More specifically, for each instance we increase the value of s in the corresponding model $IPV(s)$ from 1 to 2 to 3, ..., until the cost obtained by the corresponding model $IPV(s)$ is within 0.2% larger than the optimal cost obtained by model $IP2$. (Here we use the same requirement of the quality of the solution (0.2%) as for our heuristic methods in Sections 7.1 and 7.2.) We denote this value of s by $s_{0.2}$. We compare the execution time of models $IPRvq(\gamma_{0.2},\theta_{0.2})$ and $IPV(s_{0.2})$ in Table 8.

Table 8. Comparison of the total execution time in models $IPRvq(\gamma_{0.2},\theta_{0.2})$ and $IPV(s_{0.2})$

ins-tance	query type	number of queries	parameters		total time (sec.)		$\frac{time(IPRv)}{time(IPV)}$ (%)
			$(\gamma_{0.2},\theta_{0.2})$	$s_{0.2}$	$IPRv(\gamma_{0.2},\theta_{0.2})$	$IPV(s_{0.2})$	
1	random	220	(0.9,0.9)	2	53.0	238.5	22.22%
2	random	240	(0.8,0.8)	2	46.4	103.2	44.96%
4	bottom-level	60	(1.3,1.7)	3	5.6	127.6	4.39%
7	middle-level	60	(1.1,1.1)	2	9.3	16.9	55.03%
8	middle-level	80	(0.7,0.7)	2	7.5	58.2	12.89%
10	top-level	320	(0.4,0.4)	2	298.9	584.8	51.11%
11	top-level	340	(0.4,0.4)	2	362.8	784.1	56.27%

As observed in Table 8, for all the instances the total execution time of model $IPRvq(\gamma_{0.2},\theta_{0.2})$ is significantly smaller than that of model $IPV(s_{0.2})$. On average the time to solve each instance using model $IPRvq(\gamma_{0.2},\theta_{0.2})$ is 35.27% of the time to solve that instance using model $IPV(s_{0.2})$. Our heuristic model $IPRvq(\gamma_{0.2},\theta_{0.2})$ is more efficient in obtaining an optimal or near optimal solution for these instances.

To further compare the quality of solutions obtained by heuristic methods proposed in this paper and in [3], we constructed 4 instances over the 17-attribute TPC-H dataset in a similar way as for constructing "bottom-level" queries. The number of attributes in each query ranges from 3 to 6. The number of queries in each instance ranges from 250 to 400. For each instance we set the storage limit equal to one-tenth of the sum of the sizes of the queries. The size of the corresponding model $IP2$ for each instance is relatively large, and thus the IP solver fails to obtain the optimal value via the corresponding model $IP2$ for all these instances. For each instance we compare the quality of the solutions of our methods with that in [3, 4] by comparing the lowest cost obtained by these methods. More specifically, for models $IPRv(\gamma)$ and $IPRvq(\gamma, \theta)$, we follow the strategy of choosing parameters (γ, θ) which we introduced at the end of Section 7.2. For each instance we denote by (γ_*, θ_*) the parameter values obtained in the manner that obtain the lowest cost within the time and memory limits. For model $IPV(s)$ we increase the value of s in the model $IPV(s)$ from 1 to 2 to 3, ..., until s is the largest value such that the corresponding model $IPV(s)$ can provide a solution within the time and memory limits. We denote this value of s by s_*. We compare the objective value of models $IPRvq(\gamma_*, \theta_*)$ and $IPV(s_*)$, and we report the ratio of the cost obtained by model $IPRvq(\gamma_*, \theta_*)$ over the cost obtained by model $IPV(s_*)$ in Table 9.

Table 9. Comparison of the value of cost obtained from models $IPRvq(\gamma_*, \theta_*)$ and $IPV(s_*)$

ins-tance	number of queries	parameters		Optimal value		$\frac{Objv(IPV(s_*))}{Objv(IPRvq(\gamma_*,\theta_*))}$
		(γ_*, θ_*)	s_*	$IPV(s_*)$	$IPRvq(\gamma_*, \theta_*)$	
1	250	$(0.7, \infty)$	1[1]	990,745,690	581,082,522	1.705
2	300	$(0.5, \infty)$	1	1,161,110,365	754,989,400	1.538
3	350	$(0.4, \infty)$	1	1,281,740,747	878,162,010	1.460
4	400	$(0.4, \infty)$	1	1,420,222,618	950,496,295	1.494

[1] We fail to solve model $IPV(s)$ when $s \geq 2$ due to memory and time limits.

From Table 9 we observe that for this collection of instances the cost of query processing obtained by using model $IPV(s)$ is significantly higher than the cost obtained by our model $IPRvq(\gamma, \theta)$. This indicates that our heuristic method outperform the inexact model $IPV(s)$ for the collection of instances in Table 9 with respect to the quality of solution obtained.

We compare the scalability of the two models. For the instances in Table 9, we fail to obtain a solution by using the model $IPV(s)$ for $s \geq 2$ due to memory shortage. In general, since the number of views in the search space of model $IPV(s)$ is of the order of $O(\sum_{i=1}^{s} \binom{|Q|}{i})$ (see [3, 4]), the number of queries in the instance has a relatively large impact on the size of the model. The size of model $IPV(s)$ grows significantly fast as we increase s for the instances with large numbers of queries. For large-size instances the model $IPV(s)$ is solvable only when $s = 1$, whose search space of views consists of only the queries and v_{max} (the view with the least number of attributes that can answer all the queries). This limits the quality of solutions obtained by model $IPV(s)$ for those instances. On the other hand, the applicable values of the parameters (γ, θ) in our heuristic methods are independent of the size of the instances, which ensures high scalability of our heuristic methods. This allows us to choose appropriate values of parameter of model $IPRvq(\gamma, \theta)$ to obtain acceptable qualities of solution for relatively large instances of the problem DVS.

8 Conclusion

In this paper we undertook a systematic study of the deterministic view selection problem. We studied properties of views and queries in terms of the problem, as well as the structure of the IP model for solving the problem. We proposed a procedure to efficiently reduce the search space of views, hence reducing the size of the resulting IP models, while maintaining the optimality of the solution for the original problem. Our experiments showed that this reduction in the IP model is significant for small to large realistic-size instances. We also present two heuristic methods to further reduce the size of the resulting IP models. We showed experimentally that this reduction is even more significant, which allow us to solve larger instances of the problem while obtaining high quality solutions. Through a computational experiment we also showed that our proposed heuristic methods compare favorable with the previous heuristic approaches in [3, 4] in terms of execution time, quality of solution obtained, and scalability. Presently, we are further generalizing and incorporating our approaches in a two-stage environment for efficiently solving the two-stage view-selection problem. We plan to extend our approaches for solving the view-selection problem under the view-maintenance-cost constraint, as well as under more general constraints, such as those of [6]. We are also working on combined view and index selection.

References

1. S. Agrawal, N. Bruno, S. Chaudhuri, and V. R. Narasayya. AutoAdmin: Self-tuning database systems technology. *IEEE Data Eng. Bull.*, 29(3):7–15, 2006.
2. S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *VLDB*, pages 496–505, 2000.
3. Z. T. Asgharzadeh. *Exact and inexact methods for solving the view and index selection problem for OLAP performance improvement.* Ph.d dissertation, NCSU, 2010.
4. Z. T. Asgharzadeh, R. Chirkova, and Y. Fathi. Exact and inexact methods for solving the problem of view selection for aggregate queries. *International Journal of Business Intelligence and Data Mining*, 4(3/4):391–415, 2009.
5. Z. T. Asgharzadeh, R. Chirkova, Y. Fathi, and M. Stallmann. Exact and inexact methods for selecting views and indexes for OLAP performance improvement. In *EDBT*, pages 311–322, 2008.
6. N. Bruno and S. Chaudhuri. Interactive physical design tuning. In *ICDE*, pages 1161–1164, 2010.
7. N. Bruno, S. Chaudhuri, and G. Weikum. Database tuning using online algorithms. In *Encyclopedia of Database Systems*, pages 741–744. Springer US, 2009.
8. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.
9. S. Chaudhuri, U. Dayal, and V. R. Narasayya. An overview of business intelligence technology. *Commun. ACM*, 54(8):88–98, 2011.

10. S. Chaudhuri, V. R. Narasayya, and G. Weikum. Database tuning using combinatorial search. In *Encyclopedia of Database Systems*, pages 738–741. Springer US, 2009.
11. G. B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–288, 1957.
12. H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index selection for OLAP. In *ICDE*, pages 208–219, 1997.
13. H. Gupta and I. S. Mumick. Selection of views to materialize in a data warehouse. *IEEE Trans. Knowl. Data Eng.*, 17(1):24–43, 2005.
14. A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.
15. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *SIGMOD Conference*, pages 205–216, 1996.
16. ILOG. CPLEX 11.0 software package, 2007. `http://www.ilog.com/products/cplex/`.
17. P. Kalnis, N. Mamoulis, and D. Papadias. View selection using randomized search. *Data Knowl. Eng.*, 42(1):89–111, 2002.
18. H. J. Karloff and M. Mihail. On the complexity of the view-selection problem. In *PODS*, pages 167–173, 1999.
19. R. Kimball and M. Ross. *The Data Warehouse Toolkit: (second edition).* Wiley Computer Publishing, 2002.
20. S. Lightstone. Physical database design for relational databases. In *Encyclopedia of Database Systems*, pages 2108–2114. Springer US, 2009.
21. S. Papadomanolakis and A. Ailamaki. An integer linear programming approach to database design. In *ICDE Workshops*, pages 442–449, 2007.
22. A. Shukla, P. Deshpande, and J. F. Naughton. Materialized view selection for multidimensional datasets. In *VLDB*, pages 488–499, 1998.
23. D. Theodoratos and T. K. Sellis. Incremental design of a data warehouse. *J. Intell. Inf. Syst.*, 15(1):7–27, 2000.
24. TPC-H Revision 2.1.0. TPC Benchmark H (Decision Support). `http://www.tpc.org/tpch/spec/tpch2.1.0.pdf`.
25. J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *VLDB*, pages 136–145, 1997.