A Two-stage Stochastic View Selection Problem in Database Management Systems

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. We present a study of the two-stage stochastic view selection problem in database management systems. The objective is to minimize processing times of the given queries subject to a storage limit. We assume that the queries are given in two or more workloads. We propose a two-stage stochastic programming (SP) model for this problem and study the structure and properties of its extensive form, which is an integer programming (IP) model. We use these properties to remove variables and constraints from this IP model, and obtain a smaller model with the same optimal solution. This allows us to solve realistic-size instances of the problem using commercial IP solvers. Subsequently, we present appropriate models, and techniques to assess the value of using a two-stage SP model, and discuss the value of perfect information. We discuss the properties of these values by conducting a computational experiment.

Keywords: Database management systems, View selection, Stochastic programming, Integer programming, Value of two-stage versus one-stage, value of stochastic solution, value of perfect information

1 Introduction

Many data-intensive systems, such as commercial or scientific database systems, store vast collections of data, whose scale tends to grow massively over time. Such systems can improve some metric, such as performance, of answering user queries on stored data by using *derived data*. Derived data are computed and stored in the system in advance (that is, are *materialized)* using the stored *base data* and include cached replicas, indexes, and materialized views, which are used extensively in information integration and data warehouses. Derived data, such as materialized views in data warehouses and in information-integration systems, may facilitate greatly the repurposing, transformation, and integration of multiple, uncoordinated, and sometimes variously restricted data sources over which data users may have no control. As such, selecting appropriate derived data for use in data-intensive systems may contribute to solutions to the problems of creation, facilitation, visualization, and understanding of the diverse digital content in a variety of circumstances. For an introduction to the subject, please see [12], and for more extended discussions please see references therein.

As an illustration, consider the following scenario. Large retailer companies (such as Sears, Kmart, Target, and WalMart in the USA) all maintain significant-size databases storing information about ongoing sales transactions. For instance, WalMart is known for maintaining database relations whose size is in billions of rows, or *tuples.* All these companies generally have significant volumes of marketing analysis going on throughout the year. That is, for accounting, reporting, and business-intelligence purposes, each company's 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. In typical data-analysis, or *OLAP* (see [10] and [5]), queries, users are interested in obtaining summary information of a measure as a function of some business aspects, called *dimensions.* For instance, in a daily volume-of-sales query in analyses run for Sears, the dimensions could be *item sold*, *date and time of sale*, and *customer.*

Suppose the Sears sales data are stored in *relations* `Sales, Time,` and `Customer.` Suppose also that these relations form a *star schema*, with `Sales` as the "fact table" and `Time` and `Customer` as "dimension tables" (see [10]. Then the volume-of-sales query for November, 2010 could be expressed in the relational language SQL as the following grouping and aggregation over the star-schema join:

```
Q: SELECT itemCategory, customerType, sum(amount)
   FROM Sales NATURAL JOIN Time NATURAL JOIN Customer
   WHERE dateMonth = 11 AND dateYear = 2010
   GROUP BY itemCategory, customerType;
```

A data-management system evaluating the query `Q` on just the base relations would need to access all the sales information, including the likely large amount of irrelevant data for the sales *before* the reporting period. On the other hand, using derived data could reduce, perhaps significantly, the time to evaluate the query `Q`. For instance, using an *index* `I` that provides access to all sales events for all sales IDs for a specific month and year would obviate the need to access the irrelevant data for sales in months before (or after) November 2010. Note that evaluating the query `Q` using the index `I` would still involve examining sales data at the granularity of individual sales events. Alternatively, the data-management system could reduce the evaluation time of the query `Q` by using a *materialized view* `V`, which stores total sales per item category per customer type per month per year and is expressed in the relational language SQL as follows.

```
V: SELECT itemCategory, customerType, dateMonth, dateYear, sum(amount)
   FROM Sales NATURAL JOIN Time NATURAL JOIN Customer
   GROUP BY itemCategory, customerType, dateMonth, dateYear;
```

Observe that the query `Q` can be evaluated using a single relation `V` – that is, no *joins* of relations are required in the evaluation, and therefore the evaluation

time of `Q` when using `V` could be dramatically shorter than when not using `V`. Note also that the representation of the base data in the materialized view `V` no longer distinguishes between individual items or sales dates.

Naturally in such a setting the problem of selecting an appropriate collection of derived data has to be addressed in the context of the objectives and limitations of that setting. In recent years several researchers have addressed the subject and developed exact and inexact methods for solving the problem in a one-stage deterministic environment, that is, where all queries are assumed to be known and given in advance, and a one-stage probabilistic environment, that is, where we have a probability of occurrence associated with each query in a given collection of queries. See [1], [2], [3], [6], [7], among others. [YF: give additional references and update current refs]. In this article we study this problem in a two-stage environment and propose appropriate models and algorithms for solving the problem.

To illustrate this environment, let us refer to the Sears example described above. While some data-analysis queries would be posed on Sears' sales data throughout the year, some of the queries may be run only at certain times of the year. The focus of such seasonally relevant queries could include, for instance: (i) demand for fancy gadgets (such as large-screen TVs) during heavy sales events (such as Black Friday in the USA); (ii) sales of deeply discounted basic household appliances during Christmas sale events; and (iii) August sale events for summertime gear. In such a situation, aside from designing and using derived data to improve the processing performance of the routine queries occurring throughout the year, it would be beneficial to design and use additional derived data in order to reduce the execution time of such seasonal queries as well. Of course it is natural that sometimes the collection of derived data that is useful for the queries in one season may be different from those that are useful for the queries in the subsequent seasons. On the other hand, it is always desirable if (at least some of) the derived data that is developed for one season can also be used for the queries in the subsequent seasons, since it would save both time and resources. After all, creating derived data itself requires both human and computational resources, and any savings in this regard translates to more efficiency in the overall operation. Hence, the problem in this context would be to determine the collection of derived data for two or more seasons (stages) so as to answer all the queries in an efficient manner while keeping the total volume of derived data as low as possible. In other words, in this context we need to propose two or more collections of interrelated derived data so as to minimize the overall query response time subject to a storage limit on the amount of derived data produced.

A further complicating factor in this context is the fact that sometimes it is difficult to predict the specific set of queries that would become relevant and prominent at a given future point in time. For instance, political, economic, or environmental factors could all have an impact on the nature of the queries that become relevant in the future. This could lead the experts to forecast two or more possible sets of queries for that point in the future, with a probability

associated with each set. This, in turn, leads to a situation where we need to carry out the above mentioned analysis in a probabilistic environment, where we account for various scenarios (that is, query sets) that may occur in the future.

For instance, in a situation where we deal with only two seasons, we may need to propose a collection of derived data to answer a given set of queries immediately (first season, or first-stage, queries), and a plan for several collections of derived data, each collection associated with a possible set of future queries (second season, or second-stage, queries). In such a situation, in order to facilitate and expedite the process of creating the second-stage collections of derived data, it is clear that our decision for the first collection of derived data (that is, the first-stage decision) should be made with a view and consideration for various possibilities (queries) that may occur in the second stage. The idea here is to re-use in the second stage as many of the first-stage derived data as possible. This leads us to define and address a two-stage stochastic view selection problem as described in this paper.

Our specific contributions are as follows:

1. We define the two-stage stochastic view selection problem and model this problem as a two-stage stochastic programming (SP) model.
2. We study the structure and properties of the extensive form of the SP model, which is an integer programming (IP) model. We develop an algorithm that effectively reduces the search spaces of potentially beneficial views in the IP model, and obtain a smaller IP model whose solution is guaranteed to be optimal of the original SP model.
3. We conduct a computational experiment on the above IP models and discuss the scalability of the reduced IP model. The reduced search spaces significantly reduce the size of our original IP model, so that for realistic-size instances of the problem this IP model can be solved efficiently by a commercial IP solver such as CPLEX [9].
4. We compare our two-stage SP model with several appropriate models and present techniques to assess the value of the SP model and the value of perfect information in this context.
5. We discuss the impact of the replacement mechanism on the optimal solutions in our SP model by conducting a sensitivity analysis.

The remaining sections of this paper are organized as follow. In Section 2, we define the two-stage stochastic view selection problem and propose a stochastic integer programming model for it. In Section 3, we discuss the structure of the problem and the corresponding stochastic programming model, and use this structure to reduce the size of the problem considerably. In Section 4, we provide the results of a computational experiment with this model and make a few observations. In Section 5, we present appropriate models and techniques to determine the value of using a two-stage stochastic programming model and present several related numeric results. In Section 6, we discuss the value of perfect information in this context and conduct a related computational experiment. In Sections 7, we present a sensitivity analysis to evaluate the impact of the replacement mechanism of our model. And Section 8 contains a few concluding remarks.

2 The two-stage stochastic view selection problem

In this section, we first define the scope of the two-stage stochastic view selection problem that we consider, i.e., the type of the database, queries and views. Subsequently, we propose a stochastic programming (SP) model [4] for this problem. By studying the properties of the SP model, we then rewrite it as a model with fewer variables and constraints, and the resulting model is equivalent to the original SP model.

2.1 Formulation of the problem SVS

We consider a star-schema data warehouse (see [10] and [5]) with a single fact table and several dimension tables. In this context, we consider the evaluation cost of answering queries using unindexed materialized views such that each query can be evaluated using just one view and no other data. 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 $q \subseteq v$. In order 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. We use a_i to denote the size of each view v_i. 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.

In this environment we have a query workload Q_1 that we must answer at the present time (stage 1), and a second query workload $\mathbf{Q}_2$ that occurs at a future point in time (stage 2). We assume that the query workload Q_1 is known and given, but that the second query workload $\mathbf{Q}_2$ is a random set with a given probability distribution function. (We use boldface to emphasize that $\mathbf{Q}_2$ is a random set and differentiate it from a deterministic set such as Q_1.) We illustrate all of these notions in Example 1 later in this subsection.

At stage 1, we materialize a collection of views S_1 and use these views to answer the queries in Q_1. We assume that we have a storage limit b_1 for these views (that is, the total size of the views selected must not exceed b_1). At stage 2, once the actual collection of queries Q_2 for this stage are known, we allow for a partial replacement of some of the view relations that we constructed at stage 1, in order to obtain the collection of views S_2 for answering the query set Q_2. (Note that Q_2 represents a realization of the random set $\mathbf{Q}_2$.) In other words, at the end of stage 1 we keep some of the view relations that we materialized at stage 1 to use again at stage 2, while we discard other view relations from stage 1 and replace them with new view relations. We assume that the total size of

the views that we replace must not exceed a given storage limit b_2. Naturally, $b_2 \leq b_1$.

In this context the technical problem that we need to address prior to stage 1 is to select all views in the set S_1 and a replacement plan associated with each possible realization Q_2 of $\mathbf{Q}_2$. The objective is to minimize the total evaluation cost for the queries at stage 1 plus the expected total evaluation cost for the queries at stage 2. Note that in order to obtain a global optimal solution for this problem which we call ***S***tochastic ***V***iew ***S***election *(SVS)* problem, all decisions regarding both stages (that is, the view set S_1 and the replacement plan for every possible realization Q_2 of $\mathbf{Q}_2$) must be made prior to stage 1.

To illustrate, we present the following numeric example for a data cube [7] with 4 attributes.

Example 1. Given a database with four attributes a, b, c and d, the view lattice defined in [7] is shown in Figure 1. In this 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. The space requirement for each view in the lattice is given next to its corresponding node. In this instance, we assume that at stage 1, we are given two queries $q_1 = \{a, b\}$ and $q_2 = \{b, c\}$. At stage 2, $q_3 = \{b\}$ and $q_4 = \{a, c\}$ would occur with probability 0.5, while $q_5 = \{c\}$ and $q_6 = \{b, d\}$ would occur with probability 0.5. Equivalently, $Q_1 = \big\{\{a, b\}, \{b, c\}\big\}$, $Q_2^1 = \big\{\{b\}, \{a, c\}\big\}$, and $Q_2^2 = \big\{\{c\}, \{b, d\}\big\}$. Assume the total space limit $b_1 = 30$ and the space limit $b_2 = 15$. Our objective is to minimize the cost of answering Q_1 plus half of the cost of answering Q_2^1 plus half of the cost of answering Q_2^2. We need to determine a collection of views S_1 to materialize at stage 1, with the total size less than or equal to 30, and for each scenario at stage 2, we need to determine a subset of S_1 to keep from stage 1 to stage 2, and another set of views to materialize at stage 2, with the total size less than or equal to 15.

2.2 A mathematical programming model

In this subsection, we describe a stochastic programming model for the above two-stage stochastic view selection (SVS) problem. Although the general form of the model that we propose is valid for any probability distribution function for $\mathbf{Q}_2$, in the following model and throughout the rest of this article we assume $\mathbf{Q}_2$ is a random query set with L possible values. More specifically, we assume $\mathbf{Q}_2$ equals to query set Q_2^ℓ with probability p^ℓ, for $\ell = 1$ to L, where $\sum_{\ell=1}^{L} p^\ell = 1$. And we refer to each collection of queries Q_2^ℓ as a *scenario*. We start by defining the decision variables. For the first stage we define the following decision variables for all views $v_i \in V$ (where V is the set of all views) and for all queries $q_j \in Q_1$.

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

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

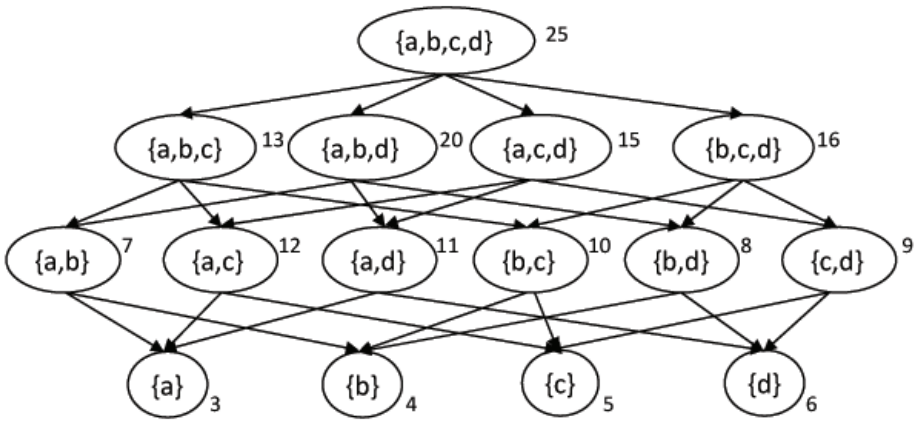

Fig. 1. Lattice example with view sizes

For the second stage we define the following decision variables for all views $v_i \in V$, and for all queries $q_j \in Q_2^\ell$, for $\ell = 1$ to L.

$$u_i^\ell = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 1 and used in stage 2 for query set } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

$$y_i^\ell = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 2 for query set } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

$$t_{ij}^\ell = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \text{ at stage 2 for query set } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

The cardinality of the view set V is 2^K, where K is the number of distinct attributes in the database. Let $I = \{1, 2, \ldots, 2^K\}$ be the set of subscripts for all the views $v_i \in V$. Also let J_1 be the set of subscripts for all the queries $q_j \in Q_1$, and let J_2^ℓ be the set of subscripts for all the queries $q_j \in Q_2^\ell$, for $\ell = 1, 2, \ldots, L$. The problem can now be written as the following stochastic programming model [4] that we denote by SP.

$$(SP) \qquad \text{minimize} \qquad \sum_{j \in J_1} \sum_{i \in I} d_{ij} z_{ij} + \mathbf{E}_{\mathbf{Q_2}} \Psi(\mathbf{x}, \mathbf{Q}_2) \tag{1}$$

$$\text{subject to} \qquad \sum_{i \in I} z_{ij} = 1 \qquad \forall j \in J_1 \tag{2}$$

$$z_{ij} \leq x_i \qquad \forall i \in I \text{ and } \forall j \in J_1 \tag{3}$$

$$\sum_{i \in I} a_i x_i \leq b_1 \tag{4}$$

All variables are binary

where $\Psi(\mathbf{x}, Q_2)$ is the minimum response time for a given set of values of the first stage variables $\mathbf{x} = (x_1, \cdots, x_{|V|})$ and for a realization of the second stage

queries Q_2, and $\mathbf{E}_{\mathbf{Q_2}}$ denotes mathematical expectation with respect to $\mathbf{Q}_2$. If the probability distribution of $\mathbf{Q}_2$ is as discussed above, we have that

$$\mathbf{E}_{\mathbf{Q}_2}\Psi(\mathbf{x}, \mathbf{Q_2}) = \sum_{\ell=1}^{L} p_\ell \Psi(\mathbf{x}, Q_2^\ell) \tag{5}$$

and for each value of ℓ, the corresponding value of $\Psi(\mathbf{x}, Q_2^\ell)$ is obtained by solving the following view selection problem.

$$\Psi(\mathbf{x}, Q_2^\ell) = \min \sum_{j \in J_2^\ell} \sum_{i \in I} d_{ij} t_{ij}^\ell \tag{6}$$

$$\text{subject to} \qquad \sum_{i \in I} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell \tag{7}$$

$$t_{ij}^\ell \le u_i^\ell + y_i^\ell \qquad \forall i \in I \ \text{ and } \ \forall j \in J_2^\ell \tag{8}$$

$$u_i^\ell \le x_i \qquad \forall i \in I \tag{9}$$

$$\sum_{i \in I} a_i y_i^\ell \le b_2 \tag{10}$$

$$\sum_{i \in I} a_i (u_i^\ell + y_i^\ell) \le b_1 \tag{11}$$

$$\textit{All variables are binary} \tag{12}$$

Constraints (2) and (7) state that each query is answered by exactly one view in the set of materialized views. Constraints (3) and (8) guarantee that a query can be answered by a view only if the view is already materialized. Constraints (4), (10), and (11) state the storage limitation of view sets. Constraint (9) guarantees that the view kept from stage 1 to stage 2 is already materialized at stage 1.

In defining various decision variables in the above model we have used the subscript i to represent view v_i, and in every case we have defined the decision variables for all values of i in the subscript set I (or equivalently for all views in the entire view set V). Naturally, in practice for each decision variable with a subscript i (associated with view v_i) we only need to consider those views v_i that are relevant in the context of that decision variable. For instance, in defining the decision variable x_i at stage 1 we only need to define this variable for those views v_i that can be used to answer at least one query either in the first stage or in the second stage. In other words, we need not consider view v_i (or define the corresponding variable x_i) if this view is not a superset of at least one query in our entire query set. In the remainder of this section we define various subsets of the overall view set V (and its corresponding subscript set I) so as to re-write the above model with few variables and constraints.

Let $\widehat{Q} = Q_1 \cup \left\{\cup_{\ell=1}^{L} Q_2^\ell\right\}$ be the collection of all queries (either in stage 1 or in stage 2). We define

$$V_1 = \left\{ v_i \in V : v_i \supseteq q \text{ for some } q \in \widehat{Q} \right\} \tag{13}$$

$$V_2^\ell = \left\{ v_i \in V : v_i \supseteq q \text{ for some } q \in Q_2^\ell \right\}, \quad \ell = 1, \ldots, L \tag{14}$$

Correspondingly, we define the sets of subscripts associated with V_1 and V_2^ℓ as I_1 and I_2^ℓ, for $\ell = 1$ to L, respectively. It follows that V_1 is the set of views that are relevant for stage 1, and V_2^ℓ is the set of views that are relevant for the ℓ^{th} sub-problem (scenario) in stage 2, for $\ell = 1$ to L.

Similarly, associated with each query q_j in the sets Q_1 and/or Q_2^ℓ we define a sub-collection of views that are relevant in answering that particular query. For each query $q_j \in Q_1$ we define $V_{1j} = \{v_i \in V_1 : v_i \supseteq q_j\}$, and for each query $q_j \in Q_2^\ell$ we define $V_{2j}^\ell = \{v_i \in V_2^\ell : v_i \supseteq q_j\}$, for $\ell = 1$ to L. Again, corresponding to the view sets V_{1j} and V_{2j}^ℓ we define the associated sets of subscripts as I_{1j} and I_{2j}^ℓ, for $\ell = 1$ to L, respectively.

Example 1 (Continued). In order to define V_1, V_2^1 and V_2^2, we only consider the views that could answer at least one query in the query set $\widehat{Q} = \{q_1, q_2, q_3, q_4, q_5, q_6\}$, Q_2^1 and Q_2^2, respectively. Thus, we obtain that

$$V_1 = \big\{\{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{b,d\}, \{c,d\}, \{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}, \{a,b,c,d\}\big\}$$
$$V_2^1 = \big\{\{b\}, \{a,b\}, \{a,c\}, \{b,c\}, \{b,d\}, \{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}, \{a,b,c,d\}\big\}$$
$$V_2^2 = \big\{\{c\}, \{a,c\}, \{b,c\}, \{b,d\}, \{c,d\}, \{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}, \{a,b,c,d\}\big\}.$$

For each query, we define the relevant view sets in answering that particular query. For example, for query $q_1 = \{a,b\} \in Q_1$ we define $V_{11} = \big\{\{a,b\}, \{a,b,c\}, \{a,b,d\}, \{a,b,c,d\}\big\}$, and for query $q_6 = \{b,d\} \in Q_2^2$ we define $V_{26}^2 = \big\{\{b,d\}, \{a,b,d\}, \{b,c,d\}, \{a,b,c,d\}\big\}$.

We can now define the first stage decision variables x_i, only for views in the set V_1, and the first stage decision variables z_{ij}, only for views v_i in the set V_{1j}. Similarly, we can define the second stage decision variables u_i^ℓ and y_i^ℓ, only for views $v_i \in V_2^\ell$, and the second stage decision variables t_{ij}^ℓ, only for views $v_i \in V_{2j}^\ell$, for $\ell = 1$ to L. It follows that we can now rewrite the above stochastic programming model SP as the following model that we denote by SP'.

$$(SP') \quad \text{minimize} \quad \sum_{j \in J_1} \sum_{i \in I_{1j}} d_{ij} z_{ij} + \mathbf{E}_{\mathbf{Q_2}} \Psi(\mathbf{x}, \mathbf{Q}_2) \tag{15}$$

$$\text{subject to} \quad \sum_{i \in I_{1j}} z_{ij} = 1 \qquad \forall j \in J_1 \tag{16}$$

$$z_{ij} \le x_i \qquad \forall j \in J_1 \text{ and } \forall i \in I_{1j} \tag{17}$$

$$\sum_{i \in I_1} a_i x_i \le b_1 \tag{18}$$

$$\textit{All variables are binary} \tag{19}$$

and the corresponding second stage subproblem can be written as

$$\Psi(\mathbf{x}, Q_2^\ell) = \min \sum_{j \in J_2^\ell} \sum_{i \in I_{2j}^\ell} d_{ij} t_{ij}^\ell \tag{20}$$

$$\text{subject to} \quad \sum_{i \in I_{2j}^\ell} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell \tag{21}$$

$$t_{ij}^\ell \leq u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell \text{ and } \forall i \in I_{2j}^\ell \tag{22}$$

$$u_i^\ell \leq x_i \qquad \forall i \in I_2^\ell \tag{23}$$

$$\sum_{i \in I_2^\ell} a_i y_i^\ell \leq b_2 \tag{24}$$

$$\sum_{i \in I_2^\ell} a_i (\, u_i^\ell + y_i^\ell) \leq b_1 \tag{25}$$

$$\textit{All variables are binary} \tag{26}$$

3 Solving the model *SP*

We begin this section by introducing an integer programming model [16] obtained from the extensive form [4] of the above stochastic programming model. Later in this section we study the properties of the integer programming model, use these properties to remove some variables and constraints, and obtain a model that is significantly smaller. The optimal solution of the resulting model is guaranteed to be optimal for the original model SP. This, in turn, allows us to solve larger instances of the SVS problem.

3.1 An integer programming model

In this subsection we propose an integer programming model [16] for the SVS problem. In order to solve the stochastic view selection model SP, we write the extensive form [4] of the stochastic programming model SP' that we discussed above, which is an integer programming model that we denote by $IP1$.

$$(IP1) \quad \text{minimize} \quad \sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j\in J_2^\ell}\sum_{i\in I_{2j}^\ell} d_{ij}t_{ij}^\ell \tag{27}$$

$$\text{subject to} \quad \sum_{i\in I_{1j}} z_{ij} = 1 \qquad \forall j \in J_1 \tag{28}$$

$$z_{ij} \le x_i \qquad \forall j \in J_1, \forall i \in I_{1j} \tag{29}$$

$$\sum_{i\in I_1} a_i x_i \le b_1 \tag{30}$$

$$\sum_{i\in I_{2j}^\ell} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell, \quad \ell = 1,\ldots,L \tag{31}$$

$$t_{ij}^\ell \le u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in I_{2j}^\ell, \quad \ell = 1,\ldots,L \tag{32}$$

$$u_i^\ell \le x_i \qquad \forall i \in I_2^\ell, \quad \ell = 1,\ldots,L \tag{33}$$

$$\sum_{i\in I_2^\ell} a_i y_i^\ell \le b_2, \quad \ell = 1,\ldots,L \tag{34}$$

$$\sum_{i\in I_2^\ell} a_i (u_i^\ell + y_i^\ell) \le b_1, \quad \ell = 1,\ldots,L \tag{35}$$

$$\textit{All variables are binary} \tag{36}$$

Note that for an instance with K attributes, n_1 queries in Q_1 at stage 1 and n_2^ℓ queries in Q_2^ℓ at stage 2, for $\ell = 1,\ldots,L$, there are at most $(n_1 + \sum_{\ell=1}^{L} n_2^\ell + 2L + 1)|V_1|$ variables and at most $(n_1 + \sum_{\ell=1}^{L} n_2^\ell + 2L)(|V_1|+1)+1$ constraints in the integer programming model $IP1$, where V_1 is as defined in Section 2.2. Thus, even for a relatively small number of K, L, n_1 and $n_2^1, \cdots, n_2^L$, the size of the model could be very large and the execution time of solving the IP model could be excessively long even with a relatively fast IP solver such as CPLEX 11 [9]. The applicability of our model is thus limited to only a small number of instances. We thus consider to reduce the size of our model by reducing the view sets V_1, V_{1j}, V_2^ℓ and V_{2j}^ℓ, for $\ell = 1,\ldots,L$. In section 4, we will provide numerical results for solving model $IP1$.

3.2 Reduction of search space

In this subsection, we make a few observations regarding the properties of the views that appear in an optimal solution for a given SVS problem. Based on these observations, we identify the reduced search spaces of views, which are several relatively small subsets of the view sets V_1, V_{1j}, V_2^ℓ and V_{2j}^ℓ. The reduced search spaces are guaranteed to contain at least one set of the optimal views. We then redefine the decision variables on the reduced search spaces in model $IP1$, and obtain an integer programming model that we denote by $IP2$. $IP2$ is significantly smaller than $IP1$, and the optimal solution of $IP2$ is guaranteed to be optimal for the model SP.

First reduction In a deterministic view selection problem, Asgharzadeh [1] proposes that one could reduce the search space of views (set of views to consider) by eliminating view v that has at least one attribute that is not in any of the queries answerable by this view. We extend the methods to our SVS problem and obtain the following observations.

Observation 1 *In an instance of SVS problem, given a view v_i in the view set V_1, if the number of attributes of v_i is strictly greater than the number of attributes in the union set of the queries in $\widehat{Q}$ that v_i could answer, that is, $|v_i| > |\cup_{\substack{q\in\widehat{Q}\\ q\subseteq v_i}} q|$, then there exists an optimal solution such that v_i is not materialized at stage one, or equivalently, $x_i = 0$.*

Proof. For each query $q \in \widehat{Q}$, if q could be answered by v_i, then $q \subseteq v_i$. Thus, $\cup_{\substack{q\in\widehat{Q}\\ q\subseteq v_i}} q \subseteq v_i$ and $|\cup_{\substack{q\in\widehat{Q}\\ q\subseteq v_i}} q| \leq |v_i|$. If $|\cup_{\substack{q\in\widehat{Q}\\ q\subseteq v_i}} q| < |v_i|$, then v_i has at least one attribute, call it attribute a, that is not in any $q \in \widehat{Q}$ such that $q \subseteq v_i$. If there exists an optimal solution such that v_i is materialized at stage one, we replace it with view $v \setminus \{a\}$. Since the size of $v_i \setminus \{a\}$ is no more than that of v_i, the replacement will not violate the space limitation. And for each query $q \in \widehat{Q}$ that is answered by v_i in the optimal solution, q could be answered by $v_i \setminus \{a\}$ with cost no more than that of v_i. Thus, we obtain an optimal solution such that v is not materialized at stage one and the corresponding decision variable $x_i = 0$.

Observation 2 *In an instance of SVS problem, given a view v_i in the view set V_2^ℓ, if the number of attributes of v_i is strictly greater than the number of attributes in the union set of the queries in Q_2^l that v_i could answer, that is, $|v_i| > |\cup_{\substack{q\in Q_2^\ell\\ q\subseteq v_i}} q|$, then there exists an optimal solution such that v_i is not materialized at stage two, or equivalently, $y_i^\ell = 0$.*

Proof. If v_i satisfies the above conditions, then v_i has at least one attribute, say attribute a, that is not in any $q \in Q_2^\ell$ and $q \subseteq v$. If v_i is materialized at stage two in an optimal solution, then replace v_i by $v_i \setminus \{a\}$ and the solution remains optimal.

Second reduction In the deterministic view selection problem, Asgharzadeh, Chirkova, Fathi [2] and Asgharzadeh [1] propose that one could reduce the search space of views based on the size of the views and on the total size of the queries that each view could answer. Let $S(\cdot)$ be the size of a query or a view. Then, we obtain the following observations.

Observation 3 *In an instance of SVS problem, given a view v_i in the view set V_1, if v_i satisfies the condition that the size of v_i is greater than the total size of the queries in $\widehat{Q}$ that this view can answer, that is, $S(v_i) > \sum_{q\in\widehat{Q}, q\subseteq v_i} S(q)$, then there exists an optimal solution in which v_i is not materialized at stage one, or equivalently, $x_i = 0$.*

Proof. Assume that $v_i \in V_1$ satisfies the above inequality condition and that v_i is materialized at stage one in an optimal solution. Note that a query could be answered by a view with the same set of attributes as the query, and the sizes of the query and the view are identical. Let $V(v, Q)$ be the set of views that have the same set of attributes as the queries that a view v could answer in a query set Q. At stage 1, if we materialize all the views in $V(v_i, \widehat{Q})$ instead of v_i, then the space limitation is not violated because of the above inequality condition. And every query in $\widehat{Q}$ that v_i answers could be answered by some view in $V(v_i, \widehat{Q})$ with cost no more than $S(v_i)$. Thus, the new solution remains optimal.

Observation 4 *In an instance of SVS problem, given a view v_i in the view set V_2^ℓ, if v_i satisfies the condition that the size of v_i is greater than the total size of the queries in Q_2^ℓ that this view can answer, that is, $S(v_i) > \sum_{\substack{q \in Q_2^\ell \\ q \subseteq v_i}} S(q)$, then there exists an optimal solution such that v_i is not materialized at stage two, or equivalently, $y_i^\ell = 0$.*

Proof. If $v_i \in Q_2^\ell$ satisfies the above condition and is materialized at stage two in an optimal solution, then replace v_i by the set of views $V(v_i, Q_2^\ell)$ and the solution remains optimal.

Note that the first and second reductions on the same set of views could be conducted simultaneously. It follows that we can now derive the reductions on the view set V_1 based on Observations 1 and 3, and derive the reductions on the view set V_2^ℓ based on Observations 2 and 4. We denote the reduced view sets of V_1 and V_2^ℓ as $\overline{V_1}$ and $\overline{V_2^\ell}$, for $\ell = 1$ to L, respectively.

We further define a new set of views for each scenario ℓ that we refer to as $\overline{V_{12}^\ell}$. This would be the set of views that are materialized at stage 1 and potentially utilized for scenario ℓ at stage 2. In the following observation, we show that $\overline{V_{12}^\ell} = \overline{V_1} \cap V_2^\ell$.

Observation 5 *For each view v_i that is materialized at stage one and kept from stage one to stage two for the ℓ^{th} scenario, $v_i \in \overline{V_1} \cap V_2^\ell$.*

Proof. The proof of Observation 5 is straightforward.

Observation 6 *$\overline{V_2^\ell} \subseteq \overline{V_{12}^\ell} \subseteq \overline{V_1}$, for $\ell = 1, \ldots, L$.*

Proof. Firstly, $\overline{V_{12}^\ell} \subseteq \overline{V_1}$ could be obtained directly from the definition of $\overline{V_{12}^\ell}$. Secondly, we show $\overline{V_2^\ell} \subseteq \overline{V_1}$. Noting that $\overline{V_2^\ell} \subseteq V_2^\ell \subseteq V_1$, we obtain that for all $v \in \overline{V_2^\ell}$, $v \in V_1$. Assume $v \notin \overline{V_1}$. Then, v must be eliminated during first or second reduction. If v satisfies the first reduction condition, that is, $|v| > |\cup_{\substack{q \in \widehat{Q} \\ q \subseteq v}} q|$. Since $Q_2^\ell \subseteq \widehat{Q}$, $\cup_{\substack{q \in Q_2^\ell \\ q \subseteq v}} q \subseteq \cup_{\substack{q \in \widehat{Q} \\ q \subseteq v}} q$. Thus, $|v| > |\cup_{\substack{q \in Q_2^\ell \\ q \subseteq v}} q|$ and v is eliminated from V_2^ℓ by first reduction. And thus $v \notin \overline{V_2^\ell}$. This contradicts the condition $v \in \overline{V_2^\ell}$. If v doesn't satisfies the first but second reduction condition,

that is, $S(v) > \sum_{\substack{q\in\widehat{Q}\\ q\subseteq v}} S(q)$. Since $Q_2^\ell \subseteq \widehat{Q}$, $\sum_{\substack{q\in\widehat{Q}\\ q\subseteq v}} S(q) \geq \sum_{\substack{q\in Q_2^\ell\\ q\subseteq v}} S(q)$. Thus, $S(v) > \sum_{\substack{q\in Q_2^\ell\\ q\subseteq v}} S(q)$ and v is eliminated from V_2^ℓ by second reduction. And thus $v \notin \overline{V_2^l}$. This contradicts the condition $v \in \overline{V_2^\ell}$. Hence, the assumption does not hold and $v \in \overline{V_1}$, which proves that $\overline{V_2^\ell} \subseteq \overline{V_1}$. Thirdly, since $\overline{V_2^\ell} \subseteq V_2^\ell$, we obtain that $\overline{V_2^l} \subseteq V_2^\ell \cap \overline{V_1} = \overline{V_{12}^\ell}$.

Correspondingly, we define the sets of subscripts for $\overline{V_1}$, $\overline{V_{12}^\ell}$ and $\overline{V_2^\ell}$ as $\overline{I_1}$, $\overline{I_{12}^\ell}$ and $\overline{I_2^\ell}$, for $\ell = 1$ to L, respectively.

For each query $q_j \in Q_1$ we define $\overline{V_{1j}} = \{v_i \in \overline{V_1} : v_i \supseteq q_j\}$, and for each query $q_j \in Q_2^\ell$ we define $\overline{V_{12j}^\ell} = \{v_i \in \overline{V_{12}^\ell} : v_i \supseteq q_j\}$ and $\overline{V_{2j}^\ell} = \{v_i \in \overline{V_2^\ell} : v_i \supseteq q_j\}$, for $\ell = 1$ to L. It follows that $\overline{V_{2j}^\ell} \subseteq \overline{V_{12j}^\ell}$ by Observation 6. Again, we define the associated sets of subscripts for $\overline{V_{1j}}$, $\overline{V_{12j}}$ and $\overline{V_{2j}^\ell}$ as $\overline{I_{1j}}$, $\overline{I_{12j}}$ and $\overline{I_{2j}^\ell}$, for $\ell = 1$ to L, respectively.

Example 1 (Continued). View $\{c, d\}$ in the set V_1 answers only one query $\{c\}$ in $\widehat{Q}$, and contains one more attribute than $\{c\}$. As a result, view $\{c, d\}$ is eliminated from V_1 by the first reduction. View $\{a, b, d\}$ in the set V_1 answers query $\{b\}$, $\{a, b\}$ and $\{b, d\}$ in $\widehat{Q}$, and the total size of the three queries is less than the size of view $\{a, b, d\}$ $(4 + 7 + 8 = 19 < 20)$. As a result, view $\{a, b, d\}$ is eliminated from V_1 by the second reduction. With the same approaches, we obtain the reduced view sets as follows.

$$\overline{V_1} = \big\{\{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{a, b, c\}, \{b, c, d\}, \{a, b, c, d\}\big\}$$
$$\overline{V_{12}^1} = \big\{\{b\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{a, b, c\}, \{b, c, d\}, \{a, b, c, d\}\big\}$$
$$\overline{V_2^1} = \big\{\{b\}, \{a, c\}, \{a, b, c\}\big\}$$
$$\overline{V_{12}^2} = \big\{\{c\}, \{a, c\}, \{b, c\}, \{b, d\}, \{a, b, c\}, \{b, c, d\}, \{a, b, c, d\}\big\}$$
$$\overline{V_2^2} = \big\{\{c\}, \{b, d\}\big\}.$$

For each query, we define the relevant view sets in answering that particular query. For example, for query $q_1 = \{a, b\} \in Q_1$ we define $\overline{V_{11}} = \big\{\{a, b\}, \{a, b, c\}, \{a, b, c, d\}\big\}$, and for query $q_6 = \{b, d\} \in Q_2^2$, we define $\overline{V_{12\,6}^2} = \big\{\{b, d\}, \{b, c, d\}, \{a, b, c, d\}\big\}$ and $\overline{V_{26}^2} = \big\{\{b, d\}\big\}$.

In Section 4, we will provide the numerical results of our reductions of the search space by a number of instances.

Modified integer programming model *IP2* We can now redefine the first stage decision variables x_i only for views in the set $\overline{V}_1$, and the first stage decision variables z_{ij} only for views $v_i \in \overline{V_{1j}}$. Similarly, we can redefine the second stage decision variables u_i^ℓ only for views $v_i \in \overline{V_{12}^\ell}$, and the second stage decision

variables y_i^ℓ only for views $v_i \in \overline{V_2^\ell}$. Since $\overline{V_{2j}^\ell} \subseteq \overline{V_{12j}^\ell}$, we can redefine the second stage variables t_{ij}^ℓ only for views $v_i \in \overline{V_{12j}^\ell}$, for $\ell = 1$ to L. It follows that we can now rewrite the integer programming model $IP1$ into an integer programming model that we denote as $IP2$ with reduced size.

$$(IP2) \quad \text{minimize} \quad \sum_{j\in J_1}\sum_{i\in\overline{I_{1j}}} d_{ij}z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j\in J_2^\ell}\sum_{i\in\overline{I_{12j}^\ell}} d_{ij}t_{ij}^\ell \tag{37}$$

$$\text{subject to} \quad \sum_{i\in\overline{I_{1j}}} z_{ij} = 1 \qquad \forall j \in J_1 \tag{38}$$

$$z_{ij} \le x_i \qquad \forall j \in J_1, \forall i \in \overline{I_{1j}} \tag{39}$$

$$\sum_{i\in\overline{I_1}} a_i x_i \le b_1 \tag{40}$$

$$\sum_{i\in\overline{I_{12j}^\ell}} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell, \quad \ell = 1,\ldots,L \tag{41}$$

$$t_{ij}^\ell \le u_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in \overline{I_{12j}^\ell} - \overline{I_{2j}^\ell}, \quad \ell = 1,\ldots,L \tag{42}$$

$$t_{ij}^\ell \le u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in \overline{I_{2j}^l}, \quad \ell = 1,\ldots,L \tag{43}$$

$$u_i^\ell \le x_i \qquad \forall i \in \overline{I_{12}^\ell}, \quad \ell = 1,\ldots,L \tag{44}$$

$$\sum_{i\in\overline{I_2^\ell}} a_i y_i^\ell \le b_2, \quad \ell = 1,\ldots,L \tag{45}$$

$$\sum_{i\in\overline{I_{12}^\ell}} a_i u_i^\ell + \sum_{i\in\overline{I_2^\ell}} a_i y_i^\ell \le b_1, \quad \ell = 1,\ldots,L \tag{46}$$

$$\textit{All variables are binary} \tag{47}$$

The optimal solution of $IP2$ is also an optimal solution of $IP1$. Thus, given an SVS problem, we could solve it by solving $IP2$. Since $\overline{V_{1j}}$, $\overline{V_{12j}^\ell}$, $\overline{V_{2j}^\ell}$, $\overline{V_{12}^\ell}$ and $\overline{V_2^\ell}$ are all subsets of $\overline{V_1}$, we could obtain an upper bound $(n_1+\sum_{\ell=1}^{L} n_2^\ell+2L+1)|\overline{V_1}|$ for the number of variables and an upper bound $(n_1+\sum_{\ell=1}^{L} n_2^\ell+2L)(|\overline{V_1}|+1)+1$ for the number of constraints in $IP2$. Compared with $IP1$, both numbers are dramatically reduced, and thus the size of $IP2$ can be significantly smaller than $IP1$. In section 4, we will provide numerical results for the comparisons of $IP1$ and $IP2$.

4 Experimental results of solving the SVS problem

In this section, we present the results of a computational experiment with models $IP1$ and $IP2$ in order to examine (i) the effectiveness of the model reductions that we proposed in Section 3.2, and (ii) the scalability of the model $IP2$. We

construct a collection of instances of the SVS problem with varying sizes using a number of databases of the TPC-H benchmark [15]. We then solve each instance using models $IP1$ and $IP2$. We observe that the search spaces of views are significantly reduced in $IP2$ compared with $IP1$, which allows us to use $IP2$ to solve several larger instances of the SVS problem of realistic size. We also observe that some realistic-size instances could not be solved due to insufficient memory. We have implemented all of our algorithms in C++, and have used CPLEX 11 [9] to solve the integer programming models $IP1$ and $IP2$.

In Section 4.1 we describe how we construct each of the instances in our experiments. In Sections 4.2 and 4.3 we evaluate the effectiveness of our models for solving the SVS problem, and report our findings. Finally in Section 4.4 we present a summary of our experimental results.

4.1 Constructing instances

The input parameters for an instance of the SVS problem are a given database $\mathcal{D}$, query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, \ldots, Q_2^L\}$, associated probability vector $\boldsymbol{p} = (p_1, \ldots, p_L)$, and the space limits b_1 and b_2. Thus, each instance is identified by $(\mathcal{D}, Q_1, \mathcal{Q}_2, \boldsymbol{p}, b_1, b_2)$.

In this section, all the instances are constructed based on three different databases of the TPC-H benchmark [15]. More specifically, we use a 7-attribute database, a 13-attribute database and a 17-attribute database to construct the collection of instances in our experiments. Each database was obtained by adding, to the original stored TPC-H tables generated with scale factor one, a single relation, with either 7, 13, or 17 grouping attributes, that results from the natural join of a subset of the set of these base relations. We measure the size of each view in terms of the number of bytes that it occupies, and estimate this value using analytical methods.

For all instances in this section, we set $L = 2$. Equivalently, there are 2 potential query sets, Q_2^1 and Q_2^2, that occur at stage 2, with respective probabilities p_1 and p_2. We further assume that $p_1 = p_2 = 0.5$.

In order to construct query sets Q_1, Q_2^1 and Q_2^2 for each instance, we comply with the following principles. In each query set, each query is generated over the given database randomly. In addition, the total sizes of all the query sets are "approximately" the same. More specifically, the difference between the sizes of any two query sets is no more than the size of the raw-data view.

Given a database fact table (that is, stored relation) $\mathcal{D}$ with K attributes, in order to randomly choose a query q, we first determine the number of attributes in q by randomly generating an integer t from $\{1, 2, \ldots, K-1\}$. 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 database $\mathcal{D}$.

To determine the queries in each query set, we first choose an integer n_1, ranging from 20 to 50, as the number of queries in Q_1. For some instances, we also set n_1 to several larger values. Then, we randomly generate n_1 queries over database $\mathcal{D}$ as query set Q_1. In order to obtain query sets Q_2^1 and Q_2^2, we randomly generate queries over database $\mathcal{D}$ successively, for as long as the total

size of the generated queries is less than or equal to $Size(Q_1)$, but not less than $Size(Q_1) - Size(v_1)$, where $Size(\cdot)$ denotes the total size of the queries in a query set $(\cdot)$, and v_1 denotes the raw-data view. Thus, we obtain that

$$S(v_1) + Size(Q_2^\ell) \geq Size(Q_1) \geq Size(Q_2^\ell), \qquad \ell = 1, 2 \tag{48}$$

The difficulty of solving a specific instance of the SVS problem depends on the relative size of the storage spaces as compared with the size of the queries. Suppose the size of storage space b_1 is expressed as

$$b_1 = \alpha \cdot Size(\widehat{Q}) \tag{49}$$

where α is a non-negative real number and $\widehat{Q} = Q_1 \cup Q_2^1 \cup Q_2^2$.

If $b_1 \geq S(v_1)$, the problem is feasible, since we could always materialize the raw-data view and use it to answer all the queries. And if $b_1 < S(v_1)$, there is no guarantee that the SVS problem is feasible.

If $\alpha \geq 1$, the SVS problem becomes trivial, since the best solution could be materializing all the views with the same sets of attributes as the queries in $\widehat{Q}$ at stage 1. Thus, in order for an instance of the SVS problem to be nontrivial, we need to have $0 < \alpha < 1$.

Note that $b_2 \leq b_1$ always holds on the space limits b_1 and b_2. Suppose b_2 is expressed as

$$b_2 = \beta b_1 \tag{50}$$

where $0 \leq \beta \leq 1$.

If $b_2 \geq \max_{Q \in \{Q_2^1, \ldots, Q_2^L\}} Size(Q)$, the second stage sub-problem of the SVS problem becomes trivial, since at stage 2, for each scenario ℓ we can materialize all the views with the same set of attributes as the queries in Q_2^ℓ.

In this section, for all instances based on the TPC-H datasets we choose $\alpha = 0.2$ and $\beta = 0.5$, that is, the storage space limit b_1 is equal to one-fifth of the sum of the sizes of the queries in $\widehat{Q}$, and b_2 is one-half of b_1. It follows that

$$b_1 = 0.2Size(\widehat{Q}) \leq (0.2)(3Size(Q_1)) = 0.6Size(Q_1)$$

$$b_2 = 0.5b_1 \leq (0.5)(0.6Size(Q_1)) \leq 0.3Size(Q_1)$$

which guarantees that the instances of the SVS problem are nontrivial.

4.2 Reducing the search space

In this subsection, we compare the sizes of the models $IP1$ and $IP2$ for some instances of the SVS problem. More specifically, we constructed 8 instances for the 7-attribute TPC-H database and 8 instances for the 13-attribute TPC-H database. The number of queries in Q_1 for each instance ranges from 20 to 50. For each instance, we compare the number of decision variables x_i, u_i^ℓ and y_i^ℓ, for models $IP1$ and $IP2$, as shown in Table 1 and Table 3. For each instance, we also report the total number of variables and constraints, as well as the time that it takes to build the model and time that it takes to solve the model by the

Table 1. Comparison of search spaces in models $IP1$ and $IP2$, for instances over the 7-attribute TPC-H database

inst-ance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	number of x_i		number of u_i^1		number of y_i^1		number of u_i^2		number of y_i^2	
		IP1 $\|V_1\|$	IP2 $\|\overline{V_1}\|$	IP1 $\|V_2^1\|$	IP2 $\|\overline{V_{12}^1}\|$	IP1 $\|V_2^1\|$	IP2 $\|\overline{V_2^1}\|$	IP1 $\|V_2^2\|$	IP2 $\|\overline{V_{12}^2}\|$	IP1 $\|V_2^2\|$	IP2 $\|\overline{V_2^2}\|$
1	(20,17,19)	125	64	80	54	80	24	108	55	108	43
2	(20,25,20)	125	84	122	82	122	47	110	75	110	38
3	(30,35,39)	127	105	124	102	124	65	120	99	120	62
4	(30,26,31)	127	92	120	88	120	42	124	89	124	58
5	(40,45,41)	127	109	122	104	122	72	123	105	123	77
6	(40,38,35)	126	103	120	98	120	62	116	99	116	44
7	(50,48,50)	127	119	121	113	121	91	126	118	126	85
8	(50,47,53)	127	113	124	110	124	74	125	111	125	87

Table 2. Comparison of sizes and computing times in models $IP1$ and $IP2$, for instances over the 7-attribute TPC-H database

inst-ance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	number of variables		number of constraints		time to build model (sec.)		time to solve model (sec.)	
		IP1	IP2	IP1	IP2	IP1	IP2	IP1	IP2
1	(20,17,19)	1621	901	1425	873	0.015	0.000	0.734	0.469
2	(20,25,20)	1937	1315	1838	1288	0.015	0.000	1.500	1.235
3	(30,35,39)	2575	2135	2527	2076	0.000	0.016	0.938	0.859
4	(30,26,31)	2385	1761	2330	1738	0.000	0.015	8.312	6.969
5	(40,45,41)	2873	2523	2859	2456	0.015	0.016	6.297	5.640
6	(40,38,35)	2600	2139	2556	2139	0.015	0.016	2.031	1.859
7	(50,48,50)	3447	3209	3459	3122	0.016	0.015	8.156	6.110
8	(50,47,53)	3437	3060	3455	3001	0.015	0.015	3.125	1.625

Table 3. Comparison of search spaces in models $IP1$ and $IP2$, for instances over the 13-attribute TPC-H database

inst-ance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	number of x_i		number of u_i^1		number of y_i^1		number of u_i^2		number of y_i^2	
		IP1 $\|V_1\|$	IP2 $\|\overline{V_1}\|$	IP1 $\|V_2^1\|$	IP2 $\|\overline{V_{12}^1}\|$	IP1 $\|V_2^1\|$	IP2 $\|\overline{V_2^1}\|$	IP1 $\|V_2^2\|$	IP2 $\|\overline{V_{12}^2}\|$	IP1 $\|V_2^2\|$	IP2 $\|\overline{V_2^2}\|$
1	(20,14,11)	7712	540	5696	525	5696	74	1338	397	1338	43
2	(20,30,22)	7871	1664	6807	1656	6807	391	6212	1604	6212	115
3	(30,30,34)	7672	1484	5769	1461	5769	347	7536	1481	7536	622
4	(30,32,38)	8064	2370	7168	2343	7168	283	7792	2352	7792	682
5	(40,45,34)	8176	3260	7856	3249	7856	1138	7434	3168	7434	284
6	(40,43,38)	8069	3250	6701	3108	6701	743	7968	3232	7968	395
7	(50,45,58)	8144	3335	7213	3233	7213	714	7616	3309	7616	989
8	(50,48,48)	8186	4092	8112	4085	8112	870	8072	4081	8072	938

Table 4. Comparison of sizes and computing times in models $IP1$ and $IP2$, for instances over the 13-attribute TPC-H database

instance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	number of variables		number of constraints		time to build model (sec.)		time to solve model (sec.)	
		IP1	IP2	IP1	IP2	IP1	IP2	IP1	IP2
1	(20,14,11)	74,472	6,352	48,114	6,550	0.297	0.266	13.859	1.094
2	(20,30,22)	87,649	21,381	73,129	22,042	0.453	0.391	72.610	10.047
3	(30,30,34)	92,164	24,247	78,437	23,866	0.516	0.453	252.484	50.375
4	(30,32,38)	108,860	36,907	98,057	37,407	0.625	0.516	135.375	46.406
5	(40,45,34)	130,380	58,623	120,144	59,060	0.750	0.672	75.594	25.656
6	(40,43,38)	120,335	54,729	108,966	55,669	0.719	0.672	253.078	69.531
7	(50,45,58)	127,522	62,275	116,430	62,234	0.813	0.766	76.609	39.766
8	(50,48,48)	157,430	86,262	148,999	86,871	0.890	0.813	88.328	27.078

CPLEX IP solver, for models $IP1$ and $IP2$. The results are shown in Table 2 and Table 4.

We observe that the number of views in the search space for model $IP1$ is significantly reduced in $IP2$ over all the instances. It is also observed that the size of $IP2$, when expressed by the total number of variables and constraints, is much smaller than that of $IP1$, yet the optimal solution of $IP2$ is also optimal for the model $IP1$. While there is no significant difference between the time to build the models $IP1$ and model $IP2$, the time to solve $IP2$ is significantly smaller than that of $IP1$, especially for relatively large problems, such as instances based on the 13-attribute database.

We observe that the reduction in the number of each group of decision variables and the reduction in the size of the model, expressed by the reduction in the total number of variables and constraints, are relatively large for instances with a small ratio of the number of queries over the total number of views in the database. For instances with larger ratios, the reduction rate decreases. More specifically, we compare the results for instances over the 7-attribute database in Tables 1 and 2. When we increase the number of queries in each query set of the instances, the ratio of the number of queries over the total number of views increases, and the magnitude of associated reductions in the number of variables decreases. We also compare the results for the instances with similar number of queries based on different databases. More specifically, we compare each pair of instances with same instance ID over the 7-attribute database and the 13-attribute database. Note that the number of views in the 7-attribute database and in the 13-attribute database are 128 and 8192, respectively. For each pair of instances, the one over the 13-attribute database has a relatively small ratio of the number of queries over the total number of views, and the associated reductions are relatively large.

We also observe that for instances with a relatively large reduction in the number of variables and constraints, the corresponding reduction of time to solve the model is relatively large.

4.3 Scalability of the model $IP2$

In this subsection we evaluate the scalability of the model $IP2$ to solve instances of the SVS problem. Based on the results in Tables 2 and 4, we could solve all the instances over the 7-attribute TPC-H database using $IP2$ within 10 seconds, and all the instances over the 13-attribute TPC-H database within 80 seconds. In order to examine the scalability of model $IP2$, we construct instances with the number of queries in Q_1 varying from 60 to 200 based on the 13-attribute TPC-H database, and instances with the number of queries in Q_1 varying from 20 to 40 based on the 17-attribute TPC-H database. We solved all the instances using model $IP2$, and reported the time required to build the model and the time required to solve the model for the corresponding $IP2$ based on each instance. The results are shown in Table 5 and Table 6.

Table 5. Scalability of $IP2$ for instances over the 13-attribute TPC-H database

instance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	time to build model (sec.)	time to solve model (sec.)
1	(60,58,64)	0.922	23.484
2	(60,60,63)	0.875	41.860
3	(70,63,59)	1.000	113.000
4	(70,73,72)	1.093	55.235
5	(80,80,86)	1.453	247.375
6	(80,83,75)	1.328	153.907
7	(90,103,99)	1.671	264.954
8	(90,87,79)	1.406	106.230
9	(100,98,95)	1.547	198.531
10	(100,103,98)	2.078	217.140
11	(120,122,124)	1.797	584.688
12	(140,140,139)	2.125	451.078
13	(160,161,153)	2.485	$>$ 20min
14	(180,191,192)	2.953	$>$ 20min
15	(200,202,215)	3.203	$>$ 20min

Table 6. Time of solving $IP2$ for instances over the 17-attribute TPC-H database

instance	query sets $(\|Q_1\|, \|Q_2^1\|, \|Q_2^2\|)$	time to build model (sec.)	time to solve model (sec.)
1	(20,19,21)	20.141	63.297
2	(20,10,15)	18.719	22.844
3	(30,32,28)	21.578	out of memory
4	(30,38,35)	22.219	out of memory
5	(40,48,45)	out of memory	—
6	(40,31,35)	out of memory	—

From Table 5, we observe that we could solve all the instances whose number of queries in Q_1 is no more than 140 based on the 13-attribute database within 10 minutes. However, when we increase the size of the query set, the solver fails to provide an optimal solution within the time limit of 20 minutes. Moreover, from Table 6, CPLEX fails to give an optimal solution for the instances based on the 17-attribute database where the size of the query set grows to 30. The failure is due to memory limitation. In addition, when the size of the query set grows further to 40, we are not able to even build the model due to insufficient memory.

Comparing "the time to build model" with "the time to solve the model" in Tables 2, 4, and 5, we observe that, as the number of queries in each query set of the instance increases, the time required to build the model $IP2$ grows slowly. At the same time, the time required to solve the model grows relatively fast. For all the instances, the time required to build the model is smaller than the time required to solve the corresponding model $IP2$. For the instances with larger size of query sets and larger execution times, this difference is quite significant.

4.4 Summary of observations

From the results of the above experiments, we observe that the search spaces for the view sets are significantly reduced in model $IP2$ compared with model $IP1$. As a result, the size of the model is much smaller in $IP2$. This allows us to solve the instances of problem SVS of realistic size using model $IP2$ to obtain an optimal solution, while $IP1$ may not able to solve the problem within the time limit. It is also observed that there are some realistic-size instances which could not be solved by $IP2$.

5 Value of the two-stage stochastic model

In this section we introduce appropriate models to assess the value of the two-stage stochastic programming model that we proposed in Section 2.2. In order to measure the gain obtained from the replacement mechanism of this model, we introduce the one-stage deterministic model for the SVS problem, and define the *value of two-stage versus one-stage*, which evaluates the difference between the optimal values of the model SP and the one-stage model. This value could be very significant over some instances, which indicates more benefits from the replacement mechanism of the two-stage model in units of the expect cost (time) of answering queries. In order to assess the value of the stochastic properties of the model SP, we introduce the expected value problem, and define the *value of stochastic solution*, which evaluates the difference between the expected response time achieved by solving the model SP and by solving the expect value problem. This value could be quite substantial over some instances, which indicates more benefits from taking into account the stochastic properties in the stochastic programming model in choosing a decision for the SVS problem.

In Section 5.1 we compare and contrast the model SP with a corresponding one-stage model, and introduce the value of two-stage versus one-stage. In Section 5.2 we compare the model SP with a two-stage model which is based on the expected value of random events, and introduce the value of stochastic solution. In Section 5.3 we provide several related numerical results.

5.1 Two-stage versus one-stage

In this subsection, we introduce the value of two-stage versus one-stage for solving the SVS problem. In the two-stage stochastic view selection model, we allow for a view replacement at stage 2. The *value* of two-stage versus one-stage measures the gain obtained via this replacement mechanism. We start by discussing the one-stage model, that is, the model in which we do not allow to drop or replace any materialized views at stage 2. In other words, at time 1 (stage 1), we decide to materialize a set of views S under a given space limit b_1. We use the views in S to answer all queries in Q_1, as well as all queries in the query set Q_2^ℓ that may occur at time 2 (stage 2), for $\ell = 1$ to L. We then compare the optimal value of this model with the optimal value of the corresponding two-stage model and define the *value of two-stage versus one-stage* in this context.

Let the decision variables x_i, z_{ij} and t_{ij}^ℓ be as defined in Section 2.2. It follows that we could formulate the one-stage problem as an integer programming model that we denote by OS as follows.

$$
\begin{array}{lll}
(OS) \quad \text{minimize} & \displaystyle\sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j\in J_2^\ell}\sum_{i\in I_{2j}^\ell} d_{ij}t_{ij}^\ell & \\
\text{subject to} & \displaystyle\sum_{i\in I_{1j}} z_{ij} = 1 & \forall j \in J_1 \\
& z_{ij} \le x_i & \forall j \in J_1, \quad \forall i \in I_{1j} \\
& \displaystyle\sum_{i\in I_{2j}^\ell} t_{ij}^\ell = 1 & \forall j \in J_2^\ell, \quad \ell = 1,\ldots,L \\
& t_{ij}^\ell \le x_i & \forall j \in J_2^\ell, \quad \forall i \in I_{2j}^\ell, \quad \ell = 1,\ldots,L \\
& \displaystyle\sum_{i\in I_1} a_i x_i \le b_1 & \\
& \textit{All variables are binary} &
\end{array}
\tag{51}
$$

We note that if we set $b_2 = 0$ in model SP, then the model is equivalent to the model OS.

Let $Optv(\cdot)$ represent the optimal value of a model $(\cdot)$. We have the following observation.

Observation 7 *The optimal value of OS is an upper bound on the optimal value of model SP, that is, $Optv(SP) \le Optv(OS)$.*

Proof. In model $IP1$, we add the following constraints:

$$y_i^\ell = 0, \quad \forall i \in I_2^\ell, \quad \ell = 1,\ldots,L$$

$$u_i^\ell = x_i, \quad \forall i \in I_2^\ell, \quad \ell = 1, \ldots, L$$

Then, in model $IP1$, constraints (33)-(35) are redundant, and constraint (32) becomes

$$t_{ij}^\ell \leq x_i \quad \forall j \in J_2^\ell, \quad \forall i \in I_{2j}^\ell, \quad \ell = 1, \ldots, L.$$

Note that after adding those constraints to $IP1$, the new IP model is equivalent to OS. In other words, $IP1$ is a relaxation of OS. We thus obtain that $Optv(OS)$ is an upper bound on the optimal value of model SP, that is, $Optv(SP) \leq Optv(OS)$.

We define *the value of two-stage versus one-stage* ($VTVO$) as the difference between the optimal values of the one-stage and two-stage models, namely,

$$VTVO = Optv(OS) - Optv(SP). \tag{52}$$

For a given instance of the problem, the corresponding value of $VTVO$ provides the magnitude of improvement in the optimal response time achieved by allowing the partial replacement of views at stage 2.

5.2 The value of stochastic solution (VSS)

In this subsection, we introduce the value of stochastic solution for the SVS problem. In the literature on stochastic programming [4], the *value of stochastic solution* (VSS) provides the possible gain obtained from solving the stochastic programming model as opposed to solving a model based on expected values. More specifically, for the SVS problem, we consider the model in which we make the first-stage decisions based on the "expected" scenario of stage 2, instead of a number of scenarios with associated probabilities as we do in model SP, and then we make the second-stage decisions given the actual query set occurring at stage 2. We compare this model with the model SP. The difference between the expected response times is the value of stochastic solution.

In order to examine the VSS from the perspective of the SVS problem, we first consider and define the *expected value problem*. In this model, instead of having a number of realizations of $\boldsymbol{Q}_2$, we consider the query workload at stage 2 as a deterministic query set $Q_2 = \cup_{\ell=1}^{L} Q_2^\ell$, with cost weight p_ℓ for query $q \in Q_2^\ell$. If a query is in more than one query set from $\{Q_2^\ell, \ell = 1, \ldots, L\}$, the weights are accumulated accordingly. In other words, we can define the weight w_j for each query $q_j \in Q_2$ as

$$w_j = \sum_{\ell : q_j \in Q_2^\ell} p_\ell \tag{53}$$

The expected value problem is then defined as follows. At time 1 (stage 1), we decide to materialize a set of views S_1 under a given space limit b_1. We use these views to answer all queries in Q_1. Subsequently, at stage 2 we allow for a partial replacement of the views materialized at stage 1 under space limit b_2, keeping the rest of the views of stage 1, to obtain a set of views S_2. We use these views to answer all queries in Q_2. Our objective is to minimize the total cost of

answering the queries occurring at time 1 and the cost of answering the queries occurring at time 2. We now construct a mathematical model for this problem which we refer to as EV.

Let J_2 be the set of subscripts for all queries $q_j \in Q_2$. We define

$$V_2 = \{v_i \in V : v_i \supseteq q \text{ for some } q \in Q_2\}$$

Note that $V_2 = \cup_{\ell=1}^{L} V_2^\ell$, where V_2^ℓ is as defined in Section 2.2. Correspondingly, we define as I_2 the set of subscripts associated with V_2. For each query $q_j \in Q_2$ we define $V_{2j} = \{v_i \in V_2 : v_i \supseteq q_j\}$. Similarly, we define as I_{2j} the set of subscripts associated with V_{2j}.

We can now define the following second stage decision variables for all views $v_i \in V_2$.

$$u_i = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 1 and kept at stage 2} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 2} \\ 0 & \text{otherwise} \end{cases}$$

In addition, we define the second stage decision variables t_{ij} for all queries $q_j \in Q_2$, and for all views $v_i \in V_{2j}$.

$$t_{ij} = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \text{ at stage 2} \\ 0 & \text{otherwise} \end{cases}$$

Let the first stage decision variables x_i and z_{ij} be as defined in Section 2.2. It follows that the expected value problem could be formulated as an integer programming model that we denote by EV, as follows.

$$\begin{array}{llll} (EV) & \text{minimize} & \displaystyle\sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}z_{ij} + \sum_{j\in J_2}\sum_{i\in I_{2j}} w_j d_{ij} t_{ij} & \\ & \text{subject to} & \displaystyle\sum_{i\in I_{1j}} z_{ij} = 1 & \forall j \in J_1 \\ & & z_{ij} \le x_i & \forall j \in J_1, \quad \forall i \in I_{1j} \\ & & \displaystyle\sum_{i\in I_1} a_i x_i \le b_1 & \\ & & \displaystyle\sum_{i\in I_{2j}} t_{ij} = 1 & \forall j \in J_2 \\ & & t_{ij} \le u_i + y_i & \forall j \in J_2, \quad \forall i \in I_{2j} \\ & & u_i \le x_i & \forall i \in I_{2j} \\ & & \displaystyle\sum_{i\in I_2} a_i y_i \le b_2 & \\ & & \displaystyle\sum_{i\in I_2} a_i (u_i + y_i) \le b_1 & \\ & & \textit{All variables are binary} & \end{array} \tag{54}$$

We denote by $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ the optimal values of the first stage variables in the model EV. We refer to $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ as the *expected value solution* (see [4]). The value

of the stochastic solution measures how poor a decision $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ is in terms of the model SP. We define the *expected result of using the EV solution* as

$$EEV = \sum_{j \in J_1} \sum_{i \in I_{1j}} d_{ij} \bar{z}_{ij} + \boldsymbol{E_{Q_2} \Psi(\bar{x}, Q_2)} \tag{55}$$

where $\boldsymbol{\Psi}(\cdot)$ is defined in (20)-(26). We make the following observation.

Observation 8 *EEV is an upper bound on the optimal value of model SP, that is, $Optv(SP) \leq EEV$.*

Proof. EEV is equal to the optimal value of model $IP1$ after fixing the first stage decision variables x_i and z_{ij} to be $\bar{x}_i$ and $\bar{z}_{ij}$, respectively, for all i and all j. Thus, EEV is an upper bound on the optimal value of model SP.

We compare EEV and $Optv(SP)$, and define the *value of stochastic solution* (VSS) as the difference between these values, namely,

$$VSS = EEV - Optv(SP). \tag{56}$$

For a given instance of the problem, the corresponding value of VSS provides the magnitude of improvement in the expected response time achieved by solving the stochastic programming model versus solving the smaller expected value problem EV.

5.3 Numerical results

In this subsection we conduct several computational experiments over a number of instances to assess the $VTVO$ and the VSS for the model SP. We observe that the magnitudes of both the $VTVO$ and the VSS vary among instances. This relies on the structure and properties of both the database and the query workloads. We also observe that for each of the two values, there are some instance where the corresponding value is very significant. This indicates that the model SP is very beneficial over these instances.

Numerical results on $\boldsymbol{VTVO}$ In order to obtain $VTVO$, the value of two-stage versus one-stage, we need to solve the model OS. Note that the model OS is equivalent to the deterministic one-stage model $OVIP'$ introduced in [2] if we define the frequency f_j for each query q_j as follows. For each query occurring only at stage 1 we define its frequency as 1, for each query occurring only at stage 2 we define its frequency as w_j, and for each query occurring at both stage 1 and stage 2 we define its frequency as $1 + w_j$, where w_j is defined in (53).

$$f_j = \begin{cases} 1 & \forall j \in Q_1 - Q_2 \\ w_j & \forall j \in Q_2 - Q_1 \\ 1 + w_j & \forall j \in Q_1 \cap Q_2 \end{cases} \tag{57}$$

Thus, we can now apply the exact methods introduced in [2] to solve the model OS.

Example 1 (Continued). For the example introduced in Section 2.1, the optimal value of model OS is 40.5. As a result, we obtain that $VTVO$ is 6.5 (=40.5-34).

In a computational experiment we constructed and solved a number of instances of the two-stage problem with varying sizes over different datasets. For each instance we obtained the corresponding value of $VTVO$. We observed that the relative magnitude of $VTVO$ varies among these instances. While in some instances this value is relatively small, in other instances it is quite significant. Following is a numeric example in which the value of $VTVO$ is relatively high.

Example 2. We construct this example based on a 13-attribute type I non-symmetric synthetic dataset [8] D_I. The master table contains all the possible entries by taking different values over the 13 attributes. The numbers of different values that the 13 attributes take are 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4 and 4, respectively. We apply a similar approach as in Section 4.1 to obtain the query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ for this instance. The number of queries in each query set is 20, 15 and 21, respectively. We set $\boldsymbol{p} = (0.5, 0.5)$ and $\boldsymbol{b} = (b_1, b_2) = (1632334, 816167)$. Then the instance is represented by $(D_I, Q_1, \mathcal{Q}_2, \boldsymbol{p}, \boldsymbol{b})$.

We compare the optimal values of models SP and OS, and we also report $VTVO$ and the ratio of $VTVO$ over $Optv(SP)$ in Table 7.

Table 7. Results of $VTVO$ over Example 2

$Optv(SP)$	$Optv(OS)$	$VTVO$	$VTVO/Optv(SP)$
1,810,103	10,892,471	9,082,368	501.760%

We observe that in this instance the value of $VTVO$ is more than 5 times the optimal value of model SP. In other words, in this instance, applying a two-stage model, that is, model SP, instead of a one-stage model reduces the corresponding response time by more than 80%. Thus, the stochastic view selection model could be very beneficial over some instances.

Numerical results on VSS In order to obtain VSS, the value of stochastic solution, we need to solve the model EV and calculate EEV. In the model EV, we are given a deterministic query set Q_2 at stage 2. Thus, the model EV could be considered as a special case of the model SP where there is only one scenario at stage 2. We can apply a similar approach as in Section 3.2 to reduce the search spaces of views for the model EV, and solve it to obtain the EV solutions. We then plug the EV solutions into equation (55), and solve the second stage problem to obtain EEV.

Example 1 (Continued). For the example introduced in Section 2.1 we compare

the first stage solutions over models SP and EV. The optimal value of the model SP is 34. We fix the first stage solutions to be as obtained in model EV, solve model SP, and obtain $EEV = 35$. Thus, $VSS = EEV - Optv(SP) = 1$.

Again in a computational experiment, we solved model EV and calculated the corresponding value EEV over a collection of instances of varying sizes bases on different types of datasets. We then calculated the associated value of VSS and the ratio of VSS over the optimal value of SP for each instance. We observed that the relative value of VSS varies among these instances, and that its magnitude depends on both the structure of the underlying database and on the corresponding size of the views, as well as on the structure of the query set. While in some instances the value of VSS is relatively small, in other instances it can be quite substantial. Following is a specific example in which this value is relatively high.

Example 3. This example is based on a 13-attribute symmetric synthetic dataset [8] $D_S(13; 2)$ (denoted as D_S for short). Each attribute takes 2 different values, and the master table contains all the possible entries by taking different values over the 13 attributes. The number of queries in query sets Q_1, $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ are 20, 19, and 20, respectively. We set $\boldsymbol{b} = (17858, 8929)$ and $\boldsymbol{p} = (0.5, 0.5)$. Then the instance is represented by (D_S, Q_1, $\mathcal{Q}_2$, $\boldsymbol{p}$, $\boldsymbol{b}$).

We compare VSS with the optimal value of model SP and report the results in Table 8.

Table 8. Results of VSS over Example 3

$Optv(SP)$	$Optv(EV)$	EEV	VSS	$VSS/Optv(SP)$
27,139	41,841	35,686	8,574	31.49%

As observed in Table 8, it is beneficial to take into account the stochastic properties of the future when performing model formulation of the stochastic view selection problems. In this instance, the expected response time would be more than 31% higher if we used the expected value model EV instead of the stochastic programming model SP.

6 The expected value of perfect information ($EVPI$)

In this section, we study the *expected value of perfect information* ($EVPI$) (see [4]) for the stochastic view selection problem as we defined in Section 2.1. The $EVPI$ evaluates the difference between the expected response time of the SVS problem and the expected response time under the perfect information of stage 2. It provides a "budget" for the decision maker to obtain the perfect information. We first introduce the ℓ^{th} scenario problem, and propose an integer programming model that we refer to as IP^ℓ for the problem. We then define the

$EVPI$ based on the optimal values of the models IP^{ℓ}s. Subsequently, we conduct a computational experiment on $EVPI$, and observe that the magnitude of $EVPI$ varies among instances. For some instances the $EVPI$ is very significant, which indicates that it is very beneficial in units of the expected response time by obtaining the perfect information of the actual query workload at stage 2.

In the literature on stochastic programming [4], $EVPI$ measures the maximum amount a decision maker would be ready to pay in return for complete information about the future. More specifically, in the SVS problem, we would benefit in units of the expected response time from the perfect information of the actual query workload occurring at stage 2. And these benefits, that is the $EVPI$, is the maximum amount a decision maker would pay to obtain the perfect information of stage 2.

Assume we know in advance which query set will actually occur at stage two. We study the stochastic view selection model that results from a particular realization of the scenarios. In other words, for each scenario ℓ we make the associated first stage and second stage decisions exclusively for it and obtain the minimum cost of answering queries at stage 1 and stage 2 under the ℓ^{th} scenario. We refer to the problem as the ℓ^{th} *scenario* problem.

We define $V_1^{\ell} = \{v_i \in V : v_i \supseteq q \text{ for some } q \in Q_1 \cup Q_2^{\ell}\}$. Note that $V_2^{\ell} \subseteq V_1^{\ell} \subseteq V_1$, where V_1 and V_2^{ℓ} are as defined in Section 2.2. For each query $q_j \in Q_1$ we define $V_{1j}^{\ell} = \{v_i \in V_1^{\ell} : v_i \supseteq q_j\}$. Correspondingly we define the sets of subscripts associated with V_1^{ℓ} and V_{1j}^{ℓ} as I_1^{ℓ} and I_{1j}^{ℓ}, respectively.

We can now redefine the first stage variables of the ℓ^{th} sub-problem (scenario) as follows, for $\ell = 1$ to L.

$$x_i^{\ell} = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at time 1 in the } \ell^{\text{th}} \text{ scenario problem} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in I_1^{\ell}$$

$$z_{ij}^{\ell} = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \text{ at time 1 in the } \ell^{\text{th}} \text{ scenario problem} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in J_1, \quad \forall i \in I_{1j}^{\ell}.$$

Let the second stage variables u_i^{ℓ}, y_i^{ℓ} and t_{ij}^{ℓ} be as defined in Section 2.2. It follows that we can now formulate an integer programming model that we denote by IP^{ℓ} for the ℓ^{th} scenario problem, for $\ell = 1$ to L.

$$(IP^{\ell}) \qquad \text{minimize} \quad \sum_{j\in J_1}\sum_{i\in I^{\ell}_{1j}} d_{ij}z^{\ell}_{ij} + \sum_{j\in J^{\ell}_2}\sum_{i\in I^{\ell}_{2j}} d_{ij}t^{\ell}_{ij} \tag{58}$$

$$\text{subject to} \quad \sum_{i\in I^{\ell}_{1j}} z^{\ell}_{ij} = 1 \qquad \forall j \in J_1 \tag{59}$$

$$z^{\ell}_{ij} \le x^{\ell}_i \qquad \forall j \in J_1, \quad \forall i \in I^{\ell}_{1j} \tag{60}$$

$$\sum_{i\in I^{\ell}_1} a_i x^{\ell}_i \le b_1 \tag{61}$$

$$\sum_{i\in I^{\ell}_{2j}} t^{\ell}_{ij} = 1 \qquad \forall j \in J^{\ell}_2 \tag{62}$$

$$t^{\ell}_{ij} \le u^{\ell}_i + y^{\ell}_i \qquad \forall j \in J^{\ell}_2, \quad \forall i \in I^{\ell}_{2j} \tag{63}$$

$$u^{\ell}_i \le x^{\ell}_i \qquad \forall i \in I^{\ell}_2 \tag{64}$$

$$\sum_{i\in I^{\ell}_2} a_i y^{\ell}_i \le b_2 \tag{65}$$

$$\sum_{i\in I^{\ell}_2} a_i (u^{\ell}_i + y^{\ell}_i) \le b_1 \tag{66}$$

$$\textit{All variables are binary} \tag{67}$$

Letting $Optv(IP^{\ell})$ denote the optimal value of the objective function of model IP^{ℓ} under the ℓ^{th} scenario, we define the *expected optimal value under perfect information* as the expected value of $Optv(IP^{\ell})$ over all scenarios. In the literature on stochastic programming [13] and [4], this value is known as *the wait-and-see solution* and it is denoted by WS. We have that

$$WS = \sum_{\ell=1}^{L} p_{\ell} \cdot Optv(IP^{\ell}). \tag{68}$$

Observation 9 *WS is a lower bound on the optimal value of model SP, that is, $WS \le Optv(SP)$.*

Proof. Suppose we redefine x^{ℓ}_i for all $i \in I_1$, and redefine z^{ℓ}_{ij} for all $j \in J_1$ and for all $i \in I_{1j}$, for $\ell = 1$ to L. We now modify model IP^{ℓ} by replacing the subscripts sets I^{ℓ}_1 and I^{ℓ}_{1j} in the objective function (58) and constraints (59)-(61) by I_1 and I_{1j}, respectively. The resulting model that we denote by modified-IP^{ℓ} has the same optimal solutions as IP^{ℓ}. Thus, it is equivalent to IP^{ℓ}. We combine the L modified-IP^{ℓ} models into one integer programming model that we refer to as IP^{*}. It is to minimize

$$\sum_{\ell=1}^{L} p_{\ell}\Big(\sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}z^{\ell}_{ij} + \sum_{j\in J^{\ell}_2}\sum_{i\in I^{\ell}_{2j}} d_{ij}t^{\ell}_{ij}\Big)$$

subject to the collection of all the constraints of submodel modified-IP^ℓ for $\ell = 1$ to L. Note that the optimal value of the model IP^* is equal to the expected value of $Optv(IP^\ell)$ over L scenarios. It follows that $WS = Optv(IP^*)$. Suppose we add the following constraints into model IP^* to obtain modified-IP^*.

$$\begin{aligned} z_{ij} &= z_{ij}^\ell, \qquad \forall j \in J_1 \quad \forall i \in I_{1j}, \quad \ell = 1, \ldots, L \\ x_i &= x_i^\ell, \qquad \forall i \in I_1, \quad \ell = 1, \ldots, L \end{aligned}$$

We can now rewrite the objective function of modified-IP^* as

$$\sum_{j \in J_1} \sum_{i \in I_{1j}} d_{ij} z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j \in J_2^\ell} \sum_{i \in I_{2j}^\ell} d_{ij} t_{ij}^\ell$$

which is the same as the objective function of $IP1$. We compare the constraints of modified-IP^* and model $IP1$ and observe that modified-IP^* is equivalent to $IP1$. In other words, model IP^* is a relaxation of $IP1$. Thus, the wait-and-see solution is a lower bound on the optimal value of model SP, that is, $WS \leq Optv(SP)$.

In the literature on stochastic programming [4], the difference between the optimal value of the stochastic programming model and its corresponding wait-and-see solution is referred to as the expected value of perfect information. We denote it by $EVPI$:

$$EVPI = Optv(SP) - WS. \tag{69}$$

6.1 Experimental results on $EVPI$

To obtain $EVPI$, the expected value of perfect information, we first solve each submodel IP^ℓ. Note that the IP^ℓ subproblem could be considered as an SVS problem with only one potential query set Q_2^ℓ (that is, scenario) occurring at stage 2. Thus, we apply here approaches similar to those of Section 3.2. The focus of our approaches is to conduct first and second reductions to solve IP^ℓ more efficiently.

Example 1 (Continued). The optimal value of the model SP is 34, while the optimal values of IP^1 and IP^2 are 33 and 30, respectively. Thus, $WS = 0.5(33+30) = 31.5$. We obtain that $EVPI$ is 2.5 units.

In a computational experiment, we compared WS with the optimal value of model SP on a collection of instances with varying sizes using a number of different datasets. For each instance we evaluated the $EVPI$ and the ratio of $EVPI$ over the optimal value of SP. Similar to the earlier measures, we observed that the value of $EVPI$ could vary significantly, depending on the specifics of database and the query instance. In the following example this value is relatively large compared with other instances in our experiment.

Example 4. This example is based on a type I non-symmetric dataset [8], $D_I(10; 2, 2, 3, 3, 4, 4, 5, 5, 6, 6)$ denoted by D_I for short. We randomly choose queries of 4, 5 or 6 attributes over D_I to obtain Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$. We set $\boldsymbol{p} = (0.5, 0.5)$ and $\boldsymbol{b} = (b_1, b_2) =$ (62174, 20174). Then the instance is represented by $(D_I, Q_1, \mathcal{Q}_2, \boldsymbol{p}, \boldsymbol{b})$.

In Table 9 we report WS and the optimal value of SP, as well as the $EVPI$ and the ratio $EVPI/Optv(SP)$.

Table 9. Results of $EVPI$ over Example 4

$Optv(SP)$	WS	$EVPI$	$EVPI/Optv(SP)$
120,601	86,892	33,709	27.95%

$EVPI$ gives the expected response time that we gain by using perfect information of the future, that is, the information of the actual query workload at stage 2. Thus, the decision maker can compare the $EVPI$ with the cost of obtaining actual query workload occurring at stage 2, and then make the decision to minimize the expected cost globally.

7 Sensitivity analysis on solutions

In the two-stage stochastic view selection problem, the amount of space limit for the replacement of views at stage two, denoted by b_2, affects the optimal cost of answering the input queries. In this section we make sensitivity analysis for the impact of this space limit on solutions. We conduct a computational experiment over a number of instances, and observe that the impact of the space limit b_2 on the optimal values varies among instances. We then compare this impact with the impact of b_2 on the cost of creating additional materialized views. This allows the decision maker to choose an appropriate space limit of replacement to make the expected total cost as low as possible.

Given an instance of the SVS problem, we fix b_1, the total storage space, but we vary b_2, the available space to materialize views at stage 2, between 0 and b_1. Equivalently, given the database $\mathcal{D}$, query sets Q_1 and $\mathcal{Q}_2$, probabilities $\boldsymbol{p}$ and the total space limit b_1, we examine the impact of the rate of b_2/b_1 (denoted as β) on the solutions by changing β over the interval $[0, 1]$.

If $\beta = 0$, the replacement of views is not allowed at stage 2. It follows that the corresponding model SP does not have second stage decision variables. And thus it is equivalent to the one-stage deterministic problem introduced in Section 5.1.

If $\beta = 1$, the problem is equivalent to two one-stage deterministic view selection problems, as follows. At stage 1, we decide to materialize a set of views S_1 under space limit b_1 in order to answer queries in Q_1; at stage 2, when the actual query set Q_2 for this stage is known, we decide another set of views S_2 under space limit b_1 in order to answer queries in Q_2. In other words, for each stage, we solve a separate one-stage deterministic view selection problem.

Given an instance $(\mathcal{D}, Q_1, \mathcal{Q}_2, \boldsymbol{p}, b_1)$ and given $0 < \beta < 1$, we denote the optimal value of the corresponding model SP as $SP(\beta)$. We have the following observation.

Observation 10 Given $0 < \beta_1 < \beta_2 < 1$, the corresponding model $SP(\beta_2)$ is a relaxation of model $SVS(\beta_1)$. Thus, if models $SP(\beta_1)$ and $SP(\beta_2)$ are both feasible, $Optv(SP(\beta_2)) \leq Optv(SP(\beta_1))$.

Proof. Note that any feasible solution of model $SP(\beta_1)$ is also a feasible solution of model $SP(\beta_2)$.

By Observation 10, the optimal value of $SP(\beta)$ is a non-increasing function of β.

In a computational experiment, we solved model $SP(\beta)$ on a collection of instances with $\beta = 0, 0.1, 0.2, \ldots, 1$. We observe that the impact of β on the optimal value of $SP(\beta)$ varies among these instances, and depends on the structure of the underlying database and on the query set. Following are two examples based on two different databases where the impact of β on $Optv(SP(\beta))$ varies as β increases.

Example 5. We construct this instance using the same data as the data of Example 2. The data is based on database D_I introduced in Section 5.3. Given $(D_I, Q_1, \mathcal{Q}_2, \boldsymbol{p}, b_1)$, we evaluate $Optv(SP(\beta))$ over this instance, for $\beta = 0, 0.1, 0.2, \ldots, 1$. The results are shown in Figure 2. We observed that the optimal value of $SP(\beta)$ decreases sharply when β changes from 0 to 0.1. When β is greater than 0.4, the optimal value of model $SP(\beta)$ remains the same.

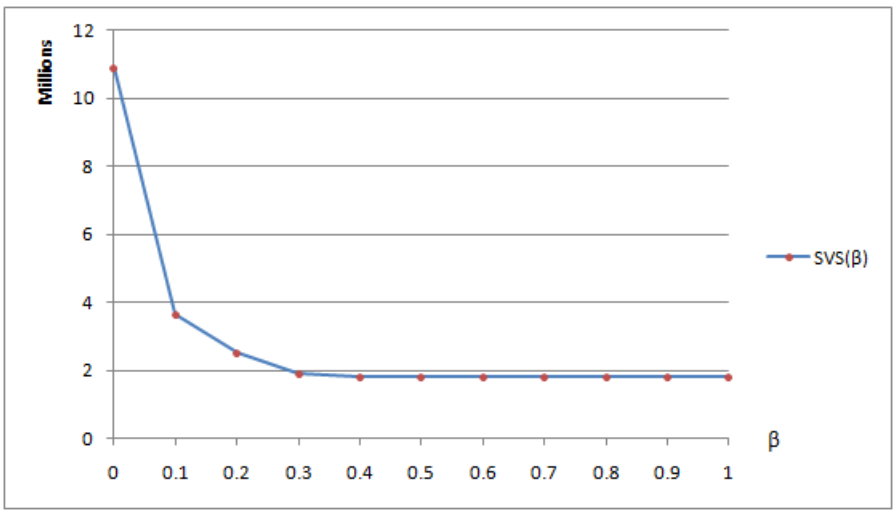

Fig. 2. The impact of β on optimal values over Example 5

Example 6. This example is based on a 13-attribute TPC-H dataset [15] denoted by D_T. The number of queries in query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ are 20, 17, and 16, respectively. We set the total space limit $b_1 = 4,739,735$ and $\boldsymbol{p} = (0.5, 0.5)$. In Figure 3 we plot the impact of β on the optimal value of $SP(\beta)$. We observe that $Optv(SP(\beta))$ declines steadily with slope around -1 as β increases from 0 to 0.9. When β is greater than 0.9, the optimal value of $SP(\beta)$ remains the same.

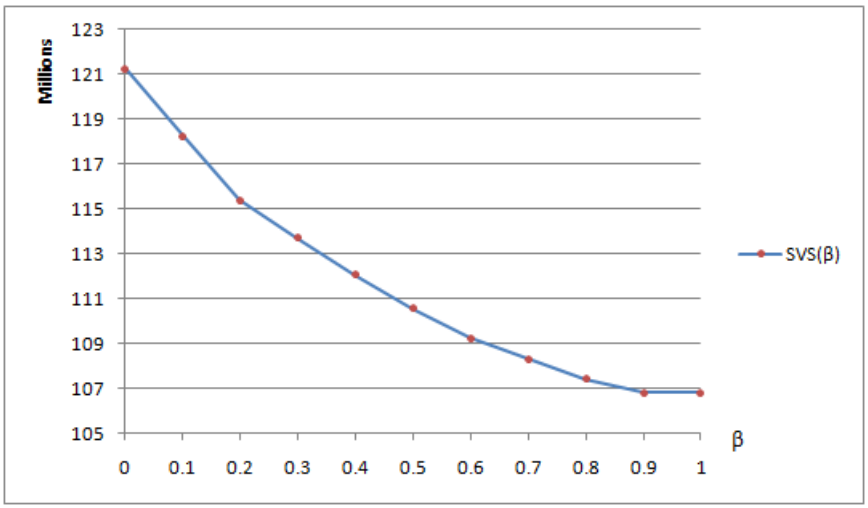

Fig. 3. The impact of β on optimal values over Example 6

Given an instance with $0 < \beta < 1$, the difference between $Optv(SP(0))$ and $Optv(SP(\beta))$ is the $VTVO$ that we introduced in Section 5.1. Since $Optv(SP(\beta))$ is a non-increasing function of β, $VTVO$ increases as β increases from 0 to 1. Larger value of $VTVO$ indicates more benefits of the two-stage stochastic programming model. Obviously, larger value of β also implies a larger magnitude of derived data that has to be materialized at stage 2. This, in turn, implies higher cost (time) of materializing views at stage 2. In other words, the benefit accrued by larger value of $VTVO$ is obtained at the cost of creating additional materialized views. For example, if $\beta = 1$, all the views materialized at stage 1 are dropped at stage 2, which induces more cost (time) of materializing views at stage 2. Thus, the decision makers would compare the cost of materializing new derived data with the difference of the optimal values between $SP(\beta)$ and $SP(1)$ to choose an appropriate value of β. For example, as observed in Figure 2, the optimal value of $SP(\beta)$ does not change much when $\beta \geq 0.3$. We can allow up to roughly 30% of the views to be updated for the second stage, since the two-stage model gives us almost all the benefits that we could obtain by treating the problem as a succession of one-stage problems.

8 Extensions and discussions

In this paper we presented the two-stage stochastic view selection problem (*SVS),* and we undertook a systematic study of the problem. We introduced a stochastic programming model for the problem, and we proposed algorithms to efficiently prune the search spaces of potentially beneficial views, while keeping at least one globally optimal solution in the search spaces. Thus, we can utilize the resulting model to solve the SVS problem optimally over a number of realistic-size instances. We compared our model with several appropriate models for the SVS problem, and demonstrated using a number of problem instances that our model is dramatically more appropriate than past approaches for solving the problem SVS. In addition, our proposed model is also beneficial in making further decisions on the SVS problem. We are now investigating the impact of the structure of the databases and the input query workload on the SVS problem, so as to improve the efficiency and scalability of our model. This will allow for a wider array of potential applications of our problem and models in the future.

References

1. Asgharzadeh, Zohreh, *Exact and inexact methods for solving the view and index selection problem for OLAP performance improvement*, Ph.D. Dissertation, North Carolina State University, 2010, 108 pages
2. Asgharzadeh, Z., R. Chirkova, Y. Fathi, "Exact and Inexact Methods for Solving the Problems of View Selection", *International Journal of Business Intelligence and Data Mining* (IJBIDM), Volume 4 Issue 3/4, November 2009
3. Asgharzadeh, Z., R. Chirkova, Y. Fathi, "An integer programming approach to the view and index selection for aggregate queries", submitted for publication. `http://www.ise.ncsu.edu/pages/people/directory/fathi/31.pdf`
4. Birdge, John, and Francois Louveaux, *Introduction to stochastic programming*, Springer 1997.
5. Chaudhuri, S. and U. Dayal, "An overview of data warehousing and OLAP technology", *SIGMOD Record*, 26(1): 65-74, 1997.
6. Gupta, H., V. Harinarayan, A. Rajaraman, and J.D. Ullman, "Index Selection for OLAP", *In Proceedings of the 13th International Conference on Data Engineering* (1997), pp. 208–219.
7. Harinarayan, V., A. Rajaraman, and J.D. Ullman, " Implementing Data Cubes Efficiently", *In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (1996), pp. 205–216.
8. Huang, R., R. Chirkova and Y. Fathi, "Synthetic Datasets", 2011. `ftp://ftp.ncsu.edu/pub/unity/lockers/ftp/csc_anon/tech/2011/TR-2011-10.pdf`
9. ILOG S.A. CPLEX 11.0 software package. `http://www.ilog.com,2007`
10. Kimball, R. and M. Ross, "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)", *Wiley Computer Publishing* 2002.
11. Li, J., Z. Asgharzadeh Talebi, R. Chirkova, and Y. Fathi, "A Formal Model for the Problem of View Selection for Aggregate Queries", *Advances in Databases and Information Systems, 9th East European Conference, Tallinn, Estonia (2005), pp.* 125–138.

12. Lightstone, Sam, "Physical Database Design for Relational Databases", *Encyclopedia of Database Systems*, *Springer*, 2009, pp. 2108-2114
13. Madansky, A., "Inequalities for stochastic linear programming problems," Management Science 6, 1960, pp. 197-204.
14. Slyke, R. M. Van, and R. Wets, "L-shaped linear programming with application to optimal control and stochastic programming", *SIAM Journal on Applied Mathematics* **17** (1969), pp. 638-663.
15. TPC-H Revision 2.1.0, 'TPC Benchmark H (Decision Support)', `http://www.tpc.org/tpch/spec/tpch2.1.0.pdf`
16. Wolsey, Laurence A. , *Integer Programming*, Wiley, 1998