ABSTRACT

HUANG, RONG. The Two-stage Stochastic View Selection Problem in OLAP Systems: Models and Algorithms. (Under the direction of Dr. Rada Chirkova and Dr. Yahya Fathi.)

We consider the problem of selecting views in an OLAP data-analysis system and propose a new paradigm to study this problem in a two-stage probabilistic environment, rather than the conventional one-stage environment. The objective is to minimize the processing time of a given collection of queries (first stage), plus the expected processing time of a (probabilistic) collection of future queries (second stage), while allowing for a partial replacement of the views in the second stage. We present a stochastic programming model for this problem and show that it is equivalent to an integer programming (IP) model. We study the structure of this IP model and reduce its size accordingly to the extent that it can be solved using available IP solvers. We then evaluate the benefits of this two-stage approach versus a comparable one-stage approach, and determine the value of perfect information in this context. In order to solve both the one-stage view selection problem and the two-stage stochastic view selection problem more efficiently, we define the cost-benefit ratio of each view which is a measure of effectiveness of the view. We conduct a theoretical analysis of the properties of the cost-benefit ratio, and employ this concept to develop exact and inexact methods to further prune the search spaces of potentially beneficial views so that we are able to solve larger instances of the problems.

The Two-stage Stochastic View Selection Problem in OLAP Systems: Models and Algorithms

by
Rong Huang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2013

APPROVED BY:

Dr. Carla Savage	Dr. Reha Uzsoy
Dr. Rada Chirkova Co-chair of Advisory Committee	Dr. Yahya Fathi Co-chair of Advisory Committee

DEDICATION

To my parents...

BIOGRAPHY

Rong Huang received her B.S. degree in Statistics and Probability from Peking University, Beijing, in 2007. She joined the Ph.D. program in Operations Reserach at North Carolina State University in Fall 2007. Her primary research interest is theory and application of mathematical programming. During her Ph.D. study, she served as a treasure of Omega Rho Honor Society NCSU chapter. She is a member of Omega Pho and INFORMS. Currently, Rong is working as a student intern at SAS Institute in Cary, NC.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors Dr. Rada Chirkova and Dr. Yahya Fathi for their supervision, thoughtful advice, kindness, patience and their continued support and encouragement throughout my Ph.D. study. Their guidance toward being a professional is an invaluable asset which I will carry in the future. Special thanks go to Dr. Carla Savage, Dr. Reha Uzsoy, and Dr. Brian Denton for serving on my committee. I highly enjoyed Dr. Savage's courses"Design and Analysis Of Algorithms" and "Graph Theory" which inspired me a lot in my research. Many thanks go to Dr. Uzsoy for his comments on my work. I have benefited a lot from Dr. Denton's Stochastic Programming course and all the enlightening advice he has given me. I also would like to thank Ran Liu, one of the most important person in my life who is always there for me. Finally, I would like to express my gratitude to my parents Chen Huang and Weiling Zeng for their motivation and support for me all the time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 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. *Materialized views*, that is precomputed and stored extra relations, are commonly used to reduce the evaluation costs of data-analysis queries in relational data-intensive systems. Intuitively, a materialized view would improve the efficiency of evaluating a query when the view relation represents the result of (perhaps time-consuming) precomputation of some subexpression of the query of interest; see [12, 28] and references therein. Materialized views with grouping and aggregation may be especially attractive for evaluating data-analysis queries, because the relations for such views store in compact form the results of (typically expensive) preprocessing of large amounts of data. Selecting appropriate materialized views 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.

As an illustration, consider the following scenario. Large retail 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 to maintain 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* [9, 24], queries, users are interested in obtaining summary information of a *measure*,

Table 1.1: The Sales table

StoreID	State	ItemID	ItemCatergory	CustID	DateID	QtySold
1	NC	1	2	3	1	70
1	NC	3	2	1	2	100
1	NC	3	2	2	3	80
1	NC	4	1	1	1	100
1	NC	4	1	2	4	100
2	NC	1	2	3	2	25
2	NC	2	1	2	3	14
2	NC	4	1	1	3	12
3	SC	1	2	2	1	30
3	SC	1	2	3	4	40
3	SC	2	1	2	1	30
3	SC	3	2	1	2	100
...	...	...	...	...	...	...

Table 1.2: The Time table

DateID	Day	Month	Year
1	24	Dec	2012
2	25	Dec	2012
3	1	Jan	2013
4	2	Jan	2013
...	...	...	...

such as sales amount, 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*, *time of sale*, and *customer*.

Suppose the Sears sales data are stored in *relations* (i.e., tables) `Sales(StoreID, State, ItemID, ItemCatergory, CustID, DateID, QtySold)`, `Time(DateID, Day, Month, Year)`, and `Customer (CustID, CustType)`. Suppose also that these relations form a *star schema* [10], with `Sales` as the "fact table" and `Time` and `Customer` as "dimension tables" (see [24]), as shown in Tables 1.1, 1.2, and 1.3, respectively.

Consider a query workload with queries `Q1` and `Q2`. `Q1` asks for the total sales per item category per customer type in December 2012. `Q2` asks for the maximum sales per store per customer type in December after 2005. Then the queries could be expressed in the standard relational-query language SQL as the following grouping and aggregation over the star-schema join [24]:

Table 1.3: The Customer table

CustID	CustType
1	Business
2	Individual
3	Individual
...	...

```
Q1: SELECT ItemCategory, CustType, SUM (QtySold)
    FROM Sales NATURAL JOIN Time NATURAL JOIN Customer
    WHERE Month = ‘December’ AND Year = 2012
    GROUP BY ItemCategory, CustType;
```

```
Q2: SELECT StoreID, CustType, MAX (QtySold)
    FROM Sales NATURAL JOIN Time NATURAL JOIN Customer
    WHERE Month = ‘December’ AND Year > 2005
    GROUP BY StoreID, CustType;
```

A data-management system evaluating the queries `Q1` and `Q2` on just the base (i.e., original stored) 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 queries `Q1` and `Q2`. 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) December 2012 for query `Q1`. Note that evaluating the queries using the index `I` would still involve examining the sales data at the granularity of individual sales events. Alternatively, the data-management system could reduce the evaluation time of the queries `Q1` and `Q2` by using a *materialized view* `V1`, which is a relation that stores the total sales (`V1amt`) and maximum sales (`V2amt`) per item category per customer type per store per month per year, and is expressed in SQL as follows:

```
V1: SELECT ItemCategory, CustType, StoreID, Month, Year,
           SUM (QtySold) AS V1amt, MAX (QtySold) AS V2amt
    FROM Sales NATURAL JOIN Time NATURAL JOIN Customer
    GROUP BY ItemCategory, CustType, StoreID, Month, Year;
```

Observe that the queries `Q1` and `Q2` can be evaluated using a single relation `V1` – that is, no *joins* of relations are required in the evaluation. Using `V1` obviates the need to access a large amount of irrelevant data, and therefore the evaluation time of `Q1` and `Q2` when using `V1` could be dramatically shorter than when not using `V1`. Note also that the representation of the base data in the materialized view `V1` no longer distinguishes between individual items or sales dates.

Ideally in such a setting, in order to maximize the efficiency of query processing, all "beneficial views", such as `V1` in the example, would be precomputed and stored (*materialized*). However, typical storage-space and computational constraints on the database system limit the number of views that can be materialized. Naturally, the problem of selecting an appropriate collection of materialized views has to be addressed in the context of the objectives and limitations of each setting. This problem is commonly known as the *View-Selection Problem*. In recent years, a number of researchers have addressed the subject and developed exact and inexact methods for solving the problem in a one-stage deterministic environment, where all queries are assumed to be known in advance, and in a one-stage probabilistic environment, where we have a probability (frequency) of occurrence associated with each query in a given collection of queries. (See, for instance, [3–5, 16, 19].) In both environments, the problem is addressed in the context of one fixed query workload, and can thus be called the *One-Stage View-Selection Problem*.

However, sometimes knowledge of future query workloads may be available, which allows us to choose appropriate views to make advanced preparation for the processing of the future queries. More specifically, sometimes it is necessary to solve the problem of view selection in settings in which we are given some information about query workloads at a future point in time, in addition to the immediate query workloads. In this dissertation we study this problem for relational OLAP systems 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 the 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.

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 better 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 the derived data produced.

A further complicating factor in this context is the fact that sometimes it is difficult to predict the specific query workload that would become relevant and prominent in a future season *(stage).* For instance, political, economic, or environmental factors could all have an impact on the nature of the queries that would become relevant in the fixed future stage. This could lead the experts to forecast two or more possible query workloads for that future stage, with a probability associated with each set. This, in turn, would lead to the need to carry out the analysis in a probabilistic environment, where we account for various *scenarios* (i.e., query workloads) 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 materialized views to answer a given set of queries immediately (these are first-season, or first-stage, queries), and a plan for several collections of views, each collection associated with a possible set of future queries (these are 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 views, our decision for the first collection of views (that is, the first-stage decision) should be made with consideration for various possibilities (queries) that may occur in the second stage. The idea here is to reuse as many of the first-stage derived data as possible in the second stage.

This leads us to define a view-selection problem in a *two-stage probabilistic environment,* with (current) *stage 1* and (fixed-season future) *stage 2,* and propose models and algorithms for solving the problem. Informally, in our *Two-Stage Stochastic View-Selection Problem,* the inputs include storage limits, a query workload for stage 1, and a collection of query workloads for stage 2 with a probability of recurrence associated with each workload. (The probabilities of the stage-2 query workloads sum up to 1.) The output *at stage 1* includes (1) the set of views to be materialized for the stage-1 queries, and (2) for each stage-2 query workload, the set of views that are to be materialized at stage 2 (including those views that will remain around from stage 1) for that query workload. *At stage 2, while the time-consuming materialization of the stage-2 views is taking place, those stage-1 materialized views that remain in the system at stage 2 remain around to improve the efficiency of evaluation of stage-2 queries.* Not surprisingly, our *Two-Stage Stochastic View-Selection Problem* is NP-hard [37]. Please see Section 3.1 for the formal statement of the problem.

As an illustration, suppose that in the Sears example, the queries `Q1` and `Q2` (given earlier) constituted the stage-1 query workload. We are also given two stage-2 queries: (1) Query `Q3` asks for the total sales per item category per customer type per store in NC in January 2013, and (2) Query `Q4` asks for the total sales per state in December 2012. Suppose that either `Q3` or `Q4` will occur at stage 2, each with probability 0.5. Then, at stage 1, instead of materializing the view `V1` (as given earlier), we can materialize a view `V2` that has the total sales per item category per customer type per store per state per month per year, and use it to answer `Q1` and `Q2`. At stage 2, if `Q3` occurs, we use `V2` to answer it. Otherwise, if `Q4` occurs, we answer it by (designing at stage 1 and) materializing *at stage 2* a view `V3` that stores the total sales per state per month per year. The larger-size relation for `V2` can be used to evaluate the queries in *both* stages, assuming that `Q3` becomes prevalent in stage 2. In contrast, the smaller-size relation for `V3` would replace *at stage 2* the larger-size relation for `V2`, if query `Q4` becomes prevalent in stage 2. Here, the use of `V3`, rather than of `V2`, would result in a more efficient evaluation of the query `Q4`.

This example illustrates that our two-stage approach is more efficient and less resource intensive than the one-stage approach to view selection. Indeed, as we show in this dissertation, if we apply a one-stage solver to materialize views only at stage 1, then a greater portion of the materialized views would tend to be irrelevant at stage 2. On the other hand, applying a one-stage solver at each of the two stages would result in creating large volumes of views at stage 2, which could be unnecessarily resource intensive.

Note also that our two-stage approach to view selection is applicable to, and sufficient for, solving multi-stage view-selection problems. The reason is, the view selection takes place during the first season, and when it comes to the second season, any modification of the set of views with respect to the new season will have been applied. During the second season, we can employ this two-stage approach for view selection for the second and third seasons, and so on.

1.1 Contributions

In this dissertation, we introduce the two-stage stochastic view-selection problem and propose an approach for solving the problem. Our approach is to develop a stochastic programming (SP) model for it. More specifically, we show that the SP model that we propose in its extended form is equivalent to an integer programming (IP) model. In general, this IP model may have a large number of variables and constraints. We use the structural properties of the models to reduce the size of the search space of views, and hence the size of the corresponding IP model, to manageable levels. Our specific contributions are as follows:

1. In Section 3.1 we define the two-stage stochastic view-selection problem, and model this

problem as a two-stage stochastic programming (SP) model in Section 3.2.

2. In Section 4.1 we study the structure and properties of the extensive form of the SP model, which is an integer programming (IP) model. In Section 4.2 we develop an algorithm that effectively reduces the search space of potentially beneficial views in the IP model, and obtain a smaller IP model in Section 4.3 (called "the reduced IP model") whose solution is guaranteed to be optimal for the original SP model.

3. In Section 4.4 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. As a result, for realistic-size instances of the problem, this IP model can be solved efficiently by a commercial IP solver such as CPLEX [21].

4. In Sections 5.1-5.4 we compare our two-stage SP model with associated or related models, and present techniques to assess the value of the SP model and the value of perfect information [6] in this context.

5. In Section 6.1 we introduce an exact method and an integer programming model to solve the one-stage deterministic view-selection problem. We show through a numerical experiment in Section 6.2 that our algorithm outperforms the one introduced in [3, 4].

6. In Sections 7.1 and 7.2 we study the properties of the views that appear in the optimal solution in the context of the one-stage view-selection problem. We propose a measure of effectiveness (called *cost-benefit ratio*) associated with each candidate view, and use this measure to devise heuristic algorithms for solving larger instances of the problem in Sections 8.1-8.3. We demonstrate the effectiveness of our algorithms through a computational experiment in Section 8.4. Our experiment shows that on average, our algorithms require less execution time than the heuristic methods proposed in [3, 4], and the resulting solutions are significantly better (i.e., have lower costs).

7. In Sections 9.1 and 9.2 we discuss how this cost-benefit ratio can be employed to devise other heuristic algorithms for solving large-size instances of the two-stage stochastic view-selection problem. In Section 9.3 we conduct a computational experiment to discuss the effectiveness of this algorithm.

1.2 Dissertation structure

The remainder of this dissertation is organized as follows. In Chapter 2 we review related work. In Chapter 3, we present a formal definition of the two-stage stochastic view selection problem, and propose a stochastic integer programming model for it. In Chapter 4 we discuss the structure of the problem and the corresponding stochastic programming model, use this structure to reduce the size of the problem considerably, and provide and discuss the results of a computational experiment with this model. In Chapter 5 we present models and techniques to determine the value of using a two-stage stochastic programming model, and present several related numeric results. In Chapters 6 through 8 we focus our attention on the one-stage deterministic view-selection problem. Our primary motivation for studying the one-stage problem is to identify the structural properties of the problem that can be used to design effective algorithms for solving the two-stage problem. In the meantime, we also developed new and effective techniques for solving the one-stage problem. In Chapter 6 we propose an exact method, which is similar to the one introduced in Chapter 4, to efficiently solve the one-stage deterministic view-selection problem. In Chapter 7 we introduce the concept of cost-benefit ratio for each view in the context of the one-stage view-selection problem, and discuss its properties. In Chapter 8 we propose heuristic methods, based on our cost-benefit ratio for further reducing the size of the search space of views, in order to improve the efficiency and scalability of the IP model for solving the one-stage view-selection problem. In Chapter 9 we return to the two-stage problem, and employ the cost-benefit ratio to develop a heuristic method for solving the two-stage stochastic view-selection problem. Finally, Chapter 10 contains a few concluding remarks and recommendations for future studies.

Chapter 2

Related work

2.1 Overview of related work

The general view-selection problem has been studied in the literature with various objectives and constraints (see [33] for a survey and the design goals of the problem). In the one-stage deterministic view-selection problem, the goal is to select a set of views to materialize from scratch for answering a given set of queries with associated frequencies of occurrence. Numerous algorithms are proposed for the selection of materialized views in the conventional one-stage form of the problem (see [18] for a survey). Significant work has also been done on index selection in such settings, both on its own and alongside view selection, see, e.g., [1, 2, 7, 8, 11].

Work including [17, 19, 30] consider greedy methods for efficiently selecting views in a generalization of the OLAP setting. Harinarayan et al. in [19] present greedy algorithms for the one-stage view-selection problem in data cubes under a disk-space constraint. To minimize the sum of query and update cost under storage space constraint, Gupta et al. in [17] extend the greedy algorithm of [19] in the selection of data-warehouse materialized views. They introduce "AND-OR" view graphs to represent all possible ways to generate warehouse views, and provide several greedy heuristics for different types of "AND-OR" view graphs. Unfortunately, in [23], Karloff and Mihail disprove the strong performance bounds of these algorithms, by showing that the underlying approach of [19] cannot provide the stated worst-case performance ratios unless P=NP. To minimize query costs under a space constraint, Nadeau et al. in [30] also give a greedy algorithm, which is polynomial in the number of dimensions. That algorithm does not perform as well as other greedy heuristics.

Solutions based on randomized algorithms have been developed in the literature for the one-stage view-selection problem. Kalnis et al. in [22] use randomized algorithms, including iterative improvement, simulated annealing, random sampling, and two-phase optimization, to search the solution space of view sets to minimize query costs, under the constraints on the space or update cost.

Considerable work, including [3–5,31,38], has been done in the open literature that employs integer programming (IP) models to obtain optimal selection of derived data for query processing in a one-stage environment. Yang et al. [38] propose an integer programming (IP) model for view selection. The model uses Multiple View Processing Plans (MVPPs) to minimize the combined cost of query evaluation and view maintenance. The solution of the IP model can guarantee optimality. However, the algorithm does not take the space constraint into account. In [31] Papadomanolaskis et al. propose an Integer Linear Programming (ILP) model for index selection, with the objective of minimizing the query and update costs under a storage-space constraint for a given collection of queries using a given set of tables. Golfarelli et al. in [15] investigate the benefits of materializing views in vertical fragments in order to minimize the response time of query workloads under a disk-space constraint. Each vertical fragment includes a subset of the measures possibly taken from one or more views, aggregated on the same grouping set. They formulate the problem as an ILP model, and solve it using a standard IP solver. A line of past work [3–5] has focused on formal approaches for selection of views (with or without indexes) to minimize the cost of query processing under the storage-space constraint. Asgharzadeh Talebi et al. propose IP models to address the problem of view selection in [4], and the problem of combined view and index selection in [3, 5]. The objective in both papers is to minimize the query-processing costs under space constraints in a deterministic environment. In both cases, they propose algorithms to effectively and efficiently prune the search space of potentially beneficial views and/or indexes. They develop both exact and inexact methods for solving the problems efficiently. The exact methods of their work guarantee the optimality of the solutions. The results of that work are scalable to relatively large instances of the problem, and compare favorably with several other approaches in the literature, including [2,19,22]. However, those integer linear programming techniques are limited to the problems in the one-stage environment. To the best of our knowledge, there is no known algorithm that addresses the problem in a multi-stage probabilistic environment. The SP and IP models presented in this dissertation are most closely associated with the models discussed in [3, 4]. We extend these models and approaches to the two-stage deterministic and probabilistic environments. In Section 2.2, we provide a brief introduction of the IP model and methods proposed in [3, 4] for the one-stage view-selection problem. In this dissertation we also extend further the reach of the approach of [3–5] for solving the one-stage view-selection problem, by studying the structural relationship between the views and the queries. These results lead us to more effective techniques and algorithms for the problem.

The multi-stage deterministic view-selection problem has been studied in the literature in a dynamic environment, known as the *dynamic view-selection problem*. The dynamic view-selection problem shares similarities with the two-stage stochastic seasonal view-selection problem discussed in this dissertation, in that they both consider the selection of materialized views for incoming queries over time periods. At the same time, there are also fundamental differences between these paradigms, as described below.

There are a number of solutions, including [7, 8, 25, 26], for the dynamic view-selection problem in the context of periodical "re-calibration" of the materialized views to optimize query-evaluation costs with changes in the query patterns. In [25, 26], Kotidis et al. present DynaMat, a system that dynamically materializes information at multiple levels of granularity, in order to match the query workload under the update and space constraints. DynaMat constantly monitors incoming queries, and materializes the best set of views subject to the space constraints. During updates on the stored data, DynaMat reconciles the current materialized views and refreshes the most beneficial subset within a given maintenance window. At the same time, DynaMat can only choose to store aggregate data that are answers to the queries from the users. As a result, even though an aggregate view that can answer multiple aggregate queries may save a lot of space without losing much efficiency in answering these queries, it will never be considered by DynaMat unless it is queried. Our approach in this dissertation allows any aggregate view that can answer any of the queries to be considered for materialization.

In [34,35], Theodoratos et al. consider what they call dynamic or incremental data warehouse design, where new queries are added and need to be answered by a data warehouse. In the first phase (stage), they are given a fixed set of queries Q, and views that answer the queries with minimum cost must be selected from multi-query AND/OR-DAGs. In the second phase (stage), an incoming query workload is given, and another set of materialized views is added to the existing set of views, so that all the incoming queries and the existing queries can be answered. However, in practice there may not be extra space available for materializing new views. In this dissertation, we always require the space for storing materialized views at each stage to be no more than a given space limit.

Lawrence et al. in [27] propose a two-phase (two-stage) operation, Startup and Online, to solve the dynamic view-selection problem for OLAP. The Startup Phase (the first stage) is a one-stage view-selection routine. For the Online Phase (the second stage), the authors propose online view selection, that is, given a set of materialized views M obtained from the startup phase and a new deterministic query workloads with query frequencies, select a new set of views M', by discarding some views from M, and by adding new ones to be materialized, to minimize the query costs under the space constraint. Regarding computationally high cost of view materialization, the authors limit the storage space for the views to be materialized in the second stage. A key difference from our work is that in our model, we assume that during the first stage we have some knowledge of the second-stage queries (at least in a probabilistic sense). We make selections of views in the first stage based on this knowledge, to make advance preparations for the query processing in both stages. In contrast, Lawrence et al. deal with the problem dynamically, in the sense that their algorithm assumes no knowledge of the second-stage queries during the first stage. They consider the second-stage queries in a deterministic sense, only after the first-stage views are materialized and employed, and solve the two stages of the problem separately. As a result, a great portion of the materialized views M obtained from the startup (first) phase tends to be irrelevant in the online (second) phase, since many of the first-stage queries are no longer of interest in the second phase. In [27], in order to select views for the two phases of the problem, the authors apply the greedy algorithm proposed by Harinarayan et al. in [19] and three randomized methods proposed by Kalnis et al. in [22]. However, these methods do not guarantee optimality of the solution.

The stochastic seasonal view-selection problem proposed in this dissertation differs from dynamic view selection, in that for the stochastic seasonal view-selection problem it is natural to only consider a two-stage (two-season) view selection. The reason is that the selection of views takes place between two seasons, and when it comes to the second season, any modification of the set of views with respect to the new season will be applied. At the end of the second season, we can employ this two-stage approach for view selection between the second and third seasons, etc.

However, none of the work for the multi-stage view-selection problem has been considered in a probabilistic environment. To the best of our knowledge, Shaman [14] is the only project described in the open literature that considers using derived data to improve query performance in presence of a number of distinctly identifiable query workloads, all of which are known beforehand. Our work in this dissertation complements the agenda of Shaman, in that in Shaman, the specific query sets are not tied to specific time points or to particular occurrence probabilities. The objective of the Shaman approach is to set up one initial and, at the same time, final index configuration that would be "robust" in some sense through all the deterministic changes in the query workloads on the system, regardless of when the workload changes are to occur. This approach does not appear to be appropriate for a probabilistic-environment problem, such as the one considered here.

In this dissertation we consider view selection in a two-stage probabilistic environment. We employ stochastic programming (SP) (see [6]) and integer programming (IP) (see [37]) to address the two-stage stochastic view selection, and apply exact and inexact method to obtain efficient solutions.

2.2 An integer programming method for the one-stage view-selection problem

In this subsection, we provide a brief introduction of the models and methods proposed in [3, 4] for the one-stage view-selection problem. The models and algorithms that we propose in this dissertation for the two-stage view-selection problem are closely related to these for the one-stage models and methods.

In [4], Asgharzadeh Talebi and colleagues consider the following One-Stage View-Selection problem, which we denote by *OVS* in this dissertation: Assume we are given a collection of queries Q, with associated frequency distribution on a given star-schema data warehouse $\mathcal{D}$, and a storage limit b on the total size of the views that we may materialize. The problem is to select a collection of views to materialize, so as to minimize the total evaluation costs of the given collection of queries.

The scope of the two-stage view-selection problem which we propose in this dissertation, i.e., the type of the database, as well as the type of queries and views, is the same as in the problem *OVS*. Hence, we provide a brief introduction here; please see Section 3.1 for a detailed description. For the problem *OVS* we consider star-schema data warehouses [9, 10, 24] with a single fact table and several dimension tables. All the views to be materialized are defined, with grouping and aggregation but without selection, on the relation that is the result of the "star-schema join" [24] of all the relations in the schema. This relation is called the *raw-data view* or *raw-data table*. All the queries in the workloads in the problem *OVS* are posed on the relevant raw-data view. The query-evaluation cost is defined in the context of answering unnested select-project-join queries with grouping and aggregation using unindexed materialized views, such that each query can be evaluated using just one view and no other data. A query can be answered using a view only if the set of grouping attributes of the view is a superset of the set of attributes in the `GROUP BY` clause of the query and of those attributes in the `WHERE` clause of the query that are compared with constants. Each query (or view) is represented by the set of these relevant attributes. The evaluation cost of a query using a given view (if this view can indeed be used to answer the query) is equal to the size of the materialized view itself. Please see Section 3.1 for detailed size calculations and estimation of the views or queries.

The *search space of views* (i.e., the collection of candidate views we consider) for the problem *OVS* is the *view lattice* introduced by Harinarayan et al. in [19], which includes all the views with grouping and aggregation defined on the raw-data table. Each lattice view has grouping on some of the attributes of the database, and has aggregation on all the attributes aggregated in the input queries.

Asgharzadeh Talebi et al. [3, 4] propose an IP model for solving the problem *OVS*. This IP model has a key role in our discussions below, hence, for completeness, we present it here. Let

V denote the search space of views for a given problem OVS with a query set Q. Let I and J denote the set of subscripts associated with the sets V and Q, respectively. For each $i \in I$ we use v_i to represent both the i^{th} view in V and the collection of grouping attributes for this view. Similarly, for each $j \in J$ we use q_j to represent both the j^{th} query in Q 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. Let a_i denote the size of the view v_i, for all $i \in I$. Let d_{ij} be the evaluation cost of answering query q_j using view v_i, for all $j \in J$ and for all $i \in I$. Define the decision variables x_i and z_{ij} for all $j \in J$ and for all $i \in I$, as follows:

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

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

The problem OVS can now be formulated as the following integer programming (IP) model, denoted by $OVIP1$. We use the notation V_j, for every query q_j, to represent the collection of views each of which can be used to answer the query q_j, i.e., $V_j = \{v \in V : v \supseteq q_j\}$. In the model $OVIP1$ we use I_j to denote the set of subscripts in V_j.

$$(OVIP1) \quad \text{minimize} \quad \sum_{j \in J} \sum_{i \in I_j} d_{ij} z_{ij} \tag{2.1}$$

$$\text{subject to} \quad \sum_{i \in I} z_{ij} = 1 \qquad \forall j \in J \tag{2.2}$$

$$z_{ij} \leq x_i \qquad \forall j \in J, \forall i \in I_j \tag{2.3}$$

$$\sum_{i \in I} a_i x_i \leq b \tag{2.4}$$

$$\textit{All the variables are binary} \tag{2.5}$$

Constraint (2.2) states that each query is answered by exactly one view; constraint (2.3) guarantees that a query can be answered by a view only if the view is materialized. Constraint (2.4) limits the storage space for the views to be materialized.

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

Observation 1. *There exists an optimal solution for an instance of the problem OVS in which view v is not materialized, and hence can be removed from the search space of views, if v contains at least one attribute that is not in any of the queries it can answer.*

Observation 2. *There exists an optimal solution for an instance of the problem OVS in which view v is not materialized, and hence can be removed from the search space of views, if v is not equal to any query in the given query set Q, and the size of v is greater than or equal to the total size of queries it can answer.*

As mentioned in Asgharzadeh Talebi et al. [3, 4], these observations allow us to remove a certain number of views at the outset, thus reducing the size of the corresponding model $OVIP1$. We can still guarantee that an optimal solution of the reduced model, which we denote by $OVIP1'$, is also optimal for the original problem OVS. This, in turn, allows us to solve larger instances of the problem and to obtain the corresponding optimal solutions. Through a comprehensive computational study, Asgharzadeh Talebi et al. [3, 4] show the effectiveness of this approach in solving relatively large instances of the problem OVS.

In order to further prune the search space of candidate views, and hence the size of the corresponding IP model, Asgharzadeh Talebi et al. in [3,4] present an inexact (heuristic) method for solving the problem OVS. The basic idea of the inexact method proposed in [3, 4] is to limit the search space of views only to those views that are "relatively close" *ancestors* of the given queries. A view (or query) v_1 is an ancestor for another view (or query) v_2 if the collection of attributes that define v_2 is a proper subset of the collection of attributes that define v_1; in this context we refer to v_2 as the child (or descendant) of v_1. More specifically, Asgharzadeh Talebi et al. [3] define a set V_s as a reduced search space of views with respect to a given parameter s, where each view in V_s is the union of exactly 1, or 2, ..., or s queries in the given query set Q. This heuristic method allows larger instances of the problem to be solved in this manner, albeit they can no longer guarantee that the resulting solution is optimal for the original problem. The results of [3,4] show that their methods outperform other heuristic methods, including [2,19,22].

In this dissertation we extend the integer programming model and pruning algorithms proposed in [3, 4] to the two-stage probabilistic environment. We also study the structural relationships between the given query set Q and each view v in the view lattice V, and use this relationship to propose effective techniques for further pruning the search space of views and for reducing the size of the corresponding IP models, both in the one-stage environment and in the two-stage probabilistic environment. We experimentally compare our proposed methods with the pruning methods in [3,4] for the one-stage view-selection problem. Our proposed techniques for pruning the search space of views are complementary to those proposed in Asgharzadeh Talebi et al. [3, 4] for solving the one-stage view-selection problem, and can be employed either separately or in conjunction with these earlier techniques. We evaluate the effectiveness of our proposed techniques through a comprehensive computational study, and report our findings. We also experimentally compare our proposed methods with the pruning methods in [3, 4] for the one-stage view-selection problem.

Chapter 3

The two-stage stochastic view-selection problem

In this chapter we define the scope of the two-stage stochastic view-selection problem that we consider, i.e., the type of the databases, queries and views. Subsequently, we propose a stochastic programming (SP) model [6] for this problem. By studying the properties of the SP model, we then rewrite it equivalently as a model with fewer variables and constraints.

3.1 Formulation of the two-stage stochastic view-selection problem *SVS*

We consider a star-schema relational OLAP system [9, 10, 24], which is a data warehouse with a single fact table and several dimension tables. (See Tables 1.1, 1.2, and 1.3 in Chapter 1 for an example.) We assume that all the views to be materialized are defined, with grouping and aggregation but without selection, on the raw-data view, which is the result of the "star-schema join" [24] of all the relations in the schema. We can show formally that for each query (of the type as defined below) posed on the original database, the query can be rewritten equivalently into a query posed on the raw-data view. Using this formal result, in the remainder of the dissertation we assume that all the queries in the workloads that we consider are posed on the relevant raw-data view. In this context, we consider the evaluation costs of answering unnested SQL select-project-join queries with grouping and aggregation using unindexed materialized views, such that each query can be evaluated using just one view and no other data. (This setting is the same as in [4, 19, 22, 32, 38].) 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 materialized view v if and only if $q \subseteq v$.

To evaluate a query using a given view (if this view can indeed be used to answer the query), we have to scan all rows of this view. Hence, the corresponding evaluation cost is equal to the size of the materialized view itself; a similar cost calculation is used in [4,19,22,32,38]. One way to estimate the view sizes in practice, as suggested in the literature, is by getting a relatively small-size sample of the raw-data view and by then evaluating the view definitions on that table, with a subsequent scaleup of the sizes of the resulting relations. We use a_i to denote the (estimated) size of each view v_i in the problem input. We also use the parameter d_{ij} to denote the evaluation cost of answering query q_j using view v_i. It follows that for each query q_j we have $d_{ij} = a_i$ if $q_j \subseteq v_i$, and we set $d_{ij} = +\infty$ otherwise, implying that q_j cannot be answered by the view v_i.

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 to 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 is 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, for each query set Q_2, we need a replacement plan, that is, at the end of stage 1 we keep some of the view relations that we materialized at stage 1, to be used again at stage 2 for Q_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. We refer to this space limit as the *replacement space limit*. 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 costs for the queries at stage 1, plus the expected total evaluation costs for the queries at stage 2. Note that in order to obtain a globally optimal solution for this problem, which we call the ***S***tochastic ***V***iew-***S***election *(SVS)* problem, all decisions regarding both stages (that is, the view set S_1 and the replacement plan

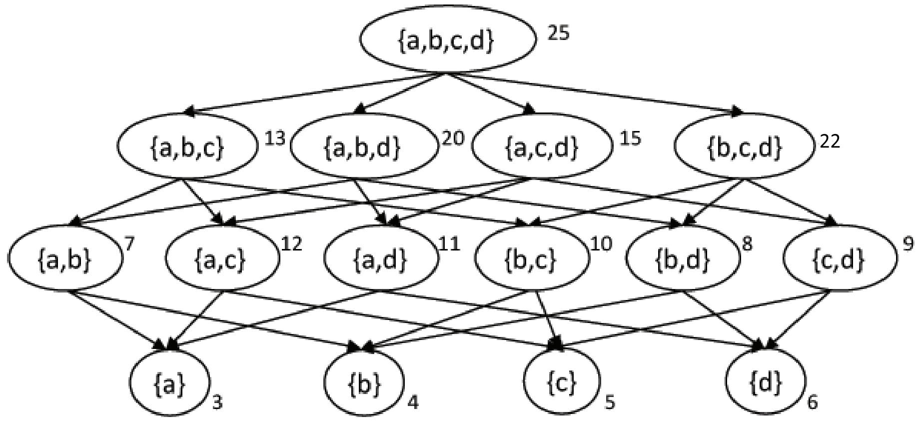

Figure 3.1: View lattice for Example 1, with view sizes shown as number of bytes.

for every possible realization Q_2 of $\mathbf{Q}_2$) must be made prior to stage 1.

The *search space of views* that we consider for a given problem *SVS* is the *view lattice* introduced by Harinarayan et al. in [19]. The view lattice includes all the views defined on the raw-data table, such that each view has aggregation on all the attributes aggregated in the input queries. In the view lattice, each node represents a view, and a directed edge from node v_1 to node v_2 implies that v_1 is a parent of v_2, that is, v_2 can be obtained from v_1 by aggregating over one attribute of v_1. To illustrate, we present the following numeric example for a data cube [19] with 4 attributes.

Example 1. Given a database with four grouping attributes a, b, c and d, the view lattice defined in [19] is shown in Figure 3.1. 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 three queries $q_1 = \{a, b\}$, $q_2 = \{b, c\}$, and $q_3 = \{c, d\}$. At stage 2, $q_4 = \{b\}$ and $q_5 = \{a, c\}$ would occur in one query set with probability 0.5, while $q_6 = \{c\}$ and $q_7 = \{b, d\}$ would occur in the other query set with probability 0.5. Equivalently, $Q_1 = \big\{\{a, b\}, \{b, c\}, \{c, d\}\big\}$, $Q_2^1 = \big\{\{b\}, \{a, c\}\big\}$, and $Q_2^2 = \big\{\{c\}, \{b, d\}\big\}$. Assume the total space limit is $b_1 = 30$, and the replacement space limit is $b_2 = 15$. 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. Furthermore, 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. $\square$

3.2 A two-stage stochastic programming model SP

In this subsection, we describe a stochastic programming model for the above two-stage stochastic view-selection problem (SVS). 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 dissertation we assume that $\mathbf{Q}_2$ is a random query set with L possible values. More specifically, we assume that $\mathbf{Q}_2$ equals query set Q_2^ℓ with probability p_ℓ, for $\ell = 1$ to L, with $\sum_{\ell=1}^{L} p_\ell = 1$. 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 each view $v_i \in V$ (where V is the set of all views) and for each query $q_j \in Q_1$.

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

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

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

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

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

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

The cardinality of the view set V is 2^K, where K is the number of distinct grouping attributes in the database. Let $I = \{1, 2, \ldots, 2^K\}$ be the set of subscripts for all the views $v_i \in V$. In addition, 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 [6] that we denote by SP.

$$(SP) \qquad \text{minimize} \qquad \sum_{j \in J_1} \sum_{i \in I} d_{ij} z_{ij} + \mathbf{E}_{\mathbf{Q_2}} \Psi(\mathbf{x}, \mathbf{Q}_2) \tag{3.1}$$

$$\text{subject to} \qquad \sum_{i \in I} z_{ij} = 1 \qquad \forall j \in J_1 \tag{3.2}$$

$$z_{ij} \le x_i \qquad \forall j \in J_1 \text{ and } \forall i \in I \tag{3.3}$$

$$\sum_{i \in I} a_i x_i \le b_1 \tag{3.4}$$

All the variables are binary

Here, $\mathbf{E}_{\mathbf{Q_2}}$ denotes the mathematical expectation of response time at stage 2 with respect to $\mathbf{Q}_2$. If the probability distribution of $\mathbf{Q}_2$ is as above, we have that

$$\mathbf{E}_{\mathbf{Q}_2}\Psi(\mathbf{x},\mathbf{Q_2}) = \sum_{\ell=1}^{L} p_\ell \Psi(\mathbf{x}, Q_2^\ell) \tag{3.5}$$

where $\Psi(\mathbf{x}, Q_2)$ is the minimum response time for a given set of values of the first-stage variables $\mathbf{x} = \left(x_1, \cdots, x_{|V|}\right)$ and for a realization of the second-stage queries Q_2. For each value of ℓ, the corresponding value of $\Psi(\mathbf{x}, Q_2^\ell)$ is obtained by solving the following view-selection problem.

$$\Psi(\mathbf{x}, Q_2^\ell) = \min \sum_{j\in J_2^\ell}\sum_{i\in I} d_{ij}t_{ij}^\ell \tag{3.6}$$

$$\text{subject to} \quad \sum_{i\in I} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell \tag{3.7}$$

$$t_{ij}^\ell \le u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell \ \text{ and } \ \forall i \in I \tag{3.8}$$

$$u_i^\ell \le x_i \qquad \forall i \in I \tag{3.9}$$

$$\sum_{i\in I} a_i y_i^\ell \le b_2 \tag{3.10}$$

$$\sum_{i\in I} a_i(u_i^\ell + y_i^\ell) \le b_1 \tag{3.11}$$

$$\textit{All the variables are binary} \tag{3.12}$$

Constraints (3.2) and (3.7) state that each query is answered by exactly one view in the set of materialized views. Constraints (3.3) and (3.8) guarantee that a query can be answered by a view only if the view is already materialized. Constraints (3.4), (3.10), and (3.11) state the storage limits on the view sets. Constraint (3.9) guarantees that the view kept from stage 1 to stage 2 is already materialized at stage 1.

In defining decision variables in the above model, we used the subscript i to represent view v_i, and in every case we 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 rewrite the above model with fewer 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

$$\begin{aligned} V_1 &= \left\{v_i \in V : v_i \supseteq q \text{ for some } q \in \widehat{Q}\right\} && (3.13)\\ V_2^\ell &= \left\{v_i \in V : v_i \supseteq q \text{ for some } q \in Q_2^\ell\right\}, \quad \ell = 1, \ldots, L && (3.14)\end{aligned}$$

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, for 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 $V_{2j}^\ell = \left\{v_i \in V_2^\ell : v_i \supseteq q_j\right\}$, 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 $\widehat{Q} = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, Q_2^1 and Q_2^2, respectively. (For Q_2^1 and Q_2^2, see the first part of this example in Section 3.1.) Thus, we obtain that

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

For each query, we define the relevant view sets in answering that particular query. For example, for the 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 the 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 the views in the set V_1, and the first-stage decision variables z_{ij} only for the 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 the views $v_i \in V_2^\ell$, and the second-stage decision variables t_{ij}^ℓ, only for the 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 which we denote by SP'.

$$
\begin{aligned}
(SP') \qquad & \text{minimize} \qquad && \sum_{j \in J_1} \sum_{i \in I_{1j}} d_{ij} z_{ij} + \mathbf{E}_{\mathbf{Q_2}} \Psi(\mathbf{x}, \mathbf{Q}_2) && (3.15) \\
& \text{subject to} && \sum_{i \in I_{1j}} z_{ij} = 1 \qquad \forall j \in J_1 && (3.16) \\
& && z_{ij} \leq x_i \qquad \forall j \in J_1 \text{ and } \forall i \in I_{1j} && (3.17) \\
& && \sum_{i \in I_1} a_i x_i \leq b_1 && (3.18) \\
& && \textit{All the variables are binary} && (3.19)
\end{aligned}
$$

The corresponding second-stage subproblem can be written as follows:

$$
\begin{aligned}
\Psi(\mathbf{x}, Q_2^\ell) = \min & \sum_{j \in J_2^\ell} \sum_{i \in I_{2j}^\ell} d_{ij} t_{ij}^\ell && (3.20) \\
\text{subject to} \qquad & \sum_{i \in I_{2j}^\ell} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell && (3.21) \\
& t_{ij}^\ell \leq u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell \text{ and } \forall i \in I_{2j}^\ell && (3.22) \\
& u_i^\ell \leq x_i \qquad \forall i \in I_2^\ell && (3.23) \\
& \sum_{i \in I_2^\ell} a_i y_i^\ell \leq b_2 && (3.24) \\
& \sum_{i \in I_2^\ell} a_i (\ u_i^\ell + y_i^\ell) \leq b_1 && (3.25) \\
& \textit{All the variables are binary} && (3.26)
\end{aligned}
$$

Chapter 4

Solving the stochastic programming model SP

We begin this chapter by introducing an integer programming model [37] obtained from the extensive form [6] of the stochastic programming model that we introduced in Section 3.2. Later in this chapter, we will study the properties of this integer programming model, use these properties to remove some variables and constraints, and obtain a model that in many cases tends to be significantly smaller. Optimal solutions of the resulting model are guaranteed to be optimal for the original model SP. This, in turn, allows us to solve larger instances of the problem SVS.

4.1 An integer programming model

In this subsection we propose an integer programming model [37] for the problem SVS. In order to solve the stochastic view-selection model SP, we write the extensive form [6] of the stochastic programming model SP' that we discussed in Section 3.2, which is an integer programming model that we denote by $IP1$.

$$(IP1) \quad \text{minimize} \quad \sum_{j \in J_1} \sum_{i \in I_{1j}} d_{ij} z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j \in J_2^\ell} \sum_{i \in I_{2j}^\ell} d_{ij} t_{ij}^\ell \tag{4.1}$$

$$\text{subject to} \quad \sum_{i \in I_{1j}} z_{ij} = 1 \qquad \forall j \in J_1 \tag{4.2}$$

$$z_{ij} \leq x_i \qquad \forall j \in J_1, \forall i \in I_{1j} \tag{4.3}$$

$$\sum_{i \in I_1} a_i x_i \leq b_1 \tag{4.4}$$

$$\sum_{i \in I_{2j}^\ell} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell, \quad \ell = 1, \ldots, L \tag{4.5}$$

$$t_{ij}^\ell \leq u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in I_{2j}^\ell, \quad \ell = 1, \ldots, L \tag{4.6}$$

$$u_i^\ell \leq x_i \qquad \forall i \in I_2^\ell, \quad \ell = 1, \ldots, L \tag{4.7}$$

$$\sum_{i \in I_2^\ell} a_i y_i^\ell \leq b_2, \quad \ell = 1, \ldots, L \tag{4.8}$$

$$\sum_{i \in I_2^\ell} a_i (u_i^\ell + y_i^\ell) \leq b_1, \quad \ell = 1, \ldots, L \tag{4.9}$$

$$\textit{All the variables are binary} \tag{4.10}$$

Note that for an instance with K attributes, with n_1 queries in Q_1 at stage 1 and with 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 3.2. Thus, even for relatively small values of K, L, n_1 and $n_2^1, \ldots, n_2^L$, 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 [21]. The applicability of our model is thus limited to smaller-size instances. We thus endeavor to reduce the size of our model by reducing the size of the view sets V_1, V_{1j}, V_2^ℓ and V_{2j}^ℓ, for $\ell = 1, \ldots, L$. In Section 4.4, we will provide numerical results for solving the model $IP1$.

4.2 Reduction of the search space of views

In this subsection, we make a few observations regarding the properties of the views that appear in an optimal solution for a given problem SVS. Based on these observations, we identify the reduced search spaces of views, which are 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 the model $IP1$, and obtain an integer programming model that we denote by $IP2$. $IP2$ is significantly smaller than $IP1$, and optimal solutions of $IP2$ are guaranteed to be optimal for the model SP.

4.2.1 First reduction

As introduced in Section 2.2, in a one-stage view-selection problem, Asgharzadeh Talebi et al. [3, 4] propose to reduce the search space of views (i.e., the set of views to consider) by the results of Observation 1 in Section 2.2. In other words, they eliminate from the search space each view v that has at least one attribute that is not in any of the queries answerable by v. We extend this pruning method to our problem SVS as follows.

Given a view v and a set of queries Q , let $Q(v)$ denote the set of queries in Q that v can answer, i.e., $Q(v) = \{q \in Q : q \subseteq v\}$. We make the following observations.

Observation 3. *In an instance of problem SVS, given a view v in the view set V_1, if the number of attributes of v is strictly greater than the number of attributes in the union set of the queries in $\widehat{Q}$ that v could answer, that is, $|v| > |\cup_{q\in\widehat{Q}(v)} q|$, then there exists an optimal solution in which v is not materialized at stage one, or equivalently, the corresponding decision variable has the value $x_i = 0$.*

Proof. If q could be answered by v, then the set of attributes for q is a subset of that for v, i.e., $q \subseteq v$. Thus, $\cup_{q\in\widehat{Q}(v)} q \subseteq v$ and $| \cup_{q\in\widehat{Q}(v)} q| \leq |v|$. If $| \cup_{q\in\widehat{Q}(v)} q| < |v|$, then v has at least one attribute, call it attribute a, that is not in any $q \in \widehat{Q}$ such that $q \subseteq v$. If there exists an optimal solution in which v is materialized at stage one, we replace it with view $v' = v \setminus \{a\}$. Since the size of v' is no more than that of v, the replacement will not violate the space limit. For each query $q \in \widehat{Q}$ that is answered by v in the optimal solution, q could be answered by v' with the cost that does not exceed that of v. Thus, we obtain an optimal solution in which v is not materialized at stage one, and the corresponding decision variable has the value $x_i = 0$. □

Observation 4. *In an instance of problem SVS, given a view v in the view set V_2^ℓ, if the number of attributes of v is strictly greater than the number of attributes in the union set of the queries in Q_2^ℓ that v could answer, that is, $|v| > | \cup_{q\in Q_2^\ell(v)} q|$, then there exists an optimal solution in which v is not materialized at stage two, or equivalently, the corresponding decision variable has the value $y_i^\ell = 0$.*

Proof. If v satisfies the above conditions, then v has at least one attribute, say attribute a, that is not in any query in Q_2^ℓ that can be answered by v. If v is materialized at stage two in an optimal solution, then replace v by $v \setminus \{a\}$, and the solution remains optimal. □

4.2.2 The second reduction

In the one-stage view-selection problem, Asgharzadeh Talebi et al. [3, 4] also provide a method for reducing 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, as introduced in Observation 2 in Section 2.2. We extend this pruning method to our problem *SVS*. We begin by introducing the following definitions. Recall that for each view v, $Q(v) = \{q \in Q : q \subseteq v\}$.

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

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

$$d(v, Q') = \sum_{q \in Q'} S(q) - S(v) \tag{4.11}$$

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

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

Based on Definitions 1 and 2, we can extend the pruning method introduced in [3, 4] to our problem *SVS*, and obtain the following observations.

Observation 5. *In an instance of problem SVS, given a view v in the view set V_1, if v is not a query in $\widehat{Q}$, and the maximum benefit of v with respect to $\widehat{Q}$ is non-positive, that is, $d(v, \widehat{Q}(v)) = \sum_{q \in \widehat{Q}(v)} S(q) - S(v) \leq 0$, then there exists an optimal solution in which v is not materialized at stage one, or equivalently, the corresponding decision variable has the value $x_i = 0$.*

Proof. Suppose we have a solution in which view $v \notin \widehat{Q}$, v is materialized at stage one, and $d(v, \widehat{Q}(v)) \leq 0$. Remove view v and replace it by the collection of views corresponding to the set of queries in $\widehat{Q}$ that v can answer. Let all the queries that are assigned to v be assigned to their corresponding views in this collection. It follows that the new solution remains feasible and satisfies the storage-space constraints, and that its overall cost is no larger than that of the previous solution. The result follows. □

Observation 6. *In an instance of problem SVS, given a view v in the view set V_2^ℓ, if $v \notin Q_2^\ell$ and the maximum benefit of v with respect to Q_2^ℓ is non-positive, that is, $d(v, Q_2^\ell(v)) =$*

$\sum_{q \in Q_2^\ell(v)} S(q) - S(v) \leq 0$, *then there exists an optimal solution in which v is not materialized at stage two, or equivalently, the corresponding decision variable has the value $y_i^\ell = 0$.*

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

4.2.3 The third reduction

In the first and second reductions that were introduced in Sections 4.2.1 and 4.2.2, we reduced the search spaces of views based on the relationship between each view and the corresponding target queries that it can answer. In this subsection, however, we reduce the search spaces of views not only by looking at the relationship between each view and the input queries, but also by considering some relationships between the views themselves in the data lattice.

We make the following observations based on the relationship between the maximum benefits, as given by Definition 2, of the views in the view lattice. The main idea is to find a set of views that could replace a given view without affecting the optimality of the solution.

Observation 7. *In an instance of problem SVS, given a view v in the view set V_1, if there exists a view v' in V_1 in which $v' \subset v$ and the maximum benefit of v' with respect to $\widehat{Q}$ is greater than or equal to that of v with respect to $\widehat{Q}$, that is, if $d(v', \widehat{Q}(v')) \geq d(v, \widehat{Q}(v))$, then there exists an optimal solution in which v is not materialized at stage one, or equivalently, the corresponding decision variable has the value $x_i = 0$.*

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

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

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

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

If v is materialized at stage one in an optimal solution, we can replace it by materializing the view v' and the view set V', which is the set of views corresponding to the queries in

$\widehat{Q}(v) \setminus \widehat{Q}(v')$. The size of the view v is greater than or equal to the total size of new views. Thus, the space limit is not violated. All the queries that are assigned to v can be answered by v' or by some view in V'. The overall cost of the new solution is no higher than the previous one. Thus, the new solution remains optimal. The result follows. □

Observation 8. *In an instance of problem SVS, given a view v in the view set V_2^ℓ, if there exists a view v' in V_2^ℓ in which $v' \subset v$ and the maximum benefit of v' with respect to Q_2^ℓ is greater than or equal to that of v with respect to Q_2^ℓ, that is, if $d(v', Q_2^\ell(v')) \geq d(v, Q_2^\ell(v))$, then there exists an optimal solution in which v is not materialized at stage two, or equivalently, the corresponding decision variable has the value $y_i^\ell = 0$.*

Proof. If v is materialized at stage two in an optimal solution, we can replace it by materializing the view v' and the view set V', which is the set of views corresponding to the queries in Q_2^ℓ that can be answered by v but not by v'. Similarly to the proof of Observation 7, we can prove that the total size of views that replace view v is no more than the size of view v. Thus, the space limit is not violated. All the queries in Q_2^ℓ that are assigned to v in the optimal solution can be answered by v' or by some view in V' with the overall cost that is no higher than before. Thus, the new solution remains optimal. The result follows. □

4.2.4 A reduction algorithm and reduced search spaces of views

We reduce the search space of views, and thus the size of the corresponding IP model, by removing certain views based on Observations 3-8. First, we can conduct the reductions on the view set V_1 by removing every view that satisfies the condition stated in at least one of Observations 3, 5, or 7. We refer to every such view as a *dominated view*. Note that the conditions stated in these observations are independent of each other, hence the results of these reductions do not depend on the order of their application. Thus, as we go through the collection of views in the set V_1 in order to identify the dominated views, for each view we test for the conditions stated in all three observations before moving to the next view. Similarly, we reduce the search space of views V_2^ℓ by identifying and removing each dominated view that satisfies the stated condition in at least one of Observations 4, 6, or 8. A detailed description of the *reduction procedure* that we have devised to implement these reductions is given below. The computational complexity of this procedure is $O(|V|^2) = O(4^K)$, where K is the number of grouping attributes in the database.

We use V' to denote one of the view sets from $\{V_1, V_2^1, \ldots, V_2^L\}$. Let Q' be the corresponding relevant query set, and let $\overline{V'}$ denote the reduced view set of V'. We use the binary representation of the subscript number for each view to indicate the set of attributes for the view. For instance, in a database with 4 attributes a, b, c and d, the binary number "1001" represents the set of attributes $\{a, d\}$. Hence, the subscript number of the view $\{a, d\}$ is 9 ($= 2^3 + 2^0$). We can now conduct the reductions on the views by iterating through all the subscript numbers of the corresponding views in V'. The pseudocode for this procedure is given in the next page.

Algorithm 1 The reduction procedure

Input: query set Q' and the search space of views V'
Output: a reduced search space of views $\overline{V'}$

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

In this procedure, we iterate through all the views in the view set V' in the increasing order of their subscript numbers, one view in each iteration (lines 1-4). Note that if a view v answers a query (view) v', the subscript number of v is no smaller than the subscript number of v'. Thus, this ordering guarantees that when conducting the reductions regarding a view v, all the views that are subsets of v have already been considered in this procedure. The last view that we consider in this procedure is the raw-data view, which we introduced in Section 2.2. In each iteration, we use a binary variable f to indicate whether the view v can be eliminated. At the beginning of each iteration, we set $f = 1$ (line 5). In each iteration, we first conduct the first and second reductions based on Observations 3-6 (lines 6-9). More specifically, for each view v, we compare the number of attributes in v with that in the union set of the queries in Q' that v can answer. We also compare the size of view v with the total size of the queries in Q' that v can answer. Note that the statements "$|v| \neq a$" and "$v \notin Q'$ **and** $d(v, Q'(v)) \leq 0$" in line 8 indicate that v can be eliminated by the results of Observation 3 (or 4) and by the results of Observation 5 (or 6), respectively. Then we conduct the third reduction based on Observation 7 (or 8) by iterating through each view $\tilde{v}$ that has been selected as a candidate view and can be answered by v (lines 11-19). For each view $\tilde{v}$, we compare the maximum benefit of v with respect to Q' ($d(v, Q'(v))$) with that of $\tilde{v}$ with respect to Q' ($d(\tilde{v}, Q'(\tilde{v}))$). The statement "$d(\tilde{v}, Q'(\tilde{v})) \geq d(v, Q'(v))$" indicates that the view v can be eliminated. At the end of each iteration, if the view is not eliminated by any of the results of Observations 3-8, we add it into the view set $\overline{V'}$ (lines 21-23). At the beginning of the procedure, the view set $\overline{V}'$ is an empty set. As a result, at the end of this procedure, $\overline{V'}$ is the new search space of views.

We denote the reduced view sets of V_1 and V_2^ℓ by $\overline{V_1}$ 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 9. *For each view v that is materialized at stage one and kept from stage one to stage two for the ℓ^{th} scenario, $v \in \overline{V_1} \cap V_2^\ell$.*

The proof of Observation 9 is straightforward.

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. Again, we denote the associated sets of subscripts for $\overline{V_{1j}}$, $\overline{V_{12j}}$ and $\overline{V_{2j}^\ell}$ by $\overline{I_{1j}}$, $\overline{I_{12j}}$ and $\overline{I_{2j}^\ell}$, for $\ell = 1$ to L, respectively.

Example 1 (Continued). The view $\{b, c\}$ in the set V_2^1 answers only one query $\{b\}$ in Q_2^1, and contains one more attribute than $\{b\}$. As a result, the view $\{b, c\}$ is eliminated from V_2^1 by the first reduction for $\overline{V_2^1}$. The view $\{a, b, d\}$ in the set V_1 answers the queries $\{b\}$, $\{a, b\}$ and $\{b, d\}$ in $\widehat{Q}$, and the total size of the three queries is less than the size of the view $\{a, b, d\}$ ($4 + 7 + 8 = 19 < 20$). As a result, the view $\{a, b, d\}$ is eliminated from V_1 by the second reduction. Comparing the views $\{a, c\}$ and $\{a, c, d\}$ in V_1, we have that $d(\{a, c\}, \widehat{Q}) = 5 + 12 - 12 = 5$ and $d(\{a, c, d\}, \widehat{Q}) = 5 + 9 + 12 - 22 = 4$. Observe that $\{a, c\}$ can be answered by $\{a, c, d\}$ and $d(\{a, c\}, \widehat{Q}) > d(\{a, c, d\}, \widehat{Q})$. Thus, the view $\{a, c, d\}$ is eliminated from V_1 by the third reduction. Using the same approaches, we obtain the reduced view sets as follows.

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

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

In Section 4.4, we provide numerical results and discuss the efficiency of the reductions on the search spaces for a number of instances.

4.3 Modified integer programming model $IP2$

We can now redefine the first-stage decision variables x_i only for the views in the set $\overline{V}_1$, and the first-stage decision variables z_{ij} only for the views $v_i \in \overline{V_{1j}}$. Similarly, we can redefine the second-stage decision variables u_i^ℓ only for the views $v_i \in \overline{V_{12}^\ell}$, and the second-stage decision variables y_i^ℓ only for the views $v_i \in \overline{V_2^\ell}$. We can also redefine the second-stage variables t_{ij}^ℓ only for the views $v_i \in \overline{V_{12j}^\ell} \cup \overline{V_{12}^\ell}$, for $\ell = 1$ to L. We can now rewrite the integer programming model $IP1$ into an integer programming model with reduced size, which we denote as $IP2$.

$$
\begin{aligned}
(IP2) \quad \text{minimize} \quad & \sum_{j \in J_1} \sum_{i \in \overline{I_{1j}}} d_{ij} z_{ij} + \sum_{\ell=1}^{L} p_\ell \sum_{j \in J_2^\ell} \sum_{i \in \overline{I_{12j}^\ell} \cup \overline{I_{12}^\ell}} d_{ij} t_{ij}^\ell && (4.12)\\
\text{subject to} \quad & \sum_{i \in \overline{I_{1j}}} z_{ij} = 1 \qquad \forall j \in J_1 && (4.13)\\
& z_{ij} \le x_i \qquad \forall j \in J_1, \forall i \in \overline{I_{1j}} && (4.14)\\
& \sum_{i \in \overline{I_1}} a_i x_i \le b_1 && (4.15)\\
& \sum_{i \in \overline{I_{12j}^\ell} \cup \overline{I_{12}^\ell}} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell, \quad \ell = 1, \ldots, L && (4.16)\\
& t_{ij}^\ell \le u_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in \overline{I_{12j}^\ell} - \overline{I_{2j}^\ell}, \quad \ell = 1, \ldots, L && (4.17)\\
& t_{ij}^\ell \le y_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in \overline{I_{2j}^l} - \overline{I_{12j}^\ell}, \quad \ell = 1, \ldots, L && (4.18)\\
& t_{ij}^\ell \le u_i^\ell + y_i^\ell \qquad \forall j \in J_2^\ell, \forall i \in \overline{I_{12j}^l} \cap \overline{I_{2j}^l}, \ \ell = 1, \ldots, L && (4.19)\\
& u_i^\ell \le x_i \qquad \forall i \in \overline{I_{12}^\ell}, \quad \ell = 1, \ldots, L && (4.20)\\
& \sum_{i \in \overline{I_2^\ell}} a_i y_i^\ell \le b_2, \quad \ell = 1, \ldots, L && (4.21)\\
& \sum_{i \in \overline{I_{12}^\ell}} a_i u_i^\ell + \sum_{i \in \overline{I_2^\ell}} a_i y_i^\ell \le b_1, \quad \ell = 1, \ldots, L && (4.22)\\
& \textit{All the variables are binary} && (4.23)
\end{aligned}
$$

Any optimal solution of $IP2$ is also an optimal solution of $IP1$. Thus, given an instance of the problem SVS, 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 $(n_1 + \sum_{\ell=1}^{L} n_2^\ell + 2L + 1)|\overline{V_1}|$ on the number of variables, and an upper bound $(n_1 + \sum_{\ell=1}^{L} n_2^\ell + 2L)(|\overline{V_1}| + 1) + 1$ on the number of constraints in $IP2$. Compared with $IP1$, both numbers are, in general, dramatically smaller, and thus the size of $IP2$ can be significantly smaller than that of $IP1$. In Section 4.4, we provide numerical results to compare $IP2$ with $IP1$.

4.4 Experimental results of solving the problem SVS

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 4.2, and (ii) the scalability of the model $IP2$. We construct a collection of instances of the problem SVS with varying sizes, using a number of databases generated via the TPC-H benchmark [36].

We then solve each instance using the 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 instances of realistic size of the problem SVS. We also observe that some realistic-size instances could not be solved due to insufficient memory. All of our algorithms have been implemented in C++, and all the experiments were carried out on a 2.66GHz Intel 2 Quad processor with 3.25 GB RAM, running Windows XP Professional. We used CPLEX 11 [21] to solve the integer programming models $IP1$ and $IP2$.

In Section 4.4.1 we describe how we construct each of the instances in our experiments. In Sections 4.4.2 and 4.4.3 we evaluate the effectiveness of our models for solving the problem SVS, and report our findings. Finally, in Section 4.4.4 we present a summary of our experimental results.

4.4.1 Constructing the instances

The input parameters for an instance of the problem SVS are the given database $\mathcal{D}$, the query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, \ldots, Q_2^L\}$, the 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 generated via the TPC-H benchmark [36]. 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 replacing the original stored TPC-H tables generated with scale factor one, by 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 $p_1 = p_2 = 0.5$.

We define the *total size* of a query set $(\cdot)$ as the total size of all the queries in $(\cdot)$ (i.e., the total size of the answers to all the queries in $(\cdot)$). In order to construct the query sets Q_1, Q_2^1 and Q_2^2 for each instance, we use the following principles. In each query set, each query is generated randomly on the given database. In addition, the total size of each query set is "approximately" the same. More specifically, the difference between the total 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 on the 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 on the database $\mathcal{D}$ as the query set Q_1. In order to obtain each of the query sets Q_2^1 and Q_2^2, we randomly generate queries on the database $\mathcal{D}$, until the total size of all the generated queries reaches or exceeds $Size(Q_1)$, but doesn't exceeds $Size(Q_1)+S(v_1)$. Here, $Size(\cdot)$ denotes the total size of a query set $(\cdot)$, v_1 denotes the raw-data view, and $S(\cdot)$ denotes the size of a query or a view $(\cdot)$. Thus, we obtain that

$$Size(Q_1) \leq Size(Q_2^\ell) \leq Size(Q_1) + S(v_1), \qquad \ell = 1, 2 \tag{4.24}$$

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

$$b_1 = \alpha \cdot Size(\widehat{Q}) \tag{4.25}$$

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

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

If $\alpha \geq 1$, then the problem *SVS* becomes trivial, since the best solution could be to materialize 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 problem *SVS* to be nontrivial, we need to have $0 < \alpha < 1$.

We require that $b_2 \leq b_1$ always holds on the storage-space limits b_1 and b_2. Suppose b_2 is expressed as

$$b_2 = \beta b_1 \tag{4.26}$$

where $0 \leq \beta \leq 1$. If $b_2 \geq \max_{Q \in \{Q_2^1, \ldots, Q_2^L\}} Size(Q)$, then the second-stage sub-problem of the problem *SVS* 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 chapter, for all the 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

Table 4.1: Comparison of the search spaces of views in the models $IP1$ and $IP2$, for instances on the 7-attribute TPC-H database

inst-ance	query sets $(\lvert Q_1\rvert, \lvert Q_2^1\rvert, \lvert Q_2^2\rvert)$	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 $\lvert V_1\rvert$	IP2 $\lvert\overline{V_1}\rvert$	IP1 $\lvert V_2^1\rvert$	IP2 $\lvert\overline{V_{12}^1}\rvert$	IP1 $\lvert V_2^1\rvert$	IP2 $\lvert\overline{V_2^1}\rvert$	IP1 $\lvert V_2^2\rvert$	IP2 $\lvert\overline{V_{12}^2}\rvert$	IP1 $\lvert V_2^2\rvert$	IP2 $\lvert\overline{V_2^2}\rvert$
1	(20,20,20)	126	78	122	74	122	40	116	72	116	46
2	(20,25,20)	125	84	122	82	122	47	110	75	110	38
3	(30,35,29)	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

queries in $\widehat{Q}$, and b_2 is one-half of b_1. It follows that

$$\begin{aligned}
b_1 &= 0.2Size(\widehat{Q}) < Size(\widehat{Q}); \\
b_2 &= 0.5b_1 = (0.5)(0.2)Size(\widehat{Q}) \\
&\leq (0.1)(Size(Q_1) + Size(Q_2^1) + Size(Q_2^2)) \\
&\leq (0.1)(3)\max\{Size(Q_2^1), Size(Q_2^2)\} \\
&< \max\{Size(Q_2^1), Size(Q_2^2)\},
\end{aligned}$$

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

4.4.2 Reducing the search space

In this subsection, we compare the sizes of the models $IP1$ and $IP2$ for several instances of the problem SVS. More specifically, we have 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. We present these instances in Appendix A. For each instance, we compare the number of the decision variables x_i, u_i^ℓ, and y_i^ℓ, for models $IP1$ and $IP2$, as shown in Tables 4.1 and 4.3. 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 the time that it takes to solve the model by the CPLEX IP solver, for models $IP1$ and $IP2$. The results are shown in Tables 4.2 and 4.4.

We observe that the number of views in the search space for the model $IP1$ is significantly reduced in $IP2$ for all the instances. It is also observed that the size of $IP2$, when expressed

Table 4.2: Comparison of the sizes and computing times in the models $IP1$ and $IP2$, for instances on 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	(20,20,20)	1986	1233	1893	1194	0.062	0.000	0.625	0.375
2	(20,25,20)	1937	1315	1838	1288	0.015	0.000	1.500	1.235
3	(30,35,29)	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 4.3: Comparison of the search spaces of views in the models $IP1$ and $IP2$, for instances on 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	(20,21,26)	7974	656	5292	641	5292	217	7924	653	7924	381
2	(20,21,23)	7381	587	6144	573	6144	83	6139	570	6139	252
3	(30,26,22)	8168	857	7952	849	7952	350	7254	836	7254	115
4	(30,36,33)	8150	1076	7840	1069	7840	422	7568	1067	7568	279
5	(40,47,41)	8144	2475	7905	2469	7905	1311	7685	2456	7685	965
6	(40,48,46)	8176	2369	7912	2345	7912	1281	7726	2332	7726	807
7	(50,53,56)	8132	2514	7484	2477	7484	733	7880	2506	7880	1048
8	(50,52,56)	8162	2932	7750	2885	7750	1000	7925	2922	7925	1947

Table 4.4: Comparison of the sizes and computing times in the models $IP1$ and $IP2$, for instances on the 13-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	(20,21,26)	84,862	11,026	63,744	10,540	0.422	0.328	20.360	1.750
2	(20,21,23)	62,705	7,225	43,110	7,180	0.344	0.313	22.485	6.875
3	(30,26,22)	124,496	18,405	101,205	18,386	0.547	0.391	284.410	8.930
4	(30,36,33)	103,044	18,858	79,590	18,620	0.563	0.438	354.004	34.094
5	(40,47,41)	134,562	53,979	110,961	52,010	0.734	0.719	134.830	19.891
6	(40,48,46)	134,156	51,231	110,481	49,502	0.735	0.735	656.508	31.875
7	(50,53,56)	117,512	47,709	94,180	46,780	0.766	0.797	181.315	36.329
8	(50,52,56)	152,722	71,438	129,048	68,582	0.875	0.891	1084.764	189.890

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 build times for the models $IP1$ and model $IP2$, the time to solve $IP2$ is significantly smaller than that for $IP1$, especially for relatively large problems, such as instances based on the 13-attribute database.

We observe that the reduction in the number of decision variables in each group, and the reduction in the size of the model, as expressed by the total number of variables and constraints, is more significant for instances with a small ratio of the number of queries over the total number of views in the database. More specifically, we compare the results for instances on the 7-attribute database in Tables 4.1 and 4.2. When we increase the number of queries in each query set in the instances, the ratio of the number of queries over the total number of views increases, and the magnitude of the associated reductions in the number of variables decreases. We also compare the results for the instances with similar numbers of queries based on different databases. More specifically, we compare each pair of instances with the same number of queries in Q_1 (same instance ID) on the 7-attribute database and on the 13-attribute database. Note that the number of views in the 7-attribute database is 128, and the number of views in the 13-attribute database is 8192, respectively. For each pair of instances, the instance on the 13-attribute database has a relatively smaller ratio of the number of queries on the total number of views, and the associated reductions are relatively large.

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

Table 4.5: Scalability of $IP2$ for instances on 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	(80,83,75)	1.328	153.907
3	(100,98,95)	1.547	198.531
4	(120,122,124)	1.797	384.688
5	(140,140,139)	2.125	451.078
6	(160,161,153)	2.485	$>$ 20min
7	(180,191,192)	2.953	$>$ 20min
8	(200,202,215)	3.203	$>$ 20min

Table 4.6: Time of solving $IP2$ for instances on 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	(20,10,15)	41.282	2.062
2	(40,48,45)	46.203	87.765
3	(60,61,65)	54.167	206.826
4	(80,86,81)	62.719	out of memory
5	(100,96,98)	83.360	out of memory

4.4.3 Scalability of the model $IP2$

In this subsection we evaluate the scalability of the model $IP2$ in solving instances of the problem *SVS*. Based on the results in Tables 4.2 and 4.4, we could solve all the instances on the 7-attribute TPC-H database using $IP2$ within 10 seconds, and all the instances on the 13-attribute TPC-H database within 80 seconds. In order to examine the scalability of the 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 100 based on the 17-attribute TPC-H database. We solved all the instances using the model $IP2$, and report here the times required to build and solve the model for the corresponding $IP2$ based on each instance. The results are shown in Tables 4.5 and 4.6. Note that for these instances we do not construct the corresponding model $IP1$, because it is excessively large and hence cannot be solved with an IP solver.

From Table 4.5, we observe that we could solve within 10 minutes all those instances based on the 13-attribute database, where the number of queries in Q_1 is no more than 140. However, when we increase the size of the query set, the solver fails to provide an optimal solution within our time limit of 20 minutes. Moreover, as can be seen in Table 4.6, CPLEX fails to give an optimal solution for the instances based on the 17-attribute database in cases where the size of the query set grows to 80. The failure is due to memory shortage.

Comparing "the time to build the model" with "the time to solve the model" in Tables 4.2, 4.4, and 4.5, 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 sizes of query sets and larger execution times, this difference is quite significant.

4.4.4 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 use the model $IP2$ to optimally solve instances of the problem SVS in realistic size, while $IP1$ may not be 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 $IP2$. In Chapter 9, we develop approaches to further prune the search space of views, in order to solve larger instances of the two-stage view-selection problem.

Chapter 5

Value of the two-stage stochastic model

Given an instance of the problem *SVS*, we can solve it by obtaining an optimal solution of the stochastic programming model *SP* that was introduced in Section 3.2. In this chapter we assess the value of the two-stage stochastic programming model *SP* by:

- measuring the gain in solution quality obtained by using the two-stage model versus the one-stage model;
- measuring the benefits resulting from taking into account the stochastic properties in the model *SP*; and by
- discussing the value of obtaining further information about future events (i.e., about second-stage queries).

In Section 5.1, in the context of the problem *SVS* we compare our two-stage model *SP* with a corresponding one-stage model, which does not allow for replacement of views in stage 2. We define the *value of two-stage versus one-stage* (*VTVO*), as the difference between the optimal values of the two-stage model and the corresponding one-stage model. *VTVO* measures the gain obtained via the view-replacement mechanism of the two-stage model *SP*.

In Section 5.2 we compare our model *SP* with a two-stage model that is based on the expected value of random events. We introduce the *value of stochastic solution* (*VSS*) [6], which evaluates the difference between the expected response times achieved by (i) solving the model *SP*, and by (ii) solving the *expected-value* model. *VSS* assesses the benefits obtained by explicitly considering the stochastic characteristics of the problem in constructing the model.

In Section 5.3 we show that our model *SP* is also beneficial in making decisions on whether or not to obtain further information about the occurrence of second-stage queries. To do so, we define the *expected value of perfect information* ($EVPI$) [6] in the context of the problem SVS, and propose appropriate models to obtain its value.

Finally, in Section 5.4 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* ($VTVO$) for solving the problem SVS. Recall that in our two-stage stochastic view-selection model, we allow for view replacement at stage 2. The $VTVO$ measures the gain obtained via this replacement mechanism. In other words, for a given instance of the problem SVS, the value of $VTVO$ provides the magnitude of improvement in the optimal query-evaluation costs, as achieved by allowing for partial replacement of views at stage 2.

We begin by discussing the one-stage model, in which we do not allow to drop or replace any materialized views at stage 2. In this model, we must determine a set of views S to materialize at stage 1 within a given space limit b_1. We use the views in S to answer the queries in Q_1, as well as all the queries in the query set Q_2^ℓ that may occur at stage 2. 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 3.2. It follows that we could formulate the one-stage problem as an integer programming model that we denote by OS, as follows.

$$
\begin{array}{llll}
(OS) & \text{minimize} & \sum\limits_{j\in J_1}\sum\limits_{i\in I_{1j}} d_{ij}z_{ij} + \sum\limits_{\ell=1}^{L} p_\ell \sum\limits_{j\in J_2^\ell}\sum\limits_{i\in I_{2j}^\ell} d_{ij}t_{ij}^\ell & \\
 & \text{subject to} & \sum\limits_{i\in I_{1j}} z_{ij} = 1 \qquad \forall j\in J_1 & \\
 & & z_{ij} \le x_i \qquad \forall j\in J_1,\quad \forall i\in I_{1j} & \\
 & & \sum\limits_{i\in I_{2j}^\ell} t_{ij}^\ell = 1 \qquad \forall j\in J_2^\ell,\quad \ell=1,\ldots,L & (5.1)\\
 & & t_{ij}^\ell \le x_i \qquad \forall j\in J_2^\ell,\quad \forall i\in I_{2j}^\ell,\quad \ell=1,\ldots,L & \\
 & & \sum\limits_{i\in I_1} a_i x_i \le b_1 & \\
 & & \textit{All the variables are binary} &
\end{array}
$$

We note that if we set $b_2 = 0$ in model SP, then the model is equivalent to the model OS. It also turns out that the model OS is similar to the integer programming model $OVIP1$ for solving the one-stage problem OVS, as proposed in [4] and discussed in Section 2.2.

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

Observation 10. *The optimal value of OS is an upper bound on the optimal value of the model SP, that is, $Optv(SP) \le Optv(OS)$.*

Proof. In the 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 the model $IP1$, constraints (4.7)-(4.9) are redundant, and constraint (4.6) becomes

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

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 the model SP, that is, $Optv(SP) \le Optv(OS)$. □

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,

$$VTVO = Optv(OS) - Optv(SP). \tag{5.2}$$

Example 1 (Continued). For the example introduced in Section 3.1, the optimal values of the models SP and OS are 34 and 40.5, respectively. As a result, we obtain that $VTVO$ is 6.5 ($= 40.5 - 34$). □

5.2 The value of stochastic solution (VSS)

In this subsection, we introduce the value of stochastic solution for the problem SVS.

The literature on stochastic programming (e.g., [6]) defines the *value of stochastic solution* (VSS) as the gain obtained from solving the stochastic programming model, as opposed to a model based on the expected values. In our context, VSS is the difference between the optimal value of the stochastic programming model (our model SP) and the corresponding *Expected Value* (EV) model. The EV model is typically defined for the same two-stage environment, under the assumption that there is only one second-stage scenario that represents the *expected value* of the random events. More specifically, for the problem SVS, we consider the model in which we make 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. Then we make second-stage decisions given the actual query set occurring at stage 2.

In order to examine the VSS from the perspective of the problem SVS, we first consider and define the *expected-value problem (EV problem)*. We adapt the term "EV problem" from [6]. (The definition in [6] is not applicable to our 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 $Q_2 = \cup_{\ell=1}^{L} Q_2^\ell$. We define the *weight* w_j associated with each query $q \in Q_2$ as the sum of the probability values of the query sets from $\{Q_2^1, Q_2^2, \ldots, Q_2^\ell\}$ that contain q, or equivalently,

$$w_j = \sum_{\ell : q_j \in Q_2^\ell} p_\ell. \tag{5.3}$$

The EV 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 the 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 the 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. It turns out that the EV problem could be considered as a special case of the problem SVS, in which there is only one scenario at stage 2. We now construct a mathematical model for this problem, which we refer to as model $IPEV$.

Let J_2 be the set of subscripts for all the 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 3.2. Correspondingly, we denote 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 denote as I_{2j} the set of subscripts associated with V_{2j}.

We can now define the following second-stage decision variables for each view $v_i \in V_2$.

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

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

In addition, we define the second-stage decision variables t_{ij} for each query $q_j \in Q_2$ and for each view $v_i \in V_{2j}$.

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

Let the first-stage decision variables x_i and z_{ij} be as defined in Section 3.2. It follows that the EV problem could be formulated as an integer programming model, which we denote by $IPEV$, as follows.

$$
\begin{array}{llll}
(IPEV) & \text{minimize} & \sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}z_{ij} + \sum_{j\in J_2}\sum_{i\in I_{2j}} w_j d_{ij} t_{ij} & \\
 & \text{subject to} & \sum_{i\in I_{1j}} z_{ij} = 1 & \forall j \in J_1 \\
 & & z_{ij} \le x_i & \forall j \in J_1, \quad \forall i \in I_{1j} \\
 & & \sum_{i\in I_1} a_i x_i \le b_1 & \\
 & & \sum_{i\in I_{2j}} t_{ij} = 1 & \forall j \in J_2 \\
 & & t_{ij} \le u_i + y_i & \forall j \in J_2, \quad \forall i \in I_{2j} \\
 & & u_i \le x_i & \forall i \in I_{2j} \\
 & & \sum_{i\in I_2} a_i y_i \le b_2 & \\
 & & \sum_{i\in I_2} a_i(u_i + y_i) \le b_1 & \\
 & & \textit{All the variables are binary} &
\end{array}
\tag{5.4}
$$

We further denote by $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ the optimal values of the first-stage variables in the model $IPEV$. We refer to $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ as the *expected value solution* or *EV solution* (see [6]). 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 thus define the *expected result of using the EV solution* as

$$EEV = \sum_{j\in J_1}\sum_{i\in I_{1j}} d_{ij}\bar{z}_{ij} + \boldsymbol{E_{Q_2}\Psi(\bar{x}, Q_2)} \tag{5.5}$$

where $\boldsymbol{\Psi}(\cdot)$ is defined in (3.20)-(3.26). We make the following observation.

Observation 11. *EEV is an upper bound on the optimal value of the model SP, that is, $Optv(SP) \le EEV$.*

Proof. EEV is equal to the optimal value of the 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 the model SP. □

We compare EEV and $Optv(SP)$, and introduce the *value of stochastic solution* (VSS) [6] as the difference between these values, namely,

$$VSS = EEV - Optv(SP). \tag{5.6}$$

VSS measures the degree to which the decision $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ is inferior to the optimal solution of the model SP. In other words, the corresponding value of VSS provides the magnitude of improvement in the expected response time, which is achieved by solving the stochastic programming model versus solving the smaller expected-value problem.

Example 1 (Continued). For the example introduced in Section 3.1 we compare the first-stage solutions for the models SP and $IPEV$. The optimal value of the model SP is 34. We fix the first-stage solutions to be as obtained in model $IPEV$, then solve the model SP, to obtain $EEV = 35$. Thus, $VSS = EEV - Optv(SP) = 1$. □

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

It turns out that our model SP can also help in deciding whether making an effort to obtain further information about the second-stage queries would pay off. To show this, we define the *expected value of perfect information* ($EVPI$) for the problem SVS. In the literature (see, e.g., [6]), $EVPI$ measures the maximum amount of money that a decision maker would be ready to pay in return for complete information about the future. In the problem SVS, $EVPI$ would be the benefit, in units of expected query-evaluation costs, from knowing the actual query workload occurring at stage 2. This benefit, $EVPI$, is the maximum amount of query-evaluation time that we could save if we knew the exact query set that will occur at stage 2. In other words, the decision maker can compare the $EVPI$ with the cost of determining the exact query workload to occur at stage 2, and can then make the decision to minimize the expected costs.

Suppose we know at stage 1 which exact query set will occur at stage 2. In this case, we only need to study the model SP that results from that particular realization of the second-stage scenarios. That is, for each scenario ℓ we make the associated first-stage and second-stage decisions exclusively for it, and obtain the minimum costs of answering the queries at stage 1 and stage 2 under that scenario. We refer to this 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 3.2. For each query $q_j \in Q_1$ we use the notation V_{1j}^ℓ for $V_{1j}^\ell = \{v_i \in V_1^\ell : v_i \supseteq q_j\}$. Correspondingly, we denote 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.

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

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

Let the second-stage variables u_i^{ℓ}, y_i^{ℓ} and t_{ij}^{ℓ} be as defined in Section 3.2. It follows that we can now formulate an integer programming model for the ℓ^{th} scenario, for $\ell = 1$ to L. We refer to it as model IP^{ℓ}.

$$(IP^{\ell}) \qquad \text{minimize} \quad \sum_{j \in J_1} \sum_{i \in I_{1j}^{\ell}} d_{ij} z_{ij}^{\ell} + \sum_{j \in J_2^{\ell}} \sum_{i \in I_{2j}^{\ell}} d_{ij} t_{ij}^{\ell} \tag{5.7}$$

$$\text{subject to} \quad \sum_{i \in I_{1j}^{\ell}} z_{ij}^{\ell} = 1 \qquad \forall j \in J_1 \tag{5.8}$$

$$z_{ij}^{\ell} \leq x_i^{\ell} \qquad \forall j \in J_1, \quad \forall i \in I_{1j}^{\ell} \tag{5.9}$$

$$\sum_{i \in I_1^{\ell}} a_i x_i^{\ell} \leq b_1 \tag{5.10}$$

$$\sum_{i \in I_{2j}^{\ell}} t_{ij}^{\ell} = 1 \qquad \forall j \in J_2^{\ell} \tag{5.11}$$

$$t_{ij}^{\ell} \leq u_i^{\ell} + y_i^{\ell} \qquad \forall j \in J_2^{\ell}, \quad \forall i \in I_{2j}^{\ell} \tag{5.12}$$

$$u_i^{\ell} \leq x_i^{\ell} \qquad \forall i \in I_2^{\ell} \tag{5.13}$$

$$\sum_{i \in I_2^{\ell}} a_i y_i^{\ell} \leq b_2 \tag{5.14}$$

$$\sum_{i \in I_2^{\ell}} a_i (u_i^{\ell} + y_i^{\ell}) \leq b_1 \tag{5.15}$$

$$\textit{All the variables are binary} \tag{5.16}$$

Letting $Optv(IP^{\ell})$ denote the optimal value of the objective function of the 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. Following a common practice in the literature [6, 29], we refer to this value as the *wait-and-see solution value*, denoted by WS. We have that

$$WS = \sum_{\ell=1}^{L} p_{\ell} \cdot Optv(IP^{\ell}). \tag{5.17}$$

Observation 12. *WS is a lower bound on the optimal value of the model SP, that is, $WS \leq Optv(SP)$.*

Proof. Suppose that in the model IP^{ℓ} we introduce additional decision variables by defining (introducing) x_i^{ℓ} for all $i \in I_1$ (instead of only for $i \in I_1^{\ell}$) and defining z_{ij}^{ℓ} for all $j \in J_1$ and for all $i \in I_{1j}$ (instead of only for $i \in I_{1j}^{\ell}$). We now modify the model IP^{ℓ} by replacing the subscript sets I_1^{ℓ} and I_{1j}^{ℓ} by I_1 and I_{1j}, respectively, in the objective function (5.7) and constraints (5.8)-(5.10). The resulting model, which we denote by modified-IP^{ℓ}, has a larger search space of views. This does not affect the optimal value of the ℓ^{th} scenario problem. Thus, we obtain that the optimal value of the model modified-IP^{ℓ} is equal to that of the model 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

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

subject to the collection of all constraints of the submodels modified-IP^{ℓ} for $\ell = 1$ to L. Since there are no common variables among all the models modified-IP^{ℓ}s, the optimal value of the model IP^* can be obtained by combining the optimal values of the models IP^{ℓ}s together. More specifically, the optimal value of the model IP^* is equal to the expected value of $Optv(IP^{\ell})$ over L scenarios, i.e., $Optv(IP^*) = \sum_{\ell=1}^{L} p_\ell \cdot Optv(IP^{\ell})$. By the definition of WS, we obtain that $WS = Optv(IP^*)$. Suppose we add the following constraints into the model IP^* to obtain modified-IP^*.

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

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 $IP1$. By comparing the constraints of modified-IP^* and of the model $IP1$, we obtain 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 the model SP, that is, $WS \leq Optv(SP)$. $\square$

In the literature [6], the difference between the optimal value of the stochastic programming model and its wait-and-see solution is referred to as "the expected value of perfect information," denoted by $EVPI$. We have that

$$EVPI = Optv(SP) - WS. \tag{5.18}$$

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

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 the value of $EVPI$ is 2.5 units. $\square$

5.4 Numerical results

In this subsection we will discuss the computational experiments that we performed to assess the values of $VTVO$, VSS, and $EVPI$, as associated with our model SP. These results show that the magnitudes of $VTVO$, VSS, and $EVPI$ vary among instances, and that it depends on the structure and properties of both the database and the query workloads, as well as on the space limits. Specifically, these values are influenced significantly by (i) the space-limit ratio $\beta = b_2/b_1$, and by (ii) the structure of the database and the relative size of each view as compared with its descendant views in the view lattice.

To examine the impact of the ratio $\beta = b_2/b_1$ on these entities, we conduct experiments on several instances, as follows. Given an instance of the problem SVS – that is, given a 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 value of β on the solutions, by changing this value in the interval $[0, 1]$.

To examine the impact of the datasets, we use three different types of datasets: the TPC-H datasets, the symmetric synthetic datasets [20], and the type I non-symmetric synthetic datasets [20].

The TPC-H dataset is discussed in [36], and is used previously in many similar studies. The latter two datasets are synthetic datasets that we developed specifically for this study, to show that the magnitudes of the measures $VTVO$, VSS, and $EVPI$ indeed depend on the structure of the dataset. We provide an introduction to the synthetic datasets in Appendix B. Readers are referred to [20] for detailed descriptions of the synthetic datasets.

5.4.1 Numerical results on $VTVO$

We evaluated $VTVO$ for a number of instances of the problem SVS, of varying sizes and on different datasets. We observed that the relative magnitude of $VTVO$ varies among these instances, depending on the relative value of the storage limits, on the structure of the dataset, as well as on the structure of both the queries and the views.

In order to obtain $VTVO$, the value of two-stage versus one-stage, we need to solve the model OS that was introduced in Section 5.1. Note that the model OS is equivalent to the one-stage model $OVIP1$ that was proposed in [4] (see Section 2.2), if we define the evaluation cost $d_{ij} = a_i \cdot f_j$ for each view v_i and for each query q_j, where a_i is the size of the view v_i and f_j is as defined below.

$$f_j = \begin{cases} 1 & \forall q_j \in Q_1 - Q_2 \\ w_j & \forall q_j \in Q_2 - Q_1 \\ 1 + w_j & \forall q_j \in Q_1 \cap Q_2 \end{cases} \tag{5.19}$$

Here, w_j is as defined in (5.3) of Section 5.2, and $Q_2 = \cup_{\ell=1}^{L} Q_2^{\ell}$. Now we can apply the exact methods introduced in [4] to solve the model OS.

We examine first the impact of the ratio $\beta = b_2/b_1$ on the *VTVO*. This is equivalent to the impact of b_2 (the replacement space limit which limits the total size of the views to be materialized at stage two), as compared with b_1 (the entire space limit), on the *VTVO*. Given an instance of the problem SVS, we consider the associated one-stage model OS and the two-stage stochastic programming model SP. Note that the model OS can be considered as a special case of the model SP if we set $\beta = 0$, or equivalently, $b_2 = 0$. (If $b_2 = 0$, then replacement of views is not allowed at stage 2.) It follows that the resulting model does not have second-stage decision variables, and the views are materialized only in stage one. Thus, the resulting model is equivalent to the one-stage model OS. Now let us change the value of b_2 (or the value of β), i.e., increase its value from zero. We observe that changing the value of b_2 (or β) affects the corresponding optimal value of the model SP. More specifically, the increase in the value of b_2 (or β) allows for more changes in the views between stages one and two, and thus may result in a lower cost of query evaluation. Of course, as we change the value of b_2 (or β), the optimal value of the model OS remains the same since this value does no depend on b_2 (or β). As a result, the *VTVO* depends on the space-limit ratio β, and does not decrease with the increase in β.

We examine the impact of β on the value of *VTVO* by conducting a computational experiment on the instances, as follows. Given an instance of the problem SVS, we fix the following input parameters: database $\mathcal{D}$, query sets Q_1 and $\mathcal{Q}_2$, probabilities $\boldsymbol{p}$, and the total space limit b_1. We then construct instances for $\beta = 0, 0.1, 0.2, \ldots, 1$. (Or equivalently, we set $b_2 = 0, 0.1b_1, 0.2b_1, \ldots, b_1$.) For each value of β we solve the models SP and OS, evaluate the corresponding value of the *VTVO*, and report the results for all instances. More specifically, we constructed instances based on a 13-attribute type I non-symmetric synthetic dataset [20] in Example 2 below, and based on the 13-attribute TPC-H dataset [36] in Example 3, as follows.

Example 2. We construct this example based on a 13-attribute type I non-symmetric synthetic dataset [20], which we denote by D_I. The master table contains all the possible entries resulting from each of the thirteen attributes taking all of its possible values. (The master table is the

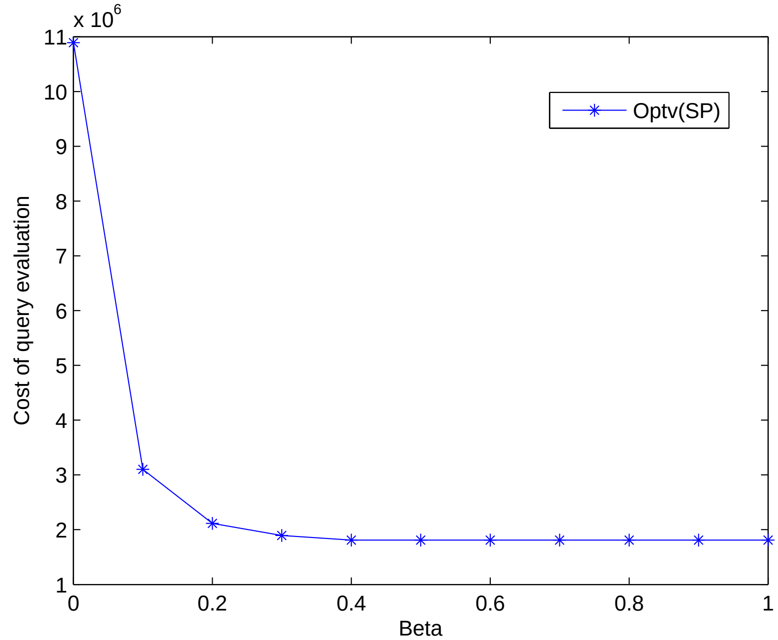

Figure 5.1: The impact of β (Beta) on the optimal values for Example 2

only table in our synthetic dataset.) The numbers of different values that the 13 attributes respectively take are 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4 and 4. To obtain the query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ for this instance, we use an approach that is similar to the approach introduced in Section 4.4.1. The number of queries in each query set is 20, 15 and 21, respectively. We set $\boldsymbol{p} = (0.5, 0.5)$ and $b_1 = 1,632,334$. We then solve this instance of the problem *SVS* associated with each value of β, and obtain the optimal value of the model *SP*. We report the results in Figure 5.1. We then evaluate the corresponding *VTVO*, for $\beta = 0, 0.1, 0.2, \ldots, 1$. All the results are shown in Table 5.1.

From Figure 5.1 and Table 5.1, we observe that when β changes from 0 to 0.1, the optimal value of the corresponding model *SP* decreases sharply. When β is greater than or equal to 0.4, the optimal value of the corresponding model *SP* remains the same. From the results in Table 5.1, we observe that *VTVO* increases with increase in b_2. When we set $\beta = 0.4$ ($b_2 = 652,934$), the value of *VTVO* is more than 5 times the optimal value of the model *SP*. In other words, in this instance, applying a two-stage model (i.e., the model *SP*), instead of a one-stage model (i.e., the model *OS*), reduces the corresponding response time by more than 80%. Thus, solving the stochastic view-selection model could be very beneficial for some instances. $\square$

Example 3. This example is based on a 13-attribute TPC-H dataset [36] denoted by D_{T-13}. The number of queries in the query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ are 20, 30, and 22, respectively.

Table 5.1: Results of *VTVO* for Example 2

β	b_2	$Optv(SP)$	*VTVO*	$\frac{VTVO}{Optv(SP)}$
0	0	10,892,471	0	0.00%
0.1	163,233	3,101,499	7,790,972	251.20%
0.2	326,467	2,114,231	8,778,240	415.20%
0.3	489,700	1,893,047	8,999,424	475.39%
0.4	652,934	1,810,103	9,082,368	501.76%
⋮	⋮	⋮	⋮	⋮[1]
1.0	1,632,334	1,810,103	9,082,368	501.76%

[1] The missing rows have identical values in each of $Optv(SP)$, *VTVO*, and $VTVO/Optv(SP)$.

Table 5.2: Results for *VTVO* for Example 3

β	b_2	$Optv(SP)$	*VTVO*	$\frac{VTVO}{Optv(SP)}$
0	0	121236449	0	0.00%
0.1	2877701	118256288	2980161	2.52%
0.2	5755402	115379409	5857040	5.08%
0.3	8633103	113727778	7508671	6.60%
0.4	11510804	112078447	9158002	8.17%
0.5	14388505	110579265	10657184	9.64%
0.6	17266205	109230599	12005850	10.99%
0.7	20143906	108330649	12905800	11.91%
0.8	23021607	107431234	13805215	12.85%
0.9	25899308	106831606	14404843	13.48%
1.0	28777009	106831606	14404843	13.48%

We set the total space limit $b_1 = 28,777,009$ and $\boldsymbol{p} = (0.5, 0.5)$. In Figure 5.2 we plot the impact of the value of β on the optimal value of the model SP. We evaluated the *VTVO* and reported the results in Table 5.2. From Figure 5.2, we observe that $Optv(SP)$ declines steadily as the value of β increases from 0 to 0.9. When β is greater than 0.9, the optimal value of the corresponding model SP remains the same. As shown in Table 5.6, the ratio of the *VTVO* over the optimal value of the model SP also increases as the space limit b_2 increases. When $\beta = 0.9$ ($b_2 = 25,899,308$), the ratio is 13.48%, indicating that the two-stage model SP can save 13.48% of the evaluation cost (response time) obtained by the one-stage model OS. $\square$

Discussion. We examine the impact of different datasets on the *VTVO* by comparing the computational results of the ratio of *VTVO* over the optimal value of SP for the instances based on different datasets. More specifically, we compare the results of Examples 2 and 3

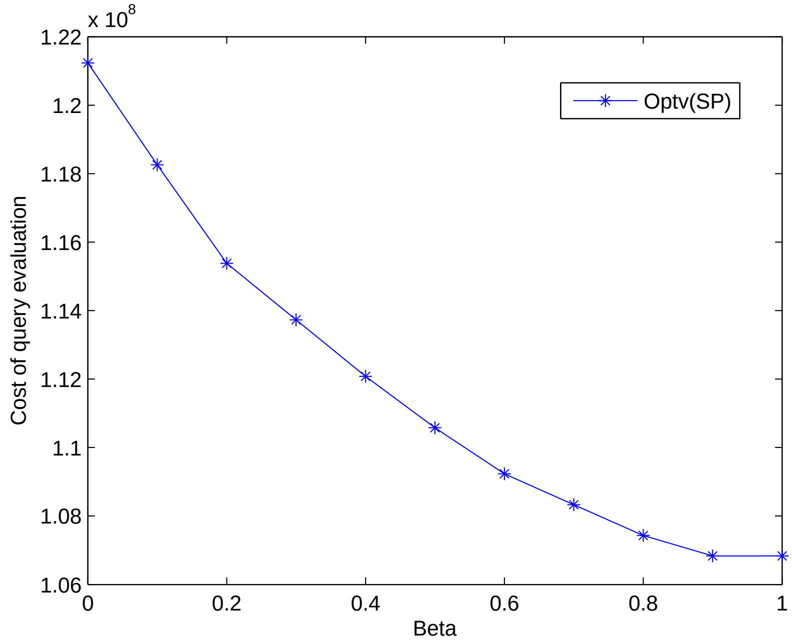

Figure 5.2: The impact of β (Beta) on optimal values for Example 3

as given in Tables 5.1 and 5.2. We observe that for the same amount of space for the views materialized in stage two, i.e., for the same value of β, the instance based on the type I non-symmetric synthetic dataset D_I has a significantly higher ratio of *VTVO* over $Optv(SP)$ than the instance based on the TPC-H dataset D_{T-13}. For example, among the instances that we have studied, the benefit for an instance based on D_I can be 501.76% of the optimal value of the associated model SP, while the benefit for an instance based on D_{T-13} is no more than 13.48% of the optimal value of the associated model SP. This indicates that while we can obtain some benefit by applying the two-stage model, instead of the one-stage model, to solve the problem *SVS*, the magnitude of the benefit depends on the structure and size of the queries and views involved.

5.4.2 Numerical results on VSS

In another computational experiment, we evaluated the Value of Stochastic Solution *VSS* for a collection of instances of varying sizes based on different types of datasets. The results show that the relative value of VSS varies among these instances, and that its magnitude depends on both the space limits and the structure of the underlying database.

In order to obtain the VSS, we need to solve the model $IPEV$ introduced in Section 5.2, and calculate the value of EEV as defined in Equation 5.5. In the model $IPEV$, we are given a deterministic query set Q_2 at stage 2. Thus, the model $IPEV$ could be considered as a special case of the model SP, in which there is only one scenario at stage 2. To reduce the search spaces of views for the model $IPEV$, we can apply an approach that is similar to the one introduced in Section 4.2. We then solve it to obtain the EV solution. We plug the EV solution into Equation (5.5), and solve the second-stage problem to obtain EEV.

Again in a computational experiment, we solved the model $IPEV$ and calculated the corresponding value EEV for a collection of instances of varying sizes based 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 space limits and on the structure of the underlying database, 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.

First we show that the magnitude of VSS depends on the value of the space-limit ratio β, and on the relative size of the space limit b_2, as compared with the sizes of the queries occurring in stage two.

In order to examine the impact of the space-limit ratio β on the value of VSS, we fix the input parameters ($\mathcal{D}$, Q_1, $\mathcal{Q}_2$, $\boldsymbol{p}$, b_1) in an instance of the problem SVS, and construct instances for $\beta = 0$, 0.1, 0.2, …, 1. We solve the models SP and $IPEV$, evaluate EEV and VSS, and compare the results for these instances. More specifically, we construct instances based on the 13-attribute symmetric synthetic dataset in Example 4 below, and based on the 13-attribute TPC-H dataset in Example 5, as follows.

Example 4. This example is based on a 13-attribute symmetric synthetic dataset [20] $D_S(13; 2)$ (denoted as D_S for brevity). In this dataset each attribute takes 2 different values, and the master table contains all the possible entries by taking all combinations of values for the thirteen attributes. The queries are generated by the same method as in Section 4.4.1. The sizes of the query sets Q_1, $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ are 20, 18, and 15, respectively. We set $b_1 = 15764$ and $\boldsymbol{p} = (0.5, 0.5)$. We construct the instances with different values of b_2, for $\beta = 0$, 0.1, 0.2, …, 1, where $\beta = b_1/b_2$. We compare EEV with the optimal values of the models SP and $IPEV$ for these instances, as shown in Figure 5.3. We also compare VSS with the optimal value of the model SP, and report the results in Table 5.3.

In Figure 5.3, we observe that the inequality $Optv(SP) \leq EEV \leq Optv(IPEV)$ always holds, and all the three values decrease as β increases from 0 to 1. As shown in Table 5.3, when $\beta = 0$ ($b_2 = 0$), the optimal value of the model SP is equal to EEV. When $\beta = 0.4$ ($b_2 = 6301$), the expected response time would be 50.14% higher if we used the expected-value model $IPEV$ instead of the stochastic programming model SP. This indicates that it is quite beneficial to take into account the stochastic properties of the future when formulating a model for solving

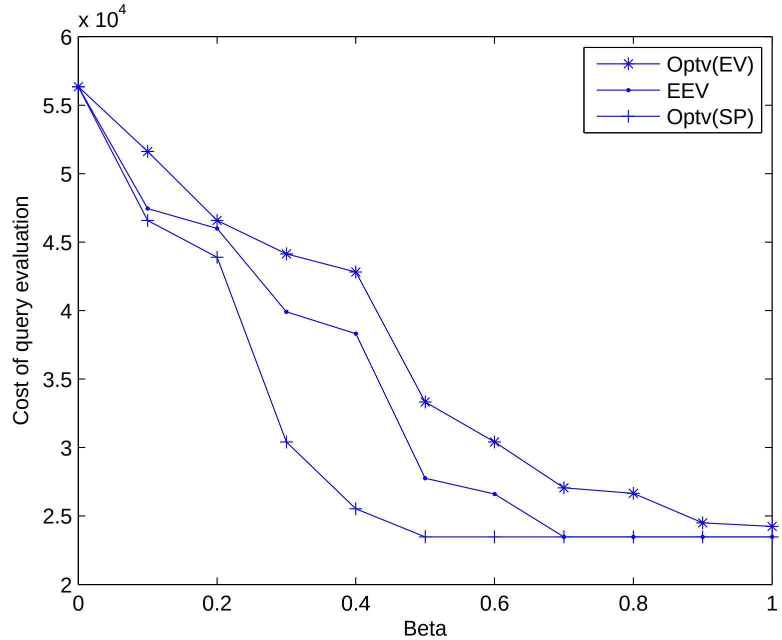

Figure 5.3: The impact of β (Beta) on the values of EEV and on the optimal values of the models SP and $IPEV$ for Example 4

this stochastic view-selection problem. In addition, when $\beta \geq 0.7$ ($b_2 \geq 11035$), the EEV and the optimal value of the model SP become the same. $\square$

Example 5. This example is based on the 7-attribute TPC-H dataset [36], denoted by D_{T-7}. The sizes of the query sets Q_1 and $\mathcal{Q}_2 = \{Q_2^1, Q_2^2\}$ are 20, 22, and 22, respectively. We set $b_1 = 21,605,174$ and $\boldsymbol{p} = (0.5, 0.5)$. We construct 11 instances with different values of b_2, for $\beta = 0,\ 0.1,\ 0.2,\ \ldots,\ 1$, where $b_2 = \beta \cdot b_1$. We compare the EEV with the optimal values of the models SP and $IPEV$ for these instances, as shown in Figure 5.4. We also compare VSS with the optimal value of the model SP, and report the results in Table 5.4.

Figure 5.4 shows that for each value of β, $Optv(SP) \leq EEV \leq Optv(EV)$, and all the three values decrease as β increases from 0 to 1. As β approaches 1, even though the difference between the optimal values of the models SP and $IPEV$ is significant, the EEV and the optimal value of the model SP are quite close. According to the results of Table 5.4, when $\beta = 0$ or $\beta \geq 0.9$ (i.e., $b_2 = 0$ or $b_2 \geq 19,444,657$), the EEV and the optimal value of SP are equal, i.e., $VSS = 0$. $\square$

Discussion. In general, given an instance of the problem SVS, when $\beta = 0$ (i.e., $b_2 = 0$), the views are materialized only in stage one. As a result, the models SP and $IPEV$ for this

Table 5.3: Values of VSS for Example 4

β	b_2	$Optv(SP)$	EEV	VSS	$VSS/Optv(SP)$
0	0	56344	56344	0	0.00%
0.1	1576	46578	47458	880	1.89%
0.2	3153	43890	46002	2112	4.81%
0.3	4729	30401	39906	9505	31.27%
0.4	6301	25522	38318	12796	50.14%
0.5	7882	23474	27762	4288	18.27%
0.6	9458	23474	26606	3132	13.34%
0.7	11035	23474	23474	0	0.00%
0.8	12611	23474	23474	0	0.00%
0.9	14188	23474	23474	0	0.00%
1.0	15764	23474	23474	0	0.00%

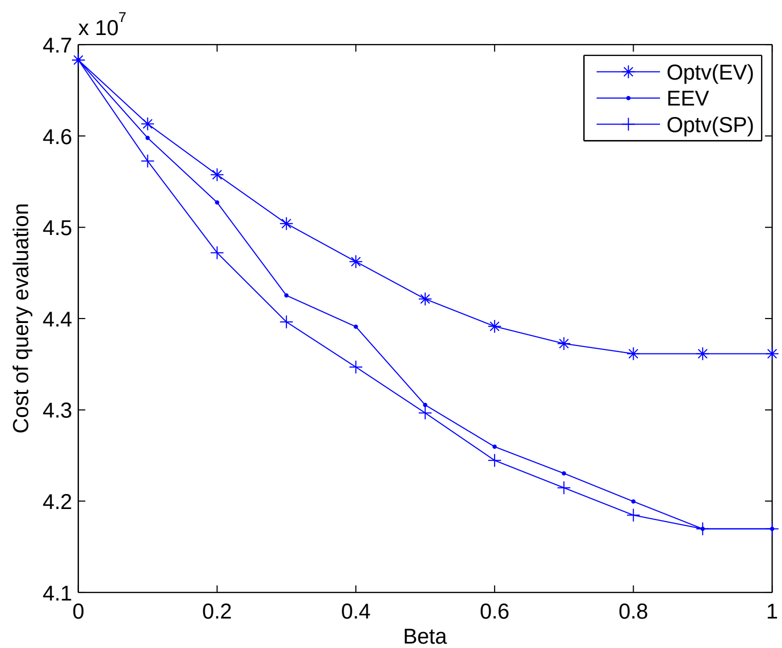

Figure 5.4: The impact of β (Beta) on the values of EEV and on the optimal values of the models SP and $IPEV$ for Example 5

Table 5.4: Values of VSS for Example 5

β	b_2	$Optv(SP)$	EEV	VSS	$VSS/Optv(SP)$
0.0	0	46,830,690.0	46,830,690.0	0.0	0.00%
0.1	2,160,517	45,725,011.0	45,978,593.5	253,582.5	0.55%
0.2	4,321,035	44,720,228.0	45,270,928.5	550,700.5	1.23%
0.3	6,481,552	43,962,738.0	44,252,958.0	290,220.0	0.66%
0.4	8,642,070	43,468,681.5	43,910,662.0	441,980.5	1.02%
0.5	10,802,587	42,965,415.5	43,052,944.0	87,528.5	0.20%
0.6	12,963,104	42,446,604.5	42,595,893.5	149,289.0	0.35%
0.7	15,123,622	42,145,588.0	42,303,533.5	157,945.5	0.37%
0.8	17,284,139	41,845,770.5	41,995,677.5	149,907.0	0.36%
0.9	19,444,657	41,696,570.5	41,696,570.5	0.0	0.00%
1.0	21,605,174	41,696,570.5	41,696,570.5	0.0	0.00%

case are equivalent to the one-stage model OS proposed in Section 5.1. We have that $EEV = Optv(SP) = Optv(IPEV)$, and that $VSS = 0$. In other words, employing the stochastic programming model SP instead of the expected-value model $IPEV$ to solve the problem SVS will bring us no benefit. When $\beta > 0$ (i.e., $b_2 > 0$), we allow the change of views at stage two. The benefits can be obtained by solving the model SP instead of the model $IPEV$. As the value of β approaches 1 (or equivalently, the value of b_2 approaches that of b_1), a greater portion of the entire storage space can be utilized at stage two. Thus, the first-stage decisions on which views to materialize in stage one become less relevant to the knowledge of the queries occurring in stage two. It results in smaller difference between the first-stage solutions of the models SP and $IPEV$. When $\beta = 1$ (i.e., $b_2 = b_1$), we can utilize the entire space to materialize the views in stage two. In other words, the views materialized in stage one can be selected based on only the queries occurring in stage one. It follows that in this case, the EV solution, which is an optimal solution of the first-stage variables for the model $IPEV$, is also an optimal solution of the first-stage variables for the model SP. Thus, we have that $EEV = Optv(SP)$ and that $VSS = 0$. This also indicates that no benefit can be obtained by taking into account the stochastic properties in solving the problem SVS. However, when $0 < \beta < 1$ (i.e., $0 < b_2 < b_1$), the impact varies for different values of β.

Note that the magnitude of VSS also depends on the structure of the database. In order to examine the impact of the structure of the data, we compare the computational results of Example 4, which is based on the symmetric synthetic dataset D_S, with the results of Example 5, which is based on the TPC-H dataset D_{T-7}. From Tables 5.3 and 5.4, we observe that the maximum value of the ratio $VSS/Optv(SP)$ among all the instances in Example 5 is 1.23%, while in Example 4 the ratio has an average value of 19.95% for the instances with non-zero values of VSS. This indicates that the benefit obtained by taking into account the stochastic properties in solving the problem SVS varies among instances on different datasets.

5.4.3 Numerical results on $EVPI$

We evaluated through a computational experiment the Expected Value of Perfect Information $EVPI$ for a collection of instances of varying sizes based on different types of datasets. We observe that the value of $EVPI$ could also vary significantly, depending on both the space limits and the specifics of the database and the query sets.

To obtain $EVPI$, the expected value of perfect information, we first solve each submodel IP^{ℓ} that we introduced in Section 5.3. Note that the subproblem IP^{ℓ} could be considered as a problem SVS with only one potential query set Q_2^{ℓ} (i.e., scenario) occurring at stage 2. Thus, we apply here approaches that are similar to those of Section 4.2 to reduce the search spaces of views for each model IP^{ℓ}, and to obtain the optimal value more efficiently.

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. Similarly to the previous experiments, we observe that the value of $EVPI$ could vary significantly, depending on the space-limit ratio β, and on the specifics of the database and of the query instance.

We first examine the impact on the $EVPI$ of the space-limit ratio β, which is the ratio of the replacement space limit for the views to be materialized in stage two (b_2) over the total space limit (b_1). We apply a similar approach as we did for *VTVO* and VSS. We fix the input parameters ($\mathcal{D}$, Q_1, $\mathcal{Q}_2$, $\boldsymbol{p}$, b_1) for each instance of the problem SVS, and construct instances for $\beta = 0,\ 0.1,\ 0.2,\ \ldots,\ 1$. More specifically, we evaluate the $EVPI$ for the instances based on the symmetric synthetic dataset D_S in Example 6 below, and based on the TPC-H dataset D_{T-7} in Example 7, as follows.

Example 6. We construct this instance using the same dataset and query sets as that of Example 4 introduced in Section 5.4.2. Given the symmetric synthetic dataset D_S, query sets Q_1 and $\mathcal{Q}_2$, and probabilities $\boldsymbol{p} = (0.5, 0.5)$, we set $b_1 = 7882$. For $\beta = 0,\ 0.1,\ \ldots,\ 1$, we solve the l^{th} sub-model IP^{ℓ} associated with Q_2^{ℓ}, for $\ell = 1, 2$, and evaluate the values of WS and $EVPI$. The results are shown in Table 5.5. We observe that when $\beta \geq 0.8$ (i.e., when $b_2 \geq 6301$), the $EVPI$ is 0. It indicates that the knowledge of perfect information of the actual query workload at stage 2 provides no more benefit. When $\beta = 0.4$ (i.e., when $b_2 = 3153$), the perfect information can reduce the cost of query evaluation by 22.24% of the optimal value of the model SP. It implies that the decision maker would pay to pursue the perfect information only if the cost of obtaining such information is less than or equal to the benefits obtained by reducing the response time by 22.24% of the optimal value of the model SP. □

Table 5.5: Results of $EVPI$ for Example 6

β	b_2	$Optv(SP)$	WS	$EVPI$	$\frac{EVPI}{Optv(SP)}$
0	0	56344	50610	5734	10.18%
0.1	6301	51940	49586	2354	4.53%
0.2	1576	46578	46036	542	1.16%
0.3	2365	46002	37586	8416	18.29%
0.4	3153	43890	34130	9760	22.24%
0.5	3941	39167	32690	6477	16.54%
0.6	4729	30401	27594	2807	9.23%
0.7	5517	26674	26098	576	2.16%
0.8	6301	25522	25522	0	0.00%
0.9	7094	25522	25522	0	0.00%
1.0	7882	23474	23474	0	0.00%

Table 5.6: Results of $EVPI$ for Example 7

β	b_2	$Optv(SP)$	WS	$EVPI$	$\frac{EVPI}{Optv(SP)}$
0	0	46,830,690.0	45,308,461.0	1,522,229.0	3.25%
0.1	2,160,517	45,725,011.0	44,652,964.5	1,072,046.5	2.34%
0.2	4,321,035	44,720,228.0	44,095,045.0	625,183.0	1.40%
0.3	6,481,552	43,962,738.0	43,699,314.0	263,424.0	0.60%
0.4	8,642,070	43,468,681.5	43,344,974.5	123,707.0	0.28%
0.5	10,802,587	42,965,415.5	42,864,201.5	101,214.0	0.24%
0.6	12,963,104	42,446,604.5	42,445,640.5	964.0	0.00%
0.7	15,123,622	42,145,588.0	42,145,588.0	0.0	0.00%
0.8	17,284,139	41,845,770.5	41,845,770.5	0.0	0.00%
0.9	19,444,657	41,696,570.5	41,696,570.5	0.0	0.00%
1.0	21,605,174	41,696,570.5	41,696,570.5	0.0	0.00%

Example 7. We construct this instance using the same data as that for Example 5. That example, which was introduced in Section 5.4.2, is based on the 7-attribute TPC-H dataset D_{T-7}. For $\beta = 0$, 0.1, ..., 1, we solve the l^{th} sub-model IP^{ℓ} associated with Q_2^{ℓ}, for $\ell = 1, 2$, and evaluate the WS and $EVPI$. The results are shown in Table 5.6. We observe that when $\beta \geq 0.6$ (i.e., when $b_2 \geq 12, 963, 104$), even though the optimal value of the corresponding model SP varies, the value of $EVPI$ is always 0. The value of $EVPI$ is relatively high when the value of β is small. When $\beta = 0$ (i.e., $b_2 = 0$), the perfect information of the actual query workload at stage two can reduce the cost of query evaluation by 3.25% of the optimal value of the model SP. □

Discussion. In general, if $\beta = 1$ (i.e., $b_2 = b_1$), the first-stage problem and the second-stage problem can be solved separately. In other words, the views to be materialized in stage one can be selected based on only the queries occurring in stage one. As a result, a first-stage optimal solution of the model SP is also a first-stage optimal solution of each model IP^{ℓ}. The value of WS is equal to the optimal value of the model SP. Thus, $EVPI = 0$. It indicates that the knowledge of the exact stage-2 query workload provides little benefit when b_2 is relatively large. When $0 < \beta < 1$ $(0 \leq b_2 < b_1)$, the impact on $EVPI$ varies.

Finally, we observe the impact of the datasets on $EVPI$. We compare the results of Example 6 which is defined using the dataset D_S, and the results of Example 7 which is defined using the dataset D_{T-7}. From Tables 5.5 and 5.6, we observe that, in general, Example 7 has significantly higher values of $EVPI$ than Example 6, and the ratios of $EVPI/Optv(SP)$ for Example 6 are generally higher than those for Example 5.4. This implies that both the relative magnitude and the absolute magnitude of $EVPI$ depend on the structure of the underlying dataset and on the specific instance of the problem.

Chapter 6

The one-stage view-selection problem

In this chapter we return to the one-stage view-selection problem OVS introduced in Section 2.2, and study this problem in view of the three reductions that we introduced in Chapter 4 for the two-stage problem. In Section 6.1 we discuss these reductions in the context of the problem OVS, and in Section 6.2 we discuss the effectiveness of these reductions in the context of the efficiency and scalability of the resulting integer programming models.

6.1 The problem OVS and the model $OVIP2$

We consider the one-stage view-selection problem OVS, which is extensively discussed in [3, 4], and which we introduced in Section 2.2. In this context we have a collection Q of queries on a given database, and a storage limit b on the total size of the views that we may materialize. The problem is to select a collection of views to materialize, so as to minimize the total response time for the given collection of queries, subject to the given storage limit. The one-stage view-selection problem can be considered as a special case of the two-stage problem that we discussed in Chapter 3.

In Section 2.2, we reviewed the work of Asgharzadeh Talebi et al. [3, 4], and introduced the integer programming model $OVIP1$. We also discussed two observations and the associated techniques that they propose to reduce (i.e., to prune) the search space of views for the model. The reduced model obtained by applying the results of these techniques is denoted by $OVIP1'$. We made similar observations in the context of the two-stage stochastic view-selection problem SVS in Sections 4.2.1 and 4.2.2, and referred to them as the first and the second reductions, respectively.

Note that we can further reduce the search space of views in the model $OVIP1'$ for the one-stage view-selection problem OVS, by conducting a reduction similar to the third reduction for the problem SVS that we introduced in Section 4.2.3. Hence, the size of the model $OVIP1'$ can be reduced accordingly. To do this, we can eliminate the view v if there exists a view v' such that v answers v' and such that the *maximum benefit* of v is less than or equal to that of v'. Recall that in Section 4.2.2 we defined the *maximum benefit* of v with respect to a set of queries Q as the amount of space that we can save by materializing a view v instead of materializing all the input queries that v can answer.

For the problem OVS, we denote by $\overline{V}$ the reduced search space of views after the first, second, and third reductions. Furthermore, for each query $q_j \in Q$, we define $\overline{V_j} = \{v_i \in \overline{V} : v_i \supseteq q_j\}$. We also define the associated sets of subscripts for $\overline{V}$ and $\overline{V_j}$ as $\overline{I}$ and $\overline{I_j}$, respectively.

Hence, we can obtain a model with reduced size whose structure is similar to the model $OVIP1'$, except that instead of V we use the reduced search space of views $\overline{V}$. We can now define the decision variable x_i for each $i \in \overline{I}$, and define the decision variable z_{ij} for $j \in J$ and for $i \in \overline{I_j}$. Using these notations, a smaller IP model can be defined as follows. We refer to this model as $OVIP2$.

$$(OVIP2) \quad \text{minimize} \quad \sum_{j \in J} \sum_{i \in \overline{I_j}} d_{ij} z_{ij} \tag{6.1}$$

$$\text{subject to} \quad \sum_{i \in \overline{I_j}} z_{ij} = 1 \qquad \forall j \in J \tag{6.2}$$

$$z_{ij} \leq x_i \qquad \forall j \in J, \forall i \in \overline{I_j} \tag{6.3}$$

$$\sum_{i \in \overline{I}} a_i x_i \leq b \tag{6.4}$$

$$\textit{All the variables are binary} \tag{6.5}$$

Note that an optimal solution for the model $OVIP2$ is guaranteed to be optimal for $OVIP$. Hence, it provides an optimal solution for the original problem OVS. Obviously, the size of the model $OVIP2$ in terms of the number of variables and constraints is potentially smaller than that of the models $OVIP1'$ and $OVIP$. The magnitude of the difference in size, however, depends on the specifics of each instance. In Section 6.2 we compare the sizes of these two models on an empirical basis, and show that the difference in the sizes of the two models can be significant.

6.2 Experimental results on the model $OVIP2$

In order to evaluate the effectiveness of the model $OVIP2$ for solving the one-stage view-selection problem OVS, we conducted a computational experiment. In the experiment, we directly compare the size of the models $OVIP2$ and $OVIP1'$ (proposed in [3,4]), for a collection of randomly generated instances. The results of this experiment show that the magnitude of the difference in the size of these two models depends on the specifics of each instance, and that for many instances this difference is quite significant.

6.2.1 Reduction in the size of the search space

We compare the sizes of the models $OVIP1$, $OVIP1'$, and $OVIP2$ for several randomly generated instances of the problem *OVS*. More specifically, we constructed 20 instances on the 17-attribute TPC-H dataset. For the instances 1 to 5 in this group, the number of attributes of each query is a random number between 1 and 16 (that is, each query is chosen from the entire view lattice). We refer to the query types of these instances as "entire view lattice". For each of the remaining instances, all of the queries are from certain levels (positions) of the view lattice. For the instances with IDs between 6 and 10 inclusively, the number of attributes in each query is a random number between 1 and 8. (That is, each query is chosen from the bottom levels of the view lattice.) For the instances with IDs between 11 and 15, the number of attributes of each query is a random number between 5 and 12. (That is, each query is chosen from the middle levels of the view lattice.) Finally for the instances with IDs between 16 and 20, the number of attributes of each query is a random number between 9 and 16. (That is, each query is chosen from the top levels of the view lattice.) We refer to the query types of those instances as "bottom-level", "middle-level", and "top-level", respectively. Within the collection of instances for each query type, the number of queries ranges from 20 to 100. We present those instances in Appendix C. For each instance, we report in Table 6.1 the number of views in models $OVIP1$, $OVIP1'$ and $OVIP2$. For each instance, we also report the number of variables and constraints in each model. (We do not construct the model $OVIP1$, since [4] has already shown that $OVIP1'$ is much more effective than $OVIP1$.) In Table 6.2 we report the corresponding execution times. More specifically, we report the time that it takes to build each model, the time that it takes to solve each model by the CPLEX IP solver, and the total execution time of each model. We make the following observations.

1. In every instance, the numbers of views in the search space for models $OVIP1'$ and $OVIP2$ are significantly smaller than the corresponding numbers in the model $OVIP1$. This reduction is relatively more significant for instances with a smaller ratio of the number of queries over the total number of views in the dataset. More specifically, when we have a larger number of queries, the ratio of the number of queries over the total number of views increases (since the total number of views for a 17-attribute dataset is constant and equal to 131072), and the magnitude of associated reductions in the number of views in the search space decreases. We made a similar observation on a collection of

Table 6.1: Comparison of the numbers of views and the sizes in the models $OVIP1$, $OVIP1'$, and $OVIP2$, for instances on the 17-attribute TPC-H dataset

instance	query type	$\|Q\|$	number of views			number of variables		number of constraints	
			$OVIP1$	$OVIP1'$	$OVIP2$	$OVIP1'$	$OVIP2$	$OVIP1'$	$OVIP2$
1	entire view lattice	20	115,200	125	100	962	784	858	705
2		40	118,420	1,920	915	15,567	8,228	13,688	7,354
3		60	127,760	3,724	1,796	41,949	22,646	38,286	20,911
4		80	127,544	7,070	4,589	86,895	60,702	79,906	56,194
5		100	128,960	16,159	7,710	216,950	118,653	200,892	111,044
6	bottom-level	20	98,744	1,071	1,004	7,793	7,426	6,743	6,443
7		40	125,576	7,409	3,192	81,768	39,027	74,400	35,876
8		60	130,076	21,678	8,048	330,371	138,305	308,754	130,318
9		80	130,916	35,927	14,248	570,725	271,489	534,879	257,322
10		100	130,984	44,229	17,937	817,183	396,196	773,055	378,360
11	middle-level	20	15,415	301	301	1,642	1,642	1,362	1,362
12		40	27,665	1,507	1,504	9,036	9,024	7,570	7,561
13		60	38,220	3,787	3,614	26,085	25,289	22,359	21,736
14		80	34,045	4,351	4,305	29,795	29,596	25,525	25,372
15		100	42,507	7,304	7,278	55,638	55,530	48,435	48,353
16	top-level	20	1,027	89	89	378	378	310	310
17		40	1,500	234	234	1,077	1,077	884	884
18		60	1,733	316	316	1,543	1,543	1,288	1,288
19		80	3,758	669	669	3,584	3,584	2,996	2,996
20		100	3,145	687	687	3,731	3,731	3,145	3,145

Table 6.2: Comparison of the computing times of the models $OVIP1'$ and $OVIP2$, for instances on the 17-attribute TPC-H dataset

inst-ance	query type	number of queries	time to build model (sec.)		time to solve model (sec.)		total time (sec.)	
			$OVIP1'$	$OVIP2$	$OVIP1'$	$OVIP2$	$OVIP1'$	$OVIP2$
1	entire view lattice	20	0.30	0.30	0.14	0.11	0.44	0.41
2		40	0.41	0.39	2.86	1.56	3.27	1.95
3		60	0.66	0.64	7.47	3.47	8.13	4.11
4		80	0.81	1.13	31.77	17.17	32.58	18.30
5		100	1.33	2.05	116.05	44.15	117.38	46.20
6	bottom-level	20	0.41	0.50	1.58	1.72	1.99	2.22
7		40	0.70	0.80	104.28	32.53	104.98	33.33
8		60	1.56	2.28	>30min	385.14	>30min	387.42
9		80	3.19	6.19	out of memory	>30min	out of memory	>30min
10		100	3.11	9.42	out of memory	>30min	out of memory	>30min
11	middle-level	20	0.25	0.25	0.42	0.42	0.67	0.67
12		40	0.31	0.34	4.44	3.89	4.75	4.23
13		60	0.45	0.67	22.42	21.05	22.87	21.72
14		80	0.52	0.80	103.19	93.26	103.71	94.06
15		100	0.61	1.39	>30min	>30min	>30min	>30min
16	top-level	20	0.25	0.22	0.11	0.05	0.36	0.27
17		40	0.30	0.27	2.56	2.28	2.86	2.55
18		60	0.31	0.33	1.08	1.06	1.39	1.39
19		80	0.34	0.38	1.63	1.62	1.97	2.00
20		100	0.41	0.44	3.01	3.14	3.42	3.58

instances on the 13-attribute dataset.

2. Comparing the number of views in $OVIP1'$ and in $OVIP2$, we note that for the instances where the queries are from the entire view lattice (instances 1 through 5) or from the bottom levels of the view lattice (instances 6 through 10), the size of $OVIP2$ is significantly smaller than that of $OVIP1'$. We also observe that for these instances the size of $OVIP2$, when expressed by the total number of variables and constraints, is much smaller than that of $OVIP1'$. In addition, the time to solve $OVIP2$ is also significantly smaller than that for $OVIP1'$. On the other hand, for those instances where the queries are from the middle levels of the view lattice (instances 11 through 15) the reduction in the number of views in the search space from model $OVIP1'$ to $OVIP2$ is not as significant, resulting in no significant difference in the size or the solving time between the corresponding models $OVIP1'$ and $OVIP2$. Moreover, for the instances from the top levels of the view lattice (instances 16 through 20), we observe no reduction in the numbers of views in the search space from model $OVIP1'$ to $OVIP2$. These observations are consistent with our expectations, as evident from the nature of Observation 7, which we introduced in Section 4.2.3.

3. It is observed that for all instances, the time to build the model $OVIP2$ is relatively larger than that for $OVIP1'$. This increase in time is much more significant for instances with large numbers of queries. Comparing the time to build and the time to solve the models $OVIP1'$ and $OVIP2$, however, we observe that the decrease in the time to solve the model $OVIP2$, as compared with the model $OVIP1'$, is much more significant than the increase in the time to build the model $OVIP2$, as compared with the model $OVIP1'$. In general, it is also observed that the total time for $OVIP2$ is relatively smaller than that for $OVIP1'$. This reduction in the total time is more significant for those instances where the queries are from the entire view lattice (instances 1 through 5) or from the bottom levels of the view lattice (instances 6 through 10).

6.2.2 Scalability of the model $OVIP2$

In order to evaluate the scalability of our approach, we have attempted to solve larger instances of the problem OVS by the model $OVIP2$.

In Table 6.2 we observe that for the instances 6 through 10, where the queries are from the bottom levels of the view lattice, the total execution time to solve the model $OVIP2$ ranges from 2.22 seconds to 387.42 seconds. Here, the number of queries in each instance is no more than 60. However, we could not solve the instances 9 or 10 within the assumed time limit of 30 minutes.

Table 6.3: Scalability of $OVIP2$ for instances on the 17-attribute TPC-H dataset

instance	query type	number of queries	number of views	time to build $OVIP2$ (sec.)	time to solve $OVIP2$ (sec.)	total time (sec.)
21	entire view lattice	220	23256	13.00	304.00	317.00
22		240	23921	14.25	374.22	388.47
23		260	25922	15.31	out of memory	out of memory
24	top-level	320	2673	0.91	559.58	560.49
25		340	2703	1.08	814.27	815.35
26		360	2930	1.20	>30min	>30min

As reported in Table 6.2, for the instances 11 through 15, where queries are from the middle levels of the view lattice on the 17-attribute dataset, we observe that we could solve all the instances whose number of queries is no more than 80. The total execution time for $OVIP2$ ranges in theses cases from 0.67 seconds to 94.06 seconds. However, when we increase the size of the query set further to 100, the solver fails to provide an optimal solution for model $OVIP2$ within our time limit of 30 minutes.

For the instances where the queries are from the entire view lattice on the 17-attribute dataset (instances 1 through 5) that we report in Table 6.2, we note that the total execution time for the model $OVIP2$ ranges from 0.41 seconds to 46.20 seconds. We also note that this time increases as we increase the number of queries in the query set.

In addition, as observed in Table 6.2, we could solve within 5 seconds all the instances where the queries are from the top levels of the view lattice on the 17-attribute dataset (instances 16 through 20).

In order to further study the execution time for solving larger instances of the problem where the queries are from the entire view lattice, and for those instances in which queries are from top levels of the lattice, we constructed several instances with even larger numbers of queries on the 17-attribute dataset, as reported in Table 6.3. Note that for these instances we did not construct the corresponding model $OVIP1'$, since we have already observed that the model $OVIP2$ is much more effective than the model $OVIP1'$ for these types of instances.

In Table 6.3, we observe that for those instances where the queries are randomly chosen from the entire view lattice, we could solve the instances in which the number of queries is no more than 240. When we increased the number of queries to 260, CPLEX failed to obtain an optimal solution, as the computer ran out of memory. For the instances in which the queries are from the top levels of the view lattice, we could not solve the problem within 30 minutes when we increased the number of queries to 360. In the following chapters, we develop algorithms to further reduce the size of the model, which permits us to solve larger instances of the problem OVS.

Chapter 7

A criterion to evaluate a view

In this chapter we introduce a criterion to evaluate the effectiveness of each view with respect to a given collection of queries in the context of the view-selection problem. We refer to this criterion as the *cost-benefit ratio* for that view. Subsequently, we discuss various properties of the cost-benefit ratio of each view, and develop an efficient algorithm for calculating the *minimum cost-benefit ratio* associated with each view. In Chapter 8, we will use this measure to effectively reduce the search space of views, hence reducing the size of the problem instance and its corresponding IP model for the one-stage view-selection problem.

7.1 The cost-benefit ratio

Consider the one-stage view-selection problem *OVS* with its collection of queries Q. Recall that for each view v we use $Q(v)$ to denote the set of queries in Q that can be answered by v, i.e., $Q(v) = \{q \in Q : q \subseteq v\}$.

We define the "extra cost" of a view v with respect to a collection of queries Q', as follows.

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

$$c(v, Q') = \sum_{q \in Q'} \big(S(v) - S(q)\big) \tag{7.1}$$

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

As introduced in Section 4.2.2, for each view v and for each subset Q' of $Q(v)$, we have already defined $d(v, Q')$, the *benefit* of the view v with respect to the query set Q', as the amount of space saved by materializing the view v instead of materializing all the queries in Q'. That is, $d(v, Q') = \sum_{q \in Q'} S(q) - S(v)$. Note that if a view v is not equal to any query in Q, and if v is selected in an optimal solution for answering a query set Q', then the benefit of v with respect to Q' must be positive, that is, it must be that $d(v, Q') > 0$. (Otherwise, it is obviously feasible and less costly to answer these queries using their respective equivalent views.) For each view v in V, we refer to any subset Q' of $Q(v)$ with a positive benefit value, that is, with $d(v, Q') > 0$, as a *Positive Subset of* $Q(v)$.

We define a subset $\widetilde{V}$ of V by excluding from V: i) each view that is equal to some query in Q, and ii) each view v whose corresponding query set $Q(v)$ has no positive subsets, (i.e., each view v such that the benefit of v with respect to Q' is non-positive, for every subset Q' of $Q(v)$). In other words, we define $\widetilde{V}$ as follows:

$$\widetilde{V} = V \setminus Q \cup \{v \in V : d(v, Q') \leq 0, \forall Q' \subseteq Q(v)\}.$$

Recall that in Section 4.2.2 we have defined $d(v, Q(v))$, the *maximum benefit* of view v with respect to the query set Q, as the amount of space that we can save by materializing view v instead of materializing all the queries in Q that v can answer. Applying this definition, the set $\widetilde{V}$ could be rewritten as follows.

$$\widetilde{V} = \{v \in V : v \notin Q \text{ and } d(v, Q(v)) > 0\} \tag{7.2}$$

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

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

$$r(v, Q') = \frac{c(v, Q')}{d(v, Q')} = \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)} \tag{7.3}$$

The cost-benefit ratio of a view v with respect to a query set Q' measures the extra cost (i.e., the extra evaluation time) incurred per unit space that we save, when we materialize v instead of materializing all the queries in Q', and when we use v to answer the queries in Q'.

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

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

7.2 The minimum cost-benefit ratio

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

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

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

$$r_{min}(v, Q) = \min_{Q' \in \mathcal{PS}(Q(v))} r(v, Q') = \min_{Q' \in \mathcal{PS}(Q(v))} \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)} \tag{7.4}$$

Observation 13. *Given the input query set Q and the view set V in the problem OVS, for every view $v \in \widetilde{V}$ (that is, for each view v in V such that $v \notin Q$ and $d(v, Q(v)) > 0$), the minimum cost-benefit ratio $r_{min}(v, Q)$ exists, and $r_{min}(v, Q) \geq 0$.*

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

In order to determine the minimum cost-benefit ratio of each view v with respect to the query set Q, according to its definition in Equation (7.4), we need to determine the cost-benefit ratio for every positive subset Q' of $Q(v)$. The computational requirement of this work is $O(2^{|Q(v)|})$. The following discussion allows us to reduce this computational requirement significantly.

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

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

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

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

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

We made the following observation.

Proposition 1. *Given an input query set* Q *and the view set* V *in the problem OVS, for each view* v *in* $\widetilde{V}$ *we have that*

$$r_{min}(v, Q) = \min_{n_v \leq n \leq N_v} r(v, Q_{(n)}(v)) = \min_{n_v \leq n \leq N_v} \frac{\sum_{j=1}^{n} \big(S(v) - S(q_{(j)})\big)}{\sum_{j=1}^{n} S(q_{(j)}) - S(v)} \tag{7.5}$$

Proof. By Observation 13, there exists a query set $Q' \in \mathcal{PS}(Q(v))$, such that $r_{min}(v, Q) = r(v, Q')$. Let N be the number of queries in Q', that is, $N = |Q'|$. The total size of the N queries in Q' is less than or equal to the total size of the queries in $Q_{(N)}(v)$ (the N largest queries in $Q(v)$). Equivalently, $\sum_{q \in Q'} S(q) \leq \sum_{j=1}^{N} S(q_{(j)})$. It follows that $\sum_{q \in Q'} \big(S(v) - S(q)\big) \geq \sum_{j=1}^{N} \big(S(v) - S(q_{(j)})\big) \geq 0$, and $0 < \sum_{q \in Q'} S(q) - S(v) \leq \sum_{j=1}^{N} S(q_{(j)}) - S(v)$. In other words, the "extra cost" of v with respect to Q' is greater than or equal to that of v with respect to $Q_{(N)}(v)$, and the "benefit" of v with respect to Q' is positive and no more than that of v with respect to $Q_{(N)}(v)$. Thus, we have that

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

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

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

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

By Proposition 1, the minimum cost-benefit ratio of v with respect to Q can be obtained by evaluating the cost-benefit ratio of v with respect to the query set $Q_{(n)}(v)$, for $n_v \leq n \leq N_v$. Hence, the computational requirement of evaluating the minimum cost-benefit ratio can be reduced from $O(2^{|Q(v)|})$ to $O\big(|Q(v)|\big)$. We can further improve the efficiency of this evaluation by the following observation.

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

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

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

Or, equivalently,

$$r(v, Q_{(n)}(v)) - r(v, Q_{(n+1)}(v)) = \frac{S(v)}{d(v, Q_{(n)}(v))d(v, Q_{(n+1)}(v))}T_n \tag{7.6}$$

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

By the same argument, we have that

$$r(v, Q_{(n+1)}(v)) - r(v, Q_{(n+2)}(v)) = \frac{S(v)}{d(v, Q_{(n+1)}(v))d(v, Q_{(n+2)}(v))}T_{n+1} \tag{7.7}$$

Since $r(v, Q_{(n)}(v)) < r(v, Q_{(n+1)}(v))$, we have that $r(v, Q_{(n)}(v)) - r(v, Q_{(n+1)}(v)) < 0$. By Equation (7.6), $T_n < 0$. It follows that $T_{n+1} = T_n - n\big(S(q_{(n+1)}) - S(q_{(n+2)})\big) < 0$. Hence, by Equation (7.7), we have that $r(v, Q_{(n+1)}(v)) - r(v, Q_{(n+2)}(v)) < 0$. □

Proposition 2 implies that once the function $r(v, Q_{(n)}(v))$ increases as we increase n, it will no longer decrease. Hence, we obtain the following corollary.

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

i) $r(v, Q_{(n)}(v))$ is a non-increasing function of n, for $n_v \leq n \leq N_v$, or

ii) $r(v, Q_{(n)}(v))$ is a non-decreasing function of n, for $n_v \leq n \leq N_v$, or

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

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

Example 8. In this example, we consider an instance of the problem *OVS* based on the same dataset as in Example 1, which was introduced in Section 3.1. The input query set Q consists of seven queries $Q = \big\{\{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{b, d\}, \{c, d\}\big\}$, which is the same as the set $\widehat{Q}$ in Example 1. The space limit $b = 30$. Our objective is to minimize the cost of answering the queries in the set Q by materializing a set of views S with the total size that is less than or equal to 30. By the results in Section 4.2.4, the reduced search space of views is $\overline{V} = \big\{\{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}, \{a, b, c\}, \{b, c, d\}, \{a, b, c, d\}\big\}$. Thus, $\widetilde{V} = \big\{\{a, b, c\}, \{b, c, d\}, \{a, b, c, d\}\big\}$. Consider the views $v_1 = \{a, b, c\}$ and $v_2 = \{b, c, d\}$ in the view set $\widetilde{V}$. We can compute $r_{min}(\{a, b, c\}, Q)$ as follows.

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

Thus, $r_{min}(\{a, b, c\}, Q) = 0.75$. We can apply the same approach to obtain that $r_{min}(\{b, c, d\}, Q) = 1.33$. The values of the function $r(v, Q_{(n)}(v))$ on different values of n for $v = v_1$ and for $v = v_2$ are shown in Figure 7.1. We observe that $r(v_1, Q_{(n)}(v_1))$ is an increasing function of n for $2 \leq n \leq 5$, and $r(v_2, Q_{(n)}(v_2))$ decreases for $2 \leq n \leq 3$, and increases for $3 \leq n \leq 5$. □

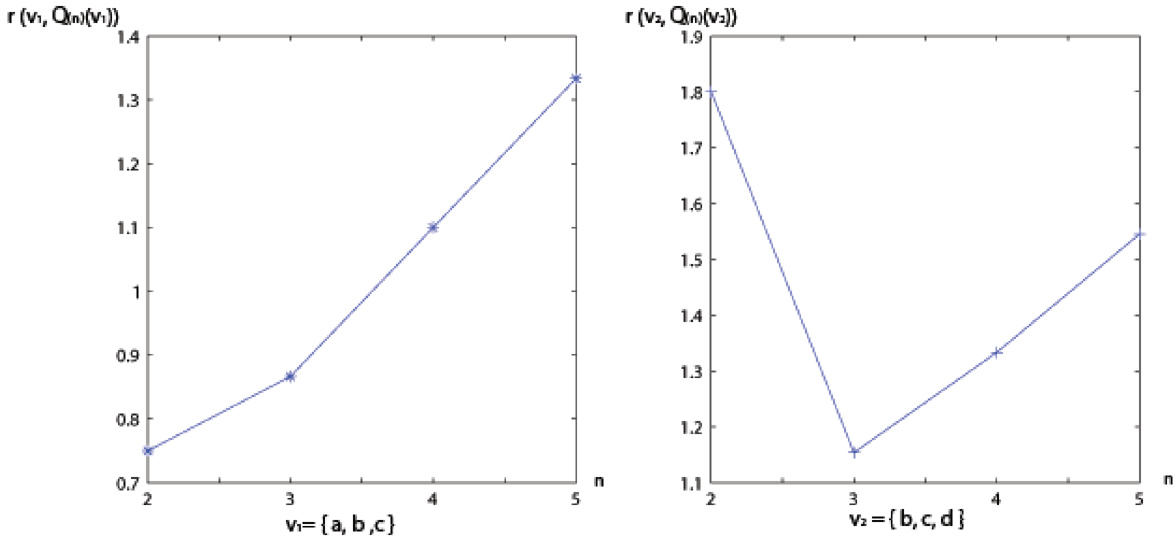

Figure 7.1: The minimum cost-benefit ratios for Example 8

In the next chapter we employ the concept of cost-benefit ratio to design and develop a heuristic procedure for solving the one-stage view-selection problem. Subsequently in Chapter 9 we employ this concept to design a heuristic procedure for the two-stage problem.

Chapter 8

Heuristic methods for the one-stage view-selection problem

In the context of the one-stage view-selection problem OVS, Observation 13 and Propositions 1 and 2 in Section 7.2 allow us to further reduce the size of the search space of views, by keeping only those views that are likely to be effective in answering the given collection of queries. In this chapter we propose several strategies for carrying out this task. These strategies lead to two distinct procedures that we refer to as heuristic methods I and II, respectively. We start the chapter by discussing the relationship between the cost-benefit ratio and the model $OVIP2$. Subsequently, we build on the discussion to derive two distinct heuristic methods for reducing the search space of views and the size of the corresponding IP model.

8.1 The cost-benefit ratio and the model $OVIP2$

In this subsection, we discuss the relationship between the cost-benefit ratio, which was introduced in Section 7.1, and the integer programming model $OVIP2$, which was introduced in Section 6.1. We also provide an intuitive justification as to why a view with a relatively low minimum cost-benefit ratio is potentially a good choice in view selection in the context of the problem OVS.

Given a query set Q and a view v, we denote by $J(v)$ the set of subscripts for the queries in $Q(v)$. As introduced in Section 3.1, if a view v_i can answer a query q_j, then the cost of answering q_j using v_i is the size of v_i. Equivalently, we have that $d_{ij} = a_i = S(v_i)$. We now show that the objective function of the model $OVIP2$ can be interpreted as the sum of the *extra costs* of answering the queries using the materialized views, instead of the queries themselves, as defined in Equation (7.1) in Section 7.1, plus a constant. To this end, recall that $\overline{I}$ and $\overline{I_j}$ are the sets of subscripts associated with the reduced search spaces of views $\overline{V}$ and $\overline{V_j}$, respectively. In the model $OVIP2$ of Section 6.1, we have also defined the binary variables z_{ij} for $j \in J$ and for $i \in \overline{I_j}$: $z_{ij} = 1$ if we use view v_i to answer query q_j ($z_{ij} = 0$, otherwise). Thus, we rewrite the objective function (6.1) of $OVIP2$ in Section 6.1 as follows.

$$\begin{aligned}
\sum_{j\in J}\sum_{i\in\overline{I_j}} d_{ij}z_{ij} &= \sum_{j\in J}\sum_{i\in\overline{I_j}} S(v_i)z_{ij} && (8.1)\\
&= \sum_{j\in J}\sum_{i\in\overline{I_j}} S(v_i)z_{ij} - \sum_{j\in J} S(q_j) + \sum_{j\in J} S(q_j) && (8.2)\\
&= \sum_{j\in J}\sum_{i\in\overline{I_j}} S(v_i)z_{ij} - \sum_{j\in J} S(q_j)\sum_{i\in\overline{I_j}} z_{ij} + \sum_{j\in J} S(q_j) && (8.3)\\
&= \sum_{j\in J}\sum_{i\in\overline{I_j}} z_{ij}\Big(S(v_i) - S(q_j)\Big) + \sum_{j\in J} S(q_j) && (8.4)\\
&= \sum_{i\in\overline{I}} x_i\bigg(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\bigg) + \sum_{j\in J} S(q_j) && (8.5)
\end{aligned}$$

To see why Equation (8.5) follows from Equation (8.4), note that for each view v_i, we define the *extra cost* of using the view v_i to answer a query $q_j \in Q(v_i)$ as $\Big(S(v_i) - S(q_j)\Big)$. Using the binary variables z_{ij}, it follows that $\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)$ is the extra cost of using the view v_i to answer all the queries that are assigned to it, and the term $\sum_{i\in\overline{I}} x_i\bigg(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\bigg)$ is the "total extra cost" of answering the queries using the materialized views, instead of using the queries themselves.

Similarly, we define the *benefit* of using a view v_i to answer a collection of queries $Q' \subseteq Q(v_i)$ as the amount of disk space that would be saved by materializing this view instead of materializing all the queries in Q'. It follows that in the context of the model $OVIP$, the benefit associated with view v_i is $\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, and the "total benefit" associated with a given solution is $\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$. Recall the storage-limit constraint (6.4) of model $OVIP2$ in Section 6.1, i.e., $\sum_{i\in\overline{I}} a_i x_i \leq b$. By expanding and rewriting the terms in the constraint (6.4), we obtain that

$$\begin{aligned}
b - \sum_{i\in\overline{I}} a_i x_i &= \sum_{j\in J} S(q_j) - \sum_{i\in\overline{I}} a_i x_i + b - \sum_{j\in J} S(q_j) && (8.6)\\
&= \sum_{j\in J} S(q_j)\sum_{i\in\overline{I_j}} x_i z_{ij} - \sum_{i\in\overline{I}} a_i x_i - \Big(\sum_{j\in J} S(q_j) - b\Big) && (8.7)\\
&= \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) - \Big(\sum_{j\in J} S(q_j) - b\Big) && (8.8)
\end{aligned}$$

It follows that the constraint (6.4) can be rewritten as

$$\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) \geq \sum_{j\in J} S(q_j) - b \tag{8.9}$$

Note that the left-hand side of the inequality (8.9) is the "total benefit" associated with a given solution in the context of the model $OVIP2$, while the right-hand side of (8.9) is a constant. In other words, the storage-limit constraint (6.4) could be interpreted as the "benefit constraint". That is, the "total benefit" obtained by materializing the views instead of materializing all the queries, i.e., $\sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, should be greater than or equal to a given constant $\sum_{j\in J} S(q_j) - b$.

We can now rewrite the model $OVIP2$ as follows.

$$\begin{aligned}
(OVIP2') \quad \min \quad & \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Big) + \sum_{j\in J} S(q_j) && (8.10)\\
s.t. \quad & \sum_{i\in\overline{I}} x_i\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big) \geq \sum_{j\in J} S(q_j) - b && (8.11)\\
& \sum_{i\in\overline{I_j}} z_{ij} = 1 \qquad \forall j \in J && (8.12)\\
& z_{ij} \leq x_i \qquad \forall j \in J, \forall i \in \overline{I_j} && (8.13)\\
& \textit{All the variables are binary} && (8.14)
\end{aligned}$$

Note that the structure of the model $OVIP2'$ is similar to the well-known 0-1 min-knapsack problem [37], where $\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Big)$ corresponds to the "value" of the i^{th} item in the knapsack problem, and $\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$ corresponds to the "weight" of the i^{th} item. Constraint (8.11) corresponds to the "knapsack constraint". Of course, in this case the presence of the variables z_{ij} and their relationship with the variables x_i (i.e., the constraints (8.12) and (8.13)) are distinct characteristics that distinguish this model from the conventional knapsack problem.

Dantzig in [13] proposed a greedy approximation algorithm to solve the knapsack problem. It first sorts the items in non-decreasing order of their "value" per unit of "weight". It then proceeds to select the items in that order, until the knapsack constraint is satisfied. In our problem OVS, the "cost" per unit of "weight" for each view is $\Big(\sum_{j\in J(v_i)} z_{ij}\Big(S(v_i) - S(q_j)\Big)\Big)/\Big(\sum_{j\in J(v_i)} z_{ij}S(q_j) - S(v_i)\Big)$, which is a function of z_{ij}, rather than a constant. For each solution of the problem OVS, the "cost" per unit of "weight" for each view v is equivalent to the cost-benefit ratio of v with respect to the query set that it answers in the solution. We can potentially use this property to devise a greedy algorithm for the problem OVS, by first sorting the views in non-decreasing order of their minimum cost-benefit ratios, and then by selecting the views in the order as needed.

Alternatively, we can employ this property to limit the search space of views in the context of the model $OVIP2$ that we introduced in Section 6.1, hence reducing the size of this IP model and improving its scalability. To this end, we can select only the views that have a relatively low minimum cost-benefit ratio with respect to the query set Q, and discard all other views. Of course, by doing so we can no longer guarantee that an optimal solution of the resulting IP model would also be optimal for the original view-selection problem. At the same time, we can potentially solve much larger instances of the problem OVS. In the next few subsections, we discuss several strategies for carrying out this task. In Section 8.4 we present the results of a computational experiment to evaluate the effectiveness of these strategies.

8.2 Heuristic method I: the single-threshold strategy

In this heuristic approach we start with the search space $\overline{V}$ of the model $OVIP2$, as introduced in Section 6.1. We than remove from this set every view whose minimum cost-benefit ratio is above a fixed input threshold γ, where γ is a positive real number. For practical reasons, we exempt from this process every view that is equal to one of the queries in the given query set Q. (We keep these views in the search space at all times.) The minimum cost-benefit ratio of each view in the set $\{v : v \in \overline{V}, v \notin Q\}$ can be obtained as discussed in Section 7.2. In order to guarantee feasibility, we also keep in the search space the view with the smallest number of attributes that can answer all the input queries. It is easy to show that this view, denoted by v_{max}, is the union of all the input queries, that is, $v_{max} = \cup_{q \in Q}\, q$. This results in a reduced search space, which we refer to as $\overline{V}^I(\gamma)$. We have

$$\overline{V}^I(\gamma) = \{v_{max}\} \cup \{v : v \in Q\} \cup \{v : v \in \overline{V}, v \notin Q, r_{min}(v, Q) \leq \gamma\}. \tag{8.15}$$

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

We now construct an integer programming model associated with the parameter γ. The model is similar to the model $OVIP2$, except that we use $\overline{V}^I(\gamma)$ (instead of $\overline{V}$) as the search space of views, and use the associated reduced view set $\overline{V}^I_j(\gamma)$ (instead of $\overline{V}_j$) for each query $q_j \in Q$. (We use the corresponding subscript sets $\overline{I}^I(\gamma)$ and $\overline{I}^I_j(\gamma)$ (instead of $\overline{I}$ and $\overline{I}_j$) in the model $OVIP2$, respectively.) We refer to this model as the *IP model with reduced search space of views associated with* γ, or $IPRv(\gamma)$ for short. We make the following observation.

Observation 14. *Given two positive real numbers* γ_1 *and* γ_2 *such that* $\gamma_1 > \gamma_2 > 0$, *we have that* $\overline{V}^I(\gamma_2) \subseteq \overline{V}^I(\gamma_1)$, *and* $\overline{V}^I_j(\gamma_2) \subseteq \overline{V}^I_j(\gamma_1)$, *for each query* $q_j \in Q$.

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

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

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

The proof of Corollary 2 is immediate from Observation 14.

For any finite threshold γ, we can no longer guarantee that an optimal solution of the associated model $IPRv(\gamma)$ is optimal for the original problem. At the same time, the model $IPRv(\gamma)$ is potentially smaller, and it is thus easier to solve. Obviously, the effectiveness of this strategy depends on the value of the parameter γ. Specifically, larger values of γ result in potentially larger sets $\overline{V}^I(\gamma)$. Thus, the corresponding IP model $IPRv(\gamma)$ would also be larger. Solving such a larger IP model is likely to result in higher execution time and memory requirements, but its optimal solution is potentially better (smaller). Indeed, at $\gamma = +\infty$, we have $\overline{V}^I(\gamma) = \overline{V}$, and the resulting model $IPRv(\gamma)$ would be identical to the model $OVIP2$. (Hence its optimal solution is guaranteed to be optimal for the original problem.) At the same time, smaller values of γ are more likely to result in smaller IP models. Thus the corresponding execution time and memory requirements may be more manageable for larger instances of the problem OVS. Of course, it is also true that smaller values of γ could result in removing potentially beneficial views from the search space, thus an optimal solution of the corresponding IP model might not be as effective.

We carry out a computational experiment to study the impact of the parameter γ on both the solvability of the corresponding model $IPRv(\gamma)$ and on the quality of its optimal solution, on an empirical basis. We report our observations in Section 8.4. Based on these observations, we also propose a strategy for selecting an appropriate value for γ in each instance.

8.3 Heuristic method II: the two-threshold strategy

Intuitively, this heuristic method is similar to method I discussed in Section 8.2, where we limit the search space of views to a "promising" subset of $\overline{V}$. At the same time, we now further reduce the size of the IP model by limiting the choice of the view-query relationships as well.

To do so, given the reduced search space of views $\overline{V}^I(\gamma)$ obtained by method I in Section 8.2, for each view v in $\overline{V}^I(\gamma)$ we remove every view-query relationship in which the corresponding cost-benefit ratio is above a given threshold θ. More specifically, given a second threshold θ, for each view v in $\overline{V}^I(\gamma)$, let $N_{v,\theta}$ be the largest integer such that the cost-benefit ratio of the view v with respect to the set of the $N_{v,\theta}$ largest input queries in $Q(v)$ is less than or equal to θ. (That is, $r(v, Q_{(N_{v,\theta})}) \leq \theta$.) Since the cost-benefit ratio $r(v, Q_{(n)}(v))$ is a unimodal function with respect to n, it follows that such a value of $N_{v,\theta}$ is well defined, and we have $r(v, Q_{(n)}(v)) > \theta$ for all $n > N_{v,\theta}$. For each view v, we now keep only the view-query relationships between the view v and the $N_{v,\theta}$ largest input queries that v can answer. All other view-query relationships for view v are discarded. We always choose the values of $\theta \geq \gamma > 0$ so as to guarantee that for each view v in $\overline{V}^I(\gamma)$, the collection of queries that result in the minimum ratio for that view remain in the view-query relationships that we keep. In addition, similarly to method I, in order to guarantee feasibility, we always keep all the choices of the view-query relationship that are associated with the view v_{max}. We build on this idea to develop our heuristic method II, and formulate for the problem OVS an integer programming model that we denote by $IPRvq(\gamma, \theta)$.

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

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

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

$$\overline{V}_j^{II}(\gamma, \theta) = \{v \in \overline{V}_j^I(\gamma) : q_j \in Q_\theta(v)\} \cup \{v_{max}\}. \tag{8.16}$$

Consider an illustration.

Example 8 (Continued). Recall Example 8 from Section 7.2 that $q_1 = \{a, b\}$, $q_2 = \{b, c\}$, $q_3 = \{c, d\}$, $q_4 = \{b\}$, $q_5 = \{a, c\}$, $q_6 = \{c\}$ and $q_7 = \{b, d\}$. Given $\gamma = 1.2$, we obtain that the views $v_1 = \{a, b, c\}$ and $v_2 = \{b, c, d\}$ are included in the reduced view set $\overline{V}^I(\gamma)$. If we set the threshold $\theta = 1.3$, then we obtain that $N_{v_1,\theta} = 4$ and $N_{v_2,\theta} = 3$. In other words, for the view v_1 we only keep the relationship between v_1 and the largest four queries in $Q(v_1)$. For the view v_2 we only keep the relationship between v_2 and the largest three queries in $Q(v_2)$. The choices of the relationships are shown in Figure 8.1. We also reduce the search space of views based on the choices of the view-query relationship associated with all the other views in $\overline{V}^I(\gamma)$, and we obtain the corresponding reduced view set for each query, as listed below.

$\overline{V}_1^{II}(\gamma, \theta) = \big\{\{a, b\}, \{a, b, c\}, \{a, b, c, d\}\big\}$

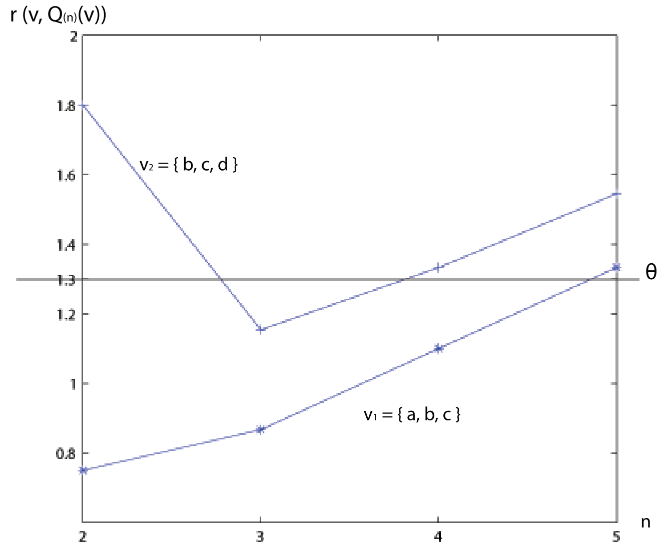

Figure 8.1: The choices of view-query relationships for Example 8 of Section 7.2

$\overline{V}_2^{II}(\gamma,\theta) = \{\{b,c\},\{a,b,c\},\{b,c,d\},\{a,b,c,d\}\}$
$\overline{V}_3^{II}(\gamma,\theta) = \{\{c,d\},\{b,c,d\},\{a,b,c,d\}\}.$
$\overline{V}_4^{II}(\gamma,\theta) = \{\{b\},\{a,b\},\{b,c\},\{b,d\},\{a,b,c,d\}\}$
$\overline{V}_5^{II}(\gamma,\theta) = \{\{a,c\},\{a,b,c\},\{a,b,c,d\}\}$
$\overline{V}_6^{II}(\gamma,\theta) = \{\{c\},\{b,c\},\{c,d\},\{a,b,c\},\{a,b,c,d\}\}$
$\overline{V}_7^{II}(\gamma,\theta) = \{\{b,d\},\{b,c,d\},\{a,b,c,d\}\}.$ □

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

Lemma 2. *Given two positive real numbers θ_1 and θ_2 such that $\theta_1 > \theta_2 > 0$, for each view v we have that $N_{v,\theta_1} \geq N_{v,\theta_2}$ and $Q_{\theta_2}(v) \subseteq Q_{\theta_1}(v)$.*

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

Observation 15.

i) Given two positive real numbers θ and γ such that $\theta \geq \gamma > 0$, for each query $q_j \in Q$ we have that $\overline{V}_j^{II}(\gamma, \theta) \subseteq \overline{V}_j^{I}(\gamma)$. If $\theta = +\infty$, then $\overline{V}_j^{II}(\gamma, \theta) = \overline{V}_j^{I}(\gamma)$.

ii) Given three positive real numbers θ_1, θ_2, and γ such that $\theta_1 > \theta_2 \geq \gamma > 0$, for each query $q_j \in Q$ we have that $\overline{V}_j^{II}(\gamma, \theta_2) \subseteq \overline{V}_j^{II}(\gamma, \theta_1)$.

iii) Given three positive real numbers θ_1, θ_2, and γ such that $\theta \geq \gamma_1 > \gamma_2 > 0$, for each query $q_j \in Q$ we have that $\overline{V}_j^{II}(\gamma_2, \theta) \subseteq \overline{V}_j^{II}(\gamma_1, \theta)$.

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

ii) If $v \in \overline{V}_j^{II}(\gamma, \theta_2)$ and $v \neq v_{max}$, then by the definition of $\overline{V}_j^{II}(\gamma, \theta)$ in Equation (8.16), we obtain that $v \in \overline{V}^{I}(\gamma)$ and that $q \in Q_{\theta_2}(v)$. By Lemma 2, we have $q \in Q_{\theta_1}(v)$. Thus, $v \in \overline{V}_j^{II}(\gamma, \theta_1)$. It follows that $\overline{V}_j^{II}(\gamma, \theta_2) \subseteq \overline{V}_j^{II}(\gamma, \theta_1)$.

iii) If $v \in \overline{V}_j^{II}(\gamma_2, \theta)$ and $v \neq v_{max}$, then by the definition of $\overline{V}_j^{II}(\gamma, \theta)$ in Equation (8.16), we obtain that $v \in \overline{V}^{I}(\gamma_2)$ and that $q \in Q_{\theta}(v)$. Since $\gamma_1 > \gamma_2 > 0$, by Observation 14, we have that $v \in \overline{V}^{I}(\gamma_2) \subseteq \overline{V}^{I}(\gamma_1)$. Thus, $v \in \overline{V}_j^{II}(\gamma_1, \theta)$. It follows that $\overline{V}_j^{II}(\gamma_2, \theta) \subseteq \overline{V}_j^{II}(\gamma_1, \theta)$. ◻

Corollary 3.

i) Given two positive real numbers θ and γ such that $\theta \geq \gamma > 0$, we have that $Objv\big(IPRv(\gamma)\big) \leq Objv\big(IPRvq(\gamma, \theta)\big)$. If $\theta = +\infty$, then the models $IPRv(\gamma)$ and $IPRvq(\gamma, \theta)$ are equivalent.

ii) Given three positive real numbers θ_1, θ_2, and γ such that $\theta_1 > \theta_2 \geq \gamma > 0$, we have that $Objv\big(IPRvq(\gamma, \theta_1)\big) \leq Objv\big(IPRvq(\gamma, \theta_2)\big)$.

iii) Given three positive real numbers θ_1, θ_2, and γ such that $\theta \geq \gamma_1 > \gamma_2 > 0$, we have that $Objv\big(IPRvq(\gamma_1, \theta)\big) \leq Objv\big(IPRvq(\gamma_2, \theta)\big)$.

Proof. i)-iii) follow directly from Observation 15. ◻

8.4 Experimental results

In this section, we present the results of a computational experiment with the heuristic methods proposed in Sections 8.2 and 8.3. We evaluate the performance of these methods by discussing the quality of the solutions and the efficiency of the reductions in the size of the corresponding IP models. The results show that our heuristic methods outperform the heuristic methods in [3,4] in terms of both the execution time and the quality of solutions obtained. More specifically, our experimental results show that:

- Size of the models obtained by the heuristic methods I and II are significantly smaller than that of $OVIP2$ (which was introduced in Section 6.1) for the same input, allowing us to obtain optimal or near optimal solutions for relatively large instances of the problem within reasonable time and memory limits.

- Magnitude of reduction in the size as well as in the execution time, and the corresponding quality of the resulting solutions, depend on the values of the parameters γ and θ, as well as on the specific characteristics of each instance.

- On average, the heuristic methods I and II require less execution time than the heuristic methods proposed in [3, 4], and the resulting solutions are significantly better (i.e., have lower costs).

8.4.1 Experiments with heuristic method I

In this subsection we evaluate the effectiveness of the single-threshold strategy of Section 8.2 in terms of its computational requirements, as well as of the quality of the solutions that it returns. We also study the impact of the threshold γ on the quality of the solutions obtained.

To examine the effect of the parameter γ on the performance of the model $IPRv(\gamma)$, for several instances of the problem OVS we have solved the model $IPRv(\gamma)$ with different values of γ. We evaluate the quality of the solutions with respect to different values of γ, by calculating the ratio of the difference between the optimal values of the models $IPRv(\gamma)$ and $OVIP2$ over the optimal value of $OVIP2$ (expressed as a percentage). We refer to this ratio as the *gap*. We have

$$gap = \frac{Objv(IPRv(\gamma)) - Objv(OVIP2)}{Objv(OVIP2)} \times 100\%,$$

where $Objv(\cdot)$ refers to the optimal value of the model $(\cdot)$. Note that if $\gamma = +\infty$, then the resulting model $IPRv(\infty)$ is identical to the model $OVIP2$. Hence, we have that $gap = 0$. Here we report our observations for the following instance of the problem. The pattern of observations for other instances is similar to this one.

Example 9. This example contains 125 queries constructed using the 17-attribute TPC-H dataset. The number of attributes of each query is a random number between 5 and 12. (That is, the query is randomly generated from the middle levels of the view lattice.) We continue to set the storage limit equal to one-fifth of the sum of the sizes of the queries. The description of this instance is presented in Appendix D. We have solved this instance optimally by the model $OVIP2$ (i.e., $IPRv(\infty)$). We also have solved it using the model $IPRv(\gamma)$ for $\gamma = 0.2, 0.3, \ldots,$ 1, and calculated the gap with respect to each value of γ. In Table 8.1, for each value of γ we present the number of views in the search space of the model $IPRv(\gamma)$, as well as the number

Table 8.1: The number of views and the size of the model $IPRv(\gamma)$ with different values of γ for Example 9

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

of variables and the number of constraints in each model. In Table 8.1 and in Figure 8.2 we present the gap with respect to each value of γ, as well as the total execution time of each model $IPRv(\gamma)$.

From Table 8.1 we observe that the number of views in the model $IPRv(\gamma)$ increases as we increase the value of γ from 0.2 to 1. The size of the corresponding model, expressed by the number of variables and constraints, is thus monotonically increasing as expected. But even for $\gamma = 1$, where the gap reduces to 0.0%, the size of the resulting model $IPRv(\gamma)$ is still significantly smaller than the size of the model $OVIP2$, or equivalently, of the model $IPRv(\infty)$.

From Figure 8.2 we observe that the value of the cost of answering the queries in this instance, as obtained via model $IPRv(\gamma)$ (as measured by the value of *gap*) is a non-increasing function of γ, as expected. It is also observed that the total execution time generally increases as we increase the threshold γ. The total execution time of solving this instance via the model $OVIP2$ is around 1.5 hours, which is far beyond the assumed time limit 30 min. However, for all $\gamma \leq 1$, the size of the model $IPRv(\gamma)$ is dramatically reduced, resulting in a significant reduction of the total execution time of the model. Notably, at the largest value of γ in this experiment ($\gamma = 1$), the execution time remains below 20 minutes. In addition, when the threshold γ equals 1.0, the gap is equal to 0, indicating that the optimal solution obtained by the respective model $IPRv(\gamma)$ is also an optimal solution for the original problem. Hence, heuristic method I can be a relatively effective technique for solving the problem OVS in terms of both the quality of the solutions and the efficiency of reductions. $\square$

To further evaluate the efficiency of the model $IPRv(\gamma)$, we solved the 12 instances with four

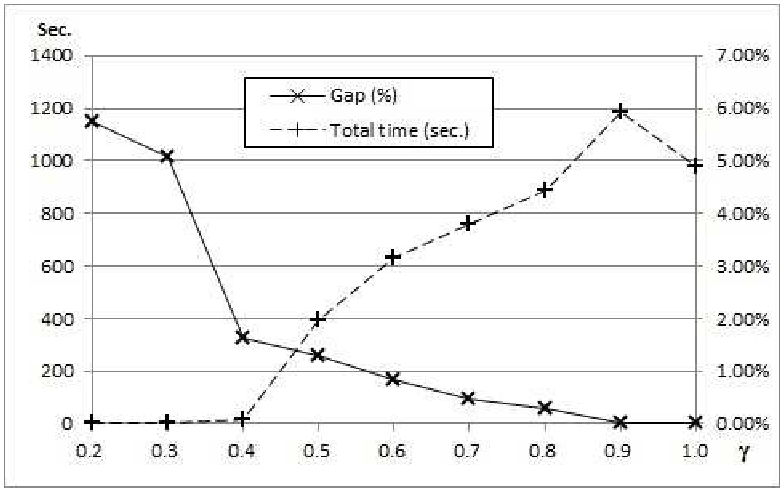

Figure 8.2: The execution time and the quality of solutions (gap) of the model $IPRv(\gamma)$ with different values of γ for Example 9

different query types that we presented in Section 6.2.1. As described earlier, these instances are defined using the 17-attribute TPC-H dataset using the model $IPRv(\gamma)$ for $\gamma = 1$. For each query type, we solved the three largest instances outlined in Section 6.2. (We present those instances in Appendix C.) For each instance we increased the value of the parameter γ in the model $IPRv(\gamma)$ from 0.1 to 0.2 to 0.3, ..., until the cost obtained by the model $IPRv(\gamma)$ would be within 0.2% of the optimal cost obtained by the model $OVIP2$. We refer to $\gamma_{0.2}$ as the smallest value of γ that has this property. For those instances that could not be solved by the model $OVIP2$ using CPLEX IP solver [21] within our time and memory limits, we used the best (largest) lower bound obtained via CPLEX instead of the corresponding optimal values, i.e., in this case we refer to $\gamma_{0.2}$ as the smallest value of γ such that the query-evaluation cost obtained by the model $IPRv(\gamma)$ is within 0.2% of the lower bound of the model $OVIP2$ obtained by CPLEX. For each instance we compare the number of views in the search space and the total execution time for the corresponding models $OVIP2$ and $IPRv(\gamma_{0.2})$. We report these results in Table 8.2.

From Table 8.2, we observe that the value of $\gamma_{0.2}$ is relatively small, ranging from 0.4 to 1.3. This confirms our intuition that the view with a relatively small value of the cost-benefit ratio is more likely to be selected for answering the queries. For each instance, the number of views in the search space of the model $IPRv(\gamma_{0.2})$ is significantly smaller than that for the model $OVIP2$. This results in a significant reduction of the total execution time of the model $IPRv(\gamma_{0.2})$ compared with the model $OVIP2$. In addition, for those instances that cannot be solved by the model $OVIP2$ due to memory shortage or time limitation (instances 3, 5, 6, 9, and 12), we can solve these instances using the model $IPRv(\gamma)$ with acceptable solution quality

Table 8.2: Comparison of the number of views in the search space, of the total execution time, and of the optimal value of the models $OVIP2$ and $IPRv(\gamma_{0.2})$

instance	query type	number of queries	$\gamma_{0.2}$	number of views		total time (sec.)	
				$OVIP2$	$IPRv(\gamma_{0.2})$	$OVIP2$	$IPRv(\gamma_{0.2})$
1	entire view lattice	220	0.9	23,256	1,774	317.0	80.0
2		240	0.8	18,783	1,450	388.5	65.1
3		260	1.0	25,922	3,947	out of memory	669.9
4	bottom-level	60	1.3	8,048	384	387.4	7.1
5		80	1.2	14,248	619	>30min	107.0
6		100	1.2	17,937	2,092	>30min	1538.3
7	middle-level	60	1.1	3,614	940	21.7	9.5
8		80	0.7	4,305	592	93.8	8.7
9		100	0.9	7,278	1,706	>30min	171.9
10	top-level	320	0.4	2,673	1,123	736.5	334.4
11		340	0.4	2,703	1,237	815.4	367.3
12		360	0.4	2,930	1,204	>30min	1662.5

Table 8.3: The impact of the space-limit parameter λ on the choice of γ for Example 9

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

(i.e., within 0.2% of the corresponding lower bound) within the time and memory limits. These results show that the model $IPRv(\gamma)$ allows us to solve larger instances of the problem OVS, and to obtain an optimal or near optimal solution more efficiently than the model $OVIP2$.

We now study the effect of the space limit and of the value of the parameter γ on the performance of the model $IPRv(\gamma)$. To this end, we have solved Example 9 under different storage-space limits, using the model $IPRv(\gamma)$.

Example 9 (Continued). We assume that the space limit is a fraction λ of the sum of the sizes of the input queries. We have solved this instance for $\lambda = 0.1, 0.2, \ldots, 0.6$. In Table 8.3, for each space limit we present the smallest value of γ such that the cost obtained by the resulting model $IPRv(\gamma)$ is optimal for the corresponding original problem.

As observed in Table 8.3, when we increase the space limit (or λ), the minimum value of γ, which still guarantees the optimality of the problem OVS, decreases. □

Generally speaking, when the space limit is relatively large, the optimal solution is more likely to comprise only the views with small values of the minimum cost-benefit ratio.

A strategy for using heuristic method I. In general, for solving the problem OVS using

Table 8.4: Comparison of the number of variables, the number of constraints, and the total execution time in the models $IPv(\gamma_{0.2})$ and $IPRvq(\gamma_{0.2}, \theta_{0.2})$

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

the model $IPRv(\gamma)$, it seems reasonable to increase the value of γ as much as possible, as long as the corresponding execution time is acceptable.

8.4.2 Experiments with heuristic method II

In this subsection, we evaluate the efficiency of our heuristic method II in solving the problem OVS, as we described in Section 8.3. We also examine the impact of the two parameters (γ, θ) in the model $IPRvq(\gamma, \theta)$ on the quality of the solutions output by the model.

To evaluate the efficiency of the heuristic method II, we solve the twelve instances of Table 8.2 using the model $IPRvq(\gamma, \theta)$. More specifically, for each instance we set $\gamma = \theta = \gamma_{0.2}$, where the value of $\gamma_{0.2}$ is presented in Table 8.2. Subsequently, we increase the value of the parameter θ from $\gamma_{0.2}$ to $(\gamma_{0.2}+0.1)$ to $(\gamma_{0.2}+0.2)$, ..., until the cost obtained by the model $IPRv(\gamma_{0.2}, \theta)$ is within 0.2% difference of the cost obtained by the model $OVIP2$. We refer to $\theta_{0.2}$ as the smallest value of θ that has such property. We compare the number of variables, the number of constraints, and the total execution time of the models $IPRv(\gamma_{0.2})$ and $IPRvq(\gamma_{0.2}, \theta_{0.2})$. The results are shown in Table 8.4.

From the results of Table 8.4 we observe that the value of $\theta_{0.2}$ is relatively close to the value of $\gamma_{0.2}$. For each instance in Table 8.4, the number of variables and the number of constraints in the model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ are smaller than those in the model $IPRv(\gamma_{0.2})$. This reduction is much more significant for the instances of "entire view lattice" and "bottom-level" queries

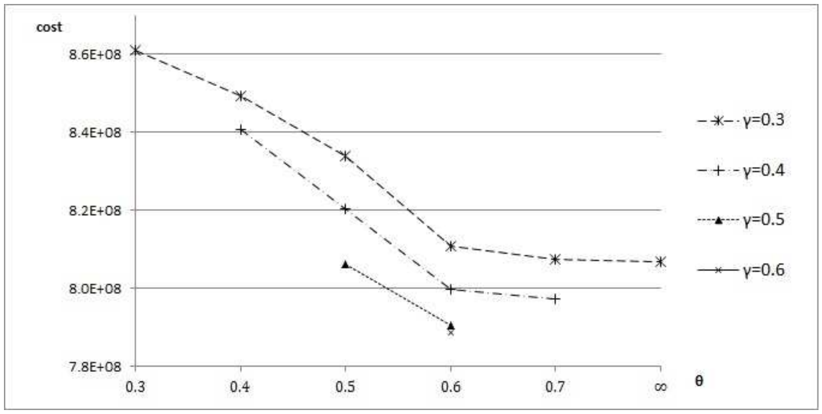

Figure 8.3: The query-evaluation cost obtained by the model $IPRvq(\gamma, \theta)$ with different values of γ and θ

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

To further examine the impact of our two parameters (γ, θ) on the effectiveness of the heuristic method II, we have solved the model $IPRvq(\gamma, \theta)$ with different values of γ and θ, for a number of instances. The instance that we report in Figure 8.3 is constructed using the 17-attribute TPC-H dataset. It contains 200 queries, which are randomly chosen from the middle levels of the view lattice. The number of attributes in each query ranges from 5 to 12. We set the storage limit as one-fifth of the sum of the sizes of the queries, as we did in Section 4.4. We solve this instance using the model $IPRvq(\gamma, \theta)$, for $\gamma = 0.3, 0.4, \ldots, 0.6$, and for $\theta = \gamma$, $\gamma + 0.1, \ldots, \gamma + 0.6$. To compare the quality of the solutions, we have also solved the instances using the model $IPRv(\gamma)$, or equivalently, the model $IPRvq(\gamma, \infty)$. The optimal value of each model is presented in Figure 8.3. Each line represents the results with respect to the same value of γ. All the missing points in this Figure indicate that we could not solve the corresponding model within our input-time and memory limits.

In Figure 8.3 we observe that for each fixed value of γ, the optimal cost obtained by the model $IPRvq(\gamma, \theta)$ monotonically decreases as we increase the value of θ. Similarly, this optimal cost is also a non-increasing function of γ when the value of θ is fixed. Both of these observations are consistent with our expectations. The lowest cost among the models for different values of the parameters in Figure 8.3 is achieved when $\gamma = \theta = 0.6$. When we attempted to solve this instance via our heuristic method I of Section 8.2, we were able to solve the model $IPRv(\gamma)$ only for $\gamma = 0.3$. For larger values of γ (i.e., $\gamma \geq 0.4$), we were not able to solve the corresponding IP model via CPLEX due to memory and time limitations. The cost obtained by the model $IPRvq(0.6, 0.6)$ is significantly smaller than the cost obtained by the model $IPRv(0.3)$.

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

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

To further evaluate our heuristic methods, in this section we carry out a computational experiment to compare the effectiveness of our heuristic methods, in terms of execution time and quality of solution, with the inexact method proposed in [3, 4] for solving the problem OVS. More specifically,

- We first compare the execution times, while requiring the same quality of solution for each method, for a collection of instances for which we can obtain the optimal value.
- Subsequently, we compare the quality of solution obtained, in the manner that obtains the lowest query-evaluation cost, by each method within the time and memory limits, for a second collection of instances, for which the optimal value cannot be obtained via the corresponding model $OVIP2$.

The results of [3, 4] show that their methods outperform other heuristic methods, including [2, 19, 22]. The basic idea of the inexact method proposed in [3, 4] is to limit the search space of views only to those views that are "relatively close" ancestors of the given queries. More specifically, Asgharzadeh Talebi et al. [3] define the set V_s as the reduced search space of views, where each view in V_s is the union of exactly 1, or 2, ..., or s queries in the given query set Q. We apply this view-reduction algorithm on the search space of views in the model $OVIP2$ that was introduced in Section 6.1. Given the parameter s, we refer to the resulting inexact model as the model $IPV(s)$.

We first compare the effectiveness of our models $IPRv(\gamma)$ and $IPRvq(\gamma, \theta)$ with the model $IPV(s)$ in terms of execution time for a collection of instances. This consists of all the instances

Table 8.5: Comparison of the total execution time in the models $IPRvq(\gamma_{0.2}, \theta_{0.2})$ and $IPV(s_{0.2})$

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

in Tables 8.2 and 8.4 for which we have the optimal value, via the corresponding model $OVIP2$ (i.e., the instances 1, 2, 4, 7, 8, 10, and 11 in Tables 8.2 and 8.4). We solve this collection of instances using the model $IPV(s)$. More specifically, for each instance we increase the value of s in the corresponding model $IPV(s)$ from 1 to 2 to 3, $\ldots$, until the cost obtained by the corresponding model $IPV(s)$ is within 0.2% of the optimal cost obtained by the model $OVIP2$. (Here we use the same requirement on the quality of the solution (0.2%) as for our heuristic methods in Sections 8.4.1 and 8.4.2.) In Table 8.5, we denote this value of s by $s_{0.2}$. We compare the execution time of the models $IPRvq(\gamma_{0.2}, \theta_{0.2})$ and $IPV(s_{0.2})$.

As observed in Table 8.5, for all the instances the total execution time of the model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ is significantly smaller than that of the model $IPV(s_{0.2})$. On average, the time to solve each instance using the model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ is 35.27% of the time to solve that instance using the model $IPV(s_{0.2})$. Our heuristic model $IPRvq(\gamma_{0.2}, \theta_{0.2})$ is more efficient in obtaining an optimal or near optimal solution than the model $IPV(s_{0.2})$ for the same instances.

To further compare the quality of solutions obtained by the heuristic methods proposed in this paper and in [3], we have constructed four instances using the 17-attribute TPC-H dataset. For these instances, the number of attributes of each query is a random number between 3 and 6. The number of queries in each instance ranges from 250 to 400. For each instance we set the storage limit to one-tenth of the sum of the sizes of the queries. The size of the corresponding model $OVIP2$ for each instance is relatively large, and thus the IP solver fails to obtain the optimal value via the corresponding model $OVIP2$. For each instance we compare the quality of the solutions of our methods with that of [3,4], by comparing the lowest cost obtained by these methods. More specifically, for the models $IPRv(\gamma)$ and $IPRvq(\gamma, \theta)$, we follow the strategy of choosing the parameters (γ, θ) as introduced at the end of Section 8.4.2. For each instance we denote by (γ_*, θ_*) the parameter values obtained, in the manner that permits us to obtain the

Table 8.6: Comparison of the cost value obtained from the models $IPRvq(\gamma_*, \theta_*)$ and $IPV(s_*)$

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

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

lowest query-evaluation cost, within the given time and memory limits. For the model $IPV(s)$ we increase the value of s in the model $IPV(s)$ from 1 to 2 to 3, ..., until s is the largest value for which the corresponding model $IPV(s)$ can provide a solution within the given time and memory limits. We denote this value of s by s_*. We compare the objective value of the models $IPRvq(\gamma_*, \theta_*)$ and $IPV(s_*)$, and in Table 8.6 we report the ratio of the cost obtained by the model $IPV(s_*)$ over the cost obtained by the model $IPRvq(\gamma_*, \theta_*)$.

From Table 8.6 we observe that for this collection of instances, the cost of query processing obtained by using the model $IPV(s)$ is significantly higher than the cost obtained by our model $IPRvq(\gamma, \theta)$. This indicates that our heuristic method outperforms the inexact model $IPV(s)$ of [3,4] for the collection of instances in Table 8.6 with respect to the quality of solution obtained.

We also compare the scalability of the two models. For the instances in Table 8.6, we failed to obtain a solution via the model $IPV(s)$ for $s \geq 2$ due to memory shortage. In general, since the number of views in the search space of the model $IPV(s)$ is of the order of $O(\sum_{i=1}^{s} \binom{|Q|}{i})$ (see [3,4]), the number of queries in the instance has a relatively large impact on the size of the model. The size of the model $IPV(s)$ grows significantly as we increase the value of s for the instances with large numbers of queries. For large-size instances, the model $IPV(s)$ is solvable only when $s = 1$, where the search space of views consists of only the input queries and of the view v_{max} (the view with the least number of attributes that can answer all the queries). This limits the quality of solutions obtained by the model $IPV(s)$ for those instances. On the other hand, the applicable values of the parameters (γ, θ) in our heuristic methods are independent of the size of the instances, which ensures high scalability of our heuristic methods. This allows us to choose appropriate values of the parameters for the model $IPRvq(\gamma, \theta)$ to obtain acceptable solution quality for relatively large instances of the problem OVS.

Chapter 9

Heuristic methods for the two-stage problem

In Chapter 8, we showed that in the context of the one-stage view-selection problem OVS, the strategies based on the cost-benefit ratio methods are effective and efficient for choosing promising views in evaluating the given query workload. In this chapter, we propose methods based on the cost-benefit ratio in the context of the two-stage stochastic view-selection problem SVS. We start this chapter by discussing the cost-benefit ratio of each view in the context of the reduced search space of views in each stage-2 subproblem, as well as for the stage-1 master problem in the model $IP2$, as discussed in Section 4.2.4. Subsequently, we propose a heuristic method and IP model to further improve the efficiency of solving the problem SVS. Finally, in Section 9.3 we conduct a computational experiment to evaluate the performance of this heuristic method.

9.1 The minimum cost-benefit ratio

As introduced in Section 4.2.4, given an instance of the problem SVS, we obtain three groups of reduced search spaces of view, i.e., $\overline{V_1}$ the search space of views materialized at stage 1, $\overline{V_{12}^{\ell}}$ the search space of views kept from stage 1 to stage 2 for the ℓ^{th} scenario, and $\overline{V_2^{\ell}}$ the search space of views materialized at stage 2 for the ℓ^{th} scenario.

First, we show the existence of the cost-benefit ratio of each view in each of the view sets from $\{\overline{V_1} \setminus \widehat{Q}, \overline{V_2^1} \setminus Q_2^1, \ldots, \overline{V_2^L} \setminus Q_2^L\}$ with respect to the corresponding relevant query set from $\{\widehat{Q}, Q_2^1, \ldots, Q_2^L\}$. Directly from Observations 5 and 6, for each view in the view set $\overline{V_1}$ with its relevant query set $\widehat{Q}$ (or the view set $\overline{V_2^{\ell}}$ with the query set Q_2^{ℓ}, for $\ell = 1$ to L), if v is not equal to any query in $\widehat{Q}$ (or in Q_2^{ℓ}, for $\ell = 1$ to L), then the benefit of v with respect to $\widehat{Q}$ (or to Q_2^{ℓ}, for $\ell = 1$ to L) must be positive. (Otherwise, it is feasible and less costly to answer these queries using their respective equivalent views.) It follows that the cost-benefit ratio of v with respect to $\widehat{Q}$ (or to Q_2^ℓ, for $\ell = 1$ to L) exists. It also indicates that for each view v in $\overline{V_1}$ (or in $\overline{V_2^\ell}$, for $\ell = 1$ to L) such that v is not a query in $\widehat{Q}$ (or in Q_2^ℓ, for $\ell = 1$ to L), the minimum cost-benefit ratio of v with respect to $\widehat{Q}$ (or to Q_2^ℓ, for $\ell = 1$ to L), denoted by $r_{min}(v, \widehat{Q})$ (or by $r_{min}(v, Q_2^\ell)$, for $\ell = 1$ to L), exists. Thus, we have:

$$r_{min}(v, \widehat{Q}) = \min_{Q' \in \mathcal{PS}(\widehat{Q}(v))} r(v, Q') = \min_{Q' \in \mathcal{PS}(\widehat{Q}(v))} \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)}, \quad \forall v \in \overline{V_1}, v \notin \widehat{Q} \tag{9.1}$$

$$r_{min}(v, Q_2^\ell) = \min_{Q' \in \mathcal{PS}(Q_2^\ell(v))} r(v, Q') = \min_{Q' \in \mathcal{PS}(Q_2^\ell(v))} \frac{\sum_{q \in Q'} \big(S(v) - S(q)\big)}{\sum_{q \in Q'} S(q) - S(v)}, \forall v \in \overline{V_2^\ell}, v \notin Q_2^\ell, \text{ for } \ell = 1 \text{ to } L \tag{9.2}$$

We can apply the strategy that introduced in Section 7.2, to obtain the minimum cost-benefit ratio of each view in polynomial time.

Example 1 (Continued). Recall Example 1, which was introduced in Section 3.1. We could calculate the cost-benefit ratio of each view in each of the view sets from $\{\overline{V_1} \setminus \widehat{Q}$, $\overline{V_2^1} \setminus Q_2^1$, $\overline{V_2^2} \setminus Q_2^2\}$. By the results in Section 4.2.4, we have that

$$\begin{aligned}
\overline{V_1} \setminus \widehat{Q} &= \big\{\{a,b,c\}, \{b,c,d\}, \{a,b,c,d\}\big\} \\
\overline{V_2^1} \setminus Q_2^1 &= \big\{\{a,b,c\}\big\} \\
\overline{V_2^2} \setminus Q_2^2 &= \emptyset
\end{aligned}$$

We can compute $r_{min}(\{a,b,c\}, \widehat{Q})$ as follows.

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

Thus, $r_{min}(\{a,b,c\}, \widehat{Q}) = 0.75$. Similarly, we have obtained that $r_{min}(\{b,c,d\}, \widehat{Q}) = 1.33$, $r_{min}(\{a,b,c,b\}), \widehat{Q}) = 3.76$, and $r_{min}(\{a,b,c\}, Q_2^1) = 3.33$. □

9.2 A heuristic method

We now introduce a heuristic method based on the minimum cost-benefit ratio of each view in the search space, in the context of the two-stage view-selection problem. In this heuristic approach, we start with the search spaces $\{\overline{V_1}$, $\overline{V_2^1}$, ..., $\overline{V_2^L}\}$ in the model $IP2$ that was introduced in Section 4.3. For each view set $\overline{V_1}$ (or $\overline{V_2^\ell}$, for $\ell = 1$ to L), we remove from this set every view whose minimum cost-benefit ratio is above a certain threshold γ_1 (or above γ_2^ℓ, for $\ell = 1$ to L). For practical reasons, we also exempt from this process every view that is equal to one of the queries in the corresponding relevant query set $\widehat{Q}$ (or in Q_2^ℓ, for $\ell = 1$ to L). (We keep these views in the corresponding search space at all times.) The minimum cost-benefit ratio of each view in the set $\{v : v \in \overline{V_1}, v \notin \widehat{Q}\}$ (or in $\{v : v \in \overline{V_2^\ell}, v \notin Q_2^\ell\}$, for $\ell = 1$ to L), can be obtained as discussed in Section 7.2. In order to guarantee feasibility, we also keep in each search space the view with the least number of attributes that can answer all the queries in the corresponding relevant query set. It is easy to show that this view, denoted by $v_{1,max}$ (or by $v_{2,max}^\ell$, for $\ell = 1$ to L), is the union of all the input queries in the corresponding relevant query set, that is, $v_{1,max} = \cup_{q \in \widehat{Q}}\, q$ (or $v_{2,max}^\ell = \cup_{q \in Q_2^\ell}\, q$, $\ell = 1$ to L). As a result, given a threshold vector $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2^1, \ldots, \gamma_2^L\}$, we obtain the reduced search spaces, denoted by $\overline{V_1}(\boldsymbol{\gamma})$ and $\overline{V_2^\ell}(\boldsymbol{\gamma})$ for $\ell = 1$ to L, as follows.

$$\overline{V_1}(\boldsymbol{\gamma}) = \{v_{1,max}\} \cup \{v : v \in \widehat{Q}\} \cup \{v : v \in \overline{V_1}, v \notin \widehat{Q}, r_{min}(v, \widehat{Q}) \leq \gamma_1\} \tag{9.3}$$

$$\overline{V_2^\ell}(\boldsymbol{\gamma}) = \{v_{2,max}^\ell\} \cup \{v : v \in Q_2^\ell\} \cup \{v : v \in \overline{V_2^\ell}, v \notin Q_2^\ell, r_{min}(v, Q_2^\ell) \leq \gamma_2^\ell\}, \text{ for } \ell = 1 \text{ to } L. \tag{9.4}$$

In addition, we define the new search space of views that are kept from stage 1 to stage 2 for the ℓ^{th} scenario as $\overline{V_{12}^\ell}(\boldsymbol{\gamma}) = \overline{V_1}(\boldsymbol{\gamma}) \cap V_2^\ell$.

Correspondingly, for each query $q_j \in Q_1$ we define $\overline{V_{1j}}(\boldsymbol{\gamma}) = \{v_i \in \overline{V_1}(\boldsymbol{\gamma}) : v_i \supseteq q_j\}$, and for each query $q_j \in Q_2^\ell$ we define $\overline{V_{12j}^\ell}(\boldsymbol{\gamma}) = \{v_i \in \overline{V_{12}^\ell}(\boldsymbol{\gamma}) : v_i \supseteq q_j\}$ and $\overline{V_{2j}^\ell}(\boldsymbol{\gamma}) = \{v_i \in \overline{V_2^\ell}(\boldsymbol{\gamma}) : v_i \supseteq q_j\}$, for $\ell = 1$ to L.

We now construct an integer programming model associated with the parameter vector $\boldsymbol{\gamma}$. This model is similar to the model $IP2$, except that we use as the new search spaces of views $\overline{V_1}(\boldsymbol{\gamma})$, $\overline{V_{12}^\ell}(\boldsymbol{\gamma})$, and $\overline{V_2^\ell}(\boldsymbol{\gamma})$, instead of $\overline{V_1}$, $\overline{V_2^\ell}$, and $\overline{V_2^\ell}$, respectively, and use the associated reduced view set $\overline{V_{1j}}(\boldsymbol{\gamma})$ instead of $\overline{V_{1j}}$ for each query $q_j \in \widehat{Q}$, and $\overline{V_{12j}^\ell}(\boldsymbol{\gamma})$ and $\overline{V_{2j}^\ell}(\boldsymbol{\gamma})$ instead of $\overline{V_{12j}^\ell}$ and $\overline{V_{2j}^\ell}$ for each query $q_j \in Q_2^\ell$, respectively. We refer to this model as the *IP model with reduced search space of views associated with* $\boldsymbol{\gamma}$, or $IPR2(\boldsymbol{\gamma})$ for short.

Various strategies can be employed to determine an appropriate set of values for the threshold parameters $\gamma_1, \gamma_2^1, \ldots, \gamma_2^L$. In our implementation of this heuristic method, we employ a simple strategy where we use the same value for all the threshold parameters, i.e., $\gamma_1 = \gamma_2^1 = \ldots = \gamma_2^L = \gamma_0$. We denote the resulting integer programming model by $IPR2(\gamma_0)$, and examine its performance through a computational experiment in the following section.

9.3 Experimental results

In this section we present the results of our computational experiments with the heuristic method based on the cost-benefit ratio for solving the problem SVS presented in the previous section. Our objectives in these experiments are as follows:

- to compare the execution time of the proposed model $IPR2(\gamma_0)$ with that of the model $IP2$ of Section 4.3, and
- to examine the quality of the solutions obtained by the model $IPR2(\gamma_0)$.

To this end, we constructed a number of instances of the problem SVS in the same way as we did in Section 4.4. These instances are based on the 7-attribute, 13-attribute, or 17-attribute TPC-H dataset. To examine the effect of the parameter γ_0 on the performance of the model $IPR2(\gamma_0)$, for each instance of the problem SVS we solve the model $IPR2(\gamma_0)$ with different values of γ_0. Note that when $\gamma_0 = +\infty$, the resulting model $IPR2(\infty)$ is identical to the model $IP2$. Here we report our observations for the following example.

Example 10. This example is based on the 13-attribute TPC-H dataset. The number of queries in the query sets Q_1, Q_2^1, and Q_2^2 are 200, 201 and 205, respectively. The number of attributes of each query is a random number between 5 and 7, (that is, the query is randomly generated from the middle levels of the view lattice). We continue to set the storage limit b_1 as one-fifth of the sum of the sizes of all the queries, and set the storage limit b_2 to half of the total space limit b_1. We present the details of this instance in Appendix E. We first tried to solve this instance using the model $IP2$ (i.e., the model $IPR2(\infty)$). However, we could not obtain an optimal solution of the model $IP2$ within the assumed time limit of 30 minutes, due to memory shortage. This was primarily due to the slow convergence of the Branch and Bound algorithm [37]. In order to resolve this difficulty, we set a relative tolerance 0.2% on the gap between the best integer objective value and the objective value of the best node remaining in the branch-and-bound tree obtained by the CPLEX IP solver [21]. It allows us to obtain a solution that is within 0.2% of the optimal solution of the original problem. It turned out that with this change in the parameter values, CPLEX was able to solve the model $IP2$ (i.e., $IPR2(\infty)$). Subsequently, we solved the model $IPR2(\gamma_0)$ for other values of γ_0, using the same relative tolerance 0.2%. We report the results in Table 9.1 and Figure 9.1.

In Table 9.1, for each value of γ_0 we present the number of views in various search spaces of the model $IPR2(\gamma_0)$, i.e., the number of the first-stage and second-stage variables, that is, $|\overline{V_1}(\gamma_0)|$ (the number of x_i), $|\overline{V_{12}^1}(\gamma_0)|$ (the number of u_i^1), $|\overline{V_2^1}(\gamma_0)|$ (the number of y_i^1), $|\overline{V_{12}^2}(\gamma_0)|$ (the number of u_i^2), and $|\overline{V_2^2}(\gamma_0)|$ (the number of y_i^2). We evaluate the quality of the solutions with respect to different values of γ_0, by calculating the *gap* as the ratio of difference between

Table 9.1: The number of views in the search spaces in the model $IPR2(\gamma_0)$ with different values of γ_0 for Example 10

Threshold γ_0	number of x_i $\lvert\overline{V_1}(\gamma_0)\rvert$	number of u_i^1 $\lvert\overline{V_{12}^1}(\gamma_0)\rvert$	number of y_i^1 $\lvert\overline{V_2^1}(\gamma_0)\rvert$	number of u_i^2 $\lvert\overline{V_{12}^2}(\gamma_0)\rvert$	number of y_i^2 $\lvert\overline{V_2^2}(\gamma_0)\rvert$	total execution time (sec.)	gap (%)
0.1	584	342	201	335	205	2.11	7.16%
0.2	609	367	201	360	206	2.52	4.11%
0.3	1031	771	252	757	246	74.92	0.10%
0.4	2346	2036	400	2011	425	607.18	0.05%
0.5	3592	3163	871	3124	853	1785.98	0.02%
∞	4565	3942	2853	3939	2829	4924.06	0.00%

the expected cost of answering the queries by the model $IPR2(\gamma_0)$ (with tolerance 0.2%) and the expected cost obtained by $IP2$ (with tolerance 0.2%), over the expected cost obtained by $IP2$ (with tolerance 0.2%), expressed as a percentage. In Figure 9.1 we present these results, as well as the total execution time of each model $IPR2(\gamma_0)$.

The size of various search spaces of views in $IPR2(\gamma_0)$ is significantly less than the corresponding size in the model $IP2$ (i.e., $IPR2(\infty)$) for all values of γ_0. In addition, those sizes become smaller as we reduce the value of γ_0. This reduction is especially significant when γ_0 is relatively small (i.e., close to 0), for example, when $\gamma = 0.1$. Hence, the size of the resulting model $IPR2(\gamma_0)$ is expected to be smaller than that of the model $IP2$, resulting in significant reductions of the total execution time of the model. This can be observed in Figure 9.1 and Table 9.1. It is also observed in Figure 9.1 and Table 9.1 that the total execution time generally increases as we increase the value of the threshold γ_0. The total execution time of solving this instance via the model $IP2$ (with tolerance 0.2%) is around 1.4 hours, which is far beyond the assumed time limit of 30 minutes. However, for all $\gamma_0 \leq 0.5$, the total execution time of solving the model $IPR2(\gamma_0)$ (with tolerance 0.2%) is dramatically reduced. Notably, at $\gamma_0 = 0.1$ the corresponding execution time is only 2.11 seconds, and at the largest value of γ_0 in this experiment ($\gamma_0 = 0.5$) the execution time remains below 30 minutes.

From Figure 9.1 we also observe that the value of the relative cost of answering the queries in this instance via the model $IPR2(\gamma_0)$ (as measured by the value of *gap*) is a non-increasing function of γ_0, as expected. In addition, when the value of the threshold γ_0 is at least 0.3, the gap is less than 0.2%, indicating that the solution obtained by the respective model $IPR2(\gamma_0)$ is relatively close to the optimal solution for the original problem. Furthermore, when $\gamma = 0.3$, we could obtain within 1.5 minutes a solution that is close to the optimum. $\square$

We have also solved some other instances of the problem *SVS* using our heuristic method. The pattern of observations for other instances is similar to Example 10. In general, our heuristic

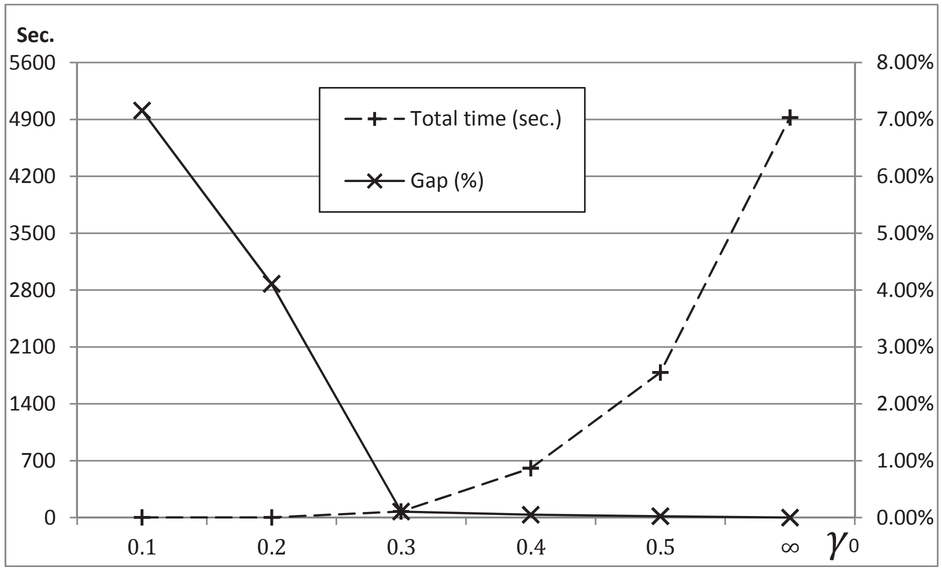

Figure 9.1: The execution time and the quality of solutions (gap) of the model $IPR2(\gamma_0)$ with different values of γ_0 for Example 10

method can be a relatively effective technique in solving the problem SVS, in terms of both the quality of solution and efficiency of reduction, when the size of the problem is relatively large.

Chapter 10

Concluding remarks

In this chapter we present a summary of our contributions and outline directions for future research.

10.1 Contributions

In this dissertation we undertook a systematic study of the view-selection problem under space-limit constraint in a two-stage stochastic environment. We developed exact and inexact methods for solving this problem. More specifically, our contributions in this context are as follows:

- We presented a formulation of the view-selection problem in a two-stage stochastic setting, that is, we presented a formal definition for the two-stage stochastic view-selection problem *SVS*.
- We introduced a stochastic programming model for the problem *SVS*, and provided an equivalent integer programming model.
- We proposed algorithms to efficiently prune the search space of potentially beneficial views for this problem, and provided a formal proof that our algorithms guarantee at least one globally optimal solution remains in the reduced search space. Using the resulting model, we can solve the problem *SVS* optimally for a number of realistic-size instances of the problem.
- We developed a heuristic algorithm to further reduce the search space of views for this problem by removing those views that are less "valuable" in the context of the given collection of queries. Of course, we could no longer guarantee that solving the resulting IP model would lead to finding an optimal solution for the original problem *SVS*. However, through a computational study we showed that using this approach we are able to find

near optimal solutions for larger instances of the problem within a reasonable execution time.

- We developed several related models in the context of the two-stage stochastic view-selection problem, and used these models to evaluate our proposed stochastic programming model. In particular we defined the value of the two-stage model versus the one-stage model (*VTVO*), the value of stochastic solution (*VSS*), and the expected value of perfect information (*EVPI*) in this context. We determined these values for several instances of the problem *SVS*.

While studying the two-stage stochastic view-selection problem, we also made several observations regarding the structure of the corresponding one-stage deterministic view-selection problem. Those observations allowed us to design effective algorithms for solving the one-stage problem. More specifically, we made the following contributions in the context of the one-stage problem:

- We proposed a new technique to prune the search space of views for the one-stage problem while guaranteeing that at least one globally optimal solution remains in the reduced search space. This, in turn, leads to a smaller corresponding IP model, and hence smaller execution time than comparable approaches in the open literature, such as those in [3, 4].

- We proposed a criterion, namely, the "cost-benefit ratio", to measure the relative "value" of each view with respect to a given collection of queries. We then used this measure to further prune the search space of views for larger instances of the problem in a heuristic manner, albeit we can no longer guarantee that the reduced search space contains a globally optimal solution. We carried out a computational experiment and showed that, on an empirical basis, the resulting heuristic procedures are more effective than comparable methods in the open literature, such as those in [3, 4]. The notion of "cost-benefit ratio" that we developed for the one-stage problem helped us design the heuristic procedure for the two-stage problem that we mentioned above.

10.2 Future research

The research we carried out in this dissertation has a wide array of potential applications and possible extensions in the future work. We outline three immediate directions we have envisioned to extend our research. One direction of research pertains to methodology development for solving the two-stage view-selection problem *SVS* based on an "adjusted" cost-benefit ratio. The second direction is to further investigate the impact of the dataset on the solution of the problem. The last but not least, a possible direction is to consider the index selection, both on its own and alongside view selection, in a two-stage stochastic environment.

10.2.1 Adjusting the "cost-benefit ratio" for the two-stage problem

The relationship between the cost-benefit ratio and the one-stage model $OVIP2$, which we presented in Section 8.1, does not hold between the cost-benefit ratio and the two-stage model $IP2$. Note that in the objective function of the two-stage model $IP2$, the cost with respect to a second-stage query is the cost of answering this query times the probability associated with the second-stage query set in which the query lies, i.e., it is a "weighted" evaluation cost. As a result, the "extra cost" of a view with respect to a second-stage query is the size difference between the view and the query times the probability associated with the query set in which the query lies, i.e., it is a "weighted" extra cost. As a result, in the context of the two-stage problem SVS, we may need to adjust the "cost-benefit ratio" of a view with respect to a set of queries by applying the "weighted" extra cost. A topic for future research could be to investigate the relationship between the "adjusted" cost-benefit ratio and the model $IP2$, and develop a heuristic method by applying its results.

10.2.2 Impact of the datasets

Through all experiments that we carried out in this dissertation, we observed that the structure of the datasets and the input query workload can make a significant impact on the solutions in the context of both the one-stage and the two-stage view-selection problems. A possible future work is to further investigate the impact of various datasets on the performance of each algorithm, and perhaps to fine-tune these algorithms by taking advantage of the structure of the dataset.

10.2.3 Index (and view) selection in the two-stage stochastic setting

Indexes, as another type of derived data, can also be beneficial for accelerating query processing. As we reviewed in Chapter 2, significant work including [1, 2, 5, 7, 8, 11] have been done on index selection in the one-stage setting, either on its own or alongside the corresponding view selection. Thus, a possible future research is to consider the index-selection problem in the two-stage stochastic setting, both on its own and alongside view selection.

REFERENCES

[1] Sanjay Agrawal, Nicolas Bruno, Surajit Chaudhuri, and Vivek R. Narasayya. AutoAdmin: Self-tuning database systems technology. *IEEE Data Eng. Bull.*, 29(3):7–15, 2006.

[2] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *VLDB*, pages 496–505, 2000.

[3] Zohreh Talebi Asgharzadeh. *Exact and inexact methods for solving the view and index selection problem for OLAP performance improvement.* Ph.d. dissertation, North Carolina State University, USA, 2010.

[4] Zohreh Talebi Asgharzadeh, Rada Chirkova, and Yahya Fathi. Exact and inexact methods for solving the problem of view selection for aggregate queries. *International Journal of Business Intelligence and Data Mining*, 4(3/4):391–415, 2009.

[5] Zohreh Talebi Asgharzadeh, Rada Chirkova, Yahya Fathi, and Matthias Stallmann. Exact and inexact methods for selecting views and indexes for OLAP performance improvement. In *EDBT*, pages 311–322, 2008.

[6] John R. Birge and François Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.

[7] Nicolas Bruno and Surajit Chaudhuri. Interactive physical design tuning. In *ICDE*, pages 1161–1164, 2010.

[8] Nicolas Bruno, Surajit Chaudhuri, and Gerhard Weikum. Database tuning using online algorithms. In *Encyclopedia of Database Systems*, pages 741–744. Springer, 2009.

[9] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.

[10] Surajit Chaudhuri, Umeshwar Dayal, and Vivek R. Narasayya. An overview of business intelligence technology. *Commun. ACM*, 54(8):88–98, 2011.

[11] Surajit Chaudhuri, Vivek R. Narasayya, and Gerhard Weikum. Database tuning using combinatorial search. In *Encyclopedia of Database Systems*, pages 738–741. Springer, 2009.

[12] Rada Chirkova and Jun Yang. Materialized views. *Foundations and Trends in Databases*, 4(4):295–405, 2012.

[13] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–288, 1957.

[14] Songyun Duan, Peter Franklin, Vamsidhar Thummala, Dongdong Zhao, and Shivnath Babu. Shaman: A self-healing database system. In *ICDE*, 2009.

[15] Matteo Golfarelli, Vittorio Maniezzo, and Stefano Rizzi. Materialization of fragmented views in multidimensional databases. *Data Knowl. Eng.*, 49(3):325–351, 2004.

[16] Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Index selection for OLAP. In *ICDE*, pages 208–219, 1997.

[17] Himanshu Gupta and Inderpal Singh Mumick. Selection of views to materialize in a data warehouse. *IEEE Trans. Knowl. Data Eng.*, 17(1):24–43, 2005.

[18] Alon Halevy. Answering queries using views: A survey. *VLDB J.*, 10:270–294, 2001.

[19] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. In *the ACM SIGMOD Conference*, pages 205–216, 1996.

[20] Rong Huang, Rada Chirkova, and Yahya Fathi. Benchmark databases for testing big-data analytics in cloud environments. In *IBM Cloud Academy Conference 2012*, pages 291–303, 2012.

[21] ILOG. CPLEX 11.0 software, 2007. `http://www.ilog.com/products/cplex/`.

[22] Panos Kalnis, Nikos Mamoulis, and Dimitris Papadias. View selection using randomized search. *Data Knowl. Eng.*, 42(1):89–111, 2002.

[23] Howard J. Karloff and Milena Mihail. On the complexity of the view-selection problem. In *PODS*, pages 167–173, 1999.

[24] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: (second edition)*. Wiley Computer Publishing, 2002.

[25] Yannis Kotidis and Nick Roussopoulos. Dynamat: A dynamic view management system for data warehouses. In *SIGMOD Conference*, pages 371–382, 1999.

[26] Yannis Kotidis and Nick Roussopoulos. A case for dynamic view management. *ACM Trans. Database Syst.*, 26(4):388–423, 2001.

[27] Michael Lawrence and Andrew Rau-Chaplin. Dynamic view selection for OLAP. In *DaWaK*, pages 33–44, 2006.

[28] Sam Lightstone. Physical database design for relational databases. In *Encyclopedia of Database Systems*, pages 2108–2114. Springer, 2009.

[29] Albert Madansky. Inequalities for stochastic linear programming problems. *Management Science*, 6(2):197–204, 1960.

[30] Thomas P. Nadeau and Toby J. Teorey. Achieving scalability in OLAP materialized view selection. In *DOLAP*, pages 28–34, 2002.

[31] Stratos Papadomanolakis and Anastassia Ailamaki. An integer linear programming approach to database design. In *ICDE Workshops*, pages 442–449, 2007.

[32] Amit Shukla, Prasad Deshpande, and Jeffrey F. Naughton. Materialized view selection for multidimensional datasets. In *VLDB*, pages 488–499, 1998.

[33] Dimitri Theodoratos and Mokrane Bouzeghoub. A general framework for the view selection problem for data warehouse design and evolution. In *DOLAP*, pages 1–8, 2000.

[34] Dimitri Theodoratos, Theodore Dalamagas, Alkis Simitsis, and Manos Stavropoulos. A randomized approach for the incremental design of an evolving data warehouse. In *ER*, pages 325–338, 2001.

[35] Dimitri Theodoratos and Timos K. Sellis. Incremental design of a data warehouse. *J. Intell. Inf. Syst.*, 15(1):7–27, 2000.

[36] TPC-H Revision 2.1.0. TPC Benchmark H (Decision Support). `http://www.tpc.org/tpch/spec/tpch2.1.0.pdf`.

[37] L. A. Wolsey. *Integer Programming.* Wiley, 1998.

[38] Jian Yang, Kamalakar Karlapalem, and Qing Li. Algorithms for materialized view design in data warehousing environment. In *VLDB*, pages 136–145, 1997.

APPENDICES

Appendix A

Instances of the problem SVS in Section 4.4

In this chapter, we provide the description of the instances of the problem SVS that are discussed in Section 4.4. The instances constructed using the 7-attribute TPC-H dataset and the 13-attribute TPC-H dataset are presented in Tables A.1 and A.2, respectively. For each instance we report the subscript number of each query in the query sets Q_1, Q_2^1, and Q_2^2. We also present the space limits b_1 (i.e., the total space limit) and b_2 (which limits the total size of the views to be materialized in stage two) for each instance.

Table A.1: Description of eight instances on the 7-attribute TPC-H dataset discussed in Section 4.4

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
1	99 24 4 59 70 54 119 6 16 71 17 116 111 32 115 103 126 109 82 64	3 18 23 54 93 44 12 51 125 29 63 91 112 80 95 103 119 97 52 33	32 27 15 116 3 67 93 8 63 72 95 71 113 35 119 120 17 37 54 28	14013254	7006627
2	25 66 64 120 43 125 119 31 16 115 90 126 6 57 67 91 93 1 47 22	27 126 1 33 3 123 2 5 18 56 115 64 111 8 108 110 12 119 20 23 93 81 96 59 40	8 102 56 100 121 59 55 36 79 81 112 95 71 12 62 21 1 61 87 68	14227050	7113525

Table A.1: (continued)

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
3	117 76 2 18 8 80 15 57 106 49 14 78 35 97 21 69 125 95 107 123 119 65 43 32 63 1 16 41 126 56	123 108 64 40 110 32 43 103 6 4 25 72 8 10 106 37 88 23 80 45 41 119 68 114 87 121 50 17 63 1 84 112 95 104 61	32 24 51 20 34 1 126 68 119 16 25 95 111 74 19 44 18 10 63 66 110 93 86 47 124 73 81 125 13	18855133	9427566
4	36 80 4 106 104 102 117 7 68 95 91 54 123 27 8 2 125 86 63 50 126 65 44 94 57 64 10 115 119 49	3 8 118 47 64 86 119 67 117 79 111 40 121 89 90 1 97 16 75 15 106 58 88 114 95 72	67 17 16 4 21 55 48 95 63 24 80 68 41 83 22 49 126 8 26 23 51 90 59 47 2 3 105 56 32 94 125	18181695	9090847
5	108 115 43 111 1 119 81 116 15 65 32 61 125 95 60 123 104 35 113 39 79 63 42 16 59 54 78 90 126 68 45 4 24 7 9 76 49 97 69 17	63 47 45 84 89 80 14 53 33 64 119 69 32 96 120 122 48 18 104 24 94 62 23 77 126 72 73 4 81 38 10 27 21 90 124 2 65 108 25 12 123 117 92 103 30	30 126 119 109 23 104 9 47 11 8 45 58 32 63 62 5 101 96 74 50 113 31 68 39 80 106 12 60 111 125 16 123 100 7 2 83 26 81 53 82 116	25934476	12967238
6	54 5 34 37 70 20 10 109 124 107 105 98 81 93 120 31 86 125 2 28 18 1 44 64 118 110 122 63 52 82 97 30 6 16 59 40 50 111 53 117	48 126 64 11 17 103 111 24 1 95 72 63 58 42 101 37 88 122 84 96 47 112 15 61 22 116 89 38 107 78 91 54 92 87 119 2 16 28	95 123 96 122 24 61 77 16 126 111 47 52 60 20 5 79 62 108 80 57 8 36 53 4 78 119 125 45 23 59 55 63 26 56 48	22332094	11166047

Table A.1: (continued)

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
7	25 95 64 16 77 119 27 30 47 12 37 111 115 1 48 61 126 104 2 52 19 94 15 20 57 23 80 63 103 51 123 78 96 10 106 101 33 74 91 120 110 34 125 32 68 11 116 112 6 8	63 90 35 57 123 1 119 30 36 104 124 111 68 12 76 79 89 125 47 19 16 31 14 2 65 3 88 80 95 33 52 4 94 126 28 106 34 87 44 23 107 37 15 49 9 66 58 7	119 44 64 46 61 54 109 22 124 18 68 95 89 50 93 66 41 42 8 32 111 106 38 4 26 65 117 1 125 123 77 72 34 2 94 122 120 115 25 92 17 29 82 96 49 74 51 63 103 40	27998417	13999208
8	107 105 126 113 83 74 53 40 2 16 4 86 119 15 62 79 38 5 49 1 18 64 9 103 35 125 87 106 48 25 55 118 37 21 123 104 95 32 31 10 63 13 111 3 69 22 70 26 72 73	69 107 86 58 125 119 4 55 62 3 31 8 36 30 73 120 89 61 124 111 32 22 49 2 67 29 64 37 80 9 114 74 63 95 11 116 47 87 18 94 40 126 79 97 85 90 48	68 93 40 66 18 118 3 111 103 121 10 114 126 110 16 24 23 95 80 61 59 62 2 37 31 12 33 87 94 115 125 4 63 79 96 60 69 26 20 47 34 105 17 64 6 8 57 88 29 50 122 119 38	30000454	15000227

Table A.2: Description of eight instances on the 13-attribute TPC-H dataset discussed in Section 4.4

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
1	4089 808 2876 6910 4200 3903 7935 5503 2068 6126 768 5 46 1591 1089 3201 4810 7164 17 6865	4089 808 2876 6910 4200 3903 7935 5503 2068 6126 768 5 46 1591 1089 3201 4810 7164 17 6865 5177	4132 5249 3311 2082 9 128 452 8155 512 2047 3139 16 8059 320 7161 1046 7583 7170 34 8062 1319 4384 32 772 8127 8063	16896857	8448428
2	1469 658 8190 8093 6699 7053 5503 552 784 7935 1317 453 8171 1058 6863 6880 4087 3080 5583 7113	4136 3552 7935 3837 6137 8031 8127 4112 5652 512 6657 7350 6125 5179 1037 7647 6143 516 160 8063 6106	7895 72 579 1204 8061 538 1472 4096 2576 2848 4736 3047 2257 8159 2340 5866 4519 280 6870 8190 8189 8054 4085	26528611	13264305
3	6143 16 66 1533 8190 2048 2049 3682 3805 128 8189 2421 4 2724 4098 7168 6274 4162 796 2115 4096 1959 521 64 5345 4113 3076 2064 6997 5483	5135 4608 7809 356 6111 16 7931 494 2048 6872 2548 768 2613 1334 32 5214 7360 3711 8183 3869 34 6212 1088 1 7733 4085	4 8190 2112 1487 4910 8051 2472 3839 987 320 8183 7007 4079 1040 7576 7808 64 40 7433 544 3577 3053	25260233	12630116

Table A.2: (continued)

inst-ance	Q_1	Q_2^1	Q_2^2	b_1	b_2
4	6467 2406 4808 4224 8022 1962 8145 8126 1024 6239 600 6143 5887 5097 6079 2292 2760 4092 34 8183 32 8063 4026 7163 4906 4789 2478 162 8159 2461	8056 6847 71 524 288 2064 7167 6942 4868 7598 2048 114 7641 3519 8175 5017 6142 3 5779 2484 1535 6 256 2038 7807 47 2497 5158 1 8034 7331 7276 2386 64 8189 8127	64 3618 857 2 272 6061 5140 7167 5596 819 1 903 7612 1370 8178 7039 133 5185 2064 8183 7679 3011 8118 8143 6388 2998 749 7487 7678 4086 862 2152 3967	42895557	21447778
5	5101 1678 4975 4232 256 832 32 7698 7887 258 7193 2252 4230 5919 592 2048 7270 7743 8031 4070 5501 4214 6111 2532 7567 1597 938 2664 7168 6183 6141 6861 7935 1122 128 4224 5122 8175 2042 4096	44 296 4699 1056 5129 4574 512 2047 5942 2708 2060 48 2068 2922 2053 6782 7096 152 2049 1962 6653 337 4973 6141 6075 2736 2 7935 6786 6213 6639 128 80 2209 4701 6208 85 5209 612 1665 7272 672 1537 4014 6143 1164 5882	147 4710 6798 6682 5844 32 5253 5161 402 1730 3216 2048 259 4097 6137 6992 4095 3724 1600 2733 7935 3559 1114 7242 136 4777 3838 6149 6091 591 8189 1157 8159 2065 2342 4096 4396 7167 34 1034 7929	45213104	22606552

Table A.2: (continued)

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
6	2103 7401 4368 5111 7668 7903 1152 3275 4162 74 767 4095 2032 2551 638 2672 3803 1344 3999 128 2289 8159 3433 512 5679 6369 208 7662 5943 7679 8175 1090 265 2079 7166 640 8158 64 1158 2328	256 4025 4095 96 8 5640 793 708 5652 8080 473 2013 579 10 7512 4 7936 5215 817 2509 5192 4096 8159 4224 2757 7115 520 6720 1032 8189 4678 8167 1669 805 6554 4781 5648 8027 2049 3146 525 1920 5951 3771 1651 7821 536 8166	80 7039 6750 1441 2200 7938 8126 836 6143 1786 4158 7165 2112 4077 6201 4260 2312 16 7943 49 4203 24 12 1745 8127 3987 258 4728 4844 7935 3203 7855 6546 3837 193 2263 1284 1713 36 8 7781 2048 7260 3478 5338 7915	50929728	25464864
7	7151 1057 8179 4079 4364 6059 925 832 6847 1125 7758 304 3815 1736 6403 3131 1795 1154 8095 1704 3586 6202 7663 6230 7871 6079 6075 3195 65 8189 8059 2124 4607 5391 8118 25 278 5503 430 7919 7039 1028 4096 5379 12 3293 2778 6100 2803 8062	3080 6036 5729 1348 32 2036 6763 5114 3978 1856 3232 192 8002 5178 5851 5375 7807 1164 7921 5637 1261 1695 8 4055 1934 1032 8190 7283 4095 7029 6786 5131 7519 5479 6111 512 6383 7818 5507 4454 7679 6143 812 7850 5442 6139 141 2133 7037 136 5196 3105 7527	8183 185 4607 7249 514 8119 7636 7151 7100 1 2236 6143 5772 3967 130 2816 4100 7250 7934 6651 3620 3503 251 8175 6276 6573 1790 3838 3071 6704 20 6142 28 5103 2888 5528 306 7684 6823 8123 6274 160 2999 512 7184 7670 6728 5762 2939 1681 1024 2072 48 8177 58 7891	68516110	34258055

Table A.2: (continued)

instance	Q_1	Q_2^1	Q_2^2	b_1	b_2
8	1396 516 8151 6886 7638 34 2686 6143 5043 1515 6639 8 6099 2081 2688 8063 64 2788 8175 7853 5071 2340 560 1024 2402 5568 8163 8076 4546 48 5726 4384 6864 5894 7679 1600 3390 8123 3455 7581 6017 1094 1 2304 2449 128 3059 3277 1497 7966	88 3154 8063 990 2 392 540 136 7349 1024 2113 1954 1509 7679 8127 2480 5594 8058 5608 7748 448 821 2762 5142 6115 244 7403 6127 1600 1535 8037 128 5474 4490 4120 5438 4224 4109 642 7743 578 7033 8159 7209 7447 822 384 4115 3071 1393 8 8183	4314 5146 3036 514 84 3518 5313 8137 3073 2944 7222 6031 2048 1792 6143 292 8173 7149 7007 41 6015 2088 6738 4616 4786 3067 128 4978 8184 5 524 1 4785 4864 7797 5187 2080 3639 6368 7675 4026 65 1728 64 775 616 4865 5110 7738 807 5970 261 289 8127 1154 7102	57784517	28892258

Appendix B

Synthetic datasets

In this appendix we provide an introduction to the following two types of synthetic datasets we designed specifically for the studies in this dissertation: the symmetric synthetic datasets and the type I non-symmetric synthetic datasets. Please refer to [20] for a detailed description.

B.1 The symmetric synthetic datasets

We define a relational dataset D as follows. The dataset D has K attributes. Each attribute takes m different values. The *master table* contains all the possible entries, by taking all the combinations of all the possible different values over the K attributes. Hence, the number of rows in the master table is m^K. We denote the dataset as $D(K, m)$. We refer to the master table $D(K, m)$ as the *symmetric synthetic dataset* with parameters K and m.

As an illustration, assume that dataset $D(3, 4)$ has three attributes, A, B, and C. Each attribute takes four values 0, 1, 2 or 3. The master table has 64 $(= 4^3)$ entries, shown in Table B.1.

B.2 Type I non-symmetric synthetic datasets

We consider a generalization of the symmetric synthetic datasets of Section B.1, by changing the condition that all the attributes take identical numbers of values. Define a dataset D as having K attributes, denoted as a_1, $a_2, \ldots, a_K$. Attribute a_i takes m_i different values, for $i = 1, 2, \ldots, K$. The master table contains all the possible entries by taking different values over K attributes. We denote the dataset as $D(K; m_1, \ldots, m_K)$. We define dataset $D(K; m_1, \ldots, m_K)$ as *type I non-symmetric synthetic dataset*. Hence, given the input parameters $(K; m_1, \ldots, m_K)$, we obtain the master table of the type I non-symmetric synthetic dataset with the number of rows $\Pi_{i=1}^{K} m_i$.

Table B.1: The symmetric dataset $D(3, 4)$ (the relation is shown using four fragments).

A	B	C	A	B	C	A	B	C	A	B	C
0	0	0	1	0	0	2	0	0	3	0	0
0	0	1	1	0	1	2	0	1	3	0	1
0	0	2	1	0	2	2	0	2	3	0	2
0	0	3	1	0	3	2	0	3	3	0	3
0	1	0	1	1	0	2	1	0	3	1	0
0	1	1	1	1	1	2	1	1	3	1	1
0	1	2	1	1	2	2	1	2	3	1	2
0	1	3	1	1	3	2	1	3	3	1	3
0	2	0	1	2	0	2	2	0	3	2	0
0	2	1	1	2	1	2	2	1	3	2	1
0	2	2	1	2	2	2	2	2	3	2	2
0	2	3	1	2	3	2	2	3	3	2	3
0	3	0	1	3	0	2	3	0	3	3	0
0	3	1	1	3	1	2	3	1	3	3	1
0	3	2	1	3	2	2	3	2	3	3	2
0	3	3	1	3	3	2	3	3	3	3	3

As an illustration, assume dataset $D(3; 2, 3, 4)$ has three attributes, A, B, and C. Attribute A takes two values, 0 or 1; attribute B takes three values, 0, 1, or 2; attribute C takes four values, 0, 1, 2, or 3. The master table thus has 24 $(= 2 \times 3 \times 4)$ entries, as shown in Table B.2.

Note that we could obtain the input parameters $(K; m_1, \ldots, m_K)$ for a type I non-symmetric dataset based on a symmetric synthetic dataset $D(K, m)$, by randomly choosing a number m_i from $\{1, 2, \ldots, m\}$ as the number of values for each attribute a_i.

Table B.2: Non-symmetric dataset $D(3; 2, 3, 4)$ (the relation is shown using two fragments).

A	B	C	A	B	C
0	0	0	1	0	0
0	0	1	1	0	1
0	0	2	1	0	2
0	0	3	1	0	3
0	1	0	1	1	0
0	1	1	1	1	1
0	1	2	1	1	2
0	1	3	1	1	3
0	2	0	1	2	0
0	2	1	1	2	1
0	2	2	1	2	2
0	2	3	1	2	3

Appendix C

Instances of the problem OVS in Section 6.2

Table C.1: Description of 20 instances on the 17-attribute TPC-H dataset discussed in Section 6.2

instance	Q	b
1	8256 46699 131067 1 512 128367 131005 106459 92145 30043 24600 8192 128867 130927 101623 2209 20071 104377 35888 118268	10947015
2	33025 81688 66064 32761 89063 47504 97520 96267 130527 24 103261 126739 114543 922 92076 131023 101216 94142 5 131070 89372 101937 94123 33136 95139 16384 23841 5141 131007 385 88850 2 114547 14608 20605 56319 128995 114623 6208 114687	22948387
3	77468 20544 42462 34135 108727 2098 125927 131007 130809 129019 5376 71027 69664 8192 2824 122747 37152 73732 512 128767 65535 3595 130031 17217 104661 28541 120406 55427 4 83912 120547 121049 130929 32 112126 8769 256 79168 131043 124761 131070 130559 60343 97802 8782 85076 68162 95343 16485 117693 13312 73728 120652 36928 64411 113455 22568 8464 131034 61207	31609208

Table C.1: (continued)

instance	Q	b
4	8 2241 56943 65692 54716 65535 50177 88510 65540 116412 17016 20740 131055 38770 46283 80622 4120 129020 91221 88766 73745 66696 4 256 81341 581 91639 128895 119485 9 2441 96255 55608 58198 114431 22528 33025 121250 9032 129018 64 102928 21997 3073 47402 4418 96099 131070 112517 64535 115086 120831 110545 31519 2048 57526 110334 60694 114687 84924 25624 14699 17749 103711 118298 63485 33024 130559 35840 126975 49544 92507 115967 71631 80895 95231 131007 99853 6944 1045	43745393
5	72 98303 33792 59649 127947 1536 23509 130815 130301 2081 136 7745 264 106521 7801 18982 57473 62009 4224 111456 110591 19467 67842 16576 33793 66592 84098 52859 32770 25130 73971 1 127662 120309 16656 69124 46669 126903 24206 130765 127951 64511 69022 112847 107479 40697 51201 128510 130943 1296 19079 26900 130047 32539 59495 99670 61758 2049 66520 34818 50530 53728 6144 17441 131063 1234 125781 31296 12832 32896 34850 52671 7426 94288 32 94079 94733 128895 17888 130559 2 12416 56991 69627 89597 123884 94164 114623 10238 113919 10187 14838 73412 122623 65600 131037 95682 556 114685 122031	51735957
6	27139 115938 33373 39194 44244 18538 71684 3 65822 2305 67596 44737 20866 79139 70482 8393 8 118978 2176 69152	7340353
7	69124 11172 77217 41312 25700 36429 40013 9 33027 74576 115409 4612 13539 28817 16960 4100 98340 136 16552 53280 512 65696 103368 12352 32832 16 1156 8260 64 5968 16936 8722 106851 9386 55688 115019 21632 41224 3258 19026	10839155
8	16512 32772 38720 66305 39072 8192 45601 69379 4608 16640 50073 401 42113 41216 4 58019 64 8705 75904 21866 16 9266 65572 82178 106560 1284 4914 4624 41257 21120 89349 70374 18592 66688 80 1 53873 16384 70660 9536 24576 64258 2304 8256 14593 108579 83081 28928 33128 98308 16896 50177 8200 1220 88242 2200 17861 36996 102626 24643	14844848

Table C.1: (continued)

instance	Q	b
9	53379 53408 17474 108677 104512 8192 65568 70099 26816 35394 13346 65536 1 90881 8 3620 35864 17353 571 1024 81922 54296 9800 704 290 69137 99849 17310 10261 34952 5184 36 12 4673 70960 26166 11859 2116 35160 256 33347 72132 66080 128 8329 35008 65552 9248 260 4172 21555 54025 33045 74986 33792 83012 34151 4614 15654 17421 51520 1960 66192 18698 108689 79936 1986 2048 54540 57692 75872 2050 33345 32768 98304 2096 35630 43076 74261 16388	22224785
10	65536 4352 75308 16388 132 8198 5731 1057 109196 11909 9069 32768 2944 98318 123162 70021 11105 55296 1538 520 10336 4180 1024 512 9157 6325 86852 20544 33826 72545 123274 98816 2945 69028 18873 8320 1064 76826 128 280 91525 7191 13088 8640 8 66221 36928 86542 16640 2180 80580 107937 75155 50177 2120 8449 115664 2056 26810 107008 45192 24576 116068 2017 2048 16 40 78692 57398 14668 66816 25361 52295 262 10390 640 17840 6187 2055 26133 5120 78208 4 6816 1942 34928 46152 36881 16384 41088 13360 1120 57344 12866 864 25729 90821 129 1576 81936	26914220
11	8108 27176 27178 105236 9664 119705 24255 11336 57805 50952 96982 114168 22146 30710 92672 112626 41778 60143 114357 57273	11810441
12	98272 106152 76921 102926 94769 8073 88633 62011 128807 3054 92973 50507 76460 32342 24662 117722 21730 55772 73358 92058 90625 307 17697 129431 73574 50014 128138 34980 9482 15157 129214 68917 83308 66698 1618 65055 78807 75796 100185 59325	22651364
13	99371 68606 59039 20930 69503 38191 119496 73628 63976 647 18999 127181 19264 43047 125287 51392 91919 78534 23927 63150 14816 14864 20554 122392 89953 8554 79816 19713 98420 46518 125372 56082 70143 82628 101275 129908 121319 8969 31071 22833 78352 41996 89496 17166 56828 35766 121963 98528 83607 56072 107661 40387 69283 35504 47290 33970 84153 20745 35932 8370	31872350

Table C.1: (continued)

instance	Q	b
14	21999 11113 100156 28349 37877 62819 82966 118585 38801 41391 20926 29137 13832 21897 42299 19619 61954 40098 97710 66882 91298 48091 98891 87099 104365 29079 92626 53947 111270 42503 34163 5032 44855 95987 45386 101747 126625 77768 49563 14716 45520 96768 41277 90136 107213 73996 121833 31996 49964 6669 113238 64483 15485 3041 103669 70781 88057 26754 110051 31984 109493 32380 35882 89910 66283 43372 44992 33688 130472 106171 62351 108032 66476 34063 155 52862 114597 66241 128963 16340	46151407
15	64466 38416 88275 2437 76167 64997 28622 84264 126094 53771 2694 38541 58788 109060 46812 130225 61469 69295 29624 13870 127691 20706 53356 102186 11539 51658 6909 83122 89913 123201 119838 66227 56295 7900 67909 55549 105256 7999 122780 50936 20412 101395 67765 72558 33618 15114 21312 17814 45262 123765 19744 96085 726 7171 112989 82116 17820 2387 8978 88039 125386 103855 94267 8915 100840 99499 59503 3796 30889 42403 14472 106697 127505 85569 11860 88992 9154 89038 85993 49721 20580 86611 126509 35712 122595 129238 10309 85263 64097 75063 61489 91130 1798 19340 96461 29891 101369 65996 6946 73590	54921588
16	131039 92133 65527 112368 32702 62169 119790 127455 124205 108407 7966 114551 114106 32671 12670 130717 46566 130031 124927 130712	15770216
17	80815 110583 121845 80807 62189 83767 124903 130527 114104 113791 94975 118775 129974 23518 130047 131054 32551 106287 124927 63353 23143 57134 128628 61415 49150 129663 81919 113393 128247 11955 130942 106124 130014 95929 126821 127774 85964 114679 73329 32718	31959164
18	11775 53089 130008 16171 128722 64493 53047 130907 111087 122110 124861 57273 124903 63357 97913 111715 122879 89678 129737 126900 98286 130863 127710 95177 129981 65471 127997 79815 56042 98047 104109 61387 114685 130985 100203 61102 48119 93950 130486 51132 126797 110015 125575 110843 128255 130559 27962 125687 130023 131063 64987 98239 131039 129023 130942 131055 130550 94134 122111 98298	49024906

Table C.1: (continued)

inst-ance	Q	b
19	124659 105851 131049 46324 114687 122783 118578 96635 31742 23805 55277 64213 73689 130028 131047 87672 13439 129527 49032 61591 92909 130623 128495 126719 7118 126972 19894 29428 65261 122298 50046 130555 90813 94207 125346 122623 113661 131070 28371 128959 48355 73324 40959 127824 122871 59314 98143 28187 128987 112565 129007 57343 40270 67069 56319 130895 65517 73663 111452 59895 88063 61182 81787 82922 122858 131003 93641 106058 100078 93179 58486 97755 126185 107762 125439 130031 15849 130557 32444 48575	62717831
20	126743 122367 89590 129021 130943 60078 130396 89038 59321 114687 59085 38894 129767 126975 105411 44984 97277 121853 90110 65023 106271 42862 125311 118713 56313 112952 85998 24458 97147 90092 64503 130559 85716 88319 31484 130799 119514 7831 129638 131067 124862 130047 124853 55097 131051 126967 130923 71215 114174 122875 130547 98303 126458 42739 95510 89767 108475 122879 128905 64511 53081 64223 130667 95325 126846 65533 63422 114582 94143 32213 118398 114349 124381 125989 126429 109039 120462 32715 49079 113019 118608 130903 123871 67583 131039 91763 20441 131066 113614 122546 32504 64957 115829 130879 128895 106486 93055 129847 120520 126942	80524919

Appendix D

Details of the instance in Example 9 of Section 8.4.1

The space limit b of this instance is $129,084,363$. The subscript numbers of the queries in the query set Q are shown as follows.

36482 22268 74867 5800 256 33540 130815 87396 52901 32768 93439 120243 56953 112547 51059 122863 131070 126782 125951 1053 112598 82599 6336 126187 114559 124401 68917 109567 33052 50776 121105 131063 21892 49186 22539 128 127736 586 11328 72576 131039 75964 130559 18433 10241 16384 122879 125943 73775 7696 63506 38131 28437 118931 87302 114047 131060 8386 51342 73985 128987 2130 70530 126076 56527 4528 126719 46910 114527 2050 97865 85500 33024 118571 136 50444 3077 129021 65007 66571 288 76909 516 24720 75176 16912 49936 2 122331 6144 64 17426 130047 128959 130943 46340 59648 131023 2176 87677 2080 113591 126885 120751 118604 8 130942 99959 16392 103415 8192 73728 4420 72900 57782 126935 130644 1544 512 18533 419 62410 106127 1056 68624

Appendix E

Details of the instance in Example 10 of Section 9.3

The space limits b_1 (i.e., the total space limit) and b_2 (which limits the total size of the views to be materialized in stage two) are $239,154,720$ and $119,577,360$, respectively. The subscript numbers of the queries in the query set Q_1 are

241 4905 6430 107 2603 5167 5812 1694 1566 5569 3203 714 1177 466 2850 469 6745 795 508 2446 5537 5808 2976 5701 1183 5944 2425 5925 5325 7302 6601 358 5475 7044 2532 5889 3924 3346 7171 992 2585 6923 2282 3361 1310 2506 1369 2546 4709 7233 1265 395 2765 7209 1438 4935 6702 1246 5286 3432 4239 3796 1415 6362 1364 3249 3192 3153 3122 1401 2787 2801 7458 7264 2515 5027 1710 2191 6174 2125 5552 7188 7784 6350 2472 5319 6632 1860 5746 4156 880 4548 4568 3624 570 409 5320 627 574 1293 2581 2582 986 7600 1197 5465 3388 497 95 2833 5748 2992 798 3852 6392 6436 6550 310 2190 7202 871 5764 7873 6920 4808 6437 3632 1846 5139 4237 917 7696 677 4890 4254 5458 7120 4788 3141 2166 6725 3490 2350 3599 1508 809 3339 2646 4582 7219 2977 6810 1870 438 541 5482 1291 1858 867 2137 1843 2831 2868 2945 4325 4455 2404 2664 2540 3681 6624 2517 4728 978 2435 1461 2936 3636 5900 1633 661 7211 3731 218 7760 1643 3358 2771 2258 7693 2198 872 5710 970 315 4683 5263 3474 5461 5446

The subscript numbers of the queries in the query set Q_2^1 are

4871 6364 1264 2500 2681 7632 271 590 7307 2839 926 2238 3188 1611 7274 6163 5889 4897 2882 434 3441 3624 558 665 1483 7494 314 678 4665 2503 1435 2708 4998 301 6539 3464 2411 4199 4461 5223 6850 7948 3233 3667 6722 4761 4615 859 1484 5253 1306 4147 4195 3524 6232 2426 183 5337 1464 988 6295 5008 2375 1993 3448 415 3525 4391 6018 3297 2087 1272 2508 3127 7472 1860 1858 1302 5193 6832 5539 1067 3986 4397 5179 1549 718 6380 7205 2347 6340 2606 1622 2634 4995 6217 3349 4345 4459 2499 7946 6961 302 3176 4874 2868 5257 7744 2095 6484 2956 535 4222 2840 61 5412 5286 3590 682 501 5032 1430 5453 63 249 5318 5646 7260

5064 2804 2573 2342 7088 1637 6417 3101 3904 1357 2702 5013 2441 6826 1656 1630 1674 4883 7730 4203 1510 3133 632 2461 4930 1450 702 4366 1588 1956 4833 6917 1864 1195 3149 3411 3537 7691 6406 2032 3371 6165 6423 6868 2281 707 2905 619 5322 6537 1825 399 7334 6189 5721 5733 7456 6988 5017 1010 3342 1595 617 1953 1606 4265 4312 327 4443 6682 1636 6236 7249

The subscript numbers of the queries in the query set Q_2^2 are

4367 758 1802 6329 4643 7197 5255 6471 8002 1370 2612 4389 1768 2315 3234 5177 3219 4472 2441 3493 2548 5650 124 5223 1306 5653 1845 4728 2119 6251 2761 1652 5844 3400 6692 4806 238 4530 5417 7195 3685 4306 2854 683 1828 3986 4464 1388 1708 6757 3698 3417 471 7617 5637 4770 3761 6689 5400 2774 7572 5895 3848 2700 1211 733 5794 4547 6798 4198 6669 757 5396 890 3854 310 286 4993 5172 7333 4182 6948 3940 845 2387 2653 2501 598 5473 3778 2667 1575 3100 3113 783 372 5300 2650 1039 3401 1606 3618 4661 7178 313 6732 5386 1660 3156 4903 3297 1880 6727 7488 2467 630 6306 3632 1675 4526 4276 936 2788 5170 2913 2606 4904 7433 3042 1884 4298 7561 729 7209 967 7876 743 6819 6535 4404 3552 4883 6458 5529 4677 371 5390 1976 6549 6313 4694 3101 4742 4793 8096 2793 5642 2765 2615 6346 1829 7337 4173 945 4881 2970 1233 4638 4302 7214 1764 6477 6151 4801 3083 1000 4852 4291 5667 3203 5361 2610 3239 5127 4976 6328 7472 3752 3212 1287 5937 2288 689 1364 1457 2981 2451 4316 4322 5477 7792 4885 4917 5281 924