Acta Informatica (2007) 44:289–321
DOI 10.1007/s00236-007-0046-z

ORIGINAL ARTICLE

View selection for *real* conjunctive queries

Foto Afrati · Rada Chirkova · Manolis Gergatsoulis · Vassia Pavlaki

Received: 28 May 2006 / Accepted: 17 April 2007 / Published online: 26 May 2007
© Springer-Verlag 2007

Abstract Given a query workload, a database and a set of constraints, the view-selection problem is to select views to materialize so that the constraints are satisfied and the views can be used to compute the queries in the workload efficiently. A typical constraint, which we consider in the present work, is to require that the views can be stored in a given amount of disk space. Depending on features of SQL queries (e.g., the DISTINCT keyword) and on whether the database relations on which the queries are applied are sets or bags, the queries may be computed under set semantics, bag-set semantics, or bag semantics. In this paper we study the complexity of the view-selection problem for conjunctive queries and views under these semantics. We show that bag semantics is the "easiest to handle"

Rada Chirkova's work on this material has been supported by the National Science Foundation under Grant No. 0307072.
The project is co-funded by the European Social Fund (75%) and National Resources (25%)- Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the program PYTHAGORAS.
A preliminary version of this paper appears in F. Afrati, R. Chirkova, M. Gergatsoulis, V. Pavlaki. Designing Views to Efficiently Answer Real SQL Queries. In Proc. of SARA 2005, LNAI Vol. 3607, pages 332-346, Springer-Verlag, 2005.

F. Afrati · V. Pavlaki
Department of Electrical and Computing Engineering,
National Technical University of Athens (NTUA), 15773 Athens, Greece
e-mail: afrati@softlab.ntua.gr

V. Pavlaki
e-mail: vpavlaki@softlab.ntua.gr

R. Chirkova
Computer Science Department, North Carolina State University,
Campus Box 8206, Raleigh, NC 27695-8206, USA
e-mail: chirkova@csc.ncsu.edu

M. Gergatsoulis (✉)
Department of Archive and Library Sciences, Ionian University,
Palea Anaktora, Plateia Eleftherias, 49100 Corfu, Greece
e-mail: manolis@ionio.gr

(we show that in this case the decision version of view selection is in NP), whereas under set and bag-set semantics we assume further restrictions on the query workload (we only allow queries without self-joins in the workload) to achieve the same complexity. Moreover, while under bag and bag-set semantics filtering views (i.e., subgoals that can be dropped from the rewriting without impacting equivalence to the query) are practically not needed, under set semantics filtering views can reduce significantly the query-evaluation costs. We show that under set semantics the decision version of the view-selection problem remains in NP only if filtering views are not allowed in the rewritings. Finally, we investigate whether the *cgalg* algorithm for view selection introduced in Chirkova and Genesereth (Linearly bounded reformulations of conjunctive databases, pp. 987–1001, 2000) is suitable in our setting. We prove that this algorithm is sound for all cases we examine here, and that it is complete under bag semantics for workloads of arbitrary conjunctive queries and under bag-set semantics for workloads of conjunctive queries without self-joins.

1 Introduction

The *view-selection* problem is the following; given a set of queries (which we call a *query workload*), a database, and a set of constraints on materialized views (e.g., *storage limit*, which is a bound on the amount of disk space available for storing the materialized views), return definitions of views that, when materialized in the database, would satisfy the constraints and reduce the evaluation costs of the queries.

The problem of view selection has received significant attention in the literature [4,5, 14–18,21,27,28]. The original motivation for the problem comes from data-warehouse design, where we need to decide which views to store in the warehouse to obtain optimal query-answering performance [5,18,27]. Another motivation is provided by several commercial database-management systems (DBMS), which support incremental updates of materialized views and use materialized views to speed up query evaluation. Choosing an appropriate set of views to materialize in the database is crucial to obtain performance benefits [4]. The increasing use of distributed systems has made the view-selection problem even more practical, since it can be viewed as a special case of intelligent data placement in a network with two nodes: the database and the warehouse (see [22] for a survey). Finally, automated design of materialized views to answer queries is needed in automated query-performance tuning [1,19,26].

The majority of existing work assumes *set semantics*, where both the database relations and query answers are sets of tuples. At the same time, under standard semantics of evaluating SQL queries duplicate tuples are not eliminated unless explicitly requested (by using the `DISTINCT` keyword). Thus, while database relations are often duplicate-free (that is, are sets), query answers are often bags. We refer to this semantics as *bag-set semantics*. As SQL is the query language used in most commercial DBMS, results on rewriting queries under bag-set semantics are useful in practice. Another semantics for query evaluation is *bag semantics*, where both database relations and query answers may have duplicate tuples. Studying the bag-semantics case is interesting not just from theoretical but also from practical perspective. The reason is, in view selection it is possible to design and materialize bag-valued views and thus to obtain bag-valued databases of stored data. Computing query answers on such databases without using the `DISTINCT` keyword obeys the laws of bag, rather than bag-set, semantics.

In this paper we study the complexity of the view-selection problem under set, bag, and bag-set semantics for conjunctive queries and views. The class of conjunctive queries covers a wide class of database queries as every select-project-join query can be written as a conjunctive query [6]. Under bag semantics, we show the decision version of the problem to be in NP. Under set and bag-set semantics we need to assume further restrictions on the query workload (we only allow queries without self-joins in the workload) to achieve the same complexity. Thus, bag semantics are "easier to handle". We also investigate the influence of *filtering views* [3,25] (i.e., of subgoals that can be dropped from the rewriting without impacting equivalence to the query) on the complexity of view selection. The usage of filtering views is that they may improve significantly the evaluation cost of the query, since they may filter out many tuples early on during query evaluation. We show that under bag and bag-set semantics filtering views are practically not needed, whereas under set semantics they may reduce significantly the query-evaluation costs. In particular, we show that under set semantics the decision version of the view-selection problem remains in NP only if filtering views are not allowed in the rewritings.

The results of this paper can be used in finding sound and complete algorithms for designing views and for rewriting queries under each of the three semantics. In that sense, we investigate whether the *cgalg* algorithm for view selection introduced in [9] is suitable in our setting. We prove that *cgalg* is sound under all three semantics, and investigate the conditions under which it is also complete. Finally, we present experimental results concerning the efficiency of *cgalg* algorithm.

1.1 Contributions and outline

In this subsection we outline the main contributions of the present work and the structure of the paper. The aim of this work is to develop results on designing views to answer queries and on rewriting queries using views in relational databases under set, bag and bag-set semantics. The main contributions are presented in Fig. 1. Note that for rows with "views (rewritings, respectively) without self joins" in the left-most entry, the "yes" entry means that if queries have no self-joins then an optimum viewset exists where all view definitions (rewritings, respectively) have no self-joins.

In more detail, the contributions of this work are as follows. We start by showing cases under all three semantics (set, bag and bag-set), where the decision version of the view-selection

Semantics & queries / **Views**	set semantics, queries without self-joins	bag semantics		bag-set semantics	
		queries with self-joins	queries without self-joins	queries with self-joins	queries without self-joins
views without self-joins	Yes (Theorem 1)	not necessarily	Yes (Corollary 6)	open	Yes (Lemma 6)
rewritings without self-joins	Yes (Theorem 2)	not necessarily	Yes (Theorem 5)	open	Yes (Corollary 9)
filtering views in rewritings	not allowed	there are no rewritings with filtering views (Corollary 3)		do not improve evaluation cost (Theorem 8)	
bound on size of view definition	Linear (Corollary 1)	Linear (Corollary 5)		open	Linear (Corollary 8)
bound on size of rewritings	Linear (Proposition 1)	Linear (Proposition 2)		open	Linear (Proposition 4)
complexity of view selection (decision version)	NP (Theorem 3)	NP (Theorem 6)		open	NP (Theorem 9)

Fig. 1 Summary of results. For rows with "views ("rewritings", respectively) without self-joins" in the left-most entry, the "yes" entry means that if queries have no self-joins then an optimum viewset exists where all view definitions (rewritings, respectively) have no self-joins

problem is in NP. We prove that by establishing linear upper bounds on (a) the size of the view definitions, and (b) the size of the rewritings. Under bag semantics we investigate the problem of designing views to answer queries in its general form (i.e., queries with or without self-joins), whereas under set semantics and bag-set semantics we only consider the special case of queries without self-joins. Under all semantics we prove that for queries without self-joins we can design optimal viewsets whose view definitions do not require self-joins. In all these cases we can also obtain rewritings that do not use self-joins of views.

Further, we investigate the influence of filtering views on query rewriting. In particular, we prove that under bag-set semantics the use of filtering views does not improve query-evaluation costs, while under bag semantics filtering views cannot be used in query rewriting. Finally, under set semantics we only consider query rewriting without using filtering views. The general problem of query rewriting under set semantics using filtering views remains open.

Finally, as we have already mentioned, the results developed in this work can be used to find sound and complete algorithms to design views under each of the three semantics. In this work we investigate the conditions under which the *cgalg* algorithm for view selection introduced in [9] is useful in our problem. In particular, we show that *cgalg* is sound under set, bag, and bag-set semantics. We also prove that *cgalg* is complete under bag semantics for workload of arbitrary queries. Concerning bag-set semantics, the *cgalg* algorithm is complete for workloads of queries without self-joins.

The rest of the paper is organized as follows. Section 1.2 presents two examples that illustrate some of the complications of the problem of selecting views to materialize when set semantics is assumed, and Sect. 1.3 reviews related work. Section 2 contains basic definitions and formalizes the view-selection problem. Section 3 presents results on query rewriting using views for queries without self-joins under set semantics. Section 4 studies the problem under bag semantics, whereas Sect. 5 presents our results under bag-set semantics. In Sect. 6 we discuss the *cgalg* algorithm, a sound view-selection algorithm that is also complete under certain conditions and depending on the semantics we adopt. Section 7 concludes the paper and discusses some directions of future work.

1.2 Examples

In this section we give two examples (Examples 1 and 2) that illustrate some of the complications of the problem of selecting views to materialize when set semantics is assumed. In both examples that follow (and in the formulation of the problem throughout this paper) we exhibit a workload of select-project-join queries and a storage limit. In order to demonstrate the complexity of the problem, in both examples we select a storage limit such that it is not possible to materialize the answers to all the workload queries without violating the storage-limit constraint. Example 1 shows that the search space of potentially useful views can be very large even for simple and common select-project-join queries.[1]

Example 1 Consider a large retail chain with multiple stores and warehouses, where products are ordered and shipped daily from the warehouses to replenish the inventory in the stores. Suppose that `Shipments` is a relation in the database, and let `warehouseID`, `warehouseCity`, `storeID`, `storeCity`, `orderNumber`, `shipmentNumber`, and `shipmentDate` be some of its attributes. Suppose the employees of the retail chain contract shipments to independent truck drivers, by attracting them with tours connecting two or more

[1] This example is a variation on Example 1 in [11]; unlike that example, here we restrict the search space of views in that we do not consider filtering views.

cities. The company predefines a number of tour types to offer to the truck drivers, and the company employees need to query the database and find out whether the tour requested by the driver exists starting at a given city. Every tour type starts and ends in the *same* city. The simplest tour is the "two-city roundtrip": The query returns all cities `warehouseCity` such that there exist two scheduled deliveries: one from `warehouseCity` to some `storeCity` on a given `shipmentDate 'date1'`, and the other from (a warehouse in) `storeCity` back to (some store in) `warehouseCity` on a later `shipmentDate 'date2'`. The following is a SQL definition of the query; note the `DISTINCT` keyword that enforces set semantics.

```
SELECT DISTINCT S1.warehouseCity
FROM Shipments S1, Shipments S2
WHERE S1.storeCity = S2.warehouseCity AND
      S1.warehouseCity = S2.storeCity AND
      S1.shipmentDate = 'date1' AND S2.shipmentDate = 'date2';
```

As we show in Example 4, under set semantics, for a given storage limit and for a query workload that has several such tour queries for tours of different lengths, we can select and materialize a *single* view such that the number of copies of the `Shipments` relation in the `FROM` clause of the view is *at least exponential* in the combined size of the query workload. Thus, in view selection under set semantics we potentially need to consider up to an exponential number of views in the size of the input query workload. At the same time, we show in this paper (see Corollary 4) that in view selection under bag semantics we do not need to consider views whose definitions have more subgoals than the number of subgoals of the longest input query.

Example 2 shows that even if we further simplify the language of queries of interest, there still remains a large number of views that could significantly reduce the evaluation costs of the queries under set semantics.

Example 2 We use here the application domain of Example 1. Suppose that in addition to the `Shipments` relation described in Example 1, the database also has an `Inventory` relation, where the attributes of interest are `storeID`, `productID`, `orderNumber`, and `isOutstanding`. We assume that the volume of daily orders to replenish store inventories is large, and that a single order is typically made for a large range of product types and potentially for several stores in the same area. Further, suppose that different product types are stored at different warehouses, and thus all the products ordered in a single order can be delivered to a store via multiple shipments. Finally, we assume that on delivery, the contents of most (but not all) shipments are put on the store shelves and reflected in the inventory records on the same day.

Suppose that at the end of each day, the management of the retailer chain routinely runs certain "daily report" queries on the deliveries. One of the queries asks for the stores and product IDs such that some quantity of the product has been ordered and was to be delivered to the store on the day in question, but the order is still listed as "outstanding" in the store inventory. The query in SQL is:

```
SELECT DISTINCT I.storeID, productID
FROM Shipments S, Inventory I
WHERE S.storeID = I.storeID AND
      S.orderNumber = I.orderNumber AND
      shipmentDate = 'today' AND isOutstanding = 'yes';
```

Note that unlike the query in Example 1, the `FROM` clause of this query has just one copy of each relation. That is, the daily-report query *has no self-joins*. We show in this paper (see Sects. 3.1, 4.2, and 5.3) that even under set semantics (as well as under bag and bag-set semantics), when doing view selection for such queries we do not need to consider views whose definitions have more subgoals than the input queries. Still, we need to consider a significant number of views.

Under the domain assumptions in this example, even using reasonable indexes on `Shipments` or `Inventory` does not eliminate most redundant tuples in the result of the join, and thus postprocessing of a large temporary join result (which by definition never even has an index) is part of any query plan that does not use materialized views. The reason is, under our domain assumptions a shipment where `shipmentDate = 'today'` typically corresponds to a large number of product IDs in the `Inventory` relation, where the value of `isOutstanding` is `'no'` for most records. Conversely, for a large number of combinations of values of `storeID` and `productID` in the `Inventory` relation where `isOutstanding = 'yes'`, the value of `shipmentDate` of the corresponding order is not `'today'`.

Consider materializing a view `V` that is a natural join of those tuples in the relations `Shipments` and `Inventory` where `shipmentDate = 'today'` and `isOutstanding ='yes'`. Suppose that in the answer to `V` we have (at least) attributes `storeID` and `orderNumber`. Then we can use this view as a *filter* (in essence like an index or a semijoin) to narrow down the number of tuples in the large result of joining the relations `Shipments` and `Inventory`. The query plans that use `V` all join `V` with either `Shipments` or `Inventory` first (an index can be maintained on `V` to make the join efficient), and then join the resulting *smaller-size* temporary relation with the remaining relation in the `FROM` clause of the query.[2]

From Examples 1 and 2 it is clear that under set semantics, having a materialized view such as `V` could improve the processing efficiency of the daily-report query, which is important when the query is asked often and regularly on large relations.[3] At the same time, the choice of output attributes in a filtering view of this type depends on a number of criteria, including the types of other queries in the query mix. Thus, potentially we would have to consider *all* subsets of the combination of all attributes of the stored relations `Shipments` and `Inventory`. Sections 4.1 and 5.2 show that, unlike the case of set semantics, filtering views do not need to be considered in view selection under bag or bag-set semantics. Thus, in view selection we can further restrict the search space of views that are useful in rewriting the given workload queries.

1.3 Related work

View selection: The view-selection problem has recently attracted the interest of many researchers [4,5,15–18,21,27,28]. In Chirkova et al. [9] considered the space requirements for the view-selection problem for the context of data warehouse design. Under certain restrictions one can limit the search of an optimal configuration to views that are subexpressions of the queries in the workload. The model of [15] does not capture selections, projections, or different join predicates. In a later paper [17], the authors consider the cost of view maintenance,

[2] In presence of frequent updates on the stored data, the update problem for the view `V` is not much more complex than the problem of updating indexes on `Shipments` or `Inventory`, because we have just two relations in the `FROM` clause of the view.

[3] For instance, WalMart maintains a database with billions of rows in stored relations.

but under the same model of inputs. Chirkova et al. in [11] showed that the complexity of view selection depends on how to model the problem for views and workloads that consist of equality-selection, project and join queries. In particular, they showed that the complexity of the problem depends crucially on the quality of the estimates that a query optimizer has on the size of the views to be materialized. If a query optimizer has good estimates of the sizes of the views, an optimal choice of views may involve an exponential number of views in the size of the database schema, whereas if an optimizer uses standard estimation heuristics, the number of necessary views and the expression size of each view are polynomially bounded. Note that in [11] the input is the database schema and the workload queries, whereas in [15,17] the set of relations in the warehouse must be also given.

In [18], Harinarayan and colleagues considered and analyzed algorithms for selection of views in the case of “data cubes”. They showed that the problem of view selection for data cubes is NP-hard, and described greedy algorithms for approximating an optimal set of views. In [16] the authors extended their results to selection of views and indexes in data cubes. View selection for data cubes is further elaborated in [21]. Agrawal et al. [4] described a system for view selection that is incorporated into the Microsoft SQL Server. They presented several effective heuristics for pruning the space of possible view configurations. An important aspect of their work is that they considered the problem of selecting views and indexes simultaneously. Because of that, they did not consider projection views, since those can often (but not always) be simulated by indexes on other views.

Bag, bag-set semantics: In [8] the containment problem of conjunctive queries under bag semantics was studied and was proved to be $\prod_2^p$-hard. Equivalence under bag semantics has the same complexity as the graph-isomorphism problem, which is in NP. Deutsch [12] presented techniques for rewriting queries for bag semantics, bag-specific constraints (UWDs), and for handling bag queries over arbitrary mixes of bag and set schema elements and views. The problem of optimizing queries with materialized views under bag semantics is studied in [7] and under set semantics in [23]. Finally, [20] studies conjunctive queries over generalized databases, to obtain an understanding of the behavior of relations as multisets (cf. SQL query-evaluation semantics).

2 Preliminaries

In this section we give the basic definitions, state formally the view-selection problem, discuss some of the challenges of the problem, and define the necessary concepts used in the remainder of the paper.

2.1 Basic definitions

A *relational database* is a collection of stored relations. Each relation R is a collection of tuples, where each tuple is a list of values of the attributes in the *relation schema* of R. A relation can be viewed either as a *set* or as a *bag* (another term is *multiset*) of tuples. A bag can be thought of as a set of elements (we call it the *core-set* of the bag) with multiplicities attached to each element. In a *set-valued database*, all stored relations are sets; in a *bag-valued database*, multiset stored relations are allowed.

A *query* is a mapping from databases to databases, usually specified by a logical formula on the schema $\mathcal{S}$ of the input databases. Typically, the output database (the *query answer*) is a database with a single relation.

In this paper we focus on select-project-join SQL queries with equality comparisons, a.k.a. safe *conjunctive queries*. A conjunctive query is a rule of the form:

$$Q: \ ans(\bar{X}) \text{ :- } e_1(\bar{X}_1), \ldots, e_n(\bar{X}_n)$$

where $e_1, \ldots e_n$ are database relations and $\bar{X}, \bar{X}_1, \ldots, \bar{X}_n$ are vectors of variables or constants. The variables in $\bar{X}$ are called *head* or *distinguished variables* of Q, whereas the variables in $\bar{X}_i$ are called *body variables* of Q. A conjunctive query is said to be *safe* if all its distinguished variables also occur in its body. A query has *self-joins* if the minimized query definition [6] has at least two subgoals with the same relation name. A *view* refers to a named query. A view is said to be *materialized* if its answer is stored in the database.

We say that a bag B is a *subbag* [8] of a bag B' (we write $B \subseteq_b B'$) if each element of B is contained in B' with a greater than or equal multiplicity. The *bag union* ($\sqcup$) [8] of two bags is obtained by adding the multiplicity factors for each tuple in each bag.

A query Q_1 is *set-contained* in a query Q_2, denoted by $Q_1 \sqsubseteq_s Q_2$, if for any set-valued database $\mathcal{D}$ the answer to Q_1 on $\mathcal{D}$ under set semantics is a subset of the answer to Q_2 on $\mathcal{D}$ under set semantics. A query Q_1 is *bag-contained* (*bag-set contained*) in Q_2, denoted by $Q_1 \sqsubseteq_b Q_2$ ($Q_1 \sqsubseteq_{bs} Q_2$, respectively), if for any bag-valued (set-valued, respectively) database $\mathcal{D}$, the answer to Q_1 on $\mathcal{D}$ under bag semantics (bag-set semantics, respectively) is a subbag of the answer to Q_2 on $\mathcal{D}$ under the same semantics. Two queries are *equivalent under set/bag/bag-set semantics* ($Q_1 \equiv_s Q_2$, $Q_1 \equiv_b Q_2$, $Q_1 \equiv_{bs} Q_2$, respectively) if they are contained in each other under that semantics.

2.2 Rewriting queries using views

Let $\mathcal{V}$ be a set of views defined on a database schema $\mathcal{S}$, and $\mathcal{D}$ be a database with the schema $\mathcal{S}$. Then by $\mathcal{D}_{\mathcal{V}}$ we denote the database obtained by computing all the view relations in $\mathcal{V}$ on $\mathcal{D}$. Let Q be a query defined on $\mathcal{S}$, and $\mathcal{V}$ be a set of views defined on $\mathcal{S}$. A query R is a *rewriting of the query Q using the views in $\mathcal{V}$* if all subgoals of R are view predicates defined in $\mathcal{V}$ or interpreted predicates.

The *expansion* R^{exp} of a rewriting R of a query Q using views is obtained from R by replacing all the view atoms in the body of R by their definitions in terms of the base relations. A rewriting R of a query Q on a set of views $\mathcal{V}$ is a *contained rewriting* of Q using $\mathcal{V}$ under set semantics if for every database $\mathcal{D}$, $R(\mathcal{D}_{\mathcal{V}}) \subseteq Q(\mathcal{D})$. A rewriting R of a query Q on a set of views $\mathcal{V}$ is an *equivalent rewriting* under set semantics if for every database $\mathcal{D}$, $R(\mathcal{D}_{\mathcal{V}}) = Q(\mathcal{D})$.

The definition of the notion *contained rewriting* for *bag* or *bag-set* semantics is analogous. The only difference is that we now require that $R(\mathcal{D}_{\mathcal{V}})$ be a subbag of $Q(\mathcal{D})$. The definition of the notion of *equivalent rewriting* for the *bag* and *bag-set* semantics is the same as above (in this case, however, the symbol '=' stands for bag equality). A conjunctive equivalent rewriting Q' of a conjunctive query Q (under some semantics) is *locally minimal* [23] if we cannot remove any atoms from Q' and still retain equivalence to Q (under the same semantics).

There are two types of conjunctive views that can be used in a conjunctive rewriting of a conjunctive query [3,25]: (1) containment-target views, and (2) filtering views. A conjunctive view V is a *containment-target view* for a conjunctive query Q if there exists a conjunctive rewriting P of Q (P uses V), and there is a containment mapping (as defined in each case of set, bag and bag-set semantics [8]) from Q to the expansion P^{exp} of P, such that V provides the image of at least one subgoal of Q under the mapping. Intuitively, in a rewriting, a containment-target view "covers" at least one query subgoal. Covering all query subgoals is enough to produce a rewriting of the query. A view is a *filtering view* for a query if it is not a containment-target view.

2.3 The view-selection problem

Given a set, or *workload*, $\mathcal{Q}$ of queries on stored relations and a database instance, we want to find and precompute offline a set of intermediate results, defined as views (we call this set of views a *viewset*) on these relations. The views can be used to compute the answers to all queries in the workload $\mathcal{Q}$.

Our goal is to design minimal-cost views, that is, views whose use in the rewriting of a query results in the most efficient computation of the query. We assume that the view relations have been precomputed and stored in the given database $\mathcal{D}$. Thus, we do not assume any cost of computing the views. We measure the size of each relation on $\mathcal{D}$ as the number of tuples in the relation.

For conjunctive queries we measure the cost of query evaluation as the sum of the costs of all the joins in the evaluation [11,16]. More precisely, suppose we are given a query Q and a database $\mathcal{D}$. We measure the cost of a join of two relations as the sum of the sizes of the input and output relations; this models faithfully hash joins in relational database-management systems. For multi-way joins, we consider specific query plans for evaluating the query Q on $\mathcal{D}$. We assume use of only left-linear query plans, where selections are pushed as far as they go and projection is the last operation. Thus, each plan is a permutation of the subgoals of the query, and the cost of this query plan on a given database instance $\mathcal{D}$ is defined inductively as follows. For $n = 1$, the cost of query plan $Q = R_1$ is the size of the relation R_1. For each $n \geq 2$, the cost of query plan $(\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_n)$ over n relations is the sum of the following four values:

1. the cost of query plan $(\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_{n-1})$,
2. the size of relation $R_1 \bowtie \ldots \bowtie R_{n-1}$,
3. the size of relation R_n, and
4. the size of relation $R_1 \bowtie \ldots \bowtie R_n$.

The cost of evaluating a query Q on a database $\mathcal{D}$ is the minimum cost over all Q's query plans when evaluated on $\mathcal{D}$.

In this paper we consider *problem inputs* that are 4-tuples $(\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{S}$ is a database schema, $\mathcal{Q}$ is a workload of queries defined on $\mathcal{S}$, $\mathcal{D}$ is a database with schema $\mathcal{S}$, and $\mathcal{L}$ is a collection of constraints on sets of materialized views. A problem input $\mathcal{P}$ is said to be *set-oriented* (*bag-oriented*, *bag-set-oriented*, respectively) if we consider set semantics (bag semantics, bag-set semantics, respectively) for computing query answers; $\mathcal{P}$ is said to be *conjunctive* if we consider the conjunctive language for queries, views and rewritings.

Results in this paper are given for a special type of constraints $\mathcal{L}$ on materialized views: $\mathcal{L}$ is a singleton $\mathcal{L} = \{C\}$, $C \in \mathbf{N}$. The *storage limit* C means that the total size $size(\mathcal{V}(\mathcal{D}))$ of the relations for the views in $\mathcal{V}$ on $\mathcal{D}$ must not exceed C. If the storage limit is sufficiently large then we can materialize all query answers, which is an *optimal viewset*. The problem becomes interesting when the storage limit is less than that.[4]

The concepts of candidate and optimal rewritings are defined as follows.

Definition 1 For a given query Q, semantics (set, bag, or bag-set) for evaluating the query on the database, viewset $\mathcal{V}$, and database $\mathcal{D}$:

1. R is a *candidate rewriting* of Q in terms of $\mathcal{V}$ if R is an equivalent rewriting of Q under the given semantics, and

[4] If the storage limit is too small, then there is no viewset that can rewrite all queries.

2. R is an *optimal rewriting* of Q in terms of $\mathcal{V}$ on $\mathcal{D}$ if R is a candidate rewriting that minimizes the cost of computing the answer to Q on $\mathcal{D}_{\mathcal{V}}$ among all candidate rewritings of Q in terms of $\mathcal{V}$.

Definitions 2 and 3 formally define admissible and optimal viewsets.

Definition 2 Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a problem input. A set of views $\mathcal{V}$ is said to be an *admissible viewset* for $\mathcal{P}$ if:

1. $\mathcal{V}$ gives equivalent (*candidate*) rewritings of all the queries in $\mathcal{Q}$,
2. For every view $V \in \mathcal{V}$, there exists an equivalent rewriting of a query in $\mathcal{Q}$ that uses V, and
3. $\mathcal{V}$ satisfies the constraints $\mathcal{L}$.

Definition 3 For a problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, an *optimal viewset* is a set of views $\mathcal{V}$ defined on $\mathcal{S}$, such that:

1. $\mathcal{V}$ is an admissible viewset for $\mathcal{P}$, and
2. $\mathcal{V}$ minimizes the total cost of evaluating the queries in $\mathcal{Q}$ on the database $\mathcal{D}_{\mathcal{V}}$, among all admissible viewsets for $\mathcal{P}$.

Definition 4 defines nonredundant viewsets.

Definition 4 For any problem input $\mathcal{P}$, a viewset $\mathcal{V}$ is said to be *nonredundant* for $\mathcal{P}$ if $\mathcal{V}$ is admissible for $\mathcal{P}$ and there is no proper subset $\mathcal{V}'$ of $\mathcal{V}$ such that $\mathcal{V}'$ is also admissible for $\mathcal{P}$.

In some results of this paper, instead of a database $\mathcal{D}$ in the definition of a problem input we will use the notion of an *oracle* $\mathcal{O}$. An oracle is supposed to give, instantaneously, exact relation sizes for all views defined on the schema $\mathcal{S}$. In this case a problem input is written as $(\mathcal{S}, \mathcal{Q}, \mathcal{O}, \mathcal{L})$. The notion of an optimal viewset is defined analogously to the case of problem inputs of the form $(\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{D}$ is a database. The results in the remainder of this paper are given for problem inputs that include a fixed database, but can be extended in a straightforward manner to problem inputs that include an oracle.

3 Queries without self-joins under set semantics

In this section we consider the view-selection problem under *set* semantics. We present the following results:

1. In Sect. 3.1 we show that for workloads of queries without self-joins there exist optimal viewsets whose view definitions do not have self-joins. Moreover, the view definitions in such viewsets have no more subgoals than any query in the workload.
2. In Sect. 3.2 we show that for workloads of queries without self-joins there exist optimal viewsets, such that rewriting any workload query does not require self-joins of containment-target views in the viewset.
3. In Sect. 3.3 we show that the decision version of the view-selection problem is in NP for workloads of queries without self-joins, provided that filtering views are not used in query rewritings.

These results are very useful in designing algorithms for constructing optimal viewsets, as they provide a bound on the number of atoms in each view in an optimal viewset and ensure that these views do not contain self-joins, provided the workload queries do not contain self-joins.

3.1 View definitions without self-joins

The following theorem holds for queries without self-joins under set semantics.

Theorem 1 *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{L}$ represents a single storage limit and all queries in $\mathcal{Q}$ are conjunctive queries without self-joins, if there exists an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ under the storage limit constraint $\mathcal{L}$, then there exists an optimal viewset $\mathcal{V}'$ under $\mathcal{L}$ such that each view in $\mathcal{V}'$ can be defined as a conjunctive query without self-joins.*

Proof Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be an arbitrary problem input that satisfies the restrictions of the theorem, and $\mathcal{W}$ be an optimal viewset for $\mathcal{P}$, such that $\mathcal{W}$ has views with self-joins. We will show that $\mathcal{W}$ can be transformed into an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ that does not have views with self-joins. We construct the viewset $\mathcal{V}$ by replacing each view definition W_i in $\mathcal{W}$ with the result V_i of applying to W_i the most general unifier (mgu) θ_i of all subgoals with the same predicate symbol in the body of W_i. Notice that, since $\mathcal{W}$ is admissible, for each view W in $\mathcal{W}$ there exists an equivalent rewriting for some query $Q \in \mathcal{Q}$ using W and thus there must exist a containment mapping from the definition of W to (some part of) the definition of the query Q [6]. As Q does not contain self-joins we can conclude that for every predicate symbol p in W, all atoms in W whose predicate symbol is p, are unifiable.

From the above we can conclude that none of the views in $\mathcal{V}$ has self-joins. Applying the mgu to the views in $\mathcal{W}$ gives view definitions with possibly multiple identical subgoals for any relation name used in the view definition; under set semantics, we can obtain equivalent definitions by removing any duplicate subgoals [6]. In addition, again by construction, each view $V_i \in \mathcal{V}$ is contained in the corresponding view $W_i \in \mathcal{W}$. Thus, the total size of the relations for $\mathcal{V}$ on the database $\mathcal{D}$ does not exceed the total size of the relations for $\mathcal{W}$ on $\mathcal{D}$:

$$size(\mathcal{V}(\mathcal{D})) \leq size(\mathcal{W}(\mathcal{D})).$$

To show that $\mathcal{V}$ is an optimal viewset for $\mathcal{P}$, it remains to prove the following two points:

1. *The viewset $\mathcal{V}$ gives equivalent rewritings of all the queries in the input workload $\mathcal{Q}$.* By our assumption, for any query $Q \in \mathcal{Q}$, the viewset $\mathcal{W}$ gives an equivalent rewriting $Q_{\mathcal{W}}$ of Q. We use $Q_{\mathcal{W}}$ to construct an equivalent rewriting $Q_{\mathcal{V}}$ of the query Q in terms of the viewset $\mathcal{V}$, as follows. For each atom $W_i(\bar{X})$ of a view W_i in the rewriting $Q_{\mathcal{W}}$, we replace $W_i(\bar{X})$ by an atom $V_i(\bar{X})$ of the corresponding view $V_i \in \mathcal{V}$; note that the atoms for W_i and V_i have *the same* list $\bar{X}$ of head arguments.
   We show that $Q_{\mathcal{V}}$ is equivalent to Q by showing that there is a pair of containment mappings, μ from the expansion $Q_{\mathcal{V}}^{exp}$ of $Q_{\mathcal{V}}$ to Q and ν in the opposite direction. We first observe that, by construction of $\mathcal{V}$, $Q_{\mathcal{V}}^{exp}$ can be obtained by applying to each view expansion W_i^{exp} in $Q_{\mathcal{W}}^{exp}$ the mapping θ_i (with suitable variable renamings) that we constructed above to obtain V_i from W_i.
   (a) *Existence of μ:* Consider an arbitrary containment mapping μ' from the expansion $Q_{\mathcal{W}}^{exp}$ of $Q_{\mathcal{W}}$ to Q. (We know that μ' exists as $Q \sqsubseteq Q_{\mathcal{W}}$.) As θ_i is an mgu for each view W_i in $\mathcal{W}$, there must exist a mapping μ that, composed with $\theta_i s$, gives μ'. (Because Q does not have self-joins, for any view atom U in any equivalent rewriting R of Q, the image in Q of the expansion of U in R^{exp} does not have self-joins.)
   (b) *Existence of ν:* Because $Q_{\mathcal{W}} \sqsubseteq Q$, then $Q_{\mathcal{W}}^{exp} \sqsubseteq Q$. Since by construction of $Q_{\mathcal{V}}$ we have $Q_{\mathcal{V}}^{exp} \sqsubseteq Q_{\mathcal{W}}^{exp}$, then $Q_{\mathcal{V}}^{exp} \sqsubseteq Q$. Therefore, there is a containment mapping ν from Q to the expansion $Q_{\mathcal{V}}^{exp}$.

2. *$\mathcal{V}$ minimizes the total cost of evaluating the queries in $\mathcal{Q}$ on the database $\mathcal{D}_{\mathcal{V}}$.* Indeed, for any optimal plan $Q(\mathcal{D}_{\mathcal{W}})$ for a query $Q \in \mathcal{Q}$ on the database $\mathcal{D}_{\mathcal{W}}$, we construct a plan $Q(\mathcal{D}_{\mathcal{V}})$, similarly to the procedure in item 1 above, as follows. For each occurrence $W_i(\bar{X})$ of a view W_i in the plan $Q(\mathcal{D}_{\mathcal{W}})$, we replace $W_i(\bar{X})$ by an occurrence of the corresponding view $V_i \in \mathcal{V}$, with *the same* list $\bar{X}$ of head arguments. We saw in item 1 that $Q(\mathcal{D}_{\mathcal{W}})$ and $Q(\mathcal{D}_{\mathcal{V}})$ compute relations that are the same as sets. Recall that each query V_i is contained in W_i. Thus, under any model for computing the costs of query plans, such that the model satisfies the condition that the cost increases with the increase in size of the participating relations (such models include well-known standard cost models as well as our sum-cost model), the cost of $Q(\mathcal{D}_{\mathcal{V}})$ does not exceed the cost of $Q(\mathcal{D}_{\mathcal{W}})$.

The two observations conclude our proof that the viewset $\mathcal{V}$ is an optimal viewset for the given problem input $\mathcal{P}$. □

Theorem 1 states that whenever workload queries have no self-joins then there exist optimal viewsets whose view definitions do not have self-joins. The following corollary goes one step further, by showing that each view in the optimal viewset has no more subgoals than the workload queries.

Corollary 1 *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ where $\mathcal{L}$ represents a single storage-limit constraint and assuming that*

1. *all queries in $\mathcal{Q}$ are without self-joins,*
2. *the number of subgoals in any query does not exceed an integer n,*
3. *there exists an optimal viewset $\mathcal{W}$ for $\mathcal{P}$ under storage limit constraint $\mathcal{L}$,*

then there exists an optimal viewset $\mathcal{V}$ of $\mathcal{P}$ under $\mathcal{L}$, such that for every view V in $\mathcal{V}$, the number of subgoals of V is bounded from above by n.

Proof Consider a problem input $\mathcal{P}$ that satisfies the premises of the corollary. From Theorem 1, we conclude that there exists an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ under $\mathcal{L}$, such that each view in $\mathcal{V}$ does not have self-joins. Since $\mathcal{V}$ is admissible, for each view V in $\mathcal{V}$ there exists an equivalent rewriting for some query $Q \in \mathcal{Q}$ using V and thus there must exist a containment mapping from the definition of V to (some part of) the definition of the query Q. We can thus conclude that each view in the viewset $\mathcal{V}$ has at most as many subgoals as some query in $\mathcal{Q}$. □

The optimal viewset in Corollary 1 may include filtering views along with containment-target views. Moreover, even an exponential number of filtering views may be necessary under set semantics; see [11]. Note that we cannot strengthen the Corollary 1 to state that under the premises of the corollary there exists an optimal viewset $\mathcal{V}$, such that each view in $\mathcal{V}$ is defined as a subexpression of some query body in the input query workload. As a counterexample, consider Example 3.

Example 3 Consider a query workload $\mathcal{Q} = \{Q_1, Q_2\}$, where:

$$Q_1 : q_1(X, Y, Z) \text{ :- } p(X, X), s(X, Y), t(Y, Z).$$
$$Q_2 : q_2(X, Y, Z) \text{ :- } p(X, Y), s(Y, Y), t(Y, Z).$$

Suppose that we are given a database $\mathcal{D}$ = { `p(a,a)`, `p(a,b)`, `p(c,c)`, `s(a,a)`, `s(b,b)`, `s(c,b)`, `t(a,d)`, `t(a,f)`, `t(b,g)`, `t(b,h)`} and a set of constraints

$\mathcal{L} = \{ L \}$, where the value of the storage limit $L = 6$ is an upper bound on the sizes of materialized views on $\mathcal{D}$. Consider a view V:

$$V : v(Z, T, W, U) \text{ :- } p(Z, T), s(T, W), t(W, U).$$

Note that the view V is not a subexpression of either query in the workload $\mathcal{Q}$. However, $\mathcal{V} = \{ V \}$ is an optimal viewset for the problem input ($\{ P(A, B), S(C, D), T(E, F) \}$, $\{Q_1, Q_2\}$, $\mathcal{D}$, $\{ L \}$), and both input queries can be rewritten using the view V:

$$\begin{aligned} Q_1' &: q_1(X, Y, Z) \text{ :- } v(X, X, Y, Z). \\ Q_2' &: q_2(X, Y, Z) \text{ :- } v(X, Y, Y, Z). \end{aligned}$$

It is important to note that when queries have self-joins, the number of subgoals of views can be up to a product of the number of subgoals of the queries. We show here an example of this effect for containment-target views; for a similar point for *filtering* views, see Example 1 in [11].

Example 4 Consider queries Q_6, Q_{10}, and Q_{15}:

$$\begin{aligned} Q_6 : q_6(X, Y) &\text{ :- } p(X, Y), p(Y, Z_1), p(Z_1, Z_2), p(Z_2, Z_3), \\ &\quad p(Z_3, Z_4), p(Z_4, X). \\ Q_{10} : q_{10}(X, Y) &\text{ :- } p(X, Y), p(Y, Z_1), p(Z_1, Z_2), \ldots, p(Z_8, X). \\ Q_{15} : q_{15}(X, Y) &\text{ :- } p(X, Y), p(Y, Z_1), p(Z_1, Z_2), \ldots, p(Z_{13}, X). \end{aligned}$$

Here, each query Q_i represents a cycle of length i. Let a view V_{30} represent a cycle of length 30:

$$V_{30} : v_{30}(X, Y) \text{ :- } p(X, Y), p(Y, Z_1), p(Z_1, Z_2), \ldots, p(Z_{28}, X).$$

As 30 is the least common multiple of 6, 10, and 15, each of Q_6, Q_{10}, Q_{15} can be equivalently rewritten using V_{30} as the only containment-target view:

$$\begin{aligned} Q_6' : r_6(X, Y) &\text{ :- } v_{30}(X, Y), v_{30}(Y, Z_1), v_{30}(Z_1, Z_2), v_{30}(Z_2, Z_3), \\ &\quad v_{30}(Z_3, Z_4), v_{30}(Z_4, X). \\ Q_{10}' : r_{10}(X, Y) &\text{ :- } v_{30}(X, Y), v_{30}(Y, Z_1), v_{30}(Z_1, Z_2), \ldots, v_{30}(Z_8, X). \\ Q_{15}' : r_{15}(X, Y) &\text{ :- } v_{30}(X, Y), v_{30}(Y, Z_1), v_{30}(Z_1, Z_2), \ldots, v_{30}(Z_{13}, X). \end{aligned}$$

Consider a database $\mathcal{D}$ and a storage limit L that equals the size of the relation for the view V_{30} on $\mathcal{D}$. Then the viewset $\mathcal{V} = \{ V_{30} \}$ is an optimal viewset for the problem input ($\{ P(A, B) \}$, $\{ Q_6, Q_{10}, Q_{15} \}$, $\mathcal{D}$, $\{ L \}$). Consider a database that has $3k$ disjoint cycles of which k are of length 2, k of length 3, and k of length 5, and suppose the storage limit on the materialized views is $3k$ tuples. (In case we measure the storage limit in bytes, we set the storage limit to $6k$ integer values and require that the database be purely integer valued; the corresponding modification of the analysis in the remainder of this paragraph is straightforward.) On this database, it is easy to see that the size of each of the views Q_6, Q_{10}, and Q_{15} is $2k$ tuples, and that the size of the view V_{30} (or of any view W_m that is a cycle whose length m is a multiple of 30) is $3k$. Thus, V_{30} (or any such view W_m) is the only view that (1) can rewrite the queries Q_6, Q_{10}, Q_{15}, and (2) satisfies the given storage limit. This observation on the view V_{30} (and on any view W_m as defined above) also holds for many other problem inputs to view selection—that is, for many other types of databases and storage limits.

This example can be extended naturally to construct workload queries for cycles of lengths that are all pairwise products of prime numbers $k_1, \ldots, k_m$, to show that a view V_m for a cycle of length $\Pi_{i=1}^{m} k_i$ constitutes the only possible optimal solution for problem inputs with those workloads and with appropriately chosen values of the storage limit. In all these examples, the number of subgoals in the view V_m is *proportional to the product* of the number of subgoals of the workload queries. Thus, we have shown that the number of subgoals in a containment-target view can be up to the product of the numbers of subgoals of the workload queries.

3.2 Rewritings without self-joins

While the results in Sect. 3.1 refer to the structure of the views in an optimal viewset, in this section we are interested in the structure of the query rewritings provided that the queries in the workload $\mathcal{Q}$ do not have self-joins. We show that there exists an optimal viewset $\mathcal{V}$, such that rewriting any query in $\mathcal{Q}$ does not require self-joins of containment-target views in $\mathcal{V}$. The following theorem states this claim.

Theorem 2 *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ and assuming that the queries in $\mathcal{Q}$ do not have self-joins, if there exists an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ under the storage limit $\mathcal{L}$, then it is possible to rewrite each query in $\mathcal{Q}$ using $\mathcal{V}$* without *self-joins of containment-target views.*

Proof The proof is given for rewritings that do not use filtering views. Let $\mathcal{V}$ be an optimal viewset for a given problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$. Let Q be a query in $\mathcal{Q}$, let R be an equivalent rewriting of Q in terms of $\mathcal{V}$, such that (1) R does not have filtering views, and (2) R has one or more self-joins of a containment-target view $V \in \mathcal{V}$. We will prove that we can construct another equivalent (under set semantics) rewriting $\hat{R}$ of Q, by removing from R all but one instance of any such view V.

Let Q be of the form:

$$Q(\bar{X}) \text{ :- } P_1(\bar{Y}_1), \ldots, P_n(\bar{Y}_n).$$

Note that Q is a minimized query, because Q does not have self-joins. Suppose that the containment-target view V is used at least twice in the equivalent rewriting R of Q. We assume that any other containment-target view for R occurs in R only once; it is straightforward to extend the proof to the case of self-joins of multiple containment-target views in query rewritings. Suppose there are m occurrences of V in R. Then R is of the form:

$$R(\bar{X}) \text{ :- } V^{(1)}(\bar{X}_1), \ldots, V^{(m)}(\bar{X}_m), G.$$

where each of $\bar{X}$, $\bar{X}_1$, ..., $\bar{X}_m$ is a list of arguments, $V^{(i)}$ ($i \in \{1, \ldots, m\}$) are occurrences of the view V in R, and G represents the instances of other views in R that are not V. (For convenience, we assume that the heads of Q and R have the same list $\bar{X}$ of arguments.)

Because $R \equiv_s Q$, $R^{exp} \equiv_s Q$, where R^{exp} is an expansion of R. Let S be the body of the query that results from minimizing R^{exp}. Thus, since there exists a mapping from Q to R^{exp} that maps the body of Q into S and vice versa, we can rename some variables of R^{exp} in such a way that S is exactly the body of Q. Note that the heads of R^{exp} and of Q already have the same list of variables. In the remainder of the proof, we consider the definition of R^{exp} whose subset S of subgoals is exactly the body of Q. Because of this choice, the containment mapping μ from Q to R^{exp} (which always exists [6] as $R^{exp} \sqsubseteq_s Q$), is an identity mapping. Consider now the set S_V of the atoms in S that belong to the expansions of the atoms $V^{(i)}$ ($i \in \{1, \ldots, m\}$). There are two cases:

Case 1: All atoms in S_V belong to the expansion of the same view atom $V^{(l)}$, with $1 \leq l \leq m$. Then $\hat{R}$, where:

$$\hat{R}: \ \hat{R}(\bar{X}) \text{ :- } V^{(l)}(\bar{X}_1), G.$$

is an equivalent rewriting of Q using $\mathcal{V}$ (as $\hat{R}^{exp}$ is isomorphic to Q.)

Case 2: The atoms in S_V belong to the expansion of at least two view occurrences of the view V in R. We will prove that we can construct from μ a new containment mapping μ' such that (1) μ' maps all atoms in S_V into one instance (say $V^{(1)}$) of V, and (2) μ' is a containment mapping from Q to R^{exp}. Let $V^{(j)}$, with $j \in \{2, \ldots, m\}$, be an instance of V in R and let $\mathcal{S}_j = \{P_1^{(j)}, \ldots, P_{k_j}^{(j)}\}$ be the set containing all the subgoals in S_V that occur in the expansion of $V^{(j)}$. (Recall that we use a definition of R^{exp} whose subset S of subgoals is exactly the body of Q.) Note that for any such set of subgoals $\mathcal{S}_j$, the variables in the atoms of $\mathcal{S}_j$ that occur also in Q *outside* $\mathcal{S}_j$ (i.e. the variables that appear either in the head of Q or in the subgoals of Q that are not in $\mathcal{S}_j$) must be head arguments of V. (Otherwise R will not be an equivalent rewriting of Q – recall that in expanding a rewriting, we use fresh variable names for all nondistinguished variables of each view.) Suppose that $S^{(j)}$ is nonempty, thus some subgoals of Q are in the expansion of $V^{(j)}$ in R^{exp}. Consider also the expansion of $V^{(1)}$ in R^{exp}. We will show how to change μ to “redirect" $\mathcal{S}_j$ from the expansion of $V^{(j)}$ to the expansion of $V^{(1)}$. Notice that $V^{(1)}$ has been chosen arbitrarily and thus in its place we can choose any atom in $V^{(i)}$ ($i \in \{1, \ldots, m\}$). As all the subgoals $P_1^{(j)}, \ldots, P_{k_j}^{(j)}$ in $\mathcal{S}_j$ (in expansion of $V^{(j)}$) are in the minimized version of R^{exp}, and as the expansion of $V^{(1)}$ has a set $\mathcal{S}_j^{(1)}$ of subgoals with the same names as in $\mathcal{S}_j$ (since $V^{(1)}$ and $V^{(j)}$ are instances of the same view V), there must be a containment mapping ν_j from $\mathcal{S}_j^{(1)}$ to $\mathcal{S}_j$. We apply ν_j to R^{exp}, to obtain a new query $\hat{R}^{exp}\nu_j$. Because the variables that occur *outside* $\mathcal{S}_j$ are head arguments of V (and thus of $V^{(1)}$ and of $V^{(j)}$), we are guaranteed that the only nonhead arguments of views in R^{exp} that are different between $\hat{R}^{exp}\nu_j$ and R^{exp} are nonhead arguments (in view $V^{(1)}$) of the set of subgoals $\mathcal{S}_j^{(1)}$. Note that now $V^{(1)}$ in $\hat{R}^{exp}\nu_j$ has the images of both $\mathcal{S}_j$ and $\mathcal{S}_1$ in Q, and thus we can remove $V^{(j)}$ from $\hat{R}\nu_j$, to obtain a query $R\nu_j$ whose expansion is contained (under a composition of μ and ν_j) in Q. (The expansion $R\nu_j$ also contains Q, using any containment mapping from R^{exp} to Q; thus, $R\nu_j \equiv_s Q$.)

In this way, we have obtained a new equivalent rewriting $R\nu_j$ of Q that does not have the subgoal $V^{(j)}$ of R. In $R\nu_j$, we consider any instance V^i, $i > 1$, of the view V, such that the expansion of $V^{(i)}$ in $R^{exp}\nu_j$ has a nonempty subset $\mathcal{S}_i$ of subgoals of the query Q, and apply the procedure described above to generate a copy of $\mathcal{S}_i$ in $V^{(1)}$. We repeat the procedure until all subgoals in $\mathcal{S}_2 \bigcup \ldots \bigcup \mathcal{S}_m$ can also be found in the expansion of $V^{(1)}$. The final result of these iterations is a new equivalent rewriting $\hat{R} = R\nu_{i_1}\nu_{i_2}\ldots\nu_{i_{m-1}}$ of Q, such that a self-join of m occurrences of V in R is replaced by just one occurrence of V in $\hat{R}$. It is easy to see that the rewriting obtained at the end of this process:

$$\hat{R}(\bar{X}) \text{ :- } V^{(1)}(\bar{X}_1), G.$$

is an equivalent rewriting of Q using $\mathcal{V}$. Note that as $\hat{R}^{exp}$ is R^{exp} with some subgoals removed, a containment mapping from $\hat{R}^{exp}$ to Q can be constructed by taking an appropriate subset of any mapping from R^{exp} to Q.

Finally, notice that the cost of computing $\hat{R}$ does not exceed the cost of computing R as $\hat{R}$ uses fewer occurrences of the same view atoms than R, and our cost model is monotonic with respect to query containment, i.e., if we replace a view V_1 in a plan by a view V_2 s.t. V_2

is contained in V_1 then the cost of the plan with V_2 is not greater than the plan with the view V_1. □

The following example shows how an equivalent rewriting that does not contain self-joins of views is obtained from an equivalent rewriting with self-joins by applying the procedure used in the proof of Theorem 2.

Example 5 Consider a query Q and a view V, where:

$$Q: \; q(X, Y) \text{ :- } p(X, T), r(T, Z), s(Z, Y).$$
$$V: \; v(A, B, C) \text{ :- } p(A, D), r(D, B), s(B, C).$$

An equivalent rewriting of Q using V which has self-joins is the following:

$$R: \; r(X, Y) \text{ :- } v^{(1)}(X, Z, U), v^{(2)}(X, Z, Y).$$

Now the expansion R^{exp} of R is:

$$\begin{aligned} R^{exp}: \; r(X, Y) \text{ :- } & p^{(1)}(X, T), r^{(1)}(T, Z), s^{(1)}(Z, U), \\ & p^{(2)}(X, W), r^{(2)}(W, Z), s^{(2)}(Z, Y). \end{aligned}$$

From R^{exp}, we obtain the sets $\mathcal{S}_1 = \{p^{(1)}(X, T), \; r^{(1)}(T, Z)\}$ and $\mathcal{S}_2 = \{s^{(2)}(Z, Y)\}$ such that $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) contains the subgoals that map into $V^{(1)}$ (resp. $V^{(2)}$). We obtain $R'\nu_2$ by applying to R^{exp} a mapping $\nu_2 = \{U \to Y\}$ and by then removing $V^{(2)}$ from the resulting query.

The lack of self-joins in queries is an essential condition in Theorem 2: Otherwise, as shown in Example 4, nontrivial self-joins of containment-target views may be required. An analogous result holds for filtering views (cf. Example 1 in [11]).

3.3 Complexity

We now consider the decision version of the view-selection problem, that is, given a set-oriented problem input $\mathcal{P}$ and a positive integer K, the problem is to determine whether there exists an admissible viewset $\mathcal{V}$ for $\mathcal{P}$, such that the cost of evaluating the queries $\mathcal{Q}$ in $\mathcal{P}$ using $\mathcal{V}$ does not exceed K. We show that this problem is in NP for workloads of queries without self-joins, provided that filtering views are not used in query rewritings. We prove the result for the *oracle* version of problem inputs, that is, we show that the size of a witness is polynomial in the size of the following components of the problem input: database schema, query workload, and constraints on the materialized views. This result is stronger than proving that the size of a witness is polynomial in the size of the above components plus the size of an input database, because database schemas, query workloads, and constraints on the materialized views are typically small in size compared to the size of possible databases conforming to the schemas. To prove the main result, we first establish an upper bound on the number of containment-target views in query rewritings.

Lemma 1 [23] *Let Q be a conjunctive query and $\mathcal{V}$ be a set of views, both Q and $\mathcal{V}$ without built-in predicates. If the body of Q has n relational subgoals and R is a locally minimal equivalent conjunctive rewriting of Q using $\mathcal{V}$, then R has at most n relational subgoals.*

The following proposition follows immediately from Lemma 1.

Proposition 1 *For any conjunctive query Q with n relational subgoals and for any locally minimal conjunctive rewriting R of Q in terms of views such that $R \equiv_s Q$, the number of containment-target views in R does not exceed n.*

This proposition is based on the observations that (a) any locally minimal rewriting does not contain filtering views, and (b) a containment-target view should "cover" at least one query subgoal.

Corollary 2 *For any set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ with any set of constraints $\mathcal{L}$, and for any optimal nonredundant viewset $\mathcal{V}$ for $\mathcal{P}$, the number of containment-target views in $\mathcal{V}$ does not exceed n, where n is the total number of relational subgoals in all the queries in the query workload $\mathcal{Q}$ in $\mathcal{P}$.*

The upper bound n in the above Corollary is based on the worst-case scenario where each atom appearing in a query body in the workload $\mathcal{Q}$ is covered by a different view in $\mathcal{V}$.

Notice that optimality here is due to our cost model. That is, if we add redundant containment-target views to a rewriting R (to obtain R'), then the cost of evaluating R' can be higher under our cost model than the cost of evaluating R. Theorem 3 is based on Proposition 1.

Theorem 3 *Given an* oracle version *of a conjunctive set-oriented problem input $\mathcal{P}$ whose queries are without self-joins, the decision version of the view-selection problem is in NP, provided that rewritings do not include filtering views.*

Proof Let $\mathcal{P}$ be an oracle version of a problem input as in Theorem 1, and let N be a positive natural number. Suppose we are given a witness, which comprises: (1) a viewset $\mathcal{V}$, (2) locally-minimal rewritings of all the input workload queries in terms of $\mathcal{V}$, and (3) a pair of containment mappings between each query and its corresponding rewriting.

Our goal is to check whether the viewset $\mathcal{V}$ provides admissible rewritings of the workload queries and gives a solution whose query-evaluating cost, on all the databases described by the input oracle, does not exceed the value N. Note that the answer to the query-evaluation-cost part of the check is "yes" if and only if there exists an *optimal* viewset for $\mathcal{P}$ whose query-evaluation-cost does not exceed N.

If $\mathcal{V}$ is an optimal solution for the problem input $\mathcal{P}$, we can do both checks in polynomial time in the size of $\mathcal{P}$, because of the following two reasons:

1. By Corollary 1, there exists an optimal viewset for $\mathcal{P}$, such that every view in the viewset has at most n subgoals, where n is the number of subgoals in the longest query definition in $\mathcal{P}$, and
2. By Proposition 1, for any locally-minimal rewriting that is set-equivalent to an input query, the number of views in the rewriting does not exceed the number of subgoals in the query. □

Note that if filtering views are allowed in query rewritings, then the view-selection problem under set semantics has an exponential-time lower bound even when none of the workload queries have self-joins (for more details see [11]).

4 Queries under bag semantics

In this section we consider the view-selection problem under bag semantics. We begin by summarizing the main results:

1. Under bag semantics, every candidate query rewriting does not have either filtering views or redundant containment-target views (Sect. 4.1).
2. For workloads of queries with or without self-joins, each view definition in any admissible viewset (and thus in any optimal viewset) is obtained as a generalization of a subexpression of some query in the workload. As a consequence, each view definition has no more subgoals than any query in the input workload (Sect. 4.2). Moreover, for workloads of queries without self-joins each view definition in an admissible viewset can be defined without self-joins.
3. For workloads of queries without self-joins and for any admissible viewset, rewriting any query in the workload does not require self-joins of view atoms (Sect. 4.3).
4. The decision version of the view-selection problem is in NP for workloads of queries with or without self-joins and for a single storage-limit constraint on materialized views (Sect. 4.4; see also [2]).

Comparing these results with those in Sect. 3, we conclude that both constructing admissible/optimal viewsets and rewriting queries using views are easier problems under bag semantics than under set semantics.

4.1 Filtering views do not exist under bag semantics

Recall the definitions of containment-target and filtering views in Sect. 2.2; those definitions were given for the case where queries are evaluated under *set* semantics. In this section we consider the *bag*-semantics case; interestingly, under bag semantics filtering views do not exist. Indeed, for any conjunctive rewriting R that is bag-equivalent to a conjunctive query Q, R is a locally minimal rewriting; this property stems from the isomorphism, rather than containment-mapping, requirement on query equivalence:

Theorem 4 (Theorem 5.2. of [8]) *Let Q and Q' be conjunctive queries. Q and Q' are equivalent under* bag semantics *iff Q and Q' are isomorphic.*

An immediate consequence of the above theorem is that under bag semantics, any candidate (i.e., equivalent) query rewriting lacks any filtering views, as well as any redundant containment-target views:

Corollary 3 *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented problem input, $\mathcal{V}$ be any* admissible *viewset for $\mathcal{P}$, and Q be any query in $\mathcal{Q}$. If R is an equivalent rewriting of Q in terms of $\mathcal{V}$, then R does not use filtering views or redundant containment-target views.*

4.2 Bounded number of subgoals

The following lemma holds for workloads of queries without *or with* self-joins under bag semantics and for arbitrary sets of constraints on materialized views.

Lemma 2 *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented problem input, where $\mathcal{L}$ is a set of* any *constraints. Let $\mathcal{V}$ be any* admissible *viewset for $\mathcal{P}$, and let Q be any query in $\mathcal{Q}$. Suppose $\mathcal{V}' \subseteq \mathcal{V}$ is the set of all views used in the equivalent rewriting R of Q in terms of $\mathcal{V}$. Then the atoms in the expansion of R corresponding to the definitions of views $\mathcal{V}'$ form a partition of the (body atoms in the) definition of Q.*

Proof This result follows trivially from the isomorphism requirement on query equivalence under bag semantics [8]. Under this requirement, for any rewriting in terms of views that is equivalent to a conjunctive query, the expansion of the rewriting is isomorphic to the query. □

We can make a more precise statement concerning how the view definitions are related to the form of the queries. For this we need the following definition.

Definition 5 We say that an expression E is a *generalization* of an expression E' if and only if E' is an instance of E—that is, there is a substitution θ such that $E' = E\theta$.

The following Lemma 3 follows easily from Lemma 2.

Lemma 3 *Assume a conjunctive bag-oriented problem input $\mathcal{P}$, and let $\mathcal{V}$ be any admissible viewset for $\mathcal{P}$. Then each view in $\mathcal{V}$ can be defined as a generalization of a subexpression of some query in the input workload $\mathcal{Q}$.*

Proof Let $V \in \mathcal{V}$. Since $\mathcal{V}$ is admissible, there is a query $Q \in \mathcal{Q}$ and an equivalent rewriting R of Q that uses V. Consider the expansion R^{exp} of R. Since $R^{exp} \equiv_b Q$, it follows that R^{exp} is isomorphic to Q. Therefore, the subexpression S of R^{exp} corresponding to the definition of V is isomorphic to a subexpression of Q. But S is an instance of the body of V. Therefore, the body of V is a generalization of a subexpression of the query Q. □

The following Corollaries follow directly from Lemmas 2 and 3, and refer to the number of subgoals in view definitions for views in admissible viewsets.

Corollary 4 *Given a conjunctive bag-oriented problem input $\mathcal{P}$, let $\mathcal{V}$ be any admissible viewset for $\mathcal{P}$. Then each view in $\mathcal{V}$ has at most n subgoals, where n is the number of subgoals in the longest query in the input workload $\mathcal{Q}$.*

It should be noted that it is not obvious at first sight that if the expansion of a view atom in a rewriting has at most n atoms, then the original view definition also has at most n atoms. Note that while this is true under bag semantics (as we cannot remove duplicate subgoals since this destroys equivalence), this does not hold under set semantics. This is the reason that a similar result as the above holds for set semantics only for queries without self-joins (see Corollary 1).

Corollary 5 *Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and let V be any view in any* admissible *viewset $\mathcal{V}$ for $\mathcal{P}$. Suppose V is used in rewriting queries $Q_{i_1}, \ldots, Q_{i_k}$ in $\mathcal{Q}$; let m be the number of subgoals in the shortest definition among the definitions of $Q_{i_1}, \ldots, Q_{i_k}$. Then V can be defined using at most m subgoals.*

Corollary 6 *Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and $\mathcal{V}$ an admissible viewset for $\mathcal{P}$. Assuming that queries in $\mathcal{Q}$ do not have self-joins, then each view in $\mathcal{V}$ can be defined as a conjunctive query without self-joins.*

4.3 Rewritings without self-joins of views

Analogously to the case of set semantics, in the case of bag semantics we can show that for problem inputs $\mathcal{P}$ whose query workloads $\mathcal{Q}$ do not have self-joins, and for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, rewriting any query in $\mathcal{Q}$ does not require self-joins of views in $\mathcal{V}$.

Theorem 5 *Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and $\mathcal{V}$ an admissible viewset for $\mathcal{P}$. Assuming that queries in $\mathcal{Q}$ do not have self-joins, then it is possible to rewrite each query in $\mathcal{Q}$ without using self-joins of views in $\mathcal{V}$.*

Proof The proof follows directly from Lemma 2. □

4.4 Complexity

In this section we show that the decision version of the view-selection problem is in NP for a single storage-limit constraint on materialized views (see also [2]). We define the decision version of the problem and state the result for the *oracle* version of problem inputs, in a way similar to the respective formulations in Sect. 3.3. At the same time, unlike the results in Sect. 3.3, the NP results for bag semantics hold for workloads of queries without *or with* self-joins.

We first establish an analog of Proposition 1 in Sect. 3.3:

Proposition 2 *For any conjunctive query Q with n relational subgoals and for any conjunctive rewriting R of Q in terms of views, such that $R \equiv_b Q$, the number of views in R does not exceed n.*

This result follows immediately from the fact that for any equivalent (under bag semantics) rewriting to a query, the rewriting does not contain filtering views or "unnecessary" containment-target views, and is thus locally minimal.

We now establish Theorem 6, which is a direct consequence of the following observation: under bag semantics, for any problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ with any set of constraints $\mathcal{L}$, and for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, the number of views in $\mathcal{V}$ does not exceed n, where n is the total number of relational subgoals in all the queries in the query workload $\mathcal{Q}$ in $\mathcal{P}$.

Theorem 6 *Given an* oracle version *of a conjunctive, bag-oriented problem input $\mathcal{P}$ and assuming that the input set of constraints $\mathcal{L}$ represents a single storage limit, the decision version of the view-selection problem is in NP.*

Proof Let $\mathcal{P}$ be an oracle version of a problem input as in Lemma 2, and let N be a number. Suppose we are given a witness, which comprises: (1) a viewset $\mathcal{V}$, (2) locally-minimal rewritings of all the input workload queries in terms of $\mathcal{V}$, and (3) bijection mappings between each query and its corresponding rewriting.

Our goal is to check whether the viewset $\mathcal{V}$ provides admissible rewritings of the workload queries and gives a solution whose query-evaluating cost, on all the databases described by the input oracle, does not exceed the value N. Note that the answer to the query-evaluation-cost part of the check is "yes" if and only if there exists an *optimal* viewset for $\mathcal{P}$ whose query-evaluation-cost does not exceed N.

If $\mathcal{V}$ is an optimal viewset for the problem input $\mathcal{P}$, we can do both checks in the time polynomial in the size of $\mathcal{P}$, because:

1. By Corollary 4, for any admissible viewset for $\mathcal{P}$, every view in the viewset has at most n subgoals, where n is the number of subgoals in the longest query definition in $\mathcal{P}$, and
2. From Proposition 2, for any rewriting that is bag-equivalent to an input query, the number of views in the rewriting does not exceed the number of subgoals in the query. □

5 Queries under bag-set semantics

In this section we consider the view-selection problem under bag-set semantics. Note that all queries mentioned in the results below may have self-joins unless it is stated otherwise. The main results of this section are as follows:

1. In Sect. 5.1 we demonstrate that for queries *with* self-joins, under bag-set semantics the definition of a view V may have *exponentially* more subgoals than the definition of the respective query Q.

2. While under bag-set semantics equivalent rewritings of queries may have filtering views, there always exist optimal rewritings without filtering views (Sect. 5.2).
3. The results in Sect. 5.3 are similar to those in Sect. 4.2, which concern bag semantics. That is, we show that for workloads of queries without self-joins, each view definition in any admissible viewset (and thus in any optimal viewset) has no more subgoals than any query in the input workload. As a consequence, for workloads of queries *without* self-joins, each view definition in an admissible viewset can be defined without self-joins. Moreover, we can show that for workloads of queries without self-joins and for any admissible viewset, rewriting any query in the workload does not require self-joins of view atoms.
4. In Sect. 5.4 we show that the decision version of the view-selection problem is in NP for workloads of queries without self-joins and for a single storage-limit constraint on materialized views.

5.1 Exponential size of views for queries with self-joins

In this section we demonstrate that for queries *with* self-joins, under bag-set semantics it is possible to construct admissible viewsets where, for a view V used in an equivalent rewriting R of an input query Q, the definition of V may have more subgoals than the definition of Q. In fact, as we show in Example 6, the definition of such a view V may have *exponentially* more subgoals than the definition of the respective query Q. At the moment, the character of the relationship between the size of a view-selection problem input and the size of the definitions of views in an optimal viewset for this input remains an open problem for queries and views with self-joins under bag-set semantics. Thus, the complexity of view selection under bag-set semantics for queries and views with self-joins is an open problem.

For queries with self-joins under bag-set semantics, for each natural number $k \geq 1$ it is possible to construct a problem instance $\mathcal{I}_k$ of view selection, such that the total size of view definitions in some admissible viewset $\mathcal{V}_k$ for $\mathcal{I}_k$ is exponential in the size of the input database schema $\mathcal{S}_k$ and of the input query workload $\mathcal{Q}_k$. Note that that size is *not* exponential in the size of the database instance $\mathcal{D}_k$ in the problem input $\mathcal{I}_k$; thus, this exponential behavior is also in the size of the entire problem input in the oracle version of view selection. We show the construction in Example 6:

Example 6 Let $k \geq 1$, and consider a problem input $\mathcal{I}_k = \{\mathcal{S}_k, \mathcal{D}_k, \mathcal{Q}_k, L_k\}$, where $\mathcal{D}_k$ is a database with schema $\mathcal{S}_k$, $\mathcal{Q}_k$ is a set of two queries with self-joins defined on $\mathcal{S}_k$, and L_k is a storage limit. Let $\mathcal{S}_k$ comprise a single relation schema for a relation P with $k+5$ attributes; the two queries Q_1^k and Q_2^k in the workload $\mathcal{Q}_k$ are defined as follows:

$$\begin{aligned} q_1^k(A, B, C, C_1, C_2, \ldots, C_k, D, F) \text{ :- } & p(A, A, B, C, C, \ldots, C, D, D), \\ & p(A, B, B, C_1, C_2, \ldots, C_k, D, F). \\ q_2^k(A, B, C, C_1, C_2, \ldots, C_k, D, F) \text{ :- } & p(A, B, B, C, C, \ldots, C, D, D), \\ & p(A, A, B, C_1, C_2, \ldots, C_k, D, F). \end{aligned}$$

Each query in the workload $\mathcal{Q}_k$ has a self-join of the relation P. The first subgoal of each query has a sequence, of length k, of attributes with the same value, all denoted by variable C. The only difference between the two queries is in the sequence of the patterns (AAB, ABB vs. ABB, AAB) of the first three attributes of the subgoals in the self-join.

Consider a view U_1:

$u_1(X_1, Y_1, X_2, Y_2, Z_2, T_{11}, \ldots, T_{1k}, T_{21}, \ldots, T_{2k}, T_{31}, \ldots, T_{3k}, W_1, W_2, W_3)$:-
$p(X_1, X_1, Y_1, T_{11}, \ldots, T_{1k}, W_1, W_2)$,
$p(X_1, Y_1, Y_1, T_{21}, \ldots, T_{2k}, W_1, W_3)$,
$p(X_2, Y_2, Z_2, T_{31}, \ldots, T_{3k}, W_1, W_1)$.

The view U_1 can be used by itself to construct rewritings of the two workload queries Q_1^k and Q_2^k, as follows:

$r_{1,1}(A, B, C, C_1, C_2, \ldots, C_k, D, F)$:-
$u_1(A, B, A, A, B, C, \ldots, C, C_1, \ldots, C_k, C, \ldots, C, D, D, F)$.
$r_{2,1}(A, B, C, C_1, C_2, \ldots, C_k, D, F)$:-
$u_1(A, B, A, B, B, C_1, \ldots, C_k, C, \ldots, C, C, \ldots, C, D, F, D)$.

Intuitively, in the rewriting $R_{1,1}$ ($R_{2,1}$, respectively) the first subgoal of the view U_1 covers the first subgoal of the query Q_1^k (the second subgoal of Q_2^k, respectively), the second subgoal of U_1 covers the second subgoal of Q_1^k (the first subgoal of Q_2^k, respectively), and the third subgoal of U_1 becomes an exact copy of the *first* subgoal of each of Q_1^k and Q_2^k in each rewriting $R_{i,1}$, $i = 1, 2$. Each rewriting $R_{i,1}$ is equivalent to the corresponding query Q_i^k under bag-set semantics because after removing exact duplicates of subgoals, the expansion of each rewriting is isomorphic to its respective query.

We use these observations on the view U_1 and on the rewritings $R_{1,1}$ and $R_{2,1}$ to construct a family of views $\{U_m\}$, $m \geq 1$; each view U_m can be used by itself to generate a rewriting $R_{1,m}$ of the query Q_1^k and a rewriting $R_{2,m}$ of the query Q_2^k, such that each rewriting is equivalent to the respective query under bag-set semantics. The idea in constructing the view U_m for each $m > 1$ is to add to the first two subgoals of the view U_1 all possible subgoals $p^{(m,1)}, p^{(m,2)}, \ldots, p^{(m,t)}$, where $t = t(k, m)$ is to be determined. Each subgoal $p^{(m,j)}$, $j \in \{1, \ldots, t\}$, is of the form $p^{(m,j)}(X_j, Y_j, Z_j, T_{j1}, \ldots, T_{jk}, W_1, W_1)$, such that exactly m variable names in $T_{j1}, \ldots, T_{jk}$ are equal to some prespecified variable name T (T is the same in all the subgoals $p^{(m,j)}$) that is not the same as any T_{is} or as any of $X_j, Y_j, Z_j, W_1, W_2, W_3$ (in the first two subgoals of U_m and in all the $p^{(m,j)}$). We require that all the attributes of each subgoal of each U_m be head variables of U_m.

For each m, the maximal possible number $t(k, m)$ of such subgoals $p^{(m,j)}$ is "k choose m". Observe that for $m = k/2$, this number $t(k, k/2)$ is exponential in k. Also observe that each U_m, $2 \leq m \leq k/2$, is *not* contained in any U_s where $s < m$; in addition, each U_m is *not* contained in a view U_0 whose definition has just the two first subgoals of the view U_1. Moreover, we can construct a database $\mathcal{D}_k$ on which the relation for $U_{k/2}$ has fewer rows than the relation for U_m for each $m < k/2$.

We can show that under bag-set semantics, each view U_m can be used by itself to generate an equivalent rewriting $R_{i,m}$ of each input query Q_i^k, $i = 1, 2$, because in the rewriting $R_{1,m}$ ($R_{2,m}$, respectively) we can instantiate the head variables of each U_m in such a way that we make each $p^{(m,j)}$ an exact copy of the *first* subgoal of Q_1^k (Q_2^k, respectively); to achieve this, in each $p^{(m,j)}$ we need to map T and each T_{jl} into C. In particular, the view $U_{k/2}$ gives us rewritings that are equivalent to the input queries under bag-set semantics.

We conclude that the viewset $\{U_{k/2}\}$ (and, more generally, the viewset $\{U_m\}$ for each $m \in \{1, \ldots, k/2\}$) is an admissible viewset for a problem input $\mathcal{I}_k = \{\mathcal{S}_k, \mathcal{D}_k, \mathcal{Q}_k, L_k\}$, where $\mathcal{D}_k$ is some database instance and where the storage limit L_k is the number of rows in the relation $U_{k/2}(\mathcal{D}_k)$ (or in the relation $U_m(\mathcal{D}_k)$, respectively). Note that the size of the definition of the admissible view $U_{k/2}$ is exponential in the size of the problem inputs $\mathcal{S}_k$ and $\mathcal{Q}_k$.

5.2 Filtering views are not needed under bag-set semantics

In this section we show that, similarly to the bag-semantics case, to find optimal viewsets under bag-set semantics we do not need to consider filtering views in query rewritings. We start by briefly reviewing the necessary and sufficient condition of bag-set equivalence of two conjunctive queries from [8]. For this the notion of *canonical representation* of a query is needed, where Q' is a canonical representation of a query Q if Q' is obtained by removing all duplicate atoms from Q.

Theorem 7 (Theorem 7.11 from [8]) *Let Q_1 and Q_2 be conjunctive queries. Q_1 and Q_2 are equivalent under* bag-set *semantics iff Q'_1 and Q'_2 are equivalent under* bag *semantics, where Q'_1 and Q'_2 are canonical representations of Q_1 and Q_2, respectively.*

The following result follows directly from Theorem 7. It says that for any view V, such that some occurrence of V is redundant in a rewriting, each subgoal of this occurrence of V in the expansion of the rewriting is an exact copy of a subgoal of another view in the expansion. In particular, that other view can be another occurrence of V in the rewriting.

Lemma 4 *Given a conjunctive query Q defined on a database schema $\mathcal{S}$, and assuming* bag-set semantics *for computing query answers, let $\mathcal{V}$ be a set of views and V be a view which may or may not belong to $\mathcal{V}$. Suppose that V and all views in $\mathcal{V}$ are conjunctive and are defined on the schema $\mathcal{S}$. Suppose also that R is an equivalent rewriting of Q in terms of $\mathcal{V}$. Let R' be a query that results from adding an atom $\bar{V}$ for V to the rewriting R. Then $R' \equiv_{bs} Q$ if and only if each subgoal of $\bar{V}$ in the expansion of R' is an exact duplicate of a subgoal in the expansion of some view atom $\bar{V}' \neq \bar{V}$ in the rewriting.*

For the next result, we introduce the notion of an *exposed atom* in a query: a relational atom L of a query Q is exposed in Q if all arguments of L are distinguished arguments of the query. The following result is a direct corollary of Lemma 4; it says that for any view V, such that some atom $\bar{V}$ of V is redundant in a rewriting, (1) each subgoal of V is exposed, and (2) for any other view atom $\bar{W}$ in the rewriting, such that some subgoal of $\bar{V}$ is a copy of a subgoal s of $\bar{W}$ in the expansion of the rewriting, that subgoal s is exposed in the view W. From (1) above it follows that for any view whose occurrence is redundant in some rewriting under bag-set semantics, all variables of the view are distinguished.

Corollary 7 *Given a conjunctive query Q defined on a database schema $\mathcal{S}$, and assuming* bag-set semantics *for computing query answers, let $\mathcal{V}$ be a set of views and V be a view that may or may not belong to $\mathcal{V}$. Suppose that V and all views in $\mathcal{V}$ are conjunctive and are defined on the schema $\mathcal{S}$. Suppose R is an equivalent rewriting of Q in terms of views in $\mathcal{V}$, and let $\bar{V}$ be a redundant view atom in R.*[5] *Let R' be a query that results from removing the atom $\bar{V}$ from the rewriting R; $R' \equiv_{bs} Q$. Then (1) each subgoal of the view V is exposed (thus, all variables of V are distinguished variables), and (2) for each relational atom $s(\bar{Y})$ in the expansion of $\bar{V}$ in R, there is an exposed atom $s(\bar{Y})$ in the expansion of at least one other view atom $\bar{W}$ in R (where $\bar{V}$ and $\bar{W}$ may or may not be atoms for the same view).*

Recall that locally minimal rewritings of queries do not have filtering views. In Lemma 5, we prove that if we only consider query plans that use left-linear join trees, then any rewriting with filtering views is at least as expensive to evaluate on a database as some locally minimal

[5] That is, removing $\bar{V}$ from R results in a rewriting that is equivalent to the query Q; cf. the notion of a locally-minimal rewriting in Sect. 3.3.

rewriting of the same query, on the same database. Moreover, that locally minimal rewriting can be defined by removing some view atoms from the rewriting with filtering views.

Before presenting Lemma 5, we first establish an auxiliary Proposition 3 which will be used in the proof of this Lemma. Proposition 3 uses the notion of *coverage* of a subgoal by another subgoal. Under bag-set semantics, for a view atom $\bar{V}$ in an equivalent rewriting R of Q, a subgoal s of $\bar{V}$ *covers* a subgoal t of Q if there exists a bijective mapping μ from Q to the expansion R^{exp} of R, where s is the image of t under μ.

Proposition 3 *For any conjunctive query Q and for its conjunctive rewriting R in terms of views $\mathcal{V}$[6], such that $Q \equiv_{bs} R$, let s be an exposed subgoal of a view atom $\bar{V}$ in R. Then there exists a subgoal t of the query Q, such that s in $\bar{V}$ in the rewriting R covers t in Q.*

Proof Without loss of generality under bag-set semantics, we assume that query and view definitions do not have self-joins of *identical* subgoals. Let R^{exp} be the expansion of the rewriting R; s is a subgoal in the expansion of $\bar{V}$ in R^{exp}. Let $Img_{\mu}(body(Q))$ be the image, in the body of R^{exp}, of the body of Q under a bijective mapping $\mu : Q \rightarrow R^{exp}$. There are two cases:

Case 1: $s \in Img_{\mu}(body(Q))$. In this case, by definition, s covers some subgoal of Q.

Case 2: s in $\bar{V}$ is an exact copy of some subgoal $s' \in Img_{\mu}(body(Q))$, where s' is not in $\bar{V}$ (see Lemma 4). By definition, s' covers some subgoal t of Q. In this case, the set $Img_{\mu}(body(Q))$ of subgoals of R^{exp} can be recast to include the subgoal s. We do the recasting by substituting s' in $Img_{\mu}(body(Q))$ by its exact copy s. As a result, s covers the subgoal t of Q. Note that the recasting does not involve any changes to the mapping μ. □

Notice that as we can see from the proof of the above proposition, the occurrence $\bar{V}$ of the view V in R can be a containment-target view w.r.t. the subgoal t of Q.

Lemma 5 *Given a database $\mathcal{D}$ with schema $\mathcal{S}$, given a conjunctive query Q on $\mathcal{D}$, and assuming* bag-set semantics *for computing query answers, suppose R is an equivalent rewriting of Q in terms of views $\mathcal{V}$[7], such that R uses filtering views. Then there exists a locally-minimal rewriting $R' \equiv_{bs} Q$, such that R' is defined as a proper subset of the view atoms in R, and the optimal cost of evaluating R' on $\mathcal{D}$ does not exceed the optimal cost of evaluating R on $\mathcal{D}$, provided all queries are evaluated using left-linear join trees.*

Proof Let Q be a query, $\mathcal{D}$ a database, and $\mathcal{V}$ a set of views. Consider a rewriting R of Q in terms of $\mathcal{V}$, such that R has filtering views. Suppose that Q has n subgoals. We will prove that in any optimal left-linear plan for evaluating Q using R, we can remove all but at most n view literals *without increasing the cost of evaluating the query*, and that the resulting plan corresponds to a locally minimal equivalent rewriting of Q.

Consider any query plan P for R on $\mathcal{D}$, such that P uses a left-linear join tree and is optimal for Q among all plans for R that use left-linear join trees. Without loss of generality, we assume that P is a join of k view atoms $\bar{V}_1, \bar{V}_2, \ldots, \bar{V}_k$ for views in $\mathcal{V}$ (each view in $\mathcal{V}$ can contribute more than one view atom to P), and that, further, P is a join, from left to right, of $\bar{V}_1, \bar{V}_2, \ldots, \bar{V}_k$:

$$P = (\ldots((\bar{V}_1 \bowtie \bar{V}_2) \bowtie \bar{V}_3) \bowtie \ \ldots) \bowtie \bar{V}_k).$$

We now transform the plan P into a plan P' that has at most n view atoms, where n is the number of subgoals of the query Q. To transform P into P', we examine the view atoms in

[6] Both Q and all views in $\mathcal{V}$ are defined on a database schema $\mathcal{S}$.

[7] All views in $\mathcal{V}$ are defined on the schema $\mathcal{S}$.

P from left to right and use *some* of the atoms in constructing P', in the same left-to-right order as in P. We prove by induction on the number of view atoms in P that after examining each view atom $\bar{V}_i$ in P and deciding whether it gets added to the partial plan for P', the following holds on *any* database (and thus on $\mathcal{D}$ in particular):

(i) computing the partial plan $\bar{V}_1 \bowtie \ldots \bowtie \bar{V}_i$ for P results in the same relation as computing the partial plan $\bar{V}_1 \bowtie \ldots \bowtie \bar{V}_j$, $j \leq i$, for P', and
(ii) computing the partial plan for P is at least as expensive as computing the partial plan for P' on the same database.

Basis of induction: Suppose that the view atom $\bar{V}_1$ in P covers subgoals $s_{11}, s_{12}, \ldots, s_{1i_1}$ of the query Q. Note that $i_1 \geq 1$, because, by Lemma 4, Corollary 7, and Proposition 3, any view atom in R can cover at least one subgoal of the query Q. We keep the view atom $\bar{V}_1$ in the plan P'. At this point, the partial plans for P and P' are identical; thus, both (i) and (ii) above hold.

Induction assumption and induction step: We assume that (i) and (ii) above hold after the step of considering the view atom $\bar{V}_{i-1}$, $i > 1$, in the plan P. At this point, let $\bar{V}_1, \ldots, \bar{V}_{i-m}$, $m > 0$, be all the view atoms so far in the plan P'. Unlike the current partial plan for P, some or all of the view atoms $\bar{V}_2, \ldots, \bar{V}_{i-1}$ may be missing in P' by construction. We now consider the view atom $\bar{V}_i$ in P and decide whether to add $\bar{V}_i$ to the partial plan for P'. There are two possibilities for $\bar{V}_i$:

Case 1: $\bar{V}_i$ can cover at least one subgoal $s_{i j_i}$ of Q, such that $s_{i j_i}$ is not already covered by the view atoms currently in P'. In that case, we add $\bar{V}_i$, as the rightmost join input in a left-linear tree, to the plan P' and to the new current partial plan for P. By induction assumption, (i) and (ii) hold in this case.

Case 2: $\bar{V}_i$ cannot cover any subgoal of Q that is not already covered by the view atoms currently in P'. By the induction assumption, the current partial plans for P and P' cover the same set of subgoals of Q. In this case, we do not add $\bar{V}_i$ to P', but do add $\bar{V}_i$ to the current partial plan for P, which thus becomes the join of view atoms $\bar{V}_1, \ldots, \bar{V}_i$. Note that the new partial plans for P and P' still cover the same set of subgoals of Q. By construction, the current (unchanged) plan P':

1. gives the same answer, under bag-set semantics, as the new partial plan for P;
2. has at most the same cost, on each database, as the new partial plan for P, by the induction assumption and because a join (with $\bar{V}_i$) is added to P but not to P'.

We also use the fact that under bag-set semantics, if a view atom $\bar{W}$ in a left-linear plan does not cover any new subgoals compared to the view atoms $\bar{U}_1, \ldots, \bar{U}_p$ to the left of $\bar{W}$, then $\bar{W}$ in the plan is a filtering view and thus all the subgoals of $\bar{W}$ are exact copies of some (exposed) subgoals of $\bar{U}_1, \ldots, \bar{U}_p$.

As P has a finite number of subgoals and as P' has, at each induction step, at most as many subgoals as the current partial plan for P, the process of constructing P' terminates in finite amount of time. This observation ends the induction proof. Note that by construction, the full final plan P' has at most n subgoals, where n is the number of subgoals in Q. In addition, P' does not have filtering views and thus by definition corresponds to a locally-minimal rewriting R' of Q, such that R' is defined as a subset of the view atoms in R.

By induction, the cost of evaluating P' on $\mathcal{D}$ does not exceed the optimal left-linear cost of evaluating R (i.e., the cost of P) on $\mathcal{D}$. As the cost of evaluating P' on $\mathcal{D}$ is greater than or equal to the optimal cost of evaluating R' on $\mathcal{D}$ using left-linear join trees (P' is not necessarily an optimal left-linear plan for R' on $\mathcal{D}$), the optimal left-linear cost of evaluating R' on $\mathcal{D}$ does not exceed the optimal left-linear cost of evaluating R on $\mathcal{D}$. □

The following Theorem 8 derives from Lemma 5.

Theorem 8 *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive, bag-set oriented problem input, where $\mathcal{L}$ is a set of* any *constraints*[8]*. To find an optimal viewset for the problem input $\mathcal{P}$ it is enough to consider locally minimal rewritings of the queries in $\mathcal{Q}$, provided only left-linear join trees are considered for query evaluation.*

Note the difference between the bag and set semantics cases: under bag semantics, all candidate rewritings (and not just some optimal rewritings) are locally minimal. Under set semantics, rewritings that are not locally minimal can be strictly cheaper to evaluate than their locally minimal counterparts; see, e.g., examples developed in [11] (in the proofs of results showing that the view-selection problem has an exponential-time lower bound under set semantics).

Example 7 Consider a query Q:

$$Q:\ q(X, Y) \text{ :- } p(X, Z), s(Z, Y).$$

Q can be equivalently rewritten using views V and W, as rewriting R_1 or R_2:

$$\begin{aligned}&V:\ v(A, B, C) \text{ :- } p(A, B), s(B, C).\\ &W:\ w(F, G) \text{ :- } s(F, G).\\ &R_1:\ r_1(X, Y) \text{ :- } v(X, Z, Y), w(Z, Y).\\ &R_2:\ r_2(X, Y) \text{ :- } v(X, Z, Y).\end{aligned}$$

The view V is a containment-target view that provides both subgoals of the query Q in rewritings of the query. The view W provides the second subgoal, S, of the query Q in rewritings of the query. Under bag-set semantics, both R_1 and R_2 are candidate (i.e., equivalent) rewritings. At the same time, unlike R_2, R_1 is neither an optimal nor a locally minimal rewriting of Q.

5.3 Bounded number of view subgoals for queries without self-joins

The results in this section hold for workloads of queries without self-joins under bag-set semantics and for arbitrary sets of constraints on materialized views.

Lemma 6 *Let $\mathcal{P}$ be a conjunctive, bag-set-oriented problem input whose queries do not have self-joins, and suppose that there is an optimal viewset $\mathcal{V}$ for $\mathcal{P}$. Then, there is an optimal viewset $\mathcal{V}'$ such that:*

1. *Each view definition in $\mathcal{V}'$ is a generalization of a subexpression of a query in the input workload $\mathcal{Q}$.*
2. *The view definitions in $\mathcal{V}'$ do not contain self-joins.*

Proof (sketch) We can construct $\mathcal{V}'$ from $\mathcal{V}$ as follows: for each view definition V in $\mathcal{V}$ (with self-joins) we create a view definition V' in $\mathcal{V}'$ (without self-joins) by replacing all atoms in V with the same predicate name by the atom obtained by applying their most general unifier (mgu) to one of them; we do this for all collections of subgoals of V that have the same predicate name. Notice that, since V must be used in an equivalent rewriting of a query Q, and Q does not have self-joins, then it is not possible for V to contain non-unifiable atoms

[8] Note that queries in $\mathcal{Q}$ may have self-joins, and that the set $\mathcal{L}$ may have any number of constraints of any type(s) (cf. assumptions in Sect. 3).

with the same predicate symbol. By construction, no view definition in $\mathcal{V}'$ has self-joins. Moreover, V' is contained in V under bag-set semantics.

Let Q be an input query and R be an equivalent rewriting of Q in terms of $\mathcal{V}$. Then the expansion R^{exp} of R is isomorphic to Q after removal of duplicate subgoals from R^{exp}. Notice that as Q does not contain self-joins, all atoms in R^{exp} with the same predicate are identical (duplicates) and therefore only a single occurrence of them is needed. By removing these duplicates we get R''. Suppose now that we replace in R each view tuple of each view V by a view tuple (with appropriately chosen head variables or constants) of the respective view V' and let R'^{exp} be the expansion of R'. It is easy to see that R'^{exp} coincides with R'' and therefore R' is equivalent to R.

Finally, notice that since R' is an equivalent rewriting of Q, R'^{exp} is isomorphic to Q (after removing some duplicate subgoals). Thus the subexpression S of R'^{exp} corresponding to V' is isomorphic to a subexpression of Q. Since S is an instance of the body of V', the body of V' is a generalization of a subexpression of the body of the query Q. □

The remaining results in this section and in Sect. 5.4 follow immediately from Lemma 6. It is easy to see from the above lemma that, since a view is defined as a generalization of a subexpression of the body of a query, each view definition has at most n atoms, where n is the number of subgoals in the longest query in $\mathcal{Q}$.

We can make a more precise statement about the number of subgoals in view definitions for views in optimal viewsets:

Corollary 8 *Let $\mathcal{P}$ be a conjunctive bag-set-oriented problem input whose queries do not have self-joins, and let V be any view in any optimal viewset $\mathcal{V}$ for $\mathcal{P}$. Suppose V is used in rewriting queries $Q_{i_1}, \ldots, Q_{i_k}$ in $\mathcal{Q}$; let m be the number of subgoals in the shortest definition among the definitions of $Q_{i_1}, \ldots, Q_{i_k}$. Then V can be defined using at most m subgoals.*

Corollary 9 *Given a conjunctive bag-set-oriented problem input $\mathcal{P}$ whose queries do not have self-joins, let $\mathcal{V}$ be an optimal viewset for $\mathcal{P}$. Then, rewriting any query in $\mathcal{Q}$ does not require self-joins of views in $\mathcal{V}$.*

5.4 Complexity for queries without self-joins

We now show that the decision version of the view-selection problem is in NP for a single storage-limit constraint on materialized views, provided that queries in the problem input have no self-joins. We formulate the decision version of the problem and state the result for the *oracle* version of problem inputs, similarly to the previous sections.

Proposition 4 *For any conjunctive query Q with n relational subgoals and for any conjunctive locally minimal rewriting R of Q in terms of views, such that $R \equiv_{bs} Q$, the number of views in R does not exceed n.*

This result follows from the definition of a locally minimal rewriting that is equivalent to a query under bag-set semantics. By definition, the rewriting does not contain filtering views or "unnecessary" containment-target views.

We now establish that the decision version of the view-selection problem is in NP, for problem inputs whose queries do not have self-joins. This result is a consequence of the following observation: under bag-set semantics, for any problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ whose queries do not have self-joins, with any set of constraints $\mathcal{L}$, and for any nonredundant viewset $\mathcal{V}$ for $\mathcal{P}$, the number of views in $\mathcal{V}$ does not exceed n, where n is the total number of relational subgoals in all the queries in the query workload $\mathcal{Q}$ in $\mathcal{P}$.

Theorem 9 *Given an* oracle version *of a conjunctive bag-set-oriented problem input $\mathcal{P}$ whose queries do not have self-joins, and assuming that the input set of constraints $\mathcal{L}$ represents a single storage limit, the decision version of the view-selection problem is in NP.*

Proof Let $\mathcal{P}$ be an oracle version of a problem input as in Lemma 6, and let N be a number. Suppose we are given a witness, which comprises: (1) a viewset $\mathcal{V}$, (2) locally-minimal rewritings of all the input workload queries in terms of $\mathcal{V}$, and (3) bijection mappings between each query and its corresponding rewriting (recall that only exact copies of query subgoals are possible in equivalent rewritings of queries under bag-set semantics).

Our goal is to check whether the viewset $\mathcal{V}$ provides admissible rewritings of the workload queries and gives a solution whose query-evaluating cost, on all the databases described by the input oracle, does not exceed the value N. Note that the answer to the query-evaluation-cost part of the check is "yes" if and only if there exists an *optimal* viewset for $\mathcal{P}$ whose query-evaluation-cost does not exceed N. If $\mathcal{V}$ is an optimal viewset for the problem input $\mathcal{P}$, we can do both checks in time polynomial in the size of $\mathcal{P}$, because of the following two reasons:

1. By Lemma 6, for an optimal viewset for $\mathcal{P}$, each view in the viewset has at most n subgoals, where n is the number of subgoals in the longest query definition in $\mathcal{P}$, and
2. From Proposition 4, for any locally minimal rewriting that is bag-set-equivalent to an input query, the number of views in the rewriting does not exceed the number of subgoals in the query.

Recall that under bag semantics, the total number of subgoals in the expansion of the rewriting is always the same as the number of query subgoals. On the other hand, under bag-set semantics the total number of subgoals in the expansion of the rewriting can be up to quadratic in the number of query subgoals, because the number of subgoals in each view can be up to the number of query subgoals. Note that unlike the bag-semantics case, under bag-set semantics it is possible to have identical query subgoals in different views. □

6 View-selection algorithm

In this section we consider a view-selection algorithm *cgalg*, which was introduced in [9, 10] for workloads of queries with or without self-joins under set semantics for query evaluation, and study its properties in the setting of this paper. We begin by summarizing the main results of this section:

1. The algorithm *cgalg* is sound for all the queries, views, and rewritings that we consider under set, bag, and bag-set semantics.
2. Under bag semantics, the algorithm *cgalg* is complete for workloads of queries both without and with self-joins.
3. Under bag-set semantics, the algorithm *cgalg* is complete for workloads of queries without self-joins.

6.1 The *cgalg* algorithm

In this section we work with the notion of an *efficient viewset*, which is defined using the notation of the definitions of Sect. 2.3: A viewset $\mathcal{V}$ is efficient for a problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ if (1) $\mathcal{V}$ is an admissible viewset for $\mathcal{P}$, and (2) the total cost of evaluating the rewritings of the queries in $\mathcal{Q}$ in terms of $\mathcal{V}$ on the database $\mathcal{D}_{\mathcal{V}}$ is lower than the total cost of evaluating the queries in $\mathcal{Q}$ on the input database $\mathcal{D}$. Intuitively, given a problem input $\mathcal{P}$, an optimal viewset (for the definition, see Sect. 2.3) for $\mathcal{P}$ is always efficient provided its total query-evaluation costs are different from the total query-evaluation costs (without views) on the input database. At the same time, an efficient viewset for $\mathcal{P}$ does not have to be an optimal viewset for $\mathcal{P}$. Note that no efficient viewset exists for problem instances where the total query-evaluation costs of an optimal viewset are the same as the total query-evaluation costs without views on the input databases.

The *cgalg* algorithm of [9,10] provides the following guarantee: For each viewset $\mathcal{V}$ that is efficient for a problem input $\mathcal{P}$, *cgalg* generates at least one efficient viewset $\mathcal{V}_{opt}$ that reduces the costs of the input query workload at least as much as $\mathcal{V}$ and satisfies the same storage limit. We say that the algorithm produces the *best* efficient viewsets. For simplicity, we give here the pseudocode for *cgalg* for problem inputs whose query workload consists of a single query; the extension to multiquery input workloads is straightforward.

Procedure *cgalg*.
Input: database schema $\mathcal{S}$, query Q, database $\mathcal{D}$, storage limit $\mathcal{L}$.
Output: R_{opt}, optimal equivalent rewriting of Q on $\mathcal{D}$,
    $\mathcal{V}_{opt}$, optimal viewset of Q on $\mathcal{D}$.

```
1 Begin:
2     minimize $Q$ to obtain a query $Q'$;
3     set $R_{opt}$ to $Q'$;
4     set the cost $C_{opt}$ of $R_{opt}$ to $C(Q', \mathcal{D})$;
5     find all views $\mathcal{V}$ whose body is a (generalization of a) subset of subgoals of $Q'$;
6     for each subset $\mathcal{W}$ of $\mathcal{V}$ such that $\Sigma_{V \in \mathcal{W}}\, size(V, \mathcal{D}) \leq \mathcal{L}$ do:
7     begin:
8         find a rewriting $R$ of $Q'$ using $\mathcal{W}$;
9         construct the expansion $R^{exp}$ of $R$;
10        if there exists a containment mapping from $Q'$ to $R^{exp}$ then:
11            if the cost $C(R, \mathcal{D})$ of answering $Q'$ on $\mathcal{D}$ using $R$ is less than $C_{opt}$
12            then begin:
13                $R_{opt} := R$;
14                $C_{opt} := C(R, \mathcal{D})$;
15                $\mathcal{V}_{opt} = \{V \in \mathcal{W} | V$ is used to the rewriting$\}$;
16            end;
17    end;
18    return $R_{opt}, \mathcal{V}_{opt}$.
19 End.
```

We now discuss some details of the algorithm: A view-size oracle $\mathcal{O}$ instantaneously gives the size of any relation defined on the database $\mathcal{D}$; we assume that for a rewriting R in terms of views and for a fixed size-monotonic cost model for query evaluation, the time required to obtain the cost $C(R, \mathcal{D})$ of evaluating R in terms of the relations for the views on $\mathcal{D}$ is negligible when using the oracle $\mathcal{O}$ (hence essentially we compute the cost $C(R, \mathcal{O}(\mathcal{D}))$. In practice, the view sizes and costs of answering Q on $\mathcal{D}$ using R can be estimated via standard formulas used in query optimizers in database-management systems [13].

6.2 Using *cgalg* under set semantics

Propositions 5 and 6 state two results of [10] from the set-semantics setting.

Proposition 5 *Under set semantics for query evaluation and provided that (i) all view atoms in all rewritings have different relation names, and (ii) filtering views are not used in query rewritings, the algorithm cgalg is sound for problem inputs with workloads of arbitrary conjunctive queries and is complete for problem inputs with workloads of conjunctive queries without self-joins.*

We note that the algorithm *cgalg* is also sound under set semantics for problem inputs with workloads of arbitrary conjunctive queries with arithmetic comparisons. In general, the algorithm is not complete (i.e., is not guaranteed to produce an optimal viewset) because some optimal rewritings may use self-joins of view atoms [11].

Proposition 6 *Under the assumptions of Proposition 5 and assuming that a view-size oracle $\mathcal{O}$ and a size-monotonic cost model for query evaluation are given, the runtime of cgalg is $\Theta(2^{2^{2m}})$, where m is the total number of subgoals of the queries in the input workload $\mathcal{Q}$.*

Intuitively, under the assumptions of Proposition 5, *cgalg* will generate all efficient viewsets if it generates only viewsets that have up to m views. Note that the step of generating a rewriting given a subset $\mathcal{W}$ of the set $\mathcal{V}$ of views takes constant time in the size of the subset $\mathcal{W}$ [3].

6.3 Using *cgalg* under bag and bag-set semantics

In this subsection we study the properties of the *cgalg* view-selection algorithm under bag and bag-set semantics in the setting of this paper. The following results are immediate consequences of the properties of the algorithm that were outlined in Sect. 6.1, as well as of our results in Sects. 4 and 5.

Proposition 7 *Under bag or bag-set semantics for query evaluation, the algorithm cgalg is sound for problem inputs with workloads of arbitrary conjunctive queries* (*i.e., both with and without self-joins*).

This result also holds for problem instances whose query workloads include queries with arithmetic comparisons. The proof derives from Propositions 5 and 6.

Proposition 8 *Under bag semantics for query evaluation and provided that all view atoms in all rewritings have different relation names, the algorithm cgalg is complete for problem inputs with workloads of arbitrary conjunctive queries.*

In proving this proposition, we use the fact that filtering views cannot be present in rewritings equivalent to queries under bag semantics. The key idea of the proof is to recall that in the expansion of each rewriting R of each input query Q, the bodies of view atoms constitute a partition of the body of Q.

Proposition 9 states the completeness conditions of the *cgalg* algorithm under bag-set semantics for query evaluation.

Proposition 9 *Under bag-set semantics for query evaluation and provided that all view atoms in all rewritings have different relation names, the algorithm cgalg is complete for problem inputs with workloads of conjunctive queries without self-joins.*

Note the no-self-joins restriction in Proposition 9 (cf. Proposition 8 for the bag-semantics case). The idea of the proof of Proposition 9 is as follows: When queries have no self-joins

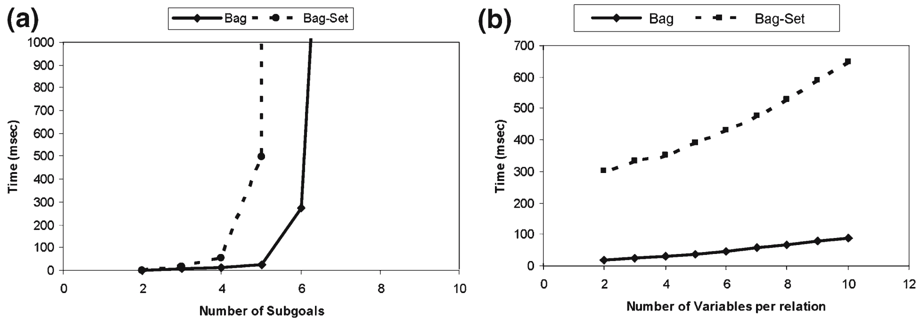

Fig. 2 Dependence of runtime on **a** the number of query subgoals, and **b** the number of variables per query subgoals

and each view in $\mathcal{V}$ is used exactly once in the rewriting of exactly one query in the input query workload, *cgalg* can look for views for each workload query separately. Recall (see Theorem 8 in Sect. 5.2) that under bag-set semantics we can always restrict ourselves to considering rewritings that do not use filtering views.

6.4 Implementation and experimental results

We have implemented the *cgalg* algorithm for both bag semantics and bag-set semantics. The algorithm has been implemented in Java. The cost function $C(Q, \mathcal{D})$ for computing the cost of evaluating $Q(\mathcal{D})$ that is used in the algorithm, is presented in Sect. 2.3. To define the function $size(V, \mathcal{D})$ which estimates the size of a view V with respect to the database $\mathcal{D}$, we estimate the cost of each operation as in Chapter 16 of [13].

In the following paragraphs we present some of the experiments conducted to evaluate the efficiency of *cgalg* algorithm. All the experiments were run on a machine with 1.8 GHz Intel Core Duo processor with 2 MB L2 Cash and 2 GB DDR2 RAM, running the Windows XP operating system.

Figure 2a and b show that the runtime of the algorithm depends strongly on the number of subgoals in the body of the user query rather than on the number of variables in the query. As it is expected, the algorithm runs significantly faster for the case of bag semantics rather than for bag-set semantics. In more detail, Fig. 2a shows the dependence of the runtime on the number of query subgoals for chain queries for both bag semantics and bag-set semantics. Figure 2b shows the dependence of the runtime on the number of variables per query subgoals. In this case, the number of subgoals in the query is kept constant (equal to five subgoals). The experiments are run on queries of the form:

$$q(X, Z) \text{ :- } q_1(X, \overline{Y}_1, Z_1), q_2(Z_1, \overline{Y}_2, Z_2), \ldots, q_k(Z_{k-1}, \overline{Y}_k, Z).$$

Figure 3 shows the dependence of the runtime on the value of the storage limit for chain queries with a self-join of seven subgoals under bag semantics. In this figure we measure the storage limit as the ratio of the value of the storage limit over the value of the size of the output of the query computed on the database $\mathcal{D}$ (i.e. $\mathcal{L}/size(Q, \mathcal{D})$). The dotted line in the figure shows the lower bound on the storage limit for which there exists at least one viewset such that an equivalent rewriting exists. We observe that for small storage limits (less than 1%) the runtime of the algorithm is considerably reduced because many of the viewsets fail the test of the storage limit and hence they are not considered for rewriting.

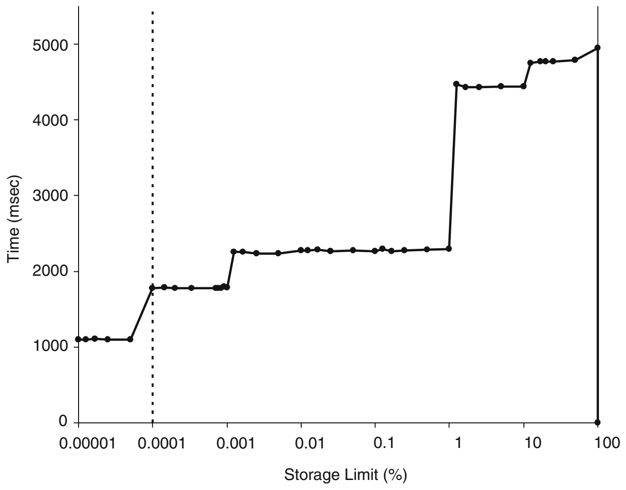

Fig. 3 Dependence of runtime on the space limit parameter

7 Conclusions and future work

In this work we investigated the problem of designing views to answer queries in relational databases under set, bag, and bag-set semantics. Although in these results we use a special type of constraints on storage limit, many of these results may be carried over for more general constraints. The results can be used in finding sound and complete algorithms for designing views and rewriting queries under each of the three assumptions. We are currently working on designing such algorithms by exploring the ideas of the *cgalg* algorithm [9,10] to design a view-selection algorithm that would (1) use our concept of generalized common subexpressions, and that (2) would be complete for more cases than what we have in the present paper. A challenging task would be to design more efficient versions of this algorithm for practically important special cases of view selection within the setting of this paper. An interesting problem in this direction is to incorporate in the `cgalg` algorithm heuristics that prune the search space of viewsets early, by discarding not promising viewsets. The problem is related to the problem of minimization of a viewset with respect to its property, to equivalently rewrite a set of queries [24]. In our future work we also plan to study the view-selection problem for conjunctive queries and views with arithmetic comparisons. Applying these results to XQuery queries on XML data is another research direction.

Acknowledgments We would like to thank Matthew Damigos for his fine work with implementing the *cgalg* algorithm and running the experiments.

References

1. AutoAdmin: Self-tuning and self-administering databases. http://research.miosoft.com/dmx/autoadmin/default.asp
2. Afrati, F., Chirkova, R.: Selecting and using views to computeaggregate queries. In: Database Theory - ICDT 2005 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3363, pp. 383–397 (2005)

3. Afrati, F., Li, C., Ullman, J.: Generating efficient plans for queries using views. In: Proceedings of ACM SIGMOD, pp. 319–330 (2001)
4. Agrawal, S., Chaudhuri, S., Narasayya, V.: Automated selection of materialized views and indexes in SQL databases. In: Proceedings of VLDB, pp. 496–505 (2000)
5. Baralis, E., Paraboschi, S., Teniente, E.: Materialized views selection in a multidimensional database. In: Proceedings of VLDB, pp. 156–165 (1997)
6. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational databases. In: Proceedings of the 9th ACM symposium on theory of computing, pp. 77–90 (1977)
7. Chaudhuri, S., Krishnamurthy, R., Potamianos, S., Shim, K.: Optimizing queries with materialized views. In: Proceedings of ICDE, pp. 190–200. Taipei, Taiwan (1995)
8. Chaudhuri, S., Vardi, M.Y.: Optimization of real conjunctive queries. In: Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, pp. 59–70. ACM Press, New York (1993)
9. Chirkova, R., Genesereth, M.: Linearly bounded reformulations of conjunctive databases. In: Proceedings of the first conference on computational logic, pp. 987–1001 (2000)
10. Chirkova, R., Genesereth, M.: Database reformulation with integrity constraints. In: Proceedings of the logic and computational complexity workshop at the logic in computer science conference (LICS) (2005)
11. Chirkova, R., Halevy, A.Y., Suciu, D.: A formal perspective on the view selection problem. VLDB J. **11**(3), 216–237 (2002)
12. Deutsch, A.: XML query reformulation over mixed and redundant storage. Ph.D. thesis, University of Pennsylvania (2002). Available at http://www.db.ucsd.edu/People/alin/thesis/thesis.pdf
13. Garcia-Molina, H., Ullman, J., Widom, J.: Database systems: the complete book. Prentice Hall, Englewood Cliffs (2002)
14. Grumbach, S., Rafanelli, M., Tininini, L.: On the equivalence and rewriting of aggregate queries. Acta Informatica **40**(8), 529–584 (2004)
15. Gupta, H.: Selection of views to materialize in a data warehouse. In: Proceedings of ICDT, pp. 98–112 (1997)
16. Gupta, H., Harinarayan, V., Rajaraman, A., Ullman, J.: Index selection for OLAP. In: Proceedings of ICDE, pp. 208–219 (1997)
17. Gupta, H., Mumick, I.S.: Selection of views to materialize under a maintenance cost constraint. In: Proceedings of ICDT, pp. 453–470 (1999)
18. Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing data cubes efficiently. In: Proceedings of ACM SIGMOD, pp. 205–216 (1996)
19. IBM: Autonomic Computing. http://www.research.ibm.com/autonomic/
20. Ioannidis, Y., Ramakrishnan, R.: Containment of conjunctive queries: Beyond relations as sets. ACM Trans. Database Syst. **20**(3), 288–324 (1995)
21. Karloff, H.J., Mihail, M.: On the complexity of the view-selection problem. In: Proceedings of PODS, pp. 167–173. Philadelphia, Pennsylvania (1999)
22. Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. **32**(4), 422–469 (2000)
23. Levy, A.Y., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, pp. 95–104. ACM Press, New York (1995)
24. Li, C., Bawa, M., Ullman, J.D.: Minimizing view sets without losing query-answering power. In: Proceedings of ICDT, pp. 99–113 (2001)
25. Pottinger, R., Halevy, A.Y.: Minicon: a scalable algorithm for answering queries using views. VLDB J. **10**(2-3), 182–198 (2001)
26. Shasha, D., Bonnet, P.: Database tuning: principles, experiments, and troubleshooting techniques. Morgan Kaufmann (2002). http://www.distlab.dk/dbtune/
27. Theodoratos, D., Sellis, T.: Data warehouse configuration. In: Proceedings of VLDB, pp. 126–135. Athens, Greece (1997)
28. Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In: Proceedings of VLDB, pp. 136–145. Athens, Greece (1997)