Journal of Computer and System Sciences 94 (2018) 98–117

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Computing marginals using MapReduce

Foto N. Afrati [a,*], Shantanu Sharma [b], Jonathan R. Ullman [c], Jeffrey D. Ullman [d]

[a] NTU of Athens, Greece
[b] UC, Irvine, United States
[c] Northeastern University, United States
[d] Stanford University, United States

ARTICLE INFO

Article history:
Received 12 September 2016
Accepted 5 February 2017
Available online 14 March 2017

Keywords:
Data-cube
MapReduce
Marginals

ABSTRACT

We consider the problem of computing data-cube marginals by a single round of MapReduce, focusing on the relationship between the reducer size and the replication rate. Initially, we simplify the problem by making the extent of each dimension the same. Several recursive constructions meet or come close to the minimum possible replication rate for a given reducer size. These ideas extend in two directions. We relax the assumption that the extents are all equal, and we consider how to compute marginals from lower-order marginals rather than from the raw data cube.

1. Introduction and preliminaries

We shall begin with the needed definitions. These include the data cube, marginals, MapReduce, and the parallelism-communication tradeoff that we represent by reducer size versus replication rate.

1.1. Data cubes

We may think of a data cube [19] as a relation, where one attribute is an aggregatable quantity, such as "price," and the other attributes are *dimensions*. Tuples represent facts, and each fact consists of a value for each dimension, which we can think of as locating that fact in the cube. Commonly, one can think of facts as representing sales, and the dimensions as representing the customer, the item purchased, the date, the store at which the purchase occurred, and so on. The aggregatable quantity might then be the total number of sales matching the values for each of the dimensions, or the total price of all those sales.

1.2. Marginals

A *marginal* of a data cube is the aggregation of the data in all those tuples that have fixed values in a subset of the dimensions of the cube. We shall assume this aggregation is the sum, but the exact nature of the aggregation is unimportant in what follows. Marginals can be represented by a list whose elements correspond to each dimension, in order. If the value in a dimension is fixed, then the fixed value represents the dimension. If the dimension is aggregated, then there is a * for that dimension. The number of dimensions over which we aggregate is the *order* of the marginal.

* Corresponding author.
E-mail address: afrati@gmail.com (F.N. Afrati).

http://dx.doi.org/10.1016/j.jcss.2017.02.007
0022-0000/

Example 1.1. Suppose there are $n = 5$ dimensions, and the data cube is a relation DataCube(D1,D2,D3,D4,D5,V). Here, D1 through D5 are the dimensions, and V is the value that is aggregated.

```
SELECT SUM(V)
FROM DataCube
WHERE D1 = 10 AND D3 = 20 AND D4 = 30;
```

will sum the data values in all those tuples that have value 10 in the first dimension, 20 in the third dimension, 30 in the fourth dimension, and any values in the second and fifth dimension of a five-dimensional data cube. We can represent this marginal by the list $[10, *, 20, 30, *]$, and it is a second-order marginal.

1.3. Contributions

We construct algorithms that can compute efficiently marginals of a fixed order k in one MapReduce round. We consider an algorithm to be efficient if it achieves a low replication rate (average number of key-value pairs per input generated by the mappers) under a fixed reducer size (number of key-value pairs reaching a single reducer).

Initially, we look at the simplified situation where the extent (number of different values) of each dimension is the same. We show that the replication rate is minimized when the reducers receive all the inputs necessary to compute one marginal of higher order. That observation lets us view the problem as one of covering sets of k dimensions with the smallest possible number of sets of a larger size m, a problem that has been studied under the name "covering numbers." We offer a number of recursive constructions that, for different values of k and m, meet or come close to yielding the minimum possible replication rate for a given reducer size.

Then, we extend these ideas in two directions. First, we relax the assumption that the extents are equal in all dimensions, and we discuss how to modify the techniques for the equal-extents case to work in the general case. Second, we consider the way that kth-order marginals could be computed in one round from lower-order marginals rather than from the raw data cube. This problem leads to a new combinatorial covering problem, and we offer some methods to get good solutions to this problem.

1.4. Assumption: all dimensions have equal extent

We shall make the simplifying assumption that in each dimension there are d different values. In practice, we do not expect to find that each dimension really has the same number of values. For example, if one dimension represents Amazon customers, there would be millions of values in this dimension. If another dimension represents the date on which a purchase was made, there would "only" be thousands of different values.

However, it probably makes little sense to compute marginals where we fix the Customer dimension to be each customer, in turn; there would be too many marginals, and each would have only a small significance. More likely, we would want to group the values of dimensions in some way, e.g., customers by state and dates by month. Moreover, we shall see that our methods really only need the parameter d to be an upper bound on the true number of distinct values in a dimension. The consequence of the *extents* (number of distinct values) of different dimensions being different is that some of the reducers will get fewer than the theoretical maximum number of inputs allowed. That discrepancy has only a small effect on the performance of the algorithm. Moreover, if there are really large differences among the extents of the dimensions, then an extension of our algorithms can improve the performance. We shall defer this issue to Section 5.

1.5. Mapping schemas for MapReduce algorithms

We assume the reader is familiar with the MapReduce computational model [15]. In the theory of [6], a problem is modeled by a set of inputs (the tuples or points of the data cube, here), a set of outputs (the values of the marginals) and a relationship between the inputs and outputs that indicates which inputs are needed to compute which outputs. In order for a MapReduce algorithm to solve this problem with a reducer size q, there must be a *mapping schema*, which is a relationship between inputs and reducers that satisfies two properties:

1. No reducer is associated with more than q inputs, and
2. For every output, there is some reducer that is associated with all the inputs that output needs for its computation.

Point (1) is the definition of "reducer size," while point (2) is the requirement that the algorithm can compute all the outputs that the problem requires. The fundamental reason that MapReduce algorithms are not just parallel algorithms in general is embodied by point (2). In a MapReduce computation, every output is computed by one reducer, independently of all other reducers.

1.6. Parallelism-communication tradeoff

Following the approach to analyzing MapReduce algorithms given in [6], we look at tradeoffs between the *reducer size* (maximum number of inputs allowed at a reducer), which we always denote by q, and the *replication rate* (average number of reducers to which an input needs to be sent), which we always denote by r. The replication rate represents the cost of communication between the mappers and reducers, and communication cost is often the dominant cost of a MapReduce computation. Typically, the larger the reducer size, the lower the replication rate. But we want to keep the reducer size low for two reasons: it enables computation to proceed in main memory, and it forces a large degree of parallelism, both of which lead to low wall-clock time to finish the MapReduce job.

1.7. Naïve solution: computing one marginal per reducer

Now, let us consider the problem of computing all the marginals of a data cube in the above model. If we are not careful, the problem becomes trivial. The marginal that aggregates over all dimensions is an output that requires all d^n inputs of the data cube. Thus, $q = d^n$ is necessary to compute all the marginals. But that means we need a single reducer as large as the entire data cube, if we are to compute all marginals in one round. As a result, it only makes sense to consider the problem of computing a limited set of marginals in one round.

The kth-order marginals are those that fix $n-k$ dimensions and aggregate over the remaining k dimensions. To compute a kth-order marginal, we need $q \geq d^k$, since such a marginal aggregates over d^k tuples of the cube. Thus, we could compute all the kth-order marginals with $q = d^k$, using one reducer for each marginal. As a "problem" in the sense of [6], there are d^n inputs, and $d^{n-k}\binom{n}{k}$ outputs, each representing one of the marginals. Each output is connected to the d^k inputs over which it aggregates. Each input contributes to $\binom{n}{k}$ marginals – those marginals that fix $n-k$ out of the n dimensions in a way that agrees with the tuple in question. That is, for $q = d^k$, we can compute all the kth-order marginals with a replication rate r equal to $\binom{n}{k}$.

For $q = d^k$, there is nothing better we can do. However, when q is larger, we have a number of options, and the purpose of this paper is to explore these options.

1.8. Outline of the paper

In the next section, we review related and relevant work. Then:

- In Section 3 we explore methods for grouping the computation of many marginals at one reducer. This problem leads to the application of covering numbers to the design of MapReduce algorithms for computing marginals.
- In Section 4 we give the proof that our strategy for grouping marginals at reducers is optimal.
- Section 5 relaxes the constraint that the extents be the same in each dimension and suggests how we can apply the techniques of Section 3 to the more general and more realistic problem.
- Section 6 looks at the problem of computing marginals of one from lower-order marginals, which is the way marginals are often computed in practice. We introduce a new covering problem that represents such computations and propose some solutions to this problem.
- Finally, Section 7 outlines both the achievements of this paper and the interesting open problems that remain.

2. Related work

This is the first work that offers algorithms to compute marginals in MapReduce with rigorous performance guarantees. However, recently, there have been a number of papers that develop solutions for computing marginals in several rounds of MapReduce or other shared-nothing architectures. Probably one of the closest work to what we present here is in [25]. This paper expresses the goal of minimizing communication, and of partitioning the work among reducers. It does not, however, present concrete bounds or algorithms that meet or approach those bounds, as we do here.

[31] and [32] offer an efficient system that extends MapReduce to compute data cubes for big data. The authors use cuboids to split the data. However optimal algorithms or any analysis of the algorithms are not discussed.

[24] considers constructing a data cube using a nonassociative aggregation function and also examines how to deal with nonuniformity in the density of tuples in the cube. It deals with constructing the entire data cube using multiple rounds of MapReduce.

[16] offers a survey of the state of the art of analytical query processing in MapReduce. It offers a classification of existing research focusing on the optimization objective.

[33] investigates analytics computation in the context of text mining. It proposes a Topic Cube to capture semantics of text and offers implementation in MapReduce. It is worth mentioning an orthogonal line of work where storage managers distribute multidimensional data across nodes for the purpose of executing operations without high replication rate, e.g., [27] (which is used in SciDB) partitions the array into sub-arrays using chunks of equal size and data at chunk boundary can optionally be replicated to create chunks with overlap.

Extracting aggregate information on the datacube has been studied intensively in the past for a single-machine computational environment (e.g., [17,20]).

The optimizations developed in this paper could be applied to specialized engines such as SciDB [2,12] which is a database system that has recently been developed to support efficient processing of multidimensional arrays in a shared-nothing cluster, or Pagrol [31,32] which extends MapReduce to compute data cubes for big data.

A number of systems have been developed recently to support analytical functions – among other things. They experimentally test their approaches for scalability and other features but they do not provide formal analysis. Some are focused on computing the data cube in Mapreduce. [3] looks at using MapReduce to form a data cube from data stored in Bigtable. [23] and [30] provide implementations of known algorithms for computing the datacube in MapReduce.

[26] is devoted to applying previously developed approaches to data intensive task of OLAP cube computation. It investigates two levels of Map-Reduce applicability (local multicore and multi-server clusters), and presents cube construction and query processing algorithms.

Other systems have a broader scope but include computation in data cube style as one of their ingredients, or offer implementations in computational frameworks more general than MapReduce. [29] proposes novel primitives for scaling calculations such as aggregation and joins. Load balancing is one of this paper's concerns.

[18] is AsterData's system that computes user defined aggregates. SciDB [2,12] processes arrays by splitting them into multidimensional sub-arrays, called chunks, and processing these chunks in parallel.

MADlib [21,13] offers a parallel database design for analytics to implement machine learning algorithms. A basic building block in MADlib is user defined aggregates [4] is a hybrid system which is built to combine the advantages of scalability and fault tolerance of MapReduce with a traditional database in each processor.

3. Computing many marginals at one reducer

We wish to study the tradeoff between reducer size and replication rate, a phenomenon that appears in many problems [28,7,6,5]. Since we know from Section 1.7 the minimum possible reducer size, we need to consider whether using a larger value of q can result in a significantly smaller value of r. Our goal is to combine marginals, in such a way that there is maximum overlap among the inputs needed to compute marginals at the same reducer.

Start by assuming the maximum overlap and minimum replication rate are obtained by combining kth-order marginals into a set of inputs suitable to compute one marginal of order higher than k. This assumption is correct, and we shall offer a proof of the fact in Section 4. However, we are still left with answering the question: how do we pack all the kth-order marginals into as few marginals of order m as possible, where $m > k$. That question leads us to the matter of "asymmetric covering codes" or "covering numbers" [10,14].

3.1. Covering marginals

Suppose we want to compute all kth-order marginals, but we are willing to use reducers of size $q = d^m$ for some $m > k$. If we fix any $n - m$ of the n dimensions of the data cube, we can send to one reducer the d^m tuples of the cube that agree with those fixed values. We then can compute all the marginals that have $n - k$ fixed values, as long as those values agree with the $n - m$ fixed values that we chose originally.

Example 3.1. Let $n = 7$, $k = 2$, and $m = 3$. Suppose we fix the first $n - m = 4$ dimensions, say using values a_1, a_2, a_3, and a_4. Then we can compute the d marginals $a_1a_2a_3a_4x**$ for any of the values x that may appear in the fifth dimension. We can also compute all marginals $a_1a_2a_3a_4*y*$ and $a_1a_2a_3a_4**z$, where y and z are any of the possible values for the sixth and seventh dimensions, respectively. Thus, we can compute a total of $3d$ second-order marginals at this one reducer. That turns out to be the largest number of marginals we can cover with one reducer of size $q = d^3$.

What we do for one assignment of $n - m$ values to $n - m$ of the dimensions we can do for all assignments of values to the same dimensions, thus creating d^{n-m} reducers, each of size d^m. Call this collection of reducers a *team* of reducers. Together, a team allows us to compute all kth-order marginals that fix the same $n - m$ dimensions (along with any $m - k$ of the remaining m dimensions).

Example 3.2. Continuing Example 3.1, the team of d^4 reducers, each member of which fixes the first four dimensions in a different way, covers any of the $3d^5$ marginals that fix the first four dimensions, along with one of dimensions 5, 6, or 7.

3.2. From marginals to sets of dimensions

To understand why the problem is more complex than it might appear at first glance, let us continue thinking about the simple case of Example 3.1. We need to cover all second-order marginals, not just those that fix the first four dimensions. If we had one team of d^4 reducers to cover each four of the seven dimensions, then we would surely cover all second-order marginals. But we don't need all $\binom{7}{2} = 21$ such teams. Rather, it is sufficient to pick a collection of sets of four of the seven dimensions, such that every set of five of the seven dimensions contains one of those sets of size four.

In what follows, we find it easier to think about the sets of dimensions that are aggregated, rather than those that are fixed. So we can express the situation above as follows. Collections of second-order marginals are represented by pairs of dimensions – the two dimensions such that each marginal in the collection aggregates over those two dimensions. These pairs of dimensions must be *covered* by sets of three dimensions – the three dimensions aggregated over by one third-order marginal. Our goal, which we shall realize in Example 3.3 below, is to find a smallest set of tripletons such that every pair chosen from seven elements is contained in one of those tripletons.

In general, we are faced with the problem of covering all sets of k out of n elements by the smallest possible number of sets of size $m > k$. Such a solution leads to a way to compute all kth-order marginals using as few reducers of size d^m as possible. Abusing the notation, we shall refer to the sets of k dimensions as *marginals*, even though they really represent teams of reducers that compute large collections of marginals with the same fixed dimensions. We shall call the larger sets of size m *handles*. The implied MapReduce algorithm takes each handle and creates from it a team of reducers that are associated, in all possible ways, with fixed values in all dimensions except for those dimensions in the handle. Each created reducer receives all inputs that match its associated values in the fixed dimensions.

Example 3.3. Call the seven dimensions $ABCDEFG$. Here is a set of seven handles (sets of size three), such that every marginal of size two is contained in one of them:

$$ABC,\ ADE,\ AFG,\ BDF,\ BEG,\ CDG,\ CEF$$

To see why these seven handles suffice, consider three cases, depending on how many of A, B, and C are in the pair of dimensions to be covered.

Case 0: If none of A, B or C is in the marginal, then the marginal consists of two of D, E, F, and G. Note that all six such pairs are contained in one of the last six of the handles.

Case 1: If one of A, B, or C is present, then the other member of the marginal is one of D, E, F, or G. If A is present, then the second and third handles, ADE and AFG together pair A with each of the latter four dimensions, so the marginal is covered. If B is present, a similar argument involving the fourth and fifth of the handles suffices, and if C is present, we argue from the last two handles.

Case 2: If the marginal has two of A, B, and C, then the first handle covers the marginal.

Incidentally, we cannot do better than Example 3.3. Since no handle of size three can cover more than three marginals of size two, and there are $\binom{7}{2} = 21$ marginals, clearly seven handles are needed.

As a strategy for evaluating all second-order marginals of a seven-dimensional cube, let us see how the reducer size and replication rate based on Example 3.3 compare with the baseline of using one reducer per marginal. Recall that if we use one reducer per marginal, we have $q = d^2$ and $r = \binom{7}{2} = 21$. For the method based on Example 3.3, we have $q = d^3$ and $r = 7$. That is, each tuple is sent to the seven reducers that have the matching values in dimensions $DEFG$, $BCFG$, and so on, each set of attributes on which we match corresponding to the complement of one of the seven handles mentioned in Example 3.3.

3.3. Covering numbers

Let us define $C(n, m, k)$ to be the minimum number of sets of size m out of n elements such that every set of k out of the same n elements is contained in one of the sets of size m. For instance, Example 3.3 showed that $C(7, 3, 2) = 7$. The function $C(n, m, k)$ is called the *covering number* in [10]. The numbers $C(n, m, k)$ guide our design of algorithms to compute kth-order marginals. There is an important relationship between covering numbers and replication rate, that justifies an interest in constructive upper bounds for $C(n, m, k)$.

Theorem 3.4. *If the reducer size is $q = d^m$, then we can compute all kth-order marginals of an n-dimensional data cube with replication rate $r = C(n, m, k)$.*

Proof. Each marginal in the set of $C(n, m, k)$ handles can be turned into a team of reducers, one for each of the d^{n-m} ways to fix the dimensions that are not in the handle. Each input gets sent to exactly one member of the team for each handle – the reducer that corresponds to fixed values that agree with the input. Thus, each input is sent to exactly $C(n, m, k)$ reducers. □

Sometimes we will want to fix some choices of m, k and study how $C(n, m, k)$ grows with the dimension n. In this case we will often write simply $C(n)$ when m and k are clear from context.

Fig. 1 summarizes the results in this section with pointers to subsections. Fig. 2 shows the upper bounds given by these algorithms.

Case	*C(n,m,1)*	*C(n,3,2)*	*C(n,m,2)*	*C(n,m,k)*	*C(n,4,3)*
Sect.	3.4	3.5, 3.6	3.7,3.8	3.9	3.10

Fig. 1. Summary of results in this section. The first row shows cases for which we provide algorithms, and the second row points to the subsections where these algorithms can be found.

Case	ReplicationRate
(n, m, k) *naive*	$\binom{n}{k}$
$(n, m, 1)$	$\lceil n/m \rceil$
$(n, 3, 2)^*$	$n^2/6 - n/6$
$(n, 3, 2)$	$n^2/4 - n/2$
$(n, m, 2)$	$\frac{n^2}{2(m-1)} - \frac{n}{2} + 1 - \frac{m}{2(m-1)}$
$(n, m, 2)^*$	$2n^2/m^2 - 1$
(n, m, k)	$\binom{n}{k}/(m - k + 1)$
$(n, 4, 3)$	$n^3/16$

Fig. 2. The replication rates achieved by our algorithms. We denote each case with (n, m, k), where n denotes the number of dimensions, k the order of the marginals we compute, and m that the reducer size is $q = d^m$. The asterisk denotes that this algorithm applies only in a special subcase. The second row gives the rate for the naive algorithm.

3.4. First-order marginals

The case $k = 1$ is quite easy to analyze. We are asking how many sets of size m are needed to cover each singleton set, where the elements are chosen from a set of size n. It is easy to see that we can group the n elements into $\lceil n/m \rceil$ sets so that each of the n elements is in at least one of the sets, and there is no way to cover all the singletons with fewer than this number of sets of size m. That is, $C(n, m, 1) = \lceil n/m \rceil$. For example, if $n = 7$ and $m = 2$, then the seven dimensions $ABCDEFG$ can be covered by four sets of size 2, such as AB, CD, EF, and FG.

Algorithm for C(n,m,1): Just partition the dimensions into $\lceil n/m \rceil$ subsets of size m; each subset is a handle. This algorithm improves over the naive by a factor of $n/\lceil n/m \rceil$, or approximately a factor of m.

3.5. 2nd-order marginals covered by 3rd-order handles

The next simplest case is $C(n, 3, 2)$, that is, covering second-order marginals by third-order marginals, or equivalently, covering sets of two out of n elements by sets of size 3. One simple observation is that a set of size 3 can cover only three pairs, so $C(n, 3, 2) \geq \binom{n}{2}/3$, or:

$$C(n, 3, 2) \geq n^2/6 - n/6 \tag{1}$$

In fact, more generally:

Theorem 3.5.

$$C(n, m, k) \geq \binom{n}{k} / \binom{m}{k}$$

Proof. There are $\binom{n}{k}$ marginals to be covered, while one handle can cover only $\binom{m}{k}$ marginals. □

Aside: Using the probabilistic method, one can show that

$$C(n, m, k) \leq 2 \ln \binom{n}{k} \cdot \frac{\binom{n}{k}}{\binom{m}{k}}$$

so this simple lower bound is actual optimal up to a factor of $2 \ln \binom{n}{k}$. However, in what follows we will give upper bounds on $C(n, m, k)$ that:

(a) Are explicit (i.e., constructive).
(b) Meet the lower bound either exactly or to within a constant factor.
(c) Are recursive. The value of recursive constructions will be seen in Section 5, when we generalize to data cubes that do not have the same extent for each dimension. There it will be seen how the ability to break the dimensions into small groups lets us deal efficiently with the differences in extents.

While [10] gives us some specific optimal values of $C(n, 3, 2)$ to use as the basis of an induction, we would like a recursive algorithm for constructing ways to cover sets of size 2 by sets of size 3, and we would like this recursion to yield solutions that are as close to the lower bound of Equation (1) as possible. We can in fact give a construction that, for an infinite number of n, matches the lower bound of Equation (1). Suppose we have a solution for n dimensions. We construct a solution for $3n$ dimensions as follows. First, group the $3n$ dimensions into three groups of n each. Let these groups be $\{A_1, A_2, \ldots, A_n\}$, $\{B_1, B_2, \ldots, B_n\}$, and $\{C_1, C_2, \ldots, C_n\}$. We construct handles of two kinds:

1. Choose all sets of three elements, one from each group, say $A_iB_jC_k$, such that $i + j + k$ is divisible by n. There are evidently n^2 such handles, since any choice from the first two groups can be completed by exactly one choice from the third group.
2. Use the assumed solution for n dimensions to cover all the pairs chosen from one of the three groups. So doing adds another $3C(n, 3, 2)$ handles.

This set of handles covers all pairs chosen from the $3n$ dimensions. In proof, if the pair has dimensions from different groups, then it is covered by the handle from (1) that has those two dimensions plus the unique member of the third group such that the sum of the three indexes is divisible by n. If the pair comes from a single group, then we can argue recursively that it is covered by a handle added in (2).

Example 3.6. Let $n = 3$, and let the three groups be $A_1A_2A_3$, $B_1B_2B_3$, and $C_1C_2C_3$. From the first rule, we get the handles $A_1B_1C_1$, $A_1B_2C_3$, $A_1B_3C_2$, $A_2B_1C_3$, $A_2B_2C_2$, $A_2B_3C_1$, $A_3B_1C_2$, $A_3B_2C_1$, and $A_3B_3C_3$. Notice that the sum of the subscripts in each handle is 3, 6, or 9. For the second rule, note that when $n = 3$, a single handle consisting of all three dimensions suffices. Thus, we need to add $A_1A_2A_3$, $B_1B_2B_3$, and $C_1C_2C_3$. the total number of handles is 12. This set of handles is as small as possible, since $\binom{9}{2}/3 = 12$.

The recurrence that results from this construction is:

$$C(3n, 3, 2) \leq n^2 + 3C(n, 3, 2) \tag{2}$$

Let us use $C(n)$ as shorthand for $C(n, 3, 2)$ in what follows. We claim that;

Theorem 3.7. *For n a power of 3:*

$$C(n, 3, 2) = n^2/6 - n/6$$

Proof. We already argued that $C(n) \geq n^2/6 - n/6$, so we have only to show $C(n) \leq n^2/6 - n/6$ for n a power of 3.

For the basis, $C(3) = 1$. Obviously one set of the three elements covers all three of its subsets of size two. Since $1 = 3^2/6 - 3/6$, the basis is proven.

For the induction, assume $C(n) \leq n^2/6 - n/6$. Then by Equation (2), $C(3n) \leq n^2 + 3n^2/6 - 3n/6 = 3n^2/2 - n/2 = (3n)^2/6 - (3n)/6$. □

We can get the same bound, or close to the same bound, for values of n that are not a power of 3 if we start with another basis. All optimal values of $C(n)$ up to $n = 13$ are given in [10]. For $n = 4, 5, \ldots, 13$, the values of $C(n)$ are 3, 4, 6, 7, 11, 12, 17, 19, 24, and 26.

Using Theorem 3.4, we have the following corollary to Theorem 3.7.

Corollary 3.8. *If $q = d^3$ and n is a power of 3, then we can compute all second-order marginals with a replication rate of $n^2/6 - n/6$.*

Note. It should be clear that similar corollaries, each converting a covering number result into a result about replication rate, can be derived in each subsection but we do not state them explicitly.

Observe that the bound on replication rate given by Corollary 3.8, which is equivalent to $\binom{n}{2}/3$, is exactly one third of the replication rate that would be necessary if we used a single reducer for each marginal.

We should mention that Theorem 3.7 is a known result in block design [9]. The general problem of computing covering numbers can be couched as a block-design problem, but in only a few cases (such as in this theorem) does a block design actually exist.

3.6. A slower recursion for 2nd-order marginals

There is an alternative recursion for constructing handles that offers solutions for $C(n, 3, 2)$. This recursion is not as good asymptotically as that of Section 3.5; it uses approximately $n^2/4$ handles, compared with approximately $n^2/6$ handles for the construction just seen. However, this new recursion gives solutions for any n, not just those that are powers of 3.

Note that if we attempt to address values of n that are not a power of 3 by simply rounding n up to the nearest power of 3 and using the recursive construction from the previous section, then we may increase the replication rate by a factor as large as 9, whereas the recursion in this section is never suboptimal by a factor larger than 3/2.

Let us call the n dimensions $A_1A_2B_1B_2\cdots B_{n-2}$. We choose handles of two kinds:

1. Handles that contain A_1, A_2, and one of
$$B_1, B_2, \ldots, B_{n-2}$$
There are clearly $n-2$ handles of this kind.
2. The $C(n-2)$ handles that recursively cover all pairs chosen from $B_1, B_2, \ldots, B_{n-2}$.

We claim that every marginal of size 2 is covered by one of these handles. If the marginal has neither A_1 nor A_2, then clearly it is covered by one of the handles from (2). If the marginal has both A_1 and A_2, then it is covered by any of the handles from (1). And if the marginal has one but not both of A_1 and A_2, then it has exactly one of the B_i's. Therefore, it is covered by the handle from (1) that has A_1, A_2, and that B_i.

Example 3.9. Let $n = 6$, and call the dimensions

$$ABCDEF$$

where A and B form the first group, and $CDEF$ form the second group. By rule (1), we include handles ABC, ABD, ABE, and ABF. By rule (2) we have to add a cover for each pair from $CDEF$. One choice is CDE, CDF, and DEF, for a total of seven handles. This choice is not exactly optimal, since six handles of size three suffice to cover all pairs chosen from six elements [10].

The resulting recurrence is

$$C(n) \leq n - 2 + C(n-2)$$

We claim that for odd $n \geq 3$, $C(n) \leq n^2/4 - n/2 + 1/4$. For the basis, we know that $C(3) = 1$. As $3^2/4 - 3/2 + 1/4 = 1$, the basis $n = 3$ is proved. The induction then follows from the fact that

$$n - 2 + (n-2)^2/4 - (n-2)/2 + 1/4 = n^2/4 - n/2 + 1/4$$

For even n, we could start with $C(4) = 3$. But we do slightly better if we start with the value $C(6) = 6$, given in [10]. That gives us $C(n) \leq n^2/4 - n/2$ for all even $n \geq 6$.

While this recurrence gives values of $C(n)$ that grow with $n^2/4$ rather than $n^2/6$, it does give us values that the recurrence of Section 3.5 cannot give us.

Example 3.10. The recurrence of Section 3.5 gives us

$$C(27) = 117$$

If we want a result for $n = 31$, we can apply the recurrence of this section twice, to get $C(29) \leq 27 + 117 = 143$ and $C(31) \leq 29 + 143 = 172$. In comparison, the lower bound on the number of handles needed for $n = 31$ is $\binom{31}{2}/3 = 155$.

Theorem 3.11. $C(n, 3, 2) \leq n^2/4 - n/2$

3.7. Covering 2nd-order marginals with larger handles

We can view the construction of Section 3.6 as dividing the dimensions into two groups; the first consisted of only A_1 and A_2, while the second group consisted of the remaining dimensions, which we called $B_1, B_2, \ldots, B_{n-2}$. We then divided the second-order marginals, which are pairs of dimensions, according to how the pair was divided between the groups. That is, either 0, 1, or 2 of the dimensions could be the first group $\{A_1, A_2\}$. We treated each of these three cases, as we can summarize in the table of Fig. 3.

That is, marginals with zero of A_1 and A_2 (Case 0) are covered recursively by the best possible set of handles that cover the B_i's. Marginals with both A_1 and A_2 (Case 2) are covered by many handles, since we add to A_1A_2 all possible sets of size 1 formed from the B_i's. The reason we do so is that we can then cover all the marginals belonging to Case 1, where exactly one of A_1 and A_2 is present, without adding any additional handles. That is, had we been parsimonious in Case 2, and only included one handle, such as $A_1A_2B_1$, then we would not have been able to skip Case 1.

Now, let us turn our attention to covering pairs of dimensions by sets of size larger than three; i.e., we wish to cover second-order marginals by handles of size m, for some $m \geq 4$. We can generalize the technique of Section 3.6 by using one group of size $m-1$, say $A_1, A_2, \ldots, A_{m-1}$ and another group with the remaining dimensions, $B_1, B_2, \ldots, B_{n-(m-1)}$. We can

Case	$\{A_1, A_2\}$	B_i's
0	none	cover
1	not	needed
2	A_1A_2	all B_i's

Fig. 3. How we cover each of the three cases: 0, 1, or 2 dimensions of the marginal are in the first group (A_1A_2).

form handles for Case 0, where none of the A_i's are in the marginal, recursively as we did in Section 3.6. That requires $C(n-(m-1), m, 2)$ handles. If we deal with Case $m-1$ by adding to $A_1A_2\cdots A_{m-1}$ each of the B_i's in turn, to form $n-(m-1)$ additional handles, we cover all the other cases. Of course all the cases except for Case 1, where exactly one of the A_i's is in the marginal, are vacuous. This reasoning gives us a recurrence:

$$C(n, m, 2) \leq n-(m-1) + C(n-(m-1), m, 2)$$

Using the technique suggested in Appendix A. Along with the obvious basis case $C(m, m, 2) = 1$, we get the following solution:

Theorem 3.12.

$$C(n, m, 2) \leq \frac{n^2}{2(m-1)} - \frac{n}{2} + 1 - \frac{m}{2(m-1)}$$

Note that asymptotically, this solution uses $\frac{n^2}{2(m-1)}$ handles, while the lower bound is $\frac{n(n-1)}{m(m-1)}$ handles. Therefore, this method is worse than the theoretical minimum by a factor of roughly $m/2$.

Example 3.13. Let $n = 9$ and $m = 4$. Call our dimensions $ABCDEFGHI$, where ABC is the first group and $DEFGHI$ the second. For Case $m-1$ we use the handles $ABCD$, $ABCE$, $ABCF$, $ABCG$, $ABCH$, and $ABCI$. For Case 0, we cover pairs from $DEFGHI$ optimally, using sets of size four; one such choice is $DEFG$, $DEHI$, and $FGHI$, for a total of nine handles.

3.8. A recursive-doubling method for covering 2nd-order marginals

For a sparse but infinite set of values of n, there is a better recursion for $C(n, m, 2)$. Suppose we have a cover of size $C(n, m, 2)$; we shall build a cover for $C(2n, m, 2)$. Divide the $2n$ dimensions into two groups of size n each. We can cover all pairs with one dimension in each group, as follows. Assuming m divides $2n$, start with sets consisting of $m/2$ members of one of the groups. That is, the first set consists of the first $m/2$ members of the group, the second set consists of the next $m/2$ members of that group, and so on. We need $2n/m$ such sets for each group. Then, pair the sets for each group in all possible ways, forming $4n^2/m^2$ handles of size m. These handles cover all pairs that have one member in each group.

To these add recursively constructed sets of handles for the two groups of size n. The implied recurrence for this method is:

$$C(2n, m, 2) \leq 4n^2/m^2 + 2C(n, m, 2)$$

If we use $C(m, m, 2) = 1$ as the basis, the upper bound on $C(n, m, 2)$ implied by this recurrence is:

Theorem 3.14. *If n is equal to m times a power of 2 then:*

$$C(n, m, 2) \leq 2n^2/m^2 - 1$$

Proof. The solution is an application of the general method in Appendix A. □

This bound applies only for those values of n that are m times a power of 2. It does, however, give us an upper bound that is only a factor of 2 (roughly) greater than the lower bound of $\binom{n}{2}/\binom{m}{2}$. Additionally, if we attempt to address values of n that are not m times a power of 2, by rounding up to the nearest such value, we increase n by a factor that approaches 2 for large values of n. Doing so increases the replication rate by a factor of at most 4, so the construction in this section improves on that of the previous section for sufficiently large m.

Example 3.15. Let $n = m = 4$, and suppose the dimensions are $ABCD$ in the first group and $EFGH$ in the second group. We cover all pairs of these eight dimensions with sets of size four, as follows. We first cover the singletons from $ABCD$ using two sets of size 2, say AB and CD. Similarly, we cover all singletons from $EFGH$ using EF and GH. Then we pair AB and CD in all possible ways with EF and GH, to get $ABEF$, $ABGH$, $CDEF$, and $CDGH$. Finally, add covers for each of the groups. A single handle of size four, $ABCD$, covers all pairs from the first group, and the handle $EFGH$ covers all pairs from the second group, for a total of six handles.

3.9. The general case

Finally, we offer a recurrence for $C(n, m, k)$ that works for all n and for all $m > k$. It does not approach the lower bound, but it is significantly better than using one handle per marginal. This method generalizes that of Section 3.6. We use two groups. The first has $m-k+1$ of the dimensions, say $A_1, A_2, \ldots, A_{m-k+1}$, while the second has the remaining $n-m+k-1$ dimensions. The handles are of two types:

1. One group of handles contains $A_1 A_2 \cdots A_{m-k+1}$, i.e., all of group 1, plus any $k-1$ dimensions from group 2. There are $\binom{n-m+k-1}{k-1}$ of these handles, and each has exactly m members.
2. The other handles are formed recursively to cover the dimensions of group 2, and have none of the members of group 1. There are $C(n-m+k-1, m, k)$ of these handles.

We claim that every marginal of size k is covered by one of these handles. If the marginal has at least one dimension from group 1, then it has at most $k-1$ from group 2. Therefore in is covered by the handles from (1). And if the marginal has no dimensions from group 1, then it is surely covered by a handle from (2). As a shorthand, let $C(n)$ stand for $C(n, m, k)$. The recurrence for $C(n)$ implied by this construction is

$$C(n) \leq \binom{n-m+k-1}{k-1} + C(n-m+k-1) \tag{3}$$

We shall prove that:

Theorem 3.16. *$C(n, m, k) \leq \binom{n}{k}/(m-k+1)$ for n equal to 1 plus an integer multiple of $m-k+1$.*

Proof. The proof is an induction on n.
BASIS: We know $C(m) = 1$, and $\binom{m}{k}/(m-k+1) \geq 1$ for any $1 \leq k < m$.
INDUCTION: We know from Equation (3) that

$$\binom{n-m+k-1}{k-1} + \frac{\binom{n-m+k-1}{k}}{(m-k+1)}$$

is an upper bound on $C(n)$. We therefore need to show that

$$\frac{\binom{n}{k}}{m-k+1} \geq \binom{n-m+k-1}{k-1} + \frac{\binom{n-m+k-1}{k}}{m-k+1}$$

Equivalently,

$$\binom{n}{k} \geq (m-k+1)\binom{n-m+k-1}{k-1} + \binom{n-m+k-1}{k} \tag{4}$$

The left side of Equation (4) is all ways to pick k things out of n. The right side counts a subset of these ways, specifically those ways that pick either:

1. Exactly one of the first $m-k+1$ elements and $k-1$ of the remaining elements, or
2. None of the first $m-k+1$ elements and k from the remaining elements.

Thus, Equation (4) holds, and $C(n, m, k) \leq \binom{n}{k}/(m-k+1)$ is proved. □

Theorem 3.16 applies only for certain n that form a linear progression. However, we can prove similar bounds for n that are not of the form 1 plus an integer multiple of $m-k+1$ by using a different basis case. The only effect the basis has is (possibly) to add a constant to the bound.

The bound of Theorem 3.16 plus Theorem 3.4 gives us an upper bound on the replication rate:

Corollary 3.17. *We can compute all kth-order marginals using reducers of size $q = d^m$, for $m > k$, with a replication rate of $r \leq \binom{n}{k}/(m-k+1)$.*

3.10. Handles of size 4 covering marginals of size 3

We can improve on Theorem 3.16 slightly for the special case of $m = 4$ and $k = 3$. The latter theorem gives us $C(n, 4, 3) \leq \binom{n}{3}/2$, or approximately $C(n, 4, 3) \leq n^3/12$, but we can get $C(n, 4, 3) \leq n^3/16$ by the following method, at least for a sparse but infinite set of values of n. Note that in comparison, the lower bound for $C(n, 4, 3)$ is approximately $n^3/24$.

To get the better upper bound, we generalize the strategy of Section 3.5. Let the $4n$ dimensions be placed into four groups, with n dimensions in each group. Assume the members of each group are assigned "indexes" 1 through n.

1. Form n^3 handles consisting of those sets of dimensions, one from each group, the sum of whose indexes is a multiple of n.
2. For each of the six pairs of groups, recursively cover the members of those two groups together by a set of $C(2n, 4, 3)$ handles.

Observe that every triple of dimensions is either from three different groups, in which case it is covered by one of the handles from (1), or it involves members of at most two groups, in which case it is covered by a handle from (2). We conclude that:

$$C(4n, 4, 3) \leq n^3 + 6C(2n, 4, 3)$$

This recurrence is satisfied by $C(n, 4, 3) = n^3/16$. If we start with, say, $C(4, 4, 3) = 1$, we can show $n^3/16$ is an upper bound on $C(n, 4, 3)$ for all $n \geq 4$ that is a power of two.

Theorem 3.18. $C(n, 4, 3) = n^3/16$

Aside: It appears that this algorithm and that of Section 3.5 are *not* instances of a more general algorithm. That is, there is no useful extension to $C(n, k+1, k)$ for $k > 3$.

4. Optimal handles are subcubes

We shall now demonstrate that for a given reducer size q, the largest number of marginals of a given order k that we can cover with a single reducer occurs when the reducer gets all tuples needed for a marginal of some higher order m. The proof extends the ideas found in [11,22] regarding isoperimetric inequalities for the hypercube. In general, an "isoperimetric inequality" is a lower bound on the size of the perimeter of a shape, e.g., the fact that the circle has the smallest perimeter of any shape of a given area. For particular families of graphs, these inequalities are used to show that any set of nodes of a certain size must have a minimum number of edges that connect the set to a node not in the set.

We need to use these inequalities in the opposite way – to give upper bounds on the number of edges *covered*; i.e., both ends of the edge are in the set. For example, in [6] the idea was used to show that a set of q nodes of the n-dimensional Boolean hypercube could not cover more than $\frac{q}{2}\log_2 q$ edges. That upper bound, in turn, was needed to give a lower bound on the replication rate (as a function of q, the reducer size) for MapReduce algorithms that solve the problem of finding all pairs of inputs at Hamming distance 1.

Here, we have a similar goal of placing a lower bound on replication rate for the problem of computing the kth-order marginals of a data cube of n dimensions, each dimension having extent d, using reducers of size q. The necessary subgoal is to put an upper bound on the number of subcubes of k dimensions that can be wholly contained within a set of q points of this hypercube. We shall call this function $f_{k,n}(q)$. Technically, d should be a parameter, but we shall assume a fixed d in what follows. We also note that the function does not actually depend on the dimension n of the data cube.

4.1. Binomial coefficients with noninteger arguments

Our bound on the function $f_{k,n}(q)$ requires us to use a function that behaves like the binomial coefficients $\binom{x}{y}$, but is defined for all nonnegative x and y, not just for integer values (in particular, x may be noninteger, while y will be an integer in what follows). The needed generalization uses the gamma function [1] $\Gamma(t) = \int_0^{\inf} x^{t-1}e^{-x}dx$. When t is an integer, $\Gamma(t) = (t-1)!$. But $\Gamma(t)$ is defined for nonintegral t as well. Integration by parts lets us show that Γ always behaves like the factorial of one less than its argument:

$$\Gamma(t+1) = t\Gamma(t) \tag{5}$$

If we generalize the expression for $\binom{u}{v}$ in terms of factorials from $\frac{u!}{v!(u-v)!}$ to

$$\binom{u}{v} = \frac{\Gamma(u+1)}{\Gamma(v+1)\Gamma(u-v+1)} \tag{6}$$

then we maintain the property of binomial coefficients that we need in what follows:

Lemma 4.1. *If $\binom{x}{y}$ is defined by the expression of Equation (6), then*

$$\binom{x}{y} = \binom{x-1}{y} + \binom{x-1}{y-1}$$

Proof. If we use Equation (6) to replace the binomial coefficients, we get

$$\frac{\Gamma(x+1)}{\Gamma(y+1)\Gamma(x-y+1)} = \frac{\Gamma(x)}{\Gamma(y+1)\Gamma(x-y)} + \frac{\Gamma(x)}{\Gamma(y)\Gamma(x-y+1)}$$

The above equality can be proved if we use Equation (5) to replace $\Gamma(x+1)$ by $x\Gamma(x)$, $\Gamma(x-y+1)$ by $(x-y)\Gamma(x-y)$, and $\Gamma(y+1)$ by $y\Gamma(y)$. □

In what follows, we shall use $\binom{u}{v}$ with the understanding that it actually stands for the expression given by Equation (6).

4.2. The upper bound on covered subcubes

We are now ready to prove the upper bound on the number of subcubes of dimension k that can be covered by a set of q nodes.

Theorem 4.2.

$$f_{k,n}(q) \leq \frac{q}{d^k}\binom{\log_d q}{k}$$

Proof. The proof is a double induction, with an outer induction on k and the inner induction on n.
BASIS: The basis is $k = 0$. The "0th-order" marginals are single points of the data cube, and the theorem asserts that $f_{0,n}(q) \leq q$. Since q is the largest number of points at a reducer, the basis is holds, independent of n.
INDUCTION: We assume the theorem holds for smaller values of k and all n, and also that it holds for the same value of k and smaller values of n. Partition the cube into d subcubes of dimension $n-1$, based on the value in the first dimension. Call these subcubes the *slices*. The inductive hypothesis applies to each slice. Suppose that the ith slice has x_i of the q points. Note $\sum_{i=1}^{d} x_i = q$. There are two ways a k-dimensional subcube can be covered by the original q points:

1. The subcube of dimension k has a fixed value in dimension 1, and it is contained in one of the d slices.
2. Dimension 1 is one of the k dimensions of the subcube, so the subcube has a $(k-1)$-dimensional projection in each of the slices.

Case (1) is easy. By the inductive hypothesis, there can be no more than

$$\sum_{i=1}^{d} \frac{x_i}{d^k}\binom{\log_d x_i}{k}$$

subcubes of this type covered by the q nodes. For Case (2), observe that the number of k-dimensional subcubes covered can be no larger than the number of subcubes of dimension $k-1$ that are covered by the smallest of the d slices. The inductive hypothesis also applies to give us an upper bound on these numbers. Therefore, we have an upper bound on $f_{k,n}(q)$:

$$f_{k,n}(q) \leq \sum_{i=1}^{d} \frac{x_i}{d^k}\binom{\log_d x_i}{k} + \min_i \frac{x_i}{d^{k-1}}\binom{\log_d x_i}{k-1} \tag{7}$$

We claim that Equation (7) attains its maximum value when all the x_i's are equal. We can formally prove this claim by studying the derivatives of this function, however for brevity we will only give an informal proof of this claim.

Suppose that were not true, and the largest value of the right side, subject to the constraint that $\sum_{i=1}^{d} x_i = q$, occurred with unequal x_i's. We could add ϵ to each of those x_i's that had the smallest value, and subtract small amounts from the larger x_i's to maintain the constraint that the sum of the x_i's is q. The result of this change is to increase the minimum in the second term on the right of Equation (7) at least linearly in ϵ. However, since any power of $\log x_i$ grows more slowly than linearly in x_i, there is a negligible effect on the first term on the right of Equation (7), since the sum of the x_i's does not change, and redistributing small amounts among logarithms will have an effect less than the amount that is redistributed.

Now, let us substitute $x_i = q/d$ for all x_i in Equation (7). That change gives us a true upper bound on $f_{k,n}(q)$ which is:

$$f_{k,n}(q) \leq \frac{q}{d^k}\left[\binom{\log_d q - 1}{k} + \binom{\log_d q - 1}{k-1}\right]$$

But Lemma 4.1 tells us $\binom{x}{y} = \binom{x-1}{y} + \binom{x-1}{y-1}$, so we can conclude the theorem when we let $x = \log_d q$ and $y = k$. □

We can now apply Theorem 4.2 to show that when q is the size we need to hold all tuples of the data cube that belong to an mth-order marginal for some $m > k$, then the number of kth-order marginals covered by this reducer is maximized if we send it all the tuples belonging to a marginal of order m.

Corollary 4.3. *If $q = d^m$ for some $m > k$, then no selection of q tuples for a reducer can cover more kth-order marginals than choosing all the tuples belonging to an mth-order marginal.*

Proof. When $q = d^m$, the formula of Theorem 4.2 becomes $f_{k,n}(q) = d^{m-k}\binom{m}{k}$. That is exactly the number of marginals of order k covered by a marginal of order m. To observe why, note that we can choose to fix any $m-k$ of the m dimensions that are not fixed in the mth-order marginal. We can thus choose $\binom{m}{m-k}$ sets of dimensions to fix, and this value is the same as $\binom{m}{k}$. We can fix the $m-k$ dimensions in any of d^{m-k} ways, thus enabling us to cover $d^{m-k}\binom{m}{k}$ marginals of order k. □

4.3. The lower bound on replication rate

An important consequence of Theorem 4.2 is that we can use our observations about handles and their covers to get a lower bound on replication rate.

Corollary 4.4. *If we compute all kth-order marginals using reducers of size q, then the replication rate must be at least $r \geq \binom{n}{k}/\binom{\log_d q}{k}$.*

Proof. Suppose we use some collection of reducers, where the ith reducer receives q_i inputs. There are $d^{n-k}\binom{n}{k}$ marginals that must be computed. By Theorem 4.2, we know that a reducer with q_i inputs can compute no more than $\frac{q_i}{d^k}\binom{\log_d q_i}{k}$ marginals of order k, so

$$d^{n-k}\binom{n}{k} \leq \sum_i \frac{q_i}{d^k}\binom{\log_d q_i}{k} \tag{8}$$

If we replace the occurrences of q_i in the expression $\log_d q_i$ by q (but leave them as q_i elsewhere), we know the right side of Equation (8) is only increased. Thus, Equation (8) implies:

$$d^{n-k}\binom{n}{k} \leq \frac{\binom{\log_d q}{k}}{d^k}\sum_i q_i$$

We can further rewrite as:

$$\frac{\sum_i q_i}{d^n} \geq \frac{\binom{n}{k}}{\binom{\log_d q}{k}}$$

The left side is in fact the replication rate, since it is the sum of the number of inputs received by all the reducers divided by the number of inputs. That observation proves the corollary. □

In the case $q = d^m$, Corollary 4.4 becomes $r \geq \binom{n}{k}/\binom{m}{k}$. In general, Corollary 4.4 says that the replication rate grows rather slowly with q. Multiplying q by d (or equivalently, adding 1 to m) has the effect of multiplying r by a factor $\binom{m+1}{k}/\binom{m}{k} = (m+1)/(m+1-k)$, which approaches 1 as m gets large.

5. Dimensions with different sizes

Let us now take up the case of nonuniform extents for the dimensions. Suppose that the ith dimension has d_i different values. Our first observation is that whether you focus on the lower bound on replication rate of Corollary 4.4 or the upper bound of Corollary 3.17, the replication rate is a slowly growing function of the reducer size. Thus, if the d_i's are not wildly different, we can take d to be $\max_i d_i$. If we select handles based on that assumption, many of the reducers will get fewer than d^m inputs. But the replication rate will not be too different from what it would have been had, say, all reducers been able to take the average number of inputs, rather than the maximum.

5.1. The general optimization problem

We can reformulate the problem of covering sets of dimensions that represent marginals by larger sets that represent handles as a problem with weights. Let the *weight* of the ith dimension be $w_i = \log d_i$. If q is the reducer size, then we can choose a handle to correspond to a marginal that aggregates over any set of dimensions, say $D_{i_1}, D_{i_2}, \ldots, D_{i_m}$, as long as

$$\sum_{j=1}^{m} w_{i_j} \leq \log q \tag{9}$$

Selecting a smallest set of handles that cover all marginals of size k and satisfy Equation (9) is surely an intractable problem. However, there are many heuristics that could be used. An obvious choice is a greedy algorithm. We select handles in turn, at each step selecting the handle that covers the most previously uncovered marginals.

5.2. Generalizing fixed-weight methods

Each of the methods we have proposed for selecting handles assuming a fixed d can be generalized to allow dimensions to vary. The key idea is that each method involves dividing the dimensions into several groups. We can choose to assign dimensions to groups according to their weights, so all the weights within each group are similar. We can then use the maximum weight within a group as the value of d for that group. If done correctly, that method lets us use larger handles to cover the group(s) with the smallest weights, although we still have some unused reducer capacity typically.

We shall consider one algorithm: the method described in Section 3.5 for covering second-order marginals by third-order handles. Recall this algorithm divides $3n$ dimensions into three groups of n dimensions each. We can take the first group to have the smallest n weights, the third group to have the largest weights, and the second group to have the weights in the middle. We then take the weight of a group to be the maximum of the weights of its members. We choose q to be 2 raised to the power that is the sum of the weights of the groups. Then just as in Section 3.5 we can cover all marginals that include one dimension from two different groups by selecting n^2 particular handles, each of which has a member from each group.

We complete the construction by recursively covering the pairs from a single group. The new element is that the way we handle a single group depends on its weight in relation to $\log q$. The effective value of m (the order of the marginals used as handles) may not be 3; it could be any number. Therefore, we may have to use another algorithm for the individual groups. We hope that an example will make the idea clear.

Example 5.1. Suppose we have 12 dimensions, four of which have extent up to 8 (weight 3), four of which have extent between 9 and 16 (weight 4), and four of which have extent between 17 and 64 (weight 6). We thus divide the dimensions into groups of size 4, with weights 3, 4, and 6, respectively. The appropriate reducer size is then $q = 2^{3+4+6} = 2^{13} = 8192$. We choose 16 handles of size three to cover the pairs of dimensions that are not from the same group. Now, consider the group of four dimensions with extent 8 (weight 3). With reducers of size 8192 we can accommodate marginals of order 4; in fact we need only half that reducer size to do so. Thus, a single handle consisting of all four dimensions in the group suffices.

Next, consider the group with extent 16 and weight 4. Here we can only accommodate a third-order marginal at a reducer of size 8192, so we have to use three handles of size three to cover any two of the four dimensions in this group. And for the last group, with extent 64 and weight 6, we can only accommodate a second-order marginal at a reducer, and therefore we need six handles, each of which is one of the $\binom{4}{2}$ pairs of dimensions in the last group. We therefore cover all pairs of the 12 dimensions with $16 + 1 + 3 + 6 = 26$ handles.

6. High-order marginals from low

It is common to compute marginals of one order from marginals of the next-lower order. Doing so is generally more efficient, because we avoid repeating some aggregation. Moreover, in many cases, we want marginals of all orders, or at least of several different orders, anyway. Suppose we have already computed jth-order marginals, and we want kth-order marginals for some $k > j$. We can think of the jth-order marginals as "points" in the data cube, and compute the kth-order marginal that aggregates over a set of k dimensions U by using the jth-order marginals that aggregate over some set of dimensions $T \subseteq U$, where the size of T is j.

If we can afford reducers of size $q = d^m$, then we can construct a reducer team corresponding to a set of dimensions S of size m, where each member of the team has fixed values for all the dimensions that are not in $S \cup T$. Each member of the team receives all the jth-order marginals that aggregate over T, have the fixed values for that reducer in dimensions outside $S \cup T$, and any values in the dimensions of S. Thus, the members of the team are able to compute all the kth-order marginals that aggregate over sets of dimensions U of size k, provided $T \subseteq U$ and $U \subseteq (S \cup T)$.

6.1. A new covering problem

This observation leads to a new combinatorial problem. We want to find the smallest possible set of pairs of sets (S, T) such that $\|S\| = m$, $\|T\| = j$, and for every set U of size k chosen from a set of n elements, there is some selected pair (S, T) such that $T \subseteq U \subseteq (S \cup T)$. Let $D(n, m, j, k)$ be that smallest number of pairs. We shall examine this covering problem, show the lower bound on D can be met in some simple cases, and then give a general technique for building good solutions to the D problem from solutions for C (the classical covering-sets problem from Section covering-numbers-subsect).

6.2. The lower bound for D

The lower bound on $D(n, m, j, k)$ is straightforward. There are $\binom{n}{k}$ sets that must be covered. Each pair (S, T) can cover $\binom{m}{k-j}$ sets, since a set U of size k is covered by (S, T) if and only if U consists of the j elements of T plus $k - j$ of the m elements of S. Thus,

Theorem 6.1. $D(n, m, j, k) \geq \binom{n}{k} / \binom{m}{k-j}$.

6.3. A fairly good upper bound for D

We can reduce the problem of finding good collections of covering pairs to the conventional covering-number problem, and we do so in a way that does not introduce any more redundancy (sets that are covered two or more ways) than is inherent in the problem of finding good covering sets. The central idea for obtaining an upper bound on $D(n, m, j, k)$ is to divide the sets U of size k that must be covered into groups according to the jth-lowest-numbered dimension in U, following some fixed order of the dimensions. That is, suppose the dimensions are $1, 2, \ldots, n$. We shall cover a set $U = \{x_1, x_2, \ldots, x_k\}$, where $x_1 < x_2 < \cdots < x_k$, by a pair (S, T) such that $T = \{x_1, x_2, \ldots x_j\}$, and the remaining $k - j$ elements of U are in S.

If $x_j = i$, then we shall place U in the ith group. Let us see what we need to cover Group i. First, since the members of U that are lower than i can be any set consisting of $j - 1$ of the dimensions $1, 2, \ldots, i - 1$, we need to have some pairs (S, T) where T is any set consisting of the dimension i and also $j - 1$ lower-numbered dimensions. We can then pick several values of S to associate with the same T to cover all the sets U in Group i that have this set T for the lowest j dimensions. The number of sets S we need to pick to cover all U in Group i is $C(n - i, m, k - j)$. Thus, we can cover Group i if we choose all pairs (S, T), where S is a member of a covering set of size $C(n - i, m, k - j)$, and T is any set of size j whose largest member is dimension i. Thereby, we cover all sets U in Group i. We can summarize these ideas in the following theorem.

Theorem 6.2.

$$D(n, m, j, k) \leq \sum_{i=j}^{n-k+j} \binom{i-1}{j-1} C(n - i, m, k - j)$$

Proof. First, note that in the construction above, i is at least j, and it can be at most $n - (k - j)$, since among the elements of any set U of size k covered by pair (S, T), exactly j are in T and $k - j$ are in S. Thus, the limits of the sum are the correct ones. The ith term of the sum corresponds to Group i. We observed above that to cover this group, we must allow T to be any set consisting of i plus $j - 1$ of the dimensions from 1 through $i - 1$. The number of different sets T that we need is therefore $\binom{i-1}{j-1}$. Each of these sets T must be paired with $C(n - i, m, k - j)$ sets S, in order that any set U whose j smallest elements are this set T, is certain to be covered. When we sum over all possible i, we get an upper bound on the total number of pairs we need to solve the covering problem implied by $D(n, m, j, k)$.

Thus, let U be any subset of size k of the n dimensions $1, 2, \ldots, n$. Let T be the j lowest dimensions in U, and suppose i is the jth-lowest of these dimensions. Let S be one of the sets in the selected cover for $C(n - i, m, k - j)$ that covers $U - T$. Then (S, T) is one of our selected pairs, and this pair covers U. □

It is worth noting that for every set U, there is exactly one value of T such that U is covered by a pair (S, T). If all of the covers for the various values of $C(n - i, m, k - j)$ were "perfect," in the sense that they covered each set of size $k - j$ by exactly one set of size m, then we would be sure that every set U was covered by exactly one pair, and therefore, the collection of pairs constructed was as small as possible. Of course it is not always possible to find a perfect covering set. However, this observation tells us that the degree to which this construction lacks perfection (i.e., there are sets covered by two or more pairs) is due entirely to the lack of perfection in the various covering sets. That is, the strategy we followed — breaking the problem according to the jth-lowest-numbered member of U — does not in itself introduce imperfection.

6.4. The case D(n, m, 1, 2)

The covering problem $D(n, m, 1, 2)$ corresponds to the important special case where we want to construct second-order marginals from first-order marginals, using some reducer size d^m. In this case, the formula of Theorem 6.2 simplifies considerably, to

$$D(n, m, 1, 2) \leq \sum_{i=1}^{n-1} C(n - i, m, 1)$$

since $\binom{i-1}{0}$ is 1 for any i.

Further, it is easy to see that $C(n, m, 1) = \lceil \frac{n}{m} \rceil$, since we can cover n dimensions by placing them in groups of m and covering a group by a single handle consisting of the members of that group. If it weren't for the ceiling function that is needed for the last group, which may have fewer than m members, we would have

$$D(n, m, 1, 2) = \sum_{i=1}^{n-1} (n - i)/m = n(n - 1)/2m$$

That quantity would match the lower bound of Theorem 6.1 exactly.

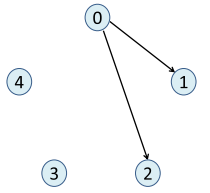

Fig. 4. Diagram for $D(5, 2, 1, 2)$.

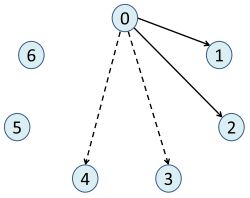

Fig. 5. Diagram for $D(7, 2, 1, 2)$.

However, because of the ceiling function, which adds, on average, $1/2$ to every term in the sum, the upper bound exceeds the lower bound by approximately $n/2$. That term is small compared with the term $n(n-1)/2m$, but it does suggest that there is a gap to be closed.

6.5. The case $D(n, 2, 1, 2)$

When $m = 2$, there is a fairly complex construction that matches the lower bound $n(n-1)/4$ given by Theorem 6.1. If $m = 2$, the pairs used for covering are of the form $(\{a, b\}, \{c\})$. These pairs cover two sets of size two: $\{a, c\}$ and $\{b.c\}$. In what follows, we shall refer to c as the *leader* of the pair and a and b as the *followers*. Every set U that is covered consists of one leader and one follower.

6.5.1. Subcase: $n = 4i + 1$

We first give a construction of an optimal selection of pairs for the case where n is one more than a multiple of four.

Example 6.3. Let us begin with the simple case of $n = 5$. Think of the five dimensions as numbered 0 through 4 and arranged in a circle, as in Fig. 4. We can represent a pair (S, T) by drawing two arrows from the leader to each of its followers. In Fig. 4, we have chosen 0 to be the leader, and 1 and 2 the followers, so this diagram represents the pair of sets $(\{1, 2\}, \{0\})$. We can select five such pairs, by picking each of the five elements x as the leader for one pair, and letting its followers be $x + 1$ and $x + 2$, where all arithmetic is done modulo 5. Since $5(5-1)/4 = 5$, the number of pairs chosen meets the lower bound.

But we claim that every set of two dimensions is covered by one of these five pairs. In proof, note that for any two integers $0 \leq x < y \leq 4$, x and y are either distance 1 or distance 2 from each other around the circle. Thus, $\{x, y\}$ will be covered by the pair whose leader is either x or y, whichever is distance 1 or distance 2 counterclockwise from the other.

We can generalize Example 6.3 to any n that is one more than a multiple of 4. For each of the n possible leaders x, we have $(n-1)/4$ pairs, which are $(\{x+1, x+2\}, \{x\})$, $(\{x+3, x+4\}, \{x\}), \ldots, (\{x + \frac{n-1}{2} - 1, x + \frac{n-1}{2}\}, \{x\})$ (all arithmetic is modulo n). Since every set of two dimensions x and y has one, say x, within distance $(n-1)/2$ counterclockwise of the other, it follows that every set U of two dimensions is covered by one of the pairs with leader x. Further, the number of pairs selected is $n(n-1)/4$, which exactly meets the lower bound.

6.5.2. Subcase: $n = 4i + 3$

Now, we give a construction for the case where n is three more than a multiple of four.

Example 6.4. Again, let us begin with a simple example: the case $n = 7$. As before, think of the seven dimensions as arranged in a circle, illustrated in Fig. 5. For each of the seven dimensions x, we have one pair with leader x and followers $x + 1$ and $x + 2$ (all arithmetic is now modulo 7). One of these pairs is suggested by the solid arrows in Fig. 5. These pairs cover all sets $U = \{x, y\}$ were one, say x, is one or two positions counterclockwise of the other.

But there are also sets U whose members are at distance three around the circle. We thus need some additional pairs suggested by the dashed arrows in Fig. 5, but we don't need all seven such pairs; four of them will do, say those with leaders 0, 1, 2, and 3. Notice that every pair of dimensions at distance three around the circle must contain one of these four leaders.

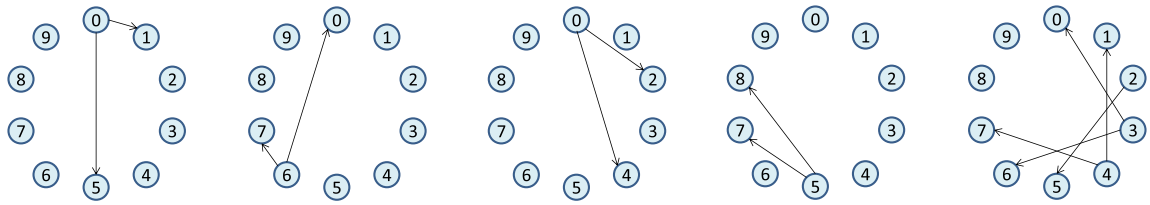

Fig. 6. Diagrams for $D(10, 2, 1, 2)$.

Thus, we use a total of 11 pairs to cover all sets of two dimensions out of $n = 7$ dimensions. However, the lower bound on the number of pairs required is $7(7-1)/4 = 10.5$. Since we must use an integer number of pairs, we know that at least 11 pairs are required, so the solution is optimal.

Example 6.4 generalizes to any n that is three more than a multiple of four. For each of the n possible leaders x, we have $(n-3)/4$ pairs, which are $(\{x+1, x+2\}, \{x\})$, $(\{x+3, x+4\}, \{x\}), \ldots, (\{x+\frac{n-3}{2}-1, x+\frac{n-3}{2}\}, \{x\})$ (all arithmetic is modulo n). In addition, for the first half of the leaders, $x = 0, 1, \ldots, \frac{n-1}{2}$ we have additional pairs

$$(\{x+\frac{n-1}{2}, x+\frac{n+1}{2}\}, \{x\})$$

The total number of pairs used is thus

$$n(n-3)/4 + (n+1)/2 = (n^2 - n + 2)/4$$

The lower bound is $(n^2 - n)/4$, i.e., 0.5 less than the number of pairs used. However, since n is three plus a multiple of four, it is easy to check that the lower bound is a half-integer, while our proposed selection of pairs is an integer number of pairs. Since any feasible solution must consist of an integer number of pairs, we know no better solution exists.

6.5.3. Subcase: even n

Last, we look at the construction for the case where n is even. This construction is more complicated than the previous subcases, so we shall start with a larger example to make the general pattern more clear.

Example 6.5. Let $n = 10$. As before, we want to think of the dimensions as arranged in a circle, in this case, numbered 0 through 9; see Fig. 6. There are 45 sets U of size two, which we can classify by the distance between the members of the set around the circle. There are ten sets at each of distances one through four. There are also five sets at distance exactly half way around the circle, at distance five, for instance $\{0, 5\}$ or $\{1, 6\}$. We do not want ten pairs to cover sets of distance five. If we did, each such set would be covered twice, and we would not be able to match the lower bound, which in the case of $n = 10$ is $10(10-1)/4 = 22.5$, or 23, since the number of chosen pairs must be an integer.

Thus, we will choose only five pairs to have a leader and a follower that are at distance five. As suggested by Fig. 6(a), we shall use pairs with leaders x in the low half (0 through 4) with two followers: $x+1$ and $x+5$. These pairs cover all the sets at distance five and also cover half of the pairs at distance one, namely $\{0, 1\}$ through $\{4, 5\}$.

However, we also have to cover the other five pairs at distance one, namely $\{5, 6\}$ through $\{9, 0\}$. We'll use leaders 5 through 9, as suggested in Fig. 6(b). But we also have to add second followers to these pairs, and we use the largest available distance, four in this case. Thus, the next five selected pairs will not only cover the remaining five sets at distance one, but will also cover five of the sets at distance four, namely $\{5, 9\}$ through $\{9, 3\}$.

Now, we have covered all ten sets at distance one, but we have only covered half of the sets at distance 4. We combine the remaining five sets at distance four with some sets at the lowest available distance: two. The next five pairs are suggested by Fig. 6(c). In each of these pairs, the leader is one of the low dimensions, 0 through 4. It's followers are at distances two and four.

These pairs let us complete the coverage of sets at distance four, but they cover only half of the sets at distance two. Therefore, for the high dimensions, 5 through 9, we construct five more pairs that have each of those as leader, one follower at distance two, and the other follower at the highest available distance: three. Thus, we complete coverage of the sets at distance two, and also half the sets at distance three.

We finish the construction by handling the remaining five sets at distance three. But we've used twenty pairs, so we have only three pairs left if we are to match the lower bound. Notice that the sets we need to cover are $\{0, 3\}$, $\{1, 4\}$, $\{2, 5\}$, $\{3, 6\}$, and $\{4, 7\}$. The key observation is that each of these sets will have one member that is in the upper half of the low dimensions, i.e., the second quartile (rounded up, because five is not evenly divisible by two). These are 2, 3, and 4. We thus can pick each of these as leaders. Dimension 3 gets followers 0 and 6, dimension 4 gets followers 1 and 7, and dimension 2 gets follower 5. It needs another follower, but it doesn't matter which we pick, because all sets of size two are now covered.

We can generalize Example 6.5 to provide an optimal construction for any even n. Let us refer to the dimensions 0 through $\frac{n}{2}-1$ as the *low* dimensions and dimensions $n/2$ through $n-1$ as the *high* dimensions. We start by choosing groups of $n/2$ pairs. Groups alternate between having low leaders and high leaders. The first group consists of pairs $(\{x+1, x+n/2\}, \{x\})$, where x is a low dimension, and, as usual, all arithmetic is modulo n. This group covers the $n/2$ pairs at distance $n/2$ as well as half of the n pairs at distance one.

The second group consists of pairs $(\{x+1, x+\frac{n}{2}-1\}, \{x\})$, where x is a high dimension. We thus cover the remaining pairs at distance one as well as half the pairs at distance $\frac{n}{2}-1$. The third group consists of pairs

$$(\{x+2, x+\frac{n}{2}-1\}, \{x\})$$

where x is a low dimension. We proceed in this way, alternating between leaders in the low and high dimensions. The end of the pattern is slightly different, depending on whether even n is or is not divisible by 4.

For n of the form $4i+2$, as for $n=10$ in Example 6.5, for each integer i, where $1 \leq i < n/4$, we have a group of $n/2$ pairs of the form $(\{x+i, x+\frac{n}{2}-i+1\}, \{x\})$, where x is a low dimension. We also have a group of $n/2$ pairs of the form $(\{x+i, x+\frac{n}{2}-i\}, \{x\})$, where x is a high dimension. These cover all sets of two dimensions, except half of those at distance $\frac{n+2}{4}$, specifically the sets $\{x, x+\frac{n+2}{4}\}$ where x is a low dimension. If we construct pairs

$$(\{x-\frac{n+2}{4}, x+\frac{n+2}{4}\}, \{x\})$$

for $x = \frac{n-2}{4}, \ldots, \frac{n}{2}-1$, we cover the remaining pairs. Except for the first of these, which covers one set that was already covered, each of the constructed pairs is the only one covering its two sets. Since when n is of the form $4i+2$, the lower bound is a half-integer and thus can only be met by the next higher integer, we know we have an optimal construction.

The case where n is divisible by four is analogous. The major difference is that the sequence of groups ends with leaders among the low dimensions, and we have to cover sets $\{x, x+\frac{n}{4}\}$, where x is a high dimension. Now, we pick $n/4$ more pairs of the form $(\{x-\frac{n}{4}, x+\frac{n}{4}\}, \{x\})$, where x is in the range $\frac{3n}{4} \leq x < n$.

We may summarize the analysis of the various cases above with the following theorem.

Theorem 6.6. *For any n, there is a selection of*

$$\lceil n(n-1)/4 \rceil$$

pairs (S, T), where $\|S\| = 2$, $\|T\| = 1$, that covers any set U of size 2, chosen from among n dimensions. This selection uses the smallest possible collection of pairs, since the number of pairs is the least integer equal to or greater than the lower bound of Theorem 6.1.

7. Conclusions and open problems

Our goal was to minimize the communication ("replication rate") for MapReduce computations of the marginals of a data cube. We showed how strategies for assigning work to reducers so that each reducer can compute a large number of marginals of fixed order can be viewed as the problem of "covering" sets of a fixed size ("marginals") by a small number of larger sets than contain them ("handles"). We have offered lower bounds and several recursive constructions for selecting a set of handles. Except in one case, Section 3.5, there is a gap between the lower and upper bounds on how many handles we need.

- We believe there are many opportunities for finding better recursive constructions of handles.

A second important contribution was the proof that our view of the problem is valid. That is, we showed that the strategy of giving each reducer the inputs necessary to compute one marginal of higher order maximized the number of marginals a reducer could compute, given a fixed bound on the number of inputs a reducer could receive. However, this result was predicated on there being the same extent for each dimension of the data cube. We offered some modifications to the proposed algorithms for the case where the extents differ.

- There is an opportunity to find better or more general approaches to the weighted covering problem, and thus to find better algorithms for computing marginals by MapReduce for the general case of unequal extents.
- There should be a generalization of Theorem 4.2 to the case of unequal extents. Part of the problem is that when the dimensions have different extents, the marginals require different numbers of inputs. Therefore, if we choose to assign one higher-order marginal to a reducer, and that marginal aggregates over many dimensions with small extent, this reducer can cover many marginals with a relatively small number of inputs. But if we want to compute all marginals of a fixed order, we must also compute the marginals that aggregate over dimensions with large extents. If the number of inputs a reducer can receive is fixed, then those marginals must be computed by reducers that cover relatively few marginals. Thus, an upper bound on the number of marginals that can be covered by a reducer of fixed size will be unrealistic, and not attainable by all the reducers used in a single MapReduce algorithm.

Finally, we looked at the matter of computing kth-order marginals from jth-order marginals, where $j < k$. We expressed the design of algorithms as a variation on the classical covering-numbers problem, where sets U are covered by pairs of sets (S, T), and $T \subseteq U \subseteq (S \cup T)$. We offered one promising approach, where we reduce the selection of such pairs to the covering-numbers problem, as well as giving some simple cases that are optimal or close to it.

- We believe there are good opportunities to study the selection of such pairs. There may be other cases where constructions can be proved optimal, and it is worth considering recursive constructions analogous to those for covering numbers.

Acknowledgments

We wish to thank Magdalena Balazinska for helpful comments and discussions about SciDB. Also, we thank Allen van Gelder for pointing out the relationship between optimal covering numbers and block designs.

Appendix A. Solving Recurrences

We propose several recurrences that describe inductive constructions of sets of handles. While we do not want to explain how one discovers the solution to each recurrence, there is a general pattern that can be used by the reader who wants to see how the solutions are derived; see [8].

A recurrence like $C(n) \leq n - 2 + C(n-2)$ from Section 3.6 will have a solution that is a quadratic polynomial, say $C(n) = an^2 + bn + c$. It turns out that the constant term c is needed only to make the basis hold, but we can get the values of a and b by replacing the inequality by an equality, and then recognizing that the terms depending on n must be 0. In this case, we get

$$an^2 + bn + c = n - 2 + a(n-2)^2 + b(n-2) + c$$

or

$$an^2 + bn + c = n - 2 + an^2 - 4an + 4a + bn - 2b + c$$

Cancelling terms and bringing the terms with n to the left, we get

$$n(4a - 1) = 4a - 2b - 2$$

Since a function of n cannot be a constant unless the coefficient of n is 0, we know that $4a - 1 = 0$, or $a = 1/4$. The right side of the equality must also be 0, so we get $4(1/4) - 2b - 2 = 0$, or $b = -1/2$. We thus know that $C(n) = n^2/4 - n/2 + c$ for some constant c, depending on the basis value.

References

[1] Gamma function, https://en.wikipedia.org/wiki/Gamma_function.
[2] Scidb, http://scidb.org.
[3] A. Abelló, J. Ferrarons, O. Romero, Building cubes with mapreduce, in: DOLAP 2011, ACM 14th International Workshop on Data Warehousing and OLAP, Glasgow, United Kingdom, Proceedings, October 28, 2011, pp. 17–24.
[4] A. Abouzeid, K. Bajda-Pawlikowski, D.J. Abadi, A. Rasin, A. Silberschatz, Hadoopdb: an architectural hybrid of mapreduce and DBMS technologies for analytical workloads, Proc. VLDB 2 (1) (2009) 922–933.
[5] F.N. Afrati, S. Dolev, S. Sharma, J.D. Ullman, Bounds for overlapping interval join on mapreduce, in: Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Brussels, Belgium, March 27th, 2015, pp. 3–6.
[6] F.N. Afrati, A.D. Sarma, S. Salihoglu, J.D. Ullman, Upper and lower bounds on the cost of a map-reduce computation, Proc. VLDB 6 (4) (2013) 277–288.
[7] F.N. Afrati, J.D. Ullman, Matching bounds for the all-pairs mapreduce problem, in: 17th International Database Engineering & Applications Symposium, IDEAS '13, Barcelona, Spain, October 9–11, 2013, pp. 3–4.
[8] A.V. Aho, J.D. Ullman, Foundations of Computer Science: C Edition, W.H. Freeman, 1995.
[9] I. Anderson, Combinatorial Designs and Tournaments.
[10] D. Applegate, E.M. Rains, N.J.A. Sloane, On asymmetric coverings and covering numbers, J. Comb. Des. 11 (2003) 2003.
[11] B. Bollabas, Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability, Cambridge University Press, 1986.
[12] P.G. Brown, Overview of scidb: large scale array storage, processing and analysis, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6–10, 2010, pp. 963–968.
[13] J. Cohen, B. Dolan, M. Dunlap, J.M. Hellerstein, C. Welton, MAD skills: new analysis practices for big data, Proc. VLDB 2 (2) (2009) 1481–1492.
[14] J.N. Cooper, R.B. Ellis, A.B. Kahng, Asymmetric binary covering codes, J. Comb. Theory, Ser. A 100 (2) (2002) 232–249.
[15] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, in: OSDI, 2004.
[16] C. Doulkeridis, K. Nørvåg, A survey of large-scale analytical query processing in mapreduce, VLDB J. 23 (3) (2014) 355–380.
[17] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, J.D. Ullman, Computing iceberg queries efficiently, in: VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, New York City, New York, USA, August 24–27, 1998, pp. 299–310.
[18] E. Friedman, P.M. Pawlowski, J. Cieslewicz, Sql/mapreduce: a practical approach to self-describing, polymorphic, and parallelizable user-defined functions, Proc. VLDB 2 (2) (2009) 1402–1413.
[19] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-total, in: Proceedings of the Twelfth International Conference on Data Engineering, New Orleans, Louisiana, February 26–March 1, 1996, pp. 152–159.
[20] V. Harinarayan, A. Rajaraman, J.D. Ullman, Implementing data cubes efficiently, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4–6, 1996, pp. 205–216.

[21] J.M. Hellerstein, C. Ré, F. Schoppmann, D.Z. Wang, E. Fratkin, A. Gorajek, K.S. Ng, C. Welton, X. Feng, K. Li, A. Kumar, The madlib analytics library or MAD skills, the SQL, Proc. VLDB 5 (12) (2012) 1700–1711.
[22] S. Hoory, N. Linial, A. Widgerson, Expander graphs and their applications, Bull. Am. Math. Soc. (N.S.) 43 (4) (2006) 439–561.
[23] S. Lee, J. Kim, Y.-S. Moon, W. Lee, Efficient distributed parallel top-down computation of rolap data cube using mapreduce, in: A. Cuzzocrea, U. Dayal (Eds.), Data Warehousing and Knowledge Discovery, in: Lect. Notes Comput. Sci., vol. 7448, Springer, Berlin, Heidelberg, 2012, pp. 168–179.
[24] A. Nandi, C. Yu, P. Bohannon, R. Ramakrishnan, Data cube materialization and mining over mapreduce, IEEE Trans. Knowl. Data Eng. 24 (10) (2012) 1747–1759.
[25] K. Rohitkumar, S. Patil, Data cube materialization using mapreduce, Int. J. Inn. Res. Comput. Commun. Eng. 11 (2) (2014) 6506–6511.
[26] K. Sergey, K. Yury, Applying map-reduce paradigm for parallel closed cube computation, in: The First International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDS 2009, Gosier, Guadeloupe, France, 1–6 March 2009, pp. 62–67.
[27] E. Soroush, M. Balazinska, D.L. Wang, Arraystore: a storage manager for complex parallel array processing, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12–16, 2011, pp. 253–264.
[28] J.D. Ullman, Designing good mapreduce algorithms, ACM Crossroads 19 (1) (2012) 30–34.
[29] S. Vemuri, M. Varshney, K. Puttaswamy, R. Liu, Execution primitives for scalable joins and aggregations in map reduce, Proc. VLDB Endow. 7 (13) (Aug. 2014) 1462–1473.
[30] B. Wang, H. Gui, M. Roantree, M.F. O'Connor, Data cube computational model with hadoop mapreduce, in: WEBIST 2014 – Proceedings of the 10th International Conference on Web Information Systems and Technologies, vol. 1, Barcelona, Spain, 3–5 April, 2014, pp. 193–199.
[31] Z. Wang, Y. Chu, K. Tan, D. Agrawal, A. El Abbadi, X. Xu, Scalable data cube analysis over big data, CoRR abs/1311.5663, 2013.
[32] Z. Wang, Q. Fan, H. Wang, K.-L. Tan, D. Agrawal, A. El Abbadi, Pagrol: parallel graph olap over large-scale attributed graphs, in: 2014 IEEE 30th International Conference on Data Engineering (ICDE), IEEE, 2014, pp. 496–507.
[33] D. Zhang, Integrative Text Mining and Management in Multidimensional Text Databases, PhD thesis, University of Illinois at Urbana-Champaign, 2012.