Assignment of Different-Sized Inputs in MapReduce[*]

Foto Afrati[†]
School of Electrical and
Computing Engineering
National Technical University
of Athens, Greece
afrati@softlab.ece.ntua.gr

Shlomi Dolev[‡]
Department of Computer
Science
Ben-Gurion University of the
Negev, Israel
dolev@cs.bgu.ac.il

Ephraim Korach
Department of Industrial
Engineering and Management
Ben-Gurion University of the
Negev, Israel
korach@bgu.ac.il

Shantanu Sharma
Department of Computer
Science
Ben-Gurion University of the
Negev, Israel
sharmas@cs.bgu.ac.il

Jeffrey D. Ullman
Department of Computer
Science
Stanford University
USA
ullman@cs.stanford.edu

ABSTRACT

A MapReduce algorithm can be described by a *mapping schema*, which assigns inputs to a set of reducers, such that for each required output there exists a reducer that receives all the inputs that participate in the computation of this output. Reducers have a capacity, which limits the sets of inputs that they can be assigned. However, individual inputs may vary in terms of size. We consider, for the first time, mapping schemas where input sizes are part of the considerations and restrictions. One of the significant parameters to optimize in any MapReduce job is communication cost between the map and reduce phases. The communication cost can be optimized by minimizing the number of copies of inputs sent to the reducers. The communication cost is closely related to the number of reducers of constrained capacity that are used to accommodate appropriately the inputs, so that the requirement of how the inputs must meet in a reducer is satisfied. In this work, we consider a family of problems where it is required that each input meets with each other input in at least one reducer. We also consider a slightly different family of problems in which, each input of a set, X, is required to meet each input of another set, Y, in at least one reducer. We prove that finding an optimal mapping schema for these families of problem is NP-hard, and

present several approximation algorithms for finding a near optimal mapping schema.

1. INTRODUCTION

MapReduce (was introduced by Dean and Ghemawat [6]) is a programming system used for parallel processing of large-scale data. Input data is processed by the *map phase* that applies a user-defined map function to produce intermediate data (of the form $\langle key, value \rangle$). Afterwards, intermediate data is processed by the *reduce phase* that applies a user-defined reduce function to keys and their associated values. The final output is provided by the reduce phase. A detailed description of MapReduce can be found in Chapter 2 of [11].

Reducers and Reducer Capacity. An important parameter to be considered in MapReduce algorithms is the "reducer capacity." A *reducer* is an application of the reduce function to a single key and its associated list of $value$s. The *reducer capacity* is an upper bound on the sum of the sizes of the $value$s that are assigned to the reducer. For example, we may choose the reducer capacity to be the size of the main memory of the processors on which the reducers run. We always assume in this paper that all the reducers have an identical capacity, denoted by q.

The term *reducer capacity* is introduced, here, for the first time. There are various works in the field of MapReduce algorithms design (*e.g.*, [10, 13, 2, 7, 12, 3]); none of them considers the reducer capacity.

Motivation and Examples. We demonstrate a new aspect of the reducer capacity in the scope of several special cases. One useful special case is where an output depends on *exactly* two inputs. We present two examples where each output depends on exactly two inputs and define two problems that are based on these examples.

Similarity-join. Similarity-join is used to find the similarity between any two inputs, *e.g.*, Web pages or documents. A set of m inputs (*e.g.*, Web pages) $WP = \{wp_1, wp_2, \ldots, wp_m\}$, a similarity function $sim(x, y)$, and a similarity threshold t are given, and each pair of inputs $\langle wp_x, wp_y \rangle$ corresponds to one output such that $sim(wp_x, wp_y) \geq t$.

It is necessary to compare all-pairs of inputs when the similarity measure is sufficiently complex that shortcuts like locality-sensitive hashing are not available. Therefore, it is mandatory that every two inputs (Web pages) of the given input
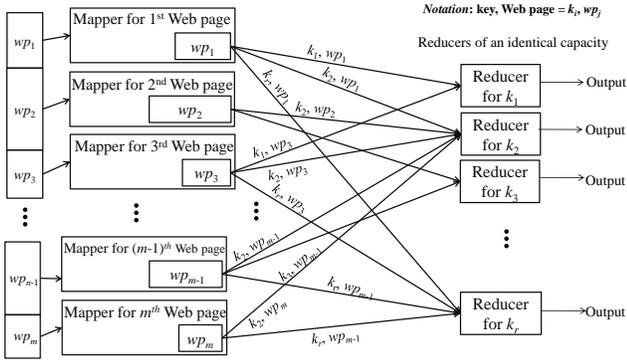
Figure 1: Similarity-join example.

Figure 2: Skew join example for a heavy hitter, b_1.

set (WP) are compared. The similarity-join is useful in various applications, mentioned in [4], *e.g.*, near-duplicate document detection and collaborative filtering.

In Figure 1, an example of similarity-join is given as it is applied to Web pages. We are given a set of m Web pages, and a mapper (a mapper is an application of the map function to a single input) would take only a single Web page, and a reducer produces pairs of every two Web pages and their similarity score.

Skew join of two relations $X(A, B)$ and $Y(B, C)$. The join of relations $X(A, B)$ and $Y(B, C)$, where the joining attribute is B, provides the output tuples $\langle a, b, c \rangle$, where (a, b) is in X and (b, c) is in Y. One or both of the relations X and Y may have a large number of tuples with the same B-value. A value of the joining attribute B that occurs many times is known as a *heavy hitter*. In skew join of $X(A, B)$ and $Y(B, C)$, all the tuples of both the relations with the same heavy hitter should appear together to provide the output tuples.

In Figure 2, b_1 is considered as a heavy hitter, hence, it is required that all the tuples of $X(A, B)$ and $Y(B, C)$ with the heavy hitter, b_1, should appear together to provide the desired output tuples, $\langle a, b_1, c \rangle$ ($a \in A, b_1 \in B, c \in C$), which depend on exactly two inputs. It is worth noting that all the tuples of both the relations that have a common value of the joining attribute B, except b_1, are now also required to appear together to provide the remaining output tuples.

Problem Statement. We define two problems where exactly two inputs are required for computing an output, as follows:

All-to-All problem. In the *all-to-all* (*A2A*) problem, a set of inputs is given, and each pair of inputs corresponds to one output. Computing common friends on a social networking site, similarity-join [4, 15, 14], the drug-interaction problem [13], and the Hamming distance 1 problem [2] are examples of tasks for which an output depends on exactly two inputs, and the set of outputs requires us to consider each pair of inputs.

X-to-Y problem. In the *X-to-Y* (*X2Y*) problem, two disjoint sets X and Y are given, and each pair of elements $\langle x_i, y_j \rangle$, where $x_i \in X, y_j \in Y, \forall i, j$, of the sets X and Y corresponds to one output. Skew join and outer product or tensor product are examples.

The *communication cost*, *i.e.*, the total amount of data transmitted from the map phase to the reduce phase, is a significant factor in the performance of a MapReduce algorithm. The communication cost comes with tradeoff in the degree of parallelism, however.

A reducer of large enough capacity can be used to accommodate all the given inputs, and provide the desired outputs. This results in the minimum communication cost but also in the minimum parallelism. Higher parallelism requires more
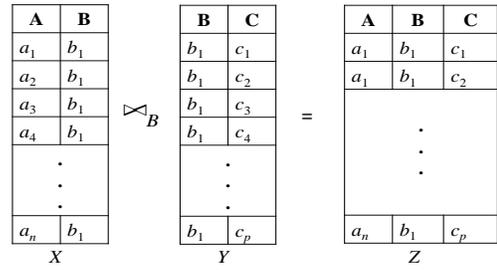
reducers (hence, of smaller reducer capacity), and hence a larger communication cost (because the copies of the given inputs are required to be assigned to more reducers).

A substantial level of parallelism can be achieved with fewer reducers, and hence, yield a smaller communication cost. Thus, we focus on minimizing the total number of reducers, for a given reducer capacity q. A smaller number of reducers results in a smaller communication cost. Thus, the reducer capacity, q, reflects also the degree of parallelism we want, since if we want more parallelism we can explore the problem in question for smaller q.

Related Work. Afrati et al. [2] presents a model for MapReduce algorithms where an output depends on two inputs, and shows a tradeoff between communication cost and parallelism. In [3], the authors consider the case where each pair of inputs produces an output and present an upper bound that meets the lower bound on communication cost as a function of the total number of inputs sent to a reducer. However, both in [2] and [3] the authors regard the reducer capacity in terms of the total number of inputs (assuming each input is of an identical size) sent to a reducer. Our setting is closely related to the settings given by Afrati et al. [2] but we allow the input sizes to be different. Thus, we consider a more realistic setting for MapReduce algorithms that can be used in various practical scenarios.

Our Contribution. In this paper, we provide:
• Mapping schemas for the *A2A* and the *X2Y* problems, which take into account the fact that inputs have different sizes, while all the reducers have an identical and fixed capacity (Section 2).
• A tradeoff between the reducer capacity and the total number of reducers, which is demonstrated using similarity-join and skew join (Section 2). A tradeoff between the reducer capacity and the parallelism at the reduce phase, and a tradeoff between the reducer capacity and the communication cost is detailed in Section 2 as well.
• A proof that the *A2A mapping schema problem* for one and two reducers has a polynomial solution, and the same problem is NP-hard in the case of more than two reducers of an identical capacity (Section 3). Also, we prove that the *X2Y mapping schema problem* for one reducer has a polynomial solution, and the same problem is NP-hard in the case of more than one reducer of an identical capacity (Section 3).
• A set of heuristics, for the *A2A mapping schema problem* and the *X2Y mapping schema problem*, that is based on First-Fit Decreasing (FFD) or Best-Fit Decreasing (BFD) bin-packing algorithm, and a pseudo polynomial bin-packing algorithm (Sections 4 and 5).

2. MAPPING SCHEMA AND TRADEOFFS

Our system setting is an extension of the standard system setting [2] for MapReduce algorithms, where we consider, for the first time, inputs of different sizes. The system setting is suitable for a variety of problems where *exactly* two inputs are required for an output. To demonstrate the influence of the extra considerations, we define

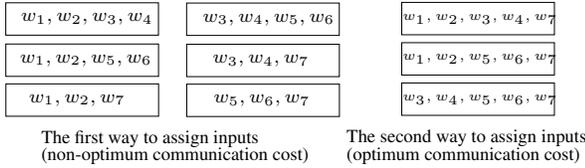$$w_1 = w_2 = w_3 = 0.20q, w_4 = w_5 = 0.19q, w_6 = w_7 = 0.18q$$

w_1, w_2, w_3, w_4	w_3, w_4, w_5, w_6	w_1, w_2, w_3, w_4, w_7
w_1, w_2, w_5, w_6	w_3, w_4, w_7	w_1, w_2, w_5, w_6, w_7
w_1, w_2, w_7	w_5, w_6, w_7	w_3, w_4, w_5, w_6, w_7

The first way to assign inputs (non-optimum communication cost) | The second way to assign inputs (optimum communication cost)

Figure 3: An example to the *A2A mapping schema problem*.

mapping schema and consider the communication cost tradeoff, as we elaborate next.

Mapping Schema. A mapping schema is an assignment of the set of inputs to some given reducers under the following two constraints:

• A reducer is assigned inputs whose sum of the sizes is less than or equal to the reducer capacity q.

• For each output, we must assign the corresponding inputs to at least one reducer in common.

Tradeoffs. The following tradeoffs appear in MapReduce algorithms and in particular in our setting:

• A tradeoff between the reducer capacity and the total number of reducers. For example, large reducer capacity allows the use of a smaller number of reducers.

• A tradeoff between the reducer capacity and the parallelism. For example, large reducer capacity results in less parallelism.

• A tradeoff between the reducer capacity and the communication cost.

In the subsequent subsections, we present two types of mapping schema problems with fitting examples and explain the three tradeoffs.

2.1 The A2A Mapping Schema Problem

The *A2A mapping schema problem* is defined in terms of a set of *inputs*, a size for each input, a set of reducers, and a mapping from outputs to sets of inputs. An instance of the *A2A mapping schema problem* consists of a set of m inputs whose input size set is $W = \{w_1, w_2, \ldots, w_m\}$ and a set of z reducers of capacity q. A solution to the *A2A mapping schema problem* assigns every pair of inputs to at least one reducer in common, without exceeding q at any reducer.

Example. We are given a set of seven inputs $I = \{i_1, i_2, \ldots, i_7\}$ whose size set is $W = \{0.20q, 0.20q, 0.20q, 0.19q, 0.19q, 0.18q, 0.18q\}$ and reducers of capacity q. In Figure 3, we show two different ways that we can assign the inputs to reducers. The best we can do to minimize the communication cost is to use three reducers. However, there is less parallelism at the reduce phase as compared to when we use six reducers. Observe that when we use six reducers, then all reducers have a lighter load, since each reducer may have capacity less than $0.8q$.

Explanation of tradeoffs. Similarity-join is an example of the *A2A mapping schema problem*, and the tradeoffs can also be explained with the help of similarity-join example. Consider that m Web pages are of $w_1, w_2, \ldots, w_m$ sizes. A single reducer of capacity $q = w_1 + w_2 + \ldots + w_m$ is able to find the similarity between every pair of Web pages. The use of only one reducer results in no parallelism at the reduce phase. But at the same time, the use of a single reducer yields the minimum possible communication cost. On the other hand, in case q is small but is still greater than or equal to $w_i + w_j$, for any i and j, then more reducers are required, and a higher level of parallelism is obtained. But, at the same time, the communication cost is higher, since every input is communicated to $m - 1$ reducers.

Inputs of set X: $w_1 = w_2 = 0.25q, w_3 = w_4 = 0.24q, w_5 = w_6 = 0.23q, w_7 = w_8 = 0.22q, w_9 = w_{10} = 0.21q, w_{11} = w_{12} = 0.20q$

Inputs of set Y: $w_1' = w_2' = 0.25q, w_3' = w_4' = 0.24q$

The first way to assign inputs using 12 reducers:

w_1, w_2, w_1', w_2'	w_1, w_2, w_3', w_4'
w_3, w_4, w_1', w_2'	w_3, w_4, w_3', w_4'
w_5, w_6, w_1', w_2'	w_5, w_6, w_3', w_4'
w_7, w_8, w_1', w_2'	w_7, w_8, w_3', w_4'
w_9, w_{10}, w_1', w_2'	w_9, w_{10}, w_3', w_4'
$w_{11}, w_{12}, w_1', w_2'$	$w_{11}, w_{12}, w_3', w_4'$

The second way to assign inputs using 16 reducers:

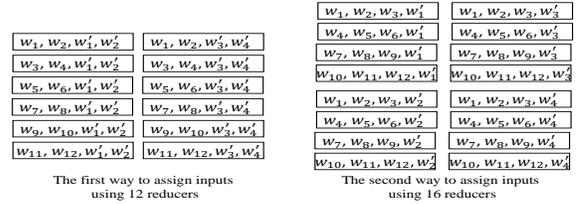|	
w_1, w_2, w_3, w_1'	w_1, w_2, w_3, w_3'
w_4, w_5, w_6, w_1'	w_4, w_5, w_6, w_3'
w_7, w_8, w_9, w_1'	w_7, w_8, w_9, w_3'
$w_{10}, w_{11}, w_{12}, w_1'$	$w_{10}, w_{11}, w_{12}, w_3'$
w_1, w_2, w_3, w_2'	w_1, w_2, w_3, w_4'
w_4, w_5, w_6, w_2'	w_4, w_5, w_6, w_4'
w_7, w_8, w_9, w_2'	w_7, w_8, w_9, w_4'
$w_{10}, w_{11}, w_{12}, w_2'$	$w_{10}, w_{11}, w_{12}, w_4'$

Figure 4: An example to the *X2Y mapping schema problem*.

2.2 The X2Y Mapping Schema Problem

The *X2Y mapping schema problem* is defined in terms of two disjoint sets X and Y of *inputs*, a size for each input, a set of reducers, and a mapping from outputs to sets of inputs. An instance of the *X2Y mapping schema problem* consists of two disjoint sets X and Y and a set of reducers of capacity q. The inputs of the set X are of sizes $w_1, w_2, \ldots, w_m$, and the inputs of the set Y are of sizes $w_1', w_2', \ldots, w_n'$. A solution to the *X2Y mapping schema problem* assigns every two inputs, the first from one set, X, and the second from the other set, Y, to at least one reducer in common, without exceeding q at any reducer.

Example. We are given two sets X of 12 inputs and Y of 4 inputs, see Figure 4, and reducers of capacity q. We show that we can assign each input of the set X with each input of the set Y in two ways. In order to minimize the communication cost, the best way is to use 12 reducers. Note that we cannot obtain a solution for the given inputs using less than 12 reducers. However, the use of 12 reducers results in less parallelism at the reduce phase as compared to when we use 16 reducers.

Explanation of tradeoffs. Skew join of two relations $X(A, B)$ and $Y(B, C)$ for a heavy hitter is an example of the *X2Y mapping schema problem*. We also explain the tradeoffs using the example of skew join. We assume that both the relations $X(A, B)$ and $Y(B, C)$ have only a single heavy hitter, say b_1. Note that we do not consider tuples with no heavy hitter.

A reducer of capacity q that is sufficient to hold all the tuples of $X(A, B)$ and $Y(B, C)$ with the heavy hitter results in the minimum communication cost. However, due to a single reducer, there is no parallelism at the reduce phase. In addition, a single reducer takes a long time to produce all the desired output tuples of the heavy hitter.

In order to decrease the time (or when q is small but still enough to hold only two tuples, the first from $X(A, B)$ and the second from $Y(B, C)$, which have the common B-value), we can restrict reducers in a way that they can hold many tuples, but not all the tuples with the heavy-hitter-value. In this case, we use more reducers, which result in a higher level of parallelism at the reduce phase. But, there is a higher communication cost, since each tuple with the heavy hitter must be sent to more than one reducer.

3. INTRACTABILITY OF FINDING A MAPPING SCHEMA

In this section, we will show that the *A2A* and the *X2Y* mapping schema problems do not possess a polynomial solution. In other words, we will show that the assignment of *two required inputs* to the minimum number of reducers to find solutions to the *A2A* and the *X2Y* mapping schema problems cannot be achieved in polynomial time.

NP-hardness of the A2A Mapping Schema Problem. A set of inputs $I = \{i_1, i_2, \ldots, i_m\}$ whose input size set is $W = \{w_1, w_2, \ldots, w_m\}$ and a set of reducers $R = \{r_1, r_2, \ldots, r_z\}$, are

an input instance to the *A2A mapping schema problem*. The *A2A mapping schema problem* is a decision problem that asks whether or not there exists a mapping schema for the given input instance such that every input, i_x, is assigned with every other input, i_y, to at least one reducer in common. An answer to the *A2A mapping schema problem* will be "yes," if for each pair of inputs ($\langle i_x, i_y \rangle$), there is at least one reducer that holds them.

Now we prove that the *A2A mapping schema problem* has a polynomial solution to one and two reducers. If there is only one reducer, then the answer is "yes" if and only if the sum of the input sizes $\sum_{i=1}^{m} w_i$ is at most q. On the other hand, if $q < \sum_{i=1}^{m} w_i$, then the answer is "no." In case of two reducers, if a single reducer is not able to accommodate all the given inputs, then there must be at least one input that is assigned to only one of the reducers, and hence, this input is not paired with all the other inputs. In that case, the answer is "no." Therefore, we achieve a polynomial solution to the *A2A mapping schema problem* for one and two reducers. Next, we will prove that the *A2A mapping schema problem* is an NP-hard problem for $z > 2$ reducers.

Theorem 1 *The problem of finding whether a mapping schema of m inputs of different input sizes exists, where every two inputs are assigned to at least one of $z \geq 3$ identical-capacity reducers, is NP-hard.*

Proof is omitted from here and given in [1].

NP-hardness of the X2Y Mapping Schema Problem. Two sets of inputs, $X = \{i_1, i_2, \ldots, i_m\}$ whose input size set is $W_x = \{w_1, w_2, \ldots, w_m\}$ and $Y = \{i'_1, i'_2, \ldots, i'_n\}$ whose input size set is $W_y = \{w'_1, w'_2, \ldots, w'_n\}$, and a set of reducers $R = \{r_1, r_2, \ldots, r_z\}$ are an input instance to the *X2Y mapping schema problem*. The *X2Y mapping schema problem* is a decision problem that asks whether or not there exists a mapping schema for the given input instance such that each input of the set X is assigned with each input of the set Y to at least one reducer in common. An answer to the *X2Y mapping schema problem* will be "yes," if for each pair of inputs, the first from X and the second from Y, there is at least one reducer that has both those inputs.

The *X2Y mapping schema problem* has a polynomial solution for the case of a single reducer. If there is only one reducer, then the answer is "yes" if and only if the sum of the input sizes $\sum_{i=1}^{m} w_i + \sum_{i=1}^{n} w'_i$ is at most q. On the other hand, if $q < \sum_{i=1}^{m} w_i + \sum_{i=1}^{n} w'_i$, then the answer is "no." Next, we will prove that the *X2Y mapping schema problem* is an NP-hard problem for $z > 1$ reducers.

Theorem 2 *The problem of finding whether a mapping schema of m and n inputs of different input sizes that belongs to set X and set Y, respectively, exists, where every two inputs, the first from X and the second from Y, are assigned to at least one of $z \geq 2$ identical-capacity reducers, is NP-hard.*

Proof is omitted from here and given in [1].

4. HEURISTICS FOR THE A2A MAPPING SCHEMA PROBLEM

Since the *A2A Mapping Schema Problem* is NP-hard, in polynomial time we cannot assign each pair of inputs to the minimum number of reducers, which results in the optimum communication between the map phase and the reduce phase. In this section, we propose several heuristics for the *A2A mapping schema problem* that are based on bin-packing algorithms, selection of a prime number p, and division of inputs into two sets based on their sizes. In addition,

the proposed heuristics assume that a *fixed* reducer capacity q is given. The heuristics have two cases depending on the sizes of the inputs, as follows:
1. Input sizes are upper bounded by $\frac{q}{2}$,
2. One input is of size, say w_i, greater than $\frac{q}{2}$, but less than q, and all the other inputs have size less than or equal to $q - w_i$,

The idea of the heuristics is: if the two largest inputs are greater than the given reducer capacity q, then there is no solution to the *A2A mapping schema problem* because these two inputs cannot be assigned to a single reducer in common.

Parameters for bounds analysis. We analyze our heuristics on three parameters, as follows:
1. Minimum number of reducers, $r(m, q)$. The minimum number of reducers of capacity q that can solve the *A2A* (and *X2Y*) mapping schema problem(s) for the given inputs with certain size restrictions.
2. Replication of individual inputs. Inputs of different sizes need to be replicated at different numbers of reducers. We therefore need to consider the minimum number of reducers to which each individual input is sent.
3. The total communication cost, c. The total communication cost is defined to be the sum of all the bits that are required to transfer from the map phase to the reduce phase.

4.1 All the inputs sizes are upper bounded by $\frac{q}{2}$

We first consider the case where all the input sizes are at most $\frac{q}{2}$ size. We consider the following two cases in this section: (*i*) all the inputs are potentially of different sizes but at most size $\frac{q}{2}$, and (*ii*) all the inputs sizes are almost equal or there are a lot of inputs of very small size. Particularity, all the inputs are of size at most $\frac{q}{k}$, where $k > 1$,

4.1.1 Different-sized inputs but at most size $\frac{q}{2}$

We first provide a heuristic for inputs of potentially different sizes, where the largest input can have at most size $\frac{q}{2}$. The heuristic uses a bin-packing algorithm to place the given m inputs into bins of size $\frac{q}{2}$. Before going into details of the heuristic, we look at the lower bound on the replication of an input i (of size w_i), the total number of reducers, and the total communication cost between the map phase and the reduce phase. The bounds are given in Table 1.

Theorem 3 (Replication of individual inputs) *For a set of m inputs and a given reducer capacity q, an input i of size $w_i < q$ is required to be sent to at least $\left\lceil \frac{s - w_i}{q - w_i} \right\rceil$ reducers for a solution to the A2A mapping schema problem, where s is the sum of all the input sizes.*

PROOF. Consider an input i of size w_i. The input i can share a reducer with inputs whose sum of the sizes is at most $q - w_i$. In order to assign the input i with all the remaining $m - 1$ inputs, it is required to assign subsets of the $m - 1$ inputs, each subset with sum of sizes at most size $q - w_i$. Such an assignment results in at least $\left\lceil \frac{s - w_i}{q - w_i} \right\rceil$ subsets of the $m - 1$ inputs. Thus, the input i is required to be sent to at least $\left\lceil \frac{s - w_i}{q - w_i} \right\rceil$ reducers to be paired with all the remaining $m - 1$ inputs. $\square$

Theorem 4 (The total communication cost and number of reducers) *For a set of m inputs and a given reducer capacity q, the total communication cost and the total number of reducers, for the A2A mapping schema problem, are at least $\frac{s^2}{q}$ and $\frac{s^2}{q^2}$, respectively, where s is the sum of all the input sizes.*

PROOF. Since an input i is replicated to at least $\left\lceil \frac{s-w_i}{q-w_i} \right\rceil$ reducers, the communication cost for the input i is $w_i \times \left\lceil \frac{s-w_i}{q-w_i} \right\rceil$. Hence, the total communication cost for all the inputs will be at least $\sum_{i=1}^{m} w_i \frac{s-w_i}{q-w_i}$. Since $s \geq q$, we can conclude $\frac{s-w_i}{q-w_i} \geq \frac{s}{q}$. Thus, the total communication cost is at least $\sum_{i=1}^{m} w_i \frac{s}{q} = \frac{s^2}{q}$.

Since the total communication cost, the total number of bits to be assigned to reducers, is at least $\frac{s^2}{q}$, and a reducer can hold inputs whose sum of the sizes is at most q, the total number of reducers must be at least $\frac{s^2}{q^2}$. $\square$

Bin-packing-based Heuristic. First-Fit Decreasing (FFD) and Best-Fit Decreasing (BFD) [5] are most notable bin-packing algorithms. We use the FFD or BFD bin-packing algorithm to place the given m inputs to bins of size $\frac{q}{2}$. Assume that FFD or BFD bin-packing algorithm needs x bins to place m inputs, and now, each of such bins is considered as a single input of size $\frac{q}{2}$. Since the reducer capacity is q, any two bins can be assigned to a single reducer. Hence, the heuristic uses at most $r(m,q) = \frac{x(x-1)}{2}$ reducers; see Figure 5. There also exists a pseudo polynomial bin-packing algorithm, suggested by Karger and Scott [9], that can place the m inputs in as few bins as possible of size $\frac{q}{2}$.

Total required reducers. FFD and BFD bin-packing algorithms provide an $\frac{11}{9} \cdot$ OPT approximation ratio [8], *i.e.*, if any optimum bin-packing algorithm needs OPT bins to place (m) inputs in the bins of a given size

$w_1 = w_2 = w_3 = 0.20q, w_4 = w_5 = 0.19q,$
$w_6 = w_7 = 0.18q$

Four bins, each of size $\frac{q}{2}$

Six reducers

Figure 5: Bin-packing-based heuristic.

(say, $\frac{q}{2}$), then FFD and BFD bin-packing algorithms always use at most $\frac{11}{9} \cdot$ OPT bins of the same size (to place the given m inputs). Since we require at most $\frac{x(x-1)}{2}$ reducers for a solution to the *A2A mapping schema problem*, the heuristic requires at most $r(m,q) = \frac{(\frac{11}{9} \cdot OPT)^2}{2}$ reducers.

Note that, here in this case, OPT does not indicate the optimum number of reducers to assign m inputs that satisfy the *A2A mapping schema problem*; OPT indicates the optimum number of bins of size $\frac{q}{2}$ that are required to place m inputs.

The following theorem gives the upper bounds that this heuristic achieves on the replication of an inputs, the total communication cost and the number of reducers.

Theorem 5 (Upper bounds from the heuristic) *The above heuristic using a bin size $b = \frac{q}{2}$ where q is the reducer capacity achieves the following three upper bounds: The total number of reducers, the replication of individual inputs, and the total communication cost, for the A2A mapping schema problem, are at most $\frac{8s^2}{q^2}$, at most $4\frac{s}{q}$, and at most $4\frac{s^2}{q}$, respectively, where s is the sum of all the input sizes.*

PROOF. A bin i can hold inputs whose sum of the sizes is at most b. Since the total sum of the sizes is s, it is required to divide the inputs into at least $\frac{s}{b}$ bins. Since the FFD or BFD bin-packing algorithm ensures that all the bins (except only one bin) are at least half-full, each bin of size $\frac{q}{2}$ has at least inputs whose sum of the sizes is at least $\frac{q}{4}$. Thus, all the inputs can be placed in at most
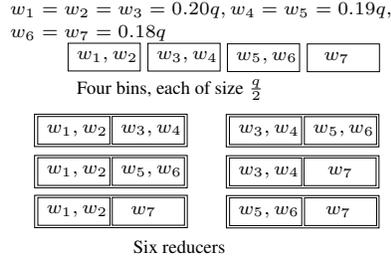
$w_1 = 0.20q, w_2 = 0.20q, w_3 = 0.20q, w_4 = 0.19q, w_5 = 0.19q,$
$w_6 = 0.18q, w_7 = 0.18q$

Four bins, each of size $\frac{q}{2}$

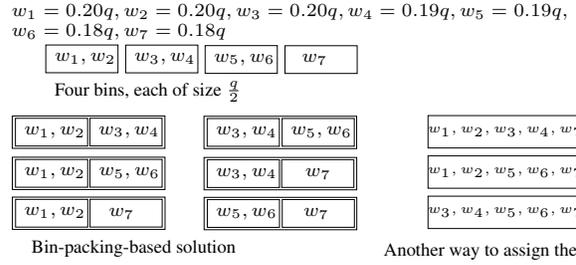Bin-packing-based solution

Another way to assign the inputs

Figure 7: Comparison between the bin-packing-based heuristic and the proposed algorithms for almost equal-sized inputs.

$\frac{s}{q/4}$ bins of size $\frac{q}{2}$. Since each bin is considered as a single input, we can assign every two bins at a reducer, and hence, we require at most $\frac{8s^2}{q^2}$ reducers. Since each bin is replicated to at most $4\frac{s}{q}$ reducers, the replication of individual inputs is at most $4\frac{s}{q}$ and the total communication cost is at most $\sum_{1 \leq i \leq m} w_i \times 4\frac{s}{q} = 4\frac{s^2}{q}$. $\square$

4.1.2 Almost equal-sized inputs

In this section, we provide two algorithms for m almost equal-sized ($\frac{q}{k}$, where $k > 1$) inputs to assign each pair of inputs to reducers of capacity q. In other word, we are given a lot of inputs of very small sizes as compared to q. We pack all these inputs to bins of unit size, and then consider each bin as a single input of unit-size. Equivalently, we can take the reducer capacity to be q and the inputs to be of unit size. In what follows, we will continue to use q as the reducer capacity and assume all inputs are of unit size.

The two algorithms can be summarized as follows: *2-step algorithms* (Algorithm 1 and Algorithm 2) handle the case of m unit-sized inputs and odd-even values of the reducer capacity q. Algorithms 1 and 2 assume that q is an odd or an even number, respectively.

Aside. Algorithms 1 and 2 have an advantage over the bin-packing-based heuristic (Section 4.1.1). When inputs of almost identical sizes are given, the bin-packing-based heuristic uses more reducers as compared to Algorithms 1 and 2. For example, we are given a set of seven inputs $I = \{i_1, i_2, \ldots, i_7\}$ whose size set is $W = \{0.20q, 0.20q, 0.20q, 0.19q, 0.19q, 0.18q, 0.18q\}$. In this case, the bin-packing-based heuristic uses at least six reducers while we can assign them to three reducers, see Figure 7.

Our goal to use Algorithms 1, 2, 3, and 4 is to minimize the communication cost between the map and reduce phases for a given number of unit-sized inputs and the reducer capacity q. Before going into details of algorithms for $q > 2$, we look at the lower bound on the total communication cost between the map and reduce phases. The case of m inputs of size one and reducers of capacity two is trivial. In this case, we can assign every pair of inputs to a single reducer, which results in $r(m,q) = \frac{m(m-1)}{2}$ reducers, and moreover, it is clearly impossible to use fewer reducers.

Theorem 6 (Replication of individual inputs) *For a given reducer capacity $q > 1$ and a set of m inputs of size one, an input i required to be sent to at least $\left\lceil \frac{m-1}{q-1} \right\rceil$ reducers for a solution to the A2A mapping schema problem.*
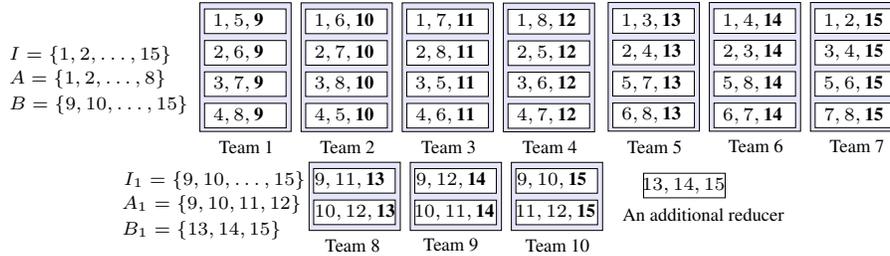
PROOF. Consider an input i. The input i can share a reducer with only $q - 1$ inputs. In order to assign the input i with all the remaining $m - 1$ inputs, it is required to create disjoint subsets of the remaining $m - 1$ inputs such that each subset can hold at most $q - 1$ inputs. In this manner, there are at least $\left\lceil \frac{m-1}{q-1} \right\rceil$ subsets of

Cases	Solutions	Sections	Theorems	Replication of individual inputs	Reducers	Communication cost
The **lower bounds** for the *A2A mapping schema problem*						
Different-sized inputs		4.1.1	3 and 4	$\frac{s}{q}$	$\frac{s^2}{q^2}$	$\frac{s^2}{q}$
Almost equal-sized inputs		4.1.2	6 and 7	$\left\lceil \frac{m-1}{q-1} \right\rceil$	$\left\lfloor \frac{m}{q} \right\rfloor \left\lceil \frac{m-1}{q-1} \right\rceil$	$m \left\lceil \frac{m-1}{q-1} \right\rceil$
The **lower bounds** for the *X2Y mapping schema problem*						
Different-sized inputs		5	13 and 14	$\frac{sum_x}{q}, \frac{sum_y}{q}$	$\frac{2 \cdot sum_x \cdot sum_y}{q^2}$	$\frac{2 \cdot sum_x \cdot sum_y}{q}$
The **upper bounds** for the *A2A mapping schema problem*						
Different-sized inputs	Bin-packing-based heuristic	4.1.1	5	$\frac{4s}{q}$	$\frac{8s^2}{q^2}$	$\frac{4s^2}{q}$
Almost equal-sized inputs	Algorithm 1	4.1.2	9	$\left\lceil \frac{2m}{q-1} \right\rceil - 1$	$(\left\lceil \frac{2m}{(q-1)} \right\rceil)^2/2$	$m\left(\left\lceil \frac{2m}{(q-1)} \right\rceil - 1\right)$
	Algorithm 2	4.1.2	11	$\left\lceil \frac{2m}{q} \right\rceil - 1$	$(\left\lceil \frac{2m}{q} \right\rceil)^2/2$	$m\left(\left\lceil \frac{2m}{q} \right\rceil - 1\right)$
An input of size $> \frac{q}{2}$	Bin-packing-based heuristic	4.2	12	$m-1$	$m-1+\frac{8s^2}{q^2}$	$(m-1) \cdot q + \frac{4s^2}{q}$
The **upper bounds** for the *X2Y mapping schema problem*						
Different-sized inputs, $q=2b$	Bin-packing-based heuristic	5	15	$\frac{2 \cdot sum_x}{b}, \frac{2 \cdot sum_y}{b}$	$\frac{4 \cdot sum_x \cdot sum_y}{b^2}$	$\frac{4 \cdot sum_x \cdot sum_y}{b}$

Table 1: The bounds for heuristics for the *A2A* and the *X2Y* mapping schema problems.

Algorithm 1, with $q = 3$, divides m unit-sized inputs into two disjoint sets A and B of y and $x \leq y - 1$ inputs respectively. (The selection of the value of y will be described later. But, note that we prefer y to be a power of 2, and if $y \neq 2^i$ and $y > 4$, $i > 2$, then we add unit-sized dummy inputs so that y is a power of 2.) When $q = 3$, we organize $(y-1) \times \left\lceil \frac{y}{2} \right\rceil$ reducers (each of capacity three) in the form of $y-1$ *teams* of $\left\lceil \frac{y}{2} \right\rceil$ *players* (or reducers) in each team, and these reducers are used to assign each input of the set A with all the remaining inputs of the sets A and B. Note that a team must have each of the inputs of the set A occurring exactly once among the reducers of that team, and this fact will be clear soon.

There are $(y-1) \times \left\lceil \frac{y}{2} \right\rceil$ pairs of inputs of the set A (each of size two) and there are the same number of reducers (each of capacity three); hence, it is possible to assign one pair to each reducer, and these two inputs become two of the three inputs allowed to each reducer. Once, we assign every pair of inputs of the set A to $(y-1) \times \left\lceil \frac{y}{2} \right\rceil$ reducers, then we assign i^{th} input of the set B to all the $\left\lceil \frac{y}{2} \right\rceil$ reducers of ith team. Further, we follow a similar procedure on inputs of the set B to assign each pair of the remaining x inputs.

$I = \{1, 2, \ldots, 15\}$
$A = \{1, 2, \ldots, 8\}$
$B = \{9, 10, \ldots, 15\}$

Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7
1, 5, **9**	1, 6, **10**	1, 7, **11**	1, 8, **12**	1, 3, **13**	1, 4, **14**	1, 2, **15**
2, 6, **9**	2, 7, **10**	2, 8, **11**	2, 5, **12**	2, 4, **13**	2, 3, **14**	3, 4, **15**
3, 7, **9**	3, 8, **10**	3, 5, **11**	3, 6, **12**	5, 7, **13**	5, 8, **14**	5, 6, **15**
4, 8, **9**	4, 5, **10**	4, 6, **11**	4, 7, **12**	6, 8, **13**	6, 7, **14**	7, 8, **15**

$I_1 = \{9, 10, \ldots, 15\}$
$A_1 = \{9, 10, 11, 12\}$
$B_1 = \{13, 14, 15\}$

Team 8	Team 9	Team 10	
9, 11, **13**	9, 12, **14**	9, 10, **15**	13, 14, 15
10, 12, **13**	10, 11, **14**	11, 12, **15**	An additional reducer

Example. We are given 15 inputs ($I = \{1, 2, \ldots, 15\}$) of size one and $q = 3$. We create two sets, namely A of $y = 8$ inputs and B of $x = 7$ inputs, and arrange $(y-1) \times \left\lceil \frac{y}{2} \right\rceil = 28$ reducers in the form of 7 *teams* of 4 *players* (or reducers) in each team. These 7 teams assign each input of the set A with all the remaining inputs of the set A and the set B. We pair every two inputs of the set A and assign them to exactly one of 28 reducers. (All these pairs of the inputs of the set A are created and assigned using lines 10, 12, and 13 of Algorithm 1.) Once every pair of $y = 8$ inputs of the set A is assigned to exactly one of 28 reducers, then we assign i^{th} input of the set B to all the four reducers of ith team, see Team 1 to Team 7. Of course, the third input in i^{th} team is i^{th} input of the set B.

Now note that the first four teams pair inputs 1-4 with inputs 5-8. The first team (Team 1) has pairs $\{1, 5\}$, $\{2, 6\}$, $\{3, 7\}$, and $\{4, 8\}$. Team 2-4 has pairs by rotation of the 5-8 inputs. Teams 5 and 6 handle pairs of 1-2 with 3-4 and 5-6 with 7-8, respectively, in the same way, and the last team has pairs $\{1, 2\}$, $\{3, 4\}$, ..., $\{7, 8\}$.

Next, we implement the same procedure on 7 inputs of the set B. We create two sets, say $A_1 = \{9, 10, 11, 12\}$ of $y_1 = 4$ inputs and $B_1 = \{13, 14, 15\}$ of $x_1 = 3$. Then, we arrange $(y_1 - 1) \times \left\lceil \frac{y_1}{2} \right\rceil = 6$ reducers in the form of 3 *teams* of 2 reducers in each team. We assign each pair of inputs of the set A_1 to these 6 reducers, and then i^{th} input of the set B_1 to all the two reducers of a team, see Team 8 to Team 10. Further, we assign the remaining inputs of the set B_1 to a single reducer. The assignment of inputs to Teams 8-10 follows the same procedure as we did for Teams 1-7.

We have three claims, as follows: (*i*) each input of the set A appears exactly once in each team, (*ii*) the set B holds $x \leq y - 1$ inputs when $q = 3$, and (*iii*) the given algorithm assigns each pair of inputs to at least one reducer. We will prove these claims in algorithm correctness.

Figure 6: 2-step algorithm (Algorithm 1) for the reducer capacity $q = 3$ and $m = 15$.

$m - 1$ inputs. Hence, the input i is required to be sent to at least $\lceil \frac{m-1}{q-1} \rceil$ reducers. $\square$

Theorem 7 (The total communication cost and number of reducers) *For a given reducer capacity $q > 1$ and a set of m inputs of size one, the total communication cost and the total number of reducers, for the A2A mapping schema problem, are at least $m \lceil \frac{m-1}{q-1} \rceil$ and at least $\lfloor \frac{m}{q} \rfloor \lceil \frac{m-1}{q-1} \rceil$, respectively.*

PROOF. Since an input i is required to be sent to at least $\lceil \frac{m-1}{q-1} \rceil$ reducers, the sum of the number of copies of (m) inputs sent to reducers is at least $m \lceil \frac{m-1}{q-1} \rceil$, which result in at least $m \lceil \frac{m-1}{q-1} \rceil$ communication cost.

There are at least $m \lceil \frac{m-1}{q-1} \rceil$ total number of copies of (m) inputs to be sent to reducers and a reducer can hold at most q inputs; hence, at least $\lfloor \frac{m}{q} \rfloor \lceil \frac{m-1}{q-1} \rceil$ reducers are required. $\square$

Algorithm 1: 2-step algorithm when the reducer capacity q is an odd number. For the sake of understanding and presentation, we first present two examples, where $q = 3$, *i.e*, a reducer can hold at most three unit-sized inputs; see Figure 6 (and $q = 5$, *i.e.*, a reducer can hold at most five unit-sized inputs; see Figure 11 in [1]).

Following the example given for $q = 3$ (in Figure 6), we present our algorithm (see Algorithm 1) that handles any odd value of q. The algorithm consists of five steps as follows:
1. Divide m inputs into two sets A and B of size $y = \lfloor \frac{q}{2} \rfloor (\lfloor \frac{2m}{q+1} \rfloor + 1)$ and $x = m - y$, respectively.
2. Group the y inputs into $u = \lceil \frac{y}{q - \lceil q/2 \rceil} \rceil$ disjoint groups, where each group holds $\lceil \frac{q-1}{2} \rceil$ inputs. (We consider each of the u ($= \lceil \frac{y}{q - \lceil q/2 \rceil} \rceil$) disjoint groups as a single input that we call the *derived input*. By making u disjoint groups[1] (or derived inputs) of y inputs of the set A, we turn the case of any odd value of q to a case where a reducer can hold only three inputs, the first two inputs are pairs of the derived inputs and the third input is from the set B.)
3. Organize $(u - 1) \times \lceil \frac{u}{2} \rceil$ reducers, each of capacity q, in the form of $u - 1$ teams of $\lceil \frac{u}{2} \rceil$ reducers in each team. Assign every two groups to one of $(u - 1) \times \lceil \frac{u}{2} \rceil$ reducers. To do so, we will prove the following Lemma 1, and its proof is omitted from here due to space and given in [1].

Lemma 1 *Each pair of $u = 2^i$, $i > 0$, inputs can be assigned to $2^i - 1$ teams of 2^{i-1} reducers in each team, where the reducer capacity is q and the size of each input is $\lceil \frac{q-1}{2} \rceil$.*

4. Once every pair of the derived inputs are assigned, then assign i^{th} input of the set B to all the reducers of ith team.
5. Apply (the above mentioned) steps 1-4 on the set B until there is a solution to the *A2A mapping schema problem* for the x inputs.

Algorithm description. Algorithm 1 provides a solution to the *A2A mapping schema problem* for unit-sized inputs when q is an odd number. First, we divide m inputs into two sets A and B. Then, we make $u = \lceil \frac{y}{q - \lceil q/2 \rceil} \rceil$ disjoint groups of y inputs of the set A such that each group holds $\frac{q-1}{2}$ inputs, lines 1, 2. (Now, each of the groups is considered as a single input that we call the *derived*

---
[1]We suppose that u is a power of 2. In case u is not a power of 2 and $u > q$, we add dummy inputs each of size $\lceil \frac{q-1}{2} \rceil$ so that u becomes a power of 2. Consider that we require d dummy inputs. If groups of inputs of the set B each of size $\lceil \frac{q-1}{2} \rceil$ are less than equal to d dummy inputs, then we use inputs of the set B in place of dummy inputs, and the set B will be empty.

input.) We do not show the addition of dummy inputs and assume that u is a power of 2. Function $2\_step\_odd\_q(lower, upper)$ recursively divides the derived inputs into two halves, line 4. Function $Assignment(lower, mid, upper)$ (line 8) pairs every two derived inputs and assigns them to the respective reducers (line 10). Each reducer of the last team is assigned using function $Last\_Team(groupA[])$, lines 14, 15.

Note that functions $2\_step\_odd\_q(lower, upper)$, $Assignment(lower, mid, upper)$, and $value\_b(lower, t, mid, upper)$ take two common parameters, namely $lower$ and $upper$ where $lower$ is the first derived input and $upper$ is the last derived input (*i.e.*, u^{th} group) at the time of the first call to functions, line 3. Once all-pairs of the derived inputs are assigned to reducers, line 10, function $Assign\_input\_from\_B(Team[])$ assigns i^{th} input of the set B to all the $\lceil \frac{u}{2} \rceil$ reducers of ith team, lines 16, 17. After that, algorithm is invoked over inputs of the set B to assign each pair of the remaining inputs of the set B to reducers until every pair to the remaining inputs is assigned to reducers.

Algorithm correctness. The algorithm correctness proves that every pair of inputs is assigned to reducers. Specifically, we prove that all those pairs of inputs, $\langle i, j \rangle$ and $\langle i', j' \rangle$, of the set A are assigned to a team whose $i \neq i'$ and $j \neq j'$ (Claim 1). Then that all the inputs of the set A appear exactly once in each team (Claim 2). We then prove that the set B holds $x \leq y - 1$ inputs, when $q = 3$ (Claim 3). At last we conclude in Theorem 8 that Algorithm 1 assigns each pair of inputs to reducers.

Note that we are proving all the above mentioned claims for $q = 3$; the cases for $q > 3$ can be generalized trivially where we make $u = \lceil \frac{y}{q - \lceil q/2 \rceil} \rceil$ derived inputs from y inputs of the set A (and assign in a manner that all the inputs of the A are paired with all the remaining $m - 1$ inputs).

Claim 1 *Pairs of inputs $\langle i, j \rangle$ and $\langle i', j' \rangle$, where $i \neq i'$ or $j \neq j'$, of the set A are assigned to a team.*

PROOF. First, consider $i = i'$ and $j \neq j'$, where $\langle i, j \rangle$ and $\langle i', j' \rangle$ must be assigned to two different teams. If $j \neq j'$, then both the j values may have an identical value of $lower$ and mid but they must have two different values of t (see lines 12, 13 of Algorithm 1), where $j = lower + t + mid$ or $j = lower + t$. Thus, for two different values of j, we use two different values of t, say t_1 and t_2, that results in an assignment of $\langle i, j \rangle$ and $\langle i', j' \rangle$ to two different teams t_1 and t_2, (note that teams are also selected based on the value of t, $(y - 2 \cdot mid + 1) + t$, see line 10 of Algorithm 1, where for $q = 3$, we have $u = y$). Suppose now that $i \neq i'$ and $j = j'$, where $\langle i, j \rangle$ and $\langle i', j' \rangle$ must be assigned to two different teams. In this case, we also have two different values of t, and hence, two different t values assign $\langle i, j \rangle$ and $\langle i', j' \rangle$ to two different teams $((y - 2 \cdot mid + 1) + t$, line 10 of Algorithm 1).

Hence, it is clear that pairs $\langle i, j \rangle$ and $\langle i', j' \rangle$, where $i \neq i'$ and $j \neq j'$, are assigned to a team. $\square$

Claim 2 *All the inputs of the set A appear exactly once in each team.*

PROOF. There are the same number of pairs of inputs of the set A and the total number of reducers ($(y - 1) \lceil \frac{y}{2} \rceil$) that can provide a solution to the *A2A mapping schema problem* for the y inputs of the set A. Note that if there is a input pair $\langle i, j \rangle$ in team t, then the team t cannot hold any pair that has either i or j in the remaining $\lceil \frac{y}{2} \rceil - 1$ reducers. For the given y inputs of the set A, there are at most $\lceil \frac{y}{2} \rceil$ disjoint pairs $\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle, \ldots, \langle i_{\lceil y/2 \rceil}, j_{\lceil y/2 \rceil} \rangle$ such that $i_1 \neq i_2 \neq \ldots \neq i_{\lceil y/2 \rceil} \neq j_1 \neq j_2 \neq \ldots \neq j_{\lceil y/2 \rceil}$. Hence,

---

Algorithm 1: 2-step algorithm for an odd value of q.

Inputs:

m: total number of unit-sized inputs

q: the reducer capacity.

Variables:

A: A set A, where the total inputs in the set A is $y = \lfloor \frac{q}{2} \rfloor (\lfloor \frac{2m}{q+1} \rfloor + 1)$.

B: A set B, where the total inputs in the B is $x = m - (\lceil \frac{y}{q - \lceil q/2 \rceil} \rceil - 1)$.

$Team[i, j]$: represents teams of reducers, where index i indicates i^{th} team and index j indicates j^{th} reducer in i^{th} team. Consider $u = \lceil \frac{y}{q - \lceil q/2 \rceil} \rceil$. There are $u - 1$ teams of $v = \lceil \frac{u}{2} \rceil$ reducers in each team.

$groupA[]$: represents disjoint groups of inputs of the set A, where $groupA[i]$ indicates i^{th} group of $\lceil \frac{q-1}{2} \rceil$ inputs of the A.

1 **Function** $create\_group(y)$ **begin**
2 | **for** $i \leftarrow 1$ **to** u **do** $groupA[i] \leftarrow \langle i, i+1\ldots, i + \frac{q-1}{2} - 1 \rangle, i \leftarrow i + \frac{q-1}{2}$;
3 | $2\_step\_odd\_q(1, u), Last\_Team(groupA[]), Assign\_input\_from\_B(Team[])$

4 **Function** $2\_step\_odd\_q(lower, upper)$ **begin**
5 | **if** $\lfloor \frac{upper - lower}{2} \rfloor < 1$ **then return**;
6 | **else**
7 | | $mid \leftarrow \lceil \frac{upper - lower}{2} \rceil, Assignment(lower, mid, upper), 2\_step\_odd\_q(lower, mid), 2\_step\_odd\_q(mid+1, upper)$

8 **Function** $Assignment(lower, mid, upper)$ **begin**
9 | **while** $mid > 1$ **do**
10 | | **foreach** $(a, t) \in [lower, lower + mid - 1] \times [0, mid - 1]$ **do**
       | | $Team\big[(u - 2 \cdot mid + 1) + t, a - \lfloor \frac{a-1}{mid} \rfloor \cdot \frac{mid}{2} \big] \leftarrow \langle groupA[a], groupA[value\_b(a, t, mid, upper)] \rangle$;

11 **Function** $value\_b(a, t, mid, upper)$ **begin**
12 | **if** $a + t + mid < upper + 1$ **then return** $(a + t + mid)$;
13 | **else if** $a + t + mid > upper$ **then return** $(a + t)$;

14 **Function** $Last\_Team(lower, mid, upper)$ **begin**
15 | **foreach** $i \in [1, v]$ **do** $Team[u - 1, i] \leftarrow groupA[2 \times i - 1], groupA[2 \times i]$;

16 **Function** $Assign\_input\_from\_B(Team[])$ **begin**
17 | **foreach** $(i, j) \in [1, u - 1] \times [1, v]$ **do** $Team[i, j] \leftarrow B[i]$;

---

all y inputs of the set A are assigned to a team, where no input is assigned twice in a team. □

Claim 3 *When the reducer capacity $q = 3$, the set B holds at most $x \leq y - 1$ inputs.*

PROOF. Since a pair of inputs of the set A requires at most $q - 1$ capacity of a reducer and each team holds all the inputs of the set A, an input from the set B can be assigned to all the reducers of the team. In this manner, all the inputs of the set A are also paired with an input of the set B. Since there are $y - 1$ teams and each team is assigned an input of the set B, the set B can hold at most $x \leq y - 1$ inputs. □

Theorem 8 *Algorithm 1 assigns each pair of the given m inputs to reducers.*

PROOF. We have $(y - 1) \lceil \frac{y}{2} \rceil$ pairs of inputs of the set A of size $q - 1$, and there are the same number of reducers; hence, each reducer can hold one input pair. Further, the remaining capacity of all the reducers of each team can be used to assign an input of B. Hence, all the inputs of A are paired with every other input and every input of B (as we proved in Claims 2 and 3). Following the fact that the inputs of the set A are paired with all the m inputs, the inputs of the set B is also paired by following a similar procedure on them. Thus, Algorithm 1 assigns each pair of the given m inputs to reducers. □

Theorem 9 *Algorithm 1 requires at most $(\lceil \frac{2m}{(q-1)} \rceil)^2 / 2$ reducers and results in at most $m(\lceil \frac{2m}{(q-1)} \rceil - 1)$ communication cost.*

PROOF. There are at most $x = \lceil \frac{2m}{q-1} \rceil$ groups (derived inputs) of the given m inputs. In order to assign each pair of the derived inputs, each derived input is required to assign to at most $x - 1$ reducers. This fact results in at most $m(\lceil \frac{2m}{(q-1)} \rceil - 1)$ communication cost, and there are at most $(\lceil \frac{2m}{(q-1)} \rceil)^2 / 2$ pairs of the derived inputs that require at most $(\lceil \frac{2m}{(q-1)} \rceil)^2 / 2$ reducers. □

Algorithm 2: 2-step algorithm when the reducer capacity q is an even number. We present our algorithm (see Algorithm 2) that handles any even value of q. For the sake of understanding and presentation, we first present an example where $q = 4$, namely the case in which a reducer can hold at most four unit-sized inputs, as demonstrated in Figure 8 (Figure 8 is self-explainable; however, interested readers may refer to Figure 12 in [1] for details). Note that unlike the algorithm for odd values of q (Algorithm 1) the algorithm for even values of q (Algorithm 2) does not divide the m inputs into two sets. The algorithm consists of two steps, as follows:

1. Group the given m inputs into $u = \lceil \frac{2m}{q} \rceil$ disjoint groups,

2. Organize $(u-1) \times \frac{u}{2}$ reducers, each of capacity q, in the form of $u - 1$ teams of $\frac{u}{2}$ reducers in each team. Assign every two groups to one of $(u-1) \times \frac{u}{2}$ reducers. We use Lemma 1 for the assignment of every two groups.

Note that we consider each of the u $(= \lceil \frac{2m}{q} \rceil)$ groups as a single input that we call the *derived input*. By making u disjoint groups of the m inputs, we turn the case of any even value of q to a case when $q = 2$ (*i.e.*, a reducer can hold only two unit-sized inputs) and assign every two derived inputs to reducers. In this
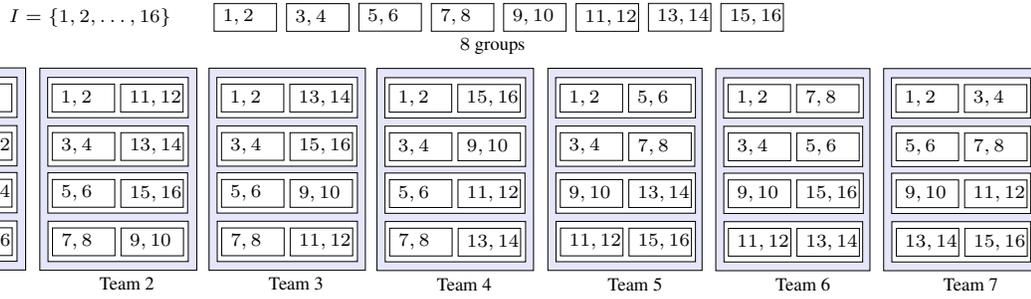
Figure 8: 2-step algorithm (Algorithm 2) when the reducer capacity $q = 4$.

Algorithm 2: 2-step algorithm for an even value of q.

Inputs: m: total number of unit-sized inputs.
q: the reducer capacity.
Variables: $Team[i, j]$: represents teams of reducers, where index i indicates i^{th} team and index j indicates j^{th} reducer in i^{th} team. Consider $u = \lceil \frac{2m}{q} \rceil$. There are $u - 1$ teams of $\lceil \frac{u}{2} \rceil$ reducers in each team.
$groupA[]$: represents disjoint groups of inputs of the set A, where $groupA[i]$ indicates i^{th} group of $\lceil \frac{q}{2} \rceil$ inputs of the set A.

1 **Function** $create\_group(m)$ **begin**
2   **for** $i \leftarrow 1$ **to** u **do**
      $groupA[i] \leftarrow \langle i, i+1 \ldots, i + \frac{q}{2} - 1 \rangle, i \leftarrow i + \frac{q}{2}$;
3     $2\_step\_even\_q(1, u), Last\_Team(1, \lceil \frac{u-1}{2} \rceil, u)$
4 **Function** $2\_step\_even\_q(lower, upper)$ **begin**
5   **if** $\lfloor \frac{upper-lower}{2} \rfloor < 1$ **then return**;
6   **else**
7       $mid \leftarrow \lceil \frac{upper-lower}{2} \rceil$,
      $Assignment(lower, mid, upper)$,
      $2\_step\_even\_q(lower, mid)$,
      $2\_step\_even\_q(mid + 1, upper)$,

---

manner, each input of the set A is assigned with all the remaining $m - 1$ inputs.

Algorithm description. Algorithm 2 provides a solution to the *A2A mapping schema problem* for unit-sized inputs when q is an even number. Recall that Algorithm 1 and Algorithm 2 are almost similar except Algorithm 2 does not create two sets A and B. We first make $u = \lceil \frac{2m}{q} \rceil$ disjoint groups of the given m inputs such that each group holds $\frac{q}{2}$ inputs (lines 2), (and consider each of the groups as a single input, the *derived input*). Function $2\_step\_even\_q(lower, upper)$ recursively divides the derived inputs into two halves, lines 4 and 7. Function $Assignment(lower, mid, upper)$ (line 7) is a similar function as given for Algorithm 1 (see line 8 of Algorithm 1) and makes pairs of every two derived inputs. Function $Last\_Team(group[])$ (lines 3) assigns inputs to the last team, *i.e.*, team $u - 1$. Note that function $Last\_Team(group[])$ is same as given for Algorithm 1 (see line 14 of Algorithm 1).

Algorithm correctness. We show that every pair of inputs is assigned to reducers. Specifically, Algorithm 2 satisfies two claims, as follows:

Claim 4 *Pairs of derived inputs $\langle i, j \rangle$ and $\langle i', j' \rangle$, where $i \neq i'$ or $j \neq j'$, are assigned to a team.*

Claim 5 *All the given m inputs appear exactly once in each team.*

Theorem 10 *Algorithm 2 assigns each pair of the given m inputs to reducers.*

Theorem 11 *Algorithm 2 requires at most $(\lceil \frac{2m}{q} \rceil)^2 / 2$ reducers and results in at most $m(\lceil \frac{2m}{q} \rceil - 1)$ communication cost.*

Claim 4, Claim 5, Theorems 10, and 11 can be proved in a similar manner as Claim 1, Claim 2, Theorems 8, and 9, respectively. Detailed proofs are given in [1].

4.2 A big input of size greater than $\frac{q}{2}$

We now consider the case of an input of size w_i, $\frac{q}{2} < w_i < q$; we call such an input as a *big input*. Note that if there are two big inputs, then they cannot be assigned to a single reducer, and hence, there is no solution to the *A2A mapping schema problem*. We assume m inputs of different sizes are given. There is a big input and all the remaining $m - 1$ inputs, which we call the *small inputs*, have at most size $q - w_i$.

We use FFD or BFD bin-packing algorithm to place the small inputs to bins of size $q - w_i$. Now, we consider each of the bins as a single input of size $q - w_i$. Let x bins are used. We assign each of the x bins to one reducer with a copy of the big input. Further, we assign the small inputs to bins of size $\frac{q}{2}$, and consider each of such bins as a single input of size $\frac{q}{2}$. Now, we can assign each pair of bins (each of size $\frac{q}{2}$) to reducers. In this manner, each pair of inputs is assigned to reducers.

Theorem 12 (Upper bounds from the heuristic) *For a set of m inputs where a big input, i, of size $\frac{q}{2} < w_i < q$ and for a given reducer capacity q, $q < s' < s$, an input is replicated to at most $m - 1$ reducers for the A2A mapping schema problem, and the total number of reducers and the total communication cost are at most $m - 1 + \frac{8s'^2}{q^2}$ and $(m-1)q + \frac{4s^2}{q}$, respectively, where s' is the sum of all the input sizes except the size of the big input and s is the sum of all the input sizes.*

PROOF. The big input i can share a reducer with inputs whose sum of the sizes is at most $q - w_i$. In order to assign the input i with all the remaining $m - 1$ small inputs, it is required to assign a subset of $m - 1$ inputs whose sum of the sizes is at most $q - w_i$. If all the small inputs are of size almost $q - w_i$, then a reducer can hold the big input and one of the small inputs. Hence, the big input is required to be sent to at most $m - 1$ reducers that results in at most $(m - 1)q$ communication cost.

Also, each pair of all the small inputs is assigned to reducers (by first placing them to bins of size $\frac{q}{2}$ using FFD or BFD bin-packing algorithm). The assignment of all the small inputs results in at most $\frac{8s'^2}{q^2} < \frac{8s^2}{q^2}$ reducers and at most $\frac{4s'^2}{q} < \frac{4s^2}{q}$ communication cost (Theorem 5). Thus, the total number of

reducers are at most $m - 1 + \frac{8s^2}{q^2}$ and the total communication cost is at most $(m-1)q + \frac{4s^2}{q}$. $\quad\square$

5. A HEURISTIC FOR THE X2Y MAPPING SCHEMA PROBLEM

We propose a heuristic for the *X2Y mapping schema problem* that is based on bin-packing algorithms. The proposed heuristic assumes a fixed reducer capacity q. Two sets, X of m inputs and Y of n inputs, are given. We assume that the sum of input sizes of the sets X, denoted by sum_x, and Y, denoted by sum_y, is greater than q. We analyze the heuristic on criteria given in Section 4. We look at the lower bounds in Theorems 13 and 14, and Theorem 15 gives an upper bound from a heuristic. The bounds are given in Table 1.

Theorem 13 (Replication of individual inputs) *For a set X of m inputs, a set Y of n inputs, and a given reducer capacity q, an input i of the set X is required to be sent to at least $\frac{sum_y}{q}$ reducers and an input j of the set Y is required to be sent to at least $\frac{sum_x}{q}$ reducers for a solution to the X2Y mapping schema problem.*

Theorem 14 (The total communication cost and number of reducers) *For a set X of m inputs, a set Y of n inputs, and a given reducer capacity q, the total communication cost and the total number of reducers, for the X2Y mapping schema problem, are at least $\frac{2 \cdot sum_x \cdot sum_y}{q}$ and $\frac{2 \cdot sum_x \cdot sum_y}{q^2}$, respectively.*

Bin-packing-based heuristic for the *X2Y mapping schema problem*. A solution to the *X2Y mapping schema problem* for different-sized inputs can be achieved using bin-packing algorithms. Let a fixed reducer capacity q, two sets X of m inputs, and Y of n inputs are given. The heuristic will not work when a set holds an input of size w_i and the another set holds an input of size greater than $q - w_i$, because these inputs cannot be assigned to a single reducer in common. Let the size of the largest input, i, of the set X is w_i; hence, all the inputs of the set Y have at most size $q - w_i$. We place inputs of the set X to bins of size w_i, and let x bins are used to place m inputs. Also, we place inputs of the set Y to bins of size $q - w_i$, and let y bins are used to place n inputs. Now, we consider each of the bins as a single input, and a solution to the *X2Y mapping schema problem* is obtained by assigning each of the x bins with each of the y bins to reducers. In this manner, we require $x \cdot y$ reducers.

Theorem 15 (Upper bounds from the heuristic) *For a bin size b, a given reducer capacity $q = 2b$, and with each input of sets X and Y being of size at most b, the total number of reducers, the replication of an individual input of the set X (resp. Y), and the total communication cost, for the X2Y mapping schema problem, are at most $\frac{4 \cdot sum_x \cdot sum_y}{b^2}$, at most $\frac{2 \cdot sum_y}{b}$ (resp. at most $\frac{2 \cdot sum_x}{b}$), and at most $\frac{4 \cdot sum_x \cdot sum_y}{b}$, respectively.*

Proofs of Theorems 13, 14, and 15 are given in [1].

6. CONCLUSION

Two new important practical aspects in the context of MapReduce, namely different-sized inputs and the reducer capacity, are introduced for the first time. The capacity of a reducer is defined in terms of the reducer's memory size. We note that processing time is typically proportional to the memory capacity. All reducers have an identical capacity, and any reducer cannot hold inputs whose input sizes are more than the reducer capacity. We demonstrated the importance of the capacity aspect by considering two common mapping schema problems of MapReduce, *A2A mapping schema problem* – every two inputs are required to be assigned to at least one common reducer – *X2Y mapping schema problem* – every two inputs, the first input from a set X and the second input from a set Y – is required to be assigned to at least one common reducer. Unfortunately, it turned out that finding solutions to the *A2A* and the *X2Y* mapping schema problems that use the minimum number of reducers is not possible in polynomial time. On the positive side, we present near optimal heuristics for the *A2A* and the *X2Y* mapping schema problems.

7. REFERENCES

[1] F. Afrati, S. Dolev, E. Korach, S. Sharma, and J. D. Ullman. Assignment problems of different-sized inputs in MapReduce. Technical Report 14-05, Department of Computer Science, Ben-Gurion University of the Negev, 2014. Also appears as a Brief Announcement in International Symposium on Distributed Computing (DISC) 2014.

[2] F. N. Afrati, A. D. Sarma, S. Salihoglu, and J. D. Ullman. Upper and lower bounds on the cost of a map-reduce computation. *PVLDB*, 6(4):277–288, 2013.

[3] F. N. Afrati and J. D. Ullman. Matching bounds for the all-pairs MapReduce problem. In *IDEAS*, pages 3–4, 2013.

[4] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, pages 131–140, 2007.

[5] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for NP-hard problems. chapter Approximation algorithms for bin packing: a survey, pages 46–93. PWS Publishing Co., 1997.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[7] M. T. Goodrich. Simulating parallel algorithms in the MapReduce framework with applications to parallel computational geometry. *CoRR*, abs/1004.4708, 2010.

[8] D. S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[9] D. R. Karger and J. Scott. Efficient algorithms for fixed-precision instances of bin packing and euclidean tsp. In *APPROX-RANDOM*, pages 104–117, 2008.

[10] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for MapReduce. In *SODA*, pages 938–948, 2010.

[11] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[12] A. Pietracaprina, G. Pucci, M. Riondato, F. Silvestri, and E. Upfal. Space-round tradeoffs for MapReduce computations. In *ICS*, pages 235–244, 2012.

[13] J. D. Ullman. Designing good MapReduce algorithms. *ACM Crossroads*, 19(1):30–34, 2012.

[14] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using MapReduce. In *SIGMOD Conference*, pages 495–506, 2010.

[15] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the 17th international conference on World Wide Web*, pages 131–140, 2008.