An Efficient Systematic Approach to Find All Cyclic Quorum Sets with All-pairs Property

Yiming Bian and Arun K. Somani
Department of Electrical and Computer Engineering
Iowa State University, Ames, Iowa 50010
{ybian, arun}@iastate.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671935

***Abstract*—The use of cyclic quorum sets has proved to be an efficient method for distributed systems to solve tasks requiring massive computations. In all-pairs data interaction problem, cyclic quorum sets can be used to avoid communication completely after initial data placement. However, searching for cyclic quorum sets with all-pairs property for a given number of objects, p, in a distributed system is a daunting task that involves massive computations as it is a combinatorial problem requiring full search. No time complexity reduction method has been found thus far. In other applications of cyclic quorum sets such as cycle-based routing problem, different cyclic quorum sets provide similar fault coverage yet generate significant resource utilization difference. Inspired by the need for all cyclic quorum sets with all-pairs property, we develop a methodology to optimize the search process by avoiding the search space where no new set could be found. After studying all possible cyclic quorum sets for a given p, we develop insights into the properties of all quorum sets that helps us to reduce the number of searches significantly. As a result, for small values of p, when doing brutal-force search, our method not only reduces the time to find the first cyclic quorum base set with all-pairs property but is also able to find all of them in a reasonable amount of time. Moreover, we notice that as p grows, up to several orders of magnitude of search reduction could be achieved compared to naïve search. For large values of p, with existing results of base sets, we also propose a heuristic method with constant time complexity that computes a feasible subsets-assigning solution that meets the all-pairs requirement.**

***Index Terms*—Searching for cyclic quorum sets with all-pairs property, Properties of cyclic quorum sets, Brute-force search optimization, Search space dimension reduction**

I. INTRODUCTION

A large number of applications use a big volume of data in the current wave of data-based decision-making processes. One of the most significant big data problems arises when many data elements need to interact with each other to derive meaningful conclusions. It is known as all-pairs data interaction problem and has many applications such as in data mining, the computation of similarity matrix is a critical step for clustering and classification. [1] It provides all pairwise similarities and dissimilarities between objects. [2] In physics and astronomy, n-body problem is commonly found and it consists of prediction the motions of a group of large, celestial objects interacting with one another. [3]

Since all-pairs interaction problems in the real world are often so large-scaled that it is almost impossible to solve them on a single machine. Distributed systems offer a promising solution to meet the challenge. When performing an all-pairs data interaction on the big data scale sizes, while the computational complexity theoretically is manageable, the data management is complex. [4] To achieve more efficiency, Driscoll et al. [5] proposed a communication-optimal n-body algorithm in 2013. Later in 2018, Kleinheksel et al. [4] applied cyclic quorum sets theory to manage data distribution. Their solutions adopt cyclic quorum sets, which solves data distribution and communication problem in a very elegant manner. After initial data placement, no further communication is needed to accomplish all-pairs computations with only partial data set stored at each node.

The idea of quorums was first introduced in Maekawa's algorithm [6]. Wai-Shing Luk et al. [7] identified one feasible cyclic quorum base set that has all-pairs property for distributed system with 4 to 111 nodes. Without an exception, all researchers mentioned that searching for those solutions cost a huge amount of time because the only approach is exhaustive search with the quorum size n starting from the theoretical lower bound [7].

To optimize this daunting search process, our idea is to reduce the overall searches by skipping certain space. In [8], authors show the significant difference when applying difference cyclic quorum sets to cycle-based routing problem. Therefore, we decide to take a step further and search for not only the first feasible base set, but find all of them.

The main contributions of this paper are as follows:

- To search for all interest sets that are feasible to generate cyclic quorum sets with all-pairs property, we develop a systematic approach that drastically decrease the computation time by avoiding certain search space.
- We explore and prove pairing properties of feasible base sets that once a feasible base set is found, there is another base set guaranteed to be feasible. We also define break point as the boundary of search. With pairing property and break point, the search space is reduced by almost 50% compared to the intermediate search space.
- We propose a novel constant time decomposition method to construct a group of quorums that guarantee to meet all-to-all interaction requirement for large value of p. It is based on existing base set results for $p = 4$ to 111.

The remainder of the paper is organized as follows. Section II provides prerequisite knowledge of cyclic quorum sets and

shows specific cyclic quorum sets have all-pairs property. Section III introduces base set search process optimization strategies with related definitions, propositions and proofs. Detailed results of computation number comparisons between naïve exhaustive search and our method and an example case of all feasible base set are provided in section IV. When p is a very large value, a constant time heuristic method that determines a group of quorums based on existing base set results to meet all-to-all interaction requirement is presented in section V. Finally, concluding remarks and potential benefits of our work are stated in section VI.

II. Cyclic Quorum Sets and All-pairs Property

In [7], authors define cyclic quorum sets as follows.

Definition 1 (Cyclic quorum sets)**.** A group of cyclic quorums is a group of sets $\{B_0, B_1, \ldots, B_{N-1}\}$, which satisfies the following properties:

B1. *i is contained in B_i, for all $i \in 0, 1, \ldots, N-1$;*

B2. *$B_i \cap B_j \neq \emptyset$, for all $i, j \in 0, 1, \ldots, N-1$;*

B3. *$B_i = \{a_1 + i, a_2 + i, \ldots, a_k + i\}$ modulo N.*

Here we define new notations in the scenario of an all-pairs interaction problem. To perform all-to-all interactions among elements in a whole data set D using a distributed system with p nodes, we first evenly divide D into p data subsets, namely $D_0, D_1, \ldots, D_{p-1}$. Then we assign n subsets to each node in the distributed system. These subsets form a quorum base set of size n, which is denoted by S_i (for $i = 0, 1, \ldots, p-1$). Applying cyclic quorum sets as data distribution strategy, when one of S_i is given, the rest can be developed in a cyclic manner, hence any of them is called a base set. For some special base sets, they guarantee that if each nodes computes the all interactions among assigned subsets, without any communication, the union of local results covers all pair-wise interactions among all data. This group of base set is said to have all-pairs property. In the case where $p = 7$, $n = 3$, two cyclic quorum sets are provided as shown in Fig. 1. Base sets on the left do not have all-pairs property because the interaction between data subset D_0 and D_3 is not present. On the other hand, base sets on the right have all-pairs property.

Specifically, we list all interactions among data subsets in Fig. 2 and mark each of them with the same color of the node that performs the computation. For example, a node performs all subset interactions in base set S_0. Thus D_0-D_1, D_0-D_3 and D_1-D_3 marked yellow. With this group of cyclic quorum sets, all possible interactions are performed. As will be discussed in Section III, this is a special case where no redundant work is done, thus every possible interaction is performed exactly once. In most cases, redundant computation is inevitable, but one could take advantage of it and enhance fault-tolerance of the system as introduced in [9]. In contrast to their applications, the search process for cyclic quorum sets with all-pairs property is the focus of this paper.

In the previous research, Kleinheksel et al. show the advantages of applying cyclic quorum sets to solve all-pairs problem including achieving load balancing among nodes, reducing memory resource requirements within a node and enabling big data scalability of the whole system. [4] We also identify the huge difference among cyclic quorum sets in a cycle-based routing problem and developed a conclusion that all feasible base sets should be provided to further explore the fault-tolerant and resource saving features of cyclic quorum set in [8]. These advantages and demand give us a motivation to optimize the search process and find all base sets that have all-pairs property for future research use.

$S_0 = \{D_0, D_1, D_2\}$	$S_0 = \{D_0, D_1, D_3\}$
$S_1 = \{D_1, D_2, D_3\}$	$S_1 = \{D_1, D_2, D_4\}$
$S_2 = \{D_2, D_3, D_4\}$	$S_2 = \{D_2, D_3, D_5\}$
$S_3 = \{D_3, D_4, D_5\}$	$S_3 = \{D_3, D_4, D_6\}$
$S_4 = \{D_4, D_5, D_6\}$	$S_4 = \{D_4, D_5, D_0\}$
$S_5 = \{D_5, D_6, D_0\}$	$S_5 = \{D_5, D_6, D_1\}$
$S_6 = \{D_6, D_0, D_2\}$	$S_6 = \{D_6, D_0, D_2\}$

Figure 1: Two groups of cyclic quorum sets where $p = 7, n = 3$ with different base sets

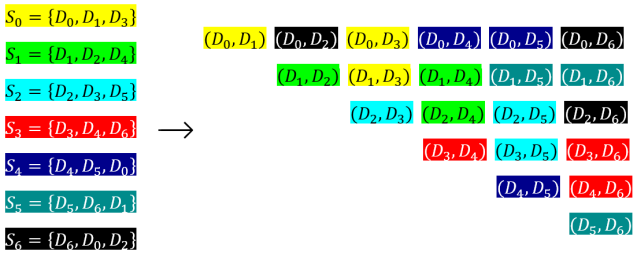

Figure 2: All-pairs property of cyclic quorum sets: in this case, $p = 7, n = 3$, there are $\frac{7\times(7-1)}{2} = 21$ pairs in total and they are listed on the right. Each base set on the left performs interactions between 3 pairs. All pairs are perfectly covered.

III. Search for All Feasible Base Sets

Notation 1. For simplicity, cyclic quorum sets are referred to as CQS and all data subsets in CQS are denoted by their indices. For example, $\{0, 1, 3\}$ denotes $S_0 = \{D_0, D_1, D_3\}$.

A. Equivalent sets

A CQS contains p base sets and could be developed from any of them. Our interest is to find one base set as the rest are equivalent and redundant. Moreover, any permutation of a base set is also equivalent.

Definition 2 (Standard form)**.** If base set elements are in ascending order, the base set is in standard form.

Definition 3 (Interest set)**.** If a feasible base set is in standard form and the starting two elements are $\{0, 1\}$, it is an interest base set or interest set.

Proposition 1. Given any feasible set, there exists an equivalent interest set.

Proof. Generate the optimal CQS using the given feasible set. Due to all-pairs property, the interaction between 0 and 1 must exist in a base set S_x. Then S_x is equivalent to the given set. Transform S_x into standard form, we get an equivalent interest set. Hence, this proposition holds. □

Introducing the concept of equivalent sets reduces the search space considerably. It not only removes two outermost loops, but also avoids repetitive searches as there are $pn! - 1$ sets equivalent to one base set.

B. Feasibility Check

A base set is feasible when the CQS generated based on it has all-pairs property. To determine if a CQS has all-pairs property, we exam its interest set. In [4], authors mentioned a method by generating a (p, n)-difference set. It is defined as follows: for a given set $A = \{a_0, \ldots, a_{n-1}\}$ and value of p, it is valid when all integers $0, \ldots, p$–1 are present by computing the differences of all possible pairs (a_i, a_j) modulus p. When a base set has a valid (p, n)-difference set, it is feasible.

Below we construct (p, n)-difference sets for two base sets mentioned in Section II. In this case, $p = 7, n = 3$ and two base sets are $\{0, 1, 2\}$ and $\{0, 1, 3\}$. In Fig. 3, we construct $(7, 3)$-difference set of $\{0, 1, 2\}$ on the left, it is not valid because difference 3 and 4 are missing. On the right is $(7, 3)$-difference set of $\{0, 1, 3\}$ and it is valid because all differences from 0 and 6 are present. Therefore, $\{0, 1, 3\}$ is an interest set while $\{0, 1, 2\}$ is not.

$(a_i - a_j)$ mod 7		a_i		
		0	1	2
a_j	0	0	1	2
	1	6	0	1
	2	5	6	0

$(a_i - a_j)$ mod 7		a_i		
		0	1	3
a_j	0	0	1	3
	1	6	0	2
	3	4	5	0

Figure 3: $(7, 3)$-difference sets of two base sets

We state the process of constructing (p, n)-difference set in $Feasibility\_check()$ function below. The complexity of this function is quadratic to the size of the base set.

C. Pairing Property

Given an interest set, another interest set could be determined and vice versa. We call it pairing property.

Definition 4 (Paired sets)**.** Given $p, n, \{0, 1, a_2, a_3, \ldots, a_{n-1}\}$ and $\{0, 1, p + 1–a_{n-1}, p + 1–a_{n-2}, \ldots, p + 1–a_2\}$ are paired sets. If two paired sets are identical, they are self-paired.

Proposition 2. Paired sets have the same feasibility.

Proof. Let X and Y be two paired sets, then we have bijection $\mathcal{F} : X \to Y$, $\mathcal{F}(x) = (p + 1–x)\ mod\ p$ and bijection $\mathcal{G} : Y \to X$, $\mathcal{G}(y) = (p + 1–y)\ mod\ p$. Thus, for any $i \neq j$, $a_i–a_j = (p+1–a_j)–(p+1–a_i)$, these two sets have the same (p, n)-difference set. Therefore, either both are feasible or not feasible. Hence, the proposition holds. □

Input: p and a base set $\{a_0, a_1, \ldots, a_{n-1}\}$
Output: pass or not pass

```
for i = 0 to p − 1 do
    difference[i] = 0
end for
for i = 0 to n − 1 do
    for j = 0 to n − 1 do
        difference[((a_i − a_j) mod p)]++
    end for
end for
if min(difference[] ≥ 1) then
    return pass
else
    return not pass
end if
```

Algorithm 1: $Feasibility\_check()$ function

D. Search Optimizations

In this section, we denote a set as $A = \{a_0, ..., a_{n-1}\}$. The naïve exhaustive search traverses all possible sets, thus, setting a_0 as the index of the outermost loop and a_{n-1} as the index of the innermost loop. There are n nested loops and the range of each index is $[0, p - 1]$. In every iteration, feasibility check is run and the total number of computations is p^n.

To optimize the search process, we choose to avoid certain space. After studying raw results generated by naïve search, we developed strategies that drastically reduce number of computations. Before introducing search optimizations, here are some prerequisite concepts we defined.

Definition 5 (Set coloring)**.** For each paired set, we mark the set traversed first yellow and the second red. Self-paired sets are marked green. Since paired sets share the same feasibility, only those marked in yellow and green should be traversed.

One set coloring example where $p = 8, n = 4$ is shown in Fig. 4. First two elements, a_0 and a_1, are omitted here because they are set to 0 and 1, as will be explained in Proposition 4, and two numbers in a pair are the values of a_2 and a_3.

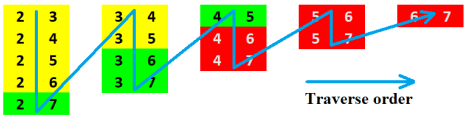

Figure 4: Set coloring and traverse order for $p = 8, n = 4$

Another case is shown in Fig. 5. There is no self-paired set and set coloring only generates yellow and red areas. Here we provide the conditions and prove when self-paired sets exist.

Proposition 3. The existence of self-paired set requires that either n is even or both p and n are odd.

Proof. To prove the necessity, when n is even, we investigate n–2 indices: $a_2, a_3, \ldots, a_{n-1}$. Assign $a_i = i$ for the first half of indices from a_2 to $a_{\frac{n}{2}}$. For the second half of indices, they

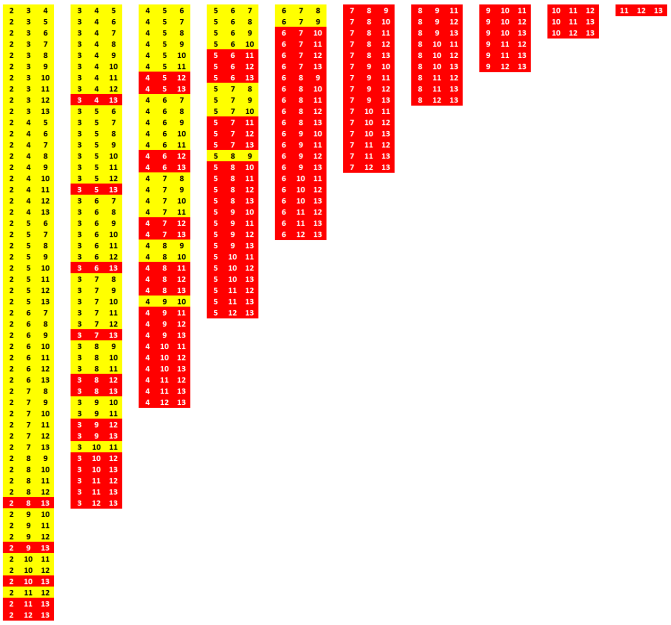

Figure 5: Set coloring for $p = 14, n = 5$

are constructed by $a_j = p + 1 - a_i$ where $i + j = n + 1$ for $2 \le i \le \frac{n}{2}, i \in \mathbb{Z}$. We construct a set $\{2, 3, \ldots, \frac{n}{2}, p + 1 - \frac{n}{2}, \ldots, p-2, p-1\}$. Its paired set is $\{p+1-(p-1), p+1-(p-2), \ldots, p+1-(p+1-\frac{n}{2}), p+1-(\frac{n}{2}), \ldots, p+1-3, p+1-2\} = \{2, 3, \ldots, \frac{n}{2}, p+1-\frac{n}{2}, \ldots, p-2, p-1\}$, which is identical. According to definition 4, this set is self-paired.

When both p and n are odd, we investigate n–2 indices $a_2, a_3, \ldots, a_{\frac{n+1}{2}}, \ldots, a_{n-1}$. We assign $a_i = i$ for indices before the middle index $a_{\frac{n+1}{2}}$, middle index $a_{\frac{n+1}{2}} = \frac{p+1}{2}$ and those after the middle index by $a_j = p + 1 - a_i$ where $i + j = n + 1$ for $2 \le i \le \frac{n-1}{2}, i \in \mathbb{Z}$. Then a set is constructed as $\{2, 3, \ldots, \frac{n-1}{2}, \frac{p+1}{2}, p+1-\frac{n-1}{2}, \ldots, p-2, p-1\}$. Its paired set is $\{p + 1 - (p - 1), p + 1 - (p - 2), \ldots, p + 1 - (p+1-\frac{n-1}{2}), p + 1 - \frac{p+1}{2}, p + 1 - \frac{n-1}{2}, \ldots, p + 1-3, p + 1-2\} = \{2, 3, \ldots, \frac{n-1}{2}, \frac{p+1}{2}, p+1-\frac{n-1}{2}, \ldots, p-2, p-1\}$, which is identical. According to Definition 4, this set is self-paired.

Therefore, when either n is even or both p and n are odd, at least one self-paired set exists in each case.

To prove the sufficiency, given a self-paired set $\{0, 1, a_2, a_3, \ldots, a_{n-1}\}$, we investigate n–2 elements starting from a_2. They satisfy:

If n is even,

$$\begin{cases} a_2 + a_{n-1} = p + 1 \\ a_3 + a_{n-2} = p + 1 \\ \quad \ldots \\ a_{\frac{n}{2}} + a_{\frac{n}{2}+1} = p + 1 \\ a_2 < a_3 < \cdots < a_{n-2} < a_{n-1} \end{cases}$$

If n is odd

$$\begin{cases} a_2 + a_{n-1} = p + 1 \\ a_3 + a_{n-2} = p + 1 \\ \quad \ldots \\ a_{\frac{n-1}{2}} + a_{\frac{n+3}{2}} = p + 1 \\ a_{\frac{n+1}{2}} = \dfrac{p+1}{2} \\ a_2 < a_3 < \cdots < a_{n-2} < a_{n-1} \end{cases}$$

Therefore, when n is even, p could be either odd or even. When n is odd, p must be odd or $a_{\frac{n+1}{2}}$ is not an integer. □

Definition 6 (Break point)**.** During the search, the first set, after which all sets are marked red is called break point.

Break points in the previous two cases where $p = 8$ and $p = 15$ are $\{(0, 1,)4, 5\}$ and $\{(0, 1,)6, 7, 9\}$, respectively.

Proposition 4. The expressions of break point are as listed in Table I with a_0 and a_1 set to 0 and 1 automatically.

Table I: Expressions of break point in all cases

p	n	break point						
odd	odd	a_2	...	$a_{\frac{n-1}{2}}$	$a_{\frac{n+1}{2}}$	$a_{\frac{n+3}{2}}$	...	a_{n-1}
		$\frac{p-n+4}{2}$	...	$\frac{p-1}{2}$	$\frac{p+1}{2}$	$\frac{p+3}{2}$	...	$\frac{p+n-2}{2}$
even	odd	a_2	...	$a_{\frac{n-1}{2}}$	$a_{\frac{n+1}{2}}$	$a_{\frac{n+3}{2}}$	...	a_{n-1}
		$\frac{p-n+3}{2}$	...	$\frac{p-1}{2}$	$\frac{p}{2}$	$\frac{p+4}{2}$	...	$\frac{p+n-1}{2}$
odd	even	a_2	...	$a_{\frac{n}{2}}$	$a_{\frac{n+2}{2}}$		...	a_{n-1}
		$\frac{p-n+4}{2}$	...	$\frac{p}{2}$	$\frac{p+2}{2}$		...	$\frac{p+n-2}{2}$
even	even	a_2	...	$a_{\frac{n}{2}}$	$a_{\frac{n+2}{2}}$		...	a_{n-1}
		$\frac{p-n+3}{2}$	...	$\frac{p-1}{2}$	$\frac{p+3}{2}$		...	$\frac{p+n-1}{2}$

Proof. Given two paired sets, $\{a_2, a_3, \ldots, a_{n-2}, a_{n-1}\}$ and $\{p + 1 - a_{n-1}, p + 1 - a_{n-2}, \ldots, p + 1 - a_3, p + 1 - a_2\}$, if the former set is marked yellow or green and the latter set is marked red, we have $a_2 \le p + 1 - a_{n-1}, a_3 \le p + 1 - a_{n-2}, \ldots, a_{n-1} \le p + 1 - a_2$.

To find the break point, the former set should be the last one in yellow or green and the latter set should be the first one in red, thus two paired set should as close as possible.

For a_2, it must be as large as possible while not exceed its lexicographical limit, $p+1-a_{n-1}$. Since $a_2 < a_3 < \cdots < a_{n-1}$ and each index must be at least 1 greater than the previous one, we have $a_3 \ge a_2 + 1, a_4 \ge a_2 + 2, \ldots, a_{n-1} \ge a_2 + n - 3$. Hence, $a_2 \le p + 1 - (a_2 + n - 3) \rightarrow a_2 \le \frac{p-n+3}{2}$.

For a_3, it is upper-bounded by $p + 1 - a_{n-2}$ and $a_{n-2} \ge a_3 + n - 5$, we obtain $a_3 \le \frac{p-n+6}{2}$. Applying the same process recursively, when all indices of the first set reach their maximum values, break point is determined.

Intuitively, after reaching the break point, there is no need to check the remaining part because those values are marked

in red. In practice, there are several conditions with a little difference according to the parity of p and n.

When both p and n are odd, the middle index is $a_{\frac{n+1}{2}}$ and its maximum value is $\frac{p+1}{2}$. Then the maximum values of its previous indices are one smaller than the other. The indices after the middle index can be calculated based on the first half.

When p is even and n is odd, the middle index is $a_{\frac{n+1}{2}}$. It satisfies $a_{\frac{n+1}{2}} \leq p+1-a_{\frac{n+1}{2}}$. Since $\frac{p+1}{2}$ is not an integer, we floor the value to $\frac{p}{2}$ and all other indices can be determined.

When n is even, all indices are divided into two halves. For the last index of the first half, $a_{\frac{n}{2}}$, it satisfies $a_{\frac{n}{2}} \leq p+1-a_{\frac{n+2}{2}}$. Since $a_{\frac{n+2}{2}} \geq a_{\frac{n}{2}}+1$, we have $a_{\frac{n}{2}} \leq \frac{p}{2}$. This value is an integer if p is even, and all other indices can be calculated. If p is odd, the maximum value of $a_{\frac{n}{2}}$ is $\lfloor \frac{p}{2} \rfloor = \frac{p-1}{2}$ and the rest indices could be determined accordingly. □

Definition 7 (Qualification test)**.** A set is compared with its paired set regarding lexicographic order. If its lexicographic order is no larger than that of its paired set. It passes the qualification test and becomes qualified for feasibility check.

Proposition 5. (Search Optimizations) First, set a_0 and a_1 to 0 and 1 respectively. Determine the break point. For sets before the break point, run feasibility check on those pass the qualification test. Return sets that pass the feasibility check.

Proof. According to Proposition 1, all feasible sets have one equivalent interest set. Once the interest set is found, all other equivalent base sets could be derived by either permutation or cyclic-manner generation. Here the goal is to find all distinguished interest sets. The first two elements of an interest set are 0 and 1. Hence, we set $a_0 = 0$ and $a_1 = 1$.

According to Definition 5 and 6, all sets after the break point and those before the break point but fail the qualification test fall into red area whose feasibility do not need to be checked as that of their paired set has been checked already. Hence, only run feasibility check on sets passing the qualification test, thus marked in yellow and green then return the feasible ones. □

This optimized search is stated as $CQS\_searching()$. It mainly consists of two functions, $Qualification\_test()$ and $Feasibility\_check()$. The former is a lexicographical comparison function and is well-implemented in many programming languages such as $lexicographical\_compare()$ in C++. Since it compares elements symmetrically until a mismatch is found, it has a time complexity up to linear. The other $Feasibility\_check()$ function requires computing $n(n-1)$ pairwise differences among n elements, therefore, it has quadratic time complexity. As the size of a set increases, $Feasibility\_check()$ is computationally more expensive than $Qualification\_test()$. Therefore, the number of computations can be approximately treated as the number of $Feasibility\_check()$ called.

There are some special values of p that a further search optimization strategy could be applied. In a distribute system with p nodes and each node is assigned n data subsets. The total amount of interactions performed by all nodes, with redundant results, is $\frac{pn(n-1)}{2}$. The amount of required interactions is $\frac{p(p-1)}{2}$. If a CQS has all-pairs property, we have $\frac{pn(n-1)}{2} \geq \frac{p(p-1)}{2}$. When they are equal, there is no redundant work and every possible interaction is performed exactly once. In this case, we have $p = n(n-1)+1$ and a further search space reduction strategy is proposed as follows.

Input: p and n
Output: all feasible interest sets
Initialization : $a_0 = 0, a_1 = 1$
Compute *break point* $B = \{b_2, b_3, \cdots, b_{n-1}\}$
for $a_2 = a_1 + 1$ **to** $p - n + 2$ **do**
  for $a_3 = a_2 + 1$ **to** $p - n + 3$ **do**
    ...
    for $a_{n-1} = a_{n-2} + 1$ **to** $p - 1$ **do**
      if $a_2 = b_2$ ...**and** $a_{n-1} = b_{n-1}$ **then**
        break
      else if $Qualification\_test() = pass$ **then**
        $Feasibility\_check()$
      end if
    end for
    ...
  end for
end for
return interest sets that pass the feasibility check
Algorithm 2: $CQS\_searching()$ function

Proposition 6. Only search the space where the differences between every two consecutive indices are mutually different.

Proof. We prove it by contradiction. Suppose there is a base set $S = \{a_0, a_1, \ldots, a_{n-1}\}$, in which $a_0 = 0, a_1 = 1, a_{j+1} - a_j = a_{i+1} - a_i = k$. We check the feasibility of this base set by first generating the whole CQS, namely $S_0, S_1, \ldots, S_{p-1}$. Their elements are denoted as $S_i = \{{a_0}^{(i)}, {a_1}^{(i)}, \ldots, {a_{n-1}}^{(i)}\}$ for $0 \leq i \leq p-1$. We focus on n of the base sets that contains element 0. Then we have ${a_{j+1}}^{(0)} - {a_j}^{(0)} = {a_{i+1}}^{(0)} - {a_i}^{(0)} = k$. According to the rule of generating CQS, in $S_{(p-{a_i}^{(0)}) \bmod p}$, we have ${a_i}^{((p-{a_i}^{(0)}) \bmod p)} = 0, {a_{i+1}}^{((p-{a_i}^{(0)}) \bmod p)} = k$ and in $S_{(p-{a_j}^{(0)}) \bmod p}$, we have ${a_j}^{((p-{a_j}^{(0)}) \bmod p)} = 0, {a_{j+1}}^{((p-{a_j}^{(0)}) \bmod p)} = k$. Thus, the interaction between 0 and k is performed three times, which violates the rule that no redundancy should exist. Therefore, base set S is not feasible. □

IV. RESULTS

Given the number of nodes in a distributed system, p, and the number of subsets each node processes, n, as mentioned in Section III, the total number of computations(N) to traverse the whole search space using naïve search is p^n. Specifically, one computation is doing one feasibility check. In our $CQS\_searching()$, we count the number of $Feasibility\_check()$ called as N.

In this section, we provide N comparisons between naïve search and $CQS\_searching()$ for p from 4 to 19 and for

special $p = n(n-1)+1$ where $n = 3, 4, 5, 8$. When $n = 7$, the corresponding value of p is $7 \times (7-1)+1 = 43$. However, the minimum size of the interest set is 8 rather than 7 as the results provided by Wai-Shing Luk et al. in [7]. We validate it using $CQS\_searching()$. Since the insight of p and n is not the focus of this paper, the case $p = 43$ is omitted for simplicity. We then show all feasible interest sets determined by $CQS\_searching()$ with an example where $p = 8, n = 4$.

Table II: Computation number comparisons between naïve search and $CQS\_searching()$ for $p = 4$ to 19, excluding special values

p	n	$N(\ddot{naive})$	$N(CQS\_searching())$	**Percentage(%)**
4	3	64	1	1.56
5	3	125	2	1.6
6	3	216	2	0.926
8	4	4096	9	0.220
9	4	6561	12	0.183
10	4	10000	16	0.16
11	4	14641	20	0.137
12	4	20736	25	0.121
14	5	537824	110	2.04×10^{-2}
15	5	759375	146	1.92×10^{-2}
16	5	1.05×10^6	182	1.73×10^{-2}
17	5	1.42×10^6	231	1.62×10^{-2}
18	5	1.89×10^6	280	1.48×10^{-2}
19	5	2.48×10^6	344	1.38×10^{-2}

Table III: Computation number comparisons between naïve search, $CQS\_searching()$ and after applying proposition 6 for special $p = n(n-1)+1$ where $n = 3, 4, 5, 6$ and 8

p	n	$N(\ddot{naive})$	$N(CQS\_searching())$	N(**Proposition 6**)
7	3	343	3	1
13	4	28561	36	9
21	5	4.08×10^6	489	108
31	6	8.87×10^8	11921	1800
57	8	1.11×10^{14}	1.45×10^7	1.09×10^6

Table II shows the computation number(N) comparisons between naïve search and $CQS\_searching()$. Not only significant amount reduction can be observed, but also as p getting larger, $CQS\_searching()$ requires fewer percentage of computations. In other words, our method has better performance as the problem size goes up.

In Table III, for these special p values, we provide an additional column that shows the value of how many feasibility checks are run after applying Proposition 6. The additional step that checks if a base set has duplicate difference between two consecutive indices cost linear time

As an example, we show all interest sets for $p = 8, n = 4$ in Table IV. They are presented in pairs and $CQS\_searching()$ produces sets in the left column. Those in the right column are calculated accordingly using pairing property.

More results of all interest sets are available at https://github.com/YimingBian/CyclicQuorumSets.

Table IV: All feasible interest sets for $p = 8$

$p = 8, n = 4$	
$\{0, 1, 2, 4\}$	$\{0, 1, 5, 7\}$
$\{0, 1, 2, 5\}$	$\{0, 1, 4, 7\}$
$\{0, 1, 2, 6\}$	$\{0, 1, 3, 7\}$
$\{0, 1, 3, 4\}$	$\{0, 1, 5, 6\}$
$\{0, 1, 3, 5\}$	$\{0, 1, 4, 6\}$

V. HEURISTICS TO CONSTRUCT QUORUMS WITH ALL-PAIRS PROPERTY FOR LARGE P

As introduced in Section I, the previous work by Wai-Shing Luk et al. [7] identifies one feasible interest set for $p = 4$ to 111 and 111 is the corresponding special p value when $n = 11$. As the value of p goes larger, the computation it takes to find even the first feasible interest set becomes extremely expensive. In real life, however, it is possible to have a system with tens of thousands or even millions of nodes. In this section, we try to solve this problem that given any value of p, especially very large ones, we provide a group of quorums that meets the all-to-all interaction requirement. However, the size of n may not be optimal. Hence, it is a compromised solution.

We first divide this problem into two sub problems based on the value of p. When p is a composite number, we present the idea using a simple example where $p = 28$ and explain the process and related propositions. Then we deal with the situation where p or its factor is a prime number by using an example where $p = 1,234,567,801$ and compare five methods that generate quorums with all-pairs property.

A. When p is a composite number

Suppose we only know interest sets for $p = 3$ to 10 and we want to construct a group of quorums for $p = 28$ that possesses all-pairs property, assuming that the whole data set is evenly divided into 28 subsets, namely $D_0, D_1, \ldots, D_{27}$.

The idea is to construct the solution in multiple layers. At each layer, we assume that the data subsets can be regrouped to compute all pair interactions. In this example, p can be decomposed into two factors as $28 = 4 \times 7$. Thus, we group 28 subsets into four groups in the first layer as shown in the yellow blocks at the top of Fig. 6. Consulting the interest set for $p = 4$, we choose $\{0, 1, 2\}$ as the interest set to generate CQS, based on which the first layer is constructed as shown in Fig. 7. Four quorums in this CQS are listed as the rest yellow blocks in Fig. 6.

To construct the second layer, we focus on each quorum of the first layer. In our example, the first quorum has three yellow blocks with index $0, 1$ and 2, thus $3 \times 7 = 21$ out of 28 data subsets. We regroup these 21 data subsets into 7 new blocks and each block contains 3 subsets. Applying the same process for the rest of the corresponding data sets in quorums of the first layer, we finally generate the map of subsets, marked in four different colors, as shown in Fig. 6.

Next, we use the interest set $\{0, 1, 3\}$ for $p = 7$, the second layer is constructed as shown in Fig. 7. Finally, we use the corresponding original data subsets according to the map and

to obtain the final quorums. This set of quorums meets all-to-all interaction requirement because each chunk of subsets guarantees to have all-pairs property, as they are constructed using an interest set, and they are independent.

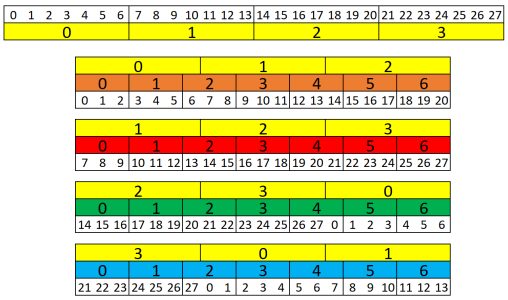

Figure 6: Regrouping phases of constructing quorums with all-pairs property for $p = 28$: since we decompose 28 into two factors, 4 and 7. Original 28 subsets are regrouped twice. They are first grouped into four yellow blocks. Based on CQS where $p = 4$, three of yellow blocks are gathered and regrouped into seven blocks, marked in orange, red, green and blue respectively

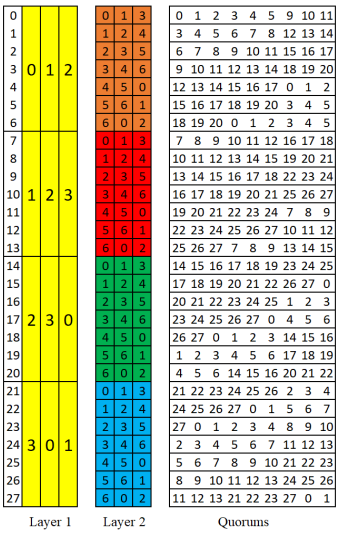

Figure 7: Constructing layers using the grouping results of the previous phase: retrieving the original subsets based on color blocks in Layer2, we construct a group of quorums that meet all-to-all interaction requirement

We state this heuristic method as follows. 1) Given a random composite number p, we perform factor decomposition and obtain $p = p_0 \times p_1 \cdots p_{k-1} = \prod_{i=0}^{k-1} p_i$. 2) Next we establish k layers. In layer i, we regroup the quorums developed in layer $i-1$ into p_i quorums, whose size is n_i, and construct the map of subsets. 3) Finally, we retrieve original data subsets using the map and form the group of quorums.

Below we provide the expression of the final quorum size in this heuristic method. In [6], Maekawa showed that the quorum size, n, of a group of quorums with all-pairs property has a lower bound of $\sqrt{p}$. Here we prove the quorum size in our heuristic method is a loose upper bound of n.

Proposition 7. (The size of the final quorum) The size of the final quorum is $\prod_{i=0}^{k-1} n_i$, where n_i is the quorum size corresponding to p_i as $p = \prod_{i=0}^{k-1} p_i$. This product is a loose upper bound of quorum size, of which a group of quorums has all-pairs property.

Proof. In the first layer, p is divided to p_0 blocks, each block has $\frac{p}{p_0}$ of the original subsets. As the quorum size of first layer is n_0, each quorum contains $p\frac{n_0}{p_0}$ original subsets. This value is passed to the next layer and the same process is applied. Thus, we have $p\frac{n_0}{p_0} \times \frac{n_1}{p_1}$ original subsets in the quorum. After processing all layers, the final quorum size is $p\frac{n_0}{p_0} \times \frac{n_1}{p_1} \times \cdots \times \frac{n_{k-1}}{p_{k-1}} = n_0 \times n_1 \cdots n_{k_1} = \prod_{i=0}^{k-1} n_i$.

As proved by Maekawa, if a group of quorums has all-pairs property, the lower bound of quorum size, n, is $\sqrt{p}$. The quorum size in our heuristic method, on the other hand, is a loose upper bound of n because this heuristic method guarantees the final set of quorums has all-pairs property. Moreover, when every n_i in $\prod_{i=0}^{k-1} n_i$ reaches its lower bound, which is $\sqrt{p_i}$, we have $\prod_{i=0}^{k-1} n_i = \prod_{i=0}^{k-1} \sqrt{p_i} = \sqrt{\prod_{i=0}^{k-1} p_i} = \sqrt{p}$. In conclusion, if a group of quorums has all-pairs property, the range of quorum size is $\sqrt{p} \leq n \leq \prod_{i=0}^{k-1} n_i$, where n_i is the corresponding quorum size to p_i and $p = \prod_{i=0}^{k-1} p_i$. □

The size of the final quorum, $\prod_{i=0}^{k-1} n_i$, depends on how p is decomposed. Larger the quorum size is, more redundant work exists in the system and more computation is needed. In order to minimize this product, there are two intuitions. First, one can minimize the number of layers and the smallest value is two. An example is shown in Table V. Given $p = 80$, three ways of factor decomposition are listed in the second column. 4 is no longer divided in to 2 because it makes little sense to apply CQS to a system with two nodes. Consulting the appendix in [7], the corresponding quorum size to $p = 4, 5, 16, 20$ are $3, 3, 5, 6$. Therefore, we compute the final quorum size as shown in the third column. Apparently, decomposing p into more factors has a higher chance that the corresponding n does not reach the lower bound. In this case, $p = 4$ does not reach its theoretical lower bound, which is $\sqrt{4} = 2$. Hence, decompose 20 or 16 further into 4 and 5 or 4 and 4 is not a good idea. Moreover, decomposing p into 2 factors generate different quorum sizes. Therefore, the second intuition is that there is a best way to factor decompose p that minimizes the final quorum size. For composite values of p, one can do exhaustive search in the worst case. The real challenge comes in the case where p is a very large prime number. We will discuss several approximation strategies in the next subsection.

B. When p is a prime number

When p or one factor of a big composite number p is a big prime number, it is highly possible that we do not know an interest base set off-the-shelf for this value because so far we only obtain interest sets for $p = 4$ to 111. In this subsection,

Table V: Quorum size comparisons with different factor decompositions of 80

p	Factor decomposition	Quorum size(n)
80	4×20	$3 \times 6 = 18$
	5×16	$3 \times 5 = 15$
	$4 \times 4 \times 5$	$3 \times 3 \times 3 = 27$

we discuss the problem of how to apply our heuristic method when p is a prime number, especially a big prime number.

Given a large prime number p, we view this problem as a system with p nodes where one could use up to p nodes, rather than exactly p nodes, to do the computation. Here we propose a compromised solution method that adjust the value of p and make it a composite number. Below we present five approximation strategies using an example where $p = 1,234,567,801$, a very big prime number whose corresponding quorum size has a theoretical lower bound of $\sqrt{p} = \sqrt{1,234,567,801} \approx 35,135$. In the first four approximation methods, we assume that p can be adjusted downwards. Then we recursively reduce p and perform factor decomposition as detailed below. In the last method, we assume p cannot be reduced and it is adjusted to a slightly greater value.

In the first method, the high-level idea is to make the difference between original value and adjusted value as small as possible. For the given $p_{original} = 1,234,567,801$, we have $p_{adjusted} = 1,234,567,801 - 1 = 1,234,567,800 = 2^3 \times 3^2 \times 5^2 \times 47 \times 14,593$. For the large prime factor $14,593$, we adjust it by $14,593 \rightarrow 14,593 - 1 = 14,592 = 2^8 \times 3 \times 19$. Next we combine factors together and make sure each of them is under 111 so that there is an interest set off-the-shelf. Here is one way of final factor combination $p_{adjusted} = 2^3 \times 3^2 \times 5^2 \times 47 \times (2^8 \times 3 \times 19) = 47 \times 57 \times 64 \times 72 \times 100 = 1,234,483,200$. Corresponding quorum sizes to $47, 57, 64, 72, 100$ are $8, 8, 9, 10, 12$ and the final quorum size is $8 \times 8 \times 9 \times 10 \times 12 = 69,120$.

In the second method, prime values are adjusted by either subtracting 1 or 2 and finally converted into the form $p \rightarrow 3^{k-1} \times p_r$. Subscript r stands for residual. We choose 3 because it is the only p value whose corresponding quorum size is the minimum value, which is 2. We convert $p_{original} = 1,234,567,801 \rightarrow 1,234,567,800 = 3^2 \times 137,174,200 \rightarrow 3^2 \times 137,174,199 = 3^3 \times 45,724,733 \rightarrow \cdots = 3^{15} \times 86 = 27 \times 81^3 \times 86 = 1,234,006,002 = p_{adjusted}$. Corresponding quorum sizes to $27, 81, 86$ are $6, 11, 11$ and the final quorum size is $6 \times 11^3 \times 11 = 87,846$.

In the third method, prime values are adjusted to the nearest multiple of 7 by subtracting a value between 0 and 6 and finally converted to the form $p \rightarrow 7^{k-1} \times p_r$. We choose 7 because it is the greatest p value whose corresponding quorum size is 3. We convert $p_{original} = 1,234,567,801 \rightarrow 1,234,567,796 = 7 \times 176,366,828 \rightarrow 7 \times 176,366,827 = 7^3 \times 514,189 \rightarrow \cdots = 7^9 \times 30 = 7 \times 30 \times 49^4 = 1,210,608,210 = p_{adjusted}$. Corresponding quorum sizes to $7, 30, 49$ are $3, 7, 8$ and the final size is $3 \times 7 \times 8^4 = 86,016$.

In the fourth method, we make the most use of 111 because it is the greatest p value whose corresponding interest set is known and we convert the original value to the form $p \rightarrow 111^{k-i} \times p_{r_0} \ldots p_{r_{i-1}}$. Different from previous two approximation method, we do not adjust prime values by subtracting a number between 0 to 110. Rather, we first find a value k' such that $111^{k'-1} < p < 111^{k'}$. Then convert p into the form $p \rightarrow 111^{k'-1} \times p_r$. In our case, $p_{original} = 1,234,567,801 \rightarrow 111^4 \times 8 = 1,214,456,328$. Corresponding quorum sizes to $111, 8$ are 12 and 4, so the final quorum size is $12^4 \times 4 = 82,944$. If $i \neq 1$, in other words, p_r does not fall into the range of 3 to 111. For further decomposition of p_r, the general rule is using large factors as much as possible. Here we provide two ways of decomposition with trade-offs. First, if a smaller difference between $p_{adjusted}$ and $p_{original}$ is preferred, we follow the same process as in the first approximation method. Second, if a smaller final quorum size is preferred, factors should only be special values. Although, in our case, $p_r = 8$ falls into the range, it can still be adjusted to the nearest smaller special value, which is 7. Thus, $p_{original} = 1,234,567,801 \rightarrow 111^4 \times 7 = 1,062,649,287$. Corresponding quorum sizes to $111, 7$ are 12 and 3, so the final quorum size is $12^4 \times 3 = 62,208$

In the fifth method, we assume that p cannot be adjusted downwards. This is common when the original dataset has already been evenly divided to p subsets and cannot be re-divided. Again we utilize $p = 111$ by first finding a value of k such that $111^{k-1} < p < 111^k$. Then we convert p to the form $p \rightarrow 111^{k-2} \times p_r$ to guarantee $111 < p_r < 111^2$. p_r is further modified so that the final $p_{adjusted}$ is slightly larger than the original value. Since $111^4 < 1,234,567,801 < 111^5$, we convert $p_{original} = 1,234,567,801 \rightarrow 111^3 \times 903 \rightarrow 111^3 \times 91 \times 10 = 1,244,544,210 = p_{adjusted}$. Corresponding quorum sizes to $111, 91, 10$ are $12, 10, 4$ and the final size is $12^3 \times 10 \times 4 = 69,120$.

Table VI: Comparisons regarding node utilization and approximation configuration among five methods

	$p_{adjusted}$	Utilization	Configuration
0	1,234,567,801	100%	NA
I	1,234,483,200	99.99%	$47 \times 57 \times 64 \times 72 \times 100$
II	1,234,006,002	99.95%	$3^{15} \times 86 = 27 \times 81^3 \times 86$
III	1,210,608,210	98.06%	$7^9 \times 30 = 7 \times 30 \times 49^4$
IV.I	1,214,456,328	98.37%	8×111^4
IV.II	1,062,649,287	86.07%	7×111^4
V	1,244,544,210	100.81%	$10 \times 91 \times 111^3$

In Table VI, We put the original case where $p = 1,234,567,801$ in Row 0 for straightforward reference. The following rows record results in the above five approximation methods. For Row IV.I, it shows the results of the fourth method with smaller p value difference preference and Row IV.II shows the results of the same method with smaller final quorum size preference. Utilization shows the percentage of nodes that are assigned work and it equals to $\frac{p_{adjusted}}{p_{original}} \times 100\%$. Intuitively, we expect utilization of a system to be high so that each node is assigned less workload and the overall computation time is shorter. Method I and II show similar high

Table VII: Comparisons regarding quorum size, subset size and number of data for a single node among five methods

	n	Subset size	Number of data for each node
0	35,135	$\frac{D}{1,234,567,801}$	$\frac{35,135D}{1,234,567,801} \approx 2.8459 \times 10^{-5}D$
I	69,120	$\frac{D}{1,234,483,200}$	$\frac{69,120D}{1,234,483,200} \approx 5.5991 \times 10^{-5}D$
II	87,846	$\frac{D}{1,234,006,002}$	$\frac{87,846D}{1,234,006,002} \approx 7.1188 \times 10^{-5}D$
III	86,016	$\frac{D}{1,210,608,210}$	$\frac{86,016D}{1,210,608,210} \approx 7.1052 \times 10^{-5}D$
IV.I	82,944	$\frac{D}{1,214,456,328}$	$\frac{82,944D}{1,214,456,328} \approx 6.8297 \times 10^{-5}D$
IV.II	62,208	$\frac{D}{1,062,649,287}$	$\frac{62,208D}{1,062,649,287} \approx 5.8540 \times 10^{-5}D$
V	69,120	$\frac{D}{1,234,567,801}$	$\frac{69,120D}{1,234,567,801} \approx 5.5538 \times 10^{-5}D$

utilization of over 99.95%. Method III and IV.I show slightly lower but comparable utilization of over 98%. Method IV.II has the worst utilization of 86.07%, thus almost 14% of given nodes are idle. Configuration column shows a way to factor decompose $p_{adjusted}$ into factors under 111.

Since the heuristic method is compromised for prime values, there is optimality among different approximations. In Table VII, we compare all approximation methods regarding final quorum size, data subset size and number of data assigned to each node. The final quorum size n indicates the number of data subset assigned to each node. It is the product of quorum sizes corresponding to the values in the configuration in Table VI. Compared to the theoretical lower bound of quorum size, $35,135$, final quorum sizes generated by all methods except IV.II are between $2\times$ to $2.5\times$ while method IV.II has the best final quorum size of $1.77\times$ to the lower bound. The size of subset equals to $\frac{D}{p}$, where D is the whole dataset and p is the number of nodes that do the computation. For the first four methods, it equals to $\frac{D}{p_{adjusted}}$ and for in the last case, it equals to $\frac{D}{p_{original}}$ because there are at most $p_{original}$ nodes participating in the computation. Last column shows the number of data assigned to each node compared to the whole dataset. Smaller value indicates fewer data is assigned, thus less workload and faster computation time. Method IV.II has the least workload assigned to each node and is roughly $2.2\times$ to the theoretical lowest workload. Therefore, it is optimal among all methods mentioned above.

VI. CONCLUSION

Facing the pressure of massive computations when solving all-pairs interaction problem, distributed system is a prevailing solution to meet the challenge. Data placement strategy is one of the key factors that decide how efficient a distributed system is. Applying cyclic quorum sets to manage data offers multiple advantages including achieving load balancing, reducing local memory resource requirement and enabling big data scalability.

In this paper, we focused on optimizing the search process for feasible base sets that could generate CQS with all-pairs property. Inspired by raw results observation, we discovered properties of feasible sets, based on which we developed search optimization strategies.

First we proved a great amount of base sets in the raw results are equivalent and defined the only set we need to look for, which are those starting with 0 and 1. Later we discovered the pairing property, taking advantage of which we managed to cut the search space roughly in half and defined the break point. The outcome is significant computation reduction.

Confronted with very big values of p, we also propose a constant time complexity method to construct a group of quorums to meet the all-to-all interaction requirement. Although the final quorums are not in cyclic manner, all intermediate steps are based on CQS generated by existing optimal-sized interest sets. If more interest sets of larger p values are provided, our decomposition method would have less redundancy and better performance.

CQS has already been applied not only for solving big data related problem, but also enhancing fault-tolerance in system design, improving link utilization in network communication problems and other scenarios. We hope our results provide researchers with diverse choices in the future study and trying different base sets could bring unexpected motivation and discovery.

ACKNOWLEDGMENT

THE RESEARCH REPORTED IN THIS PAPER IS PARTIALLY SUPPORTED BY THE PHILIP AND VIRGINIA SPROUL PROFESSORSHIP AT IOWA STATE UNIVERSITY.

REFERENCES

[1] Y.-F. Zhang, Y.-C. Tian, W. Kelly, C. Fidge, Scalable and efficient data distribution for distributed computing of all-to-all comparison problems, Future Generation Computer Systems 67 (2017) 152–162. doi:https://doi.org/10.1016/j.future.2016.08.020.

[2] A. Skabar, K. Abdalgader, Clustering sentence-level text using a novel fuzzy relational clustering algorithm, IEEE Transactions on Knowledge and Data Engineering 25 (1) (2013) 62–75. doi:10.1109/TKDE.2011.205.

[3] S. Boukhary, E. Colmenares, Study, analysis, and acceleration of an n-body simulation under many-core environments using an object oriented approach, in: 2019 International Conference on Computational Science and Computational Intelligence (CSCI), 2019, pp. 1506–1510. doi:10.1109/CSCI49370.2019.00280.

[4] C. J. Kleinheksel, A. K. Somani, Efficient distributed all-pairs algorithms: Management using optimal cyclic quorums, IEEE Transactions on Parallel and Distributed Systems 29 (2) (2018) 391–404.

[5] M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, K. Yelick, A communication-optimal n-body algorithm for direct interactions, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, pp. 1075–1084. doi:10.1109/IPDPS.2013.108.

[6] M. Maekawa, A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems, ACM Trans. Comput. Syst. 3 (2) (1985) 145–159. doi:10.1145/214438.214445.
URL https://doi.org/10.1145/214438.214445

[7] W.-S. Luk, T.-T. Wong, Two new quorum based algorithms for distributed mutual exclusion, in: Proceedings of 17th International Conference on Distributed Computing Systems, IEEE, 1997, pp. 100–106.

[8] Y. Bian, A. K. Somani, Establishing efficient all one-to-one paths by exploring cyclic quorum sets, in: 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2021, pp. 1–2. doi:10.1109/LANMAN52105.2021.9478794.

[9] C. J. Kleinheksel, A. K. Somani, Enhancing fault tolerance and resource utilization in unidirectional quorum-based cycle routing, IEEE/ACM Transactions on Networking 26 (2) (2018) 934–947. doi:10.1109/TNET.2018.2811386.

APPENDIX

Table VIII: First part of the group of quorums with all-pairs property for $p = 112$: This group of quorums are determined using heuristic method by decomposing p by $112 = 7 \times 16$. Constructing first layer with $p_1 = 7, n_1 = 3$, an interest set $\{0, 1, 3\}$ and second layer with $p_2 = 16, n_2 = 5$, an interest set $\{0, 1, 2, 5, 8\}$, the final quorum size is $n = n_1 \times n_2 = 15$, which is a very competitive value as the quorum size of $p = 111$ is 12.

Qrorums
$S_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 15, 16, 17, 24, 25, 26\}$
$S_1 = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 18, 19, 20, 27, 28, 29\}$
$S_2 = \{6, 7, 8, 9, 10, 11, 12, 13, 14, 21, 22, 23, 30, 31, 48\}$
$S_3 = \{9, 10, 11, 12, 13, 14, 15, 16, 17, 24, 25, 26, 49, 50, 51\}$
$S_4 = \{12, 13, 14, 15, 16, 17, 18, 19, 20, 27, 28, 29, 52, 53, 54\}$
$S_5 = \{15, 16, 17, 18, 19, 20, 21, 22, 23, 30, 31, 48, 55, 56, 57\}$
$S_6 = \{18, 19, 20, 21, 22, 23, 24, 25, 26, 49, 50, 51, 58, 59, 60\}$
$S_7 = \{21, 22, 23, 24, 25, 26, 27, 28, 29, 52, 53, 54, 61, 62, 63\}$
$S_8 = \{24, 25, 26, 27, 28, 29, 30, 31, 48, 55, 56, 57, 0, 1, 2\}$
$S_9 = \{27, 28, 29, 30, 31, 48, 49, 50, 51, 58, 59, 60, 3, 4, 5\}$
$S_{10} = \{30, 31, 48, 49, 50, 51, 52, 53, 54, 61, 62, 63, 6, 7, 8\}$
$S_{11} = \{49, 50, 51, 52, 53, 54, 55, 56, 57, 0, 1, 2, 9, 10, 11\}$
$S_{12} = \{52, 53, 54, 55, 56, 57, 58, 59, 60, 3, 4, 5, 12, 13, 14\}$
$S_{13} = \{55, 56, 57, 58, 59, 60, 61, 62, 63, 6, 7, 8, 15, 16, 17\}$
$S_{14} = \{58, 59, 60, 61, 62, 63, 0, 1, 2, 9, 10, 11, 18, 19, 20\}$
$S_{15} = \{61, 62, 63, 0, 1, 2, 3, 4, 5, 12, 13, 14, 21, 22, 23\}$
$S_{16} = \{16, 17, 18, 19, 20, 21, 22, 23, 24, 31, 32, 33, 40, 41, 42\}$
$S_{17} = \{19, 20, 21, 22, 23, 24, 25, 26, 27, 34, 35, 36, 43, 44, 45\}$
$S_{18} = \{22, 23, 24, 25, 26, 27, 28, 29, 30, 37, 38, 39, 46, 47, 64\}$
$S_{19} = \{25, 26, 27, 28, 29, 30, 31, 32, 33, 40, 41, 42, 65, 66, 67\}$
$S_{20} = \{28, 29, 30, 31, 32, 33, 34, 35, 36, 43, 44, 45, 68, 69, 70\}$
$S_{21} = \{31, 32, 33, 34, 35, 36, 37, 38, 39, 46, 47, 64, 71, 72, 73\}$
$S_{22} = \{34, 35, 36, 37, 38, 39, 40, 41, 42, 65, 66, 67, 74, 75, 76\}$
$S_{23} = \{37, 38, 39, 40, 41, 42, 43, 44, 45, 68, 69, 70, 77, 78, 79\}$
$S_{24} = \{40, 41, 42, 43, 44, 45, 46, 47, 64, 71, 72, 73, 16, 17, 18\}$
$S_{25} = \{43, 44, 45, 46, 47, 64, 65, 66, 67, 74, 75, 76, 19, 20, 21\}$
$S_{26} = \{46, 47, 64, 65, 66, 67, 68, 69, 70, 77, 78, 79, 22, 23, 24\}$
$S_{27} = \{65, 66, 67, 68, 69, 70, 71, 72, 73, 16, 17, 18, 25, 26, 27\}$
$S_{28} = \{68, 69, 70, 71, 72, 73, 74, 75, 76, 19, 20, 21, 28, 29, 30\}$
$S_{29} = \{71, 72, 73, 74, 75, 76, 77, 78, 79, 22, 23, 24, 31, 32, 33\}$
$S_{30} = \{74, 75, 76, 77, 78, 79, 16, 17, 18, 25, 26, 27, 34, 35, 36\}$
$S_{31} = \{77, 78, 79, 16, 17, 18, 19, 20, 21, 28, 29, 30, 37, 38, 39\}$
$S_{32} = \{32, 33, 34, 35, 36, 37, 38, 39, 40, 47, 48, 49, 56, 57, 58\}$
$S_{33} = \{35, 36, 37, 38, 39, 40, 41, 42, 43, 50, 51, 52, 59, 60, 61\}$
$S_{34} = \{38, 39, 40, 41, 42, 43, 44, 45, 46, 53, 54, 55, 62, 63, 80\}$
$S_{35} = \{41, 42, 43, 44, 45, 46, 47, 48, 49, 56, 57, 58, 81, 82, 83\}$
$S_{36} = \{44, 45, 46, 47, 48, 49, 50, 51, 52, 59, 60, 61, 84, 85, 86\}$
$S_{37} = \{47, 48, 49, 50, 51, 52, 53, 54, 55, 62, 63, 80, 87, 88, 89\}$
$S_{38} = \{50, 51, 52, 53, 54, 55, 56, 57, 58, 81, 82, 83, 90, 91, 92\}$
$S_{39} = \{53, 54, 55, 56, 57, 58, 59, 60, 61, 84, 85, 86, 93, 94, 95\}$
$S_{40} = \{56, 57, 58, 59, 60, 61, 62, 63, 80, 87, 88, 89, 32, 33, 34\}$
$S_{41} = \{59, 60, 61, 62, 63, 80, 81, 82, 83, 90, 91, 92, 35, 36, 37\}$
$S_{42} = \{62, 63, 80, 81, 82, 83, 84, 85, 86, 93, 94, 95, 38, 39, 40\}$
$S_{43} = \{81, 82, 83, 84, 85, 86, 87, 88, 89, 32, 33, 34, 41, 42, 43\}$
$S_{44} = \{84, 85, 86, 87, 88, 89, 90, 91, 92, 35, 36, 37, 44, 45, 46\}$
$S_{45} = \{87, 88, 89, 90, 91, 92, 93, 94, 95, 38, 39, 40, 47, 48, 49\}$
$S_{46} = \{90, 91, 92, 93, 94, 95, 32, 33, 34, 41, 42, 43, 50, 51, 52\}$
$S_{47} = \{93, 94, 95, 32, 33, 34, 35, 36, 37, 44, 45, 46, 53, 54, 55\}$
$S_{48} = \{48, 49, 50, 51, 52, 53, 54, 55, 56, 63, 64, 65, 72, 73, 74\}$
$S_{49} = \{51, 52, 53, 54, 55, 56, 57, 58, 59, 66, 67, 68, 75, 76, 77\}$
$S_{50} = \{54, 55, 56, 57, 58, 59, 60, 61, 62, 69, 70, 71, 78, 79, 96\}$

Table IX: Second part of the group of quorums with all-pairs property for $p = 112$

Qrorums
$S_{51} = \{57, 58, 59, 60, 61, 62, 63, 64, 65, 72, 73, 74, 97, 98, 99\}$
$S_{52} = \{60, 61, 62, 63, 64, 65, 66, 67, 68, 75, 76, 77, 100, 101, 102\}$
$S_{53} = \{63, 64, 65, 66, 67, 68, 69, 70, 71, 78, 79, 96, 103, 104, 105\}$
$S_{54} = \{66, 67, 68, 69, 70, 71, 72, 73, 74, 97, 98, 99, 106, 107, 108\}$
$S_{55} = \{69, 70, 71, 72, 73, 74, 75, 76, 77, 100, 101, 102, 109, 110, 111\}$
$S_{56} = \{72, 73, 74, 75, 76, 77, 78, 79, 96, 103, 104, 105, 48, 49, 50\}$
$S_{57} = \{75, 76, 77, 78, 79, 96, 97, 98, 99, 106, 107, 108, 51, 52, 53\}$
$S_{58} = \{78, 79, 96, 97, 98, 99, 100, 101, 102, 109, 110, 111, 54, 55, 56\}$
$S_{59} = \{97, 98, 99, 100, 101, 102, 103, 104, 105, 48, 49, 50, 57, 58, 59\}$
$S_{60} = \{100, 101, 102, 103, 104, 105, 106, 107, 108, 51, 52, 53, 60, 61, 62\}$
$S_{61} = \{103, 104, 105, 106, 107, 108, 109, 110, 111, 54, 55, 56, 63, 64, 65\}$
$S_{62} = \{106, 107, 108, 109, 110, 111, 48, 49, 50, 57, 58, 59, 66, 67, 68\}$
$S_{63} = \{109, 110, 111, 48, 49, 50, 51, 52, 53, 60, 61, 62, 69, 70, 71\}$
$S_{64} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 15, 64, 65, 72, 73, 74\}$
$S_{65} = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 66, 67, 68, 75, 76, 77\}$
$S_{66} = \{6, 7, 8, 9, 10, 11, 12, 13, 14, 69, 70, 71, 78, 79, 80\}$
$S_{67} = \{9, 10, 11, 12, 13, 14, 15, 64, 65, 72, 73, 74, 81, 82, 83\}$
$S_{68} = \{12, 13, 14, 15, 64, 65, 66, 67, 68, 75, 76, 77, 84, 85, 86\}$
$S_{69} = \{15, 64, 65, 66, 67, 68, 69, 70, 71, 78, 79, 80, 87, 88, 89\}$
$S_{70} = \{66, 67, 68, 69, 70, 71, 72, 73, 74, 81, 82, 83, 90, 91, 92\}$
$S_{71} = \{69, 70, 71, 72, 73, 74, 75, 76, 77, 84, 85, 86, 93, 94, 95\}$
$S_{72} = \{72, 73, 74, 75, 76, 77, 78, 79, 80, 87, 88, 89, 0, 1, 2\}$
$S_{73} = \{75, 76, 77, 78, 79, 80, 81, 82, 83, 90, 91, 92, 3, 4, 5\}$
$S_{74} = \{78, 79, 80, 81, 82, 83, 84, 85, 86, 93, 94, 95, 6, 7, 8\}$
$S_{75} = \{81, 82, 83, 84, 85, 86, 87, 88, 89, 0, 1, 2, 9, 10, 11\}$
$S_{76} = \{84, 85, 86, 87, 88, 89, 90, 91, 92, 3, 4, 5, 12, 13, 14\}$
$S_{77} = \{87, 88, 89, 90, 91, 92, 93, 94, 95, 6, 7, 8, 15, 64, 65\}$
$S_{78} = \{90, 91, 92, 93, 94, 95, 0, 1, 2, 9, 10, 11, 66, 67, 68\}$
$S_{79} = \{93, 94, 95, 0, 1, 2, 3, 4, 5, 12, 13, 14, 69, 70, 71\}$
$S_{80} = \{16, 17, 18, 19, 20, 21, 22, 23, 24, 31, 80, 81, 88, 89, 90\}$
$S_{81} = \{19, 20, 21, 22, 23, 24, 25, 26, 27, 82, 83, 84, 91, 92, 93\}$
$S_{82} = \{22, 23, 24, 25, 26, 27, 28, 29, 30, 85, 86, 87, 94, 95, 96\}$
$S_{83} = \{25, 26, 27, 28, 29, 30, 31, 80, 81, 88, 89, 90, 97, 98, 99\}$
$S_{84} = \{28, 29, 30, 31, 80, 81, 82, 83, 84, 91, 92, 93, 100, 101, 102\}$
$S_{85} = \{31, 80, 81, 82, 83, 84, 85, 86, 87, 94, 95, 96, 103, 104, 105\}$
$S_{86} = \{82, 83, 84, 85, 86, 87, 88, 89, 90, 97, 98, 99, 106, 107, 108\}$
$S_{87} = \{85, 86, 87, 88, 89, 90, 91, 92, 93, 100, 101, 102, 109, 110, 111\}$
$S_{88} = \{88, 89, 90, 91, 92, 93, 94, 95, 96, 103, 104, 105, 16, 17, 18\}$
$S_{89} = \{91, 92, 93, 94, 95, 96, 97, 98, 99, 106, 107, 108, 19, 20, 21\}$
$S_{90} = \{94, 95, 96, 97, 98, 99, 100, 101, 102, 109, 110, 111, 22, 23, 24\}$
$S_{91} = \{97, 98, 99, 100, 101, 102, 103, 104, 105, 16, 17, 18, 25, 26, 27\}$
$S_{92} = \{100, 101, 102, 103, 104, 105, 106, 107, 108, 19, 20, 21, 28, 29, 30\}$
$S_{93} = \{103, 104, 105, 106, 107, 108, 109, 110, 111, 22, 23, 24, 31, 80, 81\}$
$S_{94} = \{106, 107, 108, 109, 110, 111, 16, 17, 18, 25, 26, 27, 82, 83, 84\}$
$S_{95} = \{109, 110, 111, 16, 17, 18, 19, 20, 21, 28, 29, 30, 85, 86, 87\}$
$S_{96} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 15, 32, 33, 40, 41, 42\}$
$S_{97} = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 34, 35, 36, 43, 44, 45\}$
$S_{98} = \{6, 7, 8, 9, 10, 11, 12, 13, 14, 37, 38, 39, 46, 47, 96\}$
$S_{99} = \{9, 10, 11, 12, 13, 14, 15, 32, 33, 40, 41, 42, 97, 98, 99\}$
$S_{100} = \{12, 13, 14, 15, 32, 33, 34, 35, 36, 43, 44, 45, 100, 101, 102\}$
$S_{101} = \{15, 32, 33, 34, 35, 36, 37, 38, 39, 46, 47, 96, 103, 104, 105\}$
$S_{102} = \{34, 35, 36, 37, 38, 39, 40, 41, 42, 97, 98, 99, 106, 107, 108\}$
$S_{103} = \{37, 38, 39, 40, 41, 42, 43, 44, 45, 100, 101, 102, 109, 110, 111\}$
$S_{104} = \{40, 41, 42, 43, 44, 45, 46, 47, 96, 103, 104, 105, 0, 1, 2\}$
$S_{105} = \{43, 44, 45, 46, 47, 96, 97, 98, 99, 106, 107, 108, 3, 4, 5\}$
$S_{106} = \{46, 47, 96, 97, 98, 99, 100, 101, 102, 109, 110, 111, 6, 7, 8\}$
$S_{107} = \{97, 98, 99, 100, 101, 102, 103, 104, 105, 0, 1, 2, 9, 10, 11\}$
$S_{108} = \{100, 101, 102, 103, 104, 105, 106, 107, 108, 3, 4, 5, 12, 13, 14\}$
$S_{109} = \{103, 104, 105, 106, 107, 108, 109, 110, 111, 6, 7, 8, 15, 32, 33\}$
$S_{110} = \{106, 107, 108, 109, 110, 111, 0, 1, 2, 9, 10, 11, 34, 35, 36\}$
$S_{111} = \{109, 110, 111, 0, 1, 2, 3, 4, 5, 12, 13, 14, 37, 38, 39\}$