On Quasi Cycles in Hypergraph Databases

FAYED F. M. GHALEB¹, AZZA A. TAHA⊖¹, MARYAM HAZMAN², MAHMOUD M. ABD ELLATIF³,⁴, AND MONA ABBASS²

¹Department of Mathematics, Faculty of Science, Ain Shams University, Cairo 11566, Egypt
²Central Laboratory for Agricultural Experts Systems (CLAES), Agricultural Research Center, Giza 12619, Egypt
³College of Business, University of Jeddah, Jeddah 21589, Saudi Arabia
⁴Faculty of Computers and Artificial Intelligence, Helwan University, Cairo 11795, Egypt

Corresponding author: Azza A. Taha (azzataha@sci.asu.edu.eg)

ABSTRACT The notion of hypergraph cyclicity is important in numerous fields of application of hypergraph theory in computer science and relational database theory. The database scheme and query can be represented as a hypergraph. The database scheme (or query) has a cycle if the corresponding hypergraph has a cycle. An Acyclic database has several desired computational properties such as making query optimization easier and can be recognized in linear time. In this paper, we introduce a new type of cyclicity in hypergraphs via the notions of Quasi $\alpha$-cycle(s) and the set of $\alpha$-nodes in hypergraphs, which are based on the existence of an $\alpha$-cycle(s). Then, it is proved that a hypergraph is acyclic if and only if it does not contain any $\alpha$-nodes. Moreover, a polynomial-time algorithm is proposed to detect the set of $\alpha$-nodes based on the existence of Quasi $\alpha$-cycle(s), or otherwise claims the acyclicity of the hypergraph. Finally, a systematic discussion is given to show how to use the detected set of $\alpha$-nodes to convert the cyclic hypergraph into acyclic one if the conversion is possible. The acyclic database and acyclic query enjoy time and/or space-efficient access paths for answering a query.

INDEX TERMS Hypergraph, hypergraph acyclicity, relational database scheme, GYO algorithm.

I. INTRODUCTION

In the last decades, a class of “acyclic” database scheme and different degrees of acyclicity has been introduced [1]. Codd [2] has defined a relational database scheme as a collection of table skeletons (a set of subsets of the attributes, which are the column names of the database tables). These tables can be represented as hypergraphs. Each attribute of a database scheme $R$ corresponds to a node in a hypergraph $H$ and each relation scheme $R$ in $R$ corresponds to an edge in $H$ [3], [4].

The class of conjunctive queries (CQs) is one of the most important and simplest classes of database queries [5]. A conjunctive query is a form of queries with a logical conjunction operator. Each CQ can also be represented as hypergraph $H$. Acyclic CQs are efficiently solvable, i.e., all answers of an acyclic CQ can be computed in linear time [6].

A database scheme (or query) is said to be $\alpha$-acyclic (or acyclic) if the corresponding hypergraph is $[1, 7]$. In relational database theory, the notion of acyclicity has been particularly studied [1] and several acyclicity degrees have been introduced, such as $\alpha$-acyclicity, $\beta$-acyclicity, and $\gamma$-acyclicity [4], [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

The $\alpha$–acyclicity of a hypergraph is the least restrictive degree of acyclicity and it has more studies in the literature than the other two acyclicity degrees. The $\alpha$–acyclicity has many applications in relational databases for query optimization [4] and constraint programming [9].

An important function in any database system is query answering. A good database scheme design would allow information to be retrieved easily and efficiently [10]. Acyclic databases are preferred due to the variety of desired computational properties it enjoys such as making query optimization easier than in the case of cyclic database and might be recognized in linear time [11]–[17].

Due to the great importance of the acyclic database scheme, Graham [18] and Yu and Ozsoyoglu [19] have introduced a polynomial-time algorithm for detecting the acyclicity of hypergraphs that is known in the literature as Graham or the GYO algorithm. The algorithm returns either the hypergraph is acyclic or not, but does not detect the cycle(s) or the graph nodes that perform that cycle(s) in the case of cyclic hypergraphs.

This paper introduces the definitions of Quasi $\alpha$–cycle(s) and the set of $\alpha$-nodes in hypergraph $H$, which are based on the existence of $\alpha$–cycle(s). Then, it is proved that a hypergraph is acyclic if and only if it does not contain any
α-nodes. Also, a polynomial-time algorithm is proposed to detect the set of α-nodes based on the existence of Quasi α-cycle(s) or otherwise claims that the input hypergraph is an α-acyclic. Moreover, a systematic discussion is given to show how to use the detected set of α-nodes to convert the cyclic hypergraph into acyclic one if the conversion is possible. The acyclic hypergraph database and acyclic query enjoy time and/or space-efficient access paths for answering a query.

The paper is organized as follows: in Section II, the basic definitions of hypergraphs and databases scheme are given. Section III contains the related work. Section IV introduces a new formalization of the GYO Algorithm, the notions of Quasi α-cycles and the set of α-nodes, and finally the proposed polynomial algorithm for detecting the set of α-nodes, if it exists. In Section V, the result of the proposed algorithm and a systematic discussion is given, Finally, Section VI gives the conclusion of the paper and suggestions for future work.

II. PRELIMINARIES

In this section, the basic definitions of hypergraphs and its properties and database scheme are given.

Definition 1 (Hypergraph): A hypergraph H is a finite set of non-empty finite sets, called its hyperedges (or simply edges), the set V(H) of nodes of a hypergraph H, is defined to be the union of all its hyperedges [20]. For example, Figure 1, shows a hypergraph with four edges \( e_1, e_2, e_3, e_4 \), where \( e_1 = \{ABC\}, e_2 = \{CDE\}, e_3 = \{EFA\}, \) and \( e_4 = \{ACE\} \) (ABC is an abbreviation of A, B, C), etc., and \( V(H) = \{A, B, C, D, E, F\} \).

Definition 2 (Size of a Hypergraph): The size of a hypergraph \(|H|\) is defined to be the number of edges in it [3]. For example, the size of the hypergraph \(H\) in Figure 1 is 4.

Definition 3 (Trivial Set): A set \( S \) is said to be trivial, if it contains less than two elements [4], i.e., \(|S| = 1\).

Definition 4 (Size of an Edge): The size of an edge \(|e|\) is defined to be the number of nodes in it. For example, the size of any edge in the hypergraph of Figure 1 is 3.

Definition 5 (Singleton Edge): An edge \( e \in H \) is said to be a singleton, if \(|e| = 1\), i.e., if it contains exactly one node [4].

Definition 6 (Subhypergraph): A hypergraph \( H' \) is said to be a subhypergraph of a hypergraph \( H \) if \( H' \subseteq H \) [20].

Note that, the hypergraph \( H' \), in \( H' \subseteq H \), is obtained from \( H \) by removing edges, and not by removing nodes from edges. For example, Figure 2 shows that \( H' = \{e_1, e_2, e_3\} \) is a subhypergraph of the hypergraph in Figure 1 where \( V(H) = V(H') \).

Notation: Let \( H = \{e_1, \ldots, e_m\} \) with \( V(H) = \{v_1, \ldots, v_n\} \), then \( e_i - \{v_j\} \) denotes the edge \( e_i \) after removing the node \( v_j \) from it.

Definition 7 (Minimization of Hypergraph): The minimization, \( \text{Min}(H) \), of a hypergraph \( H \), is defined to be

\[
\text{Min}(H) = \{ e \in H | \exists f \in H, e \subset f \},
\]

i.e., the set of all maximal edges in \( H \) (for inclusion) [20]. It is obvious to see that definition 7 is equivalent to the definition of reduction of a hypergraph, which is introduced by Fagin [4].

Definition 8 (Minimized Hypergraph): A hypergraph \( H \) is said to be minimized if \( H = \text{Min}(H) \), that is, if no edge in \( H \) is a subset of another edge in it [4].

Definition 9 (Induced Hypergraph): The induced hypergraph of the hypergraph \( H \) on a set \( M \subseteq V(H) \), \( H[M] \), is defined to be

\[
H[M] = \{e \cap M | e \in H\} - \emptyset [20].
\]

This definition is the same as the definition of the set of partial edges generated by the set \( M \subseteq V(H) \) of a hypergraph, which is introduced by Fagin [4]. Figure 3 shows an example of the induced hypergraph of the hypergraph of Figure 1 by the set \( M = \{B, C, D, E\} \).

Definition 10 (Y-Tuple): Let \( V \) be a finite set of attributes and let \( Y \) a subset of \( V \), a \( Y \)-tuple (or tuple) is a mapping that associates a value for each attribute in \( Y \) [4].
Definition 11 (Y- Relation State): A Y- relation state r (a relation state r over Y or a relation state r) is a finite set of Y- tuples. If r is a Y- relation and X ⊆ Y, then r[X] is a projection of r onto X, which is the set of all tuples t[X], where t ∈ r[4].

Definition 12 (Database Scheme): A database scheme \( R = \{R_1, \ldots, R_p \} \), where each relation scheme \( R_i \) is a set of subsets of the attribute V, i.e., \( R_i = \{A_{i1}, A_{i2}, \ldots, A_{in}\} \), \( i = 1, \ldots, p \) and \( n_i \) is the number of the attributes of the relation scheme \( R_i \). Each attribute \( A_{ij} \) is associated with a domain \( D_{ij} \). This database scheme can be represented as hypergraph \( H = \{B_1, B_2, \ldots, B_p \} \) where \( B_i, i = 1, \ldots, p \) are its hyperedges such that, \( B_i = \bigcup_{j=1}^{n_i} A_{ij} \) and \( V(H) = \bigcup_{i=1}^{p} B_i \).

For example, in the hypergraph of Figure 5, \( H = \{\{\text{EMP}, \text{DEPT}, \text{SAL}\}, \{\text{EMP}, \text{CITY}, \text{MGR}\}, \{\text{EMP}, \text{STREET}, \text{CITY}, \text{CHILD}\}\} \) This hypergraph corresponds to the database scheme of Figure 4, where \( R = \{\text{EMP}_\text{WORK}, \text{DEPT}_\text{INFO}, \text{EMP}_\text{HOME}\} \) and consists of three relation schemes: an EMP_WORK relation scheme with attributes EMP (for employee), DEPT (for a department), and SAL (for salary); a DEPT_INFO relation scheme with attributes DEPT, STREET, CITY, and MGR (for manager); and an EMP_HOME relation scheme with attributes EMP, STREET, CITY, and CHILD.

Definition 13 (Properly Intersecting Edges): Two edges \( e \) and \( f \) of a hypergraph are called properly intersecting if \( e \varsubsetneq f \), \( f \varsubsetneq e \) and \( e \cap f \neq \emptyset \) [20]. For example, the two edges \( e_1 \) and \( e_2 \) in Figure 1 are properly intersecting.

Definition 14 (Star of a Node): Let \( H \) be a hypergraph and \( v \in V(H) \), the star \( \alpha(v) \) of the node \( v \), is defined to be the set of all edges containing \( v \), i.e., \( \alpha(v) = \{e \in E | v \in e\} \) [20]. For example, the star of the node \( A \) in the hypergraph \( H \) of Figure 1 is \( \alpha(A) = \{e_1, e_3, e_4\} \).

Definition 15 (Neighbours): Two nodes \( u \) and \( v \) in a hypergraph \( H \) are called neighbours in \( H \) if there is an edge \( e \in H \) such that \( \{u, v\} \subseteq e \). Equivalently, \( u \) and \( v \) are neighbours in \( H \) if and only if \( H(u) \cap H(v) \neq \emptyset \) [20]. For example, the two nodes \( A \) and \( B \) are neighbours in the hypergraph of Figure 1.

Definition 16 (Path): A path from node \( u \) to node \( v \), where \( u, v \in V(H) \), in a hypergraph \( H \) is a sequence of edges \( (e_1, \ldots, e_k) \) of length \( k \geq 1 \) such that:

(i) \( u \in e_1 \),
(ii) \( v \in e_k \), and
(iii) \( e_i \cap e_{i+1} \neq \emptyset \), for \( 1 \leq i < k \) [4].

It is also said that the sequence of edges \( (e_1, \ldots, e_k) \) is a path from \( e_1 \) to \( e_k \) if condition (iii) is satisfied. For example, the path from node \( A \) to node \( F \) in the hypergraph of Figure 1 is \( (e_1, e_2, e_3) \).

Definition 17 (Connected Nodes): Two nodes \( u \) and \( v \) of a hypergraph \( H \) are connected if there is a path from \( u \) to \( v \). Similarly, two edges \( e \) and \( f \) of a hypergraph \( H \) are connected if there is a path from \( e \) to \( f \). A set of nodes or edges is connected if every pair is connected [4]. A connected component of a hypergraph \( H \) is a maximal connected set of edges.

Definition 18 (Connected Hypergraph): A hypergraph \( H \) is connected if there is a path between each pair of hyperedges. Equivalently, a hypergraph is connected if it consists of only one component [21]. For example, the hypergraphs in Figures 1 and 2 are connected.

Definition 19 (Isolated Node): A node \( v \in V(H) \) is said to be isolated, if \( |\alpha(v)| = 1 \) or \( H(v) = \{e\} \), i.e., \( v \) belongs to precisely one edge [4].

III. RELATED WORK

A. \( \alpha \)-ACYCLICITY

The terms \( \alpha \)-acyclic hypergraph and acyclic hypergraph are synonymous.

Beeri et al. [11], introduced a special class of database schemes, called acyclic database scheme, which is based on the following concept of articulation set.

Definition 20 (Articulation Set): Let \( H = H[M] \) be a connected induced hypergraph by a set \( M \subseteq V(H) \), \( e, f \in H' \), and \( Q = e \cap f \). The pair \( e, f \) is an articulation pair, and the set \( Q \) is an articulation set of \( H' \) if \( H' = \{j \in H' : \forall j \in H' | Q \} \neq \emptyset \) is not connected [4]. For example, \( Q = \{AC\} \) is the articulation set for the set of all edges of the hypergraph in Figure 1, where \( Q = e_1 \cap e_4 \).

Definition 21 (Acyclic Hypergraph): A hypergraph \( H \) is \( \alpha \)-acyclic if every nontrivial connected minimized induced hypergraph by some set \( M \subseteq V(H) \) has an articulation set [4]. For example, the hypergraph in Figure 1 is \( \alpha \)-acyclic since the result of removing the articulation set \( Q = \{AC\} \) from each edge in \( H \) leaves the set of edges \( e_1 = \{B\}, e_2 = \{DE\}, e_3 = \{EF\} \), and \( e_4 = \{E\} \), which is not a connected hypergraph.

Philippe and Samba [21] proposed a definition of \( \alpha \)-cycles in hypergraphs, based on the same principle of \( \alpha \)-cycle in graph theory. This definition does not depend on the concept of articulation set, it depends instead on the following definitions:
Definition 22 (Sequence of Neighborhoods): Let e and f be two properly intersecting hyperedges of a hypergraph H. A sequence \((e = e_1, \ldots, e_p = f)\), such that \(p > 2\) is called a sequence of neighborhoods between e and f, if \(e \cap f \subseteq e_k \cap e_{k+1}\), for \(k = 1, \ldots, p - 1\) [21]. For example, \((e_1, e_4, e_2)\) in Figure 1 is a sequence of neighborhoods connecting \(e_1\) and \(e_2\).

Definition 23 (\(\alpha\)-Neighboring): Let e and f be two properly intersecting hyperedges of a hypergraph H, the two edges e and f are \(\alpha\)-neighboring if there is no sequence of neighborhoods between them [21]. For example, the two intersecting edges \(e_1\) and \(e_2\) of the hypergraph in Figure 2 are \(\alpha\)-neighboring, because there is no sequence of neighborhoods between \(e_1\) and \(e_2\), while the two intersecting edges \(e_1\) and \(e_2\) of the hypergraph in Figure 1 are not \(\alpha\)-neighboring, because \((e_1, e_4, e_2)\) is a sequence of neighborhoods between \(e_1\) and \(e_2\).

Definition 24 (\(\alpha\)-Path): An \(\alpha\)-path in a hypergraph H is a sequence of hyperedges \((e_1, \ldots, e_p)\), such that \(\forall j, 1 \leq j < p; e_j \cap e_{j+1} = \alpha\)-neighboring [21]. For example, the sequence \((e_1, e_2, e_3)\) in Figure 2 is \(\alpha\)-path.

Definition 25 (\(\alpha\)-Cycle): An \(\alpha\)-cycle in a hypergraph H is an \(\alpha\)-path \((e_1, \ldots, e_n)\) such that \(p > 3\), \(e_1 = e_p\), \(\exists 1 \leq a < b < n\), such that \(e_a \cap e_{a+1} \cap e_b \cap e_{b+1} \subseteq H\) [21]. For example, the sequence \((e_1, e_2, e_3, e_1)\) in Figure 2 is \(\alpha\)-cycle.

Theorem 1: A hypergraph H is \(\alpha\)-acyclic if and only if it does not contain an \(\alpha\)-cycle [21].

Remark 1: It is clear that, if e and f are \(\alpha\)-neighboring in hypergraph H, then for each sequence \((e_1, \ldots, e_p)\) \(\in H\), \(p > 2\), \(\exists 2\) two edges \(e_k\) and \(e_{k+1}\) in \((e_1, \ldots, e_p)\) such that \(e \cap f \subseteq e_k \cap e_{k+1}\), \(k = 1, \ldots, p - 1\).

B. THE GRAHAM (OR GYO) ALGORITHM

The GYO algorithm [18] was introduced to determine the \(\alpha\) -acyclicity of a hypergraph. The algorithm is applied to a hypergraph H and conveys to the two rules:

- Rule 1: If an edge e of H contains an isolated node, then delete this node from that edge.
- Rule 2: Delete edge e from H, if there is another edge f such that e \(\subseteq\) f.

These two rules are applied repeatedly until no rules can be applied by getting either the hypergraph H became empty and then H is acyclic, or the hypergraph H is not empty and then H is cyclic.

Theorem 2: A hypergraph H is \(\alpha\)-acyclic if and only if H became empty after applying the two rules of the GYO algorithm to H [4]. For example, the hypergraphs of Figure 2 and Figure 5 are \(\alpha\)-acyclic, while the hypergraph of Figure 1 is \(\alpha\)-acyclic.

Definition 26 (Consistent): Let r and s be two relation states of relation schemes R and S respectively. Then r and s are said to be consistent if \(r \cap S = s \cap R\), i.e., if the projections of r and s onto their common attributes are the same [1].

Definition 27 (Pairwise Consistent): Let \(r = \{r_1, \ldots, r_n\}\) be an arbitrarily state of \(R = \{R_1, \ldots, R_n\}\). Then r is called pairwise consistent if \(r_i\) and \(r_j\) are consistent for each i and j.

Also, r is called globally consistent if there is a relation s over attributes V = \(R_1 \cup \ldots \cup R_n\) such that \(r_i[R_i] = s[R_i]\) for each i, i.e., r is consistent if there is a “universal relation” s such that each \(r_i\) is a projection of s [4].

Definition 28 (Join Expression): The join operator is used to combine related tuples from two or more relations into a single tuple and is denoted by either \(r_1 \bowtie \ldots \bowtie r_n\) or \(\bowtie r\).

A join expression \(\theta\) of the relation state \(r\) is the set of all tuples t with attributes \(R_1 \cup \ldots \cup R_n\), such that \(t[R_i] = r_i\) for each i. For example, if \(r_1, r_2, r_3,\) R4 are among the relation schemes, then \(((R_2 \bowtie R_3) \bowtie (R_1 \bowtie R_4))\) is a join expression, which joining R2 and R3 relations, joining R1 and R4 relations, and then joining the two results together [4]. For example, consider the following join:

(EMP_WORK \(\bowtie_r EMP_WORK.DEPT=DEPT_INFO.DEPT\))

of the database of Figure 4. This combines each employee tuple with the tuple of the department where he/she works for into a single tuple.

Definition 29 (Sequential Join Expression): Let \(\theta\) be a join expression over \(R = \{R_1, \ldots, R_n\}\). If \(\theta\) is of the form \((\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_n))\), where \(R_1, \ldots, R_n\) is an ordering of the distinct members of R, then \(\theta\) is called sequential. For example, if \((\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_n))\) is a sequential join expression, then firstly the relations R1 and R2 are joining, then the result is joining with the R3 relation, and so on.

Notation: Let \(\theta\) be a join expression whose relation schemes are all in R, and let \(r = \{r_1, \ldots, r_n\}\) be a database over R. The relation that results by replacing each relation scheme \(R_i\) in \(\theta\) by \(r_i \in r\) and \(r_i\) has attributes \(R_i\) will be denoted by \(\theta (r)\) [4].

Definition 30 (Monotone Join Expression With Respect to \(\theta\)): Let \(\theta\) be a join expression over relation scheme \(R = \{R_1, \ldots, R_n\}\) and let \(r = \{r_1, \ldots, r_n\}\) be a database over \(R_i\), then \(\theta\) is called monotone with respect to \(r_i\), if for each subexpression \(\theta_1 \bowtie \theta_2\) of \(\theta\), the relations \(\theta_1(r)\) and \(\theta_2(r)\) are consistent [4], i.e., no tuples are lost in taking the join of relations r and s if r and s are consistent.

Definition 31 (Monotone Join Expression): A join expression \(\theta\) is called monotone if it is monotone with respect to every pairwise consistent database over \(R\) [4].

Theorem 3: Let \(r = \{r_1, \ldots, r_n\}\) be a database over relation scheme \(R\) then the following are equivalent.

1) \(R\) is \(\alpha\)-acyclic.
2) There is a monotone join expression over \(R\).
3) There is a monotone, sequential join expression over \(R\) [1].

IV. MATERIALS AND METHODS

This section will be started by reformulating the GYO algorithm followed by introducing the definition of Quasi-\(\alpha\)-cycle.
Algorithm 1 GYO (H)

Input: A connected hypergraph $H = \{e_1, \ldots, e_n\}$.
Output: A hypergraph GYO (H).

Begin
Repeat
1. For each node $v \in V(H)$ do
2. If $H(v) = \{e\}$ then
3. $e = e - v$
4. End If
5. End For
6. For all edges $e, f \in H$ do
7. If $e \subseteq f$ then
8. $H = H - e$
9. End If
10. End For
11. If $e \in H = \emptyset$ then
12. $H = H - e$
13. If $H$ contains an empty edge $e$ that results after removing isolated nodes from $e^*$
14. Return $H$

End

A new formalization of the GYO Algorithm

The GYO algorithm can be reformulated using the star of each node in Rule 1 of the algorithm as given above.

Remark 2: Let $H' = GYO(H)$ and $H' \neq \emptyset$, then:

1. $|H'| \leq |H|$ and $|H'| > 2$.
2. All nodes of $H'$ occurs at least in two edges, i.e., $\forall v \in V(H'), |H'(v)| \geq 2$. Moreover, $\exists e_1, e_2$ such that $\{e_1, e_2\} \subseteq H(v)$, and $v \in e_1 \cap e_2$, $e_1$ and $e_2$ are properly intersecting, i.e., $1 \leq i, j \leq 3$.
3. For each edge $e \in H'$, $|e| > 1$, i.e., all edges contain at least two nodes.
4. $Min(GYO(H)) = GYO(H)$.
5. If $H' = \{e_1, e_2, e_3\}$, then there exists $\{v_1, v_2, v_3\} \in V(H')$, such that $v_i \in e_i \cap e_{i+1}$ and $\{v_i, v_{i+1}\} \subseteq e_{i+1}$, where $v_4 = v_1$ and $v_4 = e_1, i, 1, \ldots, 3$.

B. A quasi $\alpha$-cycle in hypergraphs

Definition 32 (Quasi $\alpha$-Cycle): An $\alpha$-path $(e_1, \ldots, e_p)$, $p > 2$, is called quasi $\alpha$-cycle, $\alpha Q$-cycle, in a hypergraph $H$ if

(i) The two edges $e_1$ and $e_p$ are $\alpha$-neighboring and
(ii) $\exists 1 \leq a < b < p$, such that $e_a \cap e_{a+1} \subseteq e_b \cap e_{b+1}$.

For example, the sequence $(e_1, e_2, e_3)$ in Figure 2 is $\alpha Q$-cycle.

Proposition 1: If the sequence $(e_1, \ldots, e_p)$, $p > 2$ is $\alpha Q$-cycle and if $\forall k = 1 |e_k \cap e_{k+1} \notin e_p \cap e_1$, then $(e_1, \ldots, e_p, e_1)$ is $\alpha$-cycle.

Proof: Assume that the sequence $(e_1, \ldots, e_p)$, $p > 2$ is $\alpha Q$-cycle, by definition 32, $e_1$ and $e_p$ are $\alpha$-neighboring hence $(e_1, \ldots, e_p, e_1)$ is an $\alpha$-path.

Also, assuming that $\forall k = 1 |e_k \cap e_{k+1} \notin e_p \cap e_1$ by combining this assumption with condition (ii) in definition 32, we get:

$\exists 1 \leq a < b < p + 1 : e_a \cap e_{a+1} \subseteq e_b \cap e_{b+1}$

Therefore $(e_1, \ldots, e_p, e_{p+1} = e_1)$, is an $\alpha$-cycle.

Proposition 2: If the sequence $(e_1, \ldots, e_m)$, $m > 3$ is $\alpha$-cycle, then

1) The sequence $(e_1, \ldots, e_m)$ is $\alpha Q$-cycle and
2) $\forall k = 1 |e_k \cap e_{k+1} \notin e_m \cap e_1$. $\forall k = 1 |e_k \cap e_{k+1} \notin e_m \cap e_1$.

Proof: Assume that $(e_1, \ldots, e_m = e_1)$, $m > 3$, is an $\alpha$-cycle, then from definition 25 $(e_1, \ldots, e_m = e_1)$ is an $\alpha$-cycle, then each $e_1$ and $e_{m-1}$ are $\alpha$-neighboring and therefore $(e_1, \ldots, e_{m-1})$ is $\alpha Q$-cycle.

Also definition 25 yields that $\exists 1 \leq a < b < m$, such that $e_a \cap e_{a+1} \subseteq e_b \cap e_{b+1}$, this means that $\forall 1 \leq a < b < m - 1$, such that $e_a \cap e_{a+1} \notin e_b \cap e_{b+1}$, and $\forall k = 1 |e_k \cap e_{k+1} \notin e_m \cap e_1$.

Definition 33 (The set of $\alpha$-nodes): Let $H$ be a hypergraph, denote the set of all $\alpha$-cycle in $H$

$C_{\alpha Q}$-cycle, $= \{C : C$ is an $\alpha$-cycle in $H\}$. For a given cycle, $C = (e_1, \ldots, e_p)$, $p > 3$, the set

$N_{\alpha Q}(H) = (e_1 \cap e_q) \cup (\cup_{q=1}^{p-1} e_i \cap e_{i+1})$

is called the set of $\alpha$-nodes of the $\alpha$-cycle $C$ in a hypergraph $H$, and the set

$N_{\alpha Q}(H) = \cup_{C \in C_{\alpha Q}} N_{\alpha Q}(H)$

is called the set of $\alpha$-nodes in the hypergraph $H$.

Similarly, let $C_{\alpha Q}$-cycle, $= \{C : C \in C_{\alpha Q}$ is an $\alpha Q$-cycle $\}$. For a given cycle, $C_{\alpha Q} = (e_1, \ldots, e_p)$, $p > 2$, the set

$N_{\alpha Q}(H) = (e_1 \cap e_q) \cup (\cup_{q=1}^{p-1} e_i \cap e_{i+1})$

is called the set of $\alpha Q$-nodes of the $\alpha Q$-cycle, $C_{\alpha Q}$ in a hypergraph $H$, and the set

$N_{\alpha Q}(H) = \cup_{C \in C_{\alpha Q}} N_{\alpha Q}(H)$

is called the set of $\alpha Q$-nodes in $H$. For example, the set $\{A, C, E\}$ is the set of $\alpha$-nodes of the hypergraph $H$ in Figure 2. Since this set is the intersecting of the edges in the $\alpha$-cycle $(e_1, e_2, e_3, e_1)$, this set is the same as the set of $\alpha Q$-nodes of the hypergraph $H$ in the same figure since this set is the intersecting of the edges in the $\alpha Q$-cycle $(e_1, e_2, e_3)$.

Lemma 1: Let $H' = GYO(H)$ and $H' \neq \emptyset$, then each edge $e \in H'$ belongs to an $\alpha Q$-cycle, of edges $H'$, i.e., there exists an $\alpha Q$-cycle $(e_1, \ldots, e_k)$, such that $e = e_i$, for some $i$, $1 \leq i \leq k$.

Proof: Let $H' = \{e_1, \ldots, e_n\}, n > 2$, then from Remark 2, $|H'| > 2$. 

VOLUME 8, 2020
Let $e \in H'$, then from Remark 2, $e$ contains at least two nodes, i.e., $\exists v_1$ and $v_2 \in V(H)$, such that $\{v_1, v_2\} \subseteq e$, and $\exists e', e'' \in H'$, such that $e' \cap e \neq \emptyset$, and $e'' \cap e \neq \emptyset$, where $|e'| > 1$ and $|e''| > 1$.

Let $e_1 = e'$, $e_2 = e$, and $e_3 = e''$, then there exists the sequence $(e', e, e'')$, with $e_a \cap e_{a+1} \subseteq e_b \cap e_{b+1}$, $1 \leq a < b \leq 2$. This sequence satisfied either:

1. $e_1 \cap e_3 \neq \emptyset$, and the sequence $(e_1, e_2, e_3)$ is $\alpha_Q$–cycle, or
2. $\exists f \in H' : e_1 \cap f \neq \emptyset$, $e_3 \cap f \neq \emptyset$, Setting $f = e_4$, we get the sequence $(e_1, e_2, e_3, e_4)$ is $\alpha_Q$–cycle, or
3. $\exists$ Two edges $g, h \in H' : e_1 \cap g \neq \emptyset$ and $e_3 \cap h \neq \emptyset$. Setting $h = e_4$, and $g = e_5$, we get the sequence $(e_1, e_2, e_3, e_4, e_5)$ with $e_1 = e$, $\forall_{k=1}^{4} e_k, e_{k+1}$ are properly intersecting, $e_1$ and $e_k$ are properly intersecting, and $\#1 \leq a < b < 4$,

$$e_{ka} \cap e_{ka+1} \subseteq e_{kb} \cap e_{kb+1}$$

Continuing, in the same manner, we get that $\forall e \in H'$ there exists a sequence $(e_1, \ldots, e_k)$, such that $e_1$ and $e_{k+1}$ are properly intersecting, and $e_1$ and $e_k$ are properly intersecting, and $\#1 \leq a \leq b < k-1$, $e_a \cap e_{a+1} = e_b \cap e_{b+1}$, $1 \leq i \leq k - 1$. That is, $(e_1, \ldots, e_k)$ is $\alpha_Q$–cycle, $C_Q$, such that $e \in C_Q$, where $H' \subseteq \cup C_Q$ and $e \in C_Q$. \hfill \Box

Remark 3:

(i) Every $\alpha_Q$–cycle, $C_Q = (e'_1, \ldots, e'_p) \in H$, is associated with an $\alpha$–cycle, $C = (e_1, \ldots, e_n, e_1)$ in $H$ and vice versa.

(ii) Let $C = (e_1, \ldots, e_p)$ is an $\alpha$–cycle in a hypergraph $H$ and let $C_Q = (e_1, \ldots, e_{p-1})$ be its associated $\alpha_Q$–cycle in $H$ then

- $N^C_Q(H) = N^C_{\alpha_Q}(H)$, i.e., the set of $\alpha$–nodes in the cycle $C$ equals to set of $\alpha$–nodes in the Quasi cycle $C_Q$ and
- $N_a(H) = \cup_{C} N^C_a(H) = \cup_{C} N^C_{\alpha}(H) = N_{\alpha Q}(H)$

(iii) If $H' = GYO(H)$, then from Lemma 1 and the definition of the set of $\alpha$–nodes in the hypergraph $H'$ is the set of $\alpha$–nodes the hypergraph $H$, i.e., $N(H') = N_{\alpha}(H)$.

Algorithm 2 $A_0(H)$

**Input:** A hypergraph $H$ corresponding to a given database scheme.

**Output:** The set of $\alpha$-node(s) of $H$ if $H$ is cyclic or $\emptyset$ if the input hypergraph $H$ is $\alpha$-acyclic.

**Begin**

1. $H' = GYO(H)$
2. If $H' = \emptyset$ then return $\emptyset$ /* $H$ is $\alpha$-acyclic */
3. Else return $V(H')$
4. End If

**End**

The algorithm is called $A_0(H)$ and it depends on the output of the $GYO(H)$ algorithm. As indicated in Lemma 1, if $H' = GYO(H)$ and $H' \neq \emptyset$, then for each edge $e$ in $H'$, there exists a sequence $(e_1, \ldots, e_k)$, such that $e = e_i$, $1 \leq i \leq k$ and $(e_1, \ldots, e_k)$ is $\alpha_Q$–cycle. Also, from Remark 3 (iii) and the definition of the set of $\alpha$–nodes, the set of nodes of the hypergraph $H'$ is the set of $\alpha$–nodes the hypergraph $H$, i.e., $N(H') = N_{\alpha}(H)$.

Algorithm $A_0(H)$ uses the linear time $GYO$ algorithm, therefore $A_0(H)$ is also a linear time algorithm.

**V. DISCUSSION**

Detecting the set of $\alpha$–nodes is important to convert the cyclic hypergraph into acyclic one if the conversion is possible. More precisely, it is enough to detect the set of $\alpha$–nodes to convert the cyclic hypergraph into acyclic one instead of detecting the $\alpha$-cycle(s), that requires checking all permutations which is an NP problem.

Ghaleb et al. [22], proposed an algorithm, $\alpha_{Rem}(H, n, K)$, to convert a cyclic hypergraph $H$, which corresponds to a database scheme $R$ into an acyclic one. The algorithm returns an acyclic hypergraph if the conversion is possible or return failure otherwise. The input of this algorithm is:

1. An undirected hypergraph, $H$, corresponding to a cyclic database scheme in the third normal form. The third normal form guarantees that each non-key attribute $A$ in the relation scheme $R$ is fully functionally dependent on the primary key of $R$, and no non-key attribute of $R$ is transitively dependent on the primary key.
2. The set of $\alpha$-node(s), $n$, of the hypergraph $H$, which was returned from algorithm $A_0(H)$. The set of $\alpha$-node(s), $n$, returns from algorithm $A_0(H)$ since the hypergraph $H$ is cyclic.
3. A set of keys, $K$, which is the set of all keys of this database scheme $R$.

Algorithm $\alpha_{Rem}(H, n, K)$ has two steps:
1. The first step renames only the $\alpha$-node(s) which represents the non-key attributes in the database scheme $R$ according to the name of the table it belongs to.

2. After renaming the first non-key attribute, algorithm $\alpha_{Rem}(H, n, K)$ calls algorithm $A_0(H)$ to determine whether the resulted hypergraph became acyclic or still cyclic. The algorithm returns the set of $\alpha$-node(s) if it is cyclic.

These two steps are applied repeatedly until the hypergraph becomes acyclic or all the $\alpha$-nodes of $H$ are keys (which cannot be renamed) and hence the algorithm returns failure.

For instance, consider the hypergraph corresponds to the database scheme, each node corresponds to an attribute in a table of the database which can be either a key or non-key in its table. It is desirable to study the possibility of renaming an attribute to another name.

For example, after applying the proposed algorithm $A_0(H)$ to the hypergraph of Figure 5, corresponding to the database scheme of Figure 4, the following set of $\alpha$-nodes is obtained:

$$N_\alpha(H) = \{\text{EMP, DEPT, CITY}\}.$$ 

This set has two key attributes EMP, DEPT, and a non-key attribute CITY. Therefore, from the instance of the cyclic database of Figure 4, there are two distinct $\{\text{EMP, CITY}\}$ relationships. The first one has the tuple (Lorin, Giza) that relates an employee Lorin to the city Giza where she works. This relationship is obtained from joining the EMP_WORK relation with the DEPT_INFO relation. The second one has the tuple (Lorin, Cairo) that relates an employee Lorin to the city Cairo where she lives in the EMP_HOME relation. Since the attribute CITY is a non-key, we can rename it in the DEPT_INFO relation scheme to WORK_CITY, and the attribute CITY in the EMP_HOME relation scheme to HOME_CITY. There is now a unique $\{\text{EMP, HOME_CITY}\}$ relationship, which includes the tuple (Lorin, Cairo), and a unique $\{\text{EMP, WORK_CITY}\}$ relationship, which includes the tuple (Lorin, Giza). This gives the acyclic hypergraph of Figure 7, which is corresponding to the database schemes of Figure 6.

It is important to note that, when relationships are unique, it preserves the semantics of the database scheme [23] and the system has a great deal of flexibility in optimizing how to find the result of any query. It is also following from Theorem 3 that, the system might be able to exploit the fact that whatever relations in the database are joined together, the join expression is guaranteed to be monotone, and therefore be efficient.

Note that, when a relational database scheme $R$ is $\gamma$-acyclic then, there is a unique relationship among each set of attributes, for each consistent database over $R$ [4], while in the case of $\alpha$-acyclicity this uniqueness is not always guaranteed. That is because each subgraph of the $\gamma$-acyclic hypergraph is $\gamma$-acyclic, but not each subgraph of the $\alpha$-acyclic hypergraph is $\alpha$-acyclic.

The cyclic queries (that have a corresponding cyclic hypergraph) need exponential time [24] to be computed even for small outputs, (just one tuple or checking whether the answer of a query is non-empty). Therefore, many attempts have been made in the literature to specify how suitably transforming a cyclic CQ into an equivalent acyclic one [25]–[32]. In this framework, detecting the set of $\alpha$-nodes will facilitate this transformation, or finding acyclic approximations [33] for such query, i.e., to find another approximate acyclic CQ’ that will be much faster than CQ, and has an output that is close to the output of the original CQ on all databases.

VI. CONCLUSION AND FUTURE WORK

We introduced a new formalization of the GYO algorithm using the star of the hypergraph nodes. A new type of cyclicity in hypergraphs is also introduced. The new type is called the Quasi $\alpha$-cycle. Moreover, the notion of the set of $\alpha$-nodes in hypergraphs is also introduced. The Quasi $\alpha$–cycle and the set of $\alpha$-nodes are based on the existence of $\alpha$–cycles. A polynomial-time algorithm is also proposed to detect the set of $\alpha$-nodes that is based on the existence of Quasi $\alpha$-cycle(s), if it exists or otherwise claims that the input hypergraph is an $\alpha$-acyclic. The detected set of $\alpha$-nodes is important to study the possibility of converting the cyclic hypergraph into an acyclic one. More precisely, it is enough to detect the set of $\alpha$–nodes to convert the cyclic hypergraph into acyclic one instead of detecting the $\alpha$-cycle(s), that requires checking all permutations which is an NP problem. Acyclic databases are preferred due to the variety of desired computational properties it enjoys, such as making query optimization easier than in the case of cyclic database and might be recognized in linear time. The acyclic database and acyclic query enjoy time and/or space-efficient access paths for answering a query. Detecting the set of $\alpha$–nodes will facilitate transforming the cyclic query into an equivalent acyclic one or finding acyclic approximations for such a query. For future work, more study of various cases of
the attributes (key and non-key) is needed to convert a given cyclic database schema into an acyclic one. Also, we will extend our work to introduce the set of $\beta$-nodes and $\gamma$-nodes, which corresponds to $\beta$-cycle and $\gamma$-cycle respectively.

REFERENCES

[1] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, “On the desirability of acyclic database schemes,” J. ACM, vol. 30, no. 3, pp. 479–513, Jul. 1983.

[2] E. F. Codd, “A relational model of data for large shared data banks,” Commun. ACM, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[3] Y. M. Choe, L. J. Li, A. Lim, and A. K. H. Tung, “Arboricity: An acyclic hypergraph decomposition problem motivated by database theory,” Discrete Appl. Math., vol. 160, nos. 1–2, pp. 100–107, Jan. 2012.

[4] R. Fagin, “Degrees of acyclicity for hypergraphs and relational database schemes,” J. ACM, vol. 30, no. 3, pp. 514–550, Jul. 1983.

[5] H. Chen and Y. Yoshida, “Testability of homomorphism inadmissibility: Property testing meets database theory,” in Proc. 38th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst. (PODS), New York, NY, USA, Jun./Jul. 2019.

[6] M. Yannakakis, “Algorithms for acyclic database schemes,” in 7th Int. Conf. Very Large Data Bases, Cannes, France, vol. 7, Sep. 1981.

[7] D. Maier and J. D. Ullman, “Connections in acyclic hypergraphs,” Theor. Comput. Sci., vol. 32, nos. 1–2, pp. 185–199, 1984.

[8] D. Duris, “Some characterizations of $\gamma$ and $\beta$-acyclicity of hypergraphs,” Inf. Process. Lett., vol. 112, no. 16, pp. 617–620, Aug. 2012.

[9] R. Dechter, Constraint Processing, Burlington, MA, USA: Morgan Kaufmann, 2003.

[10] E. F. Codd, “A relational model of data for large shared data banks,” Commun. ACM, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[11] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, and M. Yannakakis, “Properties of acyclic database schemes,” in Proc. 13th Annu. ACM Symp. Theory Comput. (STOC), Milwaukee, WI, USA, May 1981, pp. 355–362.

[12] C. H. Papadimitriou and M. Yannakakis, “On the complexity of database queries,” J. Comput. Syst. Sci., vol. 58, no. 3, pp. 407–427, Jun. 1999.

[13] C. Chekuri and A. Rajaraman, “Conjunctive query containment revisited,” Theor. Comput. Sci., vol. 239, no. 2, pp. 211–229, May 2000.

[14] S. Chokkalingam, “Database and query analysis tools for MySQL: Exploiting hypertree and hypergraph decompositions,” M.S. thesis, Faculty Russ Coll. Eng., Tech. Ohio Univ., Columbus, OH, USA, 2006.

[15] G. Bagan, A. Durand, and E. Grandjean, “On acyclic conjunctive queries and constant delay enumeration,” in Computer Science Logic (Lecture Notes in Computer Science), J. Duparc and T. A. Henzinger, Eds, Berlin, Germany: Springer, 2007.

[16] K. Ahmet and O. Dan, “Covers of query results,” in Proc. 21st Int. Conf. Database Theory (ICDT), Vienna, Austria, Mar. 2018, pp. 1–16.

[17] X. Hu and K. Yi, “Instance and output optimal parallel algorithms for acyclic joins,” in Proc. 38th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst. (PODS), Amsterdam, The Netherlands, Jun./Jul. 2019, pp. 450–463.

[18] M. H. Graham, “On the universal relation,” Univ. Toronto, Toronto, ON, Canada, Tech. Rep., Sep. 1979.

[19] C. T. Yu and M. Z. Ozsoyoglu, “An algorithm for tree-query membership of a distributed query,” in Proc. Comput. Softv. IEEE Computer Societys 3rd Int. Appl. Conf. (COMPSAC), Chicago, IL, USA, Nov. 1979, pp. 306–312.

[20] J. Brautl-Baron, “Hypergraph acyclicity revisited,” ACM Comput. Surv., vol. 49, no. 3, p. 54, 2016.

[21] P. Jégou and S. N. Ndiaye, “On the notion of cycles in hypergraphs,” Discrete Math., vol. 309, nos. 23–24, pp. 6535–6543, Dec. 2009.

[22] F. M. Ghaleb, A. A. Taha, M. Hazman, M. A. ElIntatif, and M. Abbass, “RDF-BF-hypergraph representation for relational database,” Int. J. Math. Comput. Sci., vol. 15, no. 1, pp. 41–64, 2020.

[23] R. Hull and R. King, “Semantic database modeling: Survey, applications, and research issues,” ACM Comput. Surv., vol. 19, no. 3, pp. 201–260, Sep. 1987.

[24] W. Fischl, G. Gottlob, and R. Pichler, “General and fractional hypertree decompositions: Hard and easy cases,” 2016, arXiv:1611.01090, [Online]. Available: http://arxiv.org/abs/1611.01090

[25] M. Gyssens and J. Paredaens, “A decomposition methodology for cyclic databases,” in Advances in Data Base Theory (Lecture Notes in Advances in Data Base Theory), H. Gallaire, J. Minker, and J. M. Nicolas, Eds. Boston, MA, USA: Springer, 1984.

[26] G. Gottlob, N. Leone, and F. Scarcello, “Hypertree decompositions and tractable queries,” J. Comput. Syst. Sci., vol. 64, no. 3, pp. 579–627, May 2002.

[27] F. Scarcello, “Query answering exploiting structural properties,” ACM SIGMOD Rec., vol. 34, no. 3, pp. 91–99, Sep. 2005.

[28] G. Georg, G. Gianluigi, and F. Scarcello, “Tractable optimization problems through hypergraph-based structural restrictions,” in Automata, Languages and Programming ( Lecture Notes in Computer Science), S. Albers, A. Marchetti-Spaccamela, Y. Mattias, S. Nikoletseas, and W. Thomas, Eds. Berlin, Germany: Springer, 2009.

[29] G. Gottlob and R. Pichler, “Hypergraphs in model checking: Acyclicity and hypertree-width versus clique-width,” SIAM J. Comput., vol. 33, no. 2, pp. 351–378, Jan. 2004.

[30] M. Grohe and D. Marx, “Constraint solving via fractional edge covers,” ACM Trans. Algorithms, vol. 11, no. 1, pp. 1–20, Oct. 2014.

[31] G. Greco and F. Scarcello, “The power of local consistency in conjunctive queries and constraint satisfaction problems,” SIAM J. Comput., vol. 46, no. 3, pp. 1111–1145, Jan. 2017.

[32] F. Scarcello, “From hypertree width to submodular width and data-dependent structural decompositions,” in Proc. SEBD, 2018, pp. 1–8.

[33] P. Barceló, L. Libkin, and M. Romero, “Efficient approximations of conjunctive queries,” SIAM J. Comput., vol. 43, no. 3, pp. 1085–1130, Jan. 2014.
MAHMOUD M. ABD ELLATIF is currently a Professor of information systems with the Faculty of Computers and Information, Helwan University, Egypt. He is also a Contract with the College of Business, University of Jeddah, Saudi Arabia. His research interests include information systems focus on using data mining techniques and semantic web technologies to apply PIECES framework on software projects.

MONA ABBASS received the bachelor’s and master’s degrees in computer science. She is currently pursuing the Ph.D. degree with the Department of Mathematics, Faculty of Science, Ain Shams University. She works as a mathematics and statistics specialist at the Central Lab for Agricultural Experts Systems, Agricultural Research Centre, Egypt. Her research interests include databases, image processing, and graph theory.