Caspian Journal of Mathematical Sciences (CJMS)
University of Mazandaran, Iran
http://cjms.journals.umz.ac.ir
https://doi.org/10.22080/cjms.2025.28230.1735
Caspian J Math Sci. **15**(1)(2026), 30-49 (Research Article)

Switching Hypergraphs-Based Decision Making and Python Code

Mohammad Hamidi[1], Marzieh Rahmati [2] and Seyyed Ali Mohammadiyeh [3]
[1,2] Department of Mathematics, University of Payame Noor, Tehran, Iran.
[3] Department of Pure Mathematics, faculty of Mathematical Sciences, University of Kashan, Kashan 87317-53153, I. R.

Abstract. The concept of binary decision trees plays a wide role in applications such as computer-aided design. In this paper, we present the binary decision hypertree, considering the total binary truth table ($T.B.T$) and the new concepts of hypergraphable Boolean functions and Boolean functionable hypergraphs. Eventually, we exhibit an Algorithm and Python programming (by perfect and basic codes) that for each certain $T.B.T$, we inset a minimum irreducible switching phrase and so we obtain the binary decision hypertrees. The major contribution of this work is to introduce the novel and simple method to the design of reduced-ordered binary decision diagrams via the hypergraph(tree) for the first time in this paper.

Keywords: Hypergraphable Boolean function, Boolean functionable hypergraph, binary decision (hyper)tree.

2000 Mathematics subject classification: 05C65, 94C10, 05C05.

[1]Corresponding author: m.hamidi@pnu.ac.ir
Received: 19 December 2024
Revised: 12 February 2025
Accepted: 07 December 2025
How to Cite: Hamidi, Mohammad; Rahmati, Marzieh; Mohammadiyeh, Seyyed Ali. Switching Hypergraphs-Based Decision Making and Python Code. Casp.J. Math. Sci.,**15**(1)(2026), 30-49.

1. Introduction

Berge first introduced the new notion of hypergraph in 1960. Since hypergraphs are extensions of graphs, they are a useful and efficient tool for analyzing graph structure and extending classical graph results, which have applications in all branches of science, such as medicine and hypernetworks [12, 13, 14, 15, 16]. Research shows that hypergraphs play a prominent role in computer science, especially machine learning and relational databases. Documentation on the use of hypergraphs in social networks, web information systems, document-centered information processing, information systems is available in these sources [2, 5, 6, 7, 10, 11]. Symbolic logic was first introduced in 1847 by the English mathematician George Boole. In his research work, he introduced Boolean algebra, which is a combination of classical logic and algebra, which is now called modern algebra (complete distribution lattice). The language designed by Boole is used to determine the truth or falsehood of logical expressions using variables and symbols. We represent the Boolean operators *or, and, not* with the symbols $+, *, -$ and the *true* and *false* expressions with variables. Boolean algebra usages 1 and 0 to show the truth and falsehood of a statement, instead of T and f in truth tables. Logical computations are expressed by a function called a Boolean function to represent the relationship between a Boolean input and a Boolean output. This function plays an important role in the building of digital computer circuits and chips for cryptography, especially in the design of symmetric key algorithms. In 1937, Claude Shannon used Boolean algebra for electronic switching circuits, and by the 1950s it had become a standard part of the electronic project [4].

In the present essay, we inset the new meaning of hypergraphable Boolean function and Boolean functionable hypergraph for any certain T.B.T. We also create a hyperdiagram of each Boolean function, and check the qualifications under which a hypergraph is according to a specified Boolean function. In fact, for every desired hypergraph, we derive a Boolean function called a Boolean functional hypergraph. We show that any hypergraph is a Boolean functional hypergraph. A normal question arises, is there a Boolean expression for each specific T.B.T? The principal motive of the authors of the article is the elicitation of an irreducible switching phrase of each T.B.T. Thus we introduce the concepts of hypergraph based on Boolean functions and unitors sets. Finally, we use the aforementioned notions to demonstrate that any T.B.T corresponds to a minimum Boolean phrase through the set of unitors and provide qualifications in T.B.T to gain the minimum irreducible Boolean phrase from the switching functions. The output of

this paper is a new method to extract binary decision (hyper)diagrams and binary decision (hyper)trees via the notion of Boolean functionable hypergraphs and hypergraphable Boolean functions.

2. Preliminaries

In the present section, we remind the definitions and outcomes that we require further.

Suppose X be an optional collection. Thus we show $P^*(X) = P(X) \setminus \emptyset$, wherever $P(X)$ is the poset of X.

Definition 2.1. [2] Let $G = \{a_1, a_2, \ldots, a_n\}$ be a finite set. A *hypergraph* on G is a pair $H = (G, \{E_i\}_{i=1}^m)$ such that for all $1 \leq i \leq m$, $\emptyset \neq E_i \subseteq G$ and $\bigcup_{i=1}^{m} E_i = G$. The elements a_i of G are called *hypervertices* and the sets E_i are called the (*hyperedges*) of the hypergraph H.

Definition 2.2. [5] Let $G = \{a_1, a_2, \ldots, a_n\}$ be a finite set. A *hyperdiagram* on G is a pair $H = (G, \{E_k\}_{k=1}^m)$ such that for all $1 \leq k \leq m$, $E_k \subseteq G$ and $|E_k| \geq 1$. Clearly every hypergraph is a hyperdiagram, while the converse is not necessarily true. We say that two hyperdiagrams $H = (G, \{E_k\}_{k=1}^m)$ and $H' = (G', \{E'_k\}_{k=1}^{m'})$ are isomorphic if $m = m'$ and there exists a bijection $\gamma : G \to G'$ and a permutation $\tau : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m'\}$ such that for all $a, b \in G$, if for some $1 \leq i \leq m$, $a, b \in E_i$, then $\gamma(a), \gamma(b) \in E_{\tau(i)}$, if for all $1 \leq i \leq m$, $a, b \notin E_i$, then $\gamma(a), \gamma(b) \notin E_{\tau(i)}$ and if for some $1 \leq i \leq m$, $a \in E_i$, for all $1 \leq j \leq m, b \notin E_j$, then $\gamma(a) \in E_{\tau(i)}$ and $\gamma(b) \notin E_j$. Since every hypergraph is a hyperdiagram, define an isomorphic hypergraphs in a similar a way.

Definition 2.3. [3] A binary decision tree (BDT) for the variable order $x_{i_1} < x_{i_2} < \ldots < x_{i_n}$ satisfying the following conditions: (1) all leaves are labeled by 0 or by 1, (2) all other nodes are labeled by a variable and have exactly two children, the 0-child and the 1-child. The edges leading to these children are labeled by 0 respectively by 1, (3) if the root is not leave, then it is labeled by x_{i_1}, (4) if a node is labeled by x_{i_n} then, its two children are leaves, (5) if a node is labeled by x_{i_j} and $j < n$, then its two children are labeled by $x_{i_{j+1}}$. Every path of a decision tree determines an assignment of the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$. The Boolean function $f(x_1, x_2, \ldots, x_n)$ represented by a decision tree T is defined by as $f(x_1, x_2, \ldots, x_n) =$ label of the leaf reached by the path corresponding to the assignment $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$.

Definition 2.4. [3] A binary decision diagram (BDD) for a given variable order is an acyclic directed graph satisfying the following properties: (1) there is exactly one node without predecessors (the root) ,(2) there is one or two nodes without successors, labeled by 0 or 1 (if there are two then they carry different labels), (3) all other nodes are labeled by a variable and have exactly two distinct children, the 0-child and the 1-child, (the edges leading to these children are labeled by 0 respectively by 1), (4) a child of a node is labeled by 0, by 1, or by a variable larger than the label of its parent w.r.t. the variable order, (5) all descendant-closed subgraphs of the graph are non-isomorphic. A binary decision diagram (BDD) is a compact representation of a binary decision tree, which obtained from a decision tree through repeated application of two compression rules

(i) Sharing of identical subtrees.
(ii) Elimination of nodes for which the 0-child and the 1-child coincide (redundant nodes).

The rules are applied until all subtrees are different and there are no redundant nodes.

Theorem 2.5. [9] *Each T.B.T it matches to a least Boolean phrase.*

Definition 2.6. [9] Let $n \in \mathbb{N}$ and $\mathcal{C}(\varphi, a_1, a_2, \ldots, a_n)$ be a T.B.T. Then $\forall 1 \leq j \leq 2^n$ describe $Kernel(\varphi_j) = \{(a_1, a_2, \ldots, a_n) \mid \varphi_j(a_1, a_2, \ldots, a_n) = 0\}$ and will mark via $Ker(\varphi_j)$, in a analogous method $Kernel(\varphi)$ is described and it is marked by $Ker(\varphi)$.

Definition 2.7. [9] Let $n \in \mathbb{N}, m \in \mathbb{N}^*$, $1 \leq k \leq n$ and $\mathcal{C}(\varphi^{(0)}, \ldots, \varphi^{(m)}, a_1, \ldots, a_n)$ be a T.B.T, which for each $0 \leq t \leq m$, $\varphi^{(t)}(a_1, \ldots, a_n) = \prod_{i=1}^{2^n} \varphi_i^{(t)}(a_1, a_2, \ldots, a_n)$. Then

(i) $Z(n, \varphi^{(t)}, 0) = \{j \mid \varphi_j^{(t)}(a_1, a_2, \ldots, a_n) = 0, \text{ where } 1 \leq j \leq 2^n\}$;

(ii) $S(k, a_1, a_2, \ldots, a_k, 0) = \{\sum_{i=1}^{n} \overline{a_i} \mid \sum_{i=1}^{k} a_i + \sum_{i=k+1}^{n} \overline{a_i} = 0\}$.

Theorem 2.8. [9] *Let $n \in \mathbb{N}, 1 \leq j \leq 2^n$ and $\mathcal{C}(\varphi, \phi, a_1, a_2, \ldots, a_n)$ be a T.B.T. Then*

(i) *$Ker(\phi) = Ker(\varphi)$ if and only if $\varphi \sim \phi$;*
(ii) *If $Ker(\varphi) \subseteq Ker(\phi)$, then $(\varphi + \phi) \sim \varphi$;*
(iii) *If $Ker(\phi) \subseteq Ker(\varphi)$, then $(\varphi.\phi) \sim \varphi$;*
(iv) *$Z(n, \varphi.\phi, 0) = Z(n, \varphi, 0)$ if and only if $(\varphi.\phi) \sim \varphi$;*
(v) *$Z(n, \phi, 0) = Z(n, \varphi, 0)$ and $Ker(\phi) \subseteq Ker(\varphi)$ it implies that $\phi \sim \varphi$.*

TABLE 1. T. B. T with n variables $\mathcal{C}(\varphi^{(0)}, \varphi^{(1)}, \ldots, \varphi^{(m)}, a_1, \ldots, a_n)$

a_1	a_2	...	a_n	$\varphi^{(0)}(a_1, \ldots, a_n)$	$\varphi^{(1)}(a_1, \ldots, a_n)$	...	$\varphi^{(m)}(a_1, \ldots, a_n)$
0	0	...	0	$\varphi_1^{(0)}(a_1, \ldots, a_n)$	$\varphi_1^{(1)}(a_1, \ldots, a_n)$	...	$\varphi_1^{(m)}(a_1, \ldots, a_n)$
0	0	...	1	$\varphi_2^{(0)}(a_1, \ldots, a_n)$	$\varphi_2^{(1)}(a_1, \ldots, a_n)$	...	$\varphi_2^{(m)}(a_1, \ldots, a_n)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
0	0	...	1	$\varphi_{2^{n-1}}^{(0)}(a_1, \ldots, a_n)$	$\varphi_{2^{n-1}}^{(1)}(a_1, \ldots, a_n)$	...	$\varphi_{2^{n-1}}^{(m)}(a_1, \ldots, a_n)$
1	1	...	1	$\varphi_{2^n}^{(0)}(a_1, \ldots, a_n)$	$\varphi_{2^n}^{(1)}(a_1, \ldots, a_n)$	...	$\varphi_{2^n}^{(m)}(a_1, \ldots, a_n)$

3. Switching Hypergraph

In this part, we use T.B.T on Boolean variables and present the conception of binary decision hypertree(hyperdiagram) applying the notions of hypergraphable Boolean functions and Boolean functionable hypergraphs and peruse several of their attributes.

We assume each *Boolean function* $\varphi : A_n \to A = \{0, 1\}$ by $\varphi(a_1, a_2, \ldots, a_n) = \prod_{j=1}^{m} \sum_{i=1}^{k_j} \overline{a_i}$, wherever for every $1 \leq i \leq n, \overline{a_i}$ is a *literal* and $m, j, k_j \in \mathbb{N}$. Assume $n \in \mathbb{N}, m \in \mathbb{N}^*$, $a_1, a_2, \ldots, a_n$ be optional Boolean variables and for every $0 \leq j \leq m, \varphi^{(m)}(a_1, \ldots, a_n)$ be Boolean functions. We will define a table T.B.T on Boolean variables $a_1, a_2, \ldots, a_n$ via a collection $\mathcal{C}(\varphi^{(0)}, \varphi^{(1)}, \ldots, \varphi^{(m)}, \overline{a_1}, \ldots, \overline{a_n}) = \{\varphi^{(0)}, \varphi^{(1)}, \ldots, \varphi^{(m)}, (a_1, \ldots, a_n)\}$, where $\forall 0 \leq j \leq m$, $\varphi^{(m)}(a_1, \ldots, a_n)$, are Boolean functions and for $m = 0$, we will denote it by $\mathcal{C}(\varphi, a_1, \ldots, a_n)$. Describe a binary operation *"+"* on $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$ by

$$(\varphi + \psi)(a_1, \ldots, a_n) = \varphi(a_1, \ldots, a_n) + \psi(a_1, \ldots, a_n)$$

a binary operation *"."* on $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$ by

$$(\varphi.\psi)(a_1, \ldots, a_n) = \varphi(a_1, \ldots, a_n).\psi(a_1, \ldots, a_n)$$

and a unary operation

$$c : \mathcal{C}(\varphi, \psi, a_1, \ldots, a_n) \to \mathcal{C}(\varphi, \psi, a_1, \ldots, a_n) \text{ by } c(a_i) = 1 - a_i$$

and $c(\varphi(a_1, \ldots, a_n)) = 1 - \varphi(a_1, \ldots, a_n)$. Describe a connection $\sim$ on a T.B.T $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$ by $\varphi \sim \psi$ if for every $(a_1, \ldots, a_n) \in A_n$, we have $\varphi(a_1, \ldots, a_n) = \psi(a_1, \ldots, a_n)(\varphi \equiv \psi)$. It is obvious that $\sim$ is a congruence equivalence connection at $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$. For all $0 \leq j, j' \leq m$, we tell that $\mathcal{C}(\varphi^{(j)}, a_1, \ldots, a_n)$ and $\mathcal{C}'(\varphi^{(j')}, a_1, \ldots, a_n)$ are equivalent, if $\varphi^{(j)} \sim \varphi^{(j')}$.

With Theorem 2.5, Hamidi, et al. demonstrated that $\forall n \in \mathbb{N}$ and $\mathcal{C}(\varphi, a_1, \ldots, a_n)$, there is a least Boolean phrase $\psi(a_1, \ldots, a_n)$ in a manner that $\varphi \sim \psi$. At present, we possess the appendix definition.

Definition 3.1. Let $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ be a $T.B.T$. If $\psi(a_1, \ldots, a_n) = \prod_{j=1}^{m} \sum_{i=1}^{k_j} \overline{a_i}$ be the relevant least Boolean phrase so that $\varphi \sim \psi$. If $\forall\ 1 \leq j \leq m, a_t \in \sum_{i=1}^{k_j} \overline{a_i}$, thus presume a_t like root vertex has two subtrees, one in which $a_t = 0$ and one in which $a_t = 1$. Now each of the two subtrees hits another variable, each of the two variables hits the other two subtrees, and in the same way. The leaves have 0 because the output of the function is in the paths that are from the root to the tabs. We call the acyclic directed graph created in the mentioned qualifications as a binary decision tree and represent by 0-$BDHT$. If we subjoin vertex 1 to the last leaf as a symmetrical subtree for the available subtree, we get an acyclic-directed graph marked via $(0 \ltimes 1)$-$BDHT$.

Example 3.2. Suppose $A = \{x, y, z, x', y', z'\}$ and consider a T.B.T, $\mathcal{C}(\varphi, x, y, z)$ in Table 2.

x	y	z	$\varphi(x, y, z)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

TABLE 2. T. B. T $\mathcal{C}(\varphi, x, y, z)$

We achieve an undirected hypergraph $\mathcal{A}' = (A, E_1, E_2, E_3, E_4)$, where $E_1 = \{x, y', z\}, E_2 = \{x, y', z'\}, E_3 = \{x', y, z\}$ and $E_4 = \{x', y, z'\}$. Since $E_1 \cap E_2 = \{x, y'\}, E_1 \cap E_3 = \{z\}, E_1 \cap E_4 = \emptyset, E_2 \cap E_3 = \emptyset, E_2 \cap E_4 = \{z'\}$ and $E_3 \cap E_4 = \{x', y\}$. We get that minimum Boolean expression $\psi(x, y, z) = (x + y')(x' + y)$. Thus the 0-$BDHT$ and $(0 \ltimes 1)$-$BDHT$ are obtained in Figures 1 and 2.

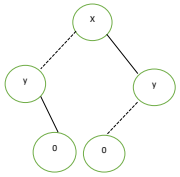

FIGURE 1. 0-$BDHT$

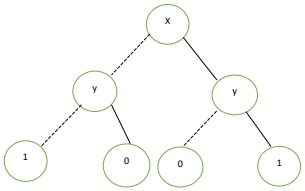

FIGURE 2. $(0 \ltimes 1)$-$BDHT$.

In the below Theorem, we demonstrate that each T.B.T compliant to a binary decision hypertree.

Theorem 3.3. *Suppose* $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ *be a T.B.T. Then* $(0 \ltimes 1)$-$BDHT$ *is compliant to T.B.T.*

Proof. It is obvious via Theorem 2.5. □

After this, for every $T.B.T$, $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$, we will display the $BDHT$ of $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ by $BDHT(\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n))$ or $BDHT(\mathcal{C}(\varphi)) = (V, E, \longrightarrow, \dashrightarrow)$, wherever $\longrightarrow$ is a strong directed line and $\dashrightarrow$ is dotted directed line in its graph. for every two certain vertices a, b in digraph $BDHT(\mathcal{C}(\varphi)) = (V, E, \rightarrow, \dashrightarrow)$, we will display $(a \rightarrow b)$ or $(a \dashrightarrow b)$ via edges of digraph $BDHT(\mathcal{C}(\varphi)) = (V, E, \rightarrow, \dashrightarrow)$ and will tall that are neighbor vertices if, $(a \rightarrow b)$ or $(a \dashrightarrow b)$. The below lemma is a useful gadget for the calculations of our method, which we demonstrate as follows.

Lemma 3.4. *Assume* $\varphi_1, \varphi_2, \ldots, \varphi_n$ *be Boolean functions. Thus*

(i) $(\varphi_1.\varphi_2) \sim \big(\varphi_1.(c(\varphi_1) + \varphi_2)\big)$.

(ii) $(\varphi_1.\varphi_2.\ldots.\varphi_n) \sim \big(\varphi_1.(c(\varphi_1) + \varphi_2.\ldots.\varphi_n)\big)$.

Proof. (i)

$(\varphi_1.\varphi_2)(a_1, \ldots, a_n) = 1.1 = 1$, *then* $(\varphi_1.(c(\varphi_1) + \varphi_2))(a_1, \ldots, a_n)$

$= \quad 1.(0 + 1) = 1.$

$(\varphi_1.\varphi_2)(a_1, \ldots, a_n) = 1.0 = 0$, *then* $(\varphi_1.(c(\varphi_1) + \varphi_2))(a_1, \ldots, a_n)$
$= 1.(0 + 0) = 0.$

$(\varphi_1.\varphi_2)(a_1, \ldots, a_n) = 0.1 = 0$, *then* $(\varphi_1.(c(\varphi_1) + \varphi_2))(a_1, \ldots, a_n)$
$= 0.(1 + 1) = 0.$

$(\varphi_1.\varphi_2)(a_1, \ldots, a_n) = 0.0 = 0$, *then* $(\varphi_1.(c(\varphi_1) + \varphi_2))(a_1, \ldots, a_n)$
$= 0.(1 + 0) = 0.$

(ii) Via item (i) and induction on n is get:

$$\begin{aligned}
&(\varphi_1.\varphi_2.\ldots.\varphi_l.\varphi_{(l+1)})\\
&\sim (\varphi_1.(c(\varphi_1)+\varphi_2.\ldots.\varphi_l).\varphi_{(l+1)})\\
&\sim (\varphi_1.(c(\varphi_1)+\varphi_2.\ldots.\varphi_l).(c(\varphi_1)+\varphi_{(l+1)}))\\
&\sim (\varphi_1.(c(\varphi_1)+c(\varphi_1)(\varphi_2.\ldots.\varphi_l+\varphi_{(l+1)})+\varphi_2.\ldots.\varphi_l.\varphi_{(l+1)}))\\
&\sim (\varphi_1.(c(\varphi_1)(1+\varphi_2.\ldots.\varphi_l+\varphi_{(l+1)})+\varphi_2.\ldots.\varphi_l.\varphi_{(l+1)}))\\
&\sim (\varphi_1.(c(\varphi_1)+\varphi_2.\ldots.\varphi_l.\varphi_{(l+1)})).
\end{aligned}$$

□

Example 3.5. Consider a T.B.T, $\mathcal{C}(\varphi, x, y, z)$ as shown in Table 3.

x	y	z	$\varphi(x,y,z)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

TABLE 3. T. B. T $\mathcal{C}(\varphi, x, y, z)$

Suppose $A = \{x, y, z, x', y', z'\}$. Consider the undirected hypergraph $\mathcal{A}' = (A, E_1, E_2, E_3, E_4)$ in Figure 3, where

$$E_1 = \{x, y, z'\}, E_2 = \{x, y', z\}, E_3 = \{x', y, z\} \; and E_4 = \{x', y, z'\}.$$

Since $E_1 \cap E_2 = \{x\}, E_1 \cap E_3 = \{y\}, E_1 \cap E_4 = \{y, z'\}, E_2 \cap E_3 = \{z\}, E_2 \cap E_4 = \emptyset$ and $E_3 \cap E_4 = \{x', y\}$, we get that minimum Boolean expression $\psi(x, y, z) = (x' + y)(x + y' + z)(y + z')$. Because the binary decision hypertree is not drawable, by Lemma 3.4 (i), we obtain $\psi(x, y, z) = (x' + y)(x + y' + z)(x + y + z')$. Thus the 0-$BDHT$ and $(0 \ltimes 1)$-$BDHT$ or $BDHT$ are obtained in Figures 4 and 5.

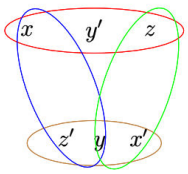

FIGURE 3. Hypergraph

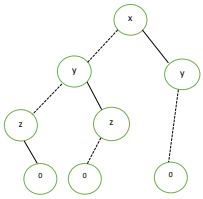

FIGURE 4. 0-$BDHT$

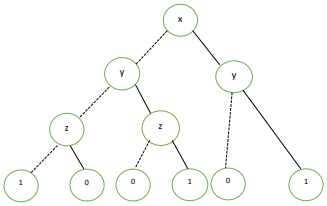

FIGURE 5. $BDHT$

Definition 3.6. Let $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ be a $T.B.T$ and $BDHT(\mathcal{C}(\varphi)) = (V, E, \rightarrow, \dashrightarrow)$, be a binary decision hypertree of $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$. We remove the last level that outputs $0, 1$, replace the two nodes as $0, 1$, and drain a tree such that all subtrees participate $0, 1$. We will tell as a binary decision hyperdiagram and will mark via $BDHD$. If eliminate 0 node in last level(output), will denote by 1-$BDHD$ and eliminate 1 node in last level(output), will denote by 0-$BDHD$.

Example 3.7. Observe a T.B.T, $\mathcal{C}(\varphi, x, y, z)$ in Table 3. The 0-$BDHD$ and $BDHD$ are obtained in Figures 6 and 7.

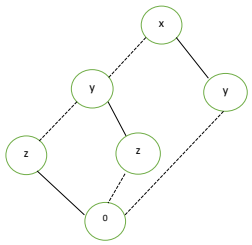

FIGURE 6. 0-$BDHD$

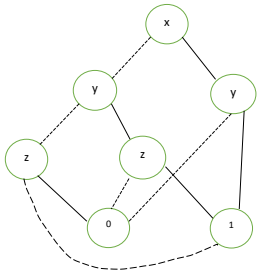

FIGURE 7. $BDHD$

Theorem 3.8. *Suppose* $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ *be a* $T.B.T$. *Then*

(i) 0-$BDHD$ *is constructed from* 1-$BDHD$;
(ii) 1-$BDHD$ *is constructed from* 0-$BDHD$.

Proof. (i) If we have 1-$BDHD$ because each vertex has two subtrees, for vertices that have only one subtree, we draw the other subtree and end it with 0 node. In this case, we have $BDHD$, which by eliminating 1 node in last level(output), 0-$BDHD$ is obtained.
(ii) Unlike the previous case, if we have 0-$BDHD$ because each vertex has two subtrees, for vertices that have only one subtree, we draw the other subtree and end it with 1 node. In this case, we have $BDHD$, which by eliminating 0 node in last level(output), 1-$BDHD$ is obtained. □

Theorem 3.9. *Suppose* $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ *be a* $T.B.T$. *Then* $BDHD$ *is isomorphic to* BDD.

Proof. Let $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ be a $T.B.T$ and $\psi(a_1, \ldots, a_n) = \prod_{j=1}^{m} \sum_{i=1}^{k_j} \overline{a_i}$ be the relevant minimum Boolean phrase so that $\varphi \sim \psi$. We are informed that $BDHD$ is the binary decision hyperdiagram corresponding to the Boolean phrase ψ and BDD is the binary decision diagram corresponding to the Boolean expression φ. As regards $\varphi \sim \psi$, $\forall 1 \leq j \leq 2^n$, $\varphi_j(a_1, \ldots, a_n) = \psi_j(a_1, \ldots, a_n)$. Right now describe a bijection $\gamma : V \to V'$ by $\gamma(a_i) = a_i$ and $\forall 1 \leq i \leq m, E'_i = \gamma(E_i)$. obviously if two nodes a, b are adjacent nodes in V, then $\gamma(a), \gamma(b)$ are adjacent in V', we obtain $BDHD \cong BDD$. □

Corollary 3.10. *Assume $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$ be a $T.B.T$. Then we attain the below chart HT in Figure 8.*

Proof. According to Definition 3.6, each $BDHD$ is constructed from $BDHT$, and each 0-$BDHD$ is constructed from 0-$BDHT$. Hence by Theorem 3.9, $BDHD \cong BDD$. □

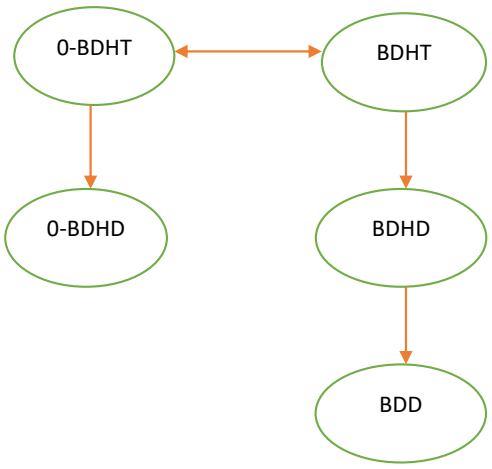

FIGURE 8. Tree chart HT of $\mathcal{C}(\varphi \not\equiv 0, a_1, \ldots, a_n)$

In Algorithm 1, the way of making a Boolean expression from a T.B.T based on [9] is described.

Algorithm 1: Start:
1. Enter a T.B.T $\mathcal{C}(\varphi, a_1, \ldots, a_n)$.
2. If $\varphi \equiv 0$ or $\varphi \equiv 1$, then put $\psi \equiv 0$ or $\psi \equiv 1$, In orderly.
3. If $\exists j \in \{1 \leq j_1, j_2, \ldots, j_s \leq 2^n\}$, so that $\varphi_j(a_1, \ldots, a_n) = 0$, then put $E_j = \{\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n} \mid \sum_{j=1}^{n} \overline{a_j} = \varphi_j(a_1, \ldots, a_n) = 0\}$.

Regarding stage 3, note below:

4. $\forall 1 \leq i \neq j \leq k$ and $k \in \mathbb{N}$, put $\varphi_{ij} = E_i \cap E_j$.
5. If $\forall 1 \leq i \neq j \leq k, \varphi_{ij} = \emptyset$ or $|\varphi_{ij}| < n-1$, then put $\psi(a_1, \ldots, a_n) = \prod_{1 \leq i \leq k} \sum_{\alpha \in E_i} \alpha$.
6. If $\exists 1 \leq r \leq k$, and $i \neq j \in \{i_1, i_2, \ldots, i_r\}$ so that $\varphi_{ij} \neq \emptyset$ and $|\varphi_{ij}| \geq n-1$, then place $\psi_{ij}(a_1, \ldots, a_n) = \sum_{\alpha \in f_{ij}} \alpha$.
7. If $Z(n, \prod_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} \psi_{ij}, 0) = Z(n, \varphi, 0)$, then place $\psi(a_1, \ldots, a_n) = \prod_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} \psi_{ij}(a_1, \ldots, a_n)$ that $Ker(\psi_{ij}) \subseteq Ker(\varphi)$.
8. If $Z(n, \prod_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} \psi_{ij}, 0) < Z(n, \varphi, 0)$, then consider $1 \leq s \leq k$, $j \in \{j_1, j_2, \ldots, j_s\}$, $\varphi_j(a_1, \ldots, a_n) = \sum_{\beta \in E_j} \beta$ as $Ker(\prod_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} \psi_{ij}) \neq Ker(\varphi_j) \subseteq Ker(\varphi)$ and $Z(n, (\prod_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} \psi_{ij}).(\prod_{j=j_1}^{j_s} \varphi_j), 0) = Z(n, \varphi, 0)$.

End.

Definition 3.11. Let $n \in \mathbb{N}$ and $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$ be a T.B.T. We tell that $BDD(\mathcal{C}(\varphi)) = (V, E, \rightarrow, \dashrightarrow)$ is isomorphic to $BDD(\mathcal{C}(\psi)) = (V', E', \rightarrow, \dashrightarrow)$, if there is a bijection $\gamma : V \rightarrow V'$ so that if two nodes a, b are adjacent nodes in $BDD(\mathcal{C}(\varphi))$, then $\gamma(a), \gamma(b)$ are adjacent in $BDD(\mathcal{C}(\psi))$, $\overrightarrow{\gamma(a, b)}$ or $(\gamma(a) \dashrightarrow \gamma(b))$ are.

Theorem 3.12. *Assume $n \in \mathbb{N}$ and $\mathcal{C}(\varphi, \psi, a_1, \ldots, a_n)$ is a T.B.T. Then*

(*i*) *if $\varphi \sim \psi$, then $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\psi))$;*
(*ii*) *if $\varphi \sim \psi$, then $BDT(\mathcal{C}(\varphi)) \cong BDT(\mathcal{C}(\psi))$.*

Proof. (*i*) Because $\varphi \sim \psi, \forall 1 \leq j \leq 2, \varphi_j(a_1, \ldots, a_n) = \psi_j(a_1, \ldots, a_n)$. Now describe a bijection $\gamma : V \rightarrow V'$ by $\gamma(x_i) = x_i$ and $\forall 1 \leq i \leq m$, $E'_i = \gamma(E_i)$. Obviously if two nodes x, y are adjacent nodes in V, then $\gamma(x), \gamma(y)$ are adjacent nodes in V', we gain that $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\psi))$.

(*ii*) It is alike to the case (*i*).

□

Based on the Theorem 3.12 and Theorem 2.8, we catch the below corollary.

Corollary 3.13. *Let $\mathcal{C}(\varphi, \varphi', a_1, \ldots, a_n)$ be a T.B.T.*

(i) If $Ker(\varphi') = Ker(\varphi)$, then $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\varphi'))$;
(ii) If $Ker(\varphi) \subseteq Ker(\varphi')$, then $BDD(\mathcal{C}(\varphi + \varphi')) \cong BDD(\mathcal{T}(\varphi))$;
(iii) If $Ker(\varphi') \subseteq Ker(\varphi)$, then $BDD(\mathcal{C}(\varphi.\varphi')) \cong BDD(\mathcal{T}(\varphi))$;
(iv) If $Z(n, \varphi.\varphi', 1) = Z(n, \varphi, 1)$, then $BDD(\mathcal{C}(\varphi.\varphi')) \cong BDD(\mathcal{T}(\varphi))$;
(v) If $Ker(\varphi') \subseteq Ker(\varphi)$ and $Z(n, \varphi', 1) = Z(n, \varphi, 1)$ imply that $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\varphi'))$.

We are demonstrated that for each $T.B.T$, there exists a minimum Boolean phrase distinct of its $c.n.f$. Currently, use Theorem 2.5 and gain the below theorem.

Theorem 3.14. *Suppose $n \in \mathbb{N}$ and $\mathcal{C}(\varphi, a_1, \ldots, a_n)$ be a T.B.T. There exists minimum Boolean phrase in distinct with to its c.n.f as $\psi(a_1, \ldots, a_n)$ so that $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\psi))$.*

Proof. Suppose $n \in \mathbb{N}$ and $\mathcal{C}(\varphi, a_1, \ldots, a_n)$ be a T.B.T. According to Theorem 2.5, there is a minimum Boolean phrase $\psi(a_1, \ldots, a_n)$ so that $\varphi \sim \psi$. By Theorem 3.12, we obtain $BDT(\mathcal{C}(\varphi)) \cong BDT(\mathcal{C}(\psi))$ and therefore $BDD(\mathcal{C}(\varphi)) \cong BDD(\mathcal{C}(\psi))$. □

Corollary 3.15. *Let $n, m \in \mathbb{N}$ and $\mathcal{C}(\varphi^{(0)}, \varphi^{(1)}, \ldots, \varphi^{(m)}, (a_1, \ldots, a_n))$ be a T.B.T. Then $BDD(\mathcal{C}(\mathcal{C}(\varphi^{(0)}, \varphi^{(1)}, \ldots, \varphi^{(m)}, (a_1, \ldots, a_n)))$ is received with Algorithm 2.*

Algorithm 2, Start:
1. Enter a T.B.T $\mathcal{C}(\varphi, a_1, \ldots, a_n)$.
2. Catch the Boolean phrase $\psi(a_1, \ldots, a_n) = \prod_{j=1}^{m} \psi_j$ from Algorithm 1.
3. If there is $1 \leq i \leq n$, so that $\overline{a_i} \in \bigcap_{j=1}^{m} \psi_j$, then Put $a_i =: a_1$ with the title the root node, otherwise (there is no $1 \leq i \leq n$, so that $\overline{a_i} \in \bigcap_{j} \psi_j$) go to stage 6.
4. Consider for a_1, leading dashed line are tagged via 0 and for a_1', leading solid line are tagged via 1.
5. $\forall 2 \leq i \leq n$, place a_i as the subsequent nodes, and for a_i, the leading dashed lines are tagged via 0, and for a_i', leading solid lines are tagged via 1.
6. According to Lemma 3.4, put $\tau(a_1, \ldots, a_n) = \prod_{j=1}^{m'} \tau_j = \psi_1.(\psi_1' + \psi_2 \ldots \psi_m)$, where $m \leq m'$ so there exists $1 \leq i \leq n$, so that $\overline{x_i} \in \bigcap_{j=1}^{m'} \tau_j$ and go to stage 3.

End.

4. Accessible Binary Decision Diagram

In the present part, we render a Python programming to access binary decision hypertree for every certain T.B.T, according to Algorithm 2.

```
import xlrd

sheet = xlrd.open_workbook("input.xlsx").sheet_by_index(0)

element_names = sheet.row_values(0)[:-1]
elements = []
for i in range(1, sheet.nrows):
    elements.append(sheet.row_values(i))

if len(elements) != pow(2, len(element_names)):
    print("Error in input file, rows count isn't equal to 2^n.")
    exit(0)

E_i = []
for i, elm in enumerate(elements):
    if(elm[-1] == 1):  # check if f is equal to 1
        E = []
        for c in range(0, len(elm)-1):
            if(elm[c] == 1):
                E.append(element_names[c])
            else:
                E.append(element_names[c]+"'")
        E_i.append([i, E])

def print_g(g_expr):
    print("g("+",".join(element_names)+") = "+g_expr)

    input("")
    exit(0)

if len(E_i) == len(elements):
    print_g("1")

if len(E_i) == 0:
    print_g("0")

```

```
f_ij = []
for i in E_i:
    f = []
    for j in E_i:
        if(i != j):
            intersection = [x for x in i[1] if x in j[1]]
            if(len(intersection) >= (len(element_names)-1)):
                f.append([j[0], intersection])
    if(f):
        f_ij.append([i[0], f])

if(f_ij == []):
    g = []
    for i in E_i:
        g.append("".join(i[1]))  # multiply elements
    print_g("+".join(g))

g_ij = []
for i in f_ij:
    for j in i[1]:
        g_ij.append("".join(j[1]))  # multiply elements
g_ij = list(dict.fromkeys(g_ij))  # remove duplicates
sigma_g_ij = "+".join(g_ij)


def calc_mult_and(expr):
    expr = expr.split("+")
    for i in expr:
        if "0" not in i:
            return 1
    return 0

true_g_ij = []
for elm in elements:
    tmp_g = sigma_g_ij
    for c in range(0, len(elm)-1):
        if(elm[c] == 1):
            tmp_g = tmp_g.replace(element_names[c]+"'", "0")
            tmp_g = tmp_g.replace(element_names[c], "1")
```

```
76            else:
77                tmp_g = tmp_g.replace(element_names[c]+"'", "1")
78                tmp_g = tmp_g.replace(element_names[c], "0")
79        true_g_ij.append(calc_mult_and(tmp_g))
80
81    f_j = []
82    for i, elm in enumerate(elements):
83        if(elm[-1] == 1 and true_g_ij[i] == 0):
84            f_j.append("".join([x for x in E_i if x[0] == i][0][1]))
85
86    print_g("+".join(g_ij + f_j))
```

In the next example, we prime exploit the binary decision hypertree by hand for the certain T.B.T. Afterwards the binary decision hypertree related to T.B.T is obtained with the help of Python programming (Figure 10). It can be seen that the output of the Python programming and the output of the manual method are the identical.

Example 4.1. Assume $G = \{x, y, z, x', y', z'\}$ and give a $(T.B.T)$ $\mathcal{C}(\varphi, x, y, z)$ as shown in Table 4.

x	y	z	$\varphi(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

TABLE 4. T. B. T $\mathcal{C}(\varphi, x, y, z)$

We achieve an undirected hypergraph $\mathcal{G}' = (G, E_1, E_2, E_3, E_4)$, where $E_1 = \{x, y, z\}, E_2 = \{x, y', z'\}$,
$E_3 = \{x', y, z'\}$ *and* $E_4 = \{x', y', z\}$. Since $E_1 \cap E_2 = \{x\}, E_1 \cap E_3 = \{y\}, E_1 \cap E_4 = \{z\}, E_2 \cap E_3 = \{z'\}, E_2 \cap E_4 = \{y'\}, E_3 \cap E_4 = \{x'\}$, we get that minimum Boolean expression $\psi(x, y, z) = (x + y + z)(x + y' + z')(x' + y + z')(x' + y' + z)$, thus the $(0 \ltimes 1)$-$BDHT(\mathcal{C}(\varphi))$ are gained in Figure 9.

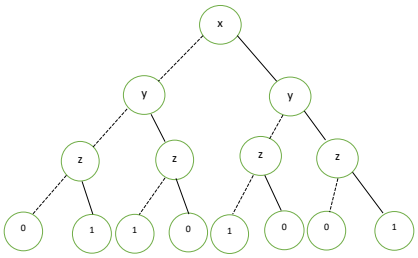

FIGURE 9. $(0 \ltimes 1)$-$BDHT(\mathcal{C}(\varphi))$

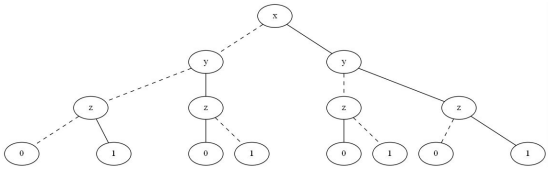

FIGURE 10. $(0 \ltimes 1)$-$BDHT(\mathcal{C}(\varphi))$

5. Application of Binary Decision Hypertree

In the present part, we use Python programming and present a sample of actual-word application of binary decision hypertree.

(a) Car insurance damage: From the point of view of economic sciences, financial violations in the insurance industry are increasingly becoming a serious and ponderable issue. One of the appropriate methods to evaluate and detect financial fraud is the binary decision hypertree. Especially when the speed of decision-making is important, the decision tree provides a convenient schematic view. If we remove the worthless and unusable data, the variables used in this model arethe history of the policyholder, the number of claims, accident plan. Three inputs are needed:
x = history of the policyholder, (1:=history of the policyholder > 25 years, 0 :=history of the policyholder ≤ 25 years), y = number of claims, (1:= number of claims $\in \{2, 3, 4\}$, 0:= number of claims $\in \{0, 1\}$) z = accident plan, (1:=having, 0:=not having). We demand to gain a binary decision supertree so that insurance companies predict all cases (fraudulent and non-fraudulent). If the final nodes of the tree are 0, the file is not fraudulent, and if the final nodes of the tree are 1, the file is fraudulent. Hence, according to the T.B.T 5, and applying Python programming, we gain the binary decision hypertree as displayed in Figure 11.

x	y	z	$\varphi(x,y,z)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

TABLE 5. T. B. T $\mathcal{C}(\varphi, x, y, z)$

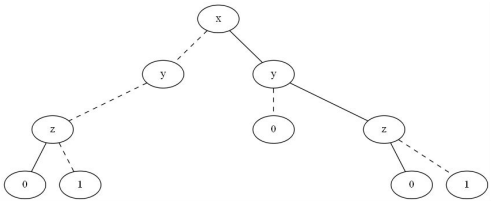

FIGURE 11. $BDHT(\mathcal{C}(\varphi))$

(b) Prediction of chromosomal genetic diseases: Human societies have long faced incurable genetic diseases such as Down syndrome and there was no solution but to tolerate these people because these people face many problems from birth and on the other hand, due to their low intelligence level, they never have the ability to live independently and only bring emotional and financial burdens to their family and society. Therefore, genetic doctors decided to prevent the birth of such babies by diagnosing this disease early in the fetal period and terminating these pregnancies. There are three methods for early prenatal diagnosis: Amniocentesis samplings, chorionic villus sampling, and non-invasive prenatal testing before birth. Using these three methods, they access the fetal DNA and determine whether the fetus has a chromosomal abnormality. Therefore, doctors, based on ultrasound, screening tests, and clinical history, calculate the risk of infection and, based on this risk, divide pregnant women into two groups: low-risk, and high-risk. So we find a binary decision tree that, using factors extracted from screening tests, ultrasounds, and clinical records, is free from common human errors and identifies those at high risk so that fewer people suffer from the exorbitant costs or potential risks of invasive or non-invasive tests. If we consider the ultrasound, screening test, and clinical history as variables x, y, and z, respectively, binary decision tree identifies which mothers

are high risk. So, based on T.B.T 6 and using Python programming, we get the binary decision tree as shown in Figure 12.

x	y	z	$\varphi(x,y,z)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

TABLE 6. T. B. T $\mathcal{C}(\varphi, x, y, z)$

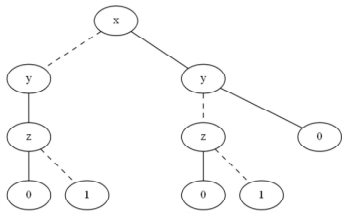

FIGURE 12. $BDHT(\mathcal{C}(\varphi))$.

Remark 5.1. Let n be the number of variables in table, and k be the number of elements of E_j. Then:

(*i*) $T(E_j) = O(n \times 2^n)$.
(*ii*) $T(f_{ij}) = O(k^2 \times n)$.
(*iii*) For any $1 \leq j \leq 2^n$, if $f_j(x_1, ..., x_n) = 0$ or $f_j(x_1, ..., x_n) = 1$, then complication is $O(2n)$ for one loop via lines.
(*iv*) $T(E_j) + T(f_{ij}) = O(n \times (k^2 + 2n))$.

6. Conclusion

In this paper, we presented a novel notion of binary decision hypertree (hyperdiagram) for each T.B.T applying the hypergraph concept and minimum Boolean function. Then, for every T.B.T, we considered the conditions for the isomorphisc binary decision hypertree and binary decision hyperdiagram. The principal prosperity of this reading is a basic documentary to really appraise the foundation of algorithms and codes

of Python programming. In fact, the input of each Python program is a T.B.T and so is a binary decision hypertree. Our procedure is based on proven theorems and Python programming written based on the algorithm presented in the paper, and calculations are performed faster. In next studies, we expect to gain more results regarding decision-making based on hypergraphs, superhypergraphs, and their usages in the real universe.

References

[1] M. Akram and A. Luqman, Certain networks models using single-valued neutrosophic directed hypergraphs, *J. Intell. Fuzzy Syst.* **33** *(2017), 575–588.*

[2] C. Berge, Graphs and Hypergraphs, North Holland, 1979.

[3] RE. Bryant, Binary Decision Diagrams–Handbook of model checking. *Springer. (2018), 191–217.*

[4] J. Eldon Whitesitt, Boolean Algebra and Its Applications. New York Dover Publications, Inc., 1995.

[5] M. Hamidi, A. Borumand saied, Accessible single-valued neutrosophic graphs, *J. Appl. Math. Comput.* **57** *(2018), 121–146.*

[6] M. Hamidi, A. Borumand saied, Achievable Single-Valued Neutrosophic Graphs in Wireless Sensor Networks,*New Math. Nat. Comput.,* **14(2)** *(2018), 157–185.*

[7] M. Hamidi and A. Borumand Saeid, On Derivable Trees,*Trans. Comb.,* **8(2***) (2019), 21–43.*

[8] M. Hamidi, M. Rahmati, A, Rezaei, Switching function based on hypergraphs with algorithm and python programming, *JJ. Intell. Fuzzy Syst.,* **39(3)** *(2020), 2845–02859.*

[9] M. Hamidi, M. Rahmati, A, Rezaei, Accessible Switching function from Hypergraphs with Algoritms, *Algebr. Struct. their Appl,***8(1)** *(2021) 41–60.*

[10] M. Hamidi and f. Smarandache, ,*J. Intell. Fuzzy Syst.,* **37(2)** *(2019) 2869-2885.*

[11] B. Molnar, Applications of Hypergraphs in Informatics a Survey and Opportunities for Research,*Annales Univ. Sci. Budapest., Sect. Comp.***42** *(2014) 261–282.*

[12] H. Rashmanlou, P. Madhumangalb, R. A. Borzooei, F. Mofidnakhaei and S. Biswajite, Product of interval-valued fuzzy graphs and degree,*J. Intell. Fuzzy Syst.,* **35(6)** *(2018), 6443-6451.*

[13] X. Shi, S. Kosari, H. Rashmanlou, S. Broumi and H. Satham, Properties of interval-valued quadripartitioned neutrosophic graphs with real-life application,*J. Intell. Fuzzy Syst.,* **44(5)** *(2023), 7683–7697.*

[14] M. Shoaib, S. Kosari, H. Rashmanlou, MA. Malik, Y. Rao, Y. Talebi and F. Mofidnakhaei, Notion of Complex Pythagorean Fuzzy Graph with Properties and Application,*Mult.-Valued Log. Soft Comput.,* **34(5)** *(2020), 553.*

[15] Z. Shao, S. Kosari, M. Shoaib and H. Rashmanlou, Certain Concepts of Vague Graphs With Applications to Medical Diagnosis,*J. Stat. Phys., a section of the journal Frontiers in Physics.* **8** *(2020), Article 357.*

[16] X. Shi, S. Kosari, A. A. Talebi, H. Sadati and H. Rashmanlou, Investigation of the Main Energies of Picture Fuzzy Graph and its Applications,*nt. J. Comput. Intell. Syst.,* **15** *(2022), 21–43.*