AUT Journal of Mathematics and Computing

AUT J. Math. Com., 5(2) (2024) 117-130
https://doi.org/10.22060/AJMC.2023.21639.1094

Original Article

On binary decision hypertree (hyperdiagram)

Mohammad Hamidi*[a], Marzieh Rahmati[a]

[a]*Department of Mathematics, University of Payame Noor, Tehran, Iran*

ABSTRACT: In computer science, a binary decision diagram is a data structure that is used to represent a Boolean function and to consider a compressed representation of relations. This paper considers the notation of T.B.T (total binary truth table), and introduces a novel concept of binary decision (hyper)tree and binary decision (hyper)diagram, directly and in as little time as possible, unlike previous methods. This study proves that every T.B.T corresponds to a binary decision (hyper)tree via minimum Boolean expression and presents some conditions on any given T.B.T for isomorphic binary decision (hyper)trees. Finally, for faster calculations and more complex functions, we offer an algorithm and so Python programming codes such that for any given T.B.T, it introduces a binary decision (hyper)tree.

Review History:
Received:31 August 2022
Revised:14 December 2022
Accepted:12 January 2023
Available Online:01 April 2024

Keywords:
Minimum Boolean function
Binary decision hypertree
Binary decision hyperdiagram
Unitor

MSC (2020):
05C05; 20F10

1. Introduction

The theory of hypergraphs was invented in 1960 by Berge, who was considered the modern of combinatory and graph theory in this regard and as well as graph theory, has some applications in hypernetworks and all sciences in this scope. Today, some features of hypergraphs are used in complex hypernetwork such as computer science. The reason why hypergraphs seem apt to depict relations in information systems, social networks, and document-centered information processing, web information systems, and computer science, are the relationships among services within a service-oriented architecture [4, 5, 7]. Further materials regarding graphs and hypergraphs are available in the literature too [1, 4, 5, 2, 6]. The theory of Boolean algebra was created in 1847 by the English mathematician George Boole. His combination of ideas from classical logic and algebra resulted in what is called Boolean algebra as modern algebra (a complemented distributive lattice). Boolean algebra has found applications in such diverse areas as anthropology, biology, chemistry, ecology, economics, sociology, and especially computer science, electronic circuit design (gating networks), programming languages, databases, complexity theory, and

*Corresponding author.
E-mail addresses: m.hamidi@pnu.ac.ir, m.rahmati@pnu.ac.ir

complex hypernetworks. In cooperative game theory, monotone Boolean functions are called simple games (voting games); this notion is applied to solve problems in social choice theory [8].

Regarding these points, this paper considers the concept of hypertrees and applies it in computer science. In this regard considers the notation of switching functions and investigates the relation between hypergraphs and switching functions. This paper considers the notation of T.B.T (total binary truth table) and introduces a novel concept of binary decision hypertree and binary decision hyperdiagram. This study proves that every T.B.T corresponds to a binary decision hypertree (hyperdiagram) via minimum Boolean expression and presents some conditions on T.B.T for the isomorphism of binary decision hyperdiagram and binary decision hyperdiagram. The main motivation of this work is extracting binary decision hypertree from a novel method different from previous methods. This method generates the binary decision tree directly and quickly, and that conveyed the importance of our work. Finally, we present an algorithm and so Python programming (with complete and original codes) such that for any given T.B.T, introduces a binary decision hypertree(hyperdiagram) and explanation of this method with two examples of real-life world problems.

2. Preliminaries

In this section, we recall some definitions and results, which we need in what follows. Let X be an arbitrary set. Then we denote $P^*(X) = P(X) \setminus \emptyset$, where $P(X)$ is the power set of X.

Definition 2.1 ([1]). *Let $G = \{x_1, x_2, \ldots, x_n\}$ 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 $x_1, x_2, \ldots, x_n$ of G are called hyper vertices, and the sets $E_1, E_2, \ldots, E_m$ are called the edges (hyperedges) of the hypergraph H.*

Definition 2.2 ([4]). *Let $G = \{x_1, x_2, \ldots, x_n\}$ 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 $\varphi : G \to G'$ and a permutation $\tau : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m'\}$ such that for all $x, y \in G$, if for some $1 \leq i \leq m, x, y \in E_i$, then $\varphi(x), \varphi(y) \in E_{\tau(i)}$, if for all $1 \leq i \leq m, x, y \notin E_i$, then $\varphi(x), \varphi(y) \notin E_{\tau(i)}$ and if for some $1 \leq i \leq m, x \in E_i$, for all $1 \leq j \leq m, y \notin E_j$, then $\varphi(x) \in E_{\tau(i)}$ and $\varphi(y) \notin E_j$. Since every hypergraph is a hyperdiagram, define isomorphic hypergraphs in a similar way.*

Theorem 2.3. *[3] Let $n \in \mathbb{N}$.*

(i) Then every (T.B.T) $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ corresponds to a hypergraph.

(ii) Every T.B.T corresponds to a minimum Boolean expression.

(iii) Let $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$ be a T.B.T. If f is a hypergraphable Boolean function, then $(f^{\nearrow})^{\swarrow} \sim f$.

(iv) Let $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$ be a T.B.T. If $f(x_1, x_2, \ldots, x_n)$ is a d.n.f and $m \geq 1+2^{n-1}$, then it is a hypergraphable Boolean function.

(v) Let $0 \leq j, j' \leq m$. If $\mathcal{T}(f^{(j)}, x_1, \ldots, x_n)$ and $\mathcal{T}'(f^{(j')}, x_1, \ldots, x_n)$ are equivalent, then their Boolean function–based hypergraph are isomorphic.

Definition 2.4 ([3]). *Let $n \in \mathbb{N}$ and $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$ be a T.B.T. For all $1 \leq j \leq 2^n$ define $Unitor(f_j) = \{(x_1, x_2, \ldots, x_n) \mid f_j(x_1, x_2, \ldots, x_n) = 1\}$ and will denote by $Un(f_j)$, in a similar way $Unitor(f)$ is defined and it is denoted by $Un(f)$.*

Definition 2.5 ([3]). *Let $n \in \mathbb{N}, m \in \mathbb{N}^*$, $1 \leq k \leq n$ and $\mathcal{T}(f^{(0)}, \ldots, f^{(m)}, x_1, \ldots, x_n)$ be a T.B.T, where for $0 \leq t \leq m, f^{(t)}(x_1, \ldots, x_n) = \sum_{i=1}^{2^n} f_i^{(t)}(x_1, x_2, \ldots, x_n)$. Then*

(i) $I(n, f^{(t)}, 1) = \{j \mid f_j^{(t)}(x_1, x_2, \ldots, x_n) = 1,$ where $1 \leq j \leq 2^n\}$;

(ii) $P(k, x_1, x_2, \ldots, x_k, 1) = \{\prod_{i=1}^{n} \overline{x_i} \mid (\prod_{i=1}^{k} x_i)(\prod_{i=k+1}^{n} \overline{x_i}) = 1\}$.

Corollary 2.6 ([3]). *Let $n \in \mathbb{N}$ and $\mathcal{T}(f, g, x_1, x_2, \ldots, x_n)$ be a T.B.T. Then*

(*i*) *if* $Un(g) \subseteq Un(f)$, *then* $(f+g) \sim f$;

(*ii*) $(f+g) \sim f$ *if and only if* $I(n, f+g, 1) = I(n, f, 1)$.

Theorem 2.7 ([3]). *Let* $n \in \mathbb{N}, 1 \leq j \leq n$ *and* $\mathcal{T}(f, g, x_1, x_2, \ldots, x_n)$ *be a T.B.T and* $g \sim \prod_{j=1}^{k} x_{i_j}$, *where* $1 \leq i_1, i_2, \ldots, i_k \leq n$.

(*i*) *If* $I(n, f, 1) < 2^{n-1}$, *then* $(f+g) \not\sim f$;

(*ii*) *If* $I(n, g, 1) = I(n, f, 1)$ *and* $Un(g) \subseteq Un(f)$ *imply that* $g \sim f$.

3. Switching Hypergraph

In this section, we apply the notation of total binary truth table (T.B.T) on Boolean variables and introduce the concept of binary decision hypertree(hyperdiagram) using the concepts of hypergraphable Boolean functions and Boolean functionable hypergraphs and investigate some of their properties.

By Theorem 2.3, Hamidi, et al. proved that for any given $n \in \mathbb{N}$ and $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$, there exists a minimum Boolean expression $h(x_1, x_2, \ldots, x_n)$ in such a way that $f \sim h$. Now, we have the following definition.

Definition 3.1. *Let* $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ *be a T.B.T and* $h(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{m} \prod_{i=1}^{k_j} \overline{x_i}$ *be the related minimum Boolean expression in such a way that* $f \sim h$ *and* $\forall\ 1 \leq j \leq m, x_t \in \prod_{i=1}^{k_j} \overline{x_i}$. *Then consider* x_t *as the root node and so have two subtrees, one for the case where* $x_t = 0$ *and one where* $x_t = 1$. *Each of the two subtrees is now testing another variable, each with another two subtrees, and so on. At the leaves, we have* 1, *which is the output of the function on the inputs that constitute the path from the root to the leaf. The obtained acyclic-directed graph that satisfies in above conditions, will call as a binary decision hypertree and will denote by* 1-$BDHT(\mathcal{T}(f))$. *If add* 0 *the node in the last level(output) as a complementary and symmetric subtree with the existing subtree, we obtain an acyclic-directed graph that will denote by* $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$.

In the following Theorem, we prove that every T.B.T corresponds to a binary decision hypertree.

Theorem 3.2. *Let* $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ *be a T.B.T. Then* $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$ *is equivalent to T.B.T.*

Proof. Let $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ be a *T.B.T*. By Theorem 2.3, Every T.B.T corresponds to a minimum Boolean expression, there exists a minimum Boolean expression $h(x_1, x_2, \ldots, x_n)$ in such a way that $f \sim h$. If $h(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{m} \prod_{i=1}^{k_j} \overline{x_i}$ be the related minimum Boolean expression in such a way that $f \sim h$. By definition 3.1, we obtain an acyclic directed graph that will denote by $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$. □

From now on, for a given $(T.B.T)$, $\mathcal{T}(f \not\equiv 0, x_1, \ldots, x_n)$, we will show the $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f \not\equiv 0, x_1, \ldots, x_n)) = (V, E, \longrightarrow, \dashrightarrow)$, where $\longrightarrow$ is a solid directed line and $\dashrightarrow$ is a dashed directed line in its graph. For any given two vertices x, y in a directed graph $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f)) = (V, E, \longrightarrow, \dashrightarrow)$, we will denote $(\overrightarrow{x, y})$ or $(\overdashrightarrow{x, y})$ by edges of a directed graph $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f)) = (V, E, \longrightarrow, \dashrightarrow)$ and will say that are adjacent vertices if, $(\overrightarrow{x, y})$ or $(\overdashrightarrow{x, y})$. Following lemma is a helpful tool in the computations of our works that we present and prove as follow.

Lemma 3.3. *Let* $f_1, f_2, \ldots, f_n$ *be Boolean functions. Then*

(*i*) $(f_1 + f_2) \sim \big(f_1 + c(f_1).f_2\big)$.

(*ii*) $(f_1 + f_2 + \cdots + f_n) \sim \big(f_1 + c(f_1).(f_2 + \cdots + f_n)\big)$.

Proof. Are clear by definition. □

Example 3.1. *Let* $H = \{x_1, x_2, x_3, x_1', x_2', x_3'\}$ *and consider a* $(T.B.T)$, $\mathcal{T}(f, x_1, x_2, x_3)$ *in Table 1.*

By Theorem 2.3, we get an undirected hypergraph $\mathcal{H}' = (H, E_1, E_2, E_3, E_4)$ *in Figure 1, where* $E_1 = \{x_1', x_2', x_3'\}, E_2 = \{x_1', x_2, x_3\}, E_3 = \{x_1, x_2, x_3'\}$ *and* $E_4 = \{x_1, x_2, x_3\}$. *Since*

$$E_1 \cap E_2 = \{x_1'\}, E_1 \cap E_3 = \{x_3'\}, E_1 \cap E_4 = \emptyset, E_2 \cap E_3 = \{x_2\}, E_2 \cap E_4 = \{x_2, x_3\} \text{ and } E_3 \cap E_4 = \{x_1, x_2\}, \quad (1)$$

Table 1: T. B. T with 3 variables $\mathcal{T}(f, x_1, x_2, x_3)$

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	$f_1(x_1, x_2, x_3) = 1$
0	0	1	$f_2(x_1, x_2, x_3) = 0$
0	1	0	$f_3(x_1, x_2, x_3) = 0$
0	1	1	$f_4(x_1, x_2, x_3) = 1$
1	0	0	$f_5(x_1, x_2, x_3) = 0$
1	0	1	$f_6(x_1, x_2, x_3) = 0$
1	1	0	$f_7(x_1, x_2, x_3) = 1$
1	1	1	$f_8(x_1, x_2, x_3) = 1$

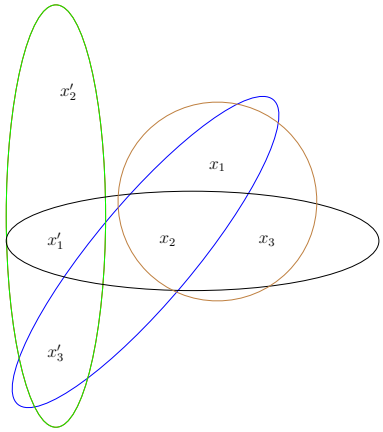

Figure 1: Hypergraph corresponding to $T.B.T$ Table 1.

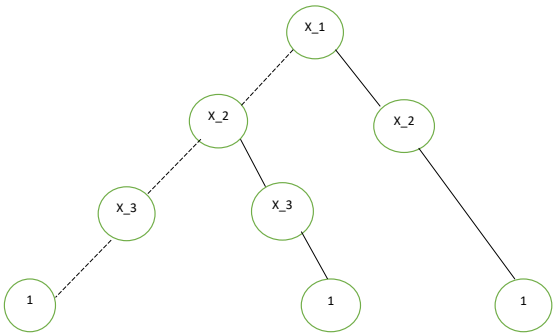

Figure 2: 1-$BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 1.

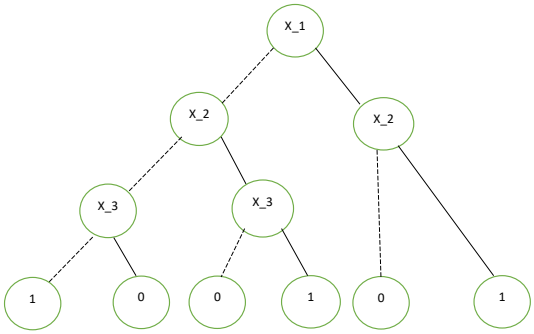

Figure 3: $(1\ltimes 0)$-$BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 1.

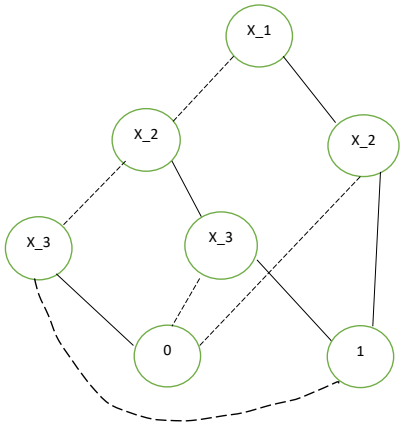

Figure 4: $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 1.

we get that minimum Boolean expression $h(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x'_1x'_2x'_3$. *By Lemma 3.3* (i), $h(x_1, x_2, x_3) = x_1x_2 + x'_1x_2x_3 + x'_1x'_2x'_3$ *and so the* 1-$BDHT(\mathcal{T}(f))$ *and* $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$ *are obtained in Figures 2 and 3, respectively.*

Definition 3.4. *Let* $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ *be a* $(T.B.T)$, *and* $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f)) = (V, E, \longrightarrow, \dashrightarrow)$, *be the binary decision hypertree of* $(T.B.T)$, $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$. *We eliminate the last level which is the output of* $0, 1$, *replace only two nodes as* $0, 1$, *and draw a tree such that all subtrees share* $0, 1$. *We will call as a binary decision hyperdiagram and will denote by* $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$.

Example 3.2. *Consider a* $(T.B.T)$, $\mathcal{T}(f, x_1, x_2, x_3)$ *in Table 1. Then the* $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$ *is obtained in Figure 4.*

Definition 3.5. *Let $n \in \mathbb{N}$ and $\mathcal{T}(f, g, x_1, x_2, \ldots, x_n)$ be a T.B.T. We say that $BDD(\mathcal{T}(f)) = (V, E, \longrightarrow, \dashrightarrow)$ is isomorphic to $BDD(\mathcal{T}(g)) = (V', E', \longrightarrow, \dashrightarrow)$, if there exists a bijection $\varphi : V \to V'$ such that if two vertices x, y are adjacent vertices in $BDD(\mathcal{T}(f))$, then $\varphi(x), \varphi(y)$ are adjacent in $BDD(\mathcal{T}(g))$, $(\overrightarrow{\varphi(x), \varphi(y)})$ or $(\varphi(x), \varphi(y))^{\dashrightarrow}$ hold. In similar a way, an isomorphic between binary decision hyperdiagrams is defined.*

In the following theorem, we investigate the conditions for the isomorphic binary decision (hyper)tree and hyperdiagram.

Theorem 3.6. *Let $n \in \mathbb{N}$ and $\mathcal{T}(f, g, x_1, x_2, \ldots, x_n)$ be a T.B.T. Then*

(i) if $f \sim g$, then $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(g))$;

(ii) if $f \sim g$, then $BDT(\mathcal{T}(f)) \cong BDT(\mathcal{T}(g))$;

(iii) if $f \sim g$, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f)) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(g))$;

(iv) if $f \sim g$, then $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f)) \cong (1 \ltimes 0)$-$BDHT(\mathcal{T}(g))$.

Proof. *(i)* Since $f \sim g$, for all $1 \leq j \leq 2^n$, $f_j(x_1, x_2, \ldots, x_n) = g_j(x_1, x_2, \ldots, x_n)$. Now define a bijection $\varphi : V \to V'$ by $\varphi(x_i) = x_i$ and for all $1 \leq i \leq m, E'_i = \varphi(E_i)$. Clearly if two vertices x, y are adjacent vertices in V, then $\varphi(x), \varphi(y)$ are adjacent in V', we get that $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(g))$.

(ii), (iii), (iv) It is similar to the item *(i)*.

□

We are proved that for any T.B.T, there is a minimum Boolean expression different from its $d.n.f$. Now, apply Theorem 2.3 and obtain the following theorem.

Theorem 3.7. *Let $n \in \mathbb{N}$ and $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$ be a T.B.T. There is minimum Boolean expression in different with to its d.n.f as $h(x_1, x_2, \ldots, x_n)$ such that $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(h))$.*

Proof. Let $n \in \mathbb{N}$ and $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$ be a T.B.T. By Theorem 2.3, there exists a minimum Boolean expression $h(x_1, x_2, \ldots, x_n)$ in such a way that $f \sim h$. Using Theorem 3.6, we have $BDT(\mathcal{T}(f)) \cong BDT(\mathcal{T}(g))$ and so $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(h))$. □

Theorem 3.8. *Let $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ be a T.B.T. Then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(f))$.*

Proof. Let $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ be a $T.B.T$ and $h(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{m} \prod_{i=1}^{k_j} \overline{x_i}$ be the related minimum Boolean expression in such a way that $f \sim h$. We know that $BDHD$ is the binary decision hyperdiagram corresponding to the Boolean expression h and BDD is the binary decision diagram corresponding to the Boolean expression f. Since $f \sim h$, for all $1 \leq j \leq 2^n$, $f_j(x_1, x_2, \ldots, x_n) = h_j(x_1, x_2, \ldots, x_n)$. Now define a bijection $\varphi : V \to V'$ by $\varphi(x_i) = x_i$ and for all $1 \leq i \leq m, E'_i = \varphi(E_i)$. Clearly if two vertices x, y are adjacent vertices in V, then $\varphi(x), \varphi(y)$ are adjacent vertices in V', we get that $(1 \ltimes 0)$-$BDHD \cong BDD$. □

Theorem 3.9. *Let $\mathcal{T}(f, f', x_1, x_2, \ldots, x_n)$ be a T.B.T.*

(i) If f is a hypergraphable Boolean function, then $BDD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong BDD(\mathcal{T}(f))$;

(ii) If $Un(f') = Un(f)$, then $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(f'))$;

(iii) If $Un(f') \subseteq Un(f)$ or $I(n, f + f', 1) = I(n, f, 1)$, then $BDD(\mathcal{T}(f + f')) \cong BDD(\mathcal{T}(f))$;

(iv) If $Un(f') \subseteq Un(f)$ and $I(n, f', 1) = I(n, f, 1)$ imply that $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(f'))$;

(v) If $f(x_1, x_2, \ldots, x_n)$ is a d.n.f and $m \geq 1 + 2^{n-1}$, then $BDD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong BDD(\mathcal{T}(f))$;

(vi) Let $0 \leq j, j' \leq m$. If $\mathcal{T}(f^{(j)}, x_1, \ldots, x_n)$ and $\mathcal{T}'(f^{(j')}, x_1, \ldots, x_n)$ are equivalent, then $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}'(f'))$.

Proof. (i) If f is a hypergraphable Boolean function, then by Theorem 2.3, $(f^{\nearrow})^{\swarrow} \sim f$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong BDD(\mathcal{T}(f))$.

(ii) If $Un(f') = Un(f)$, then $f \sim f'$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(f'))$.

(iii) If $Un(f') \subseteq Un(f)$ or $I(n, f + f', 1) = I(n, f, 1)$, then by Corollary 2.6, $f + f' \sim f$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f + f')) \cong BDD(\mathcal{T}(f))$.

(iv) If $Un(f') \subseteq Un(f)$ and $I(n, f', 1) = I(n, f, 1)$, then by Theorem 2.7, $f \sim f'$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}(f'))$.

(v) If $f(x_1, x_2, \ldots, x_n)$ is a d.n.f and $m \geq 1 + 2^{n-1}$, then by Theorem 2.3, $(f^{\nearrow})^{\swarrow} \sim f$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong BDD(\mathcal{T}(f))$.

(vi) Let $0 \leq j, j' \leq m$. If $\mathcal{T}(f^{(j)}, x_1, \ldots, x_n)$ and $\mathcal{T}'(f^{(j')}, x_1, \ldots, x_n)$ are equivalent, then by Theorem 2.3, $f \sim f'$. Thus by Theorem 3.6, $BDD(\mathcal{T}(f)) \cong BDD(\mathcal{T}'(f'))$. □

The following corollary, analyzes how to the isomorphic binary decision (hyper)tree and hyperdiagram via the unitalizer.

Corollary 3.10. *Let $\mathcal{T}(f, f', x_1, x_2, \ldots, x_n)$ be a T.B.T.*

(i) *If f is a hypergraphable Boolean function, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$;*

(ii) *If $Un(f') = Un(f)$, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f)) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(f'))$;*

(iii) *If $Un(f') \subseteq Un(f)$ or $I(n, f + f', 1) = I(n, f, 1)$, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f + f')) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$;*

(iv) *If $Un(f') \subseteq Un(f)$ and $I(n, f', 1) = I(n, f, 1)$ imply that $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f)) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(f'))$;*

(v) *If $f(x_1, x_2, \ldots, x_n)$ is a d.n.f and $m \geq 1 + 2^{n-1}$, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f^{\nearrow})^{\swarrow}) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}(f))$;*

(vi) *Let $0 \leq j, j' \leq m$. If $\mathcal{T}(f^{(j)}, x_1, \ldots, x_n)$ and $\mathcal{T}'(f^{(j')}, x_1, \ldots, x_n)$ are equivalent, then $(1 \ltimes 0)$-$BDHD(\mathcal{T}(f)) \cong (1 \ltimes 0)$-$BDHD(\mathcal{T}'(f'))$.*

In the following corollary, we summarize our results and the connection of BDD, $(1 \ltimes 0)$-$BDHD$ and $(1 \ltimes 0)$-$BDHT$ in a Hass diagram as follows.

Corollary 3.11. *Let $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$ be a (T.B.T). Then we have the following diagram HT in Figure 5.*

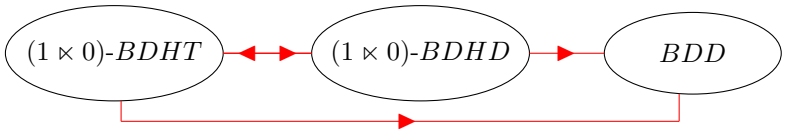

Figure 5: Tree diagram HT of $\mathcal{T}(f \not\equiv 0, x_1, x_2, \ldots, x_n)$

The method for the construction of a Boolean expression from a T.B.T is explained in Algorithm 1 based on [3].

Corollary 3.12. *Let $n, m \in \mathbb{N}$ and $\mathcal{T}(f^{(0)}, f^{(1)}, \ldots, f^{(m)}, (x_1, \ldots, x_n))$ be a T.B.T. Then $BDHT(\mathcal{T}(f^{(0)}, f^{(1)}, \ldots, f^{(m)}, (x_1, \ldots, x_n)))$ is obtained by Algorithm 2.*

3.1. Accessible Binary Decision Hypertree Based on a Program

In this subsection, we present a program(Python programming) to access of binary decision hypertree for any given T.B.T, based on Algorithm 2.

First source-code:

```
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)):
```

Algorithm 1

Input a T.B.T $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$.
if $f \equiv 0$ **or** $f \equiv 1$ **then**
consider $g \equiv 0$ or $g \equiv 1$, respectively.
end if
if If there exists $j \in \{1 \leq j_1, j_2, \ldots, j_s \leq 2^n\}$, such that $f_j(x_1, x_2, \ldots, x_n) = 1$ **then**
consider $g \equiv 0$ or $g \equiv 1$, respectively.
end if
for $1 \leq i \neq j \leq k$ **and** $k \in \mathbb{N}$ **do**
set $F_{ij} = E_i \cap E_j$
end for
if for all $1 \leq i \neq j \leq k, F_{ij} = \emptyset$ **or** $|F_{ij}| < n-1$ **then**
consider $g(x_1, x_2, \ldots, x_n) = \sum_{1 \leq i \leq k} \prod_{\alpha \in E_i} \alpha$.
end if
if there exists $1 \leq r \leq k$, **and** $i \neq j \in \{i_1, i_2, \ldots, i_r\}$ such that $F_{ij} \neq \emptyset$ **and** $|F_{ij}| \geq n-1$ **then**
put $g_{ij}(x_1, x_2, \ldots, x_n) = \prod_{\alpha \in F_{ij}} \alpha$
end if
if $I(n, \sum_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} g_{ij}, 1) = I(n, f, 1)$ **then**
consider $g(x_1, x_2, \ldots, x_n) = \sum_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} g_{ij}(x_1, \ldots, x_n)$ that $Un(g_{ij}) \subseteq Un(f)$
end if
if $I(n, \sum_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} g_{ij}, 1) < I(n, f, 1)$ **then**
consider $1 \leq s \leq k, j \in \{j_1, j_2, \ldots, j_s\}$, $f_j(x_1, x_2, \ldots, x_n) = \prod_{\beta \in E_j} \beta$ such that $Un(\sum_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} g_{ij}) \neq Un(f_j) \subseteq Un(f)$
and $I(n, \sum_{\substack{i_1 \leq i \leq i_r \\ i_1 \leq j \leq i_s}} g_{ij} + \sum_{j=j_1}^{j_s} f_j, 1) = I(n, f, 1)$
end if

```
11      print("Error in input file, rows count isn't equal to 2^n.")
12      exit(0)
13
14  E_i = []
15  for i, elm in enumerate(elements):
16      if(elm[-1] == 1):  # check if f is equal to 1
17          E = []
18          for c in range(0, len(elm)-1):
19              if(elm[c] == 1):
20                  E.append(element_names[c])
21              else:
22                  E.append(element_names[c]+"'")
23          E_i.append([i, E])
24
25  def print_g(g_expr):
26      print("g("+",".join(element_names)+") = "+g_expr)
27      input("")
28      exit(0)
29
30  if len(E_i) == len(elements):
31      print_g("1")
32
33  if len(E_i) == 0:
34      print_g("0")
35
36  F_ij = []
```

Algorithm 2

Input a T.B.T $\mathcal{T}(f, x_1, x_2, \ldots, x_n)$.

Take the Boolean expression $g(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{m} g_j$ from Algorithm 1 [3].

if there exists $1 \leq i \leq n$, such that $\overline{x_i} \in \bigcap_{j=1}^{m} g_j$ **then**

rearrange $x_i =: x_1$ as the root node,

else

there does not exist $1 \leq i \leq n$, such that $\overline{x_i} \in \bigcap_{j} g_j$) go to step 6.

end if

Consider for x_1, leading solid line are labelled by 1 and for x_1', leading dashed line are labeled by 0.

for $2 \leq i \leq n$ **do**

put x_i as the next nodes, **and** for x_i, the leading solid lines are labeled by 1, **and** for x_i', leading dashed lines are labeled by 0.

end for

By Lemma 3.3, consider $h(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{m'} h_j = g_1 + g_1'(g_2 + \ldots + g_m)$, where $m \leq m'$ so there exists $1 \leq i \leq n$, such that $\overline{x_i} \in \bigcap_{j=1}^{m'} h_j$ and go to step 3.

```
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")
        else:
            tmp_g = tmp_g.replace(element_names[c]+"'", "1")
            tmp_g = tmp_g.replace(element_names[c], "0")
    true_g_ij.append(calc_mult_and(tmp_g))

```

```
f_j = []
for i, elm in enumerate(elements):
    if(elm[-1] == 1 and true_g_ij[i] == 0):
        f_j.append("".join([x for x in E_i if x[0] == i][0][1]))

print_g("+".join(g_ij + f_j))
```

Second source-code:

```
# https://github.com/BaseMax/BinaryTreePython
# https://github.com/BaseMax/BinaryTreeDiagram
# https://github.com/BaseMax/BinaryTreeDiagramDrawing

import itertools, graphviz as gvz
from math import log2, floor

matrix = {
    "xyz": (0, 1, 2, 3, 4, 5, 6, 7),
    "xzy": (0, 1, 4, 5, 2, 3, 6, 7),
    "yxz": (0, 1, 4, 5, 2, 3, 6, 7),
    "yzx": (0, 4, 1, 5, 2, 6, 3, 7),
    "zxy": (0, 2, 4, 6, 1, 3, 5, 7),
    "zyx": (0, 4, 2, 6, 1, 5, 3, 7),
}

value = input('Enter all of minterms in one line with space:')
combf = list(map(lambda x: int(x),value.split()))
combf.sort()

def merge(lst):
    res = []
    for i in range(0, len(lst) - 1, 2):
        res.append((lst[i], lst[i + 1]))
    return res

def make_form(combf, fulltree):
    res = []
    for i in range(len(matrix[form])):
        if matrix[form][i] in combf:
            res.append(matrix[form][i])
        else:
            res.append(None)
    while (len(res) > 1):
        res = merge(res)
    return res[0]

def find_best_poly(combf, fulltree, w=0):
    new_combf = []
    for element in combf:
        if element in fulltree:
            new_combf.append(element)
            w += 1
    if len(new_combf) < 2:
        return w
    return find_best_poly(list(itertools.combinations(new_combf, 2)),merge(fulltree), w)

my_tree = []

def draw(tree, g, form, h=1):
    if (hasattr(tree, "__iter__")):

        l = draw(tree[0], g, form, 2 * h)
        r = draw(tree[1], g, form, 2 * h + 1)

        if l == r and r != (1, 1):

            g.node(f'{h}',f'{l[0]}',style="invis" if l==(0,0) else None)
            g.node(f'{2*h}',f'{l[0]}',style="invis")
            g.edge(f'{h}',f'{2*h}',style="invis")

            my_tree.append([
                str(h),
                str(2*h),
                str(l[0]),
```

```
66                  str(l[0]),
67                  "invis"
68              ])
69
70              g.node(f'{2*h+1}',f'{r[0]}',style="invis")
71              g.edge(f'{h}',f'{2*h+1}',style = "invis")
72              my_tree.append([
73                  str(h),
74                  str(2*h+1),
75                  str(l[0]),
76                  str(r[0]),
77                  "invis"
78              ])
79
80              return l
81
82          g.node(f'{h}', f'{form[floor(log2(h))]}')
83          g.node(f'{2*h}',f'{form[floor(log2(2*h))]}'  if l[1] else f'{l[0]}', style=None if l[0]
                else "invis")
84          g.edge(f'{h}', f'{2*h}',style="dashed" if l[0] else "invis")
85
86          my_tree.append([
87              str(h),
88              str(2*h),
89              str(form[floor(log2(h))]),
90              str(form[floor(log2(2*h))] if l[1] else l[0]),
91              "dashed" if l[0] else "invis"
92          ])
93
94          g.node(f'{2*h+1}',f'{form[floor(log2(2*h+1))]}'  if r[1] else f'{r[0]}', style=None if r
                [0] else "invis")
95          g.edge(f'{h}', f'{2*h+1}',style=None if r[0] else "invis")
96
97          my_tree.append([
98              str(h),
99              str(h*h+1),
100             str(form[floor(log2(h))]),
101             str(form[floor(log2(2*h+1))] if r[1] else r[0]),
102             # None if r[0] else "invis"
103             "invis" if r[0] else "invis"
104         ])
105
106         return (1, 1)
107
108     return (1, 0) if type(tree) == int else (0, 0)
109
110 def clean_tree(tree):
111     index = 0
112
113     res_tree = [item for item in tree]
114
115     removed = 0
116
117     for item in tree:
118         if item[2] == '0' or item[3] == '0':
119             del res_tree[index - removed]
120             removed = removed + 1
121         if item[2] == '1' and item[3] == '1':
122             del res_tree[index - removed]
123             removed = removed + 1
124
125         index = index+1
126
127     return res_tree
128
129 if __name__ == "__main__":
130     weight = {
131         fulltree[0]: find_best_poly(list(itertools.combinations(combf, 2)),merge(fulltree[1]))
132         for fulltree in matrix.items()
133     }
134     form = list(weight.keys())[list(weight.values()).index(max(weight.values()))]
135     g = gvz.Graph(format="png",filename="btree.gv")
136     tree = make_form(combf, matrix[form])
```

Table 2: T. B. T with 3 variables $\mathcal{T}(f, x, y, z)$

x	y	z	$f(x, y, z)$
0	0	0	$f_1(x, y, z) = 0$
0	0	1	$f_2(x, y, z) = 0$
0	1	0	$f_3(x, y, z) = 1$
0	1	1	$f_4(x, y, z) = 1$
1	0	0	$f_5(x, y, z) = 1$
1	0	1	$f_6(x, y, z) = 1$
1	1	0	$f_7(x, y, z) = 0$
1	1	1	$f_8(x, y, z) = 1$

```
137     draw(tree, g, form)
138
139     my_tree = clean_tree(my_tree)
140     node_added = 0
141
142     x = 0
143     for value in my_tree:
144         if value[3] == '1':
145             y = 0
146             reached_to_x = []
147             for value2 in my_tree:
148                 if value2[0] == value[0]:
149                     reached_to_x.append(value2)
150                 y = y + 1
151             if len(reached_to_x) == 1:
152                 g.node('o'+str(100+node_added), '0')
153                 g.edge(value[0], 'o'+str(100+node_added), style="dashed" if (reached_to_x[0][4] ==
                        'invis') else None)
154                 node_added = node_added + 1
155         x = x + 1
156
157     g.view()
```

In the following example, we first extract the binary decision hypertree manually for the given T.B.T table. Then, using Python programming, the binary decision hypertree corresponding to a T.B.T in Figure 7 is obtained. We can see that the output of the Python programming and the output of the manual method are the same.

Example 3.3. *Let $H = \{x, y, z, x', y', z'\}$ and consider a (T.B.T), $\mathcal{T}(f, x, y, z)$ in Table 2. By Theorem 2.3, we get an undirected hypergraph $\mathcal{H}' = (H, E_1, E_2, E_3, E_4, E_5)$, where $E_1 = \{x', y, z'\}, E_2 = \{x', y, z\}, E_3 = \{x, y', z'\}, E_4 = \{x, y', z\}$ and $E_5 = \{x, y, z\}$. Since*

$$E_1 \cap E_2 = \{x', y\}, E_1 \cap E_3 = \{z'\}, E_1 \cap E_4 = \emptyset, E_1 \cap E_5 = \{y\}, E_2 \cap E_3 = \emptyset, E_2 \cap E_4 = \{z\},$$
$$E_2 \cap E_5 = \{y, z\}, E_3 \cap E_4 = \{x, y'\}, E_3 \cap E_5 = \{x\} \text{ and } E_4 \cap E_5 = \{x, z\},$$

we get that minimum Boolean expression $h(x, y, z) = x'y + yz + xy' + xz$. By Lemma 3.3, $h(x, y, z) = x'y + xy' + xyz$ and so the $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$ are obtained in Figure 6.

4. Application of Binary Decision Hypertree Based on a T.B.T

In this section, we apply Python programming and introduce two samples of real-life applications of binary decision hypertree.

Fighter jet refueling: Numerous missions for fighter jets in naval air defense, escorting other fighters, suppressing enemy air defenses, strike attacks, bombing land and sea targets, air superiority missions, intercepting fighters and bombers, and Enemy missiles, close air support and air reconnaissance are defined. The design of the new generation of fighter jets to carry out round-the-clock missions in any kind of weather conditions and the ability to refuel in the air and to land and fly from aircraft carriers is of particular importance. The fighter jet refuels at least twice during the mission. If we consider the different refueling modes of a fighter jet, three inputs are required.

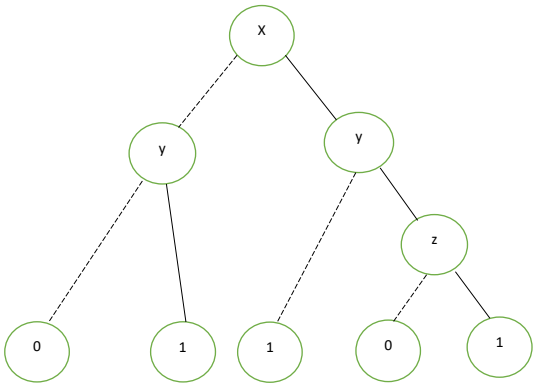

Figure 6: $(1\ltimes 0)$-$BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 2.

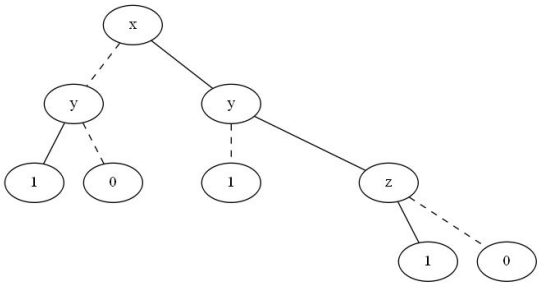

Figure 7: $(1 \ltimes 0)$-$BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 2.

Table 3: T. B. T with 3 variables $\mathcal{T}(f, x, y, z)$

x	y	z	$f(x,y,z)$
0	0	0	$f_1(x,y,z)=0$
0	0	1	$f_2(x,y,z)=0$
0	1	0	$f_3(x,y,z)=0$
0	1	1	$f_4(x,y,z)=1$
1	0	0	$f_5(x,y,z)=0$
1	0	1	$f_6(x,y,z)=1$
1	1	0	$f_7(x,y,z)=1$
1	1	1	$f_8(x,y,z)=1$

$x :=$ fighter jet refueling in the air (1=fuel, 0=do not fuel), y:= fighter jet refueling in the aircraft carrier (1=fuel, 0=do not fuel) and z:= fighter jet refueling in the nest. (1=fuel, 0=do not fuel). We want to find the binary decision hypertree in such a way that it is determined in which modes the jet is refueling in the air. Therefore, based on the T.B.T 3 and using Python programming, we get the binary decision hypertree as shown in Figure 8.

Football Matches: After the draw, the football team in the World Cup is placed in a group of four, which must compete with the teams in the same group. According to the rules of the matches, in case of two consecutive defeats, the team will be removed from the matches. Using Python programming, the binary decision hypertree (in Figure 9) corresponding to a T.B.T 4, shows the number of elimination modes of the football team:

Table 4: T. B. T with 3 variables $\mathcal{T}(f, x, y, z)$

x	y	z	$f(x,y,z)$
0	0	0	$f_1(x,y,z)=1$
0	0	1	$f_2(x,y,z)=1$
0	1	0	$f_3(x,y,z)=0$
0	1	1	$f_4(x,y,z)=0$
1	0	0	$f_5(x,y,z)=1$
1	0	1	$f_6(x,y,z)=0$
1	1	0	$f_7(x,y,z)=0$
1	1	1	$f_8(x,y,z)=0$

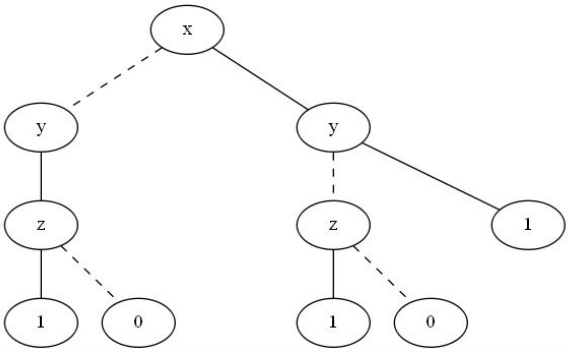

Figure 8: $BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 3.

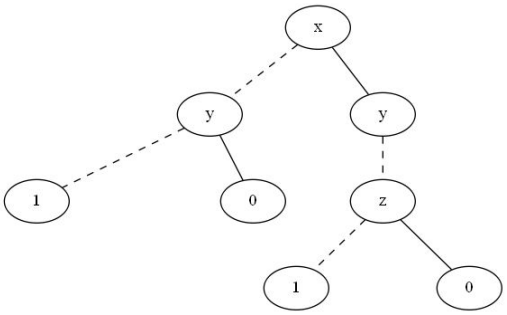

Figure 9: $BDHT(\mathcal{T}(f))$ corresponding to $T.B.T$ Table 4.

5. Conclusions and future works

The current paper has introduced a novel concept of (BDHT) binary decision hypertree (hyperdiagram) via minimum Boolean expression and presents some conditions on T.B.T for the isomorphism of binary decision hyperdiagrams and binary decision hypertrees. Finally, for faster calculations and more complex functions, we present an algorithm and so Python programming (with complete and original codes) such that for any given T.B.T, it introduces a binary decision hypertree(hyperdiagram). The main achievement of this study is a fundamental documentary to truly evaluate the base of algorithms and codes of Python programming. We firstly proved some theorems in this regard and evaluate some types of any given T.B.T with these theorems and Python programs, so we extracted some algorithms and codes of Python programs based on these proved theorems. Indeed, the input of the Python programs is any given T.B.T and the output of the Python programs is a binary decision tree the method is based on the mathematical theorems which are proved in this paper which were algorithmized and codes were written based on these algorithms. We hope that these results are helpful for further studies in the theory of graphs, hypergraphs, superhypergraphs, and decision (super)(hyper)graphs, also known as influence (hyper)diagrams. In our future studies, we hope to obtain more results regarding decision-making based on hypergraphs, superhypergraphs, and their applications in the real world.

References

[1] C. Berge, *Graphs and hypergraphs*, North-Holland Mathematical Library, Vol. 6, North-Holland Publishing Co., Amsterdam-London; American Elsevier Publishing Co., Inc., New York, 1973. Translated from the French by Edward Minieka.

[2] M. Hamidi and A. Broumand Saeid, *On derivable trees*, Trans. Comb., 8 (2019), pp. 21–43.

[3] M. Hamidi, M. Rahmati, and A. Rezaei, *Switching function based on hypergraphs with algorithm and python programming*, J. Intell. Fuzzy Syst., 39 (2020), pp. 2845–2859.

[4] M. Hamidi and A. B. Saeid, *Accessible single-valued neutrosophic graphs*, J. Appl. Math. Comput., 57 (2018), pp. 121–146.

[5] M. Hamidi and A. B. Saeid, *Achievable single-valued neutrosophic graphs in wireless sensor networks*, New Math. Nat. Comput., 14 (2018), pp. 157–185.

[6] M. Hamidi and F. Smarandache, *Single-valued neutrosophic directed (hyper)graphs and applications in networks*, J. Intell. Fuzzy Syst., 37 (2019), pp. 2869–2885.

[7] B. Molnár, *Applications of hypergraphs in informatics: a survey and opportunities for research*, Ann. Univ. Sci. Budap. Rolando Eötvös, Sect. Comput., 42 (2014), pp. 261–282.

[8] J. E. Whitesitt, *Boolean algebra and its applications*, Dover Publications, Inc., New York, 1995. Reprint of the 1961 original.

Please cite this article using:

Mohammad Hamidi, Marzieh Rahmati, On binary decision hypertree (hyperdiagram), AUT J. Math. Com., 5(2) (2024) 117-130
https://doi.org/10.22060/AJMC.2023.21639.1094