Precedence hyperstructures and graphs in assembly line design

Anastasia Taouktsoglou*
Department of Production and Management Engineering
Democritus University of Thrace
V. Sofias 12, 67 132 Xanthi
Greece
ataoukts@pme.duth.gr

Stefanos Spartalis
Department of Production and Management Engineering
Democritus University of Thrace
V. Sofias 12, 67 132 Xanthi
Greece
sspart@pme.duth.gr
and
School of Science and Technology
Studies in Physics
Hellenic Open University
18 Aristoteles by street, 26 335 Patra
Greece
spartalis.stefanos@ac.eap.gr

Abstract. In this paper we introduce the precedence hyperoperation, which constructs a precedence partial hypergroupoid, i.e. a partial hypergroupoid with some special properties. Given a precedence partial hypergroupoid, a precedence graph can be defined and vice versa. Using the precedence partial hypergroupoid of a precedence graph and the Fewer-Descendants-Vertex First algorithm (FDVF algorithm), a process flow diagram is created, which can be used in mixed-model assembly line design.

Keywords: precedence graph, precedence partial hypergroupoid, process flow diagram, assembly line design, Fewer-Descendants-Vertex First algorithm (FDVF algorithm), More-Descendants-Vertex First algorithm (MDVF algorithm).

MSC 2020: 05C20, 20N20, 05C38, 05C90, 90C35, 68R10.

1. Introduction

Algebraic hyperstructures were introduced in 1934 by Marty [20], as a generalization of the notion of the group and have been studied since by many mathematicians (s. [1]-[17], [19], [21]-[22], [24]-[28]).

Hyperstructures associated with binary relations have a special interest for many researchers, such as I. Rosenberg [22], P. Corsini [1]-[6], Y. Feng [12], V.

*. Corresponding author

Leoreanu-Fotea [4]-[6], [10], [19], B. Davvaz [10], I. Cristea and M. Stefanescu [7]-[9], M. Konstantinidou-Serafimidou and K. Serafimidis [17], [27], M. De Salvo and J. Lo Faro [11], S. J. Rasovic [21], S. Spartalis [13]-[16], [24]-[27], A. Kalampakas [13]-[16], [26], A. Taouktsoglou [27] and others.

Additionally, binary relations and directed graphs are representing complex information and describing rich structures. Therefore, the connection between hyperstructures and graph theory appear naturally and are studied in [13], [14], [15] and [26]. More precisely, in [14] the *"path hyperoperation"* and the associated *"path hypergroupoids"* were introduced, as a generalization of the C-hypergroupoids [3], [24], [25] and an application to the design and management of mixed-model assembly lines, was presented.

On the other hand, precedence graphs, first developed by M. E. Salveson [23], are traditionally used to visualize the assembly sequence of a model in many industrial environments (see also, The Assembly Line Balancing Problem [18]). A precedence graph is used in order for a product to be assembled in a proper way. Designing an assembly line one has to correspond the tasks needed for each product to workstations standing in a row. All precedence conditions have to be fulfilled and the working time among the workstations has to be balanced.

In this respect, we introduce and study a new hyperoperation, which constructs a *"precedence partial hypergroupoid"*, i.e. a partial hypergroupoid with some special properties. Given a precedence partial hypergroupoid, a precedence graph can be defined and vice versa. Using the precedence partial hypergroupoid of a precedence graph and following the Fewer-Descendants-Vertex First algorithm (FDVF algorithm) that we created, one can design and computer program a process flow diagram, which can be used in mixed-model assembly line designing.

In Section 2 basic concepts on hypergroupoids and directed graphs are presented. In Section 3 we define the precedence hyperoperation and we investigate the properties of the associated precedence hypergroupoid. In Section 4 using a precedence hypergroupoid we define a precedence graph and vice versa, proving that there is a 1-1 correspondence between the two notions. We also, set necessary conditions for a graph to be a precedence graph. In Section 5 we present an algorithm one can use, in order to derive a process flow diagram through a precedence hypergroupoid. The proposed FDVF algorithm used on precedence hypergroupoids can be applied in mixed-model assembly line designing.

2. Preliminaries

A partial hypergroupoid is a pair $(H, *)$, where H is a non empty set and $*$ is a hyperoperation i.e.

$$* : H \times H \to \wp(H), \qquad (x, y) \mapsto x * y.$$

If $A, B \in \wp(H) - \{\emptyset\}$, we set $A * B = \bigcup_{a \in A, b \in B} a * b$. We also, denote $a * B$ (resp. $A * b$) the hyperproduct $A * B$ in case that A is the singleton $\{a\}$ (resp.

B is the singleton $\{b\}$). $(H, *)$ is called a *"hypergroupoid"* if $x * y \neq \emptyset$, for all $x, y \in H$ and it is called *"degenerative* (resp. *"total") hypergroupoid"* if $x * y = \emptyset$ (resp. $x * y = H$), for all $x, y \in H$ (s. [27]).

Given a binary relation $R \subseteq H \times H$ hypergroupoids can be defined in many ways. For example, a wide class of partial hypergroupoids, named *"partial C-hypergroupoids"*, are defined by the *"Corsini's hyperoperation"* [3]:

$$*_R : H \times H \to \wp(H) : \ (x, y) \mapsto x *_R y = \{z \in H / (x, z) \in R \text{ and } (z, y) \in R\}.$$

Given a non-empty finite set V and a binary relation $R \subseteq V \times V$ *"a concrete directed graph"* G is defined as the pair (V, R). The elements of V are called *"vertices"* and the elements of R are called *"edges"* (s. [14]). Drawing a graph, a vertex is drawn as a node and an edge as an arrow connecting two vertices, which are called the *"head"* and the *"tail"* of the edge.

A vertex of a graph is called *"isolated"* if there is no edge connected to it. In what follows we consider concrete directed graphs without isolated vertices.

A graph $G' = (V', R')$ is called a *"subgraph"* of the graph $G = (V, R)$ if it holds $V' \subseteq V$ and $R' \subseteq R$. Then we say that "$G' = (V', R')$ *is included in* $G = (V, R)$" and we denote $G' \leq G$.

Given a graph $G = (V, R)$ and $v_1, v_n \in V$, *"a directed path from* v_1 *to* v_n" (or simply *"a path from* v_1 *to* v_n") is defined as a pair $P = (V_P, E_P)$ of a set of vertices $V_P = \{v_1, v_2, \ldots, v_n\} \subseteq V$ and a set of edges $E_P = \{e_1, e_2, \ldots, e_{n-1}\} \subseteq R$, where $e_i = (v_i, v_{i+1})$, for all $i = 1, 2, \ldots, n-1$, where $n \in \mathbb{N}, n > 1$. Then, $v_1, v_2, \ldots, v_n$ are called *"vertices of path* P" and $e_1, e_2, \ldots, e_{n-1}$ are called *"edges of path* P". Consequently, v_1 is called *"starting vertex"* and v_n *"ending vertex"* of path P. According to the previous definition, the pair $P = (V_P, E_P)$ is a concrete directed graph included in the graph $G = (V, R)$.

Practically, a path from v_1 to v_n is

- an alternating sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n$ beginning at v_1 and ending at v_n, where all vertices are distinct from one another and $e_i = (v_i, v_{i+1})$, $\forall i = 1, 2, \ldots, n-1$ or equivalently,
- a sequence of vertices $v_1, v_2, \ldots, v_n$, all distinct from one another, beginning at v_1 and ending at v_n, where there exist edges $e_i = (v_i, v_{i+1}) \in R$, $\forall i = 1, 2, \ldots, n-1$.

Example 2.1. In the graph of Figure 1 several paths from vertex 1 to vertex 6 are displayed, i.e.

$$P_1 : 1, 5, 6 \qquad P_2 : 1, 2, 6$$
$$P_3 : 1, 3, 5, 6 \quad P_4 : 1, 5, 2, 6$$
$$P_5 : 1, 2, 4, 6 \quad P_6 : 1, 2, 5, 6$$

"An induced path" is defined as a path, in which each two adjacent vertices are connected by an edge and each two nonadjacent vertices are not connected by any edge.

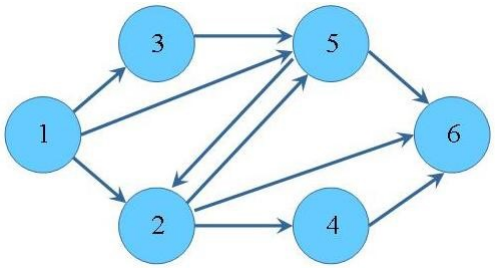

Figure 1: Several paths from vertex 1 to vertex 6

Given a graph $G = (V, R)$, *"a directed circle"* (or simply *"a circle"*) is defined as a pair $C = (V_C, E_C)$ of a set of vertices $V_C = \{v_1, v_2, \ldots, v_{n-1}\} \subseteq V$ and a set of edges $E_C = \{e_1, e_2, \ldots, e_{n-1}\} \subseteq R$, where $e_i = (v_i, v_{i+1})$, for all $i = 1, 2, \ldots, n-2$ and $e_{n-1} = (v_{n-1}, v_1)$, where $n \in \mathbb{N}, n > 2$. Then, $v_1, v_2, \ldots, v_{n-1}$ are called *"vertices of the circle C"* and $e_1, e_2, \ldots, e_{n-1}$ are called *"edges of the circle C"*.

Practically, a circle is

- a path, whose starting and ending vertices coincide, or equivalently,
- an alternating sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_1$ beginning and ending at the same vertex, where all vertices are distinct from one another, except from the starting and the ending vertex, and $e_i = (v_i, v_{i+1})$, $\forall i = 1, 2, \ldots, n-2$, $e_{n-1} = (v_{n-1}, v_1)$, where $n \in \mathbb{N}$, $n > 2$.

Given a graph $G = (V, R)$ and $v_1, v_n \in V$, *"a directed walk from v_1 to v_n"* (or simply *"a walk from v_1 to v_n"*) is defined as a union P of consecutive paths, starting at v_1 and ending at v_n, which contains at least one circle. Then, all vertices of the united paths are called *"vertices of walk P"* and all edges of the united paths are called *"edges of walk P"*. Consequently, v_1 is called *"starting vertex"* and v_n *"ending vertex"* of walk P.

3. Precedence hyperoperation

Definition 3.1. *Given a non empty set V a partial hyperoperation*

$$*_P : V \times V \to \wp(V) : \ (x, y) \mapsto x *_P y$$

can be defined, for all $x, y \in V$, in the following way:

i. $x *_P y \neq \emptyset \Rightarrow (x \in x *_P y$ *and* $y \notin x *_P y)$;

ii. $\exists\, a \in V : a *_P x \neq \emptyset$, *for all* $x \in V - \{a\}$;

iii. $\exists\, b \in V : x *_P b \neq \emptyset$, *for all* $x \in V - \{b\}$;

iv. $x *_P y = (a *_P y) \cap (x *_P b)$;

$$v.\ u \in x *_P y \Rightarrow \begin{cases} (x *_P u) \cup (u *_P y) \subseteq x *_P y & \\ x \in a *_P u, & \text{if } x \neq u \\ y \in u *_P b, & \text{if } y \neq b \end{cases}$$

Such a hyperoperation is called "precedence hyperoperation" and introduces a partial hypergroupoid $(V, *_P)$ called "precedence partial hypergroupoid". The elements a and b are called "starting" and "ending element" respectively. The results $x *_P u$ and $u *_P y$ are called "complementary to $x *_P y$".

Remark 3.1. It is obvious from Definition 3.1-i that

$$x *_P x = \emptyset, \qquad \text{for all } x \in V. \tag{1}$$

Therefore, conditions i, ii and iii of Definition 3.1 imply that

$$a \in a *_P x, \qquad \text{for all } x \in V - \{a\}, \tag{2}$$

$$x \in x *_P b, \qquad \text{for all } x \in V - \{b\}. \tag{3}$$

Moreover, setting $x = y$ in Definition 3.1-iv, we obtain that

$$(a *_P x) \cap (x *_P b) = \emptyset, \qquad \text{for all } x \in V. \tag{4}$$

Example 3.1. Let $V = \{1, 2, 3, 4, 5, 6\}$. We consider the hyperoperation $*_P : V \times V \to \wp(V)$ given by Table 1. One can see that this hyperoperation satisfies

	1	2	3	4	5	6
1	$\varnothing$	$\{1\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,2\}$	$\{1,2,3,4,5\}$
2	$\varnothing$	$\varnothing$	$\{2\}$	$\{2,3\}$	$\{2\}$	$\{2,3,4,5\}$
3	$\varnothing$	$\varnothing$	$\varnothing$	$\{3\}$	$\varnothing$	$\{3,4\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{4\}$
5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{5\}$
6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 1: Precedence Partial Hypergroupoid

conditions i-v of Definition 3.1. So, it is a precedence hyperoperation with element 1 as starting element and element 6 as ending element.

Proposition 3.1. *Given a precedence partial hypergroupoid $(V, *_P)$, for all $x, y \in V$ the following hold:*

i. $x *_P y \neq \emptyset \Rightarrow y *_P x = \emptyset$;

ii. $x *_P a = \emptyset = b *_P x$.

Proof. i. Let $x *_P y \neq \emptyset$ and $y *_P x \neq \emptyset$. Then, according to Definition 3.1-i we get $x \in x *_P y$ and $y \in y *_P x$. But $x \in x *_P y$ and Definition 3.1-iv imply that $x \in a *_P y$. Similarly, since $y \in y *_P x = (a *_P x) \cap (y *_P b)$, it holds $y \in a *_P x$ and according to Definition 3.1-v we get either $x \in y *_P b$ or $x = b$.

In case $x \in y *_P b$, it holds $x \in (a *_P y) \cap (y *_P b)$, which is a contradiction according to (4).

In case $x = b$, $x \in x *_P y$ and Definition 3.1-iv imply that $b \in b *_P b$, which is a contradiction according to (1).

ii. For $x = a$ it is obvious that $x *_P a = \emptyset$. Moreover, let $x \in V - \{a\}$. Then, according to (2), $a \in a *_P x$ and from Proposition 3.1-i we obtain that $x *_P a = \emptyset$.

For $x = b$ it is obvious that $b *_P x = \emptyset$. Moreover, let $x \in V - \{b\}$. Then, according to (3), $x \in x *_P b$ and from Proposition 3.1-i we obtain that $b *_P x = \emptyset$.

□

Remark 3.2. Considering the multiplicative table of a precedence partial hypergroupoid $(V, *_P)$ one can see the following:

- All diagonal results of the table are $\emptyset$. Moreover, the starting element a belongs to every result of its row (except the diagonal one) and every element i (except the ending element b) belongs to the result, which appears at its own row and the column of the ending element b (s. Remark 3.1).
- All results in the column of the starting element a, as well as all results in the row of the ending element b are $\emptyset$. Moreover, in case that an (i, j) cell of the multiplicative table is not $\emptyset$, then the diagonal symmetric cell is equal to $\emptyset$ (s. Proposition 3.1).

Proposition 3.2. *Given a precedence partial hypergroupoid* $(V, *_P)$ *the following hold:*

i. the starting element is unique;

ii. the ending element is unique;

iii. $V *_P y = \bigcup_{x \in V} x *_P y = a *_P y$, *for all* $y \in V$;

iv. $x *_P V = \bigcup_{y \in V} x *_P y = x *_P b$, *for all* $x \in V$;

v. $a \notin x *_P y$, *for all* $x \in V - \{a\}, y \in V$;

vi. $b \notin x *_P y$, *for all* $x, y \in V$.

Proof. i. Let a_1, a_2 be both starting elements of $(V, *_P)$, $a_1 \neq a_2$. Then, (2) implies that $a_1 \in a_1 *_P a_2$ and $a_2 \in a_2 *_P a_1$. So, $a_1 *_P a_2 \neq \emptyset$ and $a_2 *_P a_1 \neq \emptyset$, which is a contradiction, according to Proposition 3.1-i.

ii. Let b_1, b_2 be both starting elements of $(V, *_P)$, $b_1 \neq b_2$. Then, (4) implies that $b_1 \in b_1 *_P b_2$ and $b_2 \in b_2 *_P b_1$. So, $b_1 *_P b_2 \neq \emptyset$ and $b_2 *_P b_1 \neq \emptyset$, which is a contradiction, according to Proposition 3.1-i.

iii. Definition 3.1-iv implies that $x *_P y \subseteq a *_P y$, for all $x, y \in V$. So, $\bigcup_{x \in V} x *_P y \subseteq a *_P y$, for all $y \in V$. Furthermore $a *_P y \subseteq \bigcup_{x \in V} x *_P y$. So, $\bigcup_{x \in V} x *_P y = a *_P y$.

iv. Definition 3.1-iv implies that $x *_P y \subseteq x *_P b$, for all $x, y \in V$. So, $\bigcup_{y \in V} x *_P y \subseteq x *_P b$, for all $x \in V$. Furthermore $x *_P b \subseteq \bigcup_{y \in V} x *_P y$. So, $\bigcup_{y \in V} x *_P y = x *_P b$.

v. Let $x \neq a$. Since $a \in a *_P x$ (s. (3)), according to (4), we get $a \notin x *_P b$. Then, $a \notin x *_P y \subseteq x *_P b$ (s. Proposition 3.2-iv).

vi. Let x, y be two elements of V such that $b \in x *_P y$. Then, Proposition 3.2-iv implies that $b \in x *_P b$, which is a contradiction according to Definition 3.1-i.

□

Remark 3.3. Considering the multiplicative table of a precedence partial hypergroupoid $(V, *_P)$ the unique starting element a (resp. the unique ending element b) may correspond to the first row and the first column (resp. the last row and the last column) of the table. Then one can see the following:

- All results of a column are subsets of the first result of this column (i.e., for all columns except the first column the following holds: every result of a column -except the first result- is either the empty set or a proper subset of the first result of this column).
- All results of a row are subsets of the last result of this row (i.e., for all rows except the last row the following holds: every result of a row -except the last result- is either the empty set or a proper subset of the last result of this row). Furthermore, every element, except the ending element b, belongs to the last result of its own row.
- The starting element a belongs to all results of the first row, except the diagonal one, and to no other result.
- The ending element b belongs to no results.

Definition 3.2. *Let $(V, *_P)$ be a precedence partial hypergroupoid with starting vertex a and ending vertex b. The set $V *_P x$ is called "ancestors' set of x" and the set $(x *_P V - \{x\}) \cup \{b\}$ is called "descendants' set of x". The integer*

$card[(x *_P V - \{x\}) \cup \{b\}] = card(x *_P V)$ is called "number of descendants of x". For every $i, j \in V$ with $card(i *_P V) > card(j *_P V)$ we say that "vertex i has more descendants than vertex j" or equivalently we say that "vertex j has fewer descendants than vertex i". If $card(i *_P V) = card(j *_P V)$ we say that "vertices i and j have both the same number of descendants".

Proposition 3.3. *Given a precedence partial hypergroupoid $(V, *_P)$ the following hold:*

i. $a *_P b = V - \{b\}$;

ii. $x *_P y \neq \emptyset \Rightarrow \begin{cases} x \in a *_P y \\ y \in x *_P b \end{cases}$, *for all* $x, y \in V,\ y \neq b$;

iii. $x *_P y \neq \emptyset \Rightarrow \begin{cases} x \notin y *_P b \\ y \notin a *_P x \end{cases}$, *for all* $x, y \in V$;

iv. the starting element and the ending element coincide, only if V is a singleton.

Proof. i. Proposition 3.2-vi implies that $a *_P b \subseteq V - \{b\}$. Let now $x \in V - \{b\}$. Then $x \in x *_P b$ (s. Definition 3.1-iii). According to Proposition 3.2-iii we have $x *_P b \subseteq a *_P b$. So, $x \in a *_P b$, i.e. $V - \{b\} \subseteq a *_P b$. Consequently, $a *_P b = V - \{b\}$.

ii. $x *_P y \neq \emptyset$ implies that $x \in x *_P y$ (s. Definition 3.1-i). Then, according to Definition 3.1-iv we get $x \in a *_P y$. On the other hand, setting $u = x$ in Definition 3.1-v we get $y \in x *_P b$.

iii. It is obvious from (4), Proposition 3.3-ii and Proposition 3.2-vi.

iv. In case $a = b$, since $a *_P b = a *_P a = \emptyset$, but also $a *_P b = V - \{b\}$ (s. Proposition 3.3-i) we obtain that V is a singleton, i.e. $V = \{a\}$.

□

Remark 3.4. Considering the multiplicative table of a precedence partial hypergroupoid $(V, *_P)$ with the starting element a (resp. the ending element b) in the first row and the first column (resp. in the last row and the last column) of the table, one can see the following:

- If the result at an arbitrary (x, y) cell is not $\emptyset$, then the element x belongs to the first result of the y-column and the element y belongs to the last result of the x-row, i.e. x belongs to the ancestors' set of y and y belongs to descendants' set of x.
- If element x precedes element y, then y does not precede x.

Proposition 3.4. *Let V be a non empty set with $card(V) = n$.*

i. For $n = 1$, the only precedence partial hypergroupoid defined on V is the degenerative one.

ii. For $n \in \{2, 3\}$, one precedence partial hypergroupoid is defined on V.

iii. For $n = 4$, three precedence partial hypergroupoids are defined on V.

Proof. i. Let $n = 1$ i.e. $V = \{1\}$. In a precedence partial hypergroupoid $(V, *_P)$ it holds $1 *_P 1 = \emptyset$. So, $(V, *_P)$ is the degenerative one.

ii. Let $n = 2$ i.e. $V = \{1, 2\}$. Let element 1 be the starting element and element 2 be the ending element. According to Proposition 3.1-ii and Proposition 3.3-i the only precedence partial hypergroupoid $(V, *_P)$ defined on V has the multiplicative Table 2.

	1	2
1	$\emptyset$	$\{1\}$
2	$\emptyset$	$\emptyset$

Table 2: Precedence partial hypergroupoid defined on $V = \{1, 2\}$

Similarly, if $n = 3$ i.e. $V = \{1, 2, 3\}$ with element 1 as starting element and element 3 as ending element, the only precedence partial hypergroupoid $(V, *_P)$ defined on V has the multiplicative Table 3.

	1	2	3
1	$\emptyset$	$\{1\}$	$\{1,2\}$
2	$\emptyset$	$\emptyset$	$\{2\}$
3	$\emptyset$	$\emptyset$	$\emptyset$

Table 3: Precedence partial hypergroupoid defined on $V = \{1, 2, 3\}$

iii. Let $n = 4$ i.e. $V = \{1, 2, 3, 4\}$ and $(V, *_P)$ be a precedence partial hypergroupoid defined on V. Let also element 1 be the starting element and element 4 be the ending element of $(V, *_P)$. Then the multiplicative table of $(V, *_P)$ looks like Table 4 because

- since element 1 is its starting element, it belongs to every result of its own row, except the diagonal one, and all results of its own column are $\emptyset$,
- since element 4 is its ending element, every result of its column contains the element, which corresponds to the row of the result, except the diagonal result; furthermore, all results of its own row are $\emptyset$,

	1	2	3	4
1	$\varnothing$	$\{1,?\}$	$\{1,?\}$	$\{1,2,3\}$
2	$\varnothing$	$\varnothing$	?	$\{2,?\}$
3	$\varnothing$	?	$\varnothing$	$\{3,?\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 4: Precedence partial hypergroupoid defined on $V = \{1, 2, 3, 4\}$

- all diagonal results are $\emptyset$,
- $1 *_P 4 = \{1, 2, 3\} = V - \{4\}$,

where

- $\{1, ?\}$ denotes that the corresponding result contains element 1, but it may contain other elements too,
- ? denotes that the corresponding result is totally unknown.

So, we have the following options:

iii-(1) $2 *_P 3 = \emptyset$ and $3 *_P 2 = \emptyset$, which leads to the partial hypergroupoid $(V, *_P)$ given by Table 5.

	1	2	3	4
1	$\varnothing$	$\{1\}$	$\{1\}$	$\{1,2,3\}$
2	$\varnothing$	$\varnothing$	$\varnothing$	$\{2\}$
3	$\varnothing$	$\varnothing$	$\varnothing$	$\{3\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 5: 1st precedence partial hypergroupoid defined on $V = \{1, 2, 3, 4\}$

iii-(2) $3 *_P 2 \neq \emptyset$, which leads to the partial hypergroupoid $(V, *_P)$ given by Table 6.
We notice that $3 *_P 2 \neq \emptyset$ leads to $2 *_P 3 = \emptyset$, according to Proposition 3.1-i. It also holds $3 \in 1 *_P 2$ and $2 \in 3 *_P 4$, according to Proposition 3.3-ii. Then $3 *_P 2 = (1 *_P 2) \cap (3 *_P 4)$.

iii-(3) $2 *_P 3 \neq \emptyset$, which leads to the partial hypergroupoid $(V, *_P)$ given by Table 7.
We notice that $2 *_P 3 \neq \emptyset$ leads to $3 *_P 2 = \emptyset$, according to Proposition 3.1-i. It also holds $2 \in 1 *_P 3$ and $3 \in 2 *_P 4$, according to Proposition 3.3-ii. Then $2 *_P 3 = (1 *_P 3) \cap (2 *_P 4)$.

	1	2	3	4
1	$\varnothing$	$\{1,3\}$	$\{1\}$	$\{1,2,3\}$
2	$\varnothing$	$\varnothing$	$\varnothing$	$\{2\}$
3	$\varnothing$	$\{3\}$	$\varnothing$	$\{2,3\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 6: 2nd precedence partial hypergroupoid defined on $V = \{1, 2, 3, 4\}$

	1	2	3	4
1	$\varnothing$	$\{1\}$	$\{1,2\}$	$\{1,2,3\}$
2	$\varnothing$	$\varnothing$	$\{2\}$	$\{2,3\}$
3	$\varnothing$	$\varnothing$	$\varnothing$	$\{3\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 7: 3rd precedence partial hypergroupoid defined on $V = \{1, 2, 3, 4\}$

In all three options results $2 *_P 4$ and $3 *_P 4$ are obtained through Proposition 3.2-iv, Proposition 3.3-iii and Remark 3.1-(3).

One can see that the previous three hyperoperations satisfy the conditions of Definition 3.1. So, they are precedence hyperoperations. Consequently, the only precedence partial hypergroupoids that can be defined on $V = \{1, 2, 3, 4\}$ are the precedence partial hypergroupoids described by the previous three tables.

□

4. Precedence graphs

Definition 4.1. *Given a precedence partial hypergroupoid $(V, *_P)$ we construct a binary relation $R \subseteq V \times V$ in the following way:*

For every $x, y \in V$, it holds

$$(x, y) \in R \quad \textit{if and only if} \quad x *_P y \quad \textit{is a singleton.} \tag{5}$$

The above binary relation R defines a directed graph $G = (V, R)$ called a "precedence graph associated to the precedence partial hypergroupoid $(V, *_P)$".

Vice versa, given a precedence graph $G = (V, R)$ we can assume the precedence partial hypergroupoid $(V, *_P)$ that defined the graph $G = (V, R)$. Indeed the hyperoperation $*_P : V \times V \to \wp(V)$ defined as follows:

For every $x, y \in V$

$$x *_P y := \{\text{all vertices of all paths of graph } G \text{ from } x \text{ to } y, \text{ except } y\} \tag{6}$$

defines a precedence partial hypergroupoid $(V, *_P)$, whose associated precedence graph is $G = (V, R)$.

Remark 4.1. In case $V = \{1, 2, 3, 4\}$ all three precedence partial hypergroupoids $(V, *_P)$ (s. Proposition 3.4-iii) define the precedence graphs shown in Figure 2.

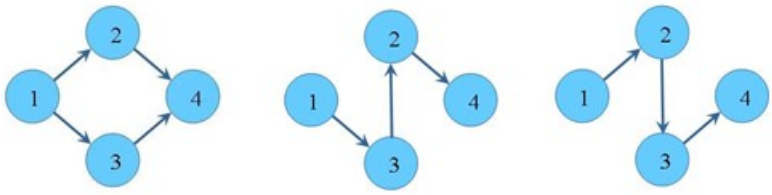

Figure 2: Precedence graphs defined by precedence partial hypergroupoids on $V = \{1, 2, 3, 4\}$

Given a graph $G = (V, R)$ the hyperoperation $*_P$ defined by (6) can check, if the graph is a precedence graph or not. In case the graph is not a precedence graph, the hyperoperation $*_P$ can derive a precedence graph from the given graph.

In the following we give two necessary conditions for a graph to be a precedence graph.

Proposition 4.1. *A precedence graph $G = (V, R)$ has no walks.*

Proof. Let $G = (V, R)$ be a precedence graph defined by a precedence hypergroupoid $(V, *_P)$ according to (5). Let $G = (V, R)$ has a walk. Every walk contains at least one circle. If i is a vertex of this circle, then $i \in i *_P i$ and so, $i *_P i \neq \emptyset$. Consequently, $(V, *_P)$ is not a precedence partial hypergroupoid (s. (1)) and graph $G = (V, R)$ is not a precedence graph. □

Proposition 4.2. *A precedence graph $G = (V, R)$ has only induced paths.*

Proof. Let $G = (V, R)$ be a precedence graph defined by a precedence hypergroupoid $(V, *_P)$ according to (5). Let also i, j be two nonadjacent vertices of a path connected by an edge i.e. $(i, j) \in R$. Then, there is at least one vertex x so that $\{i, x\} \subseteq i *_P j$. So, $i *_P j$ is not a singleton. Then (5) implies that $(i, j) \notin R$, which is a contradiction. □

5. Assembly line design using a precedence hyperoperation

Designing an assembly line one must first set the tasks [1] needed for the product to be assembled and then assign these tasks to workstations standing in a row.

1. Small element of work that cannot be conveniently fragmented further.

The fundamental problem of the assembly line design, the so-called "*assembly line balancing problem (ALB)*", is to manage this assignment to satisfy the precedence relations among the tasks and to minimize the idle time (s. [18]).

To illustrate the need of precedence relations, we usually give the standard example: "A bottle can't be filled, when the cap is already on." The precedence relations are usually visualized by a precedence graph first developed by Salveson [23].

Given a precedence graph and the task times [2] we must work out a process flow diagram, which is the final assignment of the tasks to workstations standing in a row. We assume that every workstation has the same time to complete its own set of tasks ("*Cycle Time*" or "*Task Time*"). If one task needs more time to be completed, more workers can be set at its workstation in order to reduce the working time. Furthermore some tasks with short task times can be assigned to the same workstation in order for the working time among the workstations to be balanced. So, we shall now concentrate on working out the assembly line design based on the precedence relations of the tasks only. For this purpose we shall use the precedence hyperoperation introduced in Section 3.

We consider as an example a "*one-model assembly line*" which consists of 8 tasks. The precedence relations among the tasks are given by the graph $G = (V, R)$ shown in Figure 3. In order for one unit of the model to be assembled all 8 tasks of the graph have to be completed.

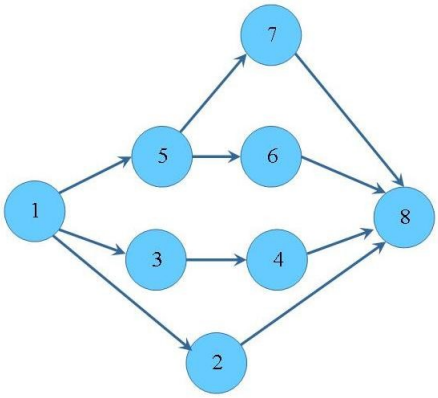

Figure 3: Example of a precedence graph in one-model assembly line

We define the precedence hyperoperation (6) on the set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ of the vertices of the graph $G = (V, R)$. So, the associative precedence partial hypergroupoid $(V, *_P)$ is defined by Table 8, where $a = 1$ is the starting element and $b = 8$ the ending element. In what follows the elements of V are also called "*vertices*".

2. Time needed to complete one task by a well trained worker.

	1	2	3	4	5	6	7	8
1	$\varnothing$	$\{1\}$	$\{1\}$	$\{1,3\}$	$\{1\}$	$\{1,5\}$	$\{1,5\}$	$\{1,2,3,4,5,6,7\}$
2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{2\}$
3	$\varnothing$	$\varnothing$	$\varnothing$	$\{3\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{3,4\}$
4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{4\}$
5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{5\}$	$\{5\}$	$\{5,6,7\}$
6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{6\}$
7	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{7\}$
8	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 8: $(V, *_P)$ defined by the graph $G = (V, R)$ (s. Figure 3)

In order to construct a process flow diagram we introduce the following algorithm, that we call "*Fewer-Descendants-Vertex First algorithm*" or shorter "*FDVF algorithm*":

- **Step 1.** We check all results $a *_P i$ of the starting vertex row and we collect all singletons. The corresponding vertices will be executed exactly after the starting vertex a. The order of the execution of these vertices will be chosen according to the increasing number of their descendants. So, we collect their complementary results to $a *_P b$ in a set I and we order the elements of I according to their increasing cardinality. If $i *_P b$ is the result we found with the minimum cardinality and $n = card(i *_P b)$, then vertex i is necessary for fewer of the following tasks and this is the vertex we will choose first after the starting vertex. Then, we continue with the rest vertices. If more than one elements of I have the same cardinality, then we choose an arbitrary one of the corresponding vertices.

  In our example, we check all results of the starting vertex row and we collect $1 *_P 2, 1 *_P 3, 1 *_P 5$. So, after vertex 1 we must execute the vertices 2, 3 and 5. To choose their order of execution, we find $card(2 *_P 8) < card(3 *_P 8) < card(5 *_P 8)$ and so, we choose vertex 2, since it is necessary for fewer of the following tasks. After vertex 2 we will execute vertex 3 and then vertex 5. In case $card(2 *_P 8) = card(3 *_P 8) = card(5 *_P 8)$ we would randomly choose the sequence of the vertices 2, 3 and 5. Consequently, our process flow diagram so far is shown in Figure 4.

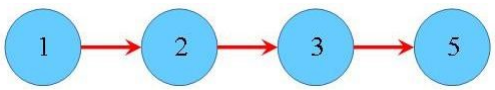

Figure 4: Process flow diagram after step 1 of FDVF algorithm

- **Step 2.** We check the number of descendants of the last executed vertex. If it is equal to 1, then we go to Step 4, otherwise we go to the next Step.

  In our example, we check $card(5 *_P 8) = 3 > 1$, so we go to the next step.

- **Step 3.** Now we check all results $i *_P b, j \neq b$, of the rows of the vertices executed in the previous step and we collect all singletons. The corresponding vertices will be executed exactly after the last executed vertex. The order of the execution of these vertices will be chosen according to the increasing number of their descendants. So, we collect their complementary results to $i *_P b$ in a set I and we order the elements of I according to their increasing cardinality. If more than one elements of I have the same cardinality, then we choose an arbitrary one of the corresponding vertices. Then we go to Step 2.

  In our example, we check all results of the rows 2, 3 and 5 executed in the previous step and we collect $3 *_P 4, 5 *_P 6, 5 *_P 7$. So, after vertex 5 we must execute the vertices 4, 6 and 7. To choose their order of execution, we find $card(4 *_P 8) = card(6 *_P 8) = card(7 *_P 8)$ and so, we randomly choose the sequence of the vertices 4, 6 and 7. Consequently, our process flow diagram so far is shown in Figure 5.

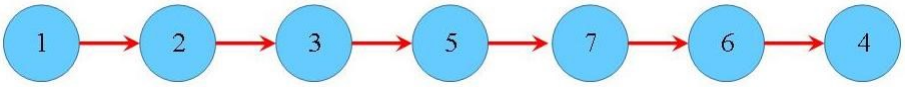

Figure 5: Process flow diagram after step 3 of FDVF algorithm

We go to Step 2 and we check $card(4 *_P 8)$. Since $card(4 *_P 8) = 1$, we go to Step 4.

- **Step 4.** We execute the last vertex i.e. the ending vertex b.

  In our example, we execute vertex 8.

Consequently, our final process flow diagram is shown in Figure 6.

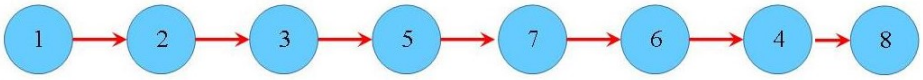

Figure 6: Process flow diagram of FDVF algorithm

Figure 7 shows the initial precedence graph (on the left) and the process flow diagram constructed by the FDVF-algorithm (on the right):

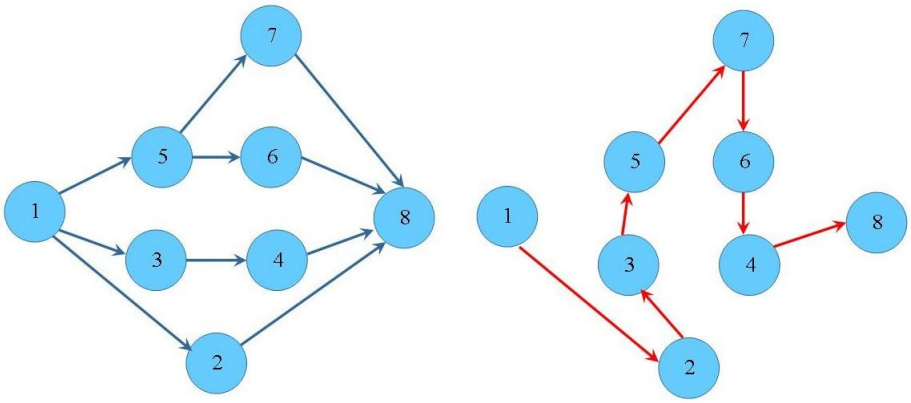

Figure 7: Process flow diagram of FDVF algorithm

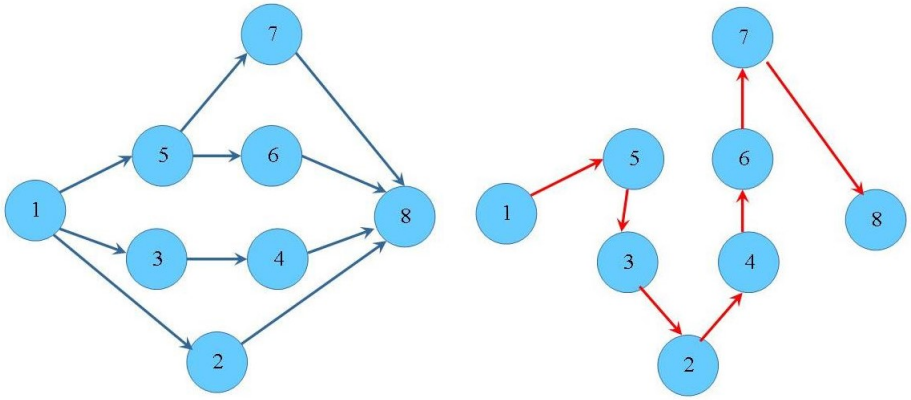

Figure 8: Process flow diagram of MDVF algorithm

Remark 5.1. The "*More-Descendants-Vertex First algorithm*" or shorter "*MDVF algorithm*" that can be defined in a similar way, will construct a similar process flow diagram illustrated in Figure 8.

One can choose FDVF or MDVF process flow diagram considering complementary benefits, such as raw material or skilled workers available at a time.

Remark 5.2. After the process flow diagram design follows the final assignment of the tasks to an ordered sequence of workstations. Some tasks with short task times can be assigned to the same workstation in order for the working time among the workstations to be balanced.

Figure 9 shows an example of a final assignment of the tasks of the process flow diagram of Figure 6 to an ordered sequence of workstations. Every square represents a workstation. In this example task times of 2 and 3 (resp. 6 and 4)

supposed to be short enough, so that their sum is almost equal to the task time of the previous and the next workstation. This is a reason why tasks 2 and 3 (resp. 6 and 4) are assigned to the same workstation.

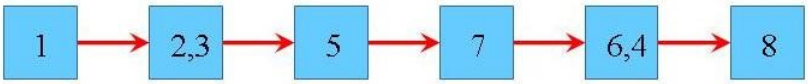

Figure 9: Assignment of the tasks to an ordered sequence of workstations

Remark 5.3. In our case study we examined "*a one-model assembly line*" i.e. in order for one unit of a model to be assembled all tasks of the precedence graph have to be completed. In case of "*a mixed-model assembly line*" we follow the same procedure: Every product unit passes through all tasks of the precedence graph (i.e. all tasks of the process flow diagram) and just skip the tasks that are not required for its specific model.

Discussion

Our research proposes an algorithm of constructing a process flow diagram starting from a precedence partial hypergroupoid of a precedence graph. The examples given deal with sets of tasks with small cardinalities. However, in the real-world implementation we have to deal with sets of tasks with big cardinalities. Then, the multiplicative tables of the associative precedence partial hypergroupoids are big, although they are "half" empty. This fact may put limitations on the application of the algorithm. A solution to the problem would be the partition of the total precedence graph into subgraphs and the application of the algorithm to each subgraph. Then, each subgraph could be considered as a simple task of the total precedence graph and the algorithm could be applied to the simplified form of the total precedence graph.

References

[1] P. Corsini, *Hypergraphs and hypergroups*, Algebra Universalis, 35 (1996), 548-555.

[2] P. Corsini, *On the hypergroups associated with binary relations*, Multi. Val. Logic, 5 (2000), 407-419.

[3] P. Corsini, *Binary relations and hypergroupoids*, Ital. J. Pure Appl. Math., 7 (2000), 11-18.

[4] P. Corsini, V. Leoreanu, *Hypergroups and binary relations*, Algebra Universalis, 43 (2000), 321-330.

[5] P. Corsini, V. Leoreanu, *Applications of hyperstructure theory*, in: Advances in Mathematics, Kluwer Academic Publishers, 2003.

[6] P. Corsini, V. Leoreanu, *Survey on new topics on hyperstructure theory and its applications*, in: Proc. of 8th Internat. Congress on AHA, 2003, 1-37.

[7] I. Cristea, M. Stefanescu, *Binary relations and reduced hypergroups*, Discrete Math., 308 (2008), 3537-3544.

[8] I. Cristea, M. Stefanescu, *Hypergroups and n-ary relations*, European J. Combin., 31 (2010), 780-789.

[9] I. Cristea, M. Stefanescu, C. Anghluta, *About the fundamental relations defined on the hypergroupoids associated with binary relations*, European J. Combin., 32 (2011), 72-81.

[10] B. Davvaz, V. Leoreanu-Fotea, *Hypergroup theory*, World Scientific, 2022.

[11] M. De Salvo, J. Lo Faro, *A new class of hypergroupoids associated to binary relations*, J. Mult.-Valued Logic Soft Comput., 9 (2003), 361-375.

[12] Y. Feng, P. Corsini, *On fuzzy Corsini's hyperoperations*, Int. J. Appl. Math., 2012 (2012), 9 pages.

[13] A. Kalampakas, S. Spartalis, K. Skoulariki, *Directed graphs representing isomorphism classes of C-hypergroupoids*, Ratio Mathematica, 23 (2012), 51-64.

[14] A. Kalampakas, S. Spartalis, A. Tsigkas, *The path hyperoperation*, An. Ştiinţ. Univ. "Ovidius" Constanţa Ser. Mat., 22 (2014), 141-153.

[15] A. Kalampakas, S. Spartalis, *Path hypergroupoids: commutativity and graph connectivity*, European J. Combin., 44 (2015), 257-264.

[16] A. Kalampakas, S. Spartalis, *Hyperoperations on directed graphs*, J. Discrete Math. Sci. Cryptogr., 27 (2024), 1011–1025.

[17] M. Konstantinidou, K. Serafimidis, *Sur les P-supertrillis*, in: T. Vougiouklis (Ed.), New Frontiers in Hyper-Structures and Rel. Algebras, Hadronic Press, Palm Harbor, U.S.A., 1996, 139-148.

[18] N. Kriengkorakot, N. Pianthong, *The assembly line balancing problem: review articles*, KKU Engineering Journal, 34 (2007), 133-140.

[19] V. Leoreanu, L. Leoreanu, *Hypergroups associated with hypergraphs*, Ital. J. Pure Appl. Math., 4 (1998), 119-126.

[20] F. Marty, *Sur une generalization de la notion de groupe*, in: Proc. 8th Congress des Mathematiciens Scandinaves, (1934), 45-49.

[21] S. J. Rasovic, *On hyperrings associated with binary relations on semihypergroup*, Ital. J. Pure Appl. Math., 30 (2013), 279-288.

[22] I. Rosenberg, *Hypergroups and join spaces determined by relations*, Ital. J. Pure Appl. Math., 4 (1998), 93-101.

[23] M.E. Salveson, *The assembly line balancing problem*, Journal of Industrial Engineering, 6 (1955), 18-25.

[24] S. Spartalis, *Hypergroupoids obtained from groupoids with binary relations*, Ital. J. Pure Appl. Math., 16 (2004), 201-210.

[25] S. Spartalis, *The hyperoperation relation and the Corsini's partial or not-partial hypergroupoids (a classification)*, Ital. J. Pure Appl. Math., 24 (2008), 97-112.

[26] S. Spartalis, A. Kalampakas, *Graph hyperstrustures,* in: Proc. of 12th Internat. Congress on AHA, 2014, 130-135.

[27] S. Spartalis, M. Konstantinidou, A. Taouktsoglou, *C-hypergroupoids obtained by special binary relations,* Comput. Math. Appl., 59 (2010), 2628-2635.

[28] T. Vougiouklis, *Hyperstructures and their representations*, Hadronic Press, Palm Harbor, U.S.A., 1994.

Accepted: October 16, 2024