Acta Ciencia Indica, Vol. XLI M, No. 2, 175 (2015)

THE DIRECTED HYPERGRAPHS IN GRAPH THEORY

A. RAMESH KUMAR

Head, Department of Mathematics, Srimad Andavan Arts & Science College, Trichy-05

AND

G. KAVITHA

Asst. Professor, Kongunadu College of Engineering and Technology, Tholurpatti, Thottiam

RECEIVED : 17 December, 2014

REVISED : 13 January, 2015

We study on directed hypergraphs as a tool to model and solve some classes of problems arising in Operations Research and in Computer Science. Concepts such as connectivity, paths and cuts are defined. An extension of the main duality results to a special class of hypergraphs is presented. We also saw that hypergraphs generalized standard graphs by defining edges between multiple vertices instead of only two vertices. Hence some properties must be generalization of graph properties and application of hypergraphs.

KEYWORDS: Directed hypergraph, Hyperpaths, connection, and cut sets.

MSC CODE: 05C65

Introduction

A directed graph is called digraph. Directed graphs arise in a natural way in many applications of graph theory. The street map of a city, abstract representation of computer programs and network flows can be represented only by directed graphs rather than by graphs. Directed graphs also are used in the study of sequential machines and system analysis in control theory.

Many of the concepts and terminology for graphs are also valid for digraphs. However, there are many concepts of digraphs involving the notion of orientation that apply only to digraphs. We discuss the condition under which one can direct the edges of a graph in such a way that the resulting digraph is strongly connected. Then we deal with the connection between digraphs of Hypergraphs and matrices.

A path or directed path is a walk in which all the vertices are distinct. A cycle or circuit is a nontrivial closed walk whose origin and interval vertices are distinct. The various types of Hypergraphs having in graph theory likewise, Dual hypergraphs, unimodular hypergraphs, balanced hypergraphs, Arbordal hypergraphs, Normal hypergraphs, mengerian hypergraphs and paranormal hypergraphs and so on.

Notations

- ϕ_j the frequency of line $l_j \in L_{I;}$
- Φ (L_i') the "combined" frequency of the lines-set L_i';
- π_j (L_i') the probability that a carrier serving line l_j will arrive at stop I before carriers serving other lines of L_i';
- T_j the expected travel time between stop I and the destination, if line l_j is used, not including the waiting time at I;
- w (L_i') the average waiting time at stop I.

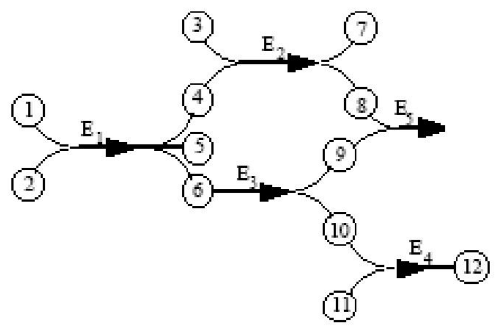
Directed hypergraphs

A hypergraph is a pair $H = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices (or nodes) and $E = \{E_1, E_2, ..., E_m\}$, with $E_i \subseteq V$ for $i = 1, \ldots, m$, is the set of hyperedges. Clearly, when $| E_i | = 2$, $i = 1,\ldots, m$, the hypergraph is a standard graph While the size of a standard graph is uniquely defined by n and m, the size of a hypergraph depends also on the cardinality of its hyperedges; we define the size of **H** as the sum of the cardinalities of its hyperedges: $size\ (H) = \Sigma\, E_i \in E \mid E_i \mid$.

It is worth noting that there is a one-to-one correspondence between hypergraphs and Boolean matrices. Indeed, any $n \times m$ matrix $A = [a_{ij}]$ such that $a_{ij} \in \{0, 1\}$ may be considered as the *incidence matrix* of a hypergraph H where each row i is associated with a vertex v_i and each column j with a hyperedge E_j.

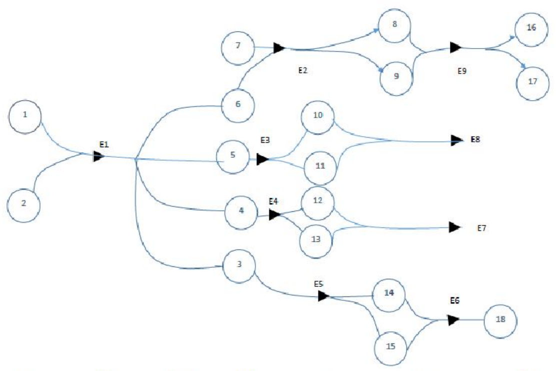
A directed hyperedge or hyperarc is an ordered pair, $E = (X, Y)$, of disjoint subsets of vertices; X is the tail of E while Y is its head. In the following, the tail and the head of hyper arc E will be denoted by T (E) and H (E), respectively.

A directed hypergraph is a hypergraph with directed hyperedges. In the following, directed hyper graphs will simply be called hypergraphs. An example of hyper graph is illustrated in Fig. 1. Note that hyperarc E_5 has an empty head.



	E_1	E_2	E_3	E_4	E_5
1	-1	0	0	0	0
2	-1	0	0	0	0
3	0	-1	0	0	0
4	1	-1	0	0	0
5	1	0	0	0	0
6	1	0	-1	0	0
7	0	1	0	0	0
8	0	1	0	0	-1
9	0	0	1	0	-1
10	0	0	1	-1	0
11	0	0	0	-1	0
12	0	0	0	1	0

Fig. 1. A hypergraph and its incidence matrix.



	E1	E2	E3	E4	E5	E6	E7	E8	E9
1	-1	0	0	0	0	0	0	0	0
2	-1	0	0	0	0	0	0	0	0
3	1	0	0	0	-1	0	0	0	0
4	1	0	0	-1	0	0	0	0	0
5	1	0	-1	0	0	0	0	0	0
6	1	-1	0	0	0	0	0	0	0
7	0	-1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	-1
9	0	1	0	0	0	0	0	0	-1
10	0	0	1	0	0	0	0	-1	0
11	0	0	1	0	0	0	0	-1	0
12	0	0	0	1	0	0	-1	0	0
13	0	0	0	1	0	0	-1	0	0
14	0	0	0	0	1	-1	0	0	0
15	0	0	0	0	1	-1	0	0	0
16	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	1	0	0	0

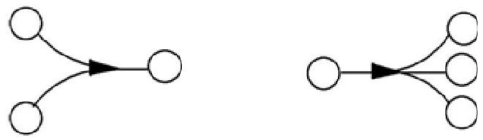
Fig. 2. A hypergraph and its incidence matrix.

Similarly Fig. 2, Note that hyperarc E_7 and E_8 has an empty head.

As for directed graphs, the incidence matrix of a hyper graph H is a next matrix $[a_{ij}]$ defined as follows (see Fig. 1):

$$a_{ij} = \begin{cases} -1 & \text{if } v_i \in T(E_j), \\ 1 & \text{if } v_i \in H(E_j), \\ 0 & \text{otherwise} \end{cases}$$

Clearly, there is a one-to-one correspondence between hyper graphs and (–1, 0, 1) matrices. A Backward hyper arc, or simply B-arc, is a hyper arc $E = (T(E), H(E))$ with $|H(E)| = 1$ (Fig. 3a). A Forward hyper arc, or simply F-arc, is a hyper arc $E = (T(E), H(E))$ with $|T(E)| = 1$ (Fig. 3b).



(a) **(b)**

Fig. 3. A *B*-arc (a) and a *F*-arc (b).

A B-graph (or B-hyper graph) is a hyper graph whose hyper arcs are B-arcs. A F-graph (or F-hyper graph) is a hyper graph whose hyper arcs are F-arcs. A BF-graph (or BF-hyper graph) is a hyper graph whose hyper arcs are either B-arcs or F-arcs.

Given a hyper graph **H = (V, E)**, we define its *symmetric image* the hyper graph $\tilde{H} = (V, \tilde{E})$ where $E = \{(X, Y): (Y, X) \in E\}$. Note that the symmetric image of a B-graph is a F-graph, and vice versa. It is always possible to transform a general hyper graph into a BF-graph, by adding a dummy node to each hyper arc which is neither a B-arc nor a F-arc, and thus replacing the hyper arc by one backward and one forward hyper arc (see Fig. 4).

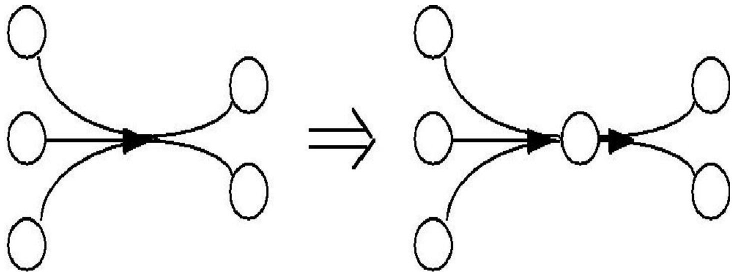


Fig. 4 - Transformation of a hyper arc into a *B*-arc and a *F*-arc.

Let $FS(v) = \{E \in E : v \in T(E)\}$ and $BS(v) = \{E \in E : v \in H(E)\}$ denote the *Forward Star* and the *Backward Star* of node v, respectively. B-graphs and F-graphs are of particular relevance in applications. Indeed, they have been introduced many times in the literature with various names. The labelled graphs [4] and B-graphs have been introduced [1]. F-graphs have been studied in the context of urban transit problems [8] and applications. Hyper graphs [9] and B-graphs, called directed hyper graphs and rule hyper graphs respectively, to represent deduction properties in data bases as paths in hyper graphs.

Paths, hyperpaths and connection

A *path* $P_{st,}$ of length q, in hyper graph $H = (V, E)$ is a sequence of nodes and hyper arcs $P_{st} = (v_1 = s, E_{i1}, v_2, E_{i2}, ..., EI_q, v_{q+1} = t)$, where : $s \in T(E_{i1})$, $t \in H(EI_q)$, and $v_j \in H(EI_{j-1}) \cap T(EI_j), j = 2, ..., q$.

Nodes s and t are the *origin* and the *destination* of P_{st}, respectively, and we say that t is *connected* to s. If $t \in T(E_{i1})$, then P_{st} is said to be a *cycle*; this is in particular true when $t = s$. In a *simple* path all hyper arcs are distinct, and a simple path is *elementary* if all nodes $v_1, v_2, \dots, v_{q+1}$ are distinct. Similarly we may define *simple* and *elementary cycles*. A path is said to be *cycle-free* if it does not contain any sub path which is a cycle.

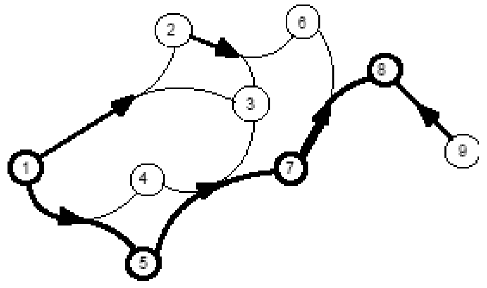


Fig. 5. A path $P_{1,8}$.

In Fig. 5, node 8 is connected to node 1, while node 9 is not. The elementary path connecting 8 to 1 is drawn in thick line.

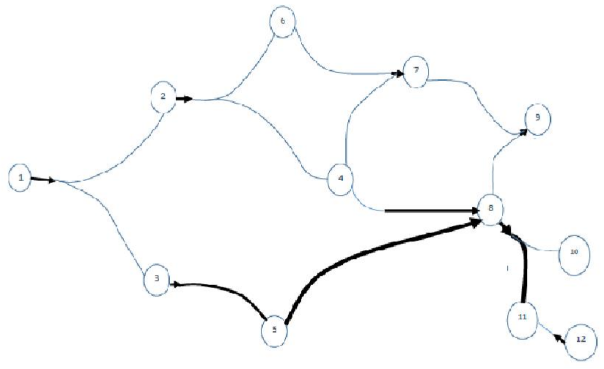


Fig. 6. A path $P_{1,11}$.

In Fig. 6, node 11 is connected to node 1, while node 12 is not. The elementary path connecting 11 to 1 is drawn in thick line. Consider a hyper graph $H = (V, E)$. A *B-path* (or *B-hyper path*) Π_{st} is a minimal hyper graph $H_\Pi = (V_\Pi, E_\Pi)$ such that:

(i) $E_\Pi \subseteq E$;

(ii) $s, t \in V\Pi = \cup E\ I \subseteq \mathrm{V}; EI \in E_\Pi$

(iii) $x \in V_\Pi \Rightarrow x$ is connected to s in H_Π by means of a cycle-free simple path.

We say that $H_\Pi = (V_\Pi, E_\Pi)$ is a *F-path* (or *F-hyper path*) from s to t if its symmetric image is a B-path from t to s. A *BF-path* (or *BF-hyper path*) from s to t is a hyper graph which is at the same time a B-path and a F-path from s to t. Node y is *B-connected* (*F-connected*, *BF-connected*) to node x if a B-path (F-path, BF-path) Π_{xy} exists in H.

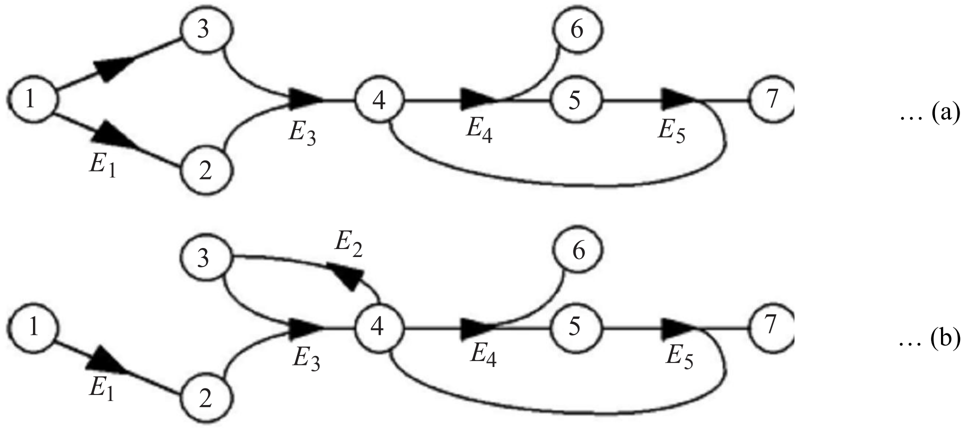


Fig. 7 - A B-path (a) and a B-graph which is not a B-path (b).

The hypergraph in Fig.7a is a B-path; note that the cycle (4, E_4, 5, E_5, 4) is not contained in any simple path from node 1 to node 7. On the contrary, the hyper graph in Fig. 7b is not a B-path because the only path connecting node 3 to the origin contains the cycle (2, E_3, 4, E_2, 3).

Cuts and cutsets

Let $H = (V, E)$ be a hyper graph and s and t be two distinguished nodes, the *source* and the *sink* respectively. A *cut* $T_{st} = (V_{s,\ Vt})$ is a partition of V into two subsets V_s and V_t such that $s \in V_s$ and $t \in V_t$. Given the cut T_{st}, its *cutest* E_{st} is the set of all hyper arcs E such that $T(E) \subseteq V_s$ and $H(E) \subseteq V_t$.

Such a cutest may be empty; see for instance the cut ({1, 2}, {3, 4, 5, 6, 7}) in the B-graph of Fig. 7b. The cardinality of a cut is the cardinality of its cutest. In Fig. 8 three cuts are indicated; the cardinality of T_{st} 1 is 2, while T_{st} 2 and T_{st} 3 have cardinality 1. Note that t is not necessarily disconnected from S by removing the hyperons of a cutest. For example, Fig 8 by removing the cutest T_{st} 1 we disconnect t from s, by removing the cutest of T_{st} 2 only the B-connection of t to s is lost, while t remains both connected and B-connected to s when we remove the cutest of T_{st} 3.

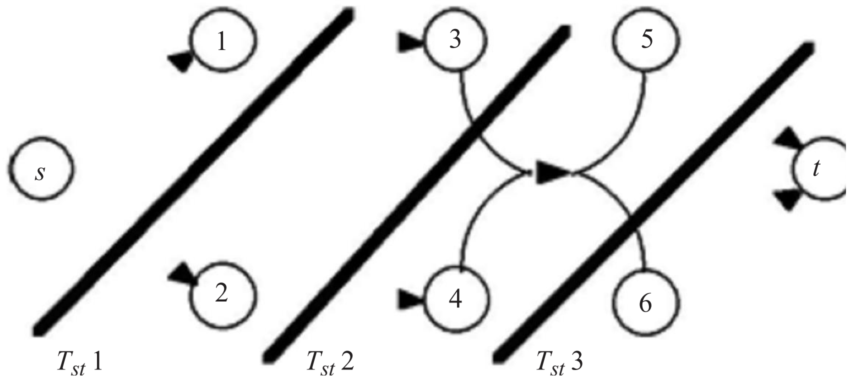


Fig. 8 - Only cut T_{st} 1 disconnects source s and sinks t.

Theorem 5.1. In a B-graph $H = (V, E)$, a cut T_{st} of cardinality 0 exists if and only if t is not B-connected to s.

Proof: ($\Rightarrow$) Assume that a cut T_{st} with an empty cutest E_{st} exists and there is a node $v \in V_t$ B-connected to s. Then a B-arc $E = (T(E), \{v\})$ must exist with the property that every node $x \in T(E)$ be B-connected to s (see Proposition 1). Clearly, as E_{st} is empty, at least one node $u \in T(E)$ must belong to V_t. By repeating the same argument on u, we may eventually conclude that s also belongs to V_t, which is a contradiction.

($\Leftarrow$) Now assume that t is not B-connected to s. Define V_s as the set of all the nodes B-connected to s and $\underline{V_t} = V \setminus V_s$. T_t is necessarily a cut of cardinality 0, for the existence of a B-arc $E = (T(E), \{v\})$ in the cut, being $T(E) \subseteq V_s$ and $v \in V_t$, imply the B-connection of v to s.

Theorem 5.2. In a hyper graph $H = (V, E)$ a cut T_{st} of cardinality 0 exists if and only if t is not super-connected to s.

Proof: ($\Rightarrow$) Let $T_{st} = (V_s, V_t)$ be a cut of cardinality 0. Consider the B-reduction H_B of H obtained by replacing each hyper arc E with a B-arc $(T(E), \{v\})$ with the condition that if $T(E) \subseteq V_s$ then also $v \in V_s$. This reduction is always possible since for any hyper arc E with $T(E) \subseteq V_s$ at least one node in its head must belong to V_s, otherwise E belongs to the cutest which, by hypothesis, is empty. By Theorem 1, t is not B-connected to s in H_B and therefore t is not super-connected to s in H.

($\Leftarrow$) If t is not super-connected to s, then a B-reduction exists such that t is not B-connected to s in it, the proof is completed.

Theorem 5.3. In a B-graph $H = (V, E)$ the following inequalities hold:

Min $\{|\ \Pi_{st}\ | : \Pi_{st}$ is a s-t B-path$\}$ $\geq$ maximum number of disjoint s-t cut sets $\geq$ min $\{|\ P_{st}\,| : P_{st}$ is a s-t path$\}$.

Proof : The first inequality follows directly from the fact that a cutest must contain at least a B-arc of every B-path, and then the number of disjoint s-t cut sets cannot exceed the cardinality of any B-path.

The second inequality can be proved as follows. Let V_k denote the set of nodes $\{I\}$ for which there exists a path P_{si} with cardinality $\leq k$. Clearly, if h is the minimum cardinality of the s-t paths, then we have $\{s\} = V_0 \subset V_1 \subset \ldots \subset V_h \subseteq V$; then $(V_0, V \setminus V_0)$, $(V_1, V \setminus V_1)$ … $(V_{h-1}, V \setminus V_{h-1})$ are s-t cuts with disjoint cut sets, for no B-arc with a tail node in V_i and the head in V_j with $j \geq i + 2$ may exist, and thus, no B-arc can belong to more than one cutest. This completes the proof.

Theorem 5.4. In a B-graph $H = (V, E)$ the following inequalities hold:

Max-number of disjoint s-t paths $\geq$ min $\{|\ E_{st}\ | : E_{st}$ is an s-t cutest$\}$ $\geq$ max-number of disjoint s-t B-paths.

Proof : Transform $H = (V, E)$ into a standard digraph $G = (V, A)$ where for each B-arc (X, y) there is a unique arc $(x, y) \in A$, with $x \in X$. The choice of $x \in X$ is arbitrary. It is easy to check that to any s-t cutest E_{st} in H corresponds a s-t cutest C_{st} in G with $|\ C_{st}\ | \geq |\ E_{st}\ |$; moreover, any set of k disjoint paths in G corresponds to a set of k disjoint paths in H, then the maximum number of disjoint paths in G is not larger than the maximum number of disjoint paths in H. Hence, from the well known max flow-min cut theorem for digraphs one has:

max-number of disjoint s-t paths in H $\geq$ max-number of disjoint s-t paths in G

$=$ min$\{|\ C_{st}\,| : C_{St}$ is a s-t cutest in $G\}$ $\geq$ Min $\{|\ E_{st}\,| : E_{st}$ is a s-t cutest in $H\}$.

The second inequality follows directly from the fact that, due to Theorem 1, any cutest must contain one B-arc from each B-path at least, and this completes the proof.

The following examples show that strict inequalities may hold in all cases.

In Fig. 9, a B-graph is presented for which the minimum cardinality of s-t B-paths is 5, the maximum number of disjoint s-t cut sets is only 4 and the minimum cardinality of s-t paths is 3.

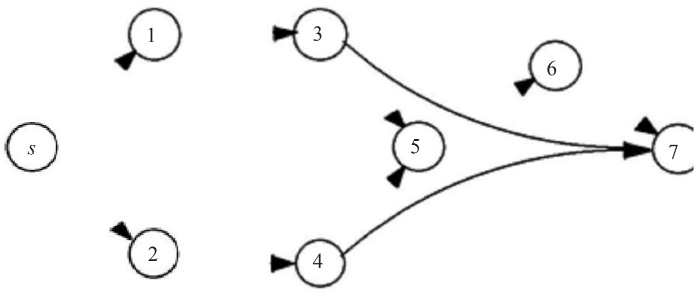


Fig. 9

In the B-graph of Fig. 10, the maximum number of disjoint s-t paths is 3, the minimum cardinality of s-t cuts is 2, and the maximum number of disjoint s-t B-paths is 1.

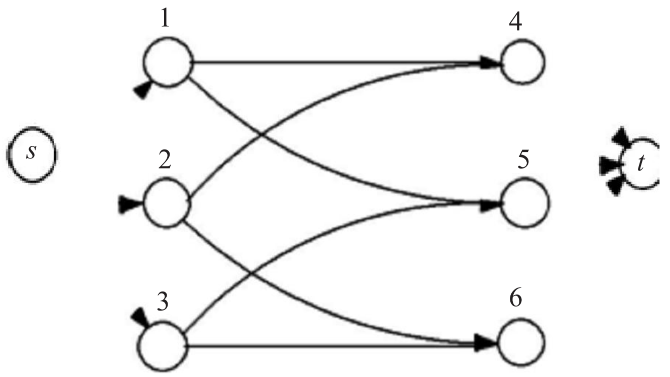


Fig. 10

WEIGHTED HYPERGRAPHS

6.1. Weighting functions

A weighted hypergraph is one in which each hyperarc E is assigned a real weight vector $w(E)$. Depending on the particular application, the components of $w(E)$ may represent costs, lengths, capacities etc. For the sake of simplicity, in the following we shall consider only scalar weights.

Given a B-path $\Pi = (V\Pi, E\Pi)$ from s to t, by *weighting function* we mean a node function $W\Pi$ which assigns weights to all its nodes depending on the weights of its hyperarcs. $W\Pi(t)$ is the *weight* of the B-path Π under the chosen weighting function.

We shall restrict ourselves to weighting functions for which $W\Pi(s) = 0$ and $W\Pi(y)$, for each $y \neq s$, depends only on the hyperarcs which *precede* y in the B-path Π, *i.e.* the hyperarcs belonging to all B-paths from s to y contained in Π.

A typical example of this kind of weighting function is the *cost*, $C\Pi$, defined as the sum of the weights of all the hyperarcs preceding node y in Π:

$$C\Pi(s) = 0;$$

$$C\Pi(y) = \Sigma w(E),\ y \in V_\Pi \backslash \{s\}, E \in \{E\Pi_{sy} : \Pi sy \subseteq \Pi\}$$

Clearly, $C\Pi(t) = \Sigma_E \in \mathrm{E}_\Pi w(E)$ is the *cost* of Π. This function is the usual cost in the graph setting, and the problem of finding a minimum cost B-path is a natural generalization of the minimum cost path problem.

When the weights are all equal to 1, the cost of Π is its *cardinality*.

A relevant class of weighting functions is the one in which the weight of node y can be written as a function of both the weights of the hyperarcs entering into y and that of the nodes in their tails:

$$W\Pi(y) = \min\{w(E) + F\Pi(T(E)) : E \in E_\Pi \cap BS(y)\},\ \ y \in V_\Pi \backslash \{s\}, \quad \ldots(1)$$

where $F\Pi(T(E))$ is a function of the weights of the nodes in $T(E)$:

$$F\Pi(T(E)) = F(\{W\Pi(x) : x \in T(E)\}),\ \ E \in E_\Pi, \quad \ldots(2)$$

where F is a non-decreasing function of $W\Pi\,(x)$ for each $x \in T\,(E)$. Such weighting functions will be called *additive weighting functions*.

In the particular case of B-graphs, the B-paths have the property that there is only one B-arc E entering into every node $y \neq s$; in this case (1) becomes:

$$W\Pi\,(y) = w\,(E) + F\Pi\,(T\,(E)),\;\; y \in V_{\Pi}\backslash\{s\} \qquad \dots (1')$$

Two particular additive weighting functions which have been presented in the literature in the context of some relevant applications of hypergraphs are the *distance* and the *value*. Given a s-t B-path $\Pi = (V\Pi, E\Pi)$, the *distance* in Π from s to all the nodes $y \in V\Pi\backslash\{s\}$ which are B-connected to s, $D\Pi\,(y)$, is defined by the following recursive equations:

$$D\Pi\,(s) = 0 \qquad \dots (3)$$

$$D\Pi\,(y) = \min\{l\,(E) + \max\{D\Pi\,(x): x \in T\,(E)\} : E \in E_{\Pi} \cap BS\,(y)\}, y \in V_{\Pi}\backslash\{s\};$$

where $l\,(E)$ is the *length* of hyperarc E.

For B-graphs, equation (3) becomes:

$$D\Pi\,(y) = l\,(E) + \max\,\{D\Pi\,(x) : x \in T\,(E)\},\quad y \in V_{\Pi}\backslash\{s\} \qquad \dots (3')$$

In the case of unit hyperarc lengths, *i.e.* $l\,(E) = 1\;\forall\; E \in E$, the distance will be called *depth*. B-graphs [4] in the context of the satisfiability analysis of propositional Horn formulae. In this case, procedure **B-Visit**, with the use of function ν and a breadth-first search strategy, finds the minimum depth B-tree in $\boldsymbol{O}$ (*size* (H)) time.

The *value*, $V\Pi$, defined by Leontiev flow problem for the case of B-graphs [2], is the solution of the following recursive equations:

$$V\Pi\,(s) = 0;$$

$$V\Pi\,(y) = c\,(E) + \sum_{X \in T\,(E)} a\,(x, E)\, V\Pi\,(x),\quad E \in \mathrm{E}_{\Pi} \cap BS\,(y),\quad y \in V_{\Pi}\backslash\{s\}; \qquad \dots (4)$$

where $c\,(E)$ is the *cost* of B-arc E and, for each E and each $x \in T\,(E)$, $a\,(x, E)$ is a non-negative real coefficient.

Application of hypergraphs

6.1. Satisfiability

Let P be a set of n atomic *propositions*, which can be either *true* or *false*, and denote by t a proposition which is always *true*, and by f a proposition which is always *false*. Let C be a set of m *clauses*, each of the form:

$$p_1 \vee p_2 \vee \dots \vee p_r \leftarrow p_{r+1} \wedge p_{r+2} \wedge \dots \quad \wedge p_q, \qquad \dots (7)$$

where, for $I = 1 \dots Q$, $p_i \in P$. The meaning of (7) is that at least one of the propositions $p_1 \dots p_r$

Must be *true* when all the propositions $p_{r+1} \dots p_q$ are *true*. If this is the case, the clause is *true*; otherwise ($p_1 \dots p_r$ are all *false*, and $p_{r+1} \dots p_q$ are *true*) the clause is *false*. The disjunction $p_1 \vee p_2 \vee \dots \vee p_r$ is also called the *consequence* of the clause, while the conjunction $p_{r+1} \wedge p_{r+2} \wedge \dots \wedge p_q$ is called the *applicant*. We allow for $r = 0$, in which case the consequence is replaced by f, and for $r = q$, in which case the implicate is replaced by t.

Clause (7) can be easily converted into *disjunctive form*:

$$p_1 \vee p_2 \vee \ldots \vee p_r \vee \neg p_{r+1} \vee \neg p_{r+2} \vee \ldots \vee \neg p_q,$$

A *truth evaluation* is a function $\mathbf{v} : P \rightarrow \{false, true\}$. If there is a truth evaluation which makes all the clauses *true*, then C is said to be *satisfiable*, otherwise it is *unsatisfiable*.

The *satisfiability problem* (*SAT*) is defined as follows:

Input: A set P of n propositions, and a set C of m clauses over $P \cup \{F, t\}$;

Output: "**yes**" if C is satisfiable, "**no**" otherwise.

Most often, in the case of *yes-instances*, a truth evaluation which satisfies C is also desired.

A particularly important case is when a clause contains only one atomic proposition, *i.e.* $r \leq 1$ in (6). Such clause is called a *Horn clause*.

It is well known that *SAT* is *NP*-complete [8]. Either *NP*-complete, or *NP*-hard, are also most of its variants such as *k-SAT* (each clause contains $k \geq 3$ atomic propositions at most) and *Max-SAT* (the maximization of the number of satisfied clauses, or equivalently, the minimization of the number of clauses to be dropped in order to make the remaining clauses satisfiable). A notable exception is the case in which C contains only Horn clauses. In this case the satisfiability problem (*HORN-SAT*) is polynomial: in fact it can be solved in linear time [4] .Unfortunately, *Max-HORN-SAT* remains *NP*-hard [6].

HORN-SAT is the set of the instances of *SAT* whose clauses are Horn clauses.

To any given instance $\pi \in SAT$ we can associate the hyper graph Hπ with one node for each element of $P \cup \{f, t\}$ and one hyper arc E with $H(E) = \{p_1, p_2, \ldots, p_r\}$ and $T(E) = \{p_{r+1}, p_{r+2}, \ldots, p_q\}$. For each clause $p_1 \vee p_2 \vee \ldots \vee p_r \leftarrow p_{r+1} \wedge p_{r+2} \wedge \ldots \wedge p_q$. Clearly, from the definition, if $\pi \in HORN\text{-}SAT$ then $H\pi$ is a *B*-graph. Note that the labeled graphs [4] to represent *HORN-SAT* instances have a direct interpretation as *B*-graphs.

Theorem 6. An instance $\pi \in SAT$ is satisfiable if and only if the associated hyper graph $H\pi$ has a cut T_{tf} with cardinality 0.

Proof : ($\Rightarrow$) if π is satisfiable, then a truth assignment $\mathbf{v}$ exists which makes all the clauses in π *true*. Consider the cut $T_{tf} = (V_t, V_f)$ with :

$$V_t = \{p : \mathbf{v}(p) = true\} \cup \{t\} \quad \text{and} \quad V_f = \{p : \mathbf{v}(p) = false\} \cup \{f\}.$$

We claim that T_{tf} has cardinality 0; in fact the existence of a hyper arc E with $T(E) \subseteq V_t$ and $H(\mathrm{E}) \subseteq V_f$ would imply the existence of a clause made *false* by $\mathbf{v}$.

($\Leftarrow$) Let $T_{tf} = (V_t, V_f)$ be a cut with 0 cardinality. It is easy to check that the function:

$$v(p) = \begin{cases} true & \text{if} \in V_t, \\ false & \text{if } p \in V_f \end{cases}$$

is a truth assignment which makes all the clauses of π *true*.♦

6.2. Relational data bases

In the last years a substantial amount of research has been devoted to the analysis of relational data bases using graph related techniques.

A *Relational Data Base* (*RDB*) is often represented by a set of relations over a certain domain of attribute values, together with a set of Functional Dependencies.

Functional Dependencies have been studied by means of several types of generalized graphs, such as FD-graphs, Implication Graphs, Deduction Graphs, etc.

Let N be the set of attributes of a RDB. A *Functional Dependency* $F(X, Y)$, with both X and Y subsets of N, defines uniquely the value of the attributes in Y once the value of the attributes in X is given.

A set of Functional Dependencies together with some *inference rules* allows us to derive new facts from that explicitly stored in the data base. Typical inference rules [10].

(i) reflexivity : $F(X, Y)$ if $Y \subseteq X$;

(ii) transitivity : $F(X, Z)$ if $F(X, Y)$ and $F(Y, Z)$;

(iii) Conjunction: $F(X, Y \cup Z)$ if $F(X, Y)$, $F(X, Z)$.

Given a set of Functional Dependencies, F, we might need to solve problems such as:

(a) find whether a given Functional Dependency $F(X, Y) \notin F$ can be *derived* from F based on inference rules;

(b) Given a set of attributes $X \in F$, find its *closure* with respect to F, *i.e.* find the largest set X^* such that $F(X, X^*)$ either belongs to or can be derived from F.

Here we show briefly that hyper graphs provide a natural and unifying formalism to deal with most problems arising in the analysis of Functional Dependencies in RDB.

A set F of Functional Dependencies on the attribute set N can be represented by a hyper graph

$H = (V, E)$, with $V = N$ and $E = \{(X, Y \backslash X) : F(X, Y) \in F, Y \subseteq X\}$. It is easy to see that a B-path on H corresponds to a sequence of implications based on rules (i), (ii) and (iii). For example, the B-path of Fig. 11 corresponds to the derivation of $F(\{1, 2, 3, 4\}, \{9, 10\})$ starting from the implication relationships $F(\{2\}, \{5\})$, $F(\{3, 4\}, \{6, 7, 8\})$, $F(\{5, 7\}, \{9\})$ and $F(\{4, 8\}, \{10\})$, where attributes are denoted by natural numbers.

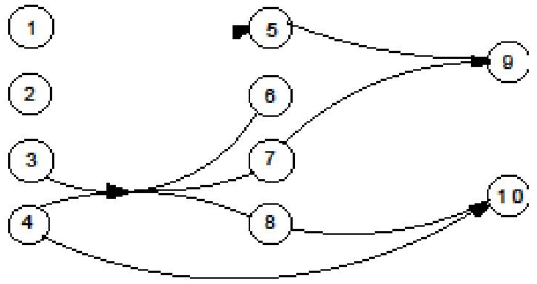


Fig. 10. A *B*-path representation of a sequence of implications.

Procedure ***B*-Visit** solves problems (a) and (b) in $\boldsymbol{O}(size(H)) = \boldsymbol{O}(size(F))$ time. In both cases, the set Q used in ***B*-Visit** is initialized to X. Let X' be the set of nodes visited by the procedure, *i.e.* the set of nodes B-connected to X. In problem (a), the answer is that $F(X, Y)$ is derivable from F if and only if $Y \subseteq X'$, while in problem (b) the answer is $X^* = X'$.

When the set Y is a singleton, *i.e.* the Functional Dependency is of the type $F(X, y)$ where $y \in N$. The directed hyper graphs representing sets of Functional Dependencies of this type are B-graphs [4, 6]. where several problems on sets of Functional Dependencies are defined, and graph algorithms for their solution presented. All these algorithms have a natural interpretation in terms of hyper graph algorithms.

6.3. Urban transit application

The analysis of passenger distribution in a transit system is an interesting application of *F*-graphs. A transit system can be modeled as a special network in which transit lines are superimposed on a ground network. Each transit line is a circuit, *i.e.* a close alternating sequence of nodes representing the *line-stops* and *arcs* representing the *in-vehicle* line segments. The ground network is formed by nodes representing geographical points in the urban area, and arcs representing *walking paths* between centurions and/or stops. For each stop node *i* on the ground network, let L_i be the set of lines which stop at *i*. Each node *i* will be connected to the corresponding nodes on the lines belonging to L_i by a *leaving arc* and a *boarding arc*. An example is given in Fig. 11 on a local standpoint.

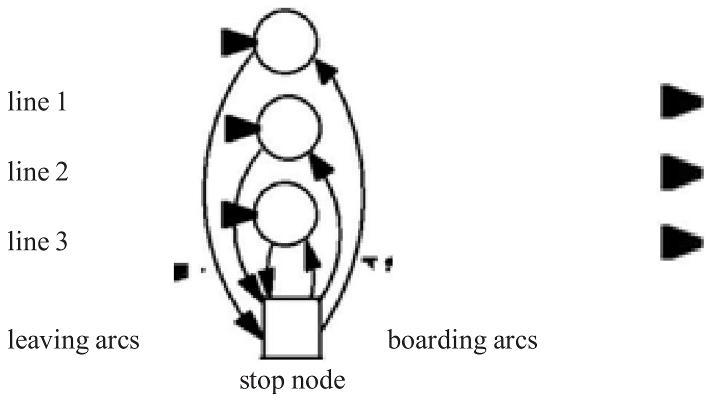


Fig. 11. A stop served by three lines.

Consider a passenger waiting at a stop *I*, who wishes to reach his/her destination *s* with the least expected travel time. The problem consists in determining the optimal subset $L^*_I \subseteq L_I$, the so called *attractive set*, such that by always boarding the first carrier of these lines arriving at the stop, the expected travel time will be minimized.

In general, the travel time's t_j are composed of walking times, in-vehicle travel times and waiting times associated with transfers from one line to another which can occur in the sequel of the trip. These times are the lengths of the associated arcs of the network; the lengths of in-vehicle arcs are the corresponding carrier travel times, the lengths of walking arcs are walking times, and the lengths of leaving arcs are set to 0. The waiting times are associated with boarding arcs; the value of a boarding arc (*I, j*) from a stop *I* to the corresponding line-stop of line l_j depends on the subset of lines L_i' considered. Moreover, all the boarding arcs of lines belonging to L_i' have the same length, which is the *average waiting time w* (L_i').

Under reasonable hypotheses on the distribution of passenger and carrier arrivals at the stops, the following results are obtained:

$$\Phi(L') = \Sigma_{l,t}^{t\in L}\, \phi_j, \qquad w(L_i') = \frac{1}{2\Phi(L_i')}, \qquad \pi_j(L_i') = \frac{\phi_j}{\Phi(L_i')}$$

and the expected travel time between stop *i* and the destination, when the set L_i' is selected, is:

$$T(L_i') = w(L_i') + \Sigma_{li} \in L^{tj}\pi_i(L_i') = \frac{1}{2\Phi(L_i')} + \Sigma_{l\in L_l'}\frac{t_i\phi}{\Phi(L_i')} = \frac{\frac{1}{2} + \Sigma_{l_i} \in L'^t \phi j}{\Phi(L_i')}$$

The optimal set L^*_I is the subset of L_i which minimizes the expected travel time : $T(L^*_I) = \min\{T(L_i'): L_i' \subseteq L_i\}$.

When travel times t_j for every $l_j \in L_I$ are known, the optimal set L^*_I is easily found with a local greedy algorithm. This algorithm works as follows: first, sort the lines in non-decreasing order of travel times, and then iteratively insert the lines one by one into L^*_I until a line l_j for which $t_j > T(L^*_{I;})$ am found [8].

The global problem is that of determining the least expected travel times t_r for every origin r and a given destination s. To solve this, the least expected travel times t_j for every $l_j \in L_i$ and the optimal sets L^*_I for all stops I must be computed simultaneously.

For this purpose, F-graphs have been introduced to represent transit networks; boarding arcs corresponding to L_i' may be modeled by a boarding F-arc $E(L_i')$ with length $w(L_i')$. The resulting F-graph is said *full* because if there is a F-arc $E = (\{i\}, H(E))$, then each $E' = (\{i\}, H(E'))$ with $E' \subseteq E$ also exists. E' is called a *contained* F-arc. The contained F-arcs are treated implicitly to keep the size of the F-graph at a reasonable level.

Let $H = \square V, E)$ be the F-graph in which contained F-arcs are omitted. The problem of finding the least expected travel times for destination s is equivalent to that of finding shortest F-paths terminating at s in F-graph H. In section 5 we mentioned that F-visits are easy when they are organized from the destination node towards origin nodes; this is also true for shortest F-paths. For the above transportation problem, the following generalized Bellman's equations can be written, in which the weighted average distances are defined separately for stops and other nodes. Let VS be the set of stops, then :

$$d_s(s) = 0;$$

$$d_s(x) = \min\{t_{xy} + d_s(y) : (x, y) \in FS(x)\} \qquad x \in V \setminus V_S;$$

$$d_s(x) = \min\{w(L'_x) + \textstyle\sum_{y_j \in H(E(L'_x))} d_s(y_j)\pi_j(L'_x) : E(L'_x) \in FS(x)\}$$

$$= \min\left[\frac{\frac{1}{2} + \sum_{y_l \in H(E(L'_x))} d_s(y_j)\phi_j}{\Phi(L'_x)} : E(L'_x) \in FS(x)\right] \qquad x \in V_s.$$

Similar to procedures SBT, Shortest F-Tree procedures (SFT) have been developed to solve the above equations. Both type of SFT-queue and SFT- Dijkstra 8.

Conclusion

Hence we conclude that, let us suppose that engineers apply for positions with the lists of proficiency they may have, the factory management then tries to hire the least possible number of engineers, so that each proficiency that the factory needs is covered by a least one engineer then we construct a hyper graph with a verteo of each engineer and an hyper edge for each proficiency, then a minimum transversal set represents the minimum group of engineers that need to be hired to cover all proficiencies at this factory.

Reference

1. Acharya, B.D., Rao, S.B. and Arumugam, S., Embeddings and NP-Complete problems for graceful graphs, *Labelings of Discrete Structures and Applications*, Narosa Publishing House, New Delhi, India 57-62 (2007).
2. Addario-Berry, L., Aldred, R.E.L., Dalal, K. and Reed, B.A., Vertex colouring edge partitions, *J. Combin. Theory*, Ser. **B94**, 237-244 (2005).
3. Chartrand, G., Nebesky, L. and Zhang, P., Hamiltonian colourings of graph, *Discrete Appl. Math*, **146,** 257-272 (2005).
4. Chudnovsky, M., Robertson, N., Seymour, P. and Thomas, R., The strong perfect graph theorem, *Ann. Math*, **164,** 51-229 (2006).
5. Delavina, E., Pepper, R. and Wolter, B., Independence, Radius and Hamiltonian paths, *MATCH commun. Math. Comput. Chem.*. **58,** 481-510 (2007).
6. Escuadro, H., Okamoto, F. and Zhang, P., A three-color problem in graph theory, *Bull. Inst. Combin. Appl.*, **52**, 65-82 (2008).
7. Khennoufa, R. and Togni, O., A note on radio autipodal colourings of paths math, *Bohem*, **130**, 277-282 (2005).
8. Karonski, M., Luczak, T. and Thomson, A., Edge weights and verto colours, *J. Combin. Theory*, Ser. **B91**, 151-157 (2004).
9. Robertson, N. and Seymour, P.D., Graph minors, XX Wagner's conjecture, *J. Combin. Theory*, Ser. **B92**, 325-357 (2004).
10. Thomassen, C., Some remarks on Hajos conjecture, *J. Combinn. Theory*, Ser. **B93**, 95-105 (2005).

❑