Alcom-FT Technical Report Series
ALCOMFT-TR-02-5

Directed Hypergraphs: Problems, Algorithmic Results, and a Novel Decremental Approach*

Giorgio Ausiello[1], Paolo G. Franciosa[2], and Daniele Frigioni[1]

[1] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza", Italy
{ausiello,frigioni}@dis.uniroma1.it
[2] Dipartimento di Statistica, Probabilità e Statistiche Applicate
Università di Roma "La Sapienza", Italy
paolo.franciosa@uniroma1.it

Abstract. The purpose of this paper is twofold. First, we review several basic combinatorial problems that have been stated in terms of directed hypergraphs and have been studied in the literature in the framework of different application domains. Among them, transitive closure, transitive reduction, flow and cut problems, and minimum weight traversal problems. For such problems we illustrate some of the most important algorithmic results in the context of both static and dynamic applications. Second, we address a specific dynamic problem which finds several interesting applications, especially in the framework of knowledge representation: the maintenance of minimum weight hyperpaths under hyperarc weight increases and hyperarc deletions. For such problem we provide a new efficient algorithm applicable for a wide class of hyperpath weight measures.

Keywords: Directed hypergraph, minimum weight hyperpath, AND/OR graph, dynamic algorithm.

1 Hypergraph structures in computer science

Hypergraph is a common name for various combinatorial structures that generalize graphs. Beside the most known undirected hypergraphs (or simply *hypergraphs* [13, 14]), a relevant role is played by *directed hypergraphs*, a generalization of directed graphs, which find applications in several areas of computer science and mathematics for representing implicative structures. In a directed hypergraph we are given a set of nodes N and a set of pairs $\langle T, h\rangle$ (*hyperarcs*) where T is a subset of N and h is a single node in N. The most obvious interpretation of a hyperarc $\langle T, h\rangle$ is that the information associated to h functionally depends on the information associated to nodes in T.

* Work partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

Directed hypergraphs and other strictly related combinatorial structures are widely used in computer science. Notably, they are used in artificial intelligence for representing problem solving relationships (And-Or graphs [32] and recursive label node hypergraphs [15]), in database theory for representing functional dependencies among attributes (FD-graphs [5] and connections in acyclic hypergraphs [29, 36]), in deductive databases [23], in fuzzy logic, for determining the reliability of facts [8], in propositional logic, for satisfiability check (namely in the case of Horn formulæ [9, 21, 33]), in formal languages (weighted context free grammars [26]), in the theory of concurrency (Petri net paths [37]), in model checking (dependency graphs [28] and boolean graphs [3]), in diagnostics [2]. More applications can be found in [1, 4, 34].

In all these applications we need to provide a formal representation for a set of many-to-one implications and to study properties of the resulting structures such as assessing reachability, evaluating flows and cuts, determining less expensive implication chains under various cost criteria ('shortest' hyperpaths). In most cases determining hypergraph properties that consist in generalizations of corresponding properties on ordinary graphs is a much more complex (usually NP-hard) task. Therefore, the study of particular polynomial restrictions and of polynomial time approximation algorithms is quite important and constitutes a largely still unexplored field.

The purpose of this paper is twofold. First we want to review several basic combinatorial problems that have been stated in terms of directed hypergraphs and have been studied in the literature in the framework of different application domains. Among them, transitive closure and transitive reduction [6, 10], flow and cut problems [16, 21], minimum weight traversal problems [12, 21, 35]. For such problems we illustrate some of the most important algorithmic results in the context of both static and dynamic (mostly incremental) applications.

In second place, we want to address a specific dynamic problem which finds several interesting applications especially in the framework of knowledge representation: the maintenance of minimum weight hyperpaths under hyperarc deletions and weight increases. For such problem we provide a new efficient algorithm applicable for a large variety of hyperpath weight measures.

The paper is organized as follows. In section 2 the basic definitions concerning hypergraphs and hyperpaths are given and various hyperpath weight measures are presented. In Section 3 the problems of transitive closure and transitive reduction are discussed and the main algorithmic results are reviewed. Section 4 is devoted to flows and cuts in hypergraphs. Section 5 is devoted to hypergraph traversal problems with respect to a variety of hyperpath weight measures. Finally, in Section 6 we give a summary of algorithmic results in the dynamic framework and present a new efficient algorithm for maintaining minimum weight hyperpaths under hyperarc eliminations and weight increase operations. In the last section we draw some conclusions and suggest future research directions.

2 Hypergraphs and hyperpaths

The following definitions concerning directed hypergraphs are from [6] and [12], and are consistent with the more general definitions given in [22].

A *directed hypergraph* $\mathcal{H}$ (see Fig. 1 for an example) is a pair $\langle N, A\rangle$, where N is a non empty set of *nodes* and A is a set of hyperarcs; a *hyperarc* e is an ordered pair $\langle T, h\rangle$, with $T \subseteq N$, $T \neq \emptyset$, and $h \in N \setminus T$; T and h are called the *tail* and the *head* of e, and are denoted by $\mathrm{tail}(e)$ and $\mathrm{head}(e)$, respectively. A set of nodes is called a *source set* if it is the tail of some hyperarc; the *source area* of $\mathcal{H}$ is the sum of the cardinalities of its source sets. The *forward star* of $v \in N$ is the set $\mathrm{fstar}(v) = \{e \in A : v \in \mathrm{tail}(e)\}$, while the *backward star* of v is the set $\mathrm{bstar}(v) = \{e \in A : v = \mathrm{head}(e)\}$.

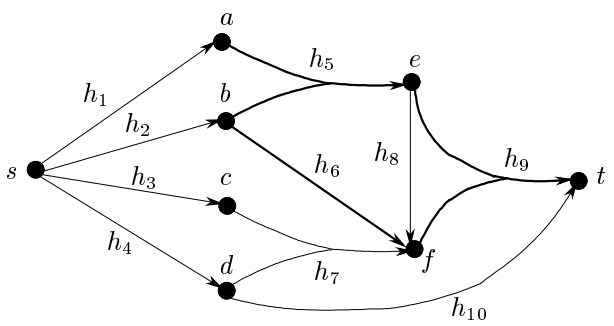

Fig. 1. A directed hypergraph

The *size* of a hypergraph $\mathcal{H}$ is defined as $\mathrm{size}(\mathcal{H}) = \sum_{e\in A} |\mathrm{tail}(e)|$. The following different notion of size has been also used in the literature: a "compact" hypergraph $\mathcal{H}^c$ is derived from $\mathcal{H}$ where a set of p hyperarcs having the same tail T is represented by a single hyperarc from T to a dummy node c, plus p hyperarcs from c to the original heads; the size of this hypergraph is denoted by $\mathrm{minsize}(\mathcal{H}) = \mathrm{size}(\mathcal{H}^c)$. This parameter is called *size* of a hypergraph $\mathcal{H}$ in [12], and is denoted by $|\mathcal{H}|$.

Given a hypergraph $\mathcal{H}$, a *subhypergraph* of $\mathcal{H}$ is a hypergraph $\mathcal{H}' = \langle N', A'\rangle$ with $N' \subseteq N$ and $A' \subseteq A$. A subhypergraph is *proper* if at least one of the inclusions is strict. A *hyperpath* in $\mathcal{H}$ from a set of nodes $S \subset N$, with $S \neq \emptyset$, called *source*, to a target node $t \in N$ is a subhypergraph $\Pi_{S,t} = \left\langle N_{\Pi_{S,t}}, A_{\Pi_{S,t}}\right\rangle$ of $\mathcal{H}$ having the following property: if $t \in S$, then $A_{\Pi_{S,t}} = \emptyset$, otherwise its $k \geq 1$ hyperarcs can be ordered in a sequence $\langle e_1, \ldots, e_k\rangle$ such that:

1. $\forall e_i \in A_{\Pi_{S,t}}, \mathrm{tail}(e_i) \subseteq S \cup \{\mathrm{head}(e_1), \ldots, \mathrm{head}(e_{i-1})\}$
2. $t = \mathrm{head}(e_k)$
3. No proper subhypergraph of $\Pi_{S,t}$ is a hyperpath from S to t in $\mathcal{H}$.

The above definition of hyperpath is consistent with the notion of *folded hyperpath* given in [12], and generalizes the notion of simple path in a directed graph. A node t is said to be *reachable* in $\mathcal{H}$ from source S if there exists a hyperpath $\Pi_{S,t}$ in $\mathcal{H}$. The *port* of a node v on $\Pi_{S,t}$, denoted by port(v), is the hyperarc in $A_{\Pi_{S,t}}$ having v as head; it is unique by condition 3 above. In Fig. 1, thick hyperarcs represent the hyperpath from source $S = \{a, b\}$ to node t consisting of hyperarcs $\langle h_5, h_6, h_9 \rangle$.

The most intuitive and classical measure of the weight of a hyperpath is its *cost*, defined as the sum of the weights of its hyperarcs (see, for example, [11]). A different approach leads to defining the weight of a hyperpath in an inductive manner. First attempts in this direction can be found in [31] and in [11]. This approach has been formalized in [12] in the following terms.

Definition 1. *A* functional hypergraph $\mathcal{H}_F = \langle N, A; F \rangle$ *is a directed hypergraph* $\mathcal{H} = \langle N, A \rangle$ *in which each hyperarc* $e = \langle X, y \rangle \in A$ *is associated to a triple* $F_e = (w_e, \psi_e, f_e)$*, where:*

- w_e *is a real value;*
- ψ_e *is a function from* $|X|$*-tuples of reals to reals;*
- f_e *is a function from pairs of reals to reals.*

Let $\Pi_{S,t}$ be a hyperpath from S to t, and let $\langle Z, t \rangle$ be the last hyperarc in $\Pi_{S,t}$ (i.e., the *port* of t), where $Z = \{z_1, z_2, \ldots, z_k\}$: then[1] $\Pi_{S,t} = \Pi_{S,z_1} \cup \Pi_{S,z_2} \cup \ldots \cup \Pi_{S,z_k} \cup \{\langle Z, t \rangle\}$, where Π_{S,z_i} is the subhyperpath of $\Pi_{S,t}$ going from S to z_i, $1 \leq i \leq k$. The weight of $\Pi_{S,t}$ depends on $w_{\langle Z,t \rangle}$, that gives the *weight* of hyperarc $\langle Z, t \rangle$, and on $\psi_{\langle Z,t \rangle}$, that takes into account the weights of all hyperpaths Π_{S,z_i}. Function $f_{\langle Z,t \rangle}$ combines these two weights. This is formalized in the following definition.

Definition 2. *Given a functional directed hypergraph* $\mathcal{H}_F = \langle N, A; F \rangle$*, a* weight measure μ *associates a real weight to a hyperpath* $\Pi_{S,t}$ *as follows:*

- *if* $\Pi_{S,t}$ *has no hyperarcs (i.e.,* $t \in S$*), then* $\mu(\Pi_{S,t}) = \mu_0$*, where* μ_0 *is a proper constant;*
- *if* $\Pi_{S,t} = \Pi_{S,z_1} \cup \Pi_{S,z_2} \cup \ldots \cup \Pi_{S,z_k} \cup \{\langle Z, t \rangle\}$*, then* $\mu(\Pi_{S,t}) = f_{\langle Z,t \rangle}(w_{\langle Z,t \rangle}, \psi_{\langle Z,t \rangle}(\mu(\Pi_{S,z_1}), \mu(\Pi_{S,z_2}), \ldots, \mu(\Pi_{S,z_k})))$.

Several weight measures have been introduced in the literature to define the weight of a hyperpath in a functional directed hypergraph, by considering, given a hyperarc e such that tail$(e) = \{x_1, x_2, \ldots, x_k\}$, different choices for functions ψ_e and f_e:

rank: $f_e(x, y) = x + y$, $\psi_e(x_1, x_2, \ldots, x_k) = \max_{1 \leq i \leq k}\{x_i\}$, and $\mu_0 = 0$;
gap: $f_e(x, y) = x + y$, $\psi_e(x_1, x_2, \ldots, x_k) = \min_{1 \leq i \leq k}\{x_i\}$, and $\mu_0 = 0$;
threshold: $f_e(x, y) = \max\{x, y\}$, $\psi_e(x_1, x_2, \ldots, x_k) = \max_{1 \leq i \leq k}\{x_i\}$, and $\mu_0 = 0$;

[1] By a little abuse of notation, given two hypergraphs $\mathcal{H}_1 = \langle N_1, A_1 \rangle$ and $\mathcal{H}_2 = \langle N_2, A_2 \rangle$ we denote the hypergraph $\langle N_1 \cup N_2, A_1 \cup A_2 \rangle$ by $\mathcal{H}_1 \cup \mathcal{H}_2$.

bottleneck: $f_e(x,y) = \min\{x,y\}$, $\psi_e(x_1,x_2,\ldots,x_k) = \min_{1\le i\le k}\{x_i\}$, and $\mu_0 = +\infty$;
traversal cost: $f_e(x,y) = x+y$, $\psi_e(x_1,x_2,\ldots,x_k) = \sum_{1\le i\le k}\{x_i\}$, when $w_e > 0$ for each $e \in A$, and $\mu_0 = 0$;
closure: $f_e(x,y) = \min\{x,y\}$, $\psi_e(x_1,x_2,\ldots,x_k) = \min_{1\le i\le k}\{x_i\}$, when $w_e = 1$ for each $e \in A$, and $\mu_0 = 0$;

For example, if we assume that all hyperarcs of the hypergraph in Fig. 1 but h_5 have unit weight, while hyperarc h_5 has weight 2, then the thick hyperpath has rank 3, gap 2 and traversal cost 4.

Some of the weight measures listed above have been used by other authors with different names. For example, in [21, 34], the *rank* is called *distance*, and the *traversal cost* is called *value*.

3 Transitive closure and transitive reduction

In this section, we summarize the main results achieved in the literature concerning the notions of *transitive closure* and *transitive reduction* in directed hypergraphs, that extend the analogous notions in the case of directed graphs. Both problems find application in the field of databases, where on one side we want to derive transitive functional dependencies, and on the other side we want to find minimal covers of sets of functional dependencies, that is a minimal set of functional dependencies equivalent to the given ones.

Transitive closure.
In [6] the authors introduce the notion of *transitive closure* of a directed hypergraph $\mathcal{H} = \langle N, A\rangle$ as the hypergraph $\mathcal{H}^+ = \langle N, A^+\rangle$ such that $\langle X, i\rangle$ is a hyperarc in A^+ if and only if one of the following conditions hold:

- $\langle X, i\rangle \in A$
- $i \in X$ *Extended reflexivity*
- there exists a set of nodes $Y = \{n_1, n_2, \ldots, n_q\}$ such that for each $1 \le j \le q$, $\langle X, n_j\rangle \in A^+$ and $\langle Y, i\rangle \in A$ *Extended transitivity*

Note that, in our framework such definition is equivalent to say that a hyperarc $\langle X, i\rangle$ is in A^+ if and only if there is a hyperpath in $\mathcal{H}$ from X to i. If X and Y are singletons, then the extended reflexivity and transitivity coincide with the usual notions of reflexivity and transitivity of graphs. Furthermore, note that the size of the closure of a directed hypergraph can be exponential in the number of nodes of the hypergraph, in fact, set X can be any element of 2^N (the power set of N).

In the same paper it is shown that using a suitable representation of $\mathcal{H}$, i.e., its FD-graph and the closure of the FD-graph (see [6] for the definitions), it is possible to compute the transitive closure of a directed hypergraph without falling into the exponential explosion of the hypergraph closure. In fact, the FD-graph closure grows at most quadratically. Using this approach the transitive closure of $\mathcal{H}$ is computed in $O(\text{size}(\mathcal{H}) \cdot n_s)$ worst case time, where n_s is the number of source sets of $\mathcal{H}$.

Transitive reduction.
The problem of finding the *transitive reduction* of a directed hypergraphs, that is finding a minimal hypergraph that has the same closure of a given one, is much more complex than in the case of graphs for two main reasons: *i*) the closure of a hypergraph has an exponential number of hyperarcs; *ii*) in the case of hypergraphs it is possible to define minimality with respect to a variety of parameters, while in the case of graph the only notion of minimality is with respect to the number of arcs. In [6], minimality in hypergraphs is defined with respect to the following criteria:

1. minimum size;
2. minimum number of hyperarcs;
3. minimum number of source sets;
4. minimum source area.

In the same paper, it has been shown that, given a directed hypergraph $\mathcal{H}$, the problems of deciding whether there exists a hypergraph $\mathcal{H}'$ with the same closure of $\mathcal{H}$, such that $\mathcal{H}'$ is minimum with respect to criteria 1, 2 or 4 are NP-hard, while deciding whether there exists a hypergraph $\mathcal{H}'$ with the same closure of $\mathcal{H}$ having the minimum number of source sets is polynomial and can be done in $O(\text{size}(\mathcal{H}) \cdot n_s)$ worst case time. Notice that the transitive reduction of a directed graph, that corresponds to minimality with respect to number of hyperarcs in a directed hypergraph, can be computed in polynomial time.

4 Flows and cuts in hypergraphs

Other classical graph problems that have been extended to hypergraphs are the problems of determining flows and cuts. In this section we review the main results achieved in the literature concerning these problems.

Flows in hypergraphs.
Flows in hypergraphs, also known as *hyperflows*, have been recently introduced as generalizations of *flows* in graphs. In particular, in [16] the *minimum cost hyperflow* problem is considered and some analogies with the standard *minimum cost flow* problem in directed graphs are shown.

In what follows, we summarize the results given in [16]. Given a directed hypergraph $\mathcal{H} = \langle N, A \rangle$, an *upper capacity* $u(e)$ and a *cost* $c(e)$ are associated with each $e \in A$, and a positive real *multiplier* $m_v(e)$ is associated with each $v \in \text{tail}(e)$. Moreover, a real *demand* $b(v)$ is associated with each $v \in N$. A *flow* in $\mathcal{H}$ is a function $f : A \to \mathbb{R}$, that satisfies the following *conservation* constraint:

$$\sum_{\text{head}(e)=v} f(e) \quad - \sum_{v \in \text{tail}(e)} m_v(e) f(e) = b(v), \quad \forall v \in N$$

The flow is *feasible* if it satisfies the following *capacity* constraint:

$$0 \leq f(e) \leq u(e), \quad \forall e \in A$$

The *minimum cost hyperflow* problem consists of finding a feasible flow in $\mathcal{H}$ which minimizes the value $\sum_{e \in A} c(e) f(e)$.

An interesting issue is computing the maximum flow in a directed hypergraph. Let us assume that $\mathcal{H}$ has only one node s such that $\mathrm{bstar}(s) = \emptyset$ and only one node t such that $\mathrm{fstar}(t) = \emptyset$; they are called the *source* and the *terminal* of $\mathcal{H}$, respectively (if s and t do not exist, dummy nodes and hyperarcs can be added in a standard way). The *maximum hyperflow* problem consists of finding a feasible flow in $\mathcal{H}$ that maximizes the amount of flow entering t, and satisfies the conservation constraint in the special case that $b(v) = 0$, for each $v \in N$.

In [16] the notion of *spanning hypertree* of a hypergraph is introduced as a generalization of the notion of spanning tree of a graph, and it is shown that there exists a correspondence between *basis matrices* and spanning hypertrees. Based on this correspondence, the authors propose a simplex like algorithm for minimum cost hyperflow computations, and show that most of the computations performed by this algorithm have a direct and elegant hypergraph interpretation.

Cuts in hypergraphs.

The notion of *cut* in a directed hypergraph has been defined in [21] as follows. A *cut* $\mathcal{C}_{st}$ between two nodes s and t of a directed hypergraph $\mathcal{H} = \langle N, A \rangle$ is a partition of N into two subsets N_s and N_t such that $s \in N_s$ and $t \in N_t$. The associated *cutset* is the set of all hyperarcs $e \in A$ such that $\mathrm{tail}(e) \subseteq N_s$ and $\mathrm{head}(e) \in N_t$. The *size* of a cut is the number of hyperarcs in its cutset.

It is well known that the unsatisfiability of a propositional Horn formula corresponds to the existence of a hyperpath in a hypergraph connecting two special nodes s and t, where s corresponds to True and t to False [9, 19]. As a consequence, it is clear that the Maximum Horn Sat problem can be reduced to the problem of finding a *minimum cardinality cut* in a directed hypergraph, i.e., a cut whose associated cutset has the minimum number of hyperarcs. In [21], the authors also show that the latter problem is equivalent to the problem of finding the minimum set of hyperarcs of $\mathcal{H}$ such that each hyperpath of $\mathcal{H}$ contains at least one of these hyperarcs. Therefore, the minimum cardinality cut problem can be formulated as the problem of assigning 0/1 weights to the hyperarcs of $\mathcal{H}$ in order to make the weight of each hyperpath larger than or equal to 1, and to minimize the sum of the assigned weights.

Based on this result, the authors propose three different IP formulations for the minimum cardinality cut problem, by using three different measures to assign hyperpath weights, that is: the *cost*, the *distance* (*rank*), and the *value* (*traversal cost*). They also show that these three formulations and the well known ILP formulation of the Max Horn SAT problem are all equivalent.

Finally, they investigate the properties of the relaxations of these IP formulations by showing that they define a hierarchy. The weakest relaxation in

the hierarchy, that is the relaxation of the formulation making use of the *value* (*traversal cost*) weight function, is shown to correspond to the *dual* of a hypergraph *max flow* problem with unit capacities [16]. This implies that the well-known max-flow-min-cut theorem for directed graphs (see, e.g., [17]) holds in the case of directed hypergraphs with unit upper capacities. For example, if we assume unit upper capacities for all the hyperarcs in the hypergraph of Fig. 1, and $m_v(e) = 1/|\text{tail}(e)|$ for each $v \in \text{tail}(e)$, then the minimum cut and the maximum flow are both equal to 2.

In the case of non unit upper capacities, the size of a cut is the sum of the upper capacities of the hyperarcs in the cutset. An interesting feature of hypergraphs is that, if hyperarcs have arbitrary upper capacities, then the max-flow-min-cut theorem does not hold any longer, even in the restrictive case in which we fix $m_v(e) = 1/|\text{tail}(e)|$ for each $v \in \text{tail}(e)$. In fact, if in the hypergraph shown in Fig. 1 we assume the following upper capacities: $u(h_1) = u(h_2) = u(h_3) = u(h_4) = 10$, $u(h_5) = 2$, $u(h_6) = 1$, $u(h_7) = 1$, $u(h_8) = 1$, $u(h_9) = 4$, $u(h_{10}) = 1$, then the minimum cut is 3 (partitioning nodes into the sets $\{s, a, b, c, d, f\}$ and $\{e, t\}$), while the maximum flow is 5.

5 Minimum weight traversal problems

One of the most important problems in hypergraphs is finding minimum weight hyperpaths. Such problem finds relevant applications in reachability in Petri nets [1], reliability in fuzzy systems [8], assumption-based truth maintenance systems [18, 27], computation of minimum models of Horn formulæ [9], search of optimal strategies in problem solving [30], analysis of dynamic programming [24].

Several results have been proposed in the literature concerning the problem of finding minimum (or maximum) weight hyperpaths in a directed hypergraph. As already mentioned, there are various ways in which a weight can be attached to a hyperpath. Depending on the weight measure used to assign a weight to a hyperpath the problem can be polynomially tractable or NP-hard. For example, in [11] it has been shown that the problem of finding minimum *cost* hyperpaths in a directed hypergraph is NP-hard, where the *cost* of a hyperpath is the sum of the weights of its hyperarcs. Conversely, when we take into consideration inductively defined measures, it is possible to characterize a number of cases in which the problem can be solved in polynomial time. Examples of inductively defined measures that can be computed in polynomial time are the *rank* and the *gap* [25]. Observe that cost, rank and gap are all generalizations of the standard notion of distance in graphs, which can be computed in polynomial time.

The more general approach for finding minimum weight hyperpaths in a functional directed hypergraph is finding a *fixed point* of the following set of equations, known as *generalized Bellman-Ford equations* (see, e.g., [21, 35]):

$$L(v) = \begin{cases} 0 & \text{if } v \in S \\ \min\limits_{e \in \text{bstar}(v)} \{f_e(w_e, \psi_e(x_1, x_2, \ldots, x_k))\} & \text{if } v \in N \setminus S \end{cases}$$

where $\text{tail}(e) = \{x_1, x_2, \ldots, x_k\}$, and f_e and ψ_e depend on the particular weight measure used. The complexity of this problem strongly depends on the characteristics of functions f_e and ψ_e, since, as we will see in the remainder of this section, in several cases the solution of the general problem can be achieved with a Dijkstra-like computation.

In his definition of grammar problems [26], Knuth introduces the concepts of *superior* and *inferior* function in the framework of context-free grammars, as follows.

Definition 3. *A function $g(x_1, \ldots, x_k)$ from $(\mathbb{R}^+)^k$ to $\mathbb{R}^+$ is:*

- superior *(SUP) if it is monotone nondecreasing in each variable and* $g(x_1, \ldots, x_k) \geq \max(x_1, \ldots, x_k)$
- inferior *(INF) if it is monotone nondecreasing in each variable and* $g(x_1, \ldots, x_k) \leq \min(x_1, \ldots, x_k)$

Examples of superior functions are $\max_{1 \leq i \leq k}\{x_i\}$, and $\Sigma_{i=1}^{k} x_i$. Examples of inferior functions are: $\min_{1 \leq i \leq k}\{x_i\}$, and $\Pi_{i=1}^{k}\{x_i\}$ when $0 \leq x_i \leq 1$, $i = 1, \ldots, k$. Ramalingam and Reps in [35] introduced the following generalizations of superior (inferior) functions.

Definition 4. *A function $g(x_1, \ldots, x_k)$ from $(\mathbb{R}^+)^k$ to $\mathbb{R}^+$ is:*

- weakly superior *(WSUP) if it is monotone nondecreasing in each variable and, for* $1 \leq i \leq k$, $g(x_1, \ldots, x_k) < x_i \Rightarrow g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$
- weakly inferior *(WINF) if it is monotone nondecreasing in each variable and, for* $1 \leq i \leq k$, $g(x_1, \ldots, x_k) > x_i \Rightarrow g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$

Examples of weakly superior functions that are not superior are $\min_{1 \leq i \leq k}\{x_i\}$, $\min_{1 \leq i \leq k}\{x_i\} * 2$, and any constant function. Examples of weakly inferior functions that are not inferior functions are: $\max_{1 \leq i \leq k}\{x_i\}$, $\max_{1 \leq i \leq k}\{x_i\}/2$, and any constant function.

Both Knuth [26] and Ramalingam and Reps [35] considered also the classes of *strict superior*, *strict inferior*, *strict weakly superior*, and *strict weakly inferior* functions, as defined below.

Definition 5. *A function $g(x_1, \ldots, x_k)$ from $(\mathbb{R}^+)^k$ to $\mathbb{R}^+$ is:*

- strict superior *(SSUP) if it is monotone nondecreasing in each variable and* $g(x_1, \ldots, x_k) > \max(x_1, \ldots, x_k)$
- strict inferior *(SINF) if it is monotone nondecreasing in each variable and* $g(x_1, \ldots, x_k) < \min(x_1, \ldots, x_k)$.

Definition 6. *A function $g(x_1, \ldots, x_k)$ from $(\mathbb{R}^+)^k$ to $\mathbb{R}^+$ is:*

- strict weakly superior *(SWSUP) if it is monotone nondecreasing in each variable and, for* $1 \leq i \leq k$, $g(x_1, \ldots, x_k) \leq x_i \Rightarrow g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$

– strict weakly inferior *(SWINF) if it is monotone nondecreasing in each variable and, for* $1 \leq i \leq k$, $g(x_1, \ldots, x_k) \geq x_i \;\Rightarrow\; g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$

We now relate the above defined classes of functions to hypergraphs. Remember that in a functional hypergraph, each hyperarc $e \in \mathcal{H}$ is associated to a triple (w_e, ψ_e, f_e). Given $e = \langle X, t \rangle$, with $X = \{x_1, x_2, \ldots, x_k\}$, the weight μ of any hyperpath $\Pi_{S,t}$ having e as the last hyperarc is given by $\mu(\Pi_{S,t}) = f_e(w_e, \psi_e(\mu(\Pi_{S,x_1}), \mu(\Pi_{S,x_2}), \ldots, \mu(\Pi_{S,x_k})))$. If functions f_e and ψ_e, for all $e \in \mathcal{H}$, are, say, superior functions, then the overall weight measure μ is superior as well (with respect to arguments w_e, for each hyperarc e in the considered hyperpath). The table in Fig. 2 summarizes the properties of the weight measures known in the literature with respect to the classes of function defined above. Concerning the relationship among the different classes of weight measures, as well as their compositions, we refer to [12].

weight measure μ	$f_e(w_e, \psi_e)$	$\psi_e(\mu_1, \ldots, \mu_k)$	resulting properties
rank $w_e > 0$	$+$ *SSUP*	max *SUP, WINF*	*SSUP*
gap $w_e > 0$	$+$ *SSUP*	min *WSUP, INF*	*SWSUP*
bottleneck $w_e > 0$	min *WSUP, INF*	min *WSUP, INF*	*WSUP, INF*
threshold $w_e > 0$	max *SUP, WINF*	max *SUP, WINF*	*SUP, WINF*
traversal cost $w_e > 0$	$+$ *SSUP*	$\sum$ *SSUP*	*SSUP*
closure $w_e = 1$	min *WSUP, INF*	min *WSUP, INF*	*WSUP, INF*

Fig. 2. Properties of hyperpath weight measures.

The generalization of Dijkstra algorithm to directed hypergraphs (or similar structures) was first tackled by Knuth in [26]. For weight measures based on superior functions, it is possible to determine minimum cost derivation trees in a weighted context free grammar (or, equivalently, a minimum weight hyperpath in a functional directed hypergraph) in $O(\text{size}(\mathcal{H}) + m \log n)$ worst case time (or $O(\text{size}(\mathcal{H}) + n \log n)$ by using Fibonacci heaps [20]). This result was subsequently extended in [12, 31, 35] to weakly superior (inferior) functions, thus including all weight measures in Fig. 2. In [12], it is also shown how the suitability of Dijkstra-based algorithms is related to the properties of weight measures and to the existence of particular types of cycles in hypergraphs.

6 Dynamic traversal algorithms

In this section we consider the problem of dynamically maintaining reachability and minimum weight hyperpaths in a directed hypergraph subject to modifications as hyperarc insertions and deletions, and hyperarc weight changes. These problems arise in several application domains as, for example, in the minimum model maintenance in Horn Formulæ [7–9, 35].

In this setting we do not want to recompute hyperpaths from scratch after each change, but we want to take advantage of the part of the previous solution that is still valid. A common classification of dynamic problems in hypergraphs is among fully-dynamic ones, where insertions and deletions of hyperarcs can be intermixed, and semi-dynamic ones (incremental or decremental), where only insertions or only deletions are allowed, respectively.

The first hypergraph problem stated in a dynamic framework has been the maintenance of the transitive closure of a directed hypergraph $\mathcal{H}$ under hyperarc insertions. In [10], it is shown that this problem can be solved in $O(\text{size}(\mathcal{H}) \cdot n_s)$ worst case time, where n_s is the number of source sets of $\mathcal{H}$, in such a way that checking the existence of a hyperpath takes $O(\log n_s)$ time.

Minimum weight traversal problems have been studied by various authors. For the incremental case, an algorithm is presented in [9], that maintains satisfiability and the minimum model of a Horn formula $\mathcal{F}$ in $O(n \cdot \text{Length}(\mathcal{F}))$ total time over a sequence of clause insertions, where $\text{Length}(\mathcal{F})$ is the sum of the number of literals in the clauses of $\mathcal{F}$. This incremental algorithm is extended in [8] to the case of Horn formulæ with uncertainty. Algorithms are given in [25] for the incremental maintenance of minimum rank and minimum gap hyperpaths under a sequence of hyperarc insertions in overall $O(n \cdot \text{size}(\mathcal{H}))$ time, under the assumption of unit hyperarc weights.

The fully dynamic version of the problem has been considered in [35]. The authors propose a Dijkstra-like procedure, applicable to strict weakly superior functions. The proposed algorithm takes $O(||\delta|| \cdot (\log ||\delta|| + M))$ worst case time per update operation, where each operation may consist of several insertions/deletions of hyperarcs and hyperarc weight changes. Parameter $||\delta||$ represents the number of nodes interested by the operation plus the total number of hyperarcs incident to these nodes, while M is the time needed to compute the weight function. Note that in the worst case $||\delta|| = m$.

In this section, we are interested in the decremental case for two reasons: first, it finds applications in the context of assumption-based truth maintenance systems [18, 27], where maintaining minimum hyperpaths under deletions of hyperarcs corresponds to maintaining a set of "small" explanations of all observations under the elimination of either an hypothesis or a clause in the background theory. Second, better complexity bounds can be obtained with respect to the fully dynamic case. For example, in [7] a decremental algorithm is proposed for maintaining the minimum rank hyperpaths under a sequence of hyperarc deletions in overall $O(n \cdot \text{size}(\mathcal{H}))$ time and $O(\text{size}(\mathcal{H}))$ space, under the assumption of unit hyperarc weights. In the case of integer hyperarc weights in $[1, C]$ the algorithm requires $O(C \cdot n \cdot \text{size}(\mathcal{H}))$ total time and $O(C + \text{size}(\mathcal{H}))$ space to

handle any sequence of hyperarc deletions and weight increase operations. This improves over the solution in [35] when applied to the decremental maintenance of minimum rank hyperpaths in the case of integer hyperarc weights in $[1, C]$.

In the next subsection, we propose a new decremental algorithm, with respect to those given in [7, 35], that applies to all weight function in *SWSUP*. This algorithm is based on a novel approach, since it essentially performs a bfs-like visit, rather than a Dijkstra-like visit.

6.1 A new decremental algorithm

Let $\mathcal{H} = \langle N, A; F \rangle$ be a functional directed hypergraph with n nodes and m hyperarcs. Procedure `Weight_Increase` (see Fig. 3), maintains the minimum weight hyperpaths from a fixed source $S \subset N$ to all other nodes of $\mathcal{H}$ under hyperarc weight increases and hyperarc deletions.

In a minimization (maximization) problem, as a consequence of a hyperarc weight increase or hyperarc deletion, the weight of some hyperpath may increase or some hyperpaths may disappear, and the weight of minimum weight hyperpaths can only increase (decrease).

In what follows, we consider the minimization problem in the case that the weight of a hyperarc e is increased by a positive quantity δ, and then discuss how to modify Procedure `Weight_Increase` in order to handle hyperarc deletions. Each f_e is supposed to assume values in an interval of integers $[1, C]$.

We denote by $\mathcal{H}'$ the hypergraph obtained from $\mathcal{H}$ after updating hyperarc $e = \langle X, y \rangle$, and by $\mathrm{w}(v)$ ($\mathrm{w}'(v)$) the weight of the minimum weight hyperpath from S to v in $\mathcal{H}$ ($\mathcal{H}'$). Analogously, $\mathrm{p}(v)$ ($\mathrm{p}'(v)$) denotes the port of v in $\mathcal{H}$ ($\mathcal{H}'$).

First of all, we observe that after updating w_e in $\mathcal{H}$, each hyperpath in $\mathcal{H}$ that does not contain e preserves its weight. Hence, $\mathrm{w}'(v) = \mathrm{w}(v)$ for each node v whose minimum weight hyperpath in $\mathcal{H}$ does not contain e. We thus concentrate on the set of nodes in $\mathcal{H}$ whose current minimum weight hyperpath from S contains e. Given one of these nodes v, either $\mathrm{w}'(v) = \mathrm{w}(v)$, or $\mathrm{w}'(v) > \mathrm{w}(v)$.

Hypergraph $\mathcal{H}$ is represented by associating to each node v two simple lists containing all hyperarcs in $\mathrm{bstar}(v)$ and $\mathrm{fstar}(v)$. Minimum weight hyperpaths are represented as follows. For each node v, we store:

- $\mathrm{weight}(v)$, that coincides with $\mathrm{w}(v)$ ($\mathrm{w}'(v)$) before (after) the update;
- $\mathrm{port}(v)$, that coincides with $\mathrm{p}(v)$ ($\mathrm{p}'(v)$) before (after) the update.

For each hyperarc e, we explicitly store the value $F(e) = f_e(w_e, \psi_e(\mathrm{weight}(x_1), \mathrm{weight}(x_2), \ldots, \mathrm{weight}(x_k)))$, where $\{x_1, x_2, \ldots, x_k\} = \mathrm{tail}(e)$. In order to keep the space occupancy within $O(\mathrm{size}(\mathcal{H}))$, we explicitly represent each hyperarc in $\mathcal{H}$ only once; all the occurrences of hyperarcs in the above data structures are implemented as references.

Procedure `Weight_Increase` explores the set of nodes whose weight and/or port changes under the update of hyperarc $e = \langle X, y \rangle$, and builds the new minimum weight hyperpaths as follows. The hypergraph is examined starting from node y by a bfs-like visit (that is, by increasing minimum weight), that

```
Procedure Weight_Increase(e = ⟨X, y⟩; δ)
begin
1.    w_e ← w_e + δ
2.    Update(e,v)
3.    if port(y) ≠ e or F(e) has not changed then EXIT
4.    NewWeightSet(weight(y)) ← {y}
5.    for i ← weight(y) to C do
6.        foreach v ∈ NewWeightSet(i) do
7.            delete v from NewWeightSet(i)
8.            search the first h in bstar(v) such that F(h) = i
9.            if h exists then
10.               port(v) ← h
11.           else
12.               port(v) ← NIL
13.               weight(v) ← min {F(h) | h ∈ bstar(v)}
14.               insert v in NewWeightSet(weight(v))
15.               foreach e' ∈ fstar(v) do
16.                   Update(e',v)
17.                   if F(e') has changed then
18.                       let z be head(e')
19.                       if e' = port(z) and z ∉ NewWeightSet(weight(z)) then
20.                           insert z into NewWeightSet(weight(z))
21.               endfor
22.       endfor
23.   endfor
end
```

Fig. 3. Procedure `Weight_Increase`.

is pruned any time a hyperarc is found that does not belong to any minimum weight hyperpath and/or whose weight does not change.

The i-th iteration of the for loop at Line 5 identifies the set of nodes having weight i (after the update), whose weight and/or port changes due to the hyperarc update, by selecting all nodes v such that $\mathrm{w}(v) = i$, $\mathrm{w}'(v) = i$ and $\mathrm{p}'(v) \neq \mathrm{p}(v)$, and all nodes v such that $\mathrm{w}(v) < i$ and $\mathrm{w}'(v) = i$.

Inspected nodes are temporarily stored in an array of sets of nodes, named NewWeightSet. A node v is put in NewWeightSet(i) if and only if $\mathrm{w}'(v)$ is known to be at least i. Nodes are extracted from set NewWeightSet(i) by increasing i, and for each node v we check whether there is a hyperarc h such that $f_h(w_h, \psi_h(\mathrm{tail}(h))) = i$. If this is the case, the weight of node v is set to i, otherwise v is inserted into some set NewWeightSet(j), where $j > i$, for future inspection.

Procedure `Update`(e,v), called at Lines 2 and 16 of Procedure `Weight_Increase`, updates $F(e)$, by re-evaluating ψ_e and f_e under the increase of weight(v), $v \in \mathrm{tail}(e)$, or w_e.

In order to manage hyperarc deletions, we add a new item NewWeightSet$(C+1)$, containing all nodes that are found to be unreachable as a consequence of a deletion. The algorithm detects unreachability from the source, by checking whether a node gets a weight greater than C or its backward star is empty. In these cases, unreachable nodes are put in the set NewWeightSet$(C+1)$ and considered at the end of the algorithm, by setting the port to nil and the weight to $+\infty$.

Provided that the weight functions associated to hyperarcs fulfill the following conditions:

- the composition of functions f_e and ψ_e is in *SWSUP* (*SWINF*) for each hyperarc e. For example, it is sufficient that f_e and ψ_e are both monotone non decreasing (non increasing), and either f_e or ψ_e are in *SWSUP* (*SWINF*);
- $f_e(w_e, \psi_e(x_1, x_2, \ldots, x_k))$ can be maintained under increments (decrements) of x_i within $O(k)$ space and $O(1)$ amortized time;
- $f_e(w_e, \psi_e(x_1, x_2, \ldots, x_k))$ can be maintained under increments (decrements) of w_e within $O(k)$ space and $O(k)$ worst case time.

we can show that Procedure `Weight_Increase` manages a sequence of hyperarc weight increases and deletions in $O((C+n) \cdot \text{size}(\mathcal{H}))$ overall worst case time.

Note that the algorithm in [7] is a special case of the algorithm in this paper, in fact it maintains minimum rank hyperpath of a directed hypergraph within the same time bounds of Procedure `Weight_Increase` and the rank is a special case of *SWSUP* weight functions (see table of Fig. 2). It is also clear that our algorithm improves that of [35] with respect to the amortized time per operation, if only hyperarc deletions and weight increases are allowed.

7 Concluding remarks

There is a large body of results on implicative structures, that is structures in which we need to provide a formal representation for a set of many-to-one implications and to study the resulting properties, such as determining less expensive implication chains under various cost criteria. Our aim has been to illustrate the most important problems and the main algorithmic results that are available in the literature. Besides we have discussed in particular the problem of maintaining chains of implications under implication elimination. This problem consists of maintaining minimum hyperpaths (under some minimality criteria) under hyperarc deletions, and can be efficiently tackled for several natural hyperpath weight measures.

It is worth noting that the higher complexity of the problems we have discussed in this paper with respect to the complexity of the corresponding problems in graphs, arises from the peculiar combinatorial features of implicative structures (see, e.g., [12, 31] for the notions of path, distance, cycle, etc). The comprehension of such aspects can be greatly enhanced by making use of a uniform representation of implicative structures in terms of directed hypergraphs.

Future research directions might address various goals. On one side, it would be desirable to determine hypergraph properties under which hypergraph problems become tractable; on the other side, approximability properties of hypergraph problems should be better explored; finally, since the connection between flows and cuts in hypergraphs is significantly different with respect to graphs, the complexity of the various formulations of flow problems in hypergraphs deserves further investigation.

References

1. P. Alimonti, E. Feuerstein, and U. Nanni. Linear time algorithms for liveness and boundedness in conflict–free petri nets. In *Proceedings of Latin American symposium on Theoretical INformatics (LATIN'92)*, volume 583 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1992.
2. C. Alonso, B. Pulido, and G. G. Acosta. On line industrial diagnosis : an attempt to apply artificial intelligence techniques to process control. In *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-98-AIE)*, volume 1415 of *Lecture Notes in Artificial Intelligence*, pages 804–813. Springer-Verlag, 1998.
3. H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, 1994.
4. J. Aráoz. Forward chaining is simple(x). *Operations Research Letters*, 26:23–26, 2000.
5. G. Ausiello, A. D'Atri, and D. Saccà. Graph algorithms for functional dependency manipulation. *Journal of the ACM*, 30(4):752–766, 1983.
6. G. Ausiello, A. D'Atri, and D. Saccà. Minimal representation of directed hypergraphs. *SIAM Journal on Computing*, 15(2):418–431, 1986.
7. G. Ausiello, P. G. Franciosa, D. Frigioni, and R. Giaccio. Decremental maintenance of minimum rank hyperpaths and minimum models of Horn formulæ. Technical Report ALCOMFT-TR-01-19, IST Programme of the EU IST-1999-14186 (ALCOM-FT), 2001. http://www.brics.dk/cgi-alcomft/db?state=reports.
8. G. Ausiello and R. Giaccio. On-line algorithms for satisfiability problems with uncertainty. *Theoretical Computer Science*, 171(1–2):3–24, 1997.
9. G. Ausiello and G. F. Italiano. On-line algorithms for polynomially solvable satisfiability problems. *Journal of Logic Programming*, 10(1/2/3–4):69–90, 1991.
10. G. Ausiello, G. F. Italiano, and U. Nanni. Dynamic maintenance of directed hypergraphs. *Theoretical Computer Science*, 72(2-3):97–117, 1990.
11. G. Ausiello, G. F. Italiano, and U. Nanni. Optimal traversal of directed hypergraphs. Technical Report TR-92-073, International Computer Science Institute, Berkeley, CA, September 1992.
12. G. Ausiello, G. F. Italiano, and U. Nanni. Hypergraph traversal revisited: Cost measures and dynamic algorithms. In *Symposium on Mathematical Foundations of Computer Science (MFCS'98)*, volume 1450 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1998.
13. C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, The Netherlands, 1973.
14. C. Berge. *Hypergraphs: combinatorics of finite sets*. North-Holland, Amsterdam, The Netherlands, 1989.

15. H. Boley. Directed recursive labelnode hypergraphs: A new representation language. *Artificial Intelligence*, 9(1):49–85, 1977.
16. R. Cambini, G. Gallo, and M. G. Scutellà. Flows on hypergraphs. *Mathematical Programming*, 78:195–217, 1997.
17. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 1992.
18. J. de Kleer. An assumption based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.
19. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulæ. *Journal of Logic Programming*, 1(3):267–284, 1984.
20. M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.
21. G. Gallo, C. Gentile, D. Pretolani, and G. Rago. Max horn SAT and the minimum cut problem in directed hypergraphs. *Mathematical Programming*, 80:213–237, 1998.
22. G. Gallo, G. Longo, S. Nguyen, and S. Pallottino. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201, 1993.
23. G. Gallo and G. Rago. A hypergraph approach to logical inference for datalog formulæ. Technical Report TR-28/90, Dipartimento di Informatica, Università di Pisa, Italy, 1990.
24. S. Gnesi, U. Montanari, and A. Martelli. Dynamic programming as graph searching: An algebraic approach. *Journal of the ACM*, 28(4):737–751, 1981.
25. G. F. Italiano and U. Nanni. On-line maintenance of minimal directed hypergraphs. In *Italian Conference on Theoretical Computer Science*, pages 335–349, 1989.
26. D. E. Knuth. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, 1977.
27. K. Konolige. Abductive theories in artificial intelligence. In *Principles of Knowledge Representation*, pages 129–152. CSLI Publications, 1996.
28. X. Liu and S. A. Smolka. Simple linear-time algorithms for minimal fixed points. In *International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 53–65. Springer-Verlag, 1998.
29. D. Maier and J. D. Ullman. Connections in acyclic hypergraphs. In *Symposium on Principles of Database Systems (PODS'82)*, pages 34–39. ACM Press, 1982.
30. A. Martelli and U. Montanari. Additive and/or graphs. In *3rd International Joint Conference on Artificial Intelligence (IJCAI'73)*, pages 1–11, 1973.
31. S. Nguyen and S. Pallottino. Hypergraphs and shortest hyperpaths. In *Combinatorial Optimization*, volume 1403 of *Lecture Notes in Mathematics*, pages 258–271. Springer-Verlag, 1986.
32. N. J. Nilsson. *Problem solving methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.
33. D. Pretolani. Satisfiability and hypergraphs. Technical Report TD-12/93, Dipartimento di Informatica, Università di Pisa, Italy, 1993.
34. D. Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123:315–324, 2000.
35. G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2):267–305, 1996.
36. J. D. Ullman. *Principles of Database Systems*. Computer Science Press, 1982.
37. H. C. Yen. A unified approach for deciding the existence of certain Petri net paths. *Information and Computation*, 96(1):119–137, 1992.