DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

# Classification and Traversal Algorithmic Techniques for Optimization Problems on Directed Hyperpaths

Giorgio Ausiello
Giuseppe F. Italiano
Luigi Laura
Umberto Nanni
Fabiano Sarracco

Technical Report n. 18, 2010

# Classification and Traversal Algorithmic Techniques for Optimization Problems on Directed Hyperpaths

Giorgio Ausiello[*] Giuseppe F. Italiano[†] Luigi Laura[*] Umberto Nanni[*‡]
Fabiano Sarracco[*]

## Abstract

Directed *hypergraphs* are used in several applications to model different combinatorial structures. A directed hypergraph is defined by a set of nodes and a set of *hyperarcs*, each connecting a set of source nodes to a single target node. A *hyperpath*, similarly to the notion of path in directed graphs, consists of a connection among nodes using hyperarcs. Unlike paths in graphs, however, hyperpaths are suitable of many different definitions of measure, which have been used in a wide set of applications. Not surprisingly, depending on the considered measure function the cost of finding optimal hyperpaths may range from NP-hard to linear time.

A first solution for finding optimal hyperpaths in case of a *superior functions (SUP)* can be found in a seminal work by Knuth [Knu77], which generalizes Dijkstra's Algorithm [Dij59] to deal with a grammar problem. This solution is further extended by Ramalingam and Reps [RR96] to deal with *weakly superior functions (WSUP)*. Dijkstra's priority queue can find optimal paths or hyperpaths if the measure function complies two hypotheses: it is monotone with respect to all its arguments and (multidimensional) triangle inequality holds. We show that monotonicity - alone - is sufficient to guarantee interesting properties, and to make some optimization algorithms effective. Hence we introduce the *generalized superior function (GSUP)*, and consider the symmetrical classes of *inferior* functions, giving rise to a hierarchy of classes of optimization problems on directed hypergraphs.

After showing that some measure functions might induce cycles in optimal hyperpaths, we come up to another taxonomy of measure functions, based on the structure of the optimal hyperpaths they determine, and relate the two hierarchies.

Finally we introduce a general algorithmic pattern for the *single-source optimal hyperpath* problem encompassing existing and new algorithms, and compare their effectiveness in various cases, including the case of optimal cyclic hyperpaths.

**Keywords:** hyperpath algorithm, directed hypergraph, optimal hyperpath, cyclic hyperpath, measure function, algorithmic pattern

[*]Dipartimento di Informatica e Sistemistica “Antonio Ruberti”, Università di Roma “La Sapienza”, via Ariosto 25, 00185, Roma, Italy. {ausiello,laura,nanni,sarracco}@dis.uniroma1.it

[†]Dipartimento di Informatica, Sistemi e Produzione, Università di Roma “Tor Vergata”, via del Politecnico 1, 00133, Roma, Italy. italiano@disp.uniroma2.it

[‡]Centro di Ricerca per il Trasporto e la Logistica, Università di Roma “La Sapienza”, via Eudossiana 18, 00184, Roma, Italy.

## Contents

## 1 Introduction

Directed hypergraphs are a generalization of directed graphs. While directed graphs are normally used for representing *one-to-one* relations over finite sets, in several areas of computer science the need for more general relations arises; a directed hypergraph is useful exactly in these scenarios.

A *directed hypergraph* $\mathcal{H}$ is a pair $\langle N, H \rangle$ where $N$ is a set of nodes and $H$ is a set of *hyperarcs*. Based on the possible definitions of hyperarcs, different models have been considered in the literature with corresponding different properties, expressivity, and computational costs. Gallo et al. [GLNP92, GLNP93] provide an extensive coverage of alternatives.

In the definition that we consider throughout this paper, each hyperarc connects a set of *source* nodes to a single *target* node. This is the case of a *many-to-one* relation, including as a special case the concept of function.

In several applications where directed hypergraphs have been adopted, the notions of path and of traversal are required. A *hyperpath*, similarly to the notion of path in directed graphs, consists of a connection among nodes using hyperarcs. However, while paths and shortest paths in directed graphs are standard concepts, and efficient algorithms for their computations are well known, for hypergraphs the corresponding notions of hyperpaths and optimal hyperpaths are very subtle. In particular, hyperpaths are susceptible of different definitions of measure, each capturing concepts arising in various applications. Not surprisingly, some of these measures make the problem of finding optimal hyperpaths NP-hard.

Contesti in cui sono utili

Directed hypergraphs have been used in several contexts to model problems in an increasing number of areas, such as: functional dependencies in databases [ADS83], transportation networks [NP88, NPG98, MN98, Pre00], Horn clauses in propositional calculus [AI91], computational linguistics and automated speech recognition [KM04, Ned03], analysis of RDF (Resource Description Framework) documents [MV07, WLHW09, JHY10], Datalog formulae [GR90], machine learning [GMKT97], rule-based expert systems [RSC97], association rules in data mining [CDP04], management of authorizations in privacy protection [MMZ06], circuit drawing [EGB06], chemical reaction networks [Ozt08], protein pathways [EKC$^+$08], composition of web services [Fra07], quality indicators in business processes [MY07], assembly lines [GP92], cellular networks [KHT09].

Furthermore, hypergraphs and related problems on hyperpaths, have been used as a key tool to derive properties of combinatorial structures having, in turn, a wide coverage of applications, namely *AND-OR* graphs [MM73, Nil82, MSS04], and Petri nets [AFN92, AFLN10].

Classical works on hypergraphs, such as [Ber73, Ber89], cover partially directed hypergraphs, and some surveys, e.g., [Aus88, GLNP93, GS98, GS99] provide an early overview of applications and techniques, but a unified view relating common problems on hyperpaths in these areas seems to be missing.

Classifying measures over directed hyperpaths, the consequences which these have on the structure of the "optimal" hyperpaths, the corresponding computational cost, and the algorithms that are available to compute statically and dynamically optimal (or suboptimal) hyperpaths is a research challenge aiming at creating synergies among a number of different areas by means of an unifying view of problems whose abstract formulation is essentially the same, or is reducible one another. The main goals are:

- collecting, classifying and exchanging solutions from different areas within a unified framework;
- providing evidence to the possible outcomes of algorithmic problems, focusing research towards targets with high rewards;
- creating the best premises in order to exploit new solutions to their full potential as soon they are devised.

Algorithms and data structures to find optimal hyperpaths have been developed by Knuth in the context of a *grammar problem* [Knu77]. In this succinct but seminal work, Knuth introduces an approach based on two balancing levers:

- a generalization of the metrics over directed hyperpaths, based on functions associated to production rules of a context-free grammar: in our point of view, each of such rules plays the role of a directed hyperarc;

- a restriction on the functions, which must comply certain properties, in order to preserve some form of monotonicity to be exploited in finding a solution with combinatorial algorithms: at this aim he introduces the *superior functions*.

Although Knuth does not mention explicitly the notion of "hypergraph" (but cites AND/OR graphs among the applications of his results), the generalization of Dijkstra's algorithm [Dij59] proposed in that paper can be considered the first solution for finding optimal hyperpaths, and works if we measure the hyperpaths by means of a superior function. In this case, for a hypergraph with $|N|$ nodes, $|H|$ hyperarcs, and an overall size of $|\mathcal{H}|$, this algorithm requires $O(|H| \log |N| + |\mathcal{H}|)$ worst case time, that can be reduced to $O(|N| \log |N| + |\mathcal{H}|)$ by using Fibonacci heaps [FT87] for the implementation of priority queues.

The possible use of this algorithm has been extended beyond the superior functions. Ramalingam and Reps considered several problems in [RR96], where they primarily introduce the output bounded complexity and propose a dynamic algorithm for the single source shortest path problem. In the same paper they introduce the *weakly superior functions (WSUP)* and, discussing the grammar problem based on Knuth, they stressed that *"this problem also subsumes the problem of finding optimal hyperpaths in directed hypergraphs under varying optimization criteria"* [RR96]. Hence, they show that this algorithm actually works correctly for the larger class of *WSUP* functions as well.

Many further contributions to the knowledge of directed hypergraphs and to connectivity and optimization problems have been developed as within specific applicative contexts. As an example, this is the case of transitive closure of functional dependencies in databases [ADS83], or traffic assignment in transportation problems [NP88, MN98].

In this paper we analize the possible formulations of a "measure" function over directed hyperpaths. According to how we define this measure, the problem of finding an optimal hyperpath can be NP-hard, or tractable. We show that a source of "hardness" is due to a combinatorial constraint over the *set* of hyperarcs. If one is interested to optimize only the "measure" of the resulting hyperpath, i.e., if the function is *value-based*, the resulting problem is tractable. But there are more problems: for many interesting cases, even with naive measure functions (such as minimizing the weight of the *last* hyperarc), the optimal hyperpaths can be *cyclic*. If one want to deal with these cases, an explicit representation of cycles cannot be avoided.

In order to capture these cases that, although tractable, still lack a solution in the literature, and following the work by Knuth and Ramalingam-Reps, we extend their work in various ways.

The superior functions introduced by Knuth are based on two properties: $(i)$ a *multidimensional triangle inequality*, which is partially relaxed in the *WSUP* functions by Ramalingam and Reps, and $(ii)$ the *monotonicity* with respect to all arguments. We prove that this last property - alone - is generic enough in order to model a variety of interesting functions arising in practice, whereas triangle inequality does not hold. Hence monotonicity is sufficient in order to guarantee fundamental properties, and to make some optimization algorithms effective: if this property holds we have a *generalized superior function (GSUP)*. We also consider the symmetrical classes of *inferior* functions, building up a hierarchy of optimization problems over directed hyperpaths.

Then we analyze optimal hyperpaths and introduce forms of equivalence coming up to another hierarchy of optimization problems based on the structure that they induce on the optimal hyperpaths; hence we relate the two hierarchies. We discuss forms of *canonical* representation of optimal hyperpaths, and show that, for any optimization problem of a *GSUP*/*GINF* measure function, there exists an optimal hyperpath that can be represented unambiguously in linear space. Such canonical representations are interesting both for the design and analysis of algorithms to find optimal hyperpaths, since it is sufficient to consider few cases having a regular well known (minimal) structure.

Finally we analyze within a common algorithmic pattern some existing and new algorithms to deal with the problem of finding optimal hyperpaths, comparing and discussing their effectiveness within the various classes of problems. One of these algorithms can find and return optimal cyclic hyperpaths in $O(|H| \log |N| + |\mathcal{H}|)$ worst case time.

## 2 Basic Definitions

Different definitions of directed hypergraph are presented in literature. For instance, Gallo et al. [GLNP93] provide quite a general definition of directed hypergraph by allowing both the source and the target of a hyperarc to be non-singleton sets of nodes.

According to their terminology, particular cases of hyperarcs are: *backward* hyperarcs (or *B-arcs*), having a single node as a target, *forward* hyperarcs (or *F-arcs*), having a single node as a source, and *BF-hyperarcs*, with multiple source and sink nodes. Depending on the types of allowed hyperarcs, one may have *B-Hypergraphs*, *F-Hypergraphs*, or *BF-Hypergraphs*. Within this nomenclature, in this paper we are considering only B-Hypergraphs.

If we are given a certain type of hyperarcs, there are different ways of defining hyperpaths, i.e., connections based on a suitable collection of hyperarcs. As an example the several notions of "hyperpath" proposed in the classical work by Gallo et al. [GLNP92] do not include the "L-hyperpaths", considered in the recent work by Thakur and Tripathi [TT09], that provide a comparison of various definitions.

Finally, for a given hyperpath, there are still many possible ways to "measure" it; this issue and its consequences will be considered in the following sections.

**Definition 2.1** *A* directed hypergraph $\mathcal{H}$ *is a pair* $\langle N, H\rangle$, *where* $N$ *is a set of* nodes *and* $H \subset 2^N \times N$ *is a set of* hyperarcs. *Each hyperarc is an ordered pair* $h = \langle S, t\rangle$, *where the* source set *(or* tail*)* $S \subseteq N$ *is an arbitrary nonempty set of nodes, and the* target node *(or* head*)* $t \in N$ *is a single node.*

In many cases, given a hypergraph, we need to discuss the topological properties of the underlying graph, resulting from the transformation of each hyperarc in a set of arcs. An example is provided in Figure 1.

**Definition 2.2** *Given a directed hypergraph* $\mathcal{H} = \langle N, H\rangle$, *its* graph reduction *is the directed graph* $G(\mathcal{H}) = \langle N, A\rangle$, *where* $A = \{(x_i, y) \mid \exists X \subseteq N \text{ such that } \langle X, y\rangle \in H \text{ and } x_i \in X\}$.

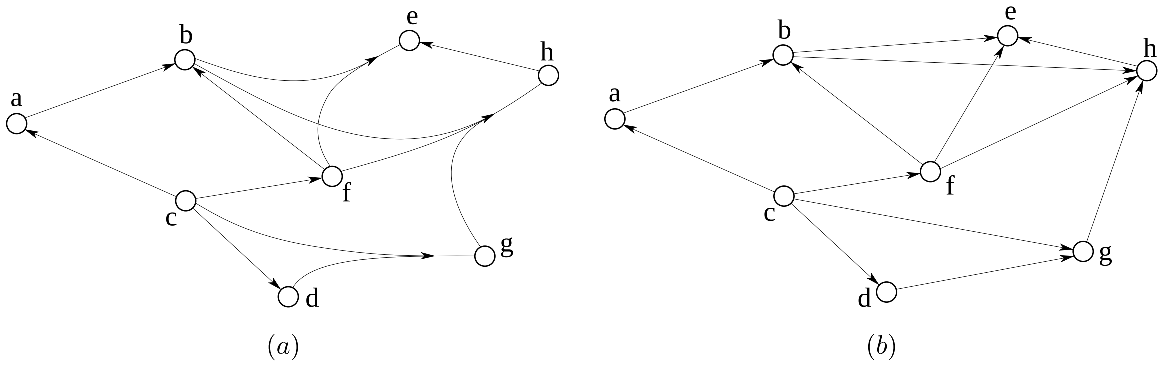


Figure 1: ($a$) A directed hypergraph and ($b$) its corresponding graph reduction $G(\mathcal{H})$.

**Definition 2.3** *A* weighted directed hypergraph $\mathcal{H}_W$ *is a triple* $\langle N, H; w\rangle$, *where* $\langle N, H\rangle$ *is a directed hypergraph and each hyperarc* $\langle S, t\rangle \in H$ *is associated to a real value* $w_{\langle S,t\rangle} \in \Re$ *called* weight of the hyperarc.

Note that a directed graph is a special case of directed hypergraph in which all the source sets have cardinality one.

**Definition 2.4** *Given a node* $n$, *the* forward star *of* $n$, *or* $fstar(n)$, *is the set of all its outgoing hyperarcs (i.e., hyperarcs having node* $n$ *in the source set), while the* backward star *of* $n$, *or* $bstar(n)$, *is the set of all its incoming hyperarcs, i.e., hyperarcs whose target node is* $n$.

*The* outdegree *of* $n$ *is the cardinality of its forward star, while the* indegree *of* $n$ *is the cardinality of its backward star, i.e.,* $outdegree(n) = |fstar(n)|$ *and* $indegree(n) = |bstar(n)|$.

An example including the previous definitions is presented in Figure 1. Observe that $\mathit{fstar}(b) = \{\langle bf, e\rangle, \langle bfg, h\rangle\}$, hence $\mathit{outdegree}(b) = 2$; $\mathit{bstar}(b) = \{\langle a, b\rangle, \langle f, b\rangle\}$ ($\mathit{indegree}(b) = 2$), while $\mathit{bstar}(c) = \emptyset$, hence $\mathit{indegree}(c) = 0$.

Since the source set of a hyperarc may have cardinality greater than one, the property that the sum of the indegree of all nodes is equal to the sum of the outdegree of all nodes cannot be extended from directed graphs to directed hypergraphs.

A *self-loop* is a hyperarc whose target node appears also in the source set.

**Definition 2.5** *A directed hypergraph $\mathcal{H}' = \langle N', H'\rangle$ is a* subhypergraph *of $\mathcal{H} = \langle N, H\rangle$ (denoted as $\mathcal{H}' \subseteq \mathcal{H}$) if:*

*a) $N' \subseteq N$,*

*b) $H' \subseteq H$, and, for each hyperarc $\langle S, t\rangle \in H'$, $S \subseteq N'$ and $t \in N'$,*

*Furthermore, let $H' \subseteq H$ be a set of hyperarcs in $\mathcal{H}$. Let $N' \subseteq N$ be the union of source sets and target nodes of hyperarcs in $H'$. The hypergraph $\mathcal{H}' = \langle N', H'\rangle$ is said to be the* subhypergraph of $\mathcal{H}$ induced by $H'$.

Before defining the notion of hyperpath in directed hypergraphs, we relate some simple graphs definitions, like paths, walks and cycles, to directed hypergraphs, since they will be recalled afterwards.

**Definition 2.6** *A* (directed) walk *of length $k$ in a directed hypergraph from a node $x$ to a node $y$, is a sequence of nodes and hyperarcs*

$$[x \equiv n_1, h_1, n_2, h_2, \ldots, h_k, n_{k+1} \equiv y]$$

*such that, for each $j = 1, \ldots, k$, $h_j = \langle S_j, n_{j+1}\rangle \in H$, and $n_j \in S_j$. A* (directed) cycle *is a walk of length $k \geq 1$ having $n_1 = n_{k+1}$. A walk is* acyclic*, or* simple*, if it does not contain any cycle (i.e., if all nodes are distinct).*

We remark that a walk in a hypergraph $\mathcal{H}$ is bijectively associated with a path in its graph reduction $G(\mathcal{H})$. Unlike the definition of path, we define (the existence of) a hyperpath in a recursive way.

**Definition 2.7** *Let $\mathcal{H} = \langle N, H\rangle$ be a directed hypergraph, $X \subseteq N$ be a non-empty subset of nodes, and $y$ be a node in $N$. There is a* hyperpath *from $X$ to $y$ in $\mathcal{H}$ if either*

*a) $y \in X$ (*extended reflexivity*);*

*b) there is a hyperarc $\langle Z, y\rangle \in H$ and hyperpaths from $X$ to each node $z_i \in Z$ (*extended transitivity*).*

*If there exists a hyperpath from $X$ to $y$ we say that $y$ is* reachable *from $X$ and, in case b), that hyperarc $\langle Z, y\rangle$ is* traversable*.*

The above recursive definition of hyperpath can be naturally represented by a tree defined as follows:

**Definition 2.8** *Let $\mathcal{H} = \langle N, H\rangle$ be a directed hypergraph, $X \subseteq N$ be a non-empty subset of nodes, and $y$ be a node in $N$. A* hyperpath *(or* unfolded hyperpath *or* hyperpath tree*) from $X$ to $y$ (if it exists) is a tree $t_{X,y}$ recursively defined as follows:*

*a) for each (sub)hyperpath obtained by extended reflexivity, the corresponding (sub)tree is empty;*

*b) if, by extended transitivity, there is a hyperarc $\langle Z, y\rangle \in H$ and hyperpaths from $X$ to each node $z_i \in Z$, then $t_{X,y}$ consists of a root labeled with hyperarc $\langle Z, y\rangle$ having as subtrees the hyperpath trees $t_{X,z_i}$ from $X$ to each node $z_i \in Z$;*

*A* branch *of $t_{X,y}$ is a path from the root to a leaf node of $t_{X,y}$.*

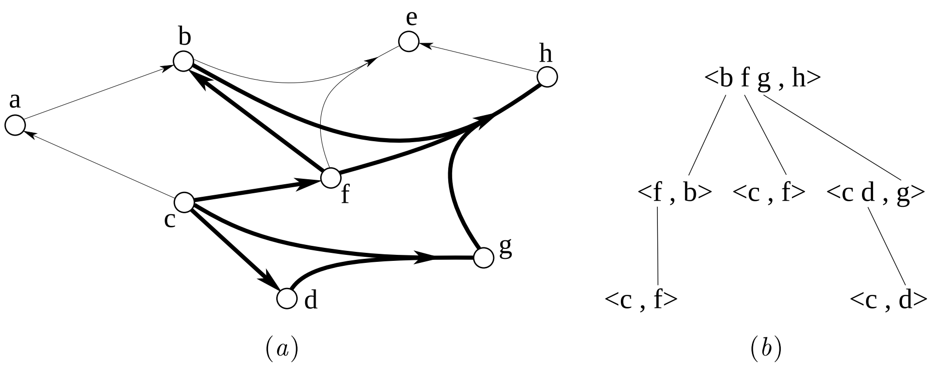


Figure 2: A hyperpath from $c$ to $h$ is highlighted with bolded hyperarcs in *(a)*. The corresponding hyperpath tree ($t_{c,h}$) is shown in *(b)*.

Note that the root of the hyperpath tree $t_{X,y}$ is a hyperarc in $bstar(y)$. Furthermore if $\langle S, t\rangle$ is a leaf in the hyperpath tree, then it must be $S \subseteq X$. An example of hyperpath tree is presented in Figure 2.

This representation explicitly describes the sequence of hyperarcs as traversed while going from $X$ to $y$. There is however an alternative and more concise way of describing hyperpaths, defined as follows:

**Definition 2.9** *Let $\mathcal{H} = \langle N, H\rangle$ be a directed hypergraph and let $t_{X,y}$ be a hyperpath from a set of nodes $X \subseteq N$ to a target node $y \in N$. The* folded hyperpath *$h(t_{X,y})$ corresponding to $t_{X,y}$ is the subhypergraph of $\mathcal{H}$ induced by the hyperarcs in $t_{X,y}$.*

It is interesting to observe that there is not a one-to-one relationship between unfolded and folded hyperpaths, since distinct (unfolded) hyperpaths may have the same folded representation.

Note also that these two representations can be used in the case of paths in a directed graph as well: given a path $\pi$ from a node $x$ to a node $y$, we can describe $\pi$ either by providing the sequence of all the edges in $\pi$ as traversed while going from $x$ to $y$ (unfolded description), or by providing the subgraph of $G$ containing exactly the edges of $\pi$ (folded description) (see, for instance, Figure 3). While the former description may contain the same edge more than once and may not even be finite, since it may contain a cycle which is traversed an unbounded number of times, the latter description is more compact but it may hide some features of the actual path.

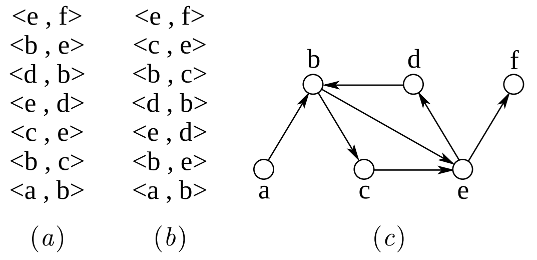
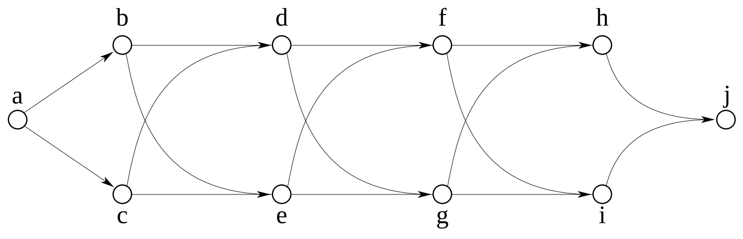
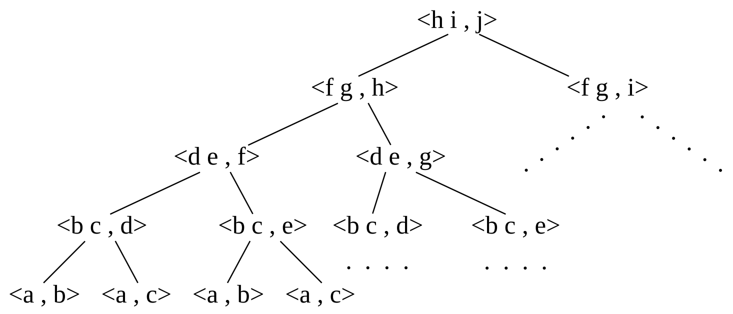


Figure 3: In a directed graph, the unfolded representation of two different paths from $a$ to $f$, such as *(a)* and *(b)*, may have the same folded representation *(c)*.

While turning from graphs to hypergraphs, we can notice moreover that there is an even deeper difference between folded and unfolded hyperpaths: unlike simple paths, in fact, there are even acyclic hyperpaths whose unfolded tree representation is exponentially larger than the corresponding folded representation. An example is shown in Figure 4. Nevertheless, compared to the traditional folded version, unfolded hyperpaths are a sharp and unambiguous representation; therefore, first of all we must analyze cyclic hypergraphs and hyperpaths, that is the topic of the following section.



(a)



(b)

Figure 4: A folded hyperpath $h_{a,j}$ and, below, its unfolded representation $t_{a,j}$ of exponential size.

## 3 Cycles in Hypergraphs and Hyperpaths

In this section we address the structure of hyperpaths in presence of cycles. Analysis of several alternative definition of cycles and acyclicity in directed hypergraphs has been considered in many studies (see, e.g., [ADS86, GLNP93, TT09]).

**Definition 3.1** *A directed hypergraph $\mathcal{H} = \langle N, H \rangle$ is* cyclic *if it contains at least one directed cycle, otherwise it is* acyclic.

Also for directed hypergraphs we can relate the acyclicity of a hypergraph with the existence of a topological ordering of its nodes.

**Lemma 3.1** *[GLNP93] A directed hypergraph $\mathcal{H} = \langle N, H \rangle$ is* acyclic *if and only if there exists a topological ordering of its nodes $\langle n_{i_1}, n_{i_2}, \dots, n_{i_n} \rangle$ such that, for each hyperarc $h = \langle S, t \rangle \in H$, each node in $S$ precedes $t$ in the ordering.*

Now we extend the notion of cyclicity to hyperpaths. The following definition relies upon the trivial observation that a folded hyperpath is - anyway - a hypergraph.

**Definition 3.2** *A hyperpath $t_{X,y}$ is* cyclic *if and only if the corresponding folded hyperpath $h(t_{X,y})$ is cyclic, otherwise it is* acyclic.

Since we need to deal with cyclic optimal hyperpaths, it would be useful to have a definition of cyclicity that can be directly applied (and checked) on hyperpath trees. However, unlike simple graphs, this task is not trivial, and requires a deeper understanding of such structures. The rest of this section is therefore devoted to introduce some concepts which will help us to efficiently characterize and manipulate hyperpaths. First we define basic measures.

**Definition 3.3** *Let $\mathcal{H} = \langle N, H \rangle$ be a directed hyperpath, and let us consider any hyperpath tree $t_{X,y}$ in $\mathcal{H}$ and its corresponding folded hyperpath $h(t_{X,y})$. We define:*

- Node-indegree *of a node $n$: the indegree of node $n$ in $h(t_{X,y})$, denoted as* N-INDEG$_{t_{X,y}}(n)$*;*
- Node-outdegree *of a node $n$: the outdegree of node $n$ in $h(t_{X,y})$, denoted as* N-OUTDEG$_{t_{X,y}}(n)$*;*
- Node-multiplicity *of a node $n$ in the unfolded hyperpath $t_{X,y}$, denoted as* N-MULT$_{t_{X,y}}(n)$*: this is the maximum number of times that node $n$ appears as target in a single branch of $t_{X,y}$.*

*Each of these quantities,* Node-indegree, Node-outdegree*, and* Node-multiplicity*, are defined on the hyperpath $t_{X,y}$ as the maximum of the corresponding quantity over all the nodes.*

The representation of a hyperpath $t_{X,y}$ has singularities when there is a cycle passing through the extremal nodes: $x_i \in X$, and/or $y$. In such situations we will consider a dummy "start" node with dummy input arcs for every $x_i \in X$, and - symmetrically - a dummy output arc from node $y$ to a dummy "end" node. These will not affect the structure or the measure of the hyperpaths at hand: we will address these special cases when required.

Note that both node-indegree and node-outdegree are defined on the folded and unfolded hyperpath representations. Actually, the node-indegree of a hyperpath $t_{X,y}$ is the maximum number of distinct hyperarcs in $t_{X,y}$ having the same node as target, and the outdegree is the maximum number of distinct hyperarcs in $t_{X,y}$ having the same node in the source.

**Lemma 3.2** *Any hyperpath $t_{X,y}$ having* N-MULT$(t_{X,y}) \geq 2$ *is cyclic.*

**Proof.** Let $h$ and $h'$ be two hyperarcs having the same node, say $n$, as target and belonging to the same branch of $t_{X,y}$ (we assume that $h$ is the deeper hyperarc). The sequence of hyperarcs belonging to the subbranch delimited by $h$ and $h'$, $\langle h = h_1, \ldots, h_k = h' \rangle$ contains a directed path from $n$ to itself in $h(t_{X,y})$. Therefore, according to Definition 3.2, the hyperpath $t_{X,y}$ is cyclic. $\square$

Note that the converse of Lemma 3.2 does not hold. There are cyclic hyperpaths whose unfolded structure does not have the same target on the same branch, i.e., N-MULT $= 1$: an example is shown in Figure 5. Let us consider any nonempty hyperpath tree $t_{X,y}$ and the corresponding folded

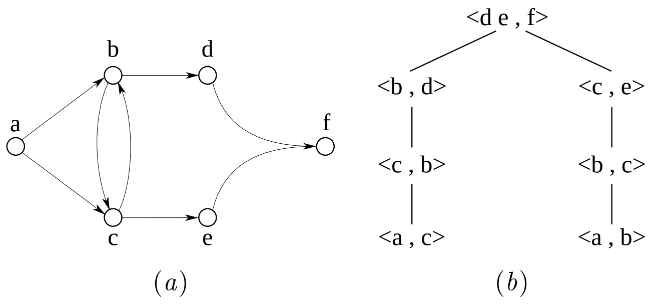


Figure 5: *(a)* folded and *(b)* unfolded representation of a cyclic hyperpath $t_{a,f}$ with N-INDEG$(t_{a,f}) = 2$ (since *indegree*$(b) =$ *indegree*$(c) = 2$), but N-MULT$(t_{a,f}) = 1$.

hyperpath $h(t_{X,y})$. Since any hyperarc in $h(t_{X,y})$ must appear (at least once) in $t_{X,y}$, any walk in the unfolded hyperpath either is entirely within a single branch of the tree, or is is *fragmented* among more branches: the latter case occurs when there are at least two nodes with indegree and/or outdegree larger than 1. More precisely, let us suppose that a node $n$ in an unfolded hyperpath $h(t_{X,y})$ has *indegree*$(n) = k$; then each hyperarc in $(X_i, n) \in$ *bstar*$(n)$, with $i = 1, 2, \ldots, k$, appears (at least) once in different branches of $t_{X,y}$. Analogously, if a node $n$ in an unfolded hyperpath $h(t_{X,y})$ has has *outdegree*$(n) = k$, then each hyperarc in $(\ldots n \ldots, y_i) \in$ *fstar*$(n)$ appears (at least) once in different branches of $t_{X,y}$.

If we consider the nodes along a cycle, if any node $n$ has N-INDEG$(n) > 1$ then it is an *input node* for that cycle, and if N-OUTDEG$(n) > 1$ then it is an *output node.* For any hyperpath $t_{X,y}$ and any cycle $C$, a branch of $t_{X,y}$ can contain a portion of cycle $C$ which starts from an input node (the lower bound of cycle $C$ in the branch) and ends at an output node (the upper bound of $C$ in the branch). In order to avoid a paradox while accounting for the degree, for any hyperpath $t_{X,y}$, if a cycle $C$ includes also a node $x_i$ in the source set $X$, then $x_i$ is also an input node for $C$; if the cycle contains the target node $y$, this is an output node for $C$. In this way, any cycle has at least one input node and at least one output node.

An example of cycle fragmented in a hyperpath tree with N-MULT $= 1$ is given in Figure 6).

The next Lemma formalizes the properties discussed above.

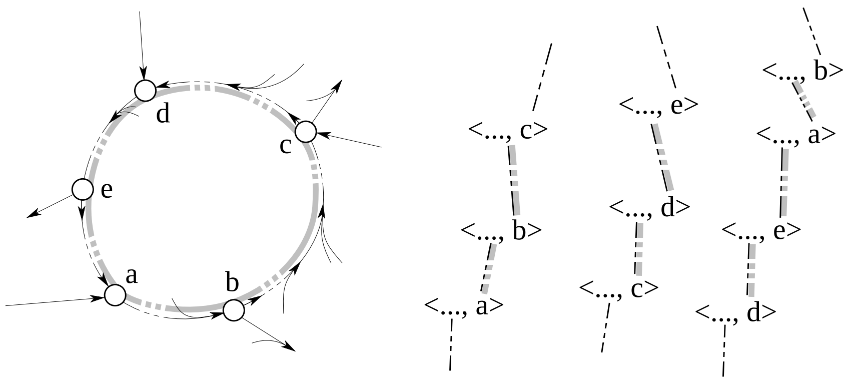


Figure 6: A cycle and a possible fragmentation within a hyperpath tree: the input nodes of the cycle (possible lower bounds of chains in a hyperpath tree) are $a, c, d$, whilst output nodes (possible upper bounds in a tree) are $b, c, e$.

**Lemma 3.3** *Let us consider any nonempty cyclic hyperpath tree $t_{X,y}$ and the corresponding folded hyperpath $h(t_{X,y})$. Any cycle in $h(t_{X,y})$ is contained within the branches $t_{X,y}$ as a collection of one or more chains which cover the cycle.*

*Any chain has a lower end at an input node of the cycle (including the source nodes in $X$) and an upper end at an output node (including the target node $t$).*

In order to investigate the structure of cyclic hyperpaths, we need to introduce operations which transform their structure.

**Definition 3.4** *Let $t_{X,y}$ be a hyperpath, and let $s_{X,z}$ and $s'_{X,z}$ be two distinct (possibly empty if $z \in X$) subtrees of $t_{X,y}$, representing two (distinct) hyperpaths from $X$ to $z$. We define as* (internal) subtree replacement *the operation of removing the subtree $s_{X,z}$ and replacing it with a copy of $s'_{X,z}$.*

Note that, if we replace a subtree $s_{X,z}$ with another generic hyperpath tree from $X$ to $z$, we might introduce new hyperarcs taken from the set $H$ in the original hyperpath. On the contrary, an *internal* replacement (considered in the previous definition) can only reduce the set of distinct hyperarcs used in $t_{X,y}$. The notation $t_{X,y} \rightsquigarrow t'_{X,y}$ denotes the fact that hyperpath $t'_{X,y}$ is obtained by a (possibly empty) sequence of subtree replacements on the originary hyperpath $t_{X,y}$. Note that $t_{X,y} \rightsquigarrow t'_{X,y}$ implies $h(t'_{X,y}) \subseteq h(t_{X,y})$.

In the hyperpath tree $t_{a,f}$ shown in Figure 5($a$), we can replace the subtree $s_{a,c} = \{\langle b, c\rangle, \langle a, b\rangle\}$ (on the right branch), with the subtree $s'_{a,c} = \{\langle a, c\rangle\}$ (copied from the left branch), leading to a new acyclic hyperpath tree $t'_{a,f}$, with N-INDEG$(t') = 1$. In this case, the following relationships hold: $t_{a,f} \rightsquigarrow t'_{a,f}$, but $t'_{a,f} \not\rightsquigarrow t_{a,f}$; furthermore $h(t'_{a,f}) \subsetneq h(t_{a,f})$, and N-INDEG$(t') <$ N-INDEG$(t)$. On the other side, the opposite replacement, i.e., if we replace subtree $s'_{a,c} = \{\langle a, c\rangle\}$ in the left branch with subtree $s_{a,c} = \{\langle b, c\rangle, \langle a, b\rangle\}$ on the right, we get a new hyperpath tree $t''_{a,f}$ with N-INDEG$(t'') >$ N-INDEG$(t)$.

The examples show that, as a consequence of subtree replacements, node indegree can only decrease, whilst node multiplicity can decrease or increase: the latter case may occur only if node indegree is greater than one.

**Lemma 3.4** *A hyperpath $t_{X,y}$ is* cyclic *if and only if there exists a hyperpath $t'_{X,y}$, such that $t_{X,y} \rightsquigarrow t'_{X,y}$ and* N-MULT$(t'_{X,y}) \geq 2$.

**Proof.**
($\Longrightarrow$) If $t_{X,y}$ is cyclic, then let us consider any cycle in $h(t_{X,y})$: according to Lemma 3.3 either this cycle is within a unique branch in $t_{X,y}$, or it is split in at most $k$ chains on as many branches. In the first case, the node multiplicity of the hyperpath tree is already larger than 1. In the latter case, any of these chains, say $c_i$, consists of a walk between an input node $n_{i-1}$ and an output node $n_i$, for $i = 1, 2, \dots, k$, $n_0 \equiv n_k$ and $n_{k+1} \equiv n_1$; we will denote as $s_{\text{high,i}}$ and $s_{\text{low,i}}$ the two subtrees whose root is, respectively, a hyperarc with target $n_{i-1}$ and $n_i$ (see Figure 7($a$)).

Let us consider two consecutive of these chains, in two different branches, $c_i$ and $c_{i+1}$. We can replace $s_{\text{low,i+1}}$ with $s_{\text{high,i}}$, since the have the same target node $n_i$ (see Figure 7). The effect of this replacement, which is still a hyperpath from $X$ to $y$, is to concatenate two successive chains. Therefore, after a sequence of at most $k$ internal subtree replacements, we can collate all the cycle in a unique branch, hence the node multiplicity of the new hyperpath tree is at least 2.

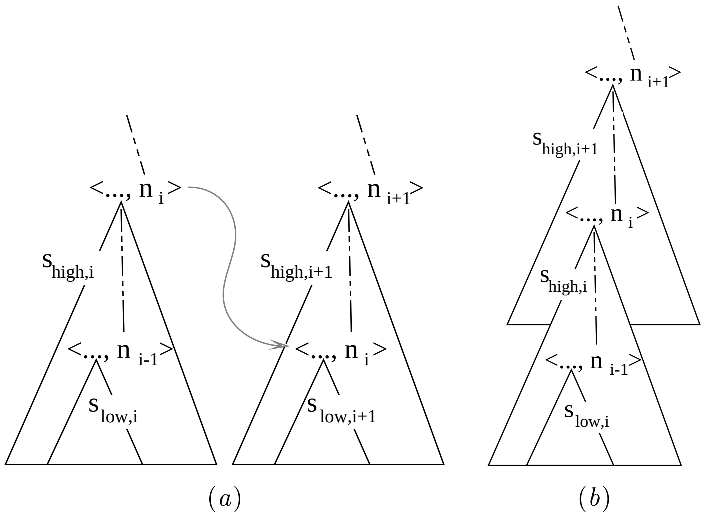


Figure 7: A subtree replacement within a cyclic hyperpath, as proposed in Lemma 3.4: *(a)* before and *(b)* after the replacement.

($\Longleftarrow$) Let us suppose now that, starting form a hyperpath $t_{X,y}$, we have obtained, by means of a finite sequence of subtree replacements, a new hyperpath $t'_{X,y}$ such that N-MULT$(t'_{X,y}) \geq 2$. By Lemma 3.2, $h(t'_{X,y})$ is cyclic, and since $h(t'_{X,y}) \subseteq h(t_{X,y})$, $h(t_{X,y})$ is cyclic as well. □

In the following we investigate the relationships between the structure of a hyperpath and its behavior under subtree replacements; in particular, Lemma 3.5 characterizes a hyperpath $t_{X,y}$ which is *replacement-invariant*, i.e., if $t_{X,y} \rightsquigarrow t'_{X,y}$ then $t'_{X,y} = t_{X,y}$.

**Lemma 3.5** *If we are given a directed hyperpath tree $t_{X,y}$, we have that:* N-INDEG$(t_{X,y}) = 1$ *if and only if $t_{X,y}$ is invariant with respect to subtree replacements.*

**Proof.**
*($\Longrightarrow$)* In this case we assume that N-INDEG$(t_{X,y}) = 1$ and, by contradiction, that there exist two different subtrees $s_{S,t}$ and $s'_{S,t}$ in $t_{X,y}$. The roots of the two subtrees, $h$ and $h'$ have the same target $t$. If $h$ and $h'$ have different source sets, there are two distinct hyperarcs with target $t$, which implies N-INDEG$(t_{X,y}) \geq$ N-INDEG$(t) \geq 2$. If this is not the case, we explore in parallel the two subtrees until we find different hyperarcs, respectively in $s$ and $s'$ with the same target node $z$, but different source sets, that would imply N-INDEG$(t_{X,y}) \geq$ N-INDEG$(z) \geq 2$ and, again, we would have a contradiction.
*($\Longleftarrow$)* Let us suppose, by contradiction, that there exists a node $z$ such that N-INDEG$_{t_{X,y}}(z) \geq 2$. Then there are at least two distinct hyperarcs, $\langle S_1, z\rangle$ and $\langle S_2, z\rangle$ which are roots of two sub hyperpaths $s_1$ and $s_2$ from $X$ to $z$ in $t_{X,y}$. Since the roots are different, necessarily $s_1 \not\equiv s_2$, and a subtree replacement of an instance of $s_1$ with a copy of $s_2$ would lead to a tree $t'_{X,y} \not\equiv t_{X,y}$. □

As a consequence of Lemmas 3.2 and 3.5 we have that a replacement-invariant hyperpath $t_{X,y}$ is acyclic. Again, also in this case, the reverse property does not hold, i.e., there are acyclic hyperpaths which are not replacement-invariant (e.g., see Figure 8).

Another consequence, due to Lemma 3.4, is that for any hyperpath $t_{X,y}$, N-INDEG$(t_{X,y}) = 1$ implies N-MULT$(t_{X,y}) = 1$. In Figure 9 we summarize the relationships between the classes of cyclic and acyclic hyperpaths having N-INDEG $> 1$ and/or N-MULT $> 1$.

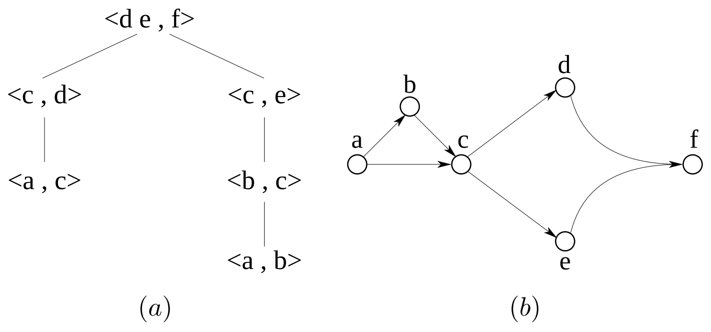


Figure 8: $(a)$ An acyclic hyperpath having N-INDEG$(t_{a,f}) = 2$; $(b)$ the corresponding folded representation.

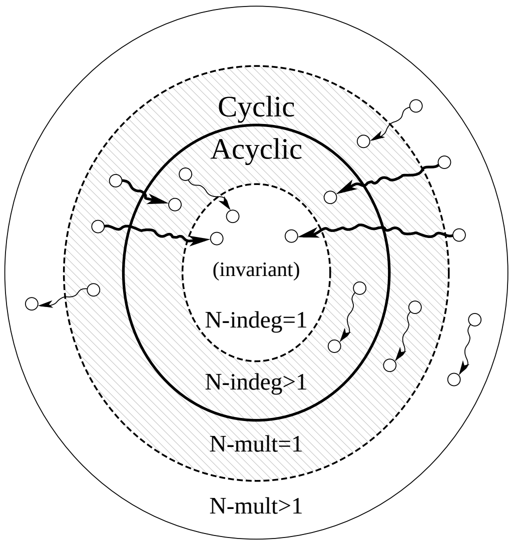


Figure 9: Node indegree and node multiplicity in cyclic and acyclic hyperpaths. Spots represent instances of hyperpath trees, and arrows represent the effects of all possible subtree replacements.

## 4 Tractable and Untractable Measure Functions

Several metrics have been introduced in order to measure directed hyperpaths, leading to a number of corresponding optimization problems. In this section we list several definitions that have been considered in the literature, starting with the very intuitive ones. We will show that in many cases the resulting optimization problem turns out to be untractable.

**Definition 4.1** *Let $\mathcal{H} = \langle N, H \rangle$ be a directed hypergraph, and $h_{X,y} = \langle N_h, H_h \rangle$ be a directed hyperpath from a source set $X \subseteq N$ to a node $y \in N$. The following are measure functions over directed hyperpaths:*

- *The* number of nodes*: $n(h_{X,y}) = |N_h|$;*
- *The* number of hyperarcs*: $h(h_{X,y}) = |H_h|$;*
- *The* source area *$a(h_{X,y}) = \sum_{S \in \mathcal{S}} |S|$ is the sum of cardinalities of all the source sets, where $\mathcal{S}$ is the set of the source sets in the hyperpath: $\mathcal{S} = \{S \mid \langle S, t \rangle \in H_h\}$;*
- *The* nonsingleton source area *$a'(h_{X,y}) = \sum_{S \in \mathcal{S}_M} |S|$ is the sum of cardinalities of all the nonsingleton source sets, where $\mathcal{S}_M$ denotes the set of nonsingleton source sets: $\mathcal{S}_M = \{S \mid \langle S, t \rangle \in H_h,\ \text{with } |S| > 1\}$;*
- *The* size *$s(h_{X,y}) = n + a' + h = \Theta(n + a + h)$ is the overall length of the description of the hypergraph (also denoted as $|\mathcal{H}|$).*

*Furthermore, if we are given a weighted directed hypergraph $\mathcal{H} = \langle N, H; w \rangle$, and a hyperpath $h_{X,y} = \langle N_h, H_h \rangle$, its* total weight *$w(h_{X,y})$ is the sum of the weights of all its hyperarcs:*

$$w(h_{X,y}) = \sum_{\langle Z,t \rangle \in H} |w_{\langle Z,t \rangle}|$$

The size is the length of a natural description of a hyperpath, represented as a collection of: (1) the nodes in $N_h$, (2) the additional nonsingleton source sets of all the hyperarcs in $H_h$ (i.e., the source sets that do not appear as a single node in $N_h$) plus (3) the adjacency lists of the source sets; each item in such lists is a target node of a hyperarc, hence the total length of this last contribution is $h$. Note that $a = n + a'$.

In the special case where a directed hypergraph is a directed graph, the number of vertices is equal to the number of nodes $n$, and the number of edges is $m = h$. Furthermore, $a' = 0$, and $s = n + m$.

The set of measures provided in Definition 4.1 bring to optimization problems whose complexity is *NP*-hard, as shown in the next theorem. These metrics are related to hyperpaths regarded as hypergraphs: actually the same metrics may be applied to hypergraphs. Ausiello et al. [ADS86] study similar properties concerning a problem of minimal representation of directed hypergraphs.

**Theorem 4.1** *Let $\mathcal{H} = \langle N, H \rangle$ be a directed hypergraph, $x$ and $y$ be two nodes in $N$, and $k$ be an integer. Consider the following problems.*
*($\mathcal{P}_1$) Find a hyperpath $h_{x,y}$ with $k$ hyperarcs or less;*
*($\mathcal{P}_2$) Find a hyperpath $h_{x,y}$ of total weight $k$ or less;*
*($\mathcal{P}_3$) Find a hyperpath $h_{x,y}$ of size $k$ or less;*
*($\mathcal{P}_4$) Find a hyperpath $h_{x,y}$ of source area $k$ or less;*
*($\mathcal{P}_5$) Find a hyperpath $h_{x,y}$ of $k$ source sets or less.*
*All these problems ($\mathcal{P}_1$)—($\mathcal{P}_5$) are NP-complete.*

**Proof.** We consider the problems separately.
($\mathcal{P}_1$). We use a reduction from *Minimum Cover* (in short MC) [GJ79]. Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set, and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be a family of subsets of $A$ such that $\bigcup_{i=1,2,\ldots,m} S_i = A$. We recall that, given $A$, $\mathcal{S}$, and an integer $k$, MC consists of finding a cover of cardinality $k$ or less, that is, a subfamily $\mathcal{S}'$ of $\mathcal{S}$ such that $|\mathcal{S}'| \leq k$, and $\bigcup_{S_i \in \mathcal{S}'} S_i = A$.

Let $\mathcal{I} = \langle A, \mathcal{S}, k \rangle$ be an instance of MC. We now define a directed hypergraph $\mathcal{H}_{\mathcal{I}} = \langle N_{\mathcal{I}}, H_{\mathcal{I}} \rangle$ as follows. The set of nodes is $N_{\mathcal{I}} = A \cup \mathcal{S} \cup \{p, q\}$ and the set of hyperarcs is $H_{\mathcal{I}} = H_1 \cup H_2 \cup H_3$, where (see Figure 10):

$H_1 = \{\langle p, S_i \rangle | S_i \in \mathcal{S}\}$,

$H_2 = \{\langle S_i, a_j \rangle | S_i \in \mathcal{S}$ and $a_j \in S_i\}$, and

$H_3$ consists of the single hyperarc $\langle A, q \rangle$.

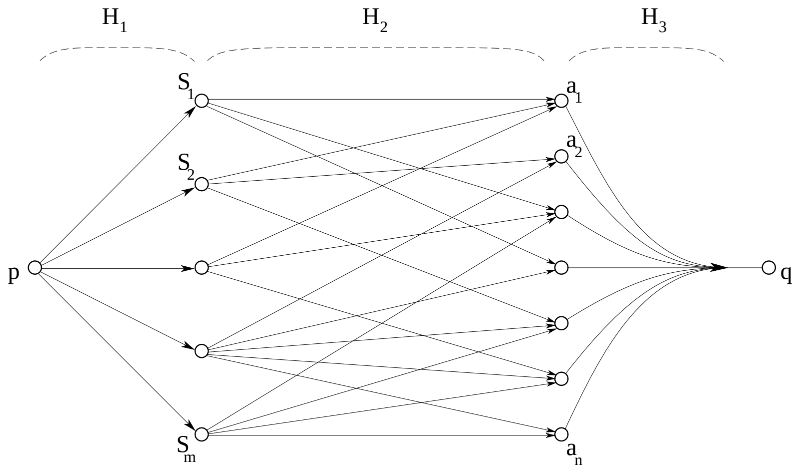


Figure 10: The hypergraph associated to an instance of MC. The nodes $a_1, a_2, \ldots, a_n$ on the right represent the set to be covered, while the nodes $S_1, S_2, \ldots, S_m$ on the left represent the collection of subsets.

We now show that there exists a hyperpath from $p$ to $q$ in $\mathcal{H}_{\mathcal{I}}$ having no more than $k + n + 1$ hyperarcs if and only if there exists a cover $\mathcal{S}'$ of $A$ whose cardinality is less than or equal to $k$.

Let us assume that there is a feasible solution for MC, that is $\bigcup_{S_i \in \mathcal{S}} S_i = A$; in this case $\mathcal{H}_{\mathcal{I}}$ always contains a hyperpath $h_{p,q}$ from $p$ to $q$ defined by transitivity (see Definition 2.7):

- the hyperarc $\langle A, q \rangle$;
- the collection of hyperpaths $h_{p,a_j}$, where each of these is a chaining of two simple arcs: $\langle p, S_i \rangle$ and $\langle S_i, a_j \rangle$.

This is possible if and only if for each $a_j \in A$ there exists some $S_i$ which contains $a_j$, that is if $\mathcal{S}$ includes a cover for $A$. Note that if the cover has cardinality $k$, then there is a hyperpath with $k + n + 1$ hyperarcs in $\mathcal{H}_{\mathcal{I}}$.

Vice versa let $h_{p,q}$ be a hyperpath with $k_n$ hyperarcs. As shown above, $h_{p,q}$ must contain the hyperarc $\langle A, q \rangle$, plus $n$ hyperarcs from the set $H_2$, plus, say, $k$ hyperarcs from the set $H_1$, with:

$$k = k_n - n - 1.$$

Let $\mathcal{S}'$ be the target nodes $S_{i_1}, S_{i_2}, \ldots S_{i_k}$ of the $k$ hyperarcs in the set $h_{p,q} \cap H_1$. Then $\mathcal{S}'$ is a cover for the set $A$. In fact for any node $a_j \in A$ there exists a hyperarc $\langle S_i, a_j \rangle$ in $h_{p,q} \cap H_2$, and by construction, this means that any element of the set $A$ is contained in some $S_i \in \mathcal{S}'$.

Therefore the problem of finding a cover with cardinality $k$ is reduced to the problem of finding a hyperpath $h_{p,q}$ with a number of hyperarcs:

$$n(h_{p,q}) = k_n = k + n + 1.$$

$(\mathcal{P}_2)$. The NP-completeness of the problem of finding a hyperpath with $k$ hyperarcs or less follows immediately from the observation that this is a special case of $(\mathcal{P}_2)$, when all the weights are 1.

$(\mathcal{P}_3)$—$(\mathcal{P}_5)$. To prove the NP-completeness of these problem, we observe that, using the same reduction given for $(\mathcal{P}_1)$, there is a cover with cardinality $k$ or less if and only if there is a hyperpath $h_{p,q}$ having, respectively:

- size $k_s = 2k + 2n + 2$ or less;
- source area $k_a = k + n + 1$ or less;
- number of source sets $k + 2$ or less; these are: the collection of $k$ nodes corresponding to a solution for MC, plus the two source sets $\{p\}$ and $\{a_1, a_2, \dots, a_n\}$.

□

Since the above decision problems are NP-complete, the NP-hardness of the corresponding optimization problems follows immediately.

**Corollary 4.2** *Let $\mathcal{H} = \langle N, H\rangle$ be a directed hypergraph, $x$ and $y$ be two nodes in $N$. The following problems are NP-hard:*

*i) finding a hyperpath with minimum* number of hyperarcs*;*

*ii) finding a hyperpath with minimum* total weight*;*

*iii) finding a hyperpath with minimum* size*;*

*iv) finding a hyperpath with minimum* source area*;*

*v) finding a hyperpath with minimum* number of source sets.

Several other measures, which are of interest in some applications, can be defined in a recursive way, based on the structure of folded and/or unfolded hyperpaths.

In many situations the interesting hyperpaths are acyclic: in this case there is no sensible difference between the two formulations. We start by considering a very common function.

**Definition 4.2** *The rank $r(h_{X,y})$ of an acyclic folded hyperpath $h_{X,y}$ is the maximum cost path from the root to a leaf in the hyperpath, and is recursively defined as follows:*

*a) if $h_{X,y}$ is an empty hypergraph then $r(h_{X,y}) = 0$;*

*b) if $h_{X,y}$ has one hyperarc $\langle Z, y\rangle$ entering $y$, with $Z = \{z_1, z_2, \dots, z_k\}$ and $h_{X,z_i} \subset h_{X,y}$, then:* $r(h_{X,y}) = w_{\langle Z,y\rangle} + \max_{z_i \in Z}\{r(h_{X,z_i})\}$.

*The rank $r(t_{X,y})$ of an (unfolded) hyperpath $t_{X,y}$ is defined as follows:*

*a) if $t_{X,y} = \emptyset$ ($y \in X$) then:* $r(t_{X,y}) = 0$;

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \dots, t_{X,z_k}$, then:* $r(t_{X,y}) = w_{\langle Z,y\rangle} + \max_{z_i \in Z}\{r(t_{X,z_i})\}$.

Since we are interested in dealing with cyclic hyperpaths, the following additional definitions will be based on the more general formulation on unfolded hyperpaths, but we remark that the same definitions may be used also for the more common folded representations, in case of acyclic hyperpaths. In case that cyclic hyperpaths have to be considered, the unfolded data structure has the advantage of providing an explicit representation of the multiplicity of each hyperarcs.

**Definition 4.3** *The gap $g(t_{X,y})$ of a hyperpath $t_{X,y}$ is the minimum cost path from the root to a leaf in the hyperpath, and is recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ ($y \in X$) then:* $g(t_{X,y}) = 0$;

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \dots, t_{X,z_k}$, then:* $g(t_{X,y}) = w_{\langle Z,y\rangle} + \min_{z_i \in Z}\{g(t_{X,z_i})\}$.

**Definition 4.4** *The average-depth $avgd(t_{X,y})$ of a hyperpath $t_{X,y}$ is the average length of the paths from the root to any leaf in the hyperpath tree, and is recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ ($y \in X$) then: $avgd(t_{X,y}) = 0$;*

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \dots, t_{X,z_k}$, then:* $avgd(t_{X,y}) = w_{\langle Z,y\rangle} + \mathrm{avg}_{z_i \in Z}\{avgd(t_{X,z_i})\}$.

**Definition 4.5** *The traversal cost $c(t_{X,y})$ of a hyperpath $t_{X,y}$ is the weight of the root of the hyperpath, plus the cost of all its subtrees, and is recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ $(y \in X)$ then: $c(t_{X,y}) = 0$;*

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k}$, then:*
$c(t_{X,y}) = w_{\langle Z,y\rangle} + \sum_{z_i \in Z}\{c(t_{X,z_i})\}$.

**Definition 4.6** *Given a hypergraph, with real hyperarc weights in the range $(0,1]$, the P-Product (P-Prod$_{X,y}$) of a hyperpath $t_{X,y}$ is computed by multiplying the weight of the root of the hyperpath by the P-Prod of all its subtrees; this is recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ $(y \in X)$ then: P-Prod$(t_{X,y}) = 1$;*

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k}$, then:*
*P-Prod*$(t_{X,y}) = w_{\langle Z,y\rangle} \times \Pi_{z_i \in Z}\{$*P-Prod*$(t_{X,z_i})\}$.

**Definition 4.7** *The bottleneck $b(t_{X,y})$ of a hyperpath $t_{X,y}$ is defined as the minimum cost of a hyperarc in $t_{X,y}$, and can be recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ $(y \in X)$ then: $b(t_{X,y}) = +\infty$;*

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k}$, then:*
$b(t_{X,y}) = \min\ \{\ w_{\langle Z,y\rangle}, \min_{z_i \in Z}\{b(t_{X,z_i})\}\ \}$.

**Definition 4.8** *The threshold $t_t(t_{X,y})$ of a hyperpath $t_{X,y}$ is defined as the maximum cost of a hyperarc in $t_{X,y}$, and can be recursively defined as follows:*

*a) if $t_{X,y} = \emptyset$ $(y \in X)$ then: $t(t_{X,y}) = 0$;*

*b) if $t_{X,y}$ has root $\langle Z, y\rangle$ with subtrees $t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k}$, then:*
$t(t_{X,y}) = \max\ \{\ w_{\langle Z,y\rangle}, \max_{z_i \in Z}\{t(t_{X,z_i})\}\ \}$.

If we consider the specific case of directed graphs, where the previous definitions are well defined, most of these collapse. As an example, we have that *gap*, *rank*, *average-depth*, number of hyperarcs, source area, number of source sets have all the same value (*size* is the double of these), and all the corresponding optimization problems on graphs can be solved in polynomial time.

On the other side, while dealing with directed hypergraphs, we have seen that some of these functions bring to untractable optimization problems but others can be optimized in polynomial time, possibly with several optimization strategies. But we will show as, even within these tractable cases, the different measure functions induce different structures of optimal hyperpaths that, in turn, will force to discriminate among the optimization strategies.

The measure functions provided here are examples that allow us to tackle with several combinatorial properties, and therefore are useful in order to characterize both classes of functions and to prove properties of the algorithms discussed in the next Sections. But, beyond their role of examples, any of them is representative of a number of different problems from many domains that can be formulated as an optimization problem according that metrics.

## 5 Generalized Superior and Inferior Functions

In the literature, several measure functions on directed hyperpaths have been considered. As shown in section 4, some of these measures, that are based the underlying hypergraphs (and hence may be thought as applied to folded hyperpaths), bring to *NP*-hard optimization problems.

The scenario completely changes if we consider instead measures over hyperpaths based on the recursive structure of these, and may be thought as applied to unfolded hyperpaths.

Well known NP-hard problems on (hyper)paths, or reducible to these, originate from constraints characterizing the sequence, or simply the set, of (hyper)arcs to be traversed. This is the case, e.g., of the Traveling Salesman Problem, or the Hamiltonian Path Problem. In these cases the formulation require to state properties of the considered path. As an example, let us consider the Longest Path Problem, that is the problem of finding a *simple* path in a given graph having maximum length. The condition of "simple" path requires that any arc can be traversed at most once.

We have seen as the problem of finding a hyperpath of minimum *size* is NP-hard (Theorem 4.1): the notion of *size* requires to sum the weights of the *set* of hyperarcs. More precisely, in order to compute the *size* of a hyperpath, one cannot rely solely upon the *values* of the *size* of the subhyperpaths, since the weight of each hyperarc has to be considered at most once, and one is forced to take into account the sets of hyperarcs constituting the subhyperpaths.

A unified view of an interesting class of tractable problems on directed hypergraphs is provided by Knuth. In his definition of grammar problem [Knu77], Knuth generalizes the single-source shortest path problem to context free grammars, and provides a classification on the functions associated to the productions of the grammar. In the following we define the superior functions [Knu77] and introduce the symmetric notion of inferior functions.

**Definition 5.1** *Let $g(x_1, \ldots, x_k)$ be a function from $D^k$ into $D$, monotone nondecreasing in each variable.*

- *$g$ is a* superior function *on $D^k$ ($g \in SUP$) if, for each $\langle x_1, \ldots, x_k \in D^k \rangle$ [Knu77]:*

$$g(x_1, \ldots, x_k) \geq \max(x_1, \ldots, x_k);$$

- *$g$ is an* inferior function *on $D^k$ ($g \in INF$) if, for each $\langle x_1, \ldots, x_k \in D^k \rangle$:*

$$g(x_1, \ldots, x_k) \leq \min(x_1, \ldots, x_k).$$

Examples of *SUP* functions are $\max\{x_1, \ldots, x_k\}$ in $\Re^k$, and $\Sigma_i\{x_i\}$ in $[0, +\infty]^k$. Examples of *INF* functions are: $\min\{x_1, \ldots, x_k\}$ in $\Re^k$, and the product $\Pi_i\{x_i\}$ in $[0, 1]^k$.

Ramalingam and Reps in [RR96] introduce a generalization of these classes of functions. Also in this case we provide the dual (inferior) functions by reversing the inequality.

**Definition 5.2** *Let $g(x_1, \ldots, x_k)$ be a function from $D^k$ into $D$, monotone nondecreasing in each variable.*

- *$g$ is a* weakly superior function *in $D^k$ ($g \in WSUP$) if, for each $x_1, \ldots, x_k \in D^k$ and, for each $i = 1, \ldots, k$ [RR96]:*

$$g(x_1, \ldots, x_k) < x_i \;\Rightarrow\; g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$$

- *$g$ is a* weakly inferior function *in $D^k$ ($g \in WINF$) if, for each $x_1, \ldots, x_k \in D^k$, and for each $i = 1, \ldots, k$:*

$$g(x_1, \ldots, x_k) > x_i \;\Rightarrow\; g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, -\infty, \ldots, x_k)$$

It is obvious that if a function is *SUP*, then it is *WSUP* (*SUP* $\subset$ *WSUP*), and if it is *INF*, then it is *WINF* (*INF* $\subset$ *WINF*). Examples of *WSUP* functions that are not *SUP* are $\min_{1 \leq i \leq k}\{x_i\}$ and any constant function. Examples of *WINF* functions that are not *INF* are $\max_i\{x_i\}$ and any constant function.

The following lemma summarizes the properties holding in the case of a generic composition of these functions, generalizing analogous statements of the cited authors.

**Lemma 5.1** *If we are given the functions $f, g_1, \ldots, g_h$ then their composition $f(g_1(\ldots), \ldots, g_h(\ldots))$ has the following properties:*

1. *if $f, g_1, \ldots, g_h \in SUP$, then $f(g_1, \ldots, g_h) \in SUP$ [Knu77];*
2. *if $f, g_1, \ldots, g_h \in INF$, then $f(g_1, \ldots, g_h) \in INF$;*
3. *if $f, g_1, \ldots, g_h \in WSUP$, then $f(g_1, \ldots, g_h) \in WSUP$ [RR96];*
4. *if $f, g_1, \ldots, g_h \in WINF$, then $f(g_1, \ldots, g_h) \in WINF$.*

Both [Knu77] and [RR96] consider also the class of *strict (weakly)* superior and inferior functions, characterized by a strict inequality between the value of the function at hand and each of its arguments: this leads to the classes *SSUP*, *SWSUP*, *SINF*, and *SWINF*.

For example a function $g(x_1, \ldots, x_k)$ from $D^k$ into $D$ is a *strict superior function* (*SSUP*) in $D$ if it is monotone nondecreasing in each variable and if:

$$g(x_1, \ldots, x_k) > \max(x_1, \ldots, x_k), \text{for each } x_1, \ldots, x_k \in D^k$$

Also for strict functions we can state a lemma summarizing the properties holding in the case of a generic composition (also in this case we extend similar properties asserted by [Knu77, RR96]).

**Lemma 5.2** *If we are given the functions $f, g_1, \ldots, g_h$ then their composition $f(g_1(\ldots), \ldots, g_h(\ldots))$ has the following properties:*

1. *if $f \in SSUP$, and $g_i \in SUP$ for all $i$ (or vice versa) then $f(g_1, \ldots, g_h) \in SSUP$;*
2. *if $f \in SINF$, and $g_i \in INF$ for all $i$ (or vice versa) then $f(g_1, \ldots, g_h) \in SINF$;*
3. *if $f \in SWSUP$, and $g_i \in WSUP$ for all $i$ (or vice versa) then $f(g_1, \ldots, g_h) \in SWSUP$;*
4. *if $f \in SWINF$, and $g_i \in WINF$ for all $i$ (or vice versa) then $f(g_1, \ldots, g_h) \in SWINF$.*

Some of the measure functions defined in the previous section can be classified within this framework. For example, if we consider a weighted hypergraph having all positive weights, the computation of the traversal costs will result in a *SSUP* function.

The *rank* function (Definition 4.2) is *SUP* if all the hyperarcs weights are non-negative, since *rank* is defined as the sum (a *SUP* function) of the weight of the last hyperarc with the maximum (*SUP*) *rank* of the subhyperpaths; hence Lemma 5.1 holds. Notice that if hyperarc weights are all positive, then rank is *SSUP*, since the sum is *SSUP* in this case, and hence Lemma 5.2 holds.

A very similar measure function, the *gap* (Definition 4.3), is defined as the sum (a *SUP* function) of the weight of the last hyperarc with the *minimum gap* of the subhyperpaths. The latter function is *WSUP* in case of non-negative weights, and *SWSUP* with positive weights: as a result, the *gap* function inherits this characterization.

Any measure function on directed hyperpaths whose value does not depend on the measure of its subhyperpaths is in $SWSUP \cap SWINF$. Let us consider the following simple measure function.

**Definition 5.3** *Given any hyperpath $t_{X,y}$ in a weighted directed hypergraph $\mathcal{H}_W$, the value of $last(t_{X,y})$ is the weight of the last hyperarc:*

$$last(t_{X,y}) = \begin{cases} 0 & \text{if } t_{X,y} = \emptyset, \text{ i.e., if } y \in X \\ w_{\langle Z,y \rangle} & \text{if } t_{X,y} \text{ has root } \langle Z,y \rangle \text{ with subtrees } t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k} \end{cases}$$

Note that this function is not *SUP* nor *INF*, since its value can be either larger or smaller of any of its argument. Nevertheless, according Definition 5.2, this function is both *SWSUP* and *SWINF*, since $last(t_{X,y}) = w_{\langle Z,y \rangle}$, intended as a function of the measure of its component subhyperpaths, does not change its value if one of its arguments is increased up to infinity.

This measure function is apparently naive: nevertheless, it is easy to check that a hyperpath of minimum (or maximum) *last* in general is cyclic.

In some cases, as for many graph problems, the nature of the domain of the edge cost may change the complexity of the resulting optimization problem. Let us consider a directed hypergraph $\mathcal{H}_W = \langle N, H; W \rangle$ whose weights can be expressed by a binary function $W : H \to [0, 1]$. The value of $gems(t_{X,y})$ of a hyperpath $t_{X,y}$ is the number of hyperarcs with weight 1 (the "gems" to be collected): in other words, this is a special case of the *traversal cost*, in the special case of a binary weight function.

On the other side, many functions leading to polynomial optimization problems do not fit into the classes introduced so far. As an example, another quite natural measure function, such as the *average-depth* (Definition 4.4), whose value is always between *gap* and *rank*, is not *WSUP* nor *WINF*. We will see that this feature has consequences on the corresponding optimization problem.

In the following, for generality, we will refer our definitions to a *"domain"* $D$, to be intended as a totally ordered set, where any two values in $D$ are comparable according an ordering relation "$\preceq$". There exist an *infimum* $a$ and a *supremum* $b$ (not necessarily in $D$) such that, for each element $x \in D$, $a \preceq x \preceq b$. As an example, one may consider the set of nonnegative reals, $\Re^+$: with the comparison relation is "$\leq$", the infimum is $0 \in \Re^+$ and the supremum is $+\infty \notin \Re^+$.

In case of a limited domain $D$, the definition of weakly superior and inferior functions should be intended based on the quantities that delimitate the domain itself, respectively: "$+\infty$" has to be intended as "supremum($D$)", and "$-\infty$" has to be intended as "infimum($D$)". This allow us to use this concepts also for domains which are widely used in applications, such as the reals in $[0, 1]$.

In order to generalize the definitions by Knuth and Ramalingam-Reps, we define classes of measure function that can be applied to unfolded hyperpaths. First of all, analogously to what was proposed for the Grammar Problem [Knu77], where each production has a corresponding function, we consider a hypergraph where each hyperarc has an associated function, analogously to [RR96].

**Definition 5.4** *Given a directed hypergraph $\mathcal{H} = \langle N, H \rangle$, a* functional hypergraph $\mathcal{H}_F = \langle N, H; \mathcal{F} \rangle$, *is defined as follows. Each hyperarc $\langle X, y \rangle \in H$ is associated to a triple $\langle w_{\langle X,y \rangle}, \psi_{\langle X,y \rangle}, f_{\langle X,y \rangle} \rangle$, where:*

$w_{\langle X,y \rangle} \in D$ *is the* weight *of the hyperarc;*

$\psi_{\langle X,y \rangle}$ *is a function from $|X|$-tuples of reals to reals: $\psi_{\langle X,y \rangle} : D^{|X|} \to D$;*

$f_{\langle X,y \rangle}$ *is a function from a pair of reals to reals: $f_{\langle X,y \rangle} : D^2 \to D$.*

*Furthermore, $\mathcal{F}$ is the collection of the functions associated to the hyperarcs in $\mathcal{H}$, i.e., $\mathcal{F} = \{F_{\langle X,y \rangle} | \langle X, y \rangle \in H\}$, where each $F_{\langle X,y \rangle} : D^k \to D$ is a function defined as follows:*

$$F_{\langle X,y \rangle}(x_1, x_2, \ldots, x_k) = f_{\langle X,y \rangle}(w_{\langle X,y \rangle}, \psi_{\langle X,y \rangle}(x_1, x_2, \ldots, x_k)).$$

In other words, each hyperarc $\langle X, y \rangle$ is associated to a corresponding function $F_{\langle X,y \rangle}(x_1, x_2, \ldots, x_k)$, as a combination of three components: the weight $w_{\langle X,y \rangle}$ and the two functions, $f$ and $\psi$. Comparing this structuring with the *SUP/ WSUP* functions introduced by Knuth and Ramalingam-Reps, the decomposition of $F$ in the triple $\langle w, f, \psi \rangle$ does not limit the expressivity of the formalism[1]. On the other side, after observing a number of definitions in practical applications, we found that:

- such a structuring occurs in most cases: we will show several examples below;
- based on this decomposition, it is easier to classify and cluster measure functions over directed hypergraphs according common properties (as shown in Section 6);
- it is possible to tailor algorithms relying on the disjoint features deriving from this structuring.

**Definition 5.5** *Given a functional directed hypergraph $\mathcal{H}_F = \langle N, H; \mathcal{F} \rangle$, and two constants $\mu_0$ and $\mu_\infty$, $\mu = \langle \mathcal{H}_F, \mu_0, \mu_\infty \rangle$ is a* Value-Based Measure Function *(VBMF) if $\mu : T(\mathcal{H}_F) \to D$ is a function from the set of the hyperpath trees in the hypergraph, $T(\mathcal{H}_F)$, to a totally ordered domain $D$ and, for any nonempty set of nodes $X \subseteq N$ and any node $y \in N$:*

- *if $X$ and $y$ are not connected, we assume that there is a conventional hyperpath $t_\infty$ connecting these nodes, with: $\mu(t_{X,y}) = \mu(t_\infty) = \mu_\infty \in D$;*
- *if $y \in X$, and $t_{X,y}$ is an empty hyperpath (defined by reflexivity), then: $\mu(t_{X,y}) = \mu_0 \in D$;*
- *if $t_{X,y} = \{\langle Z, y \rangle\} \cup t_{X,z_1} \cup t_{X,z_2} \cup \ldots \cup t_{X,z_k}$ is a hyperpath from $X$ to $y$ (defined by transitivity), then:*
$$\mu(t_{X,y}) = F_{\langle Z,y \rangle}(\mu(t_{X,z_1}), \mu(t_{X,z_2}), \ldots, \mu(t_{X,z_k})).$$

Notice that the recursive definition of value-based measure function on the structure of a hyperpath does not depend on combinatorial constraints on the set of affected hyperarcs, but only on the measures of the component sub-hyperpaths. The value of these measures must be taken from a domain $D$, where we can compare two measures according a total ordering $\preceq$: in other words, for any two hyperpaths in $T(\mathcal{H}_F)$, we can decide which is better, or if they are equivalent. If we consider the structure of an unfolded hyperpath, if a hyperarc is traversed more than once (and hence, it appears in more than one subtrees), its cost is repeatedly taken into account in all the subtrees.

Furthermore, since we are mainly aimed at algorithmic solution for tractable optimization problems, we need a simple strategy to build up optimal hyperpaths by combining optimal subhyperpaths; this can be guaranteed by the monotonicity of the considered functions. In this case we come up to a generalization of the *SUP/ WSUP* functions by Knuth and Ramalingam-Reps.

**Definition 5.6** *Given a functional directed hypergraph $\mathcal{H}_F = \langle N, H; \mathcal{F} \rangle$, and a Value-Based Measure Function $\mu = \langle \mathcal{H}_F, \mu_0, \mu_\infty \rangle$, for any nonempty hyperpath $t_{\langle S,t \rangle}$ in $\mathcal{H}_F$ with final hyperarc $\langle X, t \rangle$ ($X = \{x_1, x_2, \ldots, x_k\}$):*

- *if $F_{\langle X,t \rangle}(x_1, x_2, \ldots, x_k)$ is monotonic nondecreasing in each $x_i$, then it is a* Generalized Superior Function (*GSUP*)*;*
- *if $F_{\langle X,t \rangle}(x_1, x_2, \ldots, x_k)$ is monotonic nonincreasing in each $x_i$, then it is a* Generalized Inferior Function (*GINF*).

[1]If we chose $f(A, B) = B$ (in this case $w_{\langle X,y \rangle}$ is not meaningful), the definition collapse to $F = \psi(x_1, x_2, \ldots, x_k)$.

*In both cases we consider the "strict" version of the two classes, i.e., by comparing $\mu(t_{\langle X,y\rangle})$ with respect to the measure of each subhyperpath: a value based measure function is GSSUP (resp. GSINF) if it is monotone increasing (resp. decreasing) in each argument.*

We remark that, in some cases, a *SUP* function becomes *SSUP* by imposing a restriction on the allowed weights; as an example, this is the case of *rank*, or *traversal cost*, which are *SSUP* only in case of (strictly) positive weights.

Since we are interested in the composition of functions, also in this case we provide the basic properties of compositions for the last classes of measure functions.

**Lemma 5.3** *If we are given the functions $f, g_1, \ldots, g_h$ then their composition $f(g_1(\ldots), \ldots, g_h(\ldots))$ has the following properties:*

1. *if $f, g_1, \ldots, g_h \in$ GSUP, then $f(g_1, \ldots, g_h) \in$ GSUP;*
2. *if $f, g_1, \ldots, g_h \in$ GSSUP, then $f(g_1, \ldots, g_h) \in$ GSSUP;*
3. *if $f, g_1, \ldots, g_h \in$ GINF, then $f(g_1, \ldots, g_h) \in$ GINF;*
4. *if $f, g_1, \ldots, g_h \in$ GSINF, then $f(g_1, \ldots, g_h) \in$ GSINF*
5. *if $f, g_1, \ldots, g_h \in$ VBMF, then $f(g_1, \ldots, g_h) \in$ VBMF.*

Since we want to tailor algorithms for the best performances, we want to exploit the specific properties of functions that occur in practice and make them easier to be optimized.

In computing a function, there might be variables that do not determine the value of the function, provided that they are subject to some constraint. This is not always the case; as an example, the sum and the average functions depend on all the arguments. In some common cases, the function depends on a single value, while the others can change, with a given upper bound (such as the max) or a lower bound (such as the min) on the irrelevant variables. A function $g : D^k \to D$ will be labeled as $nU - relevant$ if, for any $(x_1, \ldots, x_k) \in D^k$, the value of $g(x_1, \ldots, x_k)$ depends on at most $n$ *relevant* variables. This means that, whenever $k > n$, there is always a set of $k - n$ *irrelevant* variables whose value can be changed in the interval $[-\infty, l] \cap D$, where $l$ is the upper bound, without changing the value of $g$. For an $nL - relevant$ function $g$ we have an analogous situation, with the irrelevant variables constrained by a given lower bound.

Any function *WSUP* function which is not *SUP* has at least one irrelevant variable with an upper bound (whenever it has at least 2 arguments). Considering some of the measure functions over directed hypergraphs introduced so far, we have that some of these have only relevant arguments, such as *traversal cost*, *P-Prod*, or *average-depth*. Instead, for some of them, their value depends on the measure of a unique subhyperpath; in particular *rank* and *threshold* are $1UR$ (i.e., there is an upper bound on the measure of the irrelevant subhyperpaths), while *gap* and *bottleneck* are $1LR$ (with a lower bound). Function *last* has no relevant variable, hence it is $0R$ (no bounds at all).

In the table below we recall a series of metrics on directed hyperpaths provided in section 4.

| measure | $\mu_0$ | $\mu_\infty$ | (transitivity) |
|---|---|---|---|
| function | $t_{X,y} = \emptyset$ | no hyp. | $t_{X,y} = \{\langle Z, y\rangle\} \cup t_{X,z_1} \cup t_{X,z_2} \cup \ldots \cup t_{X,z_k}$ |
| *rank* | 0 | $+\infty$ | $rank(t_{X,y}) = w_{\langle Z,y\rangle} + \max_{z_i \in Z}\{rank(t_{X,z_i})\}$ |
| *gap* | 0 | $+\infty$ | $gap(t_{X,y}) = w_{\langle Z,y\rangle} + \min_{z_i \in Z}\{gap(t_{X,z_i})\}$ |
| *average-depth* | 0 | $+\infty$ | $avgd(t_{X,y}) = w_{\langle Z,y\rangle} + \text{avg}_{z_i \in Z}\{avgd(t_{X,z_i})\}$ |
| *last* | 0 | $+\infty$ | $last(t_{X,y}) = w_{\langle Z,y\rangle}$ |
| *traversal cost* | 0 | $+\infty$ | $tcost(t_{X,y}) = w_{\langle Z,y\rangle} + \sum_{z_i \in Z}\{tcost(t_{X,z_i})\}$ |
| *P-Prod*[0, 1] | 1 | 0 | $P\text{-}Prod(t_{X,y}) = w_{\langle Z,y\rangle} \times \Pi_{z_i \in Z}\{P\text{-}Prod(t_{X,z_i})\}$ |
| *P-Prod*$[1, +\infty]$ | 1 | $+\infty$ | $P\text{-}Prod(t_{X,y}) = w_{\langle Z,y\rangle} \times \Pi_{z_i \in Z}\{P\text{-}Prod(t_{X,z_i})\}$ |
| *bottleneck* | $+\infty$ | 0 | $bn(t_{X,y}) = \min(w_{\langle Z,y\rangle}, \min_{z_i \in Z}\{bn(t_{X,z_i})\})$ |
| *threshold* | 0 | $+\infty$ | $ts(t_{X,y}) = \max(w_{\langle Z,y\rangle}, \max_{z_i \in Z}\{ts(t_{X,z_i})\})$ |

Table 1: A list of definitions of measure functions.

| function | growth | relevant args. |
|---|---|---|
| any constant (or "don't care") | *WSUP*, *WINF* | $0R$ |
| $+, \sum$ | *SUP* | |
| max | *SUP*, *WINF* | $1UR$ |
| min | *WSUP*, *INF* | $1LR$ |
| avg | *GSUP* | |
| $\times, \Pi$ $[0,1]$ | *INF* | |
| $\times, \Pi$ $[1,+\infty]$ | *SUP* | |

Table 2: Properties of basic functions.

All these functions are composed by means of few simple functions, whose properties, for our purposes, are listed below.

On the basis of these, we classify all the functions in the previous table in the proper class, specifying their "relevance", when meaningful. Since the combination of the values of the subhyperpaths depends on function $\psi$, this is the one which determines the resulting "relevance" of the measure function.

| measure function | $f(w,\psi)$ | $\psi(\mu_1,\ldots,\mu_k)$ | resulting growth | resulting relevance |
|---|---|---|---|---|
| *rank* | $+$ | max | *SUP* | $1UR$ |
| *gap* | $+$ | min | *WSUP* | $1LR$ |
| *average-depth* | $+$ | avg | *GSUP* | |
| *last* | $w$ | don't care | *WSUP*, *WINF* | $0R$ |
| *traversal cost* | $+$ | $\sum$ | *SUP* | |
| *P-Prod*$[0,1]$ | $\times$ | $\Pi$ | *INF* | |
| *P-Prod*$[1,+\infty]$ | $\times$ | $\Pi$ | *SUP* | |
| *bottleneck* | min | min | *WSUP*, *INF* | $1LR$ |
| *threshold* | max | max | *SUP*, *WINF* | $1UR$ |

Table 3: Properties of Measure Functions.

As remarked by Ramalingam and Reps [RR96], the Value Based Measures Functions introduced in Definition 5.5 would not generalize the *(W)SUP* classes if we do not allow different functions for each hyperarc (as in an earlier proposal [AIN92]). Still, there are differences between a Functional Hypergraph and the Grammar Problem, as discussed in detail by [RR96]. In this paper we focus on the model of Directed Hypergraphs.

In order to highlight the interest for the *GSUP* class of functions, we remark that a functional hypergraph $\mathcal{H}_F = \langle N, H; \mathcal{F} \rangle$ can be used as a representation of a *Stochastic* (or *Probabilistic*) *Context-Free Grammar.* This is a context-free grammar where each production is assigned a probability: in turn, these are used in natural language processing or to represent higher level aggregations of RNA molecules. Esparza et al. [EKL08] discuss this model while considering a System of Monotone Fixed-Point Polynomial Equations. As an example of what is (properly) enclosed in *VBMF*, but not in *GSUP*, if we remove the monotonicity constraint, we come up to the possibility of modeling a generic System of Fixed-Point Polynomial Equations.

We have the following containment relationships:

$$\mathit{SSUP} \subsetneq \mathit{SUP} \subsetneq \mathit{WSUP} \subsetneq \mathit{GSUP} \subsetneq \mathit{VBMF} \supsetneq \mathit{GINF} \supsetneq \mathit{WINF} \supsetneq \mathit{INF} \supsetneq \mathit{SINF}$$

$$\mathit{SSUP} \subsetneq \mathit{SWSUP} \subsetneq \mathit{GSSUP}, \mathit{SINF} \subsetneq \mathit{SWINF} \subsetneq \mathit{GSINF}$$

$$\text{kR-}\mathit{WSUP} \subsetneq \text{(k+1)R-}\mathit{WSUP} \subsetneq \mathit{WSUP} \text{ for } k \geq 0$$

$$\text{kR-}\mathit{WINF} \subsetneq \text{(k+1)R-}\mathit{WINF} \subsetneq \mathit{WINF} \text{ for } k \geq 0$$

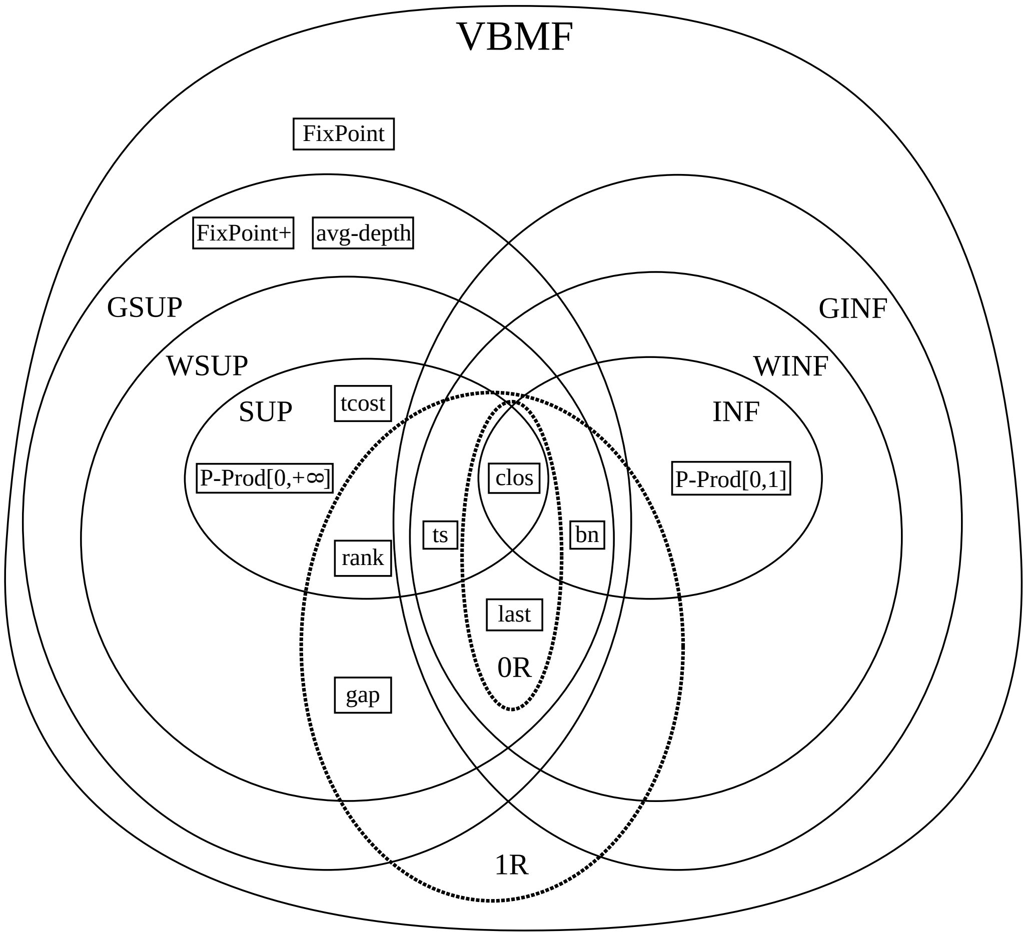


Figure 11: A taxonomy for Value-Based Measure Functions (minimization problems), with classification of some examples (*clos* = *closure*, *tcost* = *traversal cost*, *ts* = *threshold*, *bn* = *bottleneck*).

The diagram in Figure 11 summarizes these relationships, in case of minimization problems. The diagram by is populated by examples of measure functions presented in Section 4: these provide an evidence of the proper containment among the classes of functions considered above.

By comparing the classes of measure functions discussed in this Section, we have that, in the original definition by Knuth, a function $f$ is *SUP* if it is Value-Based, and is subject to the inequality constraint $f(x_1, \dots, x_k) \geq x_i$, for $i = 1, \dots, k$. This can be considered a *multidimensional triangle inequality*, extended to the case of directed hypergraphs. The original triangle inequality, $d[x, y] \geq d[x, z] + d[z, y]$, has a straighforward interpretation in directed and undirected graphs, where it is the basis of the definition of the usual notion of *distance*.

The multidimensional triangle inequality is partially relaxed in the *WSUP* functions by Ramalingam and Reps (by allowing, in our terminology, some irrelevant variable), and is completely removed in *GSUP* (and *GINF*) functions.

On the other side, monotonicity provides more generality and has interesting properties. As an example, the notion of $k$-relevance can be applied within the context of *GSUP/GINF* functions, regardless the inequality in the definition of the $(W)SUP$ functions: we will see the impact of this notion on the structure of the optimal hypergraphs, and on the optimization algorithms.

If monotonicity does not hold, the *Value Based* functions still offer a model where a problem can be formulated and dealt with a number of techniques, provided that the existing combinatorial constraints can be expressed by the structure of a directed hypergraph.

## 6 On the Structure of Optimal Hyperpaths

In Section 5 we have discussed a taxonomy for the Value-Based Measure Function, that is the input of the problem. In this section we provide a characterization of the structure of optimal hyperpaths under a given measure function $\mu$: this provides a classification of the measure functions according to the structure of the output; then we we relate these two taxonomies.

An optimization problem $\mathcal{P} = (\Phi, \mu)$ on directed hyperpaths is characterized by an *optimization criterion* $\Phi \in \{\min, \max\}$, and a measure function $\mu$ on hyperpaths.

In the following we use the notation $a \prec b$ (respectively, $a \preceq b$) to mean that the value $a$ is better (respectively, not worse) than the value $b$, according to an optimization criterion which is clarified in the context.

**Definition 6.1** *Given a functional hypergraph $\mathcal{H}_F$ and an optimization problem $\mathcal{P} = (\Phi, \mu)$, an* optimal hyperpath *from a set of nodes $X$ to a node $y$, is a hyperpath $t^*_{X,y}$ whose measure $\mu(t^*_{X,y})$ is the best (minimum or maximum) among the measures of all the hyperpaths from $X$ to $y$ in $\mathcal{H}$, i.e.:*

$$t^*_{X,y} \textit{ is optimal } \Leftrightarrow \textit{ for each hyperpath } t_{X,y} : \mu(t^*_{X,y}) \preceq \mu(t_{X,y})$$

We remark that it is always possible to transform the problem of finding a maximum, according a certain measure function $\mu$, in a problem of minimum according a "dual" measure function, and viceversa - usually in many possible ways. As an example, chosen among the metrics we have introduced so far, the bottleneck (the smallest weight in a hyperpath) and the threshold (the largest weight) may be used to build up a form of duality. If the weights used for bottleneck are real values $w_{\langle X,y\rangle} \in [0, W]$, we can define weights to be used with threshold as $w'_{\langle X,y\rangle} = W - w_{\langle X,y\rangle}$. Then we have: $(\max, \mathit{bottleneck}_w) \equiv (\min, \mathit{threshold}_{w'})$, and $(\min, \mathit{bottleneck}_w) \equiv (\max, \mathit{threshold}_{w'})$ where the equivalence means that an optimal hyperpath is the same for the two problems.

In Section 4 we have introduced the notion of replacement as a tool to analyze cycles, and let hypergraphs converge to a representation having specific properties. In this section we will use replacements among subtrees, in order to devise the mapping of each hyperpath to the corresponding canonical form.

The following Lemma expresses a basic property of hyperpaths obtained through replacements of a subtree. Note that this formulation refers to a generic subtree replacement, and not just internal replacements, that can be considered a special case. A replacement which does not worsen the measure of the current hyperpath tree will be called a *measure-preserving replacement*.

**Lemma 6.1** *Let $\mathcal{H}_F$ be a functional hypergraph and $\mathcal{P} = (\Phi, \mu)$ be an optimization problem, where $\Phi \in \{\min, \max\}$, and $\mu \in$ GSUP $\cup$ GINF.*

*Let $t_{X,y}$ be any hyperpath in $\mathcal{H}$, and $s_{X,z}$ be any subhyperpath of $t_{X,y}$. Let us consider another arbitrary hyperpath tree $s'_{X,z}$ from $X$ to $z$, and a new hyperpath tree $t'_{X,y}$ which is a copy of $t_{X,y}$, where $s$ has been replaced by $s'$. Then:*

$$\mu(s'_{X,z}) \preceq_\Phi \mu(s_{X,z}) \implies \mu(t'_{X,y}) \preceq_\Phi \mu(t_{X,y}).$$

**Proof.** Given a hyperpath tree $t_{X,y}$, we can explicit the definition of $\mu(t_{X,y})$ in terms of each of its subhyperpaths, until we find the subtree $s_{X,z}$. That is: $\mu(t_{X,y}) = \mu(\ldots, \mu(\ldots, \mu(s_{X,z}), \ldots), \ldots)$.

We recall that, according Definition 5.6, a *GSUP* (or *GINF*) function is simply a value-based measure function over directed hyperpaths which is monotone in each argument. This property is maintained under composition: by Lemma 5.3 the composition of *GSUP* (*GINF*) functions is *GSUP* (*GINF*). Then the value of $\mu(t_{X,y})$ is monotone in the value of $\mu(s_{X,z})$. □

According this Lemma, when we consider *GSUP* and *GINF* measure functions, the replacement of a subhyperpath with a "better" subhyperpath (according the metrics $\mu$ and the criterion $\Psi$) provide a new hyperpath which is not worse than the original one. This is consequence of the monotonicity, and not of the triangle inequality holding in the definition of the more restrictive *SUP* and, with a partial relaxation, in the definition of *WSUP* functions. We will show that the property stated by Lemma 6.1 has an impact on the structure of optimal hyperpaths.

We now give a characterization of optimization problems, based on the structure of the resulting optimal hyperpaths.

**Definition 6.2** *An optimization problem $\mathcal{P} = (\Phi, \mu)$ is* $k$-cycle-convergent *(k-CYCLE-CONV) for some $k \geq 0$ if, for any hypergraph $\mathcal{H}_W$ and for any optimal hyperpath $t^*_{X,y}$ from the set of nodes $X$ to the target node $y$ in $\mathcal{H}_W$, there exists an optimal hyperpath $\tilde{t}^*_{X,y}$ such that $t^*_{X,y} \rightsquigarrow \tilde{t}^*_{X,y}$ and* N-MULT$(\tilde{t}^*_{X,y}) \leq k + 1$.

*An optimization problem that is 0-CYCLE-CONV is said to be* cycle-invariant *(CY-INV).*

**Definition 6.3** *An optimization problem $\mathcal{P} = (\Phi, \mu)$ is* cycle-unbounded *(CYCLE-UNB) if it is not k-CYCLE-CONV, for any integer k.*

Note that, if we are given a *k-CYCLE-CONV* optimization problem on hyperpaths, there could exist an optimal hyperpath $t^*_{X,y}$ of unbounded size, but there always exists another optimal equivalent (bounded) hyperpath $\tilde{t}^*_{X,y}$ where each node is target at most of $k+1$ hyperarcs in the same branch of $\tilde{t}^*_{X,y}$. For a *CYCLE-UNB* optimization problem there exist optimal hyperpaths having no optimal equivalent bounded hyperpaths.

In Section 3 we have discussed the properties of a cyclic hyperpath tree $t_{X,y}$ where a cycle is fragmented on more branches, i.e., $t_{X,y}$ can have N-MULT$(t_{X,y}) = 1$. If this is the case then, from Lemma 3.4 we know that, by means of internal subtree replacements, we can transform it in a new hyperpath tree $t'_{X,y}$ with N-MULT$(t_{X,y}) \geq 2$, that is, with the cycle laying within a single branch. On the other side, internal subtree replacements in some cases can also transform the original tree in a way that it looses some hyperarcs, and may even become acyclic.

If we are given a *GSUP* measure function and any hyperpath tree $t_{X,y}$, we will show that it is always possible to transform it by means of a sequence of internal subtree replacements which are measure-preserving, and such that the final tree $t'_{X,y}$ has one of the following properties:

- either $t'$ is acyclic,or
- $t'$ is cyclic, but its node multiplicity is at least 2.

The interest for this property is that any cyclic optimal solutions has an equivalent hyperpath tree with the structure shown in Figure 12. In the next Lemma we prove this claim.

**Lemma 6.2** *Let $\mathcal{H}$ be a directed hypergraph, $\mu$ be any GSUP measure function and $t_{X,y}$ be any hyperpath tree. Then there exists a sequence of subtree replacements, with $t_{X,y} \rightsquigarrow t'_{X,y}$, where $\mu(t'_{X,y}) \leq \mu(t_{X,y})$ and furthermore: either* (1) N-MULT$(t'_{X,y}) \geq 2$, *or* (2) $t'_{X,y}$ *is acyclic.*

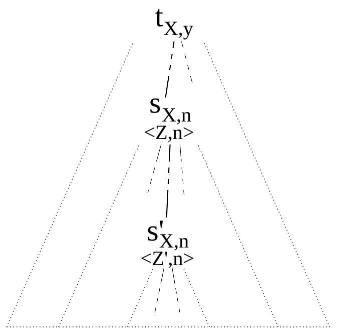


Figure 12: The structure of hyperpath tree with N-MULT$(t_{X,y}) \geq 2$. In particular, node $n$ appear as a target twice in a single branch.

**Proof.** The nontrivial case to be proved is when the hyperpath is cyclic and N-MULT$(t_{X,y}) = 1$.

By rephrasing Lemma 3.3 for our purposes, we know that any cycle appears in the branches of the hyperpath tree as one or more fragments which cover the whole cycle; each fragment starts at an input node of the cycle, and ends at an output node. If a cycle is entirely contained in a single branch of the hyperpath tree, then N-MULT$(t_{X,y}) \geq 2$, therefore we consider a case with more fragments.

At any stage of this procedure we consider a single cycle $C$ in $t_{X,y}$, and show how to build up a sequence of internal subtree replacements such that:

- each replacement is measure-preserving, i.e., it does not worsen the measure of the current hyperpath;
- at the end of the current stage, either $(a)$ cycle $C$ is entirely contained within a sigle branch, or $(b)$ a portion of cycle $C$ is completely removed from the hyperpath tree.

After a stage, in case $(a)$ we have completed the procedure and the option (1) of the thesis is satisfied, whilst in case $(b)$ we consider another cycle and start another stage; we will stop the transformation whenever case $(a)$ occurs, or go on until all cycles have been eliminated, and the final hyperpath tree complies the option (2) of the Lemma.

In each stage, we proceed by steps, and in each step, either we get a (strictly) longer fragment, or we break the cycle. We reiterate this step until either we swallow the whole cycle in a single branch of $t_{X,y}$, or we break this cycle.

In the generic step, let us consider a cycle $C$ that is fragmented among two or more branches. Since all its fragments must cover the whole cycle, there exist two consecutive fragments of cycle $C$. Two fragments, say the two chains $c_1 = \langle i_1, \ldots, o_1 \rangle$ and $c_2 = \langle i_2, \ldots, o_2 \rangle$, are "consecutive" if their endpoints satisfy the relationships: $i_1 < i_2 \leq o_1 < o_2$, where the inequalities refer to a circular ordering along cycle $C$ (see Figure 13). Both chains $c_1$ and $c_2$ contain a subtree rooted at $i_2$ (let us denote them as $s_1$ and $s_2$), but only $c_1$ contains the interval $[i_1 - i_2]$.

Then we search, within the whole tree $t_{X,y}$, the occurrence of any hyperarc where node $i_2$ is the target node, and the subtree rooted at this occurrence. The subtree $s^*$ with minimum measure, according function $\mu$, is the winner of such a tournament, i.e., $\mu(s^*) \leq \mu(s)$ for any subtree of $t_{X,y}$ having $i_2$ as the target node. Then we use $s^*$ as a replacement throughout $t_{X,y}$ wherever there is any subtree rooted at $i_2$, including $s_1$ and $s_2$.

Let us focus on the consequences of these replacements on the measure of the current hyperpath. The $j$-th of these replacements takes place on the tree $t^j_{X,y}$: in the current hyperpath tree $t^j_{X,y}$ we replace any subtree $s_i$ having a measure $\mu(s_i)$, with a subtree $s^*$ whose measure is: $\mu(s^*) \leq \mu(s_i)$ getting a new hyperpath tree $t^{j+1}_{X,y}$. Since $\mu$ is *GSUP* then, from Lemma 6.1, we know that this replacement is measure preserving, i.e., $\mu(t^{j+1}_{X,y}) \leq \mu(t^j_{X,y})$.

Let us consider the consequences of the global replacement of every subtree rooted at $i_2$ on the structure of the hyperpath tree. There are two possibilities, according to who is the winner of the tournament.

- the winner does contain the interval $[i_1 - i_2]$ ($s_1$ or any other subtree): in this case the second

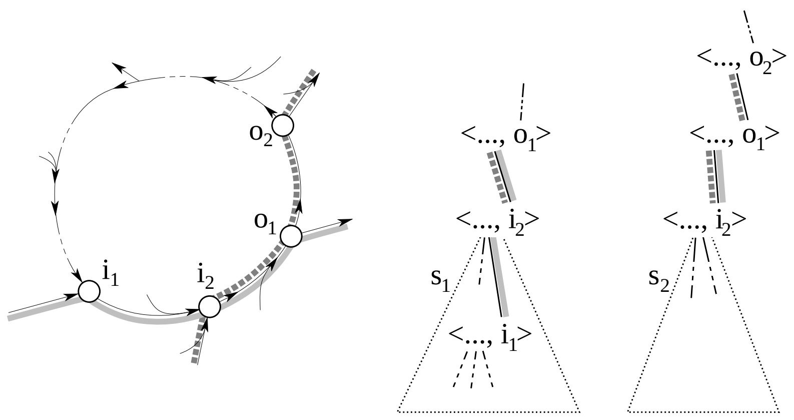


Figure 13: A cycle $C$ and two consecutive chains as fragments in a hyperpath tree.

subtree in Figure 13 will gain a (strictly) longer chain, compared to the previous stage, from the input node $i_1$ to the output node $o_2$;

- the winner does not contain the interval $[i_1 - i_2]$ ($s_2$ or any other subtree): in this case the interval $[i_1 - i_2]$ disappears within the whole hyperpath tree. Cycle $C$ is broken, since the hyperarcs in this interval disappears from the unfolded hyperpath as well.

□

The next theorem states a new characterization of Superior (and Inferior) Functions, based on the acyclicity of the optimal hyperpaths.

**Theorem 6.3** *Let $\mathcal{P}_{\min} = (\min, \mu)$ and $\mathcal{P}_{\max} = (\max, \mu)$ be respectively a minimization and a maximization problem, where $\mu$ is a Value Based Measure Function over directed hyperpaths.*

*a) $\mathcal{P}_{\min}$ is cycle invariant for any functional hypergraph if and only if $\mu$ is a Superior Function;*

*b) $\mathcal{P}_{\max}$ is cycle invariant for any functional hypergraph if and only if $\mu$ is an Inferior Function.*

**Proof.**
$(a)(\Longleftarrow)$ We assume that $\mu$ is a superior function then, for any hyperpath tree $t$ and any subhyperpath $s$ of $t$, $\mu(s) \leq \mu(t)$.

Let us suppose, by contradiction, that there exists a cyclic optimal hyperpath $t^*_{X,y}$. By Lemma 6.2, we may assume that there exists (also) an optimal hyperpath with N-MULT$(t^*_{X,y}) \geq 2$. For our purposes, this hyperpath has the structure shown in Figure 12: if $n$ is a node with N-MULT$_{t^*_{X,y}}(n) \geq 2$, then there are two occurrences of node $n$ as a target of a hyperarc within the same branch. Hence there are two nested hyperpath subtrees: $s_{X,n}$ and $s'_{X',n}$, where the first one has the root in the higher occurrence of node $n$, and $s'$ has the root in the lower occurrence in the same branch.

If we replace $s$ with its proper subtree $s'$, since $\mu(s') \leq \mu(s)$, then the resulting hyperpath tree $\tilde{t}^*_{X,y}$ is such that:

- $\mu(\tilde{t}^*) \leq \mu(t)$
- N-MULT$_{\tilde{t}^*_{X,y}}(n)$ = N-MULT$_{t^*_{X,y}}(n) - 1$; furthermore, since $s' \subset s$, this replacement cannot increase the node multiplicity of any node.

Again, the new hyperpath tree is either acyclic, or we can repeat this procedure at most $M$ times (where $M$ is the sum of node multiplicity in the original optimal hyperpath tree $t^*_{X,y}$), until the resulting hyperpath is acyclic.

$(a)(\Longrightarrow)$ We assume that $\mathcal{P}_{\min} = (\min, \mu)$ is cycle invariant for any functional hypergraph. As a contradiction, let us assume that function $\mu : \Re^k \rightarrow \Re$ is not a superior function: in this case there exists a $k$-tuple $(x_1, x_2, \dots, x_k) \in D^k$ such that $\mu(x_1, x_2, \dots, x_i, \dots, x_k) < x_i$ for some $i$. Let us consider the hypergraph $\mathcal{H}_F = (N, H)$, where: $N = \{s_1, s_2, \dots, s_k, n_1, n_2, \dots, n_k\}$ and

$H = \{(s_j, n_j) \text{ for } j = 1, \ldots, k\} \cup (n_1\, n_2\, \ldots\, n_k, n_i)$. Let us assume that measure function $\mu$ is the traversal cost, i.e.:

$$\mu(t_{X,y}) = \left\{ \begin{array}{cl} 0 & \text{if } t_{X,y} = \emptyset, \text{ i.e., if } y \in X \\ w_{\langle Z,y\rangle} + \sum_{z_i \in Z}\{\mu(t_{X,z_i})\} & \text{if } t_{X,y} \text{ has root } \langle Z, y\rangle \text{ with subtrees } t_{X,z_1}, t_{X,z_2}, \ldots, t_{X,z_k} \end{array} \right.$$

The weights are defined as follows: $w_{\langle s_j,n_j\rangle} = x_j$ (for $j = 1, \ldots, k$), and $w_{\langle n_1\, n_2\, \ldots\, n_k, n_i\rangle} = \epsilon$, where $\epsilon$ is a suitable positive quantity. This hypergraph is drawn in Figure 14.

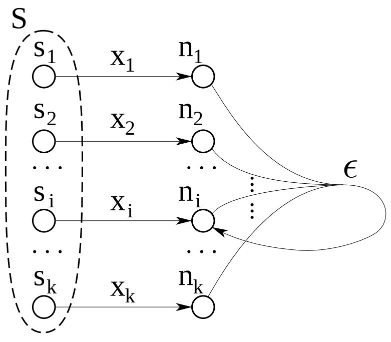


Figure 14: For any non-*SUP* function $\mu$, there exists a minimization problem having a cyclic optimal hyperpath.

Let us consider the minimization problem $\mathcal{P}_{\min} = (\min, \mu)$, which consists in finding an optimal hyperpath from the source set $S = \{s_1, s_2, \ldots, s_k\}$ to the target node $n_i$.

The unique acyclic hyperpath in $\mathcal{H}_F$ from $S$ to $n_i$ is $t_{S,n_i} = \{(s_i, n_i)\}$, whose measure is $\mu(t_{S,n_i}) = x_i$. But the cyclic hyperpath tree $t'_{S,n_i}$, with root $(n_1\, n_2\, \ldots\, n_k, n_i)$ and children $(s_j, n_j)$, for $j = 1, \ldots, k$, has a measure: $\mu(t'_{S,n_i}) = \epsilon + \mu(x_1, x_2, \ldots, x_k)$. The positive quantity $\epsilon$ can be chosen so that $\mu(t'_{S,n_i}) < \mu(t_{S,n_i})$.

Therefore, since there exists a functional hypergraph where a cyclic hyperpath is better, under $(\min, \mu)$ criterion, than any existing acyclic hyperpath, then $\mu$ is not cycle invariant.

Case $(b)$ can be proved similarly. □

Lemma 6.2 states that, for any optimal hyperpath and any *GSUP* and *GINF* measure function, we have an equivalent hyperpath tree with one of two possible canonical structures, according whether its node multiplicity is equal to 1, or larger.

In order to determine the properties of optimal optimal trees in the more specific case of a *WSUP* or *WINF* function, we need more details on the structure of optimal cyclic hyperpaths: in this case we find useful the notion of $k$-relevance, i.e., the maximum number of variables that are meaningful while searching for a minimum or maximum solution.

We remark that, even in case of a 0R-*GSUP* function, optimal hyperpaths might be cyclic, i.e., there might exist no optimal acyclic hyperpath, although the value of a 0R-*GSUP* function may be a constant, possibly depending on the hyperarc. As a typical example, we have the *last* function, which is equal to the weight of the last hyperarc. Let us consider an instance of a directed graph where we are interested in finding a path with minimum *last* weight from a source node $s$ to a given taget node $t$. The optimal path will use the last arc $(x_i, t)$ with minimum weight such that $x_i$ is reachable from $s$: reaching $x_i$ from $s$ may require to pass through $t$.

The *gap* function (Definition 4.3), defined as "sum of the min" is *WSUP*: therefore, by Theorem 6.4, the corresponding minimization problem is at most 1-*CYCLE-CONV*. The instance shown in Figure 15 and the comments in the caption prove that this bound is tight.

Let us consider any hyperpath tree $t_{X,y}$, and a 1-relevant measure function $\mu$. As discussed in Section 5, the measure of this tree, $\mu(t_{X,y})$, depends only on a unique relevant argument, say $z_r$. If we focus on the root, the hyperarc $\langle z_1\, \ldots\, z_r\, \ldots\, z_k, y\rangle$, the value depends on the subhyperpath $s_{X,z_r}$, provided that the other nodes are reachable, in order to build up a complete hyperpath,

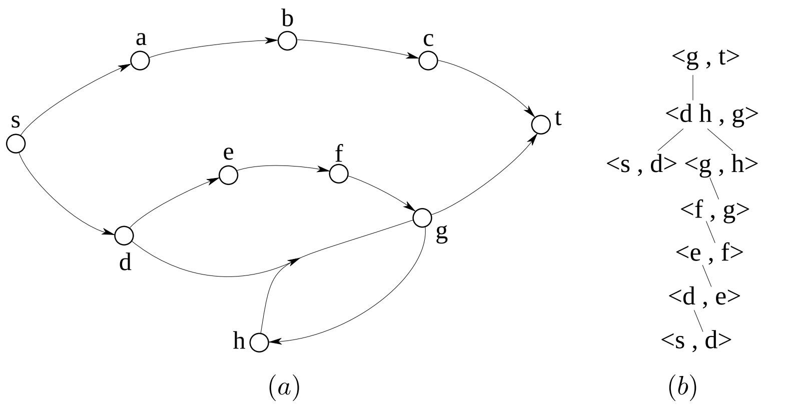


Figure 15: (*a*) A directed hypergraph. (*b*) Unfolded structure of a hyperpath from $s$ to $t$ with $gap = 3$: this is cyclic, since node $g$ must be entered twice, namely, N-MULT$(g) = 2$. No acyclic hyperpath from $s$ to $t$ has equal or smaller *gap*.

as shown in the previous example of *gap*. Since also the value of $\mu(s_{X,z_r})$ is 1-relevant, the same argument apply, and it comes out that the only relevant hyperarcs, in order to compute $\mu(t_{X,y})$ are those on a single branch of the hyperpath tree. In Figure 16 we show the structure of such a hyperpath tree, where the symbol $u$ represent a generic irrelevant variable.

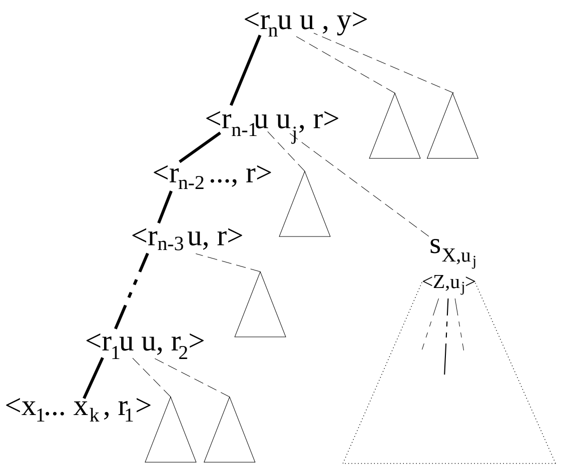


Figure 16: The canonical structure of an optimal hyperpath in case of a 1R-*GSUP* measure function: at most one argument (bold arcs) can be relevant for computing the measure function on each node.

For any *GSUP* function with irrelevant variables, the properties of the structure of an optimal tree $t_{X,y}$ can be generalized. The *relevant tree* is the portion of $t_{X,y}$ obtained by pruning all the *irrelevant subtrees*. Each irrelevant subtree is a subhyperpath $s_{X,u_j}$ from source $X$ to a node $x_j$, hence its root is a hyperarc $(Z, u_j)$ appended to the relevant tree, in particular to an instance of node $u_j$ in the source of a hyperarc $(\ldots\ u\ \ldots, n')$.

In case of a 1-relevant measure function, the relevant tree is actually a path, such as the one represented in Figure 16. In case of a 2-relevant function, the relevant tree is a binary tree, and so on, for any $k \geq 0$. Furthermore for any *GSUP* function $\mu$, and any $\mu$-minimal hyperpath $t_{S,t}$, the structure of the irrelevant subtrees can be manipulated so that eventually none of them contains any cycles: this is due to the monotonicity required by *GSUP*. The role of the irrelevant subhyperpaths is to propagate the reachability from the root to the relevant nodes, regardless their measure. Still there could be cycles on the relevant subtree.

In case of a *WSUP* function $\mu$, we can apply internal subtree replacements to a $\mu$-minimal hyperpath so that it eventually does not contain any cycle on the relevant subhyperpaths. If this would not be the case, we can remove cycles as shown in the proof of Lemma 6.2. In other words, if we are given a *WSUP* function, the structure of the relevant portion of a minimal hyperpath is the one of a *SUP* hyperpath tree, i.e., acyclic. Instead, a single loop may be required on the irrelevant branches, as seen in the case of the *gap* (Figure 15). Note that, in this case, the node $n$ having node N-MULT$(n) = 2$ must have the first occurrence in some irrelevant subtree (a first hyperpath, in order to reach a "useful" node $n$), and the second one in some highest relevant node (entering in the same node with minimum cost).

The following Lemma proves the property of an optimal hyperpaths for *WSUP* functions.

**Theorem 6.4** *Let* $\mathcal{P}_{\min} = (\min, \mu)$ *and* $\mathcal{P}_{\max} = (\max, \mu)$ *be respectively a minimization and a maximization problem, where* $\mu$ *is a value based measure function over directed hyperpaths.*

*a) if* $\mu$ *is a Weakly Superior Function, then* $\mathcal{P}_{\min}$ *is* 1*-Cycle Convergent for any functional hypergraph;*

*b) if* $\mu$ *is a Weakly Inferior Function, then* $\mathcal{P}_{\max}$ *is* 1*-Cycle Convergent for any functional hypergraph.*

**Proof.** $(a)$ Let $t_{X,y}$ be an optimal hyperpath. Due to Lemma 6.2 we may assume, without loss of generality, that N-MULT$(t_{X,y}) > 2$. Since the measure function is *WSUP*, we can split the set of nodes $Z$ into two subsets of *relevant* $(Z_R)$ and *irrelevant* $(Z_I)$ nodes.

For each irrelevant node $z_i$, the value of $\mu(t_{X,y})$ does not depend on the value of $\mu(t_{X,z_i})$, according Definition 5.2. Hence we can replace each generic hyperpath subtree with one having N-MULT$(t^a_{X,z_i}) = 1$, as in the proof of Lemma 6.2, without increasing the value of the measure of the current hyperpath tree. After eliminating cycles on all the irrelevant subtrees, we can remove all the cycles on the relevant tree, that can be made acyclic, due to Theorem 6.3. Since the relevant tree and all the irrelevant subtrees, each rooted in a node of the relevant tree, are acyclic, the node multiplicity of the whole tree cannot be larger than 2.

Case $(b)$ can be proved similarly. $\square$

In case of a $k$-*CYCLE-CONV* function $\mu$, cycles may improve the value of a measure function, up to a maximum of $k$ nested cycles. The structure of a $\mu$-optimal hyperpath can be subject to subtree replacements up to its canonical structure:

- a relevant tree, with node multiplicity at most $k$;
- irrelevant subtrees with no cycles.

Therefore, the resulting node multiplicity is at most $k + 1$.

One might ask whether the classes of $k$-*CYCLE-CONV* optimization problems do not collapse for $k \geq 2$. The following class of problems, provides a negative answer to this question.

We will use the simple example ok $k$-*gems*, introduced in Section 5. We want to maximize the number of gems, up to a maximum of $k$, collecting them from some special hyperarcs, hence the weight function is binary (a weight 1 is for a hyperarc with a gem). Equivalently, since we put a gem in one pocket, we might consider the dual problem of minimizing the number of empty *pockets* at the end of our hypertrip, having $k$ pockets.

**Theorem 6.5** *The problem* $\mathcal{P} = (\max, k\text{-}gems) \equiv (\min, k\text{-}pockets)$ *is* $(k-1)$*-CYCLE-CONV and it is not* $(k-2)$*-CYCLE-CONV, for* $k = 1 = 1, 2, \ldots, k$*.*

**Proof.** Let us consider a source node $s$ and a target node $t$ in a directed graph containing a single gem on a single arc $(a, b)$, which is part of a cycle and such that node $a$ is reachable from the source, and the target is reachable from $b$. Then, any $(\max, k\text{-}gems)$-optimal path will pass at least $k$ times through arc $(a, b)$; of course this bound is tight. $\square$

There are common problems which are value based, but *CYCLE-UNB*. Examples of hyperpaths with an unbounded structure may occur by reversing the optimization criterion (e.g., finding hyperpaths with maximum *rank*).

**Theorem 6.6** *Let $\mathcal{P} = (\Phi, \mu)$ be an optimization problem on hypergraphs. The following properties hold:*

*a) if* $\Phi = \max$ *and $\mu$ is a SSUP function then $\mathcal{P} = (\Phi, \mu)$ is CYCLE-UNB;*

*b) if* $\Phi = \min$ *and $\mu$ is a SINF function then $\mathcal{P} = (\Phi, \mu)$ is CYCLE-UNB.*

Actually, the result stated in Theorem 6.6, can be extended to all situations of (max, *SUP*) problems in presence of positive cycles, or (min, *INF*) problems with negative cycles.

An interesting case is the measure function *average-depth* (Definition 4.4). This is value-based but it is neither *GSUP*, nor *GINF*. Let us consider the hypergraph shown in Figure 17. The optimal

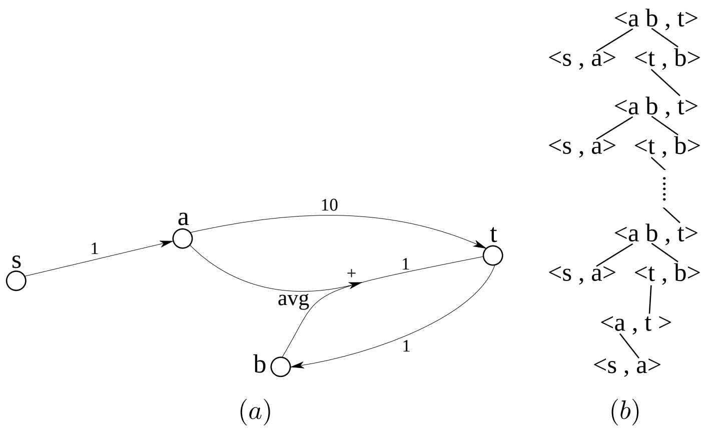


Figure 17: Building a hyperpath from $s$ to $t$ with minimum *average-depth*: $(a)$ folded and $(b)$ unfolded structure.

hyperpath is cyclic, hence its unfolded structure is unbounded. Nevertheless, the optimal solution of this instance can be found by an elementary system of linear equations:

$$\begin{cases} avgd(s) = 0 \quad \text{(by extended reflexivity)} \\ avgd(a) = \text{avg}\{avgd(s)\} + 1 = 1 \\ avgd(b) = \text{avg}\{avgd(t)\} + w_{\langle t,b\rangle} = avgd(t) + 1 \\ avgd(t) = \text{avg}\{avgd(\text{a}) + avgd(\text{b})\} + w_{\langle ab,t\rangle} = \dfrac{1 + avgd(b)}{2} + 1 \end{cases}$$

This leads to: $avgd(s) = 0$, $avgd(a) = 1$, $avgd(b) = 5$, $avgd(t) = 4$. We remark that it is not possible to find the optimal solution of this problem by an algorithm that require to traverse the returned hyperpaths.

By using the previous results, we can characterize in a general and unified framework several optimization problems on hypergraphs. We remark that the properties of *CY-INV* and 1-*CYCLE-CONV* (stated by Theorem 6.4) are upper bounds, whilst the property of *CYCLE-UNB* (stated by Theorem 6.6) is a lower bound; of course, these properties may be not tight on the specific instance.

The *rank* (Definition 4.2), syntactically very similar to the *gap*, is defined as "sum of the max", and then it is *SUP*: by Theorem 6.4, the resulting minimization problem is *CY-INV*, while the maximization problem is *CYCLE-UNB* (by Theorem 6.6), as one would expect.

A very special example is the *closure* problem, regarded as an optimization problem: actually this is the decision problem of finding whether, given a set on nodes $X$ and a node $y$, there exists a hyperpath from $X$, to $y$. In this case we can assume uniform hyperarcs weights, and the resulting measure function has the same value on any existing hyperpath. Since the value of the function is equal to its arguments, *closure* is both *SUP* and *INF*, and hence *CY-INV*.

Most of the results discussed above are summarized in Table 4.

| measure function $\mu$ | $f(w,\psi)$ | $\psi(\mu_1,\ldots,\mu_k)$ | resulting properties | MIN problem | MAX problem |
|---|---|---|---|---|---|
| *rank* | $+$ *SUP* | max *SUP, WINF* | *SUP* | *CY-INV* | *CYCLE-UNB* |
| *gap* | $+$ *SUP* | min *WSUP,INF* | *WSUP* | 1-*CYCLE-CONV* | *CYCLE-UNB* |
| *average-depth* | $+$ *SUP* | avg *GSUP* | *GSUP* | *CYCLE-UNB* | *CYCLE-UNB* |
| *last* | $w$ *WSUP, WINF* | (constant) *WSUP, WINF* | *WSUP, WINF* | 1-*CYCLE-CONV* | 1-*CYCLE-CONV* |
| *traversal cost* | $+$ *SUP* | $\sum$ *SUP* | *SUP* | *CY-INV* | *CYCLE-UNB* |
| *P-Prod*$[1,+\infty]$ | $\times$ *INF* | $\Pi$ *INF* | *SUP* | *CY-INV* | *CYCLE-UNB* |
| *P-Prod*$[0,1]$ | $\times$ *INF* | $\Pi$ *INF* | *INF* | *CYCLE-UNB* | *CY-INV* |
| *bottleneck* | min *WSUP,INF* | min *WSUP,INF* | *WSUP,INF* | 1-*CYCLE-CONV* | *CY-INV* |
| *threshold* | max *SUP, WINF* | max *SUP, WINF* | *SUP, WINF* | *CY-INV* | 1-*CYCLE-CONV* |
| *closure* | $=$ *SUP,INF* | $=$ *SUP,INF* | *SUP,INF* | (any solution) *CY-INV* | |

Table 4: Characterization of measure functions on hypergraphs.

We conclude this section by representing in Figure 18 the main relationships between the classes of measure function and the properties of the corresponding optimal hyperpath in case of minimization problems. We recall that $(\min, k\text{-}pockets) = (\max, k\text{-}gems)$. An analogous drawing could be shown in case of maximization problems.

## 7 Representation of Optimal Hyperpaths

In this section we show that any optimal hyperpath has an equivalent one with a *canonical* structure. This is useful for several reasons:

- we will see that, for all *GSUP*/*GINF* function and any directed hypergraph, there exists an optimal canonical tree having a representation of linear size, i.e., of the same size of the unfolded hyperpath;
- reducing optimal hyperpaths to few canonical cases makes easier to understand the impact of measure functions and optimization criteria that originate them;
- by using canonical structures, it is easier both to design and to analyze algorithms for finding optimal hyperpaths, since it is sufficient to find one of these canonical hyperpaths; furthermore, these hyperpath trees are those suitable of a representation with minimal size.

In the previous sections we have used replacements in order to reduce as much as possible the node multiplicity of a hyperpath; in this section we apply subtree replacements in order to converge toward a corresponding *canonical tree*, and show how these can be represented in a compact form having linear size.

**Definition 7.1** *Given an optimization criterion $(\Phi, \mu)$, where $\Phi \in \{\min, \max\}$ and $\mu \in VBMF$, we say that a hyperpath tree $t_{S,t}$ is* canonical *if, for any any two subtrees $s_{S,z}$ and $s'_{S,z}$ with the same target node $z$:*

(*a*) *if $s$ and $s'$ are not subtree one another, then $s_{S,z} = s'_{S,z}$ (i.e., they are identical);*

(*b*) *if, w.l.o.g., $s'$ is a subtree of $s$, then $\mu(s_{S,z}) \prec \mu(s'_{S,z})$ (i.e., the deeper subtree is worse).*

Note that a canonical tree, even if it is optimal, can have an unbounded structure; as an example, the instance shown in Figure 17 is a canonical unbounded optimal tree. In the following we consider Cycle-Bounded hyperpath optimization problems.

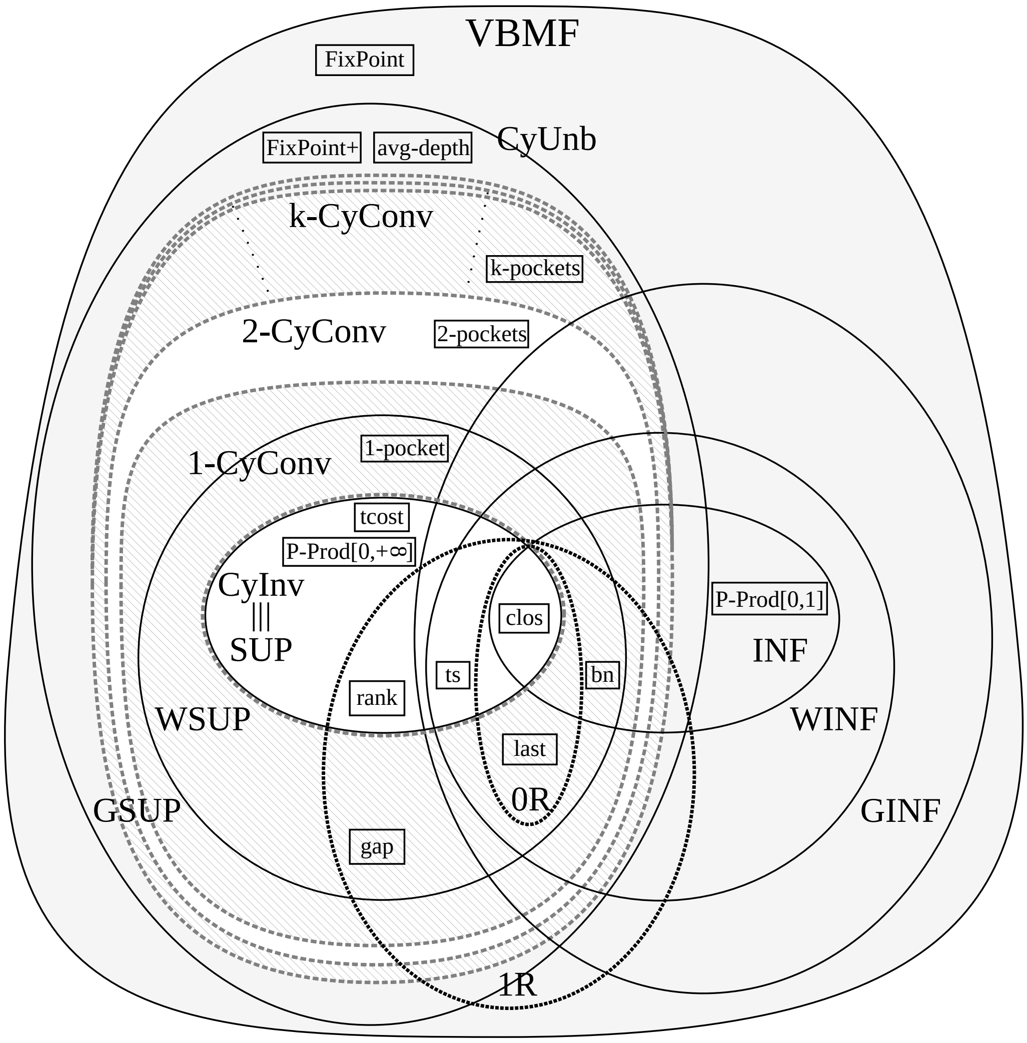


Figure 18: Relationships between the classes of measure function and the properties of the optimal hyperpaths in case of a minimization problem (*clos* = *closure*, *tcost* = *traversal cost*, *ts* = *threshold*, *bn* = *bottleneck*).

**Lemma 7.1** *Given a Cycle-Bounded hyperpath optimization problem, with criterion $(\Phi, \mu)$, where $\Phi \in \{\min, \max\}$ and $\mu \in GSUP \cup GINF$ and any functional hypergraph $\mathcal{H}_F$, for each optimal hyperpath tree $t^*_{S,t}$ there exists a canonical tree $t^{C*}_{S,t}$, and a sequence of replacements such that $t^*_{S,t} \rightsquigarrow t^{C*}_{S,t}$, and $\mu(t^{C*}_{S,t}) \preceq \mu(t^*_{S,t})$, i.e., the sequence of replacements is measure-preserving.*

**Proof.** Suppose we are given an arbitrary hyperpath tree $t^*_{S,t}$ which is not canonical, i.e., there are two subtrees which contradict Definition 7.1. We show that, starting from an arbitrary $t^*_{S,t}$, it is possible to transform it in a canonical tree $t^{C*}_{S,t}$ by means of measure-preserving subtree replacements. Following the definition of canonical tree, we assume that in $t^*_{S,t}$ there are two subtrees, $s_{S,z}$ and $s'_{S,z}$, with the same target node $z$ and, in the hypothesis that $t^*_{S,t}$ is not in a canonical form, we consider separately the two cases in Definition 7.1.

*Type $(a)$ replacements:* We assume that $s$ and $s'$ are not subtree one another, but $s_{S,z} \neq s'_{S,z}$ (i.e., they are not identical).

Since function $\mu$ is Value-Based, then it takes values from a totally ordered domain, hence one of the two subtrees, say $s'$, is not worse than the other one, i.e.: $\mu(s'_{S,z}) \preceq \mu(s_{S,z})$. This is a value preserving replacement and, according Lemma 6.1 if we replace any occurrence of $s$ in $t^*_{S,t}$ with a copy of $s'$, the new hyperpath tree $t'_{S,t}$ cannot be worse than the previous one (1). On the other side, this replacement decreases at least by one unit the number of pairs of subtrees conflicting with case $(a)$ in Definition 7.1, therefore we can apply it until case $(a)$ does not hold any more.

We remark that a single Type $(a)$ replacement can increase node multiplicity. Adopting the notation above, this situation arises when the new subtree $s'_{S,z}$ contains a node with target $y$ and, when $s'$ replaces a subtree $s_{S,z}$, the branch above the replacement point contains a node with target $y$ (see Figure 19). Note that, if this replacement introduces a conflict with constraint $(b)$, a successive Type-$(b)$ replacement will eliminate this new occurrence of node $y$.

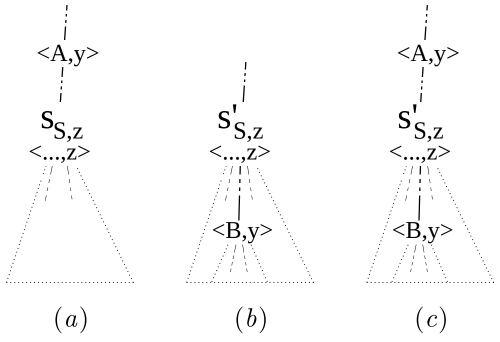


Figure 19: A Type $(a)$ subtree replacement increasing node multiplicity: Figure *(a)* shows the subtree $s_{S,z}$ to be replaced, *(b)* shows the replacing subtree $s'_{S,z}$; *(c)* shows the resulting tree.

*Type $(b)$ replacements:* We assume that $s'$ is a subtree of $s$, with $\mu(s'_{S,z}) \preceq \mu(s_{S,z})$.

By similar arguments as in case $(a)$ above, the replacement $s_{S,z} \to s'_{S,z}$ is measure-preserving, and reduce at least by one unit number of pairs of subtrees conflicting with case $(b)$. Of course, in this case node multiplicity cannot grow. □

We can now introduce the *representation* of a canonical tree by merging all the equivalent subtrees, i.e., by "sharing" the representation of all subhyperpaths wherever they are identical. Of course, if we are given a hyperpath tree $t$, its representation $R(t)$ is not a tree any more, since several nodes share their children. But an interesting feature is that any hyperarc $\langle X, z \rangle$ in a this representation will appear exactly once, and then this representation has the same size of the folded representation.

First of all, we show such an example of representation in Figure 20 in case of an acyclic hyperpath tree. The source nodes are highlighted by connecting them to a single dummy node at the bottom.

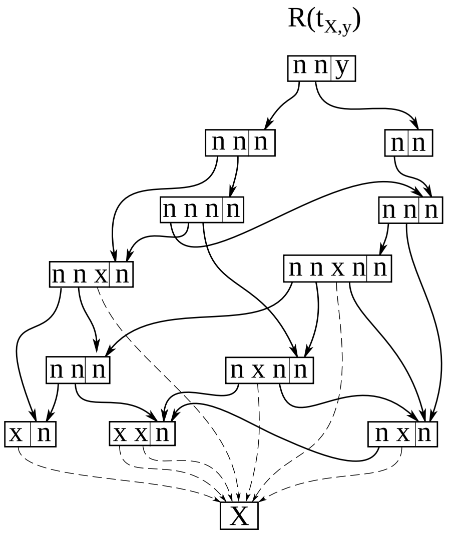


Figure 20: The representation of a canonical acyclic hyperpath tree.

As an additional example, in Figure 21 we show a linear size representation of the hyperpath tree of exponential size: the folded hyperpath and the hyperpath tree are shown in Figure 4.

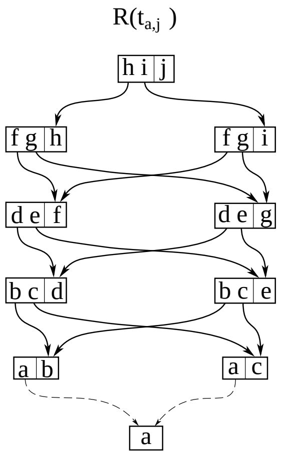


Figure 21: A linear size representation of a hyperpath tree of exponential size (shown in Figure 4).

In the following we consider this representation for the classes of measure functions in the *VBMF* hierarchy, and show the properties of the corresponding canonical representation within each interesting class of measure functions.

## *Problems in* (min, *GSUP*) $\cup$ (max, *GINF*)

In presence of cycles, while optimizing hyperpaths under *GSUP* or *GINF* measure functions, any $k$-Cycle Convergent optimization problem has, by definition, an optimal tree with node multiplicity at most $k+1$. In this case, a cyclic hyperpath tree contains at most $k$ times a node $n$ with the same target, and each time the subtree will has as a root one of the hyperarcs entering node $n$: these are N-INDEG$[n]$.

For a $k$-Cycle Convergent problem we have to represent optimal hyperpaths with node multiplicity $k+1$. In a canonical representation each hyperarc is represented exactly once, but each node may have up to $k+1$ possible connections to as many subtrees. An example is shown in Figure 22. Note that, for $k \to \infty$, this representation can be extended to unlimited hyperpath trees, provided

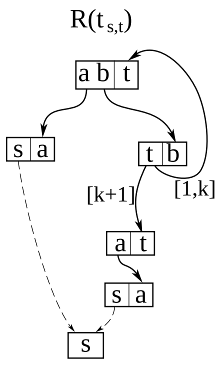


Figure 22: A linear size representation for the example in Figure 17.

that they have a "periodic" structure.

## *Problems in* (min, *WSUP*) ∪ (max, *WINF*)

We have proved that these optimization problems are 1-Cycle Convergent (Theorem 6.4). This means that any optimal hyperpath has an equivalent tree with node multiplicity at most 2.

In all these cases, the space is bounded by the size of the hypergraph, i.e., $O(|\mathcal{H}|)$ for an explicit representation Optimal Hyperpath tree. In Figure 23 we show the representation of a optimal canonical hyperpath tree.

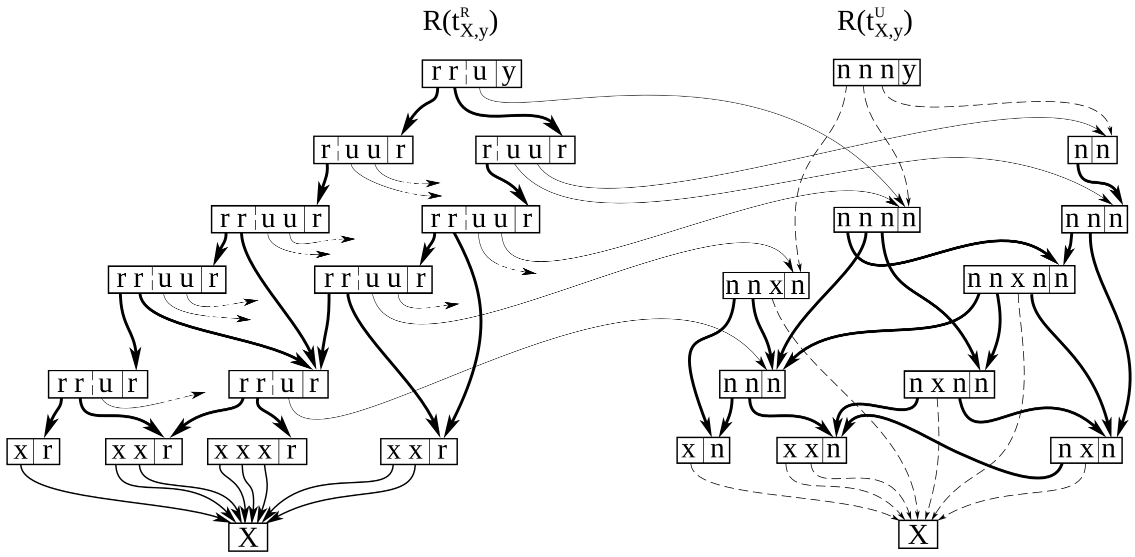


Figure 23: The representation of an optimal hyperpath tree in case of a problem in (min, *WSUP*) ∪ (max, *WINF*): the representation relevant hyperpath tree is on the left with bold arcs; the representation of the forest of irrelevant subtrees is on the right. We add up a dummy target node on the irrelevant forest.

## *Problems in* (min, *SUP*) ∪ (max, *INF*)

We have proved that this class of coincides with the Cycle Invariant max / min optimization problems (Theorem 6.3). Therefore, there exist always an optimal canonical tree where all the subtrees rooted at the same node are identical, and the previous data structure is sufficient to store an optimal tree for any of these problems.

In the following we summarize the properties that we have identified as essential in order to classify a measure functions $\mu(x_1, x_2, \ldots, x_k)$, and the consequent properties, under a certain optimization criterion (min or max), of a canonical optimal hyperpath tree $t^*_C$ and its representation $R(t^*_C)$:

- Multidimensional triangle inequality: $x_i \preceq \mu(x_1, x_2, \ldots, x_k)$, for $i = 1, 2, \ldots, k$;

  $t^*_C$ is always acyclic (meaning that its unfolded representation is acyclic);

  $R(t^*_C)$ has a linear size (even though $t^*_C$ has an exponential size).

- Weak multidimensional triangle inequality: for $i = 1, 2, \ldots, k$, each argument $x_i$ of the measure function can be:

  - *relevant*, if it complies the multidimensional triangle inequality;
  - *irrelevant*, if it can be unlimitedly worsen up to the superior/inferior of the considered domain (opposite to the optimization criterion) without affecting the value of $\mu(\ldots)$.

  $t^*_C$ is 1-Cycle-Convergent; a relevant acyclic tree, and a collection of irrelevant acyclic subtrees;

  $R(t^*_C)$ can be organized as follows: the relevant tree can be stored by means of a rooted acyclic graph, and the irrelevant subtrees can be stored as a forest of rooted dags.

- Monotonicity: for $i = 1, 2, \ldots, k$, if $x'_i \preceq x$, then $\mu(x_1, \ldots, x'_i, \ldots, x_k) \preceq \mu(x_1, \ldots, x_i, \ldots, x_k)$, i.e., if we "improve" (according the optimization criterion) one of the arguments, the value of the function cannot worsen.

  $t^*_C$ can be split as follows: a (possibly cyclic) relevant tree and, as above, a forest of acyclic irrelevant subtrees;

  $R(t^*_C)$ has a linear size (even though $t^*_C$ has unbounded size).

- Value-Based: $\mu$ has values in a totally ordered domain.

Therefore, we can summarize the properties of of the main classes of functions.

| Class of function | Value-Based | monotonicity | weak multidim. triangle ineq. | multidim. triangle ineq. |
|---|---|---|---|---|
| *SUP* [Knu77], *INF* | $\surd$ | $\surd$ | $\surd$ | $\surd$ |
| *WSUP* [RR96], *WINF* | $\surd$ | $\surd$ | $\surd$ | |
| *GSUP*, *GINF* | $\surd$ | $\surd$ | | |
| *VB* | $\surd$ | | | |

## 8 Algorithmic Approaches to Hyperpath Problems

A number of approach for finding hyperpaths have been proposed in the literature, sometimes with small variants, and sometimes formulated in specific domains, not fully aware of similar research efforts in other domains.

In this section we describe some of the basic algorithmic strategies for finding hyperpaths. Then, the effectiveness of each algorithm is clearly stated with respect to the classification of hyperpath optimization problems provided in the previous sections, based on (i) the features of the adopted metrics and (ii) the resulting optimal hyperpath.

One of these strategies (*ISBP*) is detailed here for the first time. These proposed algorithm have a simple structure, and are suitable of a simple and practical implementation. Furthermore, we could devise a unifying algorithmic pattern encompassing all of them: therefore their behaviour can be compared and similarities and differences are highlighted.

The proposed algorithmic pattern is suitable of further modifications in order to tailor it for specific optimization criteria.

In the computational costs we do not take into account the cost of computing the measure function on each hyperarc $F_{\langle X,y \rangle}(x_1, x_2, \ldots, x_k)$ when all its arguments are available. In many practical cases this is not relevant; in a general case, this computational cost has to be considered any time that the measure of a hyperpath has to be computed. This computational cost has a different nature from the kind of problem we study in this paper. Anyway, finding hyperpaths which minimize the number of times a function has to be computed on a hyperarc - in turn - might be formulated as another optimization problem.

In particular we address the Single-Source problem. We are given:

- a directed functional hypergraph $\mathcal{H}_F = \langle N, \mathcal{F} \rangle$;
- a specific source set $S \subseteq N$;
- a measure function $\mu = \langle f, \psi, \mu_0, \mu_\infty \rangle$;
- an optimization criterion $\Psi$.

We are interested in finding the $\mu$-optimal (let us say, minimal) hyperpaths from a source set $S \subseteq N$ to any other node in $N$, i.e., a *Single-Source Optimal Hyperpath* problem. Given a query node $z$, we want to know both the measure of an optimal hyperpath from $S$ to $z$, and the listing of all the hyperarcs in such hyperpath.

Given the optimization problem $\mathcal{P} = (\min, \mu)$, we will denote as "distance" of a node $y$ from a given source $S$ the value $d[y] = \mu(t^*_{S,y})$, i.e., the measure of an optimal hyperpath for problem $\mathcal{P}$.

We will make use of the following data structures:

```
DATA STRUCTURES:
INPUT:
   $N$ : set of nodes
   $H$ : set of hyperarcs
   $S$ : set of (source) nodes
OUTPUT:
   $d[1..n]$ : array of integers (measure of the optimal hyperpath from the source)
   $ChildRoot[1..n]$ : array of pointers to hyperarcs (representation of the hyperpath tree)
AUXILIARY STRUCTURES:
   $Q$ : queue (the kind of queue changes according the algorithm)
```

We represent hypergraphs by means of adjacency lists, with size $|\mathcal{H}| = \Theta(|N|+|H|+a)$, where $a$ is the source area. For the computational costs, we refer to the quantities introduced in Definition 4.1; in particular, $|d| = |ChildRoot| = |N|$. For uniform notation, for each node $u \in N$, the quantity $d[u]$ stores the measure of the best known hyperpath from source $S$ to node $u$, as well as the final (optimal) hyperpath. If there exists no hyperpath from $S$ to $u$, then we will have $d[u] = \mu_\infty$; this is the typical initialization value for all the algorithms that we present here.

Each node of a hyperpath tree $t_{S,t}$ is a hyperarc $(x_1\, x_2\, \ldots\, x_k, y)$. Any node $x_i$ has a pointer, $ChildRoot[x_i]$, to the root of a subhyperpath tree $t_{S,x_i}$ of $t_{S,t}$. For each source node $s_i \in S$, the value of $ChildRoot[s_i]$ will be *null*, in coherence with the fact that these nodes are the possible leaves of the hyperpath tree.

We will use the arrays $d[...]$ and $ChildRoot[1..n]$ even to store the measure and, respectively, the root, of the best known hyperpath from the source.

The most general data structure for a $k$-Cycle Convergent optimization problem requires that $ChildRoot[1..n, 1..k+1]$ is a two dimensional array of hyperarcs: for each node $x_i \in N$, the $i$-th row, the array $ChildRoot_i[1..k+1]$ stores the $k+1$ pointers to the children.

In order to make the pseudocode more readable and when no confusion may arise, the notation $\mu(X, y)$ denotes the measure of the hyperpath from the source $S$ to node $y$ using $(X, y)$ as its last hyperarc. In this case, the best known measures will be used for the subhyperpaths, hence: $\mu(X, y) \equiv \mu_{\langle X,y \rangle}(d[x_1], d[x_2], \ldots, d[x_k])$.

A hyperarc $(x_1\, x_2\, \ldots\, x_k, y)$ within a hyperpath tree has $k$ *ChildRoot* pointers, one leaving each $x_i$; furthermore all the hyperarcs sharing a node $x_i$ in their source sets, must share the corresponding *ChildRoot*. Note that, by representing a hyperpath tree with array *ChildRoot* of pointers with orientation from parent to child - the reverse of what one may expect, we apparently impose a restriction on the possible hyperpath trees that can be represented.

We have seen in Section 7 that this choice is not a restriction; on the other side any optimal hyperpath has an equivalent canonical one has a representation whose size is bounded by the size of the folded hyperpath.

For the case of a *WSUP* measure function, we have to arrange a data structure where each node appears in two different hyperpath subtrees. In Figure 23 we have shown an example with a representation of such a hyperpath tree. This requires to store up to two different pointers for each node. Namely, each reachable node $\xi$ appears once in the relevant tree, and once in the irrelevant tree (respectively, on the left and on the right in that Figure), in both cases as root of the representation of a subtree from source $S$ to node $\xi$. If another node has $\xi$ as child, it will point to the first root, or to the second root, according whether $\xi$ is a relevant, or irrelevant child. In particular, we use the array *ChildRoot* for the relevant pointer, and *IrrelChildRoot* for storing (within another single tree) all the irrelevant subtrees.

## *Visiting Hypergraphs*

First of all, we show a basic algorithm to build up a *Hyperpath Tree* by a simple visit of a directed hypergraph, i.e., without any optimization target beside being acyclic and spanning over the *closure* of the source set $X$. This simple algorithm, that is described in a number of sources and context (see, e.g., [AIN90, GLNP92]), is interesting since it allows us to introduce our algorithmic pattern and to fix the basics of the optimization algorithms shown in this section. We will see how - with small variations - it can be adapted to solve other simple problems on directed hypergraphs.

```
Algorithm Visit [$\mathcal{H}$ : any hypergraph]
/* visiting a hypergraph from a given source set $S$ */
Q is a SIMPLE QUEUE, $\mu_0 = 0$, $\mu_\infty = 1$
1. INITIALIZATION
      for each $n \in N$:  set $d[n] = \mu_\infty$; set $ChildRoot[n] = null$;
      for each $s \in S$:  enqueue $s$ in $Q$; set $d[x] = \mu_0$;
2. MAIN LOOP:
      while $Q$ is not empty:
3. VISIT node $x$ chosen from $Q$ according a given EXTRACTION POLICY
         extract any $x$ from $Q$;
4. for each hyperarc $(X, y) \in fstar[x]$ complying a SCAN POLICY
         for each hyperarc $(X, y) \in fstar(x)$ s.t. all nodes in $X$ have been visited:
5. IF TEST$(y)$ succeed, update data structures accordingly
            if $\mu(X, y) \prec d[y]$
               set $ChildRoot[y]$ to $(X, y)$;
               set $d[x] = \mu(X, y)$;
6. possibly ENQUEUE node $y$ in $Q$
            if$(X, y)$ is the first scanned hyperarc entering node $y$:
                 insert $y$ in $Q$;
```

The crucial points of this algorithmic pattern are summarized below in the present case of Algorithm *Visit*:

- extraction policy (step 3): type of queue and the notion of priority. In the case of *Visit*, any node is good, i.e., $Q$ is a simple queue;
- scan policy (step 4): under what conditions a hyperarc $(X, y)$ is scanned. In the case of Algorithm *Visit*, $x$ must be the last node in the source set $X$ to be visited; a simple counter on each hyperarc, with initial value $|X|$ and decremented for any visited node in $X$, can be used to test this condition in constant time;
- test on node $y$ (step 5): under what conditions a node $y$ is updated, since hyperarc $(X, y)$ is the root of a possible hyperpath tree $t_{S,y}$; in general node $y$ can be already enqueued and this would change its priority: this is not the case for Algorithm *Visit*, where a simple queue is used.

- enqueue policy (step 6): under what conditions a node $y$ is enqueued in $Q$; In the case of Algorithm *Visit*, the first scanned arc entering a node $y$, will imply its insertion in $Q$.

From another perspective, we can consider the lifecycle of the nodes in the hypergraph determined by the pattern above:

A. when a hyperarc $(X, y)$ entering node $\xi$ (denoted as "$y$") is *scanned* in step 4, the node is *tested* in step 5;

B. when a test on $\xi$ (as "$y$") in step 5 succeeds, then $\xi$ is *updated*, i.e., its new distance and parent is updated; if this is the first successful test, the node is *enqueued* in $Q$ in step 6 only if the scanned hyperarc is the first one to enter $\xi$; at this point $\xi$ becomes *reachable*; note that source nodes are reachable by an empty hyperpath, hence they are enqueued before entering the main loop;

C. when any further hyperarc $(X', y)$ entering $y \equiv \xi$ is scanned and the test on $y$ succeeds, node $\xi$ (as "$y$") is updated while it is in $Q$: in this case hyperarc $(X', y)$ would be the root of another possible hyperpath tree $t'_{S,\xi}$; in Algorithm *Visit* there is no such update;

D. when node $\xi$ (as "$x$") is dequeued from $Q$ (step 3), it is *visited*: all the hyperarcs in *fstar*$[x]$ will be possibly scanned (step 4) in order to propagate possible changes to adjacent nodes.

Due to the enqueue policy (item B above, and step 6 in the pseudocode), in Algorithm *Visit*, a node can be enqueued only once. This is not always the case for the following optimization algorithms.

When Algorithm *Visit* exits, if a node $\xi$ has not been enqueued, and then not visited, then it is not reachable from source $S$.

Algorithm *Visit* requires linear time in the size of the visited hypergraph $\mathcal{H} = (N, H)$. In fact, each node is visited (dequeued) at most once, and each hyperarc $(X, y)$ is scanned at most once, as soon as the last $x_i$ in the source $X$ has been visited. The only step that is not performed in linear time is step 4, that has a cost $|fstar(x)|$ But we have that $\sum_{x \in N}(|fstar(x)|) = \sum_{\langle X,y\rangle \in H}(|X|) = a$. Since we have $|N|$ nodes, $|H|$ hyperarcs, and source size $a$, the total time complexity is $O(|N| + |H| + a) = O(|\mathcal{H}|)$.

We remark that, in analogy with the case of directed graphs (a special case of directed hypergraphs), if $Q$ is managed as a FIFO queue, Algorithm *Visit* generates a Breadth-First Tree of the hypergraph, whilst if $Q$ is managed as a LIFO queue, we get a Depth-First Tree.

A small variation of Algorithms *Visit* may be used for an *Acyclicity Test* of the hypergraph - still requiring linear time. Of course this is very similar to analogous proposals, and extends the classical algorithm for directed graphs proposed in Knuth [Knu73]. If we change a single word and then the ENQUEUE POLICY in step 6:

- `if` $(X, y)$ `is the` *last* `scanned hyperarc entering node` $y$

the algorithm finds whether there are cycles reachable from the given source $S$. To find all cycles, it is sufficient to insert in the hypergraph a dummy node $s$ connected to all the other nodes $x \in N$ by an arc $(s, x)$, and use Algorithm *Visit* (from source $s$), with the variation above. Note that, if the hypergraph is cyclic, no node in a cycle can be enqueued, and this Algorithm exits without visiting all the nodes. If the hypergraph is acyclic, then this algorithm generates a *Topological Sort* of the nodes of the hypergraph: this is the sequence of dequeued nodes.

We will show that the optimization algorithms that we discuss below are a sort of variations on this theme, that is, they can be accommodated within this pattern, or are a sequence of steps, each being reducible to this pattern. In most cases their differences are essentially due to the policies for managing the queue, originally devised by Dijkstra.

### *SBP: Sort By Priority*

An algorithm for finding shortest path in a graph, based on a priority queue was originally proposed by Dijkstra [Dij59]. The first solution for the Single Source Optimal Hyperpath problem can be adapted from Knuth [Knu77], originally aimed at solving the so called *context free grammar problem.* He modified and extended the original algorithm by Dijkstra for *Superior CF Grammar.* This solution was "revisited" by Ramalingam and Reps [RR96], and explicitly formulated for directed hypergraphs. As a modification from the original formulation by Knuth, these authors propose to remove "useless" nonterminal symbols and productions. In our terminology, these are unreachable nodes, and hyperarcs with an unreachable node in the source. Hence they suggest of applying the algorithm above after pruning out these nodes and incident hyperarcs. Hence, this algorithm can be used to find the optimal hyperpaths from a single *source node* to all the other nodes in a weighted hypergraph $\mathcal{H} = \langle N, H; w\rangle$.

The algorithm, analogously to the original formulation by Dijkstra, uses a priority queue whose extraction policy is the minimum known "distance". Hence the nodes are visited in nonincreasing priority from the source (in this case, the "distance"): for this reason we denote this approach as *Sort By Priority (SBP)*. A generic node $y$ is enqueued when the first hyperarc entering $y$ is scanned, completing a first hyperpath from source $S$ to $y$. The priority of $y$ in the queue may decrease if further scanned hyperarcs provide better connections from the source. When a node is dequeued and visited, its distance from the source is computed. This algorithm requires $O(|H| \log |N| + |\mathcal{H}|)$ worst case time, that can be reduced to $O(|N| \log |N| + |\mathcal{H}|)$ by using Fibonacci heaps [FT87] for the implementation of priority queues.

We provide the schema of Algorithm *SBS* `Sort_By_Priority`, as a free interpretation of the algorithm described above, forced within our pattern, and generalized to decreasing functions.

```
Algorithm SBP (Sort By Priority)
/* rephrasing contributions and ideas by from [Dij59, Knu77, RR96] */
Q is a PRIORITY QUEUE, $\mu_0 = 0$, $\mu_\infty = \infty$
1. INITIALIZATION
      for each $n \in N$:  set $d[n] = \mu_\infty$; set $ChildRoot[n] = null$;
      for each $s \in S$:  enqueue $s$ in $Q$; set $d[x] = \mu_0$;
2. MAIN LOOP:
      while $Q$ is not empty:
3. VISIT node $x$ chosen from $Q$ according a given EXTRACTION POLICY
         extract $x$ from $Q$ having best priority $d[x]$;
4. for each hyperarc $(X, y) \in fstar[x]$ complying a SCAN POLICY
         for each hyperarc $(X, y) \in fstar(x)$ s.t. all nodes in $X$ have been visited:
5. IF TEST($y$) succeed, update data structures accordingly
            if $\mu(X, y) \prec d[y]$
               set $ChildRoot[y]$ to $(X, y)$;
               set $d[x] = \mu(X, y)$;
6. possibly ENQUEUE node $y$ in $Q$
               if $y \notin Q$
                  insert $y$ in $Q$;
```

We have presented this algorithm in a quite generic form. If the interpretation is as follows:

- "*best*" in steps 3 and 5 means "minimum"
- "$\prec$" in step 5 means "*strictly less than*"

we can use this algorithm for the minimization of a superior measure function, as in the original proposals [Dij59, Knu77, RR96]. However, the same algorithm can be used for any $(\max, INF)$ problem, e.g., for finding hyperpaths with maximum *bottleneck*, or maximum *P-Product* (products of real numbers in the range $[0-1]$), with the opposite interpretation ("*best*" means "maximum", and "$\prec$" means "*strictly larger than*"). In the following we will refer a problem of "minimum".

Compared with Algorithm *Visit*, Algorithm *SBP* differs in two lines

- extraction policy (step 3): $Q$ is a priority queue (smallest first), where the priority of a node $y$ in the queue is the measure of the best known hyperpath from the source to $y$;
- enqueue/update policy (step 6): if the test succeed, hyperarc $(X, y)$ is the root of the (new) known best hyperpath tree $t_{S,y}$. Hence this hyperarc is stored in *ChildRoot*$[y]$, and node $y$ is inserted or updated in $Q$ with priority $\mu(X, y)$.

Due to the enqueue policy, it is possible that a node will be enqueued several times.

The correctness of Algorithm *SBP* has been proven by Knuth [Knu77] for *SUP* functions, and up to *WSUP* functions by Ramalingam and Reps [RR96].

The time complexity, discussed in [Knu77, RR96], is $O(|H|\log|N| + |\mathcal{H}|)$ worst case time, that can be reduced to $O(|N|\log|N| + |\mathcal{H}|)$ by using Fibonacci heaps [FT87] for the implementation of priority queues. But this complexity has validity only within *SUP* functions, at least for the pseudocode provided above.

Of course we can extend all these considerations to the case of a $(\max, \mathit{INF})$ optimization problem.

### *A hard instance with a WSUP measure function*

A strategy which is purely based on a priority queue, such as Sort by Priority presented above, has problems if the considered function have some *irrelevant* variables, as in *WSUP* functions. In fact, in this case, one cannot rely upon the inequality $g(x_1, \ldots, x_k) \geq x_i$ in the definition of the *SUP* functions. In particular we show that it can be very inefficient trying to find out new hyperarcs and computing the value of optimal hyperpaths in a single run of algorithm *SBP*.

Let us consider a simple 1-relevant function, such as the *gap*, which is strictly *WSUP* in case of positive hyperarc weights. We know that *WSUP* functions are 1-Cycle Bounded (Theorem 6.4), but the problem is that, by using a pure *SBP* approach when the optimal hyperpath is cyclic, each hyperarc is scanned without knowing whether it is part of the relevant tree, or it is irrelevant. The result is that the same node can enqueued several times in the priority queue, and a portion of the hypergraph can be visited as many times.

As an example of "hard" instance, let us consider the hypergraph in Figure 24, where we are interested in computing the hyperpaths from the source $\{s\}$ with minimum *gap*. Beyond node $c$ there is a large region which can be visited from $c$ at a small cost.

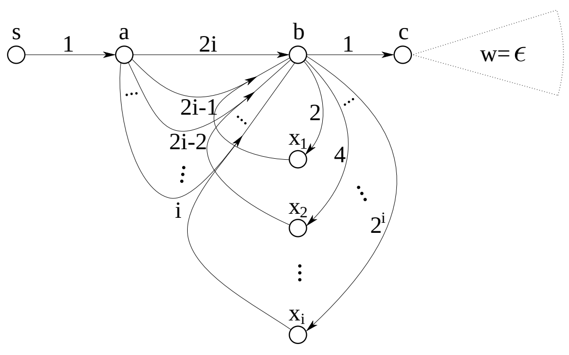


Figure 24: A hard instance for the problem of finding a hyperpath of minimum *gap*.

Let us consider Algorithm *SBP* running on this graph. After the initialization, we will analyze the priority queue $Q$ and the further steps, for executions of the main loop:

(1st-2nd) $Q$: node $s$ alone, then node $a$ alone. The adjacency list *fstar*$(a)$ considered in step 4 contains the hyperarc $(a, b)$ with weight $2i$, and a series of hyperarcs $(a\,x_j, b)$ with weight $2i - j$, for $j = 1, 2, \ldots, i$. At this point only hyperarc $(a\,x_j, b)$ can be scanned (hence $b$ is enqueued with priority $2i + 1$), and all nodes $x_1, x_2, \ldots, x_i$ are tested, but they cannot be enqueued.

(3rd) $Q$: node $b$ alone, which is visited: its distance is computed as $d[b] = 2i+1$. All the (hyper)arcs in its adjacency list are scanned and, since the source set of all these, $\{b\}$, contains a single visited node, the tested nodes are all enqueued: $c$, and all $x_j$ with $j = 1, 2, \ldots, i$.

(4th) $Q$ contains: node $c$ with priority $2+2i$, and each $x_j$ with priority $2i+1+2^j$, with $j = 1, 2, \ldots, i$. The problem is here: $c$ is dequeued and its distance is computed as $d[c] = 2i + 2$, and from now on Algorithm *SBP* will visit the large component on the right.

(z-th) $Q$: $Q$ contains: each $x_j$ with priority $2i + 1 + 2^j$, with $j = 1, 2, \ldots, i$. Hence $x_1$ is dequeued and visited, and the only hyperarc in its adjacency list, $(x_1, b)$ can be scanned. Node $b$ is enqueued *a second time*, since the new hyperpath with last hyperarc $(x_1, b)$ has gap $2i$, which is better (smaller) than the previous one $(2i + 1)$.

(further) From now on Algorithm *SBS* will visit $b$, $c$ and the large component beyond $c$ for each of the $x_2, \ldots, x_i$.

Eventually, the algorithm terminates correctly, and finds out the hyperpath with optimal gap to $c$ and beyond.

Hence, let us consider a hypergraph of size $|\mathcal{H}|$, with source area $a = \Theta|\mathcal{H}|$, and run on this instance the Algorithm pseudocoded as Sort By Priority. In this case $\Omega(|N|)$ nodes (a portion of the hypergraph) are enqueued and dequeued $\Omega(|N|)$ times, every time scanning their *fstar*, up to a bound $\Omega(|N| \cdot |\mathcal{H}|)$ hyperarc scan (without considering the priority queue). We remark that this event can never occur in any instance if the measure function is *SUP*.

## *ISBP: Impatient Sort By Priority*

The previous example shows the inconvenient of propagating at the same time the reachability from the source while searching for the best hyperpath, at least in case of weak functions.

We propose a simple approach, based on a sequence of two phases:

1. Execute a *Visit* of hypergraph $\mathcal{H}$ from the given source $S$, in order to find out the *reachable nodes* and the *scannable hyperarcs*; let $\mathcal{H}_R = (N_R, H_R)$ be the reachable portion of the original hypergraph from the given source $S$.
2. Execute Algorithm Impatient Sort-By-Priority, on the subhypergraph $\mathcal{H}_R = (N_R, H_R)$, returned in the first phase.

In this algorithm, shown below, we deal separately with the two aspects - reachability, and optimization.

```
Algorithm ISBP (Impatient Sort By Priority)
/* This runs on the subhypergraph $\mathcal{H}_R = (N_R, H_R)$, known to be reachable from source $S$ */
Q is a PRIORITY QUEUE, $\mu_0 = 0$, $\mu_\infty = \infty$
1. INITIALIZATION
      for each $n \in N_R$:  set $d[n] = \mu_\infty$; set $ChildRoot[n] = null$;
      for each $s \in S$:  enqueue $s$ in $Q$; set $d[x] = \mu_0$;
2. MAIN LOOP:
      while $Q$ is not empty:
3. VISIT node $x$ chosen from $Q$ according a given EXTRACTION POLICY
         extract $x$ from $Q$ having best priority $d[x]$;
4. for each hyperarc $(X, y) \in fstar[x]$ complying a SCAN POLICY
         for each hyperarc $(X, y) \in fstar(x)$ {ISBP doesn't wait that all nodes in $X$ have been visited}
5. IF TEST$(y)$ succeed, update data structures accordingly
               if $\mu(X, y) \prec d[y]$
                  set $ChildRoot[y]$ to $(X, y)$;
                  set $d[x] = \mu(X, y)$;
6. possibly ENQUEUE node $y$ in $Q$
                  if $y \notin Q$
                     insert $y$ in $Q$;
```

This “impatient” version of Sort By Priority is different from the previous one because it exploit the fact that all the considered nodes are known to be reachable, and therefore, all the hyperarcs are known to to be “scannable”, hence we have split the two tasks - reachability and optimization. Beyond that, the code differs only in one row: the scan policy in step 4: when *ISBP* takes into consideration a hyperarc $(X, y) \in fstar(x)$, there is no scanning policy at all, i.e., this is scanned anyway *without waiting* that all nodes in the source set $X$ have been visited, therefore the test on $y$ is performed all the times, with the computation of $\mu(X, y) = F_{\langle X,y \rangle}(d[x_1], d[x_2], \ldots, d[x_k])$.

Hence algorithm *ISBP* computes the value of $\mu$ as soon as the value of a first argument (any of the $d[x_i]$) is available, and keeps on recomputing $\mu$ as new arguments (the measure of further subhyperpaths) are available.

If function $\mu$ depends on all its arguments, this is a disadvantage with respect to *SBP*, although not in terms of asymptotic complexity. We shortly discuss these disadvantages together with considerations on the complexity of *ISBP*. Of course this discussion applies to the case of a *WSUP*/ *WINF* measure functions, where both *SBP* and *ISBP* work.

- *ISBP* will enqueue a node $y$ as soon as the *first* node in the source set of the *first* entering hyperarc is visited. Therefore, many nodes will be enqueued earlier: this implies a priority queue with larger size, in the average. In any case this queue contains at most $|N|$ nodes.
- Whereas *SBP* tests in constant time whether all nodes in $X$ have been visited, *ISBP* computes the value of $\mu$ and tests whether this "improves" the current best hyperpath to $y$. If $\mu$ can be (re)computed in constant time (note that only one of its arguments has changed - only as an improvement), this computation does not change the complexity; minimum, maximum, sum, product are examples of functions requiring a constant time to be recomputed when one of the arguments "improves". Anyway, in the worst case, for each hyperarc $(X, y)$, function $\mu$ is (re)computed $O(|X|)$ times when each $x_i$ is visited and the value of $d[x_i]$ is assessed. The total number of times this can happen is $a$, where $a$ is the source area, hence the overall is $O(|\mathcal{H}|)$; the number of times that function $\mu$ is computed by *SBP* is $O(|H|)$.

On the positive side, we have that Algorithm *ISBP* computes the correct measure of the optimal hyperpath as soon as the last *relevant* argument is available. If function $\mu$ is 1-relevant (such as the *gap*), this is the first one. As an example, let us consider the optimization problem proposed in figure 24. When node $a$ is visited in the second execution of the main loop, node $b$ is enqueued, say with priority $2i + 1$, but all hyperarcs in *fstar*$(a)$ are known to be scannable and then, before leaving the loop corresponding to the visit of node $a$, the possible value of the *gap*$(b)$ is computed by using all the hyperarcs $(a\, x_j, b)$ for possible updates on node $b$ in the queue. When node $b$ will be scanned in the third execution of the loop, its priority corresponds to the optimal value $i + 1$.

We have proved that, for any *WSUP* (and *WINF*) function, there exists always a 1-Cycle Convergent optimal tree (Theorem 6.4) where the two relevant and irrelevant portions, taken separately are acyclic 6.2.

In the first phase, Algorithm *Impatient Sort By Priority* builds up a *Visit* that will contain a representation of all the irrelevant subtrees: an example is the data structure on the right in Figure 23. In the second phase *ISBP* builds up only a representation of the relevant portion of the Optimal tree (on the left in the same figure). Any claim done for a *SUP* optimal tree are valid in this case, as well: this tree is acyclic, and can be found by a Sort-By-Priority approach.

Algorithm *ISBP* is suitable for finding a *Single Source Optimal Hyperpath tree* for any problem $\mathcal{P} \in (\min, \mathit{WSUP}) \cup (\max, \mathit{WINF})$. In particular this algorithm, finds optimal hyperpaths that can be returned by a suitable navigation of the data structure, within the classes of *WSUP* and *WINF* functions, even in case of a *cyclic* hyperpaths. In particular, in this case, when the algorithm exits, the array *ChildRoot* provides the relevant tree of the optimal hyperpath. All the irrelevant subtrees can be collected by linking each occurrence of an irrelevant node $u$ in the tree with the root of the subtree rooted in $u$ generated in the first phase of the algorithm, as shown in Figure 23 (right side): this tree is actually a spanning hyperpath tree and it is acyclic.

As a summary of compared performances, *ISBP* works on problems in $(\min, \mathit{WSUP}) \cup (\max, \mathit{WINF})$ and has the same asymptotic cost as *SBP*, i.e., $O(|H| \log |N| + |\mathcal{H}|)$ worst case time, or $O(|N| \log |N| + |\mathcal{H}|)$ time by using Fibonacci heaps [FT87]. On the other side, *SBP* achieve this performance only on problems in $(\min, \mathit{SUP}) \cup (\max, \mathit{INF})$. By the way, the number of times that function $\mu$ is computed is: $O(|H|)$ by *SBP*, and $O(\min\{a, k|H|\})$ by *ISBP* in any $k$-relevant hypergraph.

## *SBS: Sort By Structure*

A different approach to compute optimal hyperpaths, which we will refer to as *SBS*, may be used in case of acyclic hypergraphs.

Directed acyclic hypergraphs naturally arise in several applications. As an example, in Automatic Speech Recognition (ASR) directed hypergraphs have been adopted as a computational tool in order to select the path with maximum score in a word lattice, based on a Probabilistic Context-Free Grammar [KM04, Ned03]. In such cases, the adoption of techniques, specifically conceived for *acyclic* hypergraphs, provide both extremely fast algorithms (i.e., linear time), a very simple implementation, and works on $(\min, \mathit{GSUP}) \cup (\max, \mathit{GINF})$ problems, i.e., we do not need triangle inequality, but only monotonicity.

The proposed strategy consists in computing shortest hyperpaths following a topological sort of the target nodes. This approach replies on hypergraphs a well known strategy for computing a Single-Source Shortest-Path Tree in a directed acyclic graph (see, e.g., [CLRS09], Section 24.2).

We propose a simple approach, based on a sequence of two phases, as in the previous proposal:

1. Execute a *Visit* of hypergraph $\mathcal{H}$ from the given source $S$, in order to find out the *reachable nodes* and the *scannable hyperarcs*; let $\mathcal{H}_R = (N_R, H_R)$ be the reachable portion of the original hypergraph from the given source $S$.
2. Execute Algorithm Impatient Sort-By-Structure, on the subhypergraph $\mathcal{H}_R = (N_R, H_R)$, returned by phase 0.

```
   Algorithm Sort_By_Structure [$\mathcal{H}$ : acyclic hypergraph]
/* This runs on the subhypergraph $\mathcal{H}_R = (N_R, H_R)$, known to be reachable from source $S$ */
Q is a SIMPLE QUEUE, $\mu_0 = 0$, $\mu_\infty = 1$
1. INITIALIZATION
      for each $n \in N_R$:  set $d[n] = \mu_\infty$; set $ChildRoot[n] = null$;
      for each $s \in S$:  enqueue $s$ in $Q$; set $d[x] = \mu_0$;
2. MAIN LOOP:
      while $Q$ is not empty:
3. VISIT node $x$ chosen from $Q$ according a given EXTRACTION POLICY
         extract any $x$ from $Q$;
4. for each hyperarc $(X, y) \in fstar[x]$ complying a SCAN POLICY
         for each hyperarc $(X, y) \in fstar(x)$ s.t. all nodes in $X$ have been visited:
5. IF TEST$(y)$ succeed, update data structures accordingly
            if $\mu(X, y) \prec d[y]$
               set $ChildRoot[y]$ to $(X, y)$;
               set $d[x] = \mu(X, y)$;
6. possibly ENQUEUE node $y$ in $Q$
            if$(X, y)$ is the last scanned hyperarc entering node $y$:
                  insert $y$ in $Q$;
```

Since this algorithm uses a simple queue, it performs a sort of "visit"; differently from the basic Algorithm *Visit*, algorithm *SBS* uses the word *"last"* to decide whether enqueue a node in step 6, as suggested above, in order to force the vertices to be visited according a topological sort.

For a weighted hypergraph $\mathcal{H}_W = \langle N, H; w \rangle$ having a representation of size $|H| = |N| + |H| + |S|$, where $|S| = \sum\{|X| \text{ s.t. } X \text{ is a source in } \mathcal{H}\}$, the algorithm requires $O(|H|)$ time and space.

## *Selecting an Algorithm for a Hyperpath Optimization Problem*

We summarize the selection criteria among the algorithms proposed in this Section. We consider the *Single-Source Optimal Hyperpath problem*, and we are given: an optimization problem $\mathcal{P} = (\Phi, \mu)$ on a directed hypergraph $\mathcal{H} = \langle N, H \rangle$, where $\mu$ is a measure function, $\Psi \in \{\min, \max\}$ is the optimization criterion. We are interested in finding the $(\Phi, \mu)$-optimal hyperpaths from a given source set $S \subseteq N$ to any other node in $N$, therefore we need to compute both the measure of an optimal hyperpath from $S$ to $z$, and the listing of all the hyperarcs in such hyperpath.

- If the hypergraph is acyclic and t the problem is $\mathcal{P} \in (\min, \mathit{GSUP}) \cup (\max, \mathit{GINF})$, the best solution is *Sort By Structure*, running in linear time, i.e., $O(|\mathcal{H}|)$ for a hypergraph of size $|\mathcal{H}|$.
- If the problem is $\mathcal{P} \in (\min, \mathit{SUP}) \cup (\max, \mathit{INF})$, the best solution is *Sort By Priority*, requiring $O(|\mathcal{H}| \log |\mathcal{H}|)$ worst case time, while function $\mu$ is computed $O(|H|)$ times.
- If the problem is $\mathcal{P} \in (\min, \mathit{WSUP}) \cup (\max, \mathit{WINF})$, then it is 1-Cycle-Convergent and the best choice is *Impatient Sort By Priority*, still requiring $O(|\mathcal{H}| \log |\mathcal{H}|)$. In this case, function $\mu$ is computed $O(\min\{k \cdot |H|, a\})$ times, if the function is $k$-relevant.

In all these cases the algorithms will return both the distances, and the optimal hyperpath, that can be easily collected from the available data structures.

If the problem on a given hypergraph consists in: computing the reachable portion from a source set $S$, determining the acyclicity, or finding a topological sort, a solution in linear time is given by algorithm *Visit* and its variants.

## 9 Conclusions and Open Issues

Following and extending previous works, primarily by Knuth [Knu77] and Ramalingam-Reps [RR96], we have proposed a classification of tractable hyperpath optimization problems based on:

- the adopted measure function $\mu(x_1, x_2, \ldots, x_k)$ under the minimal feature of being *Value-Based*;
- the resulting structure of the optimal hyperpaths, finding properties and new characterizations of these classes.

We show how, when additional constraints are imposed on the set of hyperarcs (as opposed to a purely Value-Based approach), very often we came up to untractable problems.

We have provided a series of algorithms for these problems, providing a unifying view within a common algorithmic pattern, investigating and clearing their effectiveness when applied to optimization problems in the proposed classes.

Many optimization problems remain open. One is to fill with effective algorithms more general problems with respect to the ones covered by the proposed algorithms. In particular, an issue that we have only skimmed is the possible mutual interaction between Linear Programming and hypergraphs.

In this paper we have focused on the "static" problem. Many results have been developed for dynamic hyperpath problems, often arising in practice - at least as often as the propagation of cell updates within a spreadsheet. A study of properties, algorithms, and computational models for incremental hyperpath optimization problems deserves a dedicated effort.

An experimentation of the algorithms proposed here, as well as many others invented and reinvented in so many domains, would provide a sharper evidence of the properties stated in this paper, and of the relative merits of each solution, and would suggest further improvements.

We look with attention to the wide range of applications where the directed hypergraphs provide a strong leverage in order to model and tackle combinatorial problems. We strongly hope that a clear and simple way of classifying problems arising in any realm, together with a sharp vision of whether and why a given algorithm works in a given situations, will boost the adoption of better solutions - since too often good achievements in this field are underused, because sometimes they tend to be clumsy.

Quoting a consideration from [KM04], *"... we believe that the hypergraph presentation allows easier access to a greater variety of algorithmic tools, and presents a clearer, more visually appealing intuition."*

## References

[ADS83] Giorgio Ausiello, Alessandro D'Atri, and Domenico Saccà. Graph algorithms for functional dependency manipulation. *Journal of the ACM*, 30:752–766, 1983.

[ADS86] Giorgio Ausiello, Alessandro D'Atri, and Domenico Saccà. Minimal representation of directed hypergraphs. *SIAM Journal on Computing*, 15:418–431, 1986.

[AFLN10] Paola Alimonti, Esteban Feuerstein, Luigi Laura, and Umberto Nanni. Linear Time Analysis of Properties of Conflict-Free and General Petri nets. *Theoretical Computer Science*, (to appear), 2010.

[AFN92] Paola Alimonti, Esteban Feuerstein, and Umberto Nanni. Linear time algorithms for Liveness and Boundedness in Conflict-Free Petri nets. In *1st Latin American Theoretical Informatics*, volume 583, pages 1–14. Lecture Notes in Computer Science, Springer-Verlag, 1992.

[AI91] Giorgio Ausiello and Giuseppe F. Italiano. Online algorithms for polynomially solvable satisfiability problems. *Journal of Logic Programming*, 10:69–90, 1991.

[AIN90] Giorgio Ausiello, Giuseppe F. Italiano, and Umberto Nanni. Dynamic maintenance of directed hypergraphs. *Theoretical Computer Science*, 72(2-3):97–117, 1990.

[AIN92] Giorgio Ausiello, Giuseppe F. Italiano, and Umberto Nanni. Optimal traversal of directed hypergraphs. Technical Report TR-92-073, ICSI - International Computer Science Institute, Berkeley (CA), 1992.

[Aus88] Giorgio Ausiello. Directed hypergraphs: Data structures and applications. In *13th Colloquium on Trees in Algebra and Programming, CAAP '88*, volume 299 of *Lecture Notes in Computer Science*, pages 295–303. Springer Berlin / Heidelberg, 1988.

[Ber73] Claude Berge. *Graphs and Hypergraphs.* North Holland, Amsterdam, 1973.

[Ber89] Claude Berge. *Hypergraphs - Combinatorics of Finite Sets.* North Holland, Amsterdam, 1989.

[CDP04] Sanjay Chawla, Joseph Davis, and Gaurav Pandey. On local pruning of association rules using directed hypergraphs. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, pages 832–841. IEEE Computer Society, 2004.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 2009.

[Dij59] Edsger W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[EGB06] Thomas Eschbach, Wolfgang Günther, and Bernd Becker. Orthogonal hypergraph drawing for improved visibility. *Journal of Graph Algorithms and Applications*, 10:141–157, 2006.

[EKC$^+$08] Brendan Elliott, Mustafa Kirac, Ali Cakmak, Gokhan Yavas, Stephen Mayes, En Cheng, Yuan Wang, Chirag Gupta, Gultekin Ozsoyoglu, and Zehra Meral Ozsoyoglu. PathCase: pathways database system. *Bioinformatics*, 24(21):2526–2533, 2008.

[EKL08] Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Solving Monotone Polynomial Equations. In *5th IFIP International Conference On Theoretical Computer Science (TCS)*, volume 273 of *IFIP International Federation for Information Processing*, pages 285–298. Springer, 2008.

[Fra07] Ganna Frankova. Service level agreements: web services and security. In *ICWE'07: Proceedings of the 7th international conference on Web engineering*, pages 556–562, Berlin, Heidelberg, 2007. Springer-Verlag.

[FT87] Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-Completeness.* W. H. Freeman, 1979.

[GLNP92] Giorgio Gallo, Giustino Longo, Sang Nguyen, and Stefano Pallottino. Directed hypergraphs and applications. Technical Report 03/90, Dip. di Informatica, Univ. of Pisa, Italy, Corso Italia 40, I-56125 Pisa, Italy, 1990 (revised version, 1992).

[GLNP93] Giorgio Gallo, Giustino Longo, Sang Nguyen, and Stefano Pallottino. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201, 1993.

[GMKT97] Dimitrios Gunopulos, Heikki Mannila, Roni Khardon, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning (extended abstract). In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems*, pages 209–216, New York, NY, USA, 1997. ACM.

[GP92] Giorgio Gallo and Stefano Pallottino. Hypergraph models and algorithms for the assembly problem. Technical Report 06/92, Dip. di Informatica, Univ. of Pisa, Italy, Corso Italia 40, I-56125 Pisa, Italy, 1992.

[GR90] Giorgio Gallo and Gabriella Rago. A hypergraph approach to logical inference for datalog formulae. Technical Report 28/90, Dip. di Informatica, Univ. of Pisa, Italy, Corso Italia 40, I-56125 Pisa, Italy, 1990.

[GS98] Giorgio Gallo and Maria Grazia Scutellà. Directed hypergraphs as a modelling paradigm. *Rivista di matematica per le scienze economiche e sociali*, 21:97–123, 1998.

[GS99] Giorgio Gallo and Maria Grazia Scutellà. Directed hypergraphs as a modelling paradigm. Technical Report 02/99, Dip. di Informatica, Univ. of Pisa, Italy, Corso Italia 40, I-56125 Pisa, Italy, 1999.

[JHY10] Hai Jin, Li Huang, and Pingpeng Yuan. K-radius subgraph comparison for rdf data cleansing. In Lei Chen, Changjie Tang, Jun Yang, and Yunjun Gao, editors, *Web-Age Information Management*, volume 6184 of *Lecture Notes in Computer Science*, pages 309–320. 2010.

[KHT09] Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. Hypergraphs and cellular networks. *PLoS Computational Biology*, 5(5), May 2009.

[KM04] Dan Klein and Christopher D. Manning. Parsing and hypergraphs. In *H. Bunt, J. Carroll, G. Satta (eds.), New developments in parsing technology*, pages 351–372. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[Knu73] Donald E. Knuth. *The Art of Computer Programming.* Addison-Wesley, Reading, Massachusetts, 1973.

[Knu77] Donald E. Knuth. A generalization of dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, 1977.

[MM73] A. Martelli and U. Montanari. Additive and/or graphs. In *IJCAI'73: Proceedings of the 3rd international joint conference on Artificial intelligence*, pages 1–11, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[MMZ06] Fabio Massacci, John Mylopoulos, and Nicola Zannone. Hierarchical hippocratic databases with minimal disclosure for virtual organizations. *The VLDB Journal*, 15(4):370–387, 2006.

[MN98] Patrice Marcotte and Sang Nguyen. Hyperpath formulations of traffic assignment problems. In *Marcotte, P., Nguyen, S. (Eds.), Equilibrium and Advanced Transportation Modelling*, pages 175–200. Kluwer Academic Publishers, Dordrecht, 1998.

[MSS04] Rolf H. Möhring, Martin Skutella, and Frederik Stork. Scheduling with and-or precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

[MV07] Amadis A. M. Morales and Maria E. Vidal. A directed hypergraph model for rdf. In *Knowledge Web PhD Symposium (co-located with the 4th Annual European Semantic Web Conference)*, 2007.

[MY07] Fabio Massacci and Artsiom Yautsiukhin. Modelling quality of protection in outsourced business processes. In *IAS '07: Proceedings of the Third International Symposium on Information Assurance and Security*, pages 247–252, Washington, DC, USA, 2007. IEEE Computer Society.

[Ned03] Mark-Jan Nederhof. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003.

[Nil82] Nils J. Nilsson. *Principles of Artificial Intelligence.* Springer Verlag, Berlin, 1982.

[NP88] Sang Nguyen and Stefano Pallottino. Equilibrium traffic assignment for large scale transit networks. *European Journal of Operational Research*, 37(2):176–186, 1988.

[NPG98] Sang Nguyen, Stefano Pallottino, and Michel Gendreau. Implicit enumeration of hyperpaths in a logit model for transit networks. *Transportation Science*, 32(1):54–64, 1998.

[Ozt08] Can Ozturan. On finding hypercycles in chemical reaction networks. *Applied Mathematics Letters*, 21(9):881–884, 2008.

[Pre00] Daniele Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123(2):315–324, 2000.

[RR96] Ganesan Ramalingam and Thomas Reps. An incremental algorithm for a generalization of the shortest path problem. *Journal of Algorithms*, 21(2):267–305, 1996.

[RSC97] Mysore Ramaswamy, Sumit Sarkar, and Ye-Sho Chen. Using directed hypergraphs to verify rule-based expert systems. *IEEE Trans. on Knowl. and Data Eng.*, 9(2):221–237, 1997.

[TT09] Mayur Thakur and Rahul Tripathi. Linear connectivity problems in directed hypergraphs. *Theoretical Computer Science*, 410(27-29):2592–2618, 2009.

[WLHW09] Gang Wu, Juan-Zi Li, Jian-Qiang Hu, and Ke-Hong Wang. System $\pi$: A native rdf repository based on the hypergraph representation for rdf data model. *Journal of Computer Science and Technology*, 24:652–664, 2009.