WORKING PAPER L-2004-04

WP

Lars Relund Nielsen, Daniele Pretolani & Kim Allan Andersen

Finding the *K* shortest hyperpaths using reoptimization

Logistics/SCM Research Group

Department of Accounting, Finance and Logistics

Finding the K shortest hyperpaths using reoptimization

Lars Relund Nielsen*
Biometry Research Unit
Research Centre Foulum
P.O. Box 50
DK-8830 Tjele
Denmark

Kim Allan Andersen
Department of Accounting, Finance and Logistics
Aarhus School of Business
Fuglesangs Allé 4
DK-8210 Aarhus V
Denmark

Daniele Pretolani
Dipartimento di Matematica e Informatica
Università di Camerino
Via Madonna delle Carceri
I-62032 Camerino (MC)
Italy

November 15, 2004

Abstract

The shortest hyperpath problem is an extension of the classical shortest path problem and has applications in many different areas. Recently, algorithms for finding the K shortest hyperpaths in a directed hypergraph have been developed by Andersen, Nielsen and Pretolani. In this paper we improve the worst-case computational complexity of an algorithm for finding the K shortest hyperpaths in an acyclic hypergraph. This result is obtained by applying new reoptimization techniques for shortest hyperpaths.

The algorithm turns out to be quite effective in practice and has already been successfully applied in the context of stochastic time-dependent networks, for finding the K best strategies and for solving bicriterion problems.

Keywords: Network programming, Directed hypergraphs, K shortest hyperpaths, K shortest paths.

1 Introduction

Directed hypergraphs are an extension of directed graphs and undirected hypergraphs, and represent a general modelling and algorithmic tool successfully used in many different research areas such as artificial intelligence, database systems, propositional logic and transportation networks. For a more general overview on directed hypergraphs see Gallo, Longo, Pallottino, and Nguyen [2]; see Ausiello, Franciosa, and Frigioni [1] for a more recent survey.

The concept of hyperpath and shortest hyperpath was introduced by Nguyen and Pallottino [6]. Some particular shortest hyperpath problems were studied by Jeroslow, Martin,

*Corresponding author (e-mail: lars@relund.dk)

Rardin, and Wang [4] within the more general setting of Leontief flow problems. A general formulation of the shortest hyperpath problem is given in Gallo et al. [2].

Shortest hyperpath problems arise from important practical applications, e.g. in production planning (Gallo and Scutellà [3]). In partiular, they are the core of traffic assignment methods for transit networks, see for instance Wu, Florian, and Marcotte [14] and Nguyen, Pallottino, and Gendreau [7]. Finally, as shown by Pretolani [12], directed hypergraphs can be used to model discrete stochastic (or random) time-dependent networks, where the problem of finding an optimal time-adaptive routing strategy reduces to solving a shortest hyperpath problem in a suitable acyclic *time-expanded* hypergraph. In a stochastic time-dependent network the travel time through an arc is a random variable whose distribution depends on the departure time. Transportation problems on stochastic time-dependent networks have recently attracted a growing attention, see Miller-Hooks and Mahmassani [5], Nielsen, Pretolani, and Andersen [11].

Often in a real application hard constraints not intercepted by the model may occur. In this case an optimal hyperpath satisfying the constraint may be found by enumerating suboptimal hyperpaths, until the hard constraints are satisfied. Furthermore, algorithms based on K shortest hyperpaths procedures can be used to solve bicriterion hyperpath problems. Several algorithms for finding the K shortest hyperpaths in a directed hypergraph were developed by Nielsen, Andersen, and Pretolani [10]. Here computational results show that the CPU times can be reduced dramatically if *reoptimization* is used.

In this paper we improve the complexity of the algorithm finding the K shortest hyperpaths in an acyclic hypergraph by using new reoptimization techniques for shortest hyperpaths. These techniques extend to directed hypergraphs some well known results for the shortest path problem; as we shall see, this extension is not trivial, and is technically rather involved. The resulting algorithm has already been successfully applied to finding the K best strategies in a stochastic time-dependent network (Nielsen [8]), in particular, it has been exploited within algorithms for bicriterion best strategy problems, see Nielsen, Andersen, and Pretolani [9].

The paper is organized as follows. In Section 2 we shortly recall some definitions related to directed hypergraphs and present new reoptimization results. In Section 3 these results are used to develop a new algorithm with improved complexity. Conclusions are drawn in Section 4.

2 Directed hypergraphs

A *directed hypergraph* is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = (v_1, ..., v_n)$ is the set of *nodes*, and $\mathcal{E} = (e_1, ..., e_m)$ is the set of *hyperarcs*. A hyperarc $e \in \mathcal{E}$ is a pair $e = (T(e), h(e))$, where $T(e) \subset \mathcal{V}$ denotes the set of *tail* nodes and $h(e) \in \mathcal{V} \setminus T(e)$ denotes the *head* node. Note that a hyperarc has exactly one node in the head and possibly several nodes in the tail.

The *cardinality* of a hyperarc e is the number of nodes it contains, i.e. $|e| = |T(e)| + 1$. We call e an *arc* if $|e| = 2$. We denote by κ the *size* of $\mathcal{H}$, i.e.

$$\kappa = \sum_{e \in \mathcal{E}} |e|$$

Without loss of generality, we assume $\kappa > n$. We denote by

$$FS(v) = \{e \in \mathcal{E} \mid v \in T(e)\}, \quad BS(v) = \{e \in \mathcal{E} \mid v = h(e)\}$$

the *forward star* and the *backward star* of node v, respectively.

A hypergraph $\tilde{\mathcal{H}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ is a *subhypergraph* of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, if $\tilde{\mathcal{V}} \subseteq \mathcal{V}$ and $\tilde{\mathcal{E}} \subseteq \mathcal{E}$. This is written $\tilde{\mathcal{H}} \subseteq \mathcal{H}$, or we say that $\tilde{\mathcal{H}}$ is *contained* in $\mathcal{H}$. A subhypergraph is *proper* if at least one of the inclusions is strict. Moreover, we denote by $FS_{\tilde{\mathcal{H}}}(v)$ and $BS_{\tilde{\mathcal{H}}}(v)$ the forward and backward star of node v in subhypergraph $\tilde{\mathcal{H}}$, respectively.

A *valid ordering* $V = (v_1, v_2, \ldots, v_n)$ of $\mathcal{H}$ is a topological ordering of the nodes such that, for any $e \in \mathcal{E}$, if $h(e) = v_i$ and $v_j \in T(e)$ then $j < i$. Note that, in a valid ordering any node $v_j \in T(e)$ precedes node $h(e)$. A *valid sub-ordering* $\tilde{V}$ of V is a subsequence of V, that is, the topological ordering of a subset of the nodes in V induced by the ordering V. We write $\tilde{V} \subseteq V$ if $\tilde{V}$ is a valid sub-ordering of V. A *path* P_{st} in $\mathcal{H}$ is a sequence

$$P_{st} = (s = v_1, e_1, v_2, e_2, ..., e_q, v_{q+1} = t)$$

where, for $i = 1, \ldots, q$, $v_i \in T(e_i)$ and $v_{i+1} = h(e_i)$. A node v is *connected* to node u if a path P_{uv} exists in $\mathcal{H}$. A *cycle* is a path P_{st}, where $t \in T(e_1)$. This is in particular true if $t = s$. If $\mathcal{H}$ contains no cycles, it is *acyclic*. It is well-known that $\mathcal{H}$ is acyclic if and only if a valid ordering of the nodes in $\mathcal{H}$ is possible (see [2]).

2.1 Hyperpaths and hypertrees

Definition 1 A hyperpath $\pi_{st} = (\mathcal{V}_\pi, \mathcal{E}_\pi)$ from source s to target t is a subhypergraph of $\mathcal{H}$ such that $\mathcal{E}_\pi = \varnothing$ if $t = s$, otherwise, the $q \geq 1$ hyperarcs in $\mathcal{E}_\pi$ can be ordered in a sequence $(e_1, ..., e_q)$ such that

1. $t = h(e_q)$.
2. $T(e_i) \subseteq \{s\} \cup \{h(e_1), ..., h(e_{i-1})\}, \quad \forall e_i \in \mathcal{E}_\pi$.
3. No proper subhypergraph of π_{st} is an s-t hyperpath.

A node t is *hyperconnected* to s in $\mathcal{H}$ if there exists a hyperpath π_{st} in $\mathcal{H}$. Note that condition 2 implies that a valid ordering of π_{st} is $(s, h(e_1), ..., h(e_q))$. That is, a hyperpath is *acyclic*. Furthermore, condition 3 implies that, for each $u \in \mathcal{V}_\pi \setminus \{s\}$, there exists a unique hyperarc $e \in \mathcal{E}_\pi$, such that $h(e) = u$ and hence for each node $u \in \mathcal{V}_\pi$ there is a unique subhyperpath π_{su} contained in π_{st}. We denote hyperarc e as the *predecessor* of u in π_{st}. The definition of hyperpath can be extended to *hypertrees*.

Definition 2 A directed hypertree of $\mathcal{H}$ with *root* s is an acyclic subhypergraph $\mathcal{T}_s = (\{s\} \cup \mathcal{N}, \mathcal{E}_\mathcal{T})$ with $s \notin \mathcal{N}$ satisfying

$$BS(s) = \emptyset, \quad |BS(v)| = 1, \quad \forall v \in \mathcal{N}$$

A directed hypertree $\mathcal{T}_s$ contains a unique s-u hyperpath for each node $u \in \mathcal{N}$ (see [2]). That is, $\mathcal{T}_s$ is the union of hyperpaths from s to all nodes in $\mathcal{N}$. Moreover, $\mathcal{T}_s$ can be described by a *predecessor function* $p : \mathcal{N} \to \mathcal{E}$; for each $u \in \mathcal{N}$, $p(u)$ is the unique hyperarc in $\mathcal{T}_s$ which has node u as the head. Note that any hyperpath is a hypertree, in particular, it can be defined by a predecessor function on $\mathcal{V}_\pi \setminus \{s\}$.

```
procedure SHTacyclic(s, V, H)
    W(v_1) := 0; for (i = 2 to n) do W(v_i) := ∞;
    for (i = 2 to n) do
        for (e ∈ BS(v_i)) do
            if (W(v_i) > w(e) + F(e)) then
                W(v_i) := w(e) + F(e); p(v_i) := e;
        end for
    end for
end procedure
```

Figure 1: A shortest hyperpath procedure on acyclic hypergraphs.

2.2 The shortest hyperpath problem

Assume that each hyperarc e is assigned a nonnegative real weight $w(e)$. The weight of a path P_{st} is the sum of the weights of the hyperarcs in P_{st}. Given an s-t hyperpath π defined by predecessor function p, a weighting function W is a node function assigning weights $W(u)$ to all nodes in π. The weight of hyperpath π is $W(t)$. We shall restrict ourselves to *additive weighting functions* introduced by Gallo et al. [2], defined by the recursive equations:

$$W(v) = \begin{cases} 0 & v = s \\ w(p(v)) + F(p(v)) & v \in \mathcal{V}_\pi \setminus \{s\} \end{cases} \tag{1}$$

Here $F(e)$ denotes a non-decreasing function of the weights in the nodes of $T(e)$. Note that the weights $W(v)$ can be found by processing the nodes in $\mathcal{V}_\pi$ according to a valid ordering. We shall consider two particular weighting functions, namely the *distance* and the *value*. The *distance* function is obtained by defining $F(e)$ as follows:

$$F(e) = \max_{u \in T(e)} \{W(u)\}$$

and the *value* function is obtained as follows:

$$F(e) = \sum_{u \in T(e)} a_e(u) W(u)$$

where $a_e(v)$ is a nonnegative multiplier defined for each hyperarc e and node $u \in T(e)$.

The *shortest hyperpath problem* can be viewed as a natural generalization of the shortest path problem and consists in finding a *shortest hypertree* containing the shortest hyperpaths from a source s to all nodes in $\mathcal{H}$ hyperconnected to s. In the following the weight of a shortest hyperpath from s to v in $\mathcal{H}$ will be referred to as the weight of node v in a shortest hypertree.

Several efficient algorithms for the shortest hyperpath problem have been proposed; if $\mathcal{H}$ is acyclic a quite fast procedure can be devised (see [2]). The procedure is shown in Figure 1 and needs a valid ordering $V_{\mathcal{H}} = (s = v_1, ..., v_n)$ of $\mathcal{H}$. Since each hyperarc is examined once, the procedure runs in $O(\kappa)$ time.

2.3 End-trees and reoptimization techniques

Consider an s-t hyperpath π defined by predecessor function p.

Definition 3 (end-tree) Consider the subhypergraph $\eta = (\mathcal{V}_\eta, \mathcal{E}_\eta) \subseteq \pi$ defined by a subset of nodes $I_\eta \subseteq \mathcal{V}_\pi$ as follows: if $I_\eta = \varnothing$ then $\eta = (\{t\}, \varnothing)$, otherwise

$$\mathcal{E}_\eta = \bigcup_{v \in I_\eta} p(v), \quad \mathcal{V}_\eta = \bigcup_{e \in \mathcal{E}_\eta} \left(T(e) \cup \{h(e)\}\right);$$

then η is an *end-tree* if and only if it contains at least one v-t path for each node $v \in \mathcal{V}_\eta$.

For an end-tree η we refer to I_η as the set of *inner-nodes* and denote by E_η the set of *leaf-nodes* in η:

$$E_\eta = \mathcal{V}_\eta \setminus I_\eta = \{v \in \mathcal{V}_\eta : |BS_\eta(v)| = 0\}.$$

Note that $E_\eta = \{t\}$ if and only if $I_\eta = \varnothing$. In the following we show that the weight of a hyperpath can be rewritten according to a given end-tree $\eta \subset \pi$. We consider the value weighting function first.

Theorem 1 *Given an end-tree $\eta \subseteq \pi$ the weight of hyperpath π using the value weighting function can be written as*

$$W(t) = \sum\nolimits_{v \in E_\eta} W(v)\, f^\eta(v) \;+\; \sum\nolimits_{v \in I_\eta} w(p(v))\, f^\eta(v) \tag{2}$$

where f^η is recursively defined as follows:

$$f^\eta(u) = \begin{cases} 1 & u = t \\ \sum_{e \in FS_\eta(u)} a_e(u) f^\eta(h(e)) & u \in \mathcal{V}_\eta \setminus \{t\} \end{cases} \tag{3}$$

Proof Let us consider a generic valid ordering $V_\eta = (v_1, ..., v_q = t)$ of η. Moreover, given a generic subset $I \subseteq \mathcal{V}_\eta$ such that $t \in I$, let us define the value $f^I(u)$ for all $u \in \mathcal{V}_\eta$:

$$f^I(u) = \begin{cases} 1 & u = t \\ \sum_{e \in FS^I_\eta(u)} a_e(u) f^I(h(e)) & u \neq t \end{cases}$$

where

$$FS^I_\eta(u) = \left\{e \in FS_\eta(u) :\; h(e) \in I\right\}.$$

Note $f^I(u)$ is a lower bound on $f^\eta(u)$ found by using only the hyperarcs in $FS_\eta(u)$ with head belonging to the set I. Moreover, if I is the set of nodes following node u in V_η, then $f^I(u) = f^\eta(u)$.

If $\eta = (\{t\}, \emptyset)$ then (2) holds trivially. Assume that $I_\eta \neq \emptyset$ and consider the valid ordering V_η. We show (2) holds using induction, processing nodes in V_η in reverse order, i.e., starting with node v_q and proceeding down to node v_1. Given node v_i let I^i and E^i denote the set of inner and leaf-nodes already considered, i.e.

$$I^i = \{v_j \in I_\eta : i \leq j \leq q\}, \quad E^i = \{v_j \in E_\eta : i \leq j \leq q\}.$$

Moreover, let B^i denote the set of not yet considered nodes that belong to (the tails of) the predecessors of nodes in I^i:

$$B^i = \left\{v_j \in \bigcup\nolimits_{u \in I^i} T(p(u)) : j < i\right\}.$$

Consider node $t = v_q$, i.e., $i = q$. According to equation (1) we have

$$W(t) = w(p(v_q)) + \sum\nolimits_{u \in T(p(v_q))} a_{p(v_q)}(u)\, W(u)\,. \tag{4}$$

Since $t \in I_\eta$ we have that $E^q = \emptyset$ and $B^q = T(p(v_q))$; thus by letting $k = q$ we can rewrite (4) as follows

$$\begin{aligned} W(t) = {} & \sum\nolimits_{v \in I^k} w(p(v))\, f^\eta(v) + \sum\nolimits_{v \in E^k} f^\eta(v)\, W(v) \\ & + \sum\nolimits_{u \in B^k} f^{I^k}(u)\, W(u)\,. \end{aligned} \tag{5}$$

Assume that nodes $v_q, ..., v_{i+1}$ have been considered and that (5) holds for $k = i+1$. Note that $v_i \in B^{i+1}$ and $f^{I^{i+1}}(v_i) = f^\eta(v_i)$. We consider two cases. If $v_i \in I_\eta$, we write out $W(v_i)$:

$$\begin{aligned} W(t) = {} & \sum\nolimits_{v \in I^{i+1}} w(p(v))\, f^\eta(v) + \sum\nolimits_{v \in E^{i+1}} f^\eta(v)\, W(v) \\ & + \sum\nolimits_{u \in B^{i+1} \setminus \{v_i\}} f^{I^{i+1}}(u)\, W(u) \\ & + f^{I^{i+1}}(v_i) \left(w(p(v_i)) + \sum\nolimits_{u \in T(p(v_i))} a_{p(v_i)}(u)\, W(u) \right). \end{aligned} \tag{6}$$

Note that $E^{i+1} = E^i$ and, for each $u \in T(p(v_i))$:

$$f^{I^i}(u) = f^{I^{i+1}}(u) + f^{I^{i+1}}(v_i) a_{p(v_i)}(u)$$

thus (6) becomes

$$W(t) = \sum\nolimits_{v \in I^i} w(p(v))\, f^\eta(v) + \sum\nolimits_{v \in E^i} f^\eta(v)\, W(v) + \sum\nolimits_{u \in B^i} f^{I^i}(u)\, W(u)$$

and (5) holds for $k = i$. If $v_i \in E_\eta$ then (5) still holds for $k = i$ since $I^i = I^{i+1}$, $E^i = E^{i+1} \cup \{v_i\}$ and $B^i = B^{i+1} \setminus \{v_i\}$. Proceeding down to node v_1, we have that (5) can be written as

$$W(t) = \sum\nolimits_{v \in E_\eta} f^\eta(v)\, W(v) + \sum\nolimits_{v \in I_\eta} w(p(v))\, f^\eta(v)$$

since $B^1 = \varnothing$. ■

Note that the values of f^η can be computed by processing the nodes backwards with respect to a valid ordering $V_\eta \subseteq V_\pi$, i.e. in $O(\kappa)$ operations. Given two different s-t hyperpaths π and $\tilde{\pi}$ containing the same end-tree η we now have

$$\tilde{W}(t) - W(t) = \sum\nolimits_{v \in E_\eta} \left(\tilde{W}(v) - W(v) \right) f^\eta(v)$$

where $\tilde{W}(v)$ denote the weight of node v in $\tilde{\pi}$, resulting in the following theorem.

Theorem 2 *The weight of hyperpath $\tilde{\pi} \subseteq \tilde{\mathcal{H}}$ is*

$$\tilde{W}(t) = W(t) + \sum\nolimits_{v \in E_\eta} \left(\tilde{W}(v) - W(v) \right) f^\eta(v)\,. \tag{7}$$

Moreover, if

$$\tilde{W}(v) = W(v), \forall v \in E_\eta \setminus \{u\} \tag{8}$$

then (7) reduces to

Corollary 1 *The weight of hyperpath $\tilde{\pi}$ is*

$$\tilde{W}(t) = W(t) + \left(\tilde{W}(u) - W(u)\right) f^\eta(u). \tag{9}$$

Consider the distance weighting function. Given end-tree η, the maximum weight $l^\eta(u)$ of an u-t path contained in η can be found by using the following recursive equations

$$l^\eta(u) = \begin{cases} 0 & u = t \\ \max_{e \in FS_\eta(u)} \{l^\eta(h(e)) + w(e)\} & u \in \mathcal{V}_\eta \setminus \{t\} \end{cases} \tag{10}$$

Moreover, it is easy to see that the following theorem holds (see Nielsen [8, Section 2.4.2] for a formal proof).

Theorem 3 *Given leaf-nodes E_η of η, we have that the weight of s-t hyperpath π is*

$$W(t) = \max_{v \in E_\eta} \{W(v) + l^\eta(v)\}. \tag{11}$$

Now consider two different s-t hyperpaths π and $\tilde{\pi}$ containing end-tree η.

Corollary 2 *Assume that condition* (8) *holds and that $\tilde{W}(u) \geq W(u)$. Then the weight of hyperpath $\tilde{\pi}$ is*

$$\tilde{W}(t) = \max\left\{W(t), \tilde{W}(u) + l^\eta(u)\right\}.$$

Proof Using (11) we have that

$$\begin{aligned} \tilde{W}(t) &= \max_{v \in E_\eta}\left\{\tilde{W}(v) + l^\eta(v)\right\} \\ &= \max\left\{\max_{v \in E_\eta}\{W(v) + l^\eta(v)\}, \tilde{W}(u) + l^\eta(u)\right\} \\ &= \max\left\{W(t), \tilde{W}(u) + l^\eta(u)\right\}. \end{aligned}$$

■

3 Finding the K shortest hyperpaths

The K shortest hyperpath problem addressed in this paper is given as follows: given an acyclic hypergraph $\mathcal{H}$, an origin node s and a destination node t generate the K shortest s-t hyperpaths in $\mathcal{H}$ in nondecreasing order of weight using the value or distance weighting function. An $O(n\kappa K)$ algorithm for this problem, denoted *AYen*, was developed in [10].

Let Π denote the set of s-t hyperpaths in $\mathcal{H}$. The algorithms in [10] are based on an implicit enumeration method, where the set Π is partitioned into smaller subsets by recursively applying a branching operation. Assume that a shortest s-t hyperpath π in Π is known, and defined by predecessor function p. Let

$$V_\pi = (s, u_1, u_2, \ldots, u_q = t) \subseteq V_\mathcal{H}$$

denote the valid ordering of π. Moreover, for $i = 1, ..., q-1$ let $\eta^i = (\mathcal{E}_\eta, \mathcal{V}_\eta) \subseteq \pi$ be the end-tree defined by the set of inner-nodes $I_\eta^i = \{u_{i+1}, ..., u_q\}$, and let $\eta^q = (\{t\}, \emptyset)$. The set $\Pi \setminus \{\pi\}$ is partitioned into smaller subsets using the following branching operation.

Branching Operation 1 *Given the shortest hyperpath π of Π and the valid ordering $V_\pi \subseteq V_\mathcal{H}$, the set $\Pi \setminus \{\pi\}$ is partitioned into q disjoint subsets Π^i, $1 \leq i \leq q$, by letting hyperpaths in Π^i contain η^i and not contain hyperarc $p(u_i)$.*

Clearly, the second shortest hyperpath can be found by finding the shortest hyperpaths in the sets Π^i, $i = 1, ..., q$. Moreover, we can apply Branching Operation 1 to a subset $\Pi^i \subset \Pi$ using a hyperpath $\pi^i \in \Pi^i$, and so on recursively. Now consider the problem of finding the shortest hyperpath $\pi^i \in \Pi^i$, that is, finding a shortest s-t hyperpath containing the end-tree η^i. As shown in [10] this reduces to solving a shortest hyperpath problem on a subhypergraph $\mathcal{H}^i$ defined as follows.

Definition 4 Given π, let subhypergraph $\mathcal{H}^i$, $i = 1, ..., q$ be obtained from $\mathcal{H}$ as follows

1. For each node u_j, $i + 1 \leq j \leq q$, remove each hyperarc in $BS(u_j)$ except $p(u_j)$.
2. Remove hyperarc $p(u_i)$ from $BS(u_i)$.

We say that $\mathcal{H}^i$ is obtained from $\mathcal{H}$ by *fixing* hyperarcs $p(u_j)$, $i + 1 \leq j \leq q$ and *deleting* hyperarc $p(u_i)$.

Let $W(v)$, $v \in \mathcal{V}$ denote the weight of node v in the shortest hypertree $\mathcal{T}_s$, defined by the predecessor function p, and containing the shortest s-t hyperpath π. Consider the subhypergraphs $\mathcal{H}^i$, $i = 1, ..., q$ corresponding to Branching Operation 1 on π. The following theorem has been given in [8].

Theorem 4 *Let $W^i(v)$, $v \in \mathcal{V}$, denote the weight of node v in a shortest hypertree in subhypergraph $\mathcal{H}^i$. Then $W^i(v) = W(v)$ for all nodes v preceding node u_i in the valid ordering $V_\mathcal{H}$. Moreover,*

$$W^i(u_i) = \min_{e \in BS_{\mathcal{H}^i}(u_i)} w(e) + F(W, e) \tag{12}$$

where $F(W, e)$ denotes the function $F(e)$ (as defined in Section 2.1) using weights $W(u)$.

Proof The first claim follows from acyclicity, since none of the hyperarcs removed from $\mathcal{H}$ to obtain $\mathcal{H}^i$ can appear in an s-v hyperpath in $\mathcal{H}$ if v precedes u_i in $V_\mathcal{H}$. The second claim then follows trivially. ■

According to Branching Operation 1 the shortest hyperpath in $\mathcal{H}^i$ contains the end-tree $\eta^i \subset \pi$. Moreover, it is obvious that u_i is a leaf node in η^i. Hence using Theorem 4, Corollary 1 for the value weighting function, and Corollary 2 for the distance weighting function we have the following result.

Theorem 5 *The weight $W^i(t)$ of the shortest hyperpath π^i in $\mathcal{H}^i$ is equal to*

$$W^i(t) = W(t) + \left(W^i(u_i) - W(u_i)\right) f^\eta(u_i)$$

if the value weighting function is considered, where f^η is defined as in (3) *for $\eta = \eta^i$. Similarly, the weight of the minimal hyperpath π^i in $\mathcal{H}^i$ is equal to*

$$W^i(t) = \max\left\{W(t), W^i(u_i) + l^\eta(u_i)\right\}$$

if the distance weighting function is considered. Here l^η is defined as in (10) *for $\eta = \eta^i$.*

Theorems 4 and 5 also imply that, by storing the predecessor function p defining the shortest hypertree in $\mathcal{H}$, we can find the shortest hyperpath π^i in $\mathcal{H}^i$ without computing the shortest hypertree. More precisely, we have the following result:

Corollary 3 *The predecessor function defining the shortest hyperpath $\pi^i = (\mathcal{V}_{\pi^i}, \mathcal{E}_{\pi^i})$ in $\mathcal{H}^i$ is equal to*

1. *Predecessor $p(v)$ for $v \in I_{\eta^i}$.*
2. *The predecessor defined by equation* (12) *for node u_i.*
3. *Predecessor $p(v)$ for $v \in \mathcal{V}_{\pi^i} \setminus (I_{\eta^i} \cup \{u_i\})$.*

A K shortest hyperpaths algorithm using reoptimization can now be formulated. First recall that using Definition 4 each set Π^i can be represented by its corresponding subhypergraph $\mathcal{H}^i$. The algorithm implicitly maintains a candidate set of pairs $(\tilde{\pi}, \tilde{\mathcal{H}})$, where $\tilde{\pi}$ is a shortest hyperpath in subhypergraph $\tilde{\mathcal{H}}$. Assuming that the first k shortest hyperpaths $\pi_1, ..., \pi_k$ have been found, the current candidate set represents a partition of $\Pi \setminus \{\pi_1, ..., \pi_k\}$. Hyperpath π_{k+1} is then found by selecting and removing the pair $(\tilde{\pi}, \tilde{\mathcal{H}})$ containing the hyperpath with minimum weight in the candidate set. Then Branching Operation 1 is applied using hyperpath $\tilde{\pi}$, possibly obtaining new pairs that are added to the candidate set.

Procedure *K-SHPreopt*, shown in Figure 2, describes our algorithm for the value weighting function; the distance function requires minor changes, discussed later. With each node u in $\mathcal{H}$ we associate the labels $W(u)$ and $p(u)$, denoting the weight and the predecessor of u in the shortest hypertree in $\mathcal{H}$. At each iteration of the procedure, the labels $\tilde{p}$ store the predecessor function defining the current hyperpath $\tilde{\pi}$, while the labels $\tilde{f}$ are used to compute the values f^η in the branching operation. The following subprocedures are used.

SHTacyclic$(s, \mathcal{H})$: Find the shortest hypertree of $\mathcal{H}$, i.e. node labels $W(u)$ and $p(u)$, $\forall u \in \mathcal{V}$ (see Figure 1). If t is hyperconnected to s, π denotes the shortest s-t hyperpath. Procedure *SHTacyclic* takes $O(\kappa)$ time.

delMin(): Select and remove from the candidate set and return the pair $(\tilde{\pi}, \tilde{\mathcal{H}})$ with minimum hyperpath weight.

rebuild$(\tilde{\pi}, \tilde{\mathcal{H}})$: Rebuild the subhypergraph $\tilde{\mathcal{H}}$ and the corresponding shortest hyperpath $\tilde{\pi}$ in the pair $(\tilde{\pi}, \tilde{\mathcal{H}})$. This is necessary since each pair is represented implicitly as discussed below. In particular, sets the predecessor $\tilde{p}(u)$ and the weight $\tilde{W}(u)$ for each node u in $\tilde{\pi}$.

findV$(\tilde{\pi})$: Return a valid ordering $V_{\tilde{\pi}} \subseteq V_{\mathcal{H}}$ of the nodes in $\tilde{\pi}$. Requires $O(n)$ time.

insert$(\hat{\pi}, \hat{\mathcal{H}})$: Insert pair $(\hat{\pi}, \hat{\mathcal{H}})$ into the candidate set.

1 **procedure** *K-SHPreopt*$(\mathcal{H}, s, t, K)$
2   *SHTacyclic*$(s, \mathcal{H})$;
3   **if** $(W(t) < \infty)$ **then** *insert*$(\pi, \mathcal{H})$;
4   **else** STOP (there is no s-t hyperpath);
5   **for** $(k := 1 \text{ to } K)$ **do**
6         $(\tilde{\pi}, \tilde{\mathcal{H}}) :=$ *delMin*$()$;
7   **if** $((\tilde{\pi}, \tilde{\mathcal{H}}) = null)$ **then** STOP (there are no more s-t hyperpaths);
8   *rebuild*$(\tilde{\pi}, \tilde{\mathcal{H}})$ and OUTPUT the k'th hyperpath $\tilde{\pi}$;
9         $(s, u_1, ..., u_q = t) :=$ *findV*$(\tilde{\pi})$;
10         $\tilde{f}(t) := 1$; **for** $(i := 1 \text{ to } q - 1)$ **do** $\tilde{f}(u_i) := 0$;
11   **for** $(i := q \text{ to } 1)$ **do**
12             $\tilde{W}^i(t) := calcW(u_i, \tilde{f}(u_i))$;
13   **if** $(\tilde{W}^i(t) < \infty)$ **then** *insert*$(\tilde{\pi}^i, \tilde{\mathcal{H}}^i)$;
14   **for** $(v \in T(\tilde{p}(u_i)))$ **do** $\tilde{f}(v) := \tilde{f}(v) + a_{\tilde{p}(u_i)}(v)\tilde{f}(u_i)$;
15   **end for**
16   **end for**
17 **end procedure**

Figure 2: Finding the K shortest hyperpaths using reoptimization.

$calcW(u_i, \tilde{f}(u_i))$: First, compute the minimum weight $\tilde{W}^i(u_i)$ in $\tilde{\mathcal{H}}^i$ using Theorem 4. Then, compute and return the weight $\tilde{W}^i(t)$ of the shortest hyperpath $\tilde{\pi}^i$ in $\tilde{\mathcal{H}}^i$ using Theorem 5.

Note that Branching Operation 1 is performed on line 11-15. On line 14, we update labels $\tilde{f}$ so that, at each iteration, we have $\tilde{f}(u_i) = f^{\eta}(u_i)$ for the current end-tree $\eta = \tilde{\eta}^i$. If an s-t hyperpath exists in $\tilde{\mathcal{H}}^i$ we insert the pair $(\tilde{\pi}^i, \tilde{\mathcal{H}}^i)$ into the candidate set (line 13). We assume that $calcW$ returns $+\infty$ if no s-t hyperpath exists in $\tilde{\mathcal{H}}^i$.

In order to evaluate the computational complexity of procedure *K-SHPreopt*, we need to describe the implicit representation of the pairs in the candidate set. According to Definition 4, each pair $(\tilde{\pi}, \tilde{\mathcal{H}})$ can be represented by an end-tree $\tilde{\eta}$ and a hyperarc $\tilde{a}$. The subhypergraph $\tilde{\mathcal{H}}$ can be easily built in $O(\kappa)$ time by fixing the hyperarcs in $\tilde{\eta}$ and deleting $\tilde{a}$. Moreover, the shortest hyperpath $\tilde{\pi}$ can be obtained by taking advantage of Corollary 3. To this aim, it suffices to scan the nodes backward, according to the valid order $V_{\mathcal{H}}$, computing the predecessor $\tilde{p}$ for each node in $\tilde{\pi}$ according to Corollary 3. Clearly, this can be done in $O(\kappa)$ time, obtaining an $O(\kappa)$ time complexity for procedure *rebuild*$(\tilde{\pi}, \tilde{\mathcal{H}})$.

Note that when using Branching Operation 1 hypergraph $\mathcal{H}^{i+1}$ only differs slightly from hypergraph $\mathcal{H}^i$. Both contain end-tree η^{i+1}; in $\mathcal{H}^{i+1}$ we remove $p(u_{i+1})$ while in $\mathcal{H}^i$ we fix $p(u_{i+1})$ and remove $p(u_i)$. As a consequence, we can store in a compact and natural way the representations of the pairs $(\pi^i, \mathcal{H}^i)$, $1 \leq i \leq q$, by means of a binary *branching tree*, where each node is labelled by a hyperarc $p(u_i)$, and a left (respectively right) branch correspond to deleting (respectively fixing) the corresponding hyperarc. In this way, we can insert into the candidate set the q pairs generated by Branching Operation 1 with an overall $O(q)$ time and space requirement. The representation of a pair can be recovered in $O(n)$ time by traversing the unique path from the corresponding node to the root in the branching tree.

In addition to the branching tree, a d-heap (see e.g. [13]) is used to select the pair with minimum hyperpath weight in the candidate set. Since we insert at most Kn pairs in the heap the worst case time complexity of procedures *delMin* and *insert* is $O(\log Kn)$. For further details on data structures representing $\mathcal{H}$ and the branching tree, see [8].

It is easy to see that, each time Branching Operation 1 is used, at most $O(\kappa)$ time is required by procedure *calcW* and by the computation of the labels $\tilde{f}$. Our main result then follows.

Theorem 6 *Procedure* K-SHPreopt *finds the K best strategies in $O(\kappa K)$ time.*

Finally, let us consider how procedure *K-SHPreopt* changes, when the distance weighting function is considered. In this case we use labels $\tilde{l}$, instead of labels $\tilde{f}$, in order to compute the path lengths l^η defined in (10). Labels $\tilde{l}$ are initialized to zero in line 10. Then in line 14 we update each $\tilde{l}(v)$ by setting $\tilde{l}(v) := \max\{\tilde{l}(u_i) + w(p(u_i)),\ \tilde{l}(v)\}$.

4 Conclusion

In this paper we presented a new algorithm for finding the K shortest hyperpaths in an acyclic hypergraph. We achieve an $O(\kappa K)$ complexity by exploiting new reoptimization results for shortest hyperpaths in directed hypergraphs. This improves, by a factor n, the $O(\kappa n K)$ complexity of the algorithms presented in [10].

The new algorithm has already been successfully applied to finding the K best strategies in a stochastic time-dependent network (see [8]). Here computational results show that the CPU time can be reduced dramatically if reoptimization is used. The algorithm has also been used successfully as a subalgorithm for solving bicriterion problems in stochastic time-dependent networks (see [9]).

References

[1] G. Ausiello, P.G. Franciosa, and D. Frigioni. Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. *Lecture Notes in Computer Science*, 2202:312–328, 2001.

[2] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201, 1993.

[3] G. Gallo and M.G. Scutellà. A note on minimum makespan assembly plans. *European Journal of Operational Research*, 142(2):309–320, 2002.

[4] R.G. Jeroslow, K. Martin, R.L. Rardin, and J. Wang. Gainfree Leontief substitution flow problems. *Mathematical Programming*, 57:375–414, 1992.

[5] E.D. Miller-Hooks and H.S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, 2000.

[6] S. Nguyen and S. Pallottino. Hyperpaths and shortest hyperpaths. In *Combinatorial optimization (Como, 1986)*, volume 1403 of *Lecture Notes in Math*, pages 258–271. Springer, 1989.

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

[8] L.R. Nielsen. *Route Choice in Stochastic Time-Dependent Networks.* PhD thesis, Department of Operations Research, University of Aarhus, 2004. URL `http://www.imf.au.dk/`.

[9] L.R. Nielsen, K.A. Andersen, and D. Pretolani. Bicriterion shortest hyperpaths in random time-dependent networks. *IMA Journal of Management Mathematics*, 14(3):271–303, 2003.

[10] L.R. Nielsen, K.A. Andersen, and D. Pretolani. Finding the K shortest hyperpaths. *To appear in Computers & Operations Research*, 2004. URL `http://dx.doi.org/10.1016/j.cor.2003.11.014`.

[11] L.R. Nielsen, D. Pretolani, and K.A. Andersen. K shortest paths in stochastic time-dependent networks. URL `http://www.research.relund.dk/`. Submitted, August 2004.

[12] D. Pretolani. A directed hypergraph model for random time-dependent shortest paths. *European Journal of Operational Research*, 123:315–324, 2000.

[13] R.E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Conference Series*. SIAM, 1983.

[14] J.H. Wu, M. Florian, and P. Marcotte. Transit equilibrium assignment: A model and solution algorithms. *Transportation Science*, 28(3):193–203, 1994.

Working Papers from Logistics/SCM Research Group

L-2004-04 Lars Relund Nielsen, Daniele Pretolani & Kim Allan Andersen: Finding the K shortest hyperpaths using reoptimization.

L-2004-03 Søren Glud Johansen & Anders Thorstenson: The (r,q) policy for the lost-sales inventory system when more than one order may be outstanding.

L-2004-02 Erland Hejn Nielsen: Streams of events and performance of queuing systems: The basic anatomy of arrival/departure processes, when focus is set on autocorrelation.

L-2004-01 Jens Lysgaard: Reachability cuts for the vehicle routing problem with time windows.

Handelshøjskolen i Århus

Aarhus
School of Business

ISBN 87-7882-005-7

Department of Accounting, Finance and Logistics
Faculty of Business Administration

Aarhus School of Business
Fuglesangs Allé 4
DK-8210 Aarhus V - Denmark

Tel. +45 89 48 66 88
Fax +45 86 15 01 88

www.asb.dk