JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 413-428 (2017)

DOI:10.1688/JISE.2017.33.2.8

Finding a Minimum Source Set in Directed Hypergraphs

HYUN JUN KIM, KWANG HEE LEE AND MYOUNG HO KIM
Department of Computer Science
Korea Advanced Institute of Science and Technology, KAIST
Daejeon, 34141 Korea
E-mail: {kimhyunjun; kh0611; mhkim}@kaist.ac.kr

A directed hypergraph has hyperedges that connect multiple source nodes and multiple target nodes. These hyperedges are useful to model relationships between multiple entities, which often occur in the biological network. This paper focuses on the problem to find a minimum cardinality source set shortly, a Minimum Source Set (MinSS) that can reach a set of given target nodes. However, this problem is NP-hard, and hence requires a large amount of computation time even for a problem with a small size. We propose the MSS index where for each node the set of precomputed all minimal source sets is maintained. By constructing an MSS index in advance, a MinSS for any node can be obtained in a very short time. To avoid too much large size of the MSS index for a very large hypergraph, we propose a reduction method using the source set proxy. We also describe how to maintain the MSS index for insertion and deletion of nodes and hyperedges. We show through performance experiments that our proposed method can be a viable solution for the MinSS problem in the sense that the MSS index can handle the problems with reasonably large sizes in acceptable amounts of time.

Keywords: directed hypergraph, minimal source set, index, graph processing, knowledge discovery

1. INTRODUCTION

A directed hypergraph is a generalization of a directed graph. An edge of a directed hypergraph can connect more than two nodes, in contrast to edges of directed graphs that can connect only two nodes. In directed graphs, an edge is an ordered pair of a source node and a target node. In directed hypergraphs, an edge, which is called a hyperedge, is an ordered pair of a set of source nodes and a set of target nodes. A path in directed hypergraphs is called a hyperpath. While a path on a directed graph is a sequence of edges, a hyperpath is a tree of directed hyperedges. A reachable set $R(X)$ is a set of all nodes to which there exists a hyperpath from a set X of nodes. $R(X)$ can be computed by a hypergraph traversal algorithm in polynomial time. Inversely, we can consider a problem of finding a minimum cardinality source set X from $R(X)$ where a source set is a set of nodes that can reach $R(X)$. Here, a member of a source set must be a *startable* node where a set of startable nodes is designated in advance. We define such a problem as the Minimum Source Set (MinSS) problem. MinSS is NP-hard, and hence any approach to directly compute MinSS may require a large amount of time even for problems whose sizes are not very large. We propose an indexing approach that precomputes all minimal source sets (MSS) of each node in advance for fast computation of the MinSS.

The MinSS problem can be found in many applications in practice. For example, consider a hypergraph of chemicals, genes, and diseases in Fig. 1. In this hypergraph,

Received December 30, 2015; revised February 23, 2016; accepted April 26, 2016.
Communicated by Hee Kap Ahn.

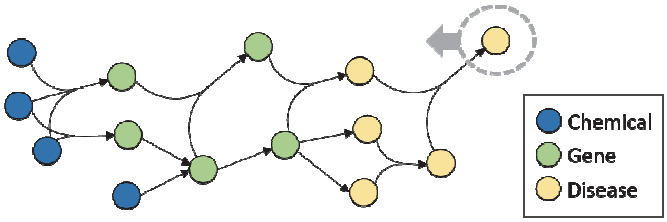

Fig. 1. A directed hypergraph of chemicals, genes, and diseases.

suppose we want to find the smallest combination of chemicals that can affect a certain disease. This question can be answered by solving the MinSS problem on the given hypergraph, where nodes representing chemicals are startable nodes.

The remainder of this paper is organized as follows. In Section 2, we describe directed hypergraphs and other basic concepts. Related works are in Section 3. Section 4 defines the MinSS problem and discusses a naive approach. We propose the MSS index in Section 5. In section 6 we describe an extended problem that can be handled by the MSS index. Experimental results are discussed in Section 7. Finally Section 8 concludes the paper.

2. PRELIMINARY

Directed hypergraphs have edges that connect one or more source nodes with one or more target nodes. Those edges are called hyperedges. A simple edge that has one source node and one target node is also a hyperedge. For convenience of description, we focus on hyperedges with a single target node, which are defined as B-arcs in [9]. A directed hypergraph used in this paper is defined as follows.

Definition 1: A directed hypergraph H is a pair $<V, E>$ where V is a set of nodes and E is a set of hyperedges. A hyperedge $<S, t>$ in E is an ordered pair of a set of source nodes $S \subseteq V$ and a target node t in V.

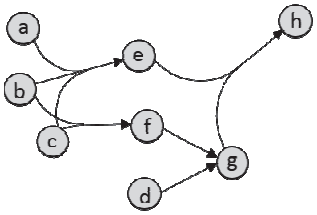

Fig. 2. A directed hypergraph.

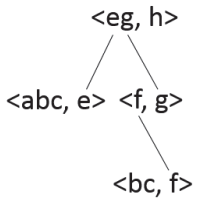

Fig. 3. A hyperpath $T_{abc,h}$ in Fig. 2.

In Fig. 2, the directed hypergraph H is $<V, E>$ where $V=\{a, b, c, d, e, f, g, h\}$ and $E=\{<abc, e>, <bc, f>, <f, g>, \cdots\}$. Note that a simple edge *e.g.*, $<f, g>$ is also a hyperedge. In directed graphs, a path is defined as a sequence of edges. In directed hypergraphs, a hyperpath is defined as a tree of hyperedges [3].

Definition 2: A hyperpath from a source set $X \subseteq V$ to a target node y in V on a directed hypergraph $H=<V, E>$ if exists, is a tree $T_{X,y}$ that is recursively defined as follows. (1) If

$y \in X$ then $T_{X,y}$ is an empty tree. (2) If $y \notin X$, then $T_{X,y}$ is a tree where a hyperedge $<Z, y>$ is the root that has subtrees T_{X,z_i} for all z_i in Z.

Fig. 3 shows a hyperpath $T_{abc,h}$ from the set of nodes $\{a, b, c\}$ to node h in Fig. 2. We use a shorthand notation to express a set of nodes. For example, *abc* represents $\{a, b, c\}$. In the figure, the hyperedge $<eg, h>$ is the root. It has the left subtree for a hyperpath from *abc* to *e* and the right subtree for a hyperpath from *abc* to *g*. Note that a hyperpath from *abc* to *g* is represented by the subtree whose root node is $<f, g>$ in Fig. 3. In addition, a hyperpath can be expressed in the form of the induced hypergraph. In Fig. 2 the hypergraph we have after removing the node *d* is the induced hypergraph of the hyperpath $T_{abc,h}$.

In a directed graph, the set of all incoming edges (outgoing edges) of a node is sometimes called the backward star (forward star, respectively). Forward star and backward star can also be defined on directed hypergraphs [9].

Definition 3: The forward star of a node v, denoted by $fstar(v)$, is the set of hyperedges having the node v as one of source nodes. The backward star of a node v, denoted by $bstar(v)$, is the set of hyperedges whose target node is the node v.

A traversal on a directed hypergraph uses hyperedges, all of whose source nodes are already visited. A traversal algorithm satisfying this constraint is introduced in [9] as B-visit. Suppose we start from a set of nodes S. Initially, we insert all the nodes in the set S into the queue and mark them as visited. Then, take out one node v from the queue, and check each outgoing hyperedge h of v whether all the source nodes of h are visited. If all the source nodes are visited, we insert the target node of h into the queue and mark the target node as visited. The above steps are repeated until the queue becomes empty. This algorithm runs in $O(|H|)$, that is $O(|V|+|E|+\sum_{v \in V}|fstar(v)|)$ as explained in [3].

By using the definition of the hyperpath, we can define the reachable set from $X \subseteq V$ as follows.

Definition 4: The reachable set from $X \subseteq V$, denoted by $R(X)$, is the set of all nodes to which there exists a hyperpath from X, *i.e.*, $R(X) = \{y \in V \mid \exists T_{X,y}\}$.

A node y is reachable from X if $y \in R(X)$, and a set of nodes Y is reachable from X if $Y \subseteq R(X)$. The reachable set from X can be obtained by the hypergraph traversal starting from X.

3. RELATED WORK

A Petri Net, which is a bipartite graph consisting of places and transitions, can be given semantics similar to a directed hypergraph. For example, in certain applications we may make nodes and hyperedges in a directed hypergraph correspond to places and transitions, respectively in a Petri Net. Problems on Petri Net similar to the MinSS problem can be found in [6, 7, 11]. However, they focus on finding the degree of truth of each node when all the startable nodes have some truth value while we want to find the minimum subset of startable nodes in this paper. They also did not discuss how to handle cycles. Thus, their methods are not appropriate to solve the MinSS problem.

Backward chaining algorithms such as in [2] focus on checking whether the goal can be satisfied or not. This corresponds to the problem that finds a hyperpath or finds a

source set that can reach given targets.

For various optimization problems on directed hypergraphs, [1, 3] discuss the boundary between NP-hard and polynomial time solvable problems. The general framework to solve polynomial time solvable optimization problems is also suggested. But the MinSS problem is NP-hard and no approach has been suggested yet.

Carbonell *et al.* [5] and Cottrest *et al.* [8] also try to solve some minimal set problems. In [5], they focused on finding minimal hyperpaths in directed hypergraphs. For a given source set and a target node, their algorithm finds all the minimal hyperpaths where a minimal hyperpath is a hyperpath that is not a proper superset of any other hyperpaths. Note that our MSS index provides minimal source sets for any target node. In [8], they propose an algorithm that computes minimal source sets for a given target node. Though they consider various complex conditions that are important in practice, the basic computation structure of their algorithm is similar to that of the naive approach in this paper. Since the complexity of the problem is NP-Hard, their algorithm can only be used for problems with very small sizes.

4. MINSS PROBLEM

4.1 Problem Statement

The Minimum Source Set (MinSS) problem is an inverse problem of the reachable set problem.

Definition 5: Let $H=<V, E>$ be a directed hypergraph, and V_S be a set of startable nodes that is a subset of V. For a given target set of nodes $Y \subseteq V$, the Minimum Source Set (MinSS) of Y is the minimum cardinality source set $S \subseteq V_S$ that can reach every member of Y.

The MinSS problem is to find the smallest, *i.e.*, the minimum cardinality set of nodes S that can reach given target nodes. If there are more than one such sets, we find all of them. Here, only *startable* nodes can be members of S. A startable node is a node that can be a starting node, *i.e.*, a source node of leaf hyperedges in a hyperpath. A set of chemicals in Fig. 1 is an example of a set of startable nodes. We assume that a set of startable nodes are given by users in advance.

In Fig. 2, suppose that startable nodes are a, b, c, d, f. Then the MinSS of target node e is $\{a, b, c\}$. There are two MinSS's of target node g, *i.e.*, $\{d\}$ and $\{f\}$. The MinSS of target set $\{e, g, h\}$ is $\{a, b, c\}$.

Unlike the reachable set problem, the MinSS problem is NP-hard. NP-hardness of the MinSS problem can be proved by reducing the Set Cover problem to the MinSS problem in a manner similar to the proof of NP-hardness of the hyperarc minimization problem in [1].

Theorem 1: The MinSS problem is NP-hard.

***Proof*:** We will show that the MinSS problem can be reduced from the Set Cover problem in polynomial time.

Consider a universe $U=\{a_1, \ldots, a_n\}$ and a set of families $F=\{f_1, \ldots, f_m\}$, $f_i \subseteq U$ for

i=1, …, m. A Set Cover is the smallest subset of F that can cover all the elements in U. The Set Cover problem can be converted to the MinSS problem as follows. First, create nodes for all elements a_1, …, a_n in U and all elements f_1, …, f_m in F. Then for each f_i, add a hyperedge from node f_i to each node a_j that is a member of f_i. For example, if f_1={a_1, a_2}, add two hyperedges <f_1, a_1> and <f_1, a_2>. Finally, create a node t and add a hyperedge <U, t>. Then a solution of the Set Cover problem can be obtained by computing a MinSS in the hypergraph in Fig. 4 where a set of startable nodes is F and a target node is t.

Suppose that there is a solution for a Set Cover problem and let the solution be S = {f_{i_1}, f_{i_2}, …, f_{i_t}} where $i_j \in [1, m]$. If we select nodes in S as a solution of the corresponding MinSS problem then there must be paths to all nodes a_i's because all the a_n's are covered by S and the hyperedge <U, t> can be the root of the hyperpath $T_{S,h}$. Moreover, S has the minimum cardinality since S is the solution of the Set Cover problem.

Conversely, let S be a solution of the MinSS problem in the generated hypergraph. Then there must be a hyperpath from S to t. The root of the hyperpath must be the hyperedge <U, t> because there is only one hyperedge connected to the node t. By definition of the hyperpath, sub-hyperpaths to all a_i's must exist. Thus, all the a_i's can be covered by S and S has the minimum cardinality by definition of the MinSS problem.

The reduction can be done in polynomial time because it just creates n+m+1 nodes and $\sum|f_i|$+1 hyperedges. ❑

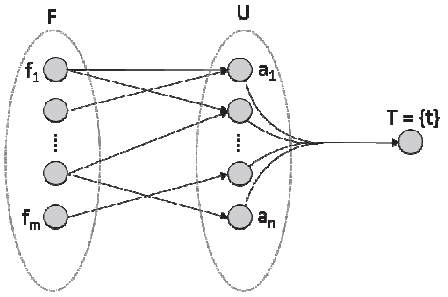

Fig. 4. A reduction of the Set Cover problem to the MinSS problem.

4.2 Finding MinSS: Naive Approach

As a naive approach, we first find all possible hyperpaths to a given target set T of nodes, and then select paths that have source sets whose cardinalities are smallest. Backward traversals can be used to find hyperpaths to the given target nodes. The backward traversal starts from a target set T, and visits nodes by following hyperedges incident to T backward. Let the visited nodes be v_1, …, v_k. We attempt to find a set of startable nodes from which there exists a hyperpath to each of v_1, …, v_k, and continue this process recursively. When the visited node is startable, we add the node into the result set, and then continue the backward traversal through an incoming hyperedge of the node. If the visited node is neither startable nor having any incoming hyperedges, then this backward is unsuccessful. When we visit a node having multiple incoming hyperedges, we need to try all the hyperedges to traverse. A completion of the above process provides a set S_T of startable nodes of hyperpaths to T. Then, we choose all the

minimum subsets of S_T that can reach to T, by forward traversing from all the subsets of S_T. We will show in Section 7 that this naive approach takes huge amounts of time even for problems with small sizes.

5. PROPOSED APPROACH

5.1 The MSS Index

A minimal source set, shortly MSS of a node v is a set of startable nodes that can reach the node v, without any redundant nodes, *i.e.*, any proper subset cannot reach to v. To quickly obtain the solution of the MinSS problem, we propose an index, called the MSS index that is a collection of the MSS's of all the nodes. Fig. 5 shows an example of an MSS index. $\{b, c\}$ is a minimal source set of node f while $\{a, b, c\}$ is not because a is redundant. We define $\{v\}$ be a minimal source set of v if v is a startable node.

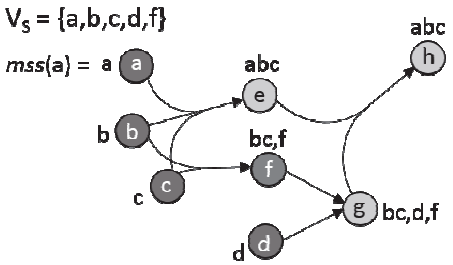

Fig. 5. An example of minimal source sets. *abc* is a shorthand notation of $\{a, b, c\}$. Startable nodes: a, b, c, d, f.

Definition 6: For a directed hypergraph $H=<V, E>$ and a set of startable nodes $V_S \subseteq V$, a minimal source set of a node v in V is a set $S \subseteq V_S$ that can reach v and any proper subset of S does not have a hyperpath to v.

Definition 7: The MSS of a node v, denoted by $mss(v)$, is the collection of all minimal source sets of v, *i.e.*, $mss(v)=\{S \mid S \subseteq V_S; \exists T_{S,v}$, and $\forall S' \subset S; S' \neq S$ and $\nexists T_{S',v}\}$

The MSS index represents $mss(v)$ for all v in V. With the MSS index, a solution of the MinSS problem for any target node t can be easily obtained by selecting sets with the minimum cardinality among all the minimal source sets in $mss(t)$. To describe the construction of the MSS index, we define the MSS combination operator, denoted by *, whose meaning is "combine two minimal source sets" as follows.

Definition 8: For distinct nodes u and v, the MSS combination operator * is defined as follows: $mss(u)*mss(v)=\{S_u \cup S_v \mid S_u \in mss(u)$ and $S_v \in mss(v)\}$.

Note that the MSS combination operator * has the semantics similar to that of the Cartesian product operator because we compute $S_u \cup S_v$ for every possible combination of S_u in $mss(u)$ and S_v in $mss(v)$. Note also that the MSS combination operator * is commutative and associative. Since the proof of this property is straightforward, we omit the proof.

In Fig. 5, the MSS of each node is listed beside the node. For example, the MSS of

node g is $\{bc, d, f\}$. Note that bc is a shorthand notation for $\{b, c\}$, so $\{bc, d, f\}$ actually implies the set $\{\{b, c\}, \{d\}, \{f\}\}$. The solution of the MinSS problem of the target node g is $\{d\}$ and $\{f\}$ because they have the minimum cardinality in $mss(g)$.

The solution of the MinSS problem of target set $T=\{e, f\}$ can be computed by combining $mss(e)$ and $mss(f)$ with the MSS combination operator. The MinSS of T is $\{abc\}$ that has the minimum cardinality among sets in the combined set $mss(e)*mss(f) = \{\{a, b, c\} \cup \{b, c\}, \{a, b, c\} \cup \{f\}\} = \{abc, abcf\}$. If a MinSS problem has n target nodes ($n \geq 2$), then n minimal source sets corresponding to n target nodes must be combined.

5.2 Building the MSS Index

The MSS index can be built with a vertex centric manner. Let us define the MSS of a hyperedge. Then the MSS of a node can be computed by choosing minimal sets in the union of the MSS's of its incoming hyperedges.

Definition 9: The MSS of a hyperedge e, denoted by $mss(e)$, whose source nodes are $v_1, \ldots, v_k$, is defined as follows: $mss(e)=\{S | S \in mss(v_1)*\ldots*mss(v_k)$ and S is minimal$\}$.

Since all $m_1 \cup m_2 \cup \ldots \cup m_k$ where $m_i \in mss(v_i)$ for $i=1, \ldots, k$ are in $mss(v_1)*\ldots*mss(v_k)$, any element in $mss(v_1)*\ldots*mss(v_k)$ must reach all the source nodes of hyperedge e while elements not in $mss(v_1)*\ldots*mss(v_k)$ cannot.

The MSS of a node v can be obtained by computing the union of $mss(e_i)$, $i=1, \ldots, d$, where $\{e_1, e_2, \ldots, e_d\}=bstar(v)$, and then selecting minimal sets. If v is a startable node, then $mss(v)$ must contain $\{v\}$ itself. Thus, we have

$$mss(v) = \begin{cases} \{S \mid S \in \{\{v\}\} \cup mss(e_1) \cup \cdots \cup mss(e_d), S \text{ is minimal}\} & \text{if } v \text{ is a startable node;} \\ \{S \mid S \in mss(e_1) \cup \cdots \cup mss(e_d), S \text{ is minimal}\} & \text{otherwise.} \end{cases} \quad (1)$$

The MSS index can be constructed by using the above Eq. (1). We start from V_S, and compute the MSS of each of visiting nodes in the forward direction. Once the MSS index is constructed, for any given target node, we can immediately find out its minimum source set. For a set of target nodes, we compute the minimum source set based on Definition 8.

Algorithm Building the MSS index

Input H: A directed hypergraph $H = \langle V, E \rangle$
V_S: Startable nodes $V_S \subseteq H.V$
Output $mss(v)$ $(\forall v \in H.V)$

```
Q ← empty priority queue
For v ∈ V_S do
    v.visited ← true
    mss(v) ← {{v}}
    Q.enqueue(v)
EndFor
While Q is not empty do
    v ← Q.dequeue()
    Step(v)
EndWhile
```

```
Function Step (v)
  For e = ⟨S, t⟩ ∈ fstar(v) do
    If all s ∈ S are visited then
      mss(t) ← minimal(mss(t) ∪ mss(e))
      If mss(t) is changed then
        Q.enqueue(t)
      EndIf
    EndIf
  EndFor
```

Fig. 6. MSS index construction algorithm.

5.2.1 Construction on acyclic directed hypergraphs

If a directed hypergraph H is acyclic, then all nodes are topologically ordered [9]. The traversal starts from startable nodes that have no incoming hyperedges. Before visiting a node v, all the source nodes of incoming hyperedges of node v have to be visited. Thus, $mss(v)$ for each node v can be calculated by Eq. (1) together with Definition 9. To visit nodes in topological order, we use a priority queue. Nodes in the priority queue are sorted in ascending order by the number of incoming hyperedges whose source nodes have not been completely visited yet. Fig. 6 shows the MSS index construction algorithm.

Consider the hypergraph in Fig. 7, for example. In the figure, the MinSS of node y is *abcdef*, and the MinSS of node t is *ab*. Nodes such as node v has more than one MSS's, *i.e.*, *cef* and *def*. Once an MSS index is constructed, the MinSS of any node can be easily obtained without any complex traversals.

Proposition 1: The worst-case time complexity of constructing MSS indices in an acyclic directed hypergraph $H=<V, E>$ is

$$O(|V_S|(|H|+|V|\log|V|+\alpha\sum_{v\in V}\sum_{e=<X,y>\in bstar(v)}|X|))$$

where $\alpha=\frac{k}{2}\cdot\binom{k}{k/2}$ and $k=|V_S|$. $\binom{k}{k/2}$ means "k choose $k/2$."

***Proof*:** Let α stand for the maximum size of $mss(v)$, *i.e.* for all $v\in V$, the maximum of sum of $|S|$ where $S\in mss(v)$. The unit of α is the number of nodes. Then for each startable node, the time cost of building MSS indices on an acyclic directed hypergraph H is $O(|H|+|V|\log|V|+\alpha\sum_{v\in V}\sum_{e=<X,y>\in bstar(v)}|X|)$. The traversal requires $O(|H|)$ and the priority queue requires $O(|V|\log|V|)$ to sort nodes. For each node v, the MSS of v is calculated maximally sum of $|X|$ times where X is a source set of an incoming hyperedge of v.

Now, we consider α. If a set S is contained in $mss(v)$, neither any subset of S nor any superset of S cannot be contained in $mss(v)$ because $mss(v)$ contains only minimal sets. Since $mss(v)$ contains some combinations of nodes in V_S, suppose that $mss(v)$ contains $\binom{k}{i}$ combinations of V_S for some $i\in[1, k]$. Then the $\binom{k}{j}$ combinations for $j>i$ or $j<i$ cannot be in $mss(v)$. In this reason, the maximum possible number of minimal source sets for each node and each hyperedge is $\binom{k}{k/2}$ and each set has size of $k/2$. Therefore, in the worst case, $\alpha=\frac{k}{2}\cdot\binom{k}{k/2}$. ❑

5.2.2 Construction on cyclic directed hypergraphs

Some nodes can be visited many times during the traversal. Thus we use a strategy that updates the previously computed MSS. If we visit a node v through a hyperedge $e=<S, v>$, then $mss(v)$ can be updated as follows:

$$mss(v)\leftarrow\{S\mid S\in mss(v)\cup mss(e)\text{ and }S\text{ is minimal}\}$$

The construction algorithm for cyclic directed hypergraphs uses the same priority queue as in the algorithm for acyclic directed hypergraphs to reduce the number of updates. Note that a topological order is not defined in a cyclic directed hypergraph. However, if all the nodes in the cycle are treated to have the same priority, a cyclic hypergraph can be handled with the same construction algorithm for acyclic hypergraphs. That is, the MSS index construction algorithm in Fig. 6 works for both acyclic and cyclic hypergraphs. Note that, as shown in the algorithm, a node is reentered into the queue only if its MSS is changed. In the cyclic directed hypergraph, as in the acyclic case we start from startable nodes with no incoming hyperedge. If there is no such node, we start from startable nodes whose numbers of incoming hyperedges are minimum.

Proposition 2: The worst-case time complexity of constructing MSS indices in a cyclic directed hypergraph $H=<V, E>$ is

$$O((1+\beta)\,|V_S|\,(|H|+|V|\log|V|+\alpha\sum_{v\in V}\sum_{e=<X,y>\in bstar(v)}|X|))$$

where β is $|V|!$, $\alpha=\frac{k}{2}\cdot\binom{k}{k/2}$, and $k=|V_S|$.

***Proof*:** Let β stand for the number of cycles in a directed hypergraph H. The cost of each cycle in the worst case is the same as the cost for constructing MSS indices in an acyclic hypergraph. The cost for acyclic directed hypergraphs becomes a special case of $\beta = 0$. In the worst case, the number of cycles in $H=<V, E>$, *i.e.* β can increase up to $O(|V|!)$. ❑

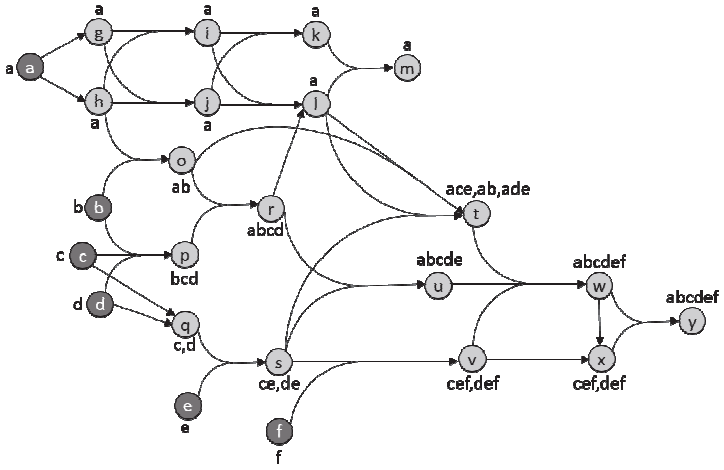

Fig. 7. An example of the MSS index on a directed hypergraph; The MinSS of a node can be easily obtained from the MSS index without any complex traversals.

Depending on the set of startable nodes, certain set of nodes may or may not be reachable from the set of startable nodes. This also affects the execution time. Note that actual execution time is affected by the number of nodes, hyperedges, startable nodes, and many-to-one hyperedges.

5.3 Handling of Insertions and Deletions

Suppose a new node v is inserted. If the node v has no outgoing hyperedge, then it can be simply added and only requires computing the MSS of the inserted node. This computation can be done by Eq. (1). But if the node has outgoing hyperedges, the MSS's of target nodes of hyperedges in *fstar*(v) must be updated. This update is propagated until the MSS index does not change. In the worst case, entire nodes in the hypergraph may need to be accessed during the propagation, but in practice, the propagation will be local.

Deleting a node entails, in addition to deletion of the node itself, deletion of hyperedges incident on the node. When deleting a hyperedge, the MSS of the target node of the hyperedge needs to be modified. When deleting a node together with its incident hyperedges, the MSS index is updated by the deletions of the hyperedges as well as the deletion of the node. Update propagation is handled in a manner similar to that in node insertion.

5.4 The Reduction of the MSS Index

The size of the MSS index can be $\frac{k}{2}\cdot\binom{k}{k/2}$ in the worst case as discussed in Section 5.2. The MSS computation may require too much time and space for a large and complex hypergraph. To reduce such time and space we propose a reduction strategy that replaces a certain set of minimal source sets with a symbol, called a source set proxy (shortly, proxy).

If the size of $mss(v)$ for a certain node v exceeds a predefined threshold during the MSS index construction, the entire sets in $mss(v)$ are replaced by a symbol "$P(v)$" which we call the proxy of $mss(v)$, *i.e.*, $mss(v)=\{P(v)\}$. Conceptually a proxy $P(v)$ is a virtual source node representing source sets that reach the node v. Thus, the MSS index manages values of $P(v)$ as well as values of $mss(v)$.

A proxy $P(v)$ can be a member of $mss(w)$ for some node w which is in turn replaced by the proxy $P(w)$ if the size of $mss(w)$ exceeds the threshold. $P(w)$ can also be a member of $mss(x)$ that is replaced by $P(x)$, and so on. By utilizing the concept of the proxy, we can significantly reduce the space requirement of the MSS index.

Definition 10: Let P denote the set of all proxies. Then, $mss(v)$ can be defined as follows: $mss(v)=\{S \mid S\subseteq V_S\cup P$ and $\exists T_{S,v}$ and $\forall S'\subset S: \nexists T_{S',v}\}$

When computing minimum cardinality source sets (*i.e.*, a solution of the MinSS problem) for a given target node by using the MSS index, every proxy involved in this computation is expanded to its corresponding minimal source sets.

Let us assume $mss(v)$ contains a proxy $P(u)$. Then $mss(v)$ can be separated as a set of sets containing a proxy $P(u)$ and a set of sets that do not contain $P(u)$. Let α be a set of sets containing $P(u)$ and β denote the other set. Let α' be a set obtained by removing $P(u)$ from all sets in α. Then $mss(v)$ can be reconstructed with sets α' and β recursively as follows:

$$mss(v) = \{S \mid S\in(mss(u)*\alpha')\cup\beta \text{ and } S \text{ is minimal}\}.$$

The same procedure is applied repeatedly until no proxy remains. That is, if $mss(u)$ contains a proxy symbol $P(k)$, we can remove $P(k)$ by the following equation: $mss(u)=$

$\{S|S \in (mss(k)^* \ \alpha'') \cup \beta'$ and S is minimal$\}$. Here, α'' and β' are defined similarly as described for α' and β.

6. THE MINSS WITH TARGETABLE NODES

As a simple extension of the MinSS problem, we may provide a restriction on the target set. Targetable nodes are nodes that can be a target of the MinSS problem. For example, a set of disease can be a set of targetable nodes in Fig. 1.

The Minimum Source Set problem (MinSS) with targetable nodes on a directed hypergraph $H=<V, E>$ is the problem of finding a minimum cardinality source set S among subsets of *startable* nodes V_S that can reach the given set of target nodes $T \subseteq V_T \subseteq V$, where V_T is a set of *targetable* nodes.

Let $H'=<V', E'>$ be the largest sub-hypergraph of hypergraph $H=<V, E>$, which is obtained by traversing the hypergraph H in backward directions from a set of targetable nodes T. For the sub-hypergraph H', startable nodes are reduced to $V'_S = V_S \cap V'$. The MSS index construction algorithm for this case is shown in Fig. 8.

Algorithm The MinSS with Targetable Nodes

Input H: A directed hypergraph $H = \langle V, E \rangle$
V_S: Startable nodes $V_S \subseteq H.V$
T: Targetable nodes $T = \{t_1, ..., t_k\}$ where $t_i \in H.V$
Output MSS Indices

```
H' = ⟨V', E'⟩ ← BackwardTraversal(T)
V'_S ← V' ∩ V_S
Build the MSS index for (H', V'_S)
```

```
Function BackwardTraversal (T)
    V', E' ← {}
    Q ← empty queue
    For v ∈ T do
        Q.enqueue(v)
        V' ← V' ∪ {v}
    EndFor
    While Q is not empty do
        v ← Q.dequeue()
        For e = ⟨S, v⟩ ∈ bstar(v) do
            E' ← E' ∪ {e}
            For u ∈ S do
                If u ∉ V' then
                    Q.enqueue(u)
                    V' ← V' ∪ {u}
                EndIf
            EndFor
        EndFor
    EndWhile
    Return ⟨V', E'⟩
```

Fig. 8. Construction on the MSS index that has restriction on the target set.

7. EXPERIMENT

7.1 Implementation

We have implemented a hypergraph store on the top of Neo4j [10]. Since only simple edges are supported (*i.e.*, hyperedges are not allowed) in Neo4j, we extended Neo4j

to process hypergraphs by using special purpose nodes called *h*-nodes which play the role of hyperedges. For a hyperedge with source nodes n_1, …, n_k and a target node t, we create an *h*-node where it has incoming edges from all the nodes n_1, …, n_k and has an outgoing edge to the node t. We can visit the target node t, only after we visit this *h*-node through all the source nodes n_1, …, n_k, *i.e.*, all the source nodes have been visited. To check this condition, each *h*-node has a counter variable for counting the number of visited source nodes. A similar idea is used in the FD-graph [4]. Computed values of MSS's and proxies are stored as properties of nodes.

7.2 Dataset

7.2.1 Synthetic dataset

In this experiment, we use synthetic datasets and a real dataset. Synthetic datasets are randomly generated directed hypergraphs of n nodes and m hyperedges. A hyperedge is generated by randomly selecting one to three nodes as the source set and one node as the target node. This is because in the biological network most hyperedges consist of less than four source nodes. In this experiment, we use the same number for n and m, and set 10% of nodes startable and 10% of nodes targetable. The number of many-to-one hyperedges is about 60% of whole hyperedges.

7.2.2 Real dataset

The real datasets used in this paper are subsets of CODA (COntext specific Directed Associations) dataset in [12]. CODA contains molecular interactions (intracellular interactions and intercellular interactions) and associations between molecules and overmolecules. It contains various interactions such as drug-gene, gene-gene, gene-metabolite, and gene-disease. We used two subsets of CODA data set. The number of nodes, hyperedges, startable nodes, and many-to-one hyperedges in the first data set are 67750, 438570, 1107, 274, respectively. Those in the second data set are 73014, 447319, 3107, 8791, respectively. The type of startable nodes in the first data set is drug. And that in the second data set is drug or metabolite.

7.3 Experimental Setup

Experiments are performed in a Windows 7 machine with Intel i5-4430 3.00GHz CPU, 8GB main memory and Samsung SSD 840 PRO.

7.4 Results of Experiments

As far as we are aware of, there is no other work, except for the naive approach, that can be compared with our proposed method on the MinSS problem in this paper. We show in Table 1 the execution times of the naive approach described in Section 4.2 for solutions of the MinSS problem with small synthetic data sets. In the table, the naive approach required 3133.7 seconds even when the hypergraph has only 140 nodes. It returns out that the naive approach cannot be used even for problems with small sizes.

The MSS index can handle MinSS problems with much larger sizes. We tested the

MSS index on hypergraphs that have synthetically generated 10000, 20000, 30000, 40000, and 50000 nodes and the same number of edges. The results are shown in Table 2 and Fig. 9. While the time for constructing an MSS index (*i.e.*, build time) increases steeply with the increase of the number of nodes, solutions of the MinSS problem are quickly obtained for any arbitrarily chosen target node once the index is constructed. Note that the size of the MSS index grows slowly because of the reduction strategy discussed in Section 5.

If the set of targetable nodes is restricted to only a certain subset of V, the MSS index construction can be faster. Table 3 shows build times for this case. We selected 10% of nodes as targetable nodes.

We also tested the MSS index with the subsets of CODA dataset. The first data set took 15.08 hours to construct the MSS index. The average access time to the MSS index, *i.e.*, the average time to obtain the solution of the MinSS problem for an arbitrarily chosen target node is 180.07 ms. The second data set took 58.87 hours to construct the MSS index. We also tested effects of deletions. When a randomly chosen hyperedge is deleted in the second data set, the average number of nodes whose MSS indices are changed is 2,750. We computed the average out of 1,000 repeated experiments. If we insert a deleted hyperedge, then the number of nodes whose MSS indices are changed is the same as the one in deletion. Thus, we omit experiments for insertions.

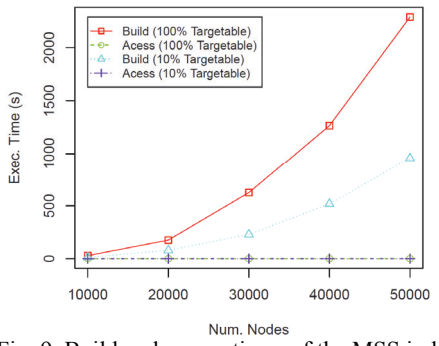

Fig. 9. Build and access times of the MSS index on large hypergraphs.

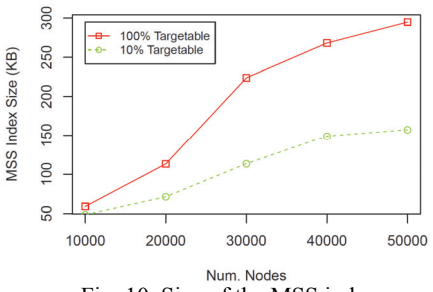

Fig. 10. Size of the MSS index.

Table 1. Execution time of the naive algorithm on small synthetic hypergraphs.

Number of nodes	60	80	100	120	140
Time (s)	1.126	3.052	11.794	432.23	3133.7

Table 2. MSS build time, MinSS computing time (*i.e.*, access time) and size of the MSS index on large synthetic hypergraphs.

Number of nodes	10000	20000	30000	40000	50000
Build time (s)	27.99	175.55	626.16	1265.8	2288.1
Access time (ms)	0.04	0.02	0.04	0.01	0.01
MSS index (KB)	59.07	113.09	223.73	268.18	294.83

Table 3. Experimental results of the MSS index with targetable nodes on large synthetic hypergraphs.

Number of nodes	10000	20000	30000	40000	50000
Build time (s)	12.870	78.467	229.05	518.44	957.07
Access time (ms)	0.038	0.077	0.038	0.077	0.192
MSS index (KB)	48.528	71.328	113.43	148.91	157.29

8. CONCLUSION

The MinSS problem, defined in this paper, is to find the smallest set of startable nodes that can reach given target nodes in the directed hypergraphs. The MinSS problem has many applications in practice such as finding minimum combinations of chemicals that can be effective for a certain disease in the complex biological network. The MinSS problem is NP-hard, so cannot be solved in reasonable amounts of time even for problems with small sizes. We have proposed a precomputed approach, *i.e.*, construction of the MSS index that has all the minimal source sets of each node in advance. Once an MSS index is constructed, for any arbitrarily chosen target node, the minimum cardinality source set among all the minimal source sets of that target node can be found in a very short time. We have also proposed a strategy to reduce the size of the MSS index. Though we have focused on hyperedges with a simple target node, the results of this paper are easily extended to the case where a hyperedge has multiple target nodes. For constructing an MSS index, a hyperedge with n target nodes can be implemented by applying a hyperedge with single target node n times.

We have shown through performance experiments that the MSS index can be constructed in acceptable amount of time by using a single PC for problems with reasonably large sizes such as a hypergraph with about 120K nodes and 140K hyperedges. However, our current approach is difficult to be applied to huge hypergraphs with tens of millions nodes and hyperedges. Specialized graph computation engines that properly utilize parallel computing may help. Since, however, the problem itself is NP-Hard, large amount of computation time may not be avoidable for problems with huge sizes.

ACKNOWLEGEMENT

This work was supported by the Bio-Synergy Research Project (NRF-2013M3A9C-4078137) of the MSIP (Ministry of Science, ICT and Future Planning), Korea, through the NRF, and by the MSIP, Korea under the ITRC support program (IITP-2016-H8501-16-1013) supervised by the IITP.

REFERENCES

1. G. Ausiello, R. Giaccio, G. F. Italiano, and U. Nanni, "Optimal traversal of directed hypergraphs," Technical Report No. TR-92-073, International Computer Science Institute in Berkeley, CA, 1992.
2. G. Ausiello and G. F. Italiano, "On-line algorithms for polynomially solvable satisa-

bility problems," *The Journal of Logic Programming*, Vol. 10, 1991, pp. 69-90.
3. G. Ausiello, G. F. Italiano, L. Laura, U. Nanni, and F. Sarracco, "Classification and traversal algorithmic techniques for optimization problems on directed hyperpaths," Technical Reports 2(18), Department of Computer and System Sciences Antonio Ruberti, 2010.
4. G. Ausiello, U. Nanni, and G. F. Italiano, "Dynamic maintenance of directed hypergraphs," *Theoretical Computer Science*, Vol. 72, 1990, pp. 97-117.
5. P. Carbonell, D. Fichera, S. B. Pandit, and J. L. Faulon, "Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms," *BMC Systems Biology*, Vol. 6, 2012, DOI 10.1186/1752-0509-6-10, http://www.biomedcentral.com/1752-0509/6/10.
6. S. M. Chen, "Fuzzy backward reasoning using fuzzy petri nets," *Systems*, *Man*, *and Cybernetics*, Part B, *IEEE Transactions on Cybernetics*, Vol. 30, 2000, pp. 846-856.
7. S. M. Chen, J. S. Ke, and J. F. Chang, "Knowledge representation using fuzzy petri nets," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, 1990, pp. 311-319.
8. L. Cottret, P. V. Milreu, V. Acuna, A. Marchetti-Spaccamela, F. V. Martinez, M. F. Sagot, and L. Stougie, "Enumerating precursor sets of target metabolites in a metabolic network, " *Lecture Notes in Computer Science*, LNBI 5251, 2008, pp. 233-244.
9. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete Applied Mathematics*, Vol. 42, 1993, pp. 177-201.
10. Neo Technology, I., Neo4j http://neo4j.com, 2015.
11. R. Yang, P. A. Heng, and K. S. Leung, "Backward reasoning on rule-based systems modeled by fuzzy petri nets through backward tree," *Computational Intelligence for Modelling and Prediction*, Springer, Germany, 2005, pp. 61-71.
12. S. Yoon, J. Jung, H. Yu, M. Kwon, S. Choo, K. Park, D. Jang, S. Kim, and D. Lee, "Context-based resolution of semantic conflicts in biological pathways," *BMC Medical Informatics and Decision Making*, Vol. 15 (Suppl 1), S3, 2015.

Hyun Jun Kim received his B.S. and M.S. degrees in Computer Science from KAIST, Daejeon, Korea in 2010 and 2015, respectively. His research interests include database systems, information retrieval and data mining.

Kwang Hee Lee received his B.S. degree in Mathematical Science from KAIST, Daejeon, Korea in 2014. He is currently pursuing the M.S. degree in the Department of Computer Science, KAIST, Daejeon, Korea. His research interests include hypergraph theory, database systems and graph data mining.

Myoung Ho Kim received his B.S. and M.S. degrees in Computer Engineering from Seoul National University, Seoul, Korea in 1982 and 1984, respectively, and received his Ph.D. degree in Computer Science from Michigan State University, East Lansing, MI, in 1989. He joined the faculty of the Department of Computer Science at KAIST, Daejeon, Korea in 1989 where currently he is a Professor. His research interests include database systems, information retrieval, data mining, multimedia data, data stream, sensor networks, mobile computing, OLAP, XML, workflow and distributed processing.