Scientia Iranica D (2018) 25(3), 1441–1455

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
http://scientiairanica.sharif.edu

Making problem: A new approach to reachability assurance in digraphs

M. Valizadeh[a], M.H. Tadayon[a,*], and A. Bagheri[b]

a. *Iran Telecommunication Research Center (ITRC), Tehran, P.O. Box 14155-3961, Iran.*
b. *Faculty of Computer Engineering, Amirkabir University of Technology, Tehran, Iran.*

Received 31 December 2016; received in revised form 7 October 2017; accepted 13 January 2018

KEYWORDS
Marking problem;
Reachability assurance;
Pathfinding;
Software testing.

Abstract. Considering G as a weighted digraph, and s and t as two vertices of G, the Reachability Assurance (RA) problem is how to label the edges of G such that every path starting at s finally reaches t and the sum of the weights of the labeled edges, called the RA cost, is minimal. The common approach to the RA problem is pathfinding, in which a path is sought from s to t and, then, the edges of the path are labeled. This paper introduces a new approach, the Marking Problem (MP), to the RA problem. Compared to the common pathfinding approach, the proposed MP approach has a lower RA cost. It is shown that the MP is NP-complete, even when the underlying digraph is an unweighted Directed Acyclic Graph (DAG) or a weighted DAG with an out-degree of two. An appropriate heuristic algorithm to solve the MP in polynomial time is provided. To mitigate the RA problem as a serious challenge in this area, application of the MP in software testing is also presented. By evaluating the datasets from various program flow graphs, it is shown that the MP is superior to the pathfinding in the context of test case generation.

1. Introduction

The goal of the reachability query is to determine whether or not it is possible to reach a target vertex from a source vertex in a given digraph. The reachability query problem has been extensively discussed in the literature [1]. When the size of the underlying digraph is small, the reachability query can be easily answered using primitive algorithms such as depth-first-search or transitive-closure. However, if the underlying digraph is very large, primitive approaches are not efficient. Most of existing reachability query approaches belong to either transitive closure compression [2-4] or online search [5-7] categories.

In this paper, it is assumed that the target vertex is reachable from the start vertex and is merely intended to determine how to assure reaching the target from the source. In this respect, G is considered to be a weighted digraph, and s and t are two vertices of G. The Reachability Assurance (RA) problem is meant to address how to label the edges of G such that every path starting at s finally reaches t and the sum of the weights of the labeled edges, called the RA cost or simply the reachability cost, is minimal. Various practical problems can be reduced to the RA problem. For instance, in the context of graph-based test case generation, the main problem is to generate test cases in order to cover the vertices or edges of a given digraph [8].

The common solution to the RA problem is pathfinding, in which a path p is sought from s to t

*. *Corresponding author. Tel.: +98 84977629*
E-mail addresses: valizadeh80@gmail.com (M. Valizadeh); tadayon@itrc.ac.ir (M.H. Tadayon); ar_bagheri@aut.ac.ir (A.R. Bagheri).

doi: 10.24200/sci.2018.20104

and, then, every edge of p is labeled as T, which implies that the edge should be followed [9,10]. Starting at s, we should pass through the labeled outgoing edge of s (e_i) and, then, to the labeled outgoing edge of the head of e_i and so on, until we reach t. The lower bound of the reachability cost of this solution is the shortest path weight. Some techniques use the pathfinding approach to solving the RA problem. For example, in the software-testing context, a symbolic execution technique uses a path to assure reaching a vertex of a given digraph [11]. Although pathfinding is efficient, it is generally not effective, especially when the size of G or the weight of the edges of G increases. The symbolic execution technique suffers from the problems of path explosion and path complex constraints because of the ineffectiveness of the pathfinding approach [11,12].

Pathfinding carries out total labeling to assure that the target vertex is reached, implying that it provides full information about reaching the target from the source. In order to decrease the reachability cost, labeling should be done as infrequently as possible. Because of this, the proposed solution to the RA problem is to label arbitrary edges (and not necessarily consecutive edges) of G. Moreover, it is possible to use two labels T and F on the edges of G. Starting from s, when reaching a vertex v_i of G, if an outgoing edge of v_i is labeled with T, then we must pass through it. In contrast, if an outgoing edge of v_i is labeled with F, then we must not pass through it. If we reach a vertex where none of the outgoing edges is labeled, we can optionally pass through any of those edges. The proposed solution to the RA problem is called the Marking Problem (MP) approach.

Example 1. Figure 1 represents flow graph, G, of a computer program with start and final vertices, v_1 and v_{10}, respectively. An edge of the flow graph denotes a logical expression (le). For instance, if x and y are the input variables of the program, then $le(e_1) = (x > y)$ and $le(e_2) = (x \leq y)$. Suppose that this study intends to assure reaching vertex v_8. The pathfinding approach uses path p from start vertex v_1 to target vertex v_8. In this approach, the logical expression of every edge of path p must be satisfied (evaluated as $True$). Because the length of the shortest path from v_1 to v_8 is 3, in order to assure reaching v_8 by a path, it is required to satisfy 3 logical expressions, e.g. the logical expression of every edge of the shortest path $e_1e_3e_9$ or $e_1e_4e_{10}$. Hence, the reachability cost in the pathfinding approach is 3. Since G is the flow graph of a computer program, execution of the program with any input leads to traversal of G starting at v_1 and ending at v_{10}. The MP approach assures reaching v_8 from v_1 by satisfying the logical expression of only edge e_9 ($e_9 = T$). The second solution of the MP approach is to unsatisfy (evaluate as $False$) the logical expression of only edge e_{15} ($e_{15} = F$). Thus, the reachability cost in the MP approach is 1. Solution $e_{15} = F$ or $e_9 = T$ means that, in order to assure reaching v_8 from v_1, edge e_{15} (F-marked edge or every sibling edge of the T-marked edge) should be removed from G. To verify this solution, it is enough to note that, by the removal of e_{15} from G, every path starting at v_1 finally reaches v_8. This simple example shows why the MP approach is superior to the pathfinding approach.

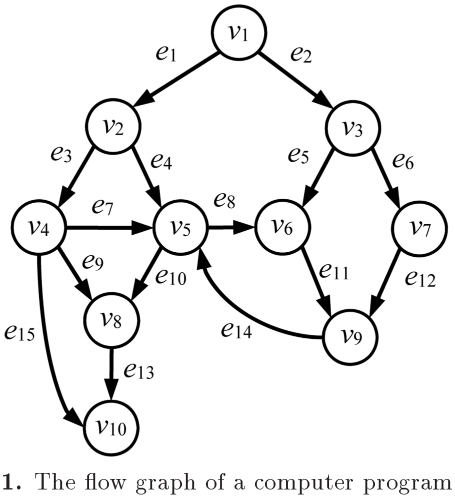

Figure 1. The flow graph of a computer program with start and final vertices v_1 and v_{10}, respectively.

Reachability assurance is a serious challenge to software testing [8]. Different methods could be used to design test cases for a computer program to detect the faults. Node coverage is one method which states that every vertex of the flow graph of the underlying program should be reached [8]. To achieve this goal, input data (test data) should be provided to the program in which every statement of the program is reached at least once. Let G be the flow graph of a given program and s be the start vertex of G. To extract such test data using the pathfinding approach, a path from s to each vertex of G should be found and, then, the labels (Boolean expressions) of all the edges of the path be satisfied. If G is a small digraph, this may not be difficult, but if G is large enough, this goal can become very hard to achieve, implying that it might not be solvable using the current SAT solvers [11]. Moreover, the labels of the edges of G might be dependent on each other, complicating the test case extraction problem.

The results of the benchmarks performed on thousands of program flow graphs show that the reachability cost of the pathfinding approach is 3.5 times greater than that of the MP approach.

This article is structured as follows. Section 2 presents the required notation and terminology. Section 3 provides a formal definition of the marking problem and its basic properties. Section 4 discusses the computational complexity of the marking problem and presents a heuristic algorithm to solve it. Section 5 compares the marking problem and pathfinding approaches by evaluating them on datasets from various program flow graphs. Section 6 concludes the research findings and proposes future work.

2. Preliminaries and notation

To move further, $G = (V, E)$ is considered here to be a digraph, and the vertex and edge sets of G are $V(G)$ and $E(G)$, respectively. A path in a digraph is a sequence of vertices such that, from each vertex, there is an edge to the next vertex in the sequence. A simple path is one in which all vertices are distinct. The term SP denotes the shortest path. If the first and last vertices of a path are the same, it is called a cycle. The set of reachable vertices from vertex v is denoted as reach(v). The set of outgoing edges of a vertex v is denoted as $oe(v)$.

It is supposed here that $e = (v_i, v_j)$ is a an edge of G, and v_i and v_j are the tail and head of edge e, respectively. E' is a subset of E, and $Tail(E')$ and $head(E')$ are the sets containing the tail and head of every edge of E'. H is a subgraph of G. The removal of subgraph H from G is denoted as $(G-H)$. Edge e_i is an outgoing edge of H if $tail(e_i) \in H$ and $head(e_i) \in H$. Edges e_1 and e_2 are said to be siblings if their tails are the same. The out-degree of a vertex v of G is denoted as $od(v)$, and the out-degree of G is the maximum out-degree of the vertices of G.

Digraph G is said to be a binary DAG if G has no cycle, and the out-degree of G is two. A subgraph H of a graph G is said to be induced, provided that, for any pair of vertices v_i and v_j of H, (v_i, v_j) is an edge of H if and only if (v_i, v_j) is an edge of G. If the vertex set of H is the subset S of $V(G)$, then H can be written as $G[S]$ and is said to be induced by S. A Flow Graph (FG) is a triple (V, E, s) where (V, E) is a digraph, $s \in V$ is the unique start vertex of the digraph, and there is a path from s to each vertex of G [13]. If $G = (V, E)$ is a digraph and $v_i \in V$, then a flow graph can be formed with start vertex v_i by the removal of any vertex of G (and its adjacent edges) that is not reachable from v_i. In this paper, function $FG(G, v_i)$ is used for this purpose. Thus, $FG(G, v_i) = (V', E', v_i) = G[V']$ s.t. $V' = \{v \in V | v \in \text{reach}(v_i)\}$. $G = (V, E, s)$ is a flow graph and v_i and v_j are two vertices of G. It can be said that v_i dominates v_j in G if every path from s to v_j contains v_i [14]. Edge $e = (v_i, v_j)$ is a back edge if every path from s to v_i goes through v_j; thus, v_j dominates v_i [15]. A flow graph is said to be reducible if the removal of its back edges leads to an acyclic digraph where each vertex can be reached from s.

3. Marking problem

Definition 1 (Marking problem). *Let $G = (V, E)$ be a digraph with non-negative edge weights and $v_i, v_j \in V$. The MP describes how to assign marks T and F to some of the edges of G, such that every path starting at source v_i will reach target v_j. When the digraph is traversed and a vertex v_k is visited, these marks have the following interpretation:*

1. *If some outgoing edges of v_k are marked with T, then we must pass through one of these edges;*
2. *If some outgoing edges of v_k are marked with F, then these edges must not be chosen;*
3. *If some outgoing edges of v_k are not marked, then any of these edges may be chosen. The optimization problem consists of minimizing the total weight of the marked edges. The marking problem is denoted by the triple (G, v_i, v_j). The sum of the weights of the marked edges is called the reachability cost.*

Observation 1. *Although two outgoing edges of a vertex can be marked with T, such marking is not minimal, implying that it cannot be an optimal solution to the marking problem.*□

Observation 2. *Marking problem $MP = (G, v_i, v_j)$ has a solution if and only if v_j is reachable from v_i.*□

Observation 3. *Let $MP = (G, v_i, v_j)$ be an instance of the marking problem and v_k be a vertex of G from which v_j is not reachable. In an optimal solution to the MP, none of the outgoing edges of v_k is marked. In fact, the outgoing edges of v_k can be removed with no effect on the solution to the MP.*□

A solution to marking problem MP $= (G, v_i, v_j)$ is a partial function from domain $E(G)$ to co-domain $\{T, F\}$. In the context of a pure (non-labeled) digraph, marking edge e with F means the removal of the edge, and marking edge e with T means the removal of every sibling edge of e. If G denotes the flow graph of a computer program, marking edge e with T/F is equivalent to making $TRUE/FALSE$ the label (logical expression) of e. In flow graphs, the terms "making $TRUE/FALSE$ label of an edge" and "marking an edge with T/F" are interchangeable.

Observation 4. *Let $G = (V, E)$ be a digraph and $v_i, v_j \in V$. If $FG(G, v_i)$ is the function converting G to flow graph G' with start vertex v_i, $MP_1 = (G, v_i, v_j)$ and $MP_2 = (G', v_i, v_j)$, and then G' is a reduced subgraph of G; so, MP_1 is reducible to MP_2.*

Note that when digraph $G = (V, E)$ is converted into a flow graph with start vertex v_i, only those vertices of G and their adjacent edges called E' which are not reachable from v_i are removed. Moreover, because any edge of E' is not reachable from v_i, there is no need to mark that edge.

Observation 5. *Let $G = (V, E, s)$ be a flow graph and MP $= (G, s, v_j)$ be an instance of the marking problem. In an optimal solution to the MP, the outgoing edges of a vertex of G cannot be marked with both marks T and F.*

Lemma 1. *Let $G = (V, E, s)$ be a flow graph and MP $= (G, s, v_j)$ be an instance of the marking problem. In an optimal solution to the MP, all outgoing edges of a vertex of G cannot be marked with F.*

Proof. Suppose that all outgoing edges of vertex v_k of G are marked with F. Thus, there will be a path from s to v_k which does not reach v_j, hence contradicting Definition 1.□

Lemma 2. *Let $G = (V, E, s)$ be an unweighted flow graph and MP $= (G, s, v_j)$ be an instance of the marking problem. In an optimal solution to the MP, for each vertex v_k of G, at most one of the outgoing edges of v_k can be marked with F.*

Proof. Let two outgoing edges of v_k, called e_1 and e_2, be marked with F. By Lemma 1, v_k has another outgoing edge called e_3 which is not marked with F. Moreover, by Observation 5, e_3 is not marked with T. Thus, edge e_3 has no mark. Through similar reasoning, every outgoing edge of v_k, except e_1 and e_2, is not marked. By Definition 1, any unmarked outgoing edge of v_k can be chosen to reach the target. Hence, instead of marking edges e_1 and e_2 with F, only one of the unmarked outgoing edges of v_k, such as e_3, can be marked with T, which implies that the marking is not minimal, which is a contradiction. As G is unweighted, the cost of marking any edge is 1.□

Lemma 3. *Let $G = (V, E, s)$ be an unweighted flow graph and MP $= (G, s, v_j)$ be an instance of the marking problem. In this case, the MP has a solution such that every marked edge is T-marked and, at most, one of outgoing edges of a vertex is marked.*

Proof. Let the total function $f : E_1 \to \{T, F\}$ s.t. $E_1 \subset E$ be a solution to the marking problem MP $= (G, s, v_j)$. By Observation 5, for each vertex v of G, the outgoing edges of v cannot be marked with both marks T and F. By Lemma 2, for each vertex v of G, at most one of the outgoing edges of v can be marked with F. By Lemma 1, if a vertex of G has only one outgoing edge, that edge cannot be marked with F. Thus, for each edge e_1 of E_1, if the mark of e_1 is F, then (e_2, T) can be substituted for (e_1, F) such that e_2 is an unmarked sibling edge of e_1.□

4. Solving marking problem

4.1. Computational complexity of marking problem

The computational complexity of the marking problem can be studied in both unweighted and weighted digraphs. Moreover, the complexity of the marking problem can be considered in three practical cases including general digraphs, acyclic digraphs (DAGs), and binary DAGs. MPII and MPI denote the decision versions of the marking problem in the weighted and unweighted digraphs, respectively. Decision problems MPI, MPII, and HSD (Hitting Set) are shown in Tables 1 to 3. The hitting set problem will be used to prove the NP-hardness of the marking problem.

Theorem 1. *If the underlying digraph is a weighted DAG, then the marking problem is NP-complete.*

Proof. Let G be a weighted DAG, s and t be two vertices of G, and MPII $= (G, s, t)$ be a decision problem of the marking problem. Firstly, it is shown

Table 1. Decision version of marking problem in weighted digraphs (MPII).

Input:	A digraph $G = (V, E)$ with non-negative edge weights, vertices s and t of G, and a real value w_1.
Parameter:	w_1
Question:	Is it possible to mark some edges of G with $\{T, F\}$, where every path starting at s will reach t and the sum of the weights of the marked edges is at most w_1?

Table 2. Decision version of marking problem in unweighted digraphs (MPI).

Input:	An unweighted digraph $G = (V, E)$, vertices s and t of G, and an integer k_1.
Parameter:	k_1
Question:	Is it possible to mark some edges of G with $\{T, F\}$, such that every path starting at s will reach t and the number of the marked edges is at most k_1?

Table 3. Decision version of hitting set problem (HSD).

Input: A ground set $\{a_1, a_2, ..., a_m\}$, a collection of n subsets s_i of that ground set and an integer k_1.
Parameter: k_1
Question: Does there exist a subset A of the ground set, such that $|A| \leq k_1$ and for each $i = 1, ..., n$, $s_i \cap A \neq \phi$?

that MPII is NP. A given solution to MPII can be verified in polynomial time as follows: Suppose that $f_1 : E_1 \rightarrow \{T, F\}$ is a given solution to MPII which needs to be verified such that $f_1 = f_2 \cup f_3, f_2 : E \rightarrow \{F\}$, $f_3 : E_3 \rightarrow \{T\}$, and E_1, E_2, E_3 are the subsets of E. Remove every element (edge) of E_2 from G as well as every sibling edge of any edge of E_3 and denote the new digraph as G'. For each vertex v of G', check whether v is reachable from s and does not reach t. If such a vertex, v, does not exist and the sum of the weights of all elements (edges) of E_1 is less than or equal to w_1, then f_1 is a solution to MPII; otherwise, it is not. Clearly, this check can be performed in polynomial time. Now, it can be demonstrated that MPII is NP-hard. The decision version of the hitting-set problem is reduced, which is one of 21 classic NP-complete problems proved by Karp in 1972 [16], to MPII. Suppose that $S = \{s_1, s_2, ..., s_n\}$ are the given sets and $\{a_1, a_2, ..., a_m\}$ is the union of all the sets. Given the number k_1, the decision version of the hitting set problem states whether or not there exists a set A with k_1 or fewer elements such that every element of S (every set s_i s.t. $i = 1, ..., n$) contains at least one element of A. The hitting set decision problem is denoted as HSD(S). The DAG H from the given set S is created as follows (Figure 2(a)). s is considered as the start vertex of DAG H. For each set, s_i, of HSD s.t. $i = 1, ..., n$, the corresponding vertex, s_i, is considered and an edge is added with an infinite weight from s to each s_i. Then, for each element a_j of the union of the input sets s.t. $j = 1, ..., m$, the corresponding vertex a_j is considered and an edge is added from each s_i to any a_j s.t. $a_j \in s_i$ in HSD. Furthermore, two final vertices called k and t are considered and two edges are added from each a_j s.t. $j = 1, ..., m$ to both final vertices. Finally, each vertex s_i s.t. $i = 1, ..., n$ is connected to vertex k. Clearly, H can be made in polynomial time.

Because the weight of every outgoing edge of s is infinite in H, no outgoing edges of s are marked. If all other edges of H are considered, namely edges $E_1 = E(H) - oe(s)$, it can be observed that every edge of E_1 has the same weight. Therefore, the sum of the weights of the marked edges of H equals the number of marked edges. It is now shown that HSD(S) has a solution with k_1 or fewer elements if and only if MPII = (H, s, t) has a solution with $n + k_1$ or fewer marked edges s.t. $n = |S|$.

HSD → MPII. Suppose that A is a set with k_1 or fewer elements such that every element of S contains at least one element of A. It will now be shown that MPII = (H, s, t) has a solution with $n + k_1$ or fewer marked edges. For each element $a_j \in A$, mark edge (a_j, t) with T. Furthermore, because A contains at

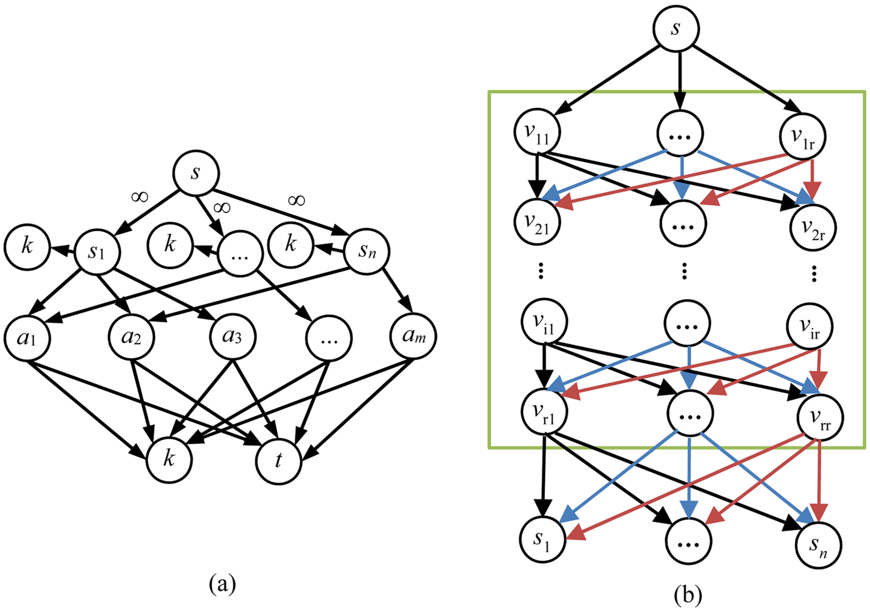

Figure 2. (a) Digraph of marking problem corresponding to hitting set problem. (b) Square digraph M with $r * r$ vertices s.t. $r = |E| + 1$. Subgraph M is placed between start vertex s and s_i' s.

least one element of any element of S, for each set $s_i \in S$, mark one and only one outgoing edge of s_i called (s_i, a_p) with T s.t. $a_p \in s_i \cap A$. Now, when moving from vertex s of H, vertex s_i s.t. $i = 1, ..., n$ is reached first. Because one outgoing edge of every element of S has been marked, by starting from s_i, vertex a_p s.t. $a_p \in A$ will be reached. Finally, as the outgoing edge (a_p, t) of any element of A is marked, t will be reached and the total number of marked edges is at most $n + k_1$. Notice that H has no cycle and any path of H is finite.

MPII → HSD. Suppose that function $f_1 : E_1 \rightarrow \{T, F\}$ is a solution to MPII $= (H, s, t)$ s.t. $E_1 \subset E(H)$. Any solution to the marking problem needs to mark one and only one outgoing edge of every s_i s.t. $i = 1, ..., n$. At least one outgoing edge of each s_i must be marked because vertex s_i has a direct edge to vertex k which never reaches t. In addition, at most one outgoing edge of s_i must be marked because either edge (s_i, k) can be marked with F or edge (s_i, a_j) with T s.t. $a_j \in s_i$ in HSD(S) and the weights of both the edges are the same. Hence, it is not necessary to mark more than one outgoing edge of s_i. Furthermore, one and only one outgoing edge of some a_j's must be marked s.t. $1 \leq j \leq m$. Indeed, depending on which outgoing edge of any element of S is marked, the outgoing edges of the corresponding a_j's, but not all a_j's, must be marked. Again, at least one outgoing edge of each of such a_j must be marked, because vertex a_j has a direct edge to vertex k which never reaches t. In addition, at most one outgoing edge of each of such a_j must be marked because either (a_j, k) can be marked with F or (a_j, t) with T, and the weights of both the edges are the same. Hence, there is no need to mark both of the outgoing edges of a_j. Thus, the solution to MPII has used at most $n + k_1$ marked edges s.t. n is the number of marks used in the form of $((s_i, a_j) \rightarrow T$ or $(s_i, k) \rightarrow F)$ for all s_i's, and k_1 is the number of marks used in the form of $((a_j, t) \rightarrow T$ or $(a_j, k) \rightarrow F)$ for some a_j's. Thus, the solution to MPII can be considered to be $f_1 = f_2 \cup f_3$, such that $f_2 : (s_i, x) \rightarrow \{T, F\}$ s.t. ($x = a_j$ or $x = k$, $i = 1, ..., n$, $1 \leq j \leq m$), and $f_3 : (a_j, y) \rightarrow \{T, F\}$ s.t. ($y = t$ or $y = k$, $1 \leq j \leq m$). The tail set of the domain of function f_3 can be denoted as A_1. It is claimed that A_1 with size k_1 is a solution to HSD(S). Suppose that A_1 does not hit one of the elements of S, e.g., s_i. This means that, in digraph H, no outgoing edges of some a_j's have been marked s.t. $a_j \in s_i$ in HSD(S). Hence, path $s.s_i.a_j.k$ in H starts at s, but does not reach t, which is a contradiction because f_1 is the solution to marking problem MPII $= (H, s, t)$.□

Theorem 2. *The marking problem is NP-complete even if the underlying digraph is an unweighted DAG.*

Proof. Let G be an unweighted DAG, s and t be two vertices of G, MPI $= (G, s, t)$ be a decision problem of the marking problem, and MPX $= (G, s, t)$ be another decision problem of the marking problem with the extra condition of "the outgoing edges of s are not markable". It can be observed that, in the proof of Theorem 1, no outgoing edges of s are marked, as if the outgoing edges of s are not markable, implying that MPX is NP-complete. By Theorem 1, the decision version of the marking problem is NP in a weighted DAG, so MPI is NP. To show the NP-hardness of MPI, the MPX is reduced to MPI as follows: make square digraph M with $(|E|+1)*(|E|+1)$ vertices such that each vertex of any row of the digraph is connected to all vertices of the next row of M (Figure 2). Now, DAG $G' = (V', E')$ is made from G as follows: First, remove every outgoing edge of s in G. Then, connect s to each vertex of the first row of digraph M. After that, connect each vertex of the last row of M to every successor of s in G. Moreover, set the weight of every edge of G' to 1. Notice that the structure of DAG G' is similar to that of DAG G, except that subgraph M has been added between vertex s and its successors. Clearly, DAG G' can be made in polynomial time. Now, it will be shown that MPI has a solution with k or fewer marked edges in G' if and only if MPX has a solution with k or fewer marked edges in G.

MPX → MPI. Let function $f_1 : E_1 \rightarrow \{T, F\}$ be a solution to MPX in DAG G s.t. $E_1 \subset E$. Moreover, let $k = |E_1|$. If function f_1 is applied to G', then the reachability of t from s is assured in G' and the number of the marked edges is k.

MPI → MPX. Let $c_1, ..., c_n$ be the successors of s in G. DAG G' is made such that the reachability cost of any vertex c_i from s is at least $|E|+1$ s.t. $1 \leq i \leq n$. For instance, consider a path from s to c_i in G' and mark every edge of the path with T. As the length of p is $(|E| + 2)$, then $(|E| + 2)$ edges are marked with T. Also, consider the edge set connecting the last row of M to c_i and mark them with T. In this case, the $(E+1)$ edges are marked with T. Also, the reachability cost of some c_i's from s is at least $(|E|+1)$. Indeed, in G', it is impossible to reach c_i or a set of c_i's from s (e.g., c_1 or c_2) with $|E|$ marks. Let function $f_2 : E'_1 \rightarrow \{T, F\}$ be a solution to MPI in DAG G' s.t. $E'_1 \subset E'$. Moreover, let $k = |E'_1|$. Firstly, suppose that $E'_1 \cap E(M) = \phi$. In this case, if function f_2 is applied to G, the reachability of t from s is assured in G and the number of marked edges is k. Now, suppose that $E'_1 \cap E(M) = \phi$. If $k \geq |E|$, and then function f_2 can be ignored. Indeed, for each c_i s.t. $1 \leq i \leq n$, it is possible to select the shortest path p from c_i to t in G and mark every edge of p with T using fewer than $|E|$ marks, because $|E|$ is the total number of edges of G. If $k < |E|$, the marked edges of set $E'_1 \cup E(M)$ cannot assure reaching c_i or a set of c_i's in G'. Thus, the marked edges of set $E'_1 \cap E(M)$ can be ignored, implying that only the marked edges of $(G' - M)$ should be considered in G to assure reaching t from s, and that the number of marked edges will be less than k.□

Theorem 3. *If the underlying digraph is a weighted binary DAG, then the marking problem is NP-complete.*

Proof. Let G be a weighted binary DAG, s and t be two vertices of G, and MPII $= (G, s, t)$ be a decision problem of the marking problem. By Theorem 1, MPII is NP-complete. Theorem 3 explains that MPII remains NP-complete even if the maximum out-degree of any vertex of G is two. The proof is exactly the same as that of Theorem 1, except that the new binary DAG H'' should be considered instead of DAG H in the reduction of HSD(S) to MPII. In DAG H from the proof of Theorem 1, the out-degree of s and any s_i s.t. $i = 1, ..., n\,(n = |S|)$ can be greater than 2; however, the out-degree of any a_j is 2 s.t. $j = 1, ..., m$ and m is the number of elements of the union of all elements of S. Hence, to convert H to a binary DAG, the structure of the outgoing edges of s and s_i's should be modified. DAG H can be converted into binary DAG H' in polynomial time. This can be demonstrated by example using $n = 3$ and $m = 5$. Figure 3(a) shows the DAG H of HSD(S) with $n = 3$ and $m = 5$.

The outgoing edges of s are substituted with the binary DAG given in Figure 3(b). Herein, because the weight of every edge of the binary DAG is infinite, none of the edges is marked. Therefore, to assure reaching target vertex t, the outgoing edges of every s_i s.t. $i = 1, ..., n$ should be marked. Therefore, replacing the outgoing edges of s with the binary DAG given in Figure 3(b) will not change the solution to MPII in H. For each s_i of H, its outgoing edges are substituted with the binary DAG given in Figure 3(c). In order to assure reaching t, it must be assured that it is possible to reach a_j from s_i (in this case s_1) s.t. $a_j \in s_i$ in HSD(S). The binary DAG given in Figure 3(c) is made such that the reachability cost of any a_j from s_i is the same. In order to keep the same marking cost in H and H', the weights of the outgoing edges of s_i's in H should be changed. If the out-degree of any s_i in H is greater than 2, then the weight of every outgoing edge of s_i in H should be changed to (out-degree(s_i) + 2). Now, the cost of reaching any a_j from s_1 in Figure 3(c) is 5, which equals the cost of marking one outgoing edge of s_1 in H, namely $(3 + 2)$. Thus, replacing the outgoing edges of any s_i with the binary DAG given in Figure 3(c) does not change the sum of weights of the marked edges. Thus, substituting the outgoing edges of s and any s_i with the binary DAGs given in Figure 3(b) and (c) does not change the cost of marking to reach t, implying that $(H, s, t) = (H', s, t)$. Hence, digraph H' can be considered instead of H in the proof of Theorem 1; therefore, the theorem holds. This proof was given for $n = 3$ and $m = 5$. For arbitrary values of n and m, only the heights of the binary DAGs presented in Figure 3(b) and (c) are increased polynomially. The presented idea is generalizable to different values of n and m. Moreover, the infinite weight of any edge of H or H' can be substituted with M such that M is the sum of the weights of all edges with finite weights.□

Conjecture. The marking problem is NP-complete even if the underlying digraph is an unweighted binary DAG.

4.2. Heuristic algorithm for solving the marking problem

Definition 2 (complete flow graph). *Quadruple $G = (V, E, s, f)$ where (V, E) is a digraph, $s \in V$ is the*

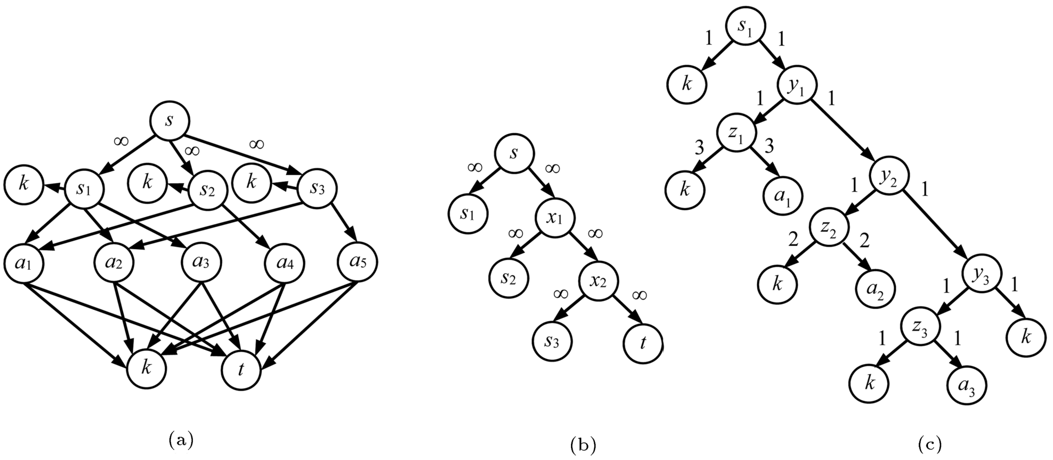

Figure 3. (a) Digraph H of marking problem corresponding to hitting set problem with $n = 3$ and $m = 5$. (b) The conversion of outgoing edges of s to binary mode. (c) The conversion of outgoing edges of s_1 to binary mode.

```
$H = \phi$
for each $v_k \in V(G)$ do
  if ($v_k \in reach(v_i)$ and $v_j \in reach(v_k)$) then $H = H \cup \{v_k\}$
end for
$markedEdges = \phi$
for each $v_k \in H$ do
  $outgoingEdgesToMarkT = \phi$, $outgoingEdgesToMarkF = \phi$
  for each $e \in outgoingEdges(v_k)$ in $G$ do
    if ($head(e) \notin H$) then
      $outgoingEdgesToMarkF = outgoingEdgesToMarkF \cup \{e\}$
    else
      $outgoingEdgesToMarkT = outgoingEdgesToMarkT \cup \{e\}$
    end if
  end for
  if ($outgoingEdgesToMarkF \neq \phi$) then
    $e = edgeWithMinWeight(outgoingEdgesToMarkT)$
    if ($weight(e) < sumOfWeights(outgoingEdgesToMarkF)$)
      $markedEdges = markedEdges \cup \{(e, T)\}$
    else
      $markedEdges = markedEdges \cup U\{(e, F)\}_{e \in outgoingEdgesToMarkF}$
    end if
  end if
end for
return $markedEdges$
```

Algorithm 1. CFG(G, v_i, v_j).

unique start vertex of G, and $f \in V$ is the unique final vertex of G, there is a path from start vertex s to each vertex of G; as well as a path from each vertex of G to final vertex f.

Let $G = (V, E)$ be a weighted DAG and v_i and v_j be two vertices of G. Algorithm 1 assures reaching v_j from v_i by removing any vertex of G which is not reachable from v_i or does not reach v_j. To remove such vertices, it is sufficient to consider induced sub-graph $G[H]$ s.t. $H = \{k \in V | k \in \text{reach}(v_i) \, \text{and} \, v_j \in \text{reach}(k)\}$ and, then, remove every outgoing edge of H, namely every edge e of G s.t. $tail(e) \in H$ and $head(e) \notin H$. By the removal of the outgoing edges of sub-graph H of G, digraph G will be converted to a complete flow graph with start and final vertices v_i and v_j, respectively, where every path starting at v_i will finally reach v_j. Instead of removing (marking with F) multiple sibling outgoing edges $e_1, e_2, ...$ of H, their sibling edge e_k can be marked with T s.t. $head(e_k) \in H$ and the weight of e_k is less than the sum of weights of the F-marked edges.

Let $G = (V, E)$ be a weighted DAG, s and t be two vertices of G, E_1, E_2, and E_3 be the subsets of E, $\text{MP} = (G, s, t)$ be an instance of the marking problem, $f_1 : E_1 \rightarrow \{T, F\}$ be an optimal solution to the MP, $f_2 : E_2 \rightarrow \{T, F\}$ be a heuristic solution to the MP, and $\text{SP} : E_3 \rightarrow \{T\}$ be the T-marked edges of the shortest path from s to t in G. $|E_i|$ s.t. $E_i \subset E$ denotes the sum of the edge weights. According to the definition of the marking problem, $|E_1| \leq |E_3|$, which means that the shortest path length is an upper bound to the solution of the marking problem. Thus, the solution of a good heuristic algorithm to the MP would be better than $\text{SP}(s, t)$, which means that $|E_1| \leq |E_2| \leq |E_3|$. One trivial way to compute f_2 is to compute $\text{SP}(s, t)$ and, then, mark every edge of the shortest path with T. However, in order to compute f_2, marking a *set* of edges can be considered instead of marking the *sequential* edges of a path (Algorithm 1). In some cases, the value of function $\text{CFG}(G, s, t)$ is a good initial value for the solution of the MP. However, the size of $\text{CFG}(G, s, t)$, which is the sum of weights of the edges marked by $\text{CFG}(G, s, t)$, may be greater than that of $\text{SP}(s, t)$. Therefore, $\min(\text{SP}(s, t), \text{CFG}(G, s, t))$ should

```
1. $G' = (V', E')$, $V' = V, E' = E$
2. $E' = E' - outgoingEdges(t)$
3. $V_1 = \{v \in V' | v \notin reach(s)\}$
4. $V' = V' - V_1$, $E' = E' - adjacentEdges(V_1)$
5. MP: A matrix $|V'|^*|V'|$ such that each cell is a set of $(e', X)$ s.t. $e' \in E'$ and $X \in \{T, F\}$
6. for each $v_i \in V'$ do
7.     for each $v_j \in$ reach$(v_i)$ do
8.         SP$(v_i, v_j) = U\{(e, T)\}_{e \in shortestPath(v_i, v_j)}$
9.         MP$(v_i, v_j) = $ min(SP$(v_i, v_j)$, CFG$(G', v_i, v_j)$)// get the set with minimal total weight
10.    end for
11. end for
12. $vertexList = reverseTopologicalSort(G')$
13. for each $v_i$ in $vertexList$ do// $t$ is the first element of the list
14.    for each $v_j \in$ reach$(v_i)$ do
15.        MP$(v_i, t) = $ min(MP$(v_i, t)$, MP$(v_i, v_j) \cup$ MP$(v_j, t)$)// get the set with minimal total weight
16.    end for
17. end for
18. return MP$(s, t)$
```

Algorithm 2. MP_DAG $(G = (V, E), s, t)$.

be considered as the initial value for the solution of the MP, namely, the one with the minimal total weight of the marked edges. Algorithm 2 improves this initial value by using an iterative improvement technique.

Lemma 4. *Algorithm 2 provides a heuristic solution to the marking problem in a given weighted DAG.*

Proof. The removal of the outgoing edges of t obviously has no effect on the solution of the marking problem. Moreover, by Observation 4, the marking problem can be considered in flow graph $G' = (V', E', s) = \text{FG}(G, s)$ instead of digraph G (lines 1-4 of Algorithm 2). In the initialization phase of the algorithm, for each pair (v_i, v_j) s.t. $v_j \in \text{reach}(v_i)$, consider the minimum $\text{SP}(v_i, v_j)$ and $\text{CFG}(G', v_i, v_j)$ as the initial values for the solution of marking problem (G', v_i, v_j) (lines 6-11 of Algorithm 2). Function $\text{SP}(v_i, v_j)$ is a T-marking of the shortest path from v_i to v_j in G'. Function $\text{CFG}(G', v_i, v_j)$ uses the marking approach to assuring that v_j is reached from v_i in G'. It considers the set of every vertex reachable from v_i and to v_j. This vertex set with all edges among them creates induced subgraph H. Now, in order to assure reaching t from s in G', it is enough to mark the outgoing edges of H with F or to mark a sibling edge of the F-marked edges with T, if the weight of T-marked edge is less than the total weight of the F-marked edges.

To find a better solution to marking problem $\text{MP} = (G', s, t)$, an iterative improvement technique is used to assure reaching intermediate vertex v from s (for each v of V' s.t. $v \in \text{reach}(s)$ and $t \in \text{reach}(v)$) and, then, reaching vertex t from v. To apply this technique, G' is sorted reverse-topologically (line 12 of Algorithm 2) where t and predecessors of t become the first vertices of G' for the computation of the MP. For each vertex v_i in the reverse-topologically sorted vertex list of G', the formula in line 15 of Algorithm 2 is used to assure reaching intermediate vertex v_j from v_i, for each vertex v_j between v_i and t and, then, reaching t from v_j. Because the order of v_j is less than that of v_i in the sorted list, $\text{MP}(v_j, t)$ should be computed prior to $\text{MP}(v_i, t)$. When execution of lines 13-17 of Algorithm 2 is finished, $\text{MP}(v_i, t)$ is computed for the last vertex of the list, namely for $v_i = s$. Hence, $\text{MP}(s, t)$ is a heuristic solution to the given marking problem, as it assures reaching t from s.□

Complexity. The complexity of finding single-source shortest paths in the weighted DAG G is $\Theta(|V| + |E|)$ [17]. Thus, the complexity of finding all-pairs shortest paths is $\Theta((|V| + |E|) * |V|)$ which equals $\Theta(|V| * |E|)$ in a flow graph. Suppose that the function *shortestPath* stores the shortest path between every pair of vertices in a matrix $|V| * |V|$ for subsequent access. Moreover, suppose that the function *reach* has already been computed and stored in matrix $|V|*|V|$. Hence, the memory consumption of algorithm MP_DAG is $\Theta(|V|^2+|E|)$. The complexity of algorithm CFG is $\Theta(|V|+|E|)$ in the worst case. Hence, the complexity of lines 6-11 of Algorithm 2 is $\Theta(|V|^2 * (|V|+|E|))$, which equals $\Theta(|V|^2 * |E|)$ in a program flow graph. Moreover, the complexity of lines 13-17 of Algorithm 2 is $\Theta(|V|^2 * |E|)$ because of the union of two edge sets. The complexity of other parts of Algorithm 2 is linear or constant. Thus, the total complexity of Algorithm 2 is $\Theta(|V|^2 * |E|)$. If G indicates the flow graph of a computer program, then usually $\Theta(|E|) = \Theta(|V|)$. Therefore, in this particular case, the total complexity of MP_DAG is $\Theta(|V|^3)$.

Lemma 5. *The back edges of flow graph G have no effect on the computation of the solution of the marking problem in G.*

Proof. Let $G = (V, E, s)$ be a flow graph, t be a vertex of G, and MP(G, s, t) be an instance of the marking problem. At first, it is claimed that the statement "every path starting at s reaches t" is equivalent to the statement "every simple path starting at s reaches t". If there exists a non-simple path p that starts at s and does not reach t, then there also exists a simple path p' that starts at s and does not reach t. The converse holds trivially. Thus, the claim holds. Any simple path starting at s cannot contain a back edge; otherwise, it will contain a cycle, which is a contradiction. Hence, the back edges of flow graph G do not change the set of simple paths from s to t in G. This implies that the back edges of G can be removed and the solution of the MP can be computed in G' s.t. $G' = G - backEdges(G)$. Note that the back edges of a flow graph can be computed using a dominator tree of the flow graph with linear complexity [18].

Observation 6. *Algorithm MP_DAG can be used for cyclic flow graphs, which are reducible. By Lemma 5, the back edges of the reducible flow graph can be removed and, then, the solution of the marking problem is computed in the obtained acyclic flow graph.*

4.3. Evaluation of heuristic algorithm

The heuristic algorithm is implemented and its quality and running time evaluated in many randomly generated and user-specified digraphs. The quality of a heuristic algorithm in fact implies the distance of the heuristic solution from the optimal one. In most cases, the heuristic algorithm gives an optimal solution to the marking problem. However, if the solution of the marking problem cannot be computed by composing the solutions of the constituent marking problems, the heuristic algorithm does not provide an optimal result. Figure 4 shows such an example.

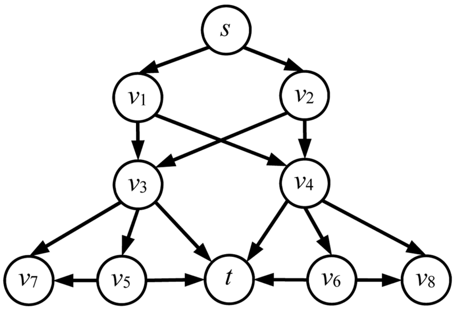

Figure 4. An example of a digraph in which the heuristic algorithm does not give an optimal solution to the marking problem (G, s, t).

Example 2. In the unweighted DAG G of Figure 4, one solution to MP $= (G, s, t)$ using Algorithm 2 is function $f_1 : \{(v_1, v_3), (v_2, v_3), (v_3, t)\} \rightarrow \{T\}$; however, the optimal solution to the MP is function $f_2 : \{(v_3, t), (v_4, t)\} \rightarrow \{T\}$.

Herein, the quality of the proposed heuristic algorithm is compared with that of the optimal one based on thousands of digraphs generated by four graph generators focused mainly on the flow graphs, namely the connected digraphs whose vertices are reachable from the start vertex.

FileBasedGraphGenerator allows definition of the specification of a digraph in a file in the form of an adjacency list. The program can parse this file and generate an object of the digraph. Except for the file-based graph generator, every other graph generator first creates a base graph and, then, adds randomly generated edges to the base graph. *ChainBasedRandomGraphGenerator* generates a random digraph based on a chain. It first creates a chain (a sequence of connected edges) with the length of $n-1$ for a digraph with n vertices and, then, adds randomly generated edges to the chain.

TreeBasedRandomGraphGenerator generates a random digraph based on a binary tree. It first creates a binary tree having a specific depth and, then, adds randomly generated edges to it. Finally, *CounterExampleBasedRandomGraphGenerator* generates a random graph based on a counterexample digraph. The counterexample digraph specifies a case in which the heuristic algorithm does not give the optimal solution to the marking problem. It first creates a counterexample digraph and, then, adds randomly generated edges to the digraph. Note that the number of vertices of every random graph generated by a particular graph generator is fixed. However, the number of generated edges varies.

The evaluation results are shown in Table 4. In most cases, the heuristic algorithm gives the optimal solution to the marking problem. In order to compute this optimal solution, an exponential-time algorithm is implemented that considers different combinations of

Table 4. Evaluating the quality and running time of the proposed heuristic algorithm to the MP. 'H' denotes the heuristic algorithm and 'E' denotes the exponential-time algorithm to the MP.

Benchmark	**Description**	**Quality**	**Average running time (mili-second)**
1	It generates the single digraph of Figure having 10 vertices using *FileBasedGraphGenerator* .	The heuristic solution is NOT optimal for the target vertex t, but the distance of the solutions is 1.	$H : 47$ $E : 250$
2	It generates the single digraph of Figure 5 having 26 vertices using *FileBasedGraphGenerator* and considers each vertex of the digraph (except the start vertex) as a target vertex. Hence, the comparison is performed for $(26 - 1) = 25$ cases.	For all 25 cases, the heuristic solution is optimal.	$H : 807$ ms $E : 10919$ ms
3	It generates 34 random digraphs having 31 vertices using *TreeBasedRandomGraphGenerator* and considers each vertex of each digraph (except the start vertex) as a target vertex. Hence, the comparison is performed for $(31 - 1)^*34 = 1020$ cases.	For all 1020 cases, the heuristic solution is optimal.	$H : 436$ ms $E : 299234$ms
4	It generates 68 random digraphs having 16 vertices using *ChainBasedRandomGraphGenerator* and considers each vertex of each digraph (except the start vertex) as a target vertex. Hence, the comparison is performed for $(16 - 1)^*68 = 1020$ cases.	The heuristic solution is optimal for 1010 cases out of 1020 ones. Therefore, it gives the optimal solution in 99% of the cases.	$H : 46$ ms $E : 4681$ ms
5	It generates 43 random digraphs having 25 vertices using *CounterExampleBasedRandomGraphGenerator* and considers each vertex of each digraph (except the start vertex) as a target vertex. Therefore, the comparison is performed for $(25 - 1)^*43 = 1032$ cases.	For all 1032 cases, the heuristic solution is optimal.	$H : 259$ ms $E : 160335$ ms

edges to be marked. The algorithms are implemented and tested in Java 1.7 with 1 GB of heap memory. The computer used for testing is an ASUS X554L laptop with Windows 8.1 equipped with an Intel Core i5-5200U CPU running at 2.20 GHz with 7.90 GB of usable main memory. As observed, the worst-case running time of the heuristic algorithm was less than one second, whereas the exponential-time algorithm consumed minutes.

5. Comparison of reachability assurance approaches

This section provides the empirical and theoretical results of the comparison of the reachability cost in the current and proposed RA approaches, namely the SP and MP, respectively. In this section, sample flow graph G is first considered; then, the reachability cost of each vertex of G is computed. The reachability costs are computed for both the pathfinding and the MP approaches, and it is shown that the MP always gives a better solution to the RA problem than the pathfinding approach does.

Let G be the unweighted flow graph of Figure 5 with start and final vertices s and f, respectively. Moreover, let t be an arbitrary vertex of G. The goal is to assure that target vertex t is reached from start vertex s in G. By Lemma 3, in unweighted flow graphs, the MP has a solution that uses only mark T. In the following, the use of only mark T is considered for the marking approach. Moreover, by Observation 6, the back-edges of G, namely the set $\{e''_{10}, e''_{11}, e''_{18}, e''_{23}, e'_{24}, e'_{25}\}$, have no effect on solution of the MP; hence, they can be removed from G in advance.

For each vertex v of G, Table 5 depicts the reachability condition and the reachability cost of v by both RA approaches. Because G is unweighted,

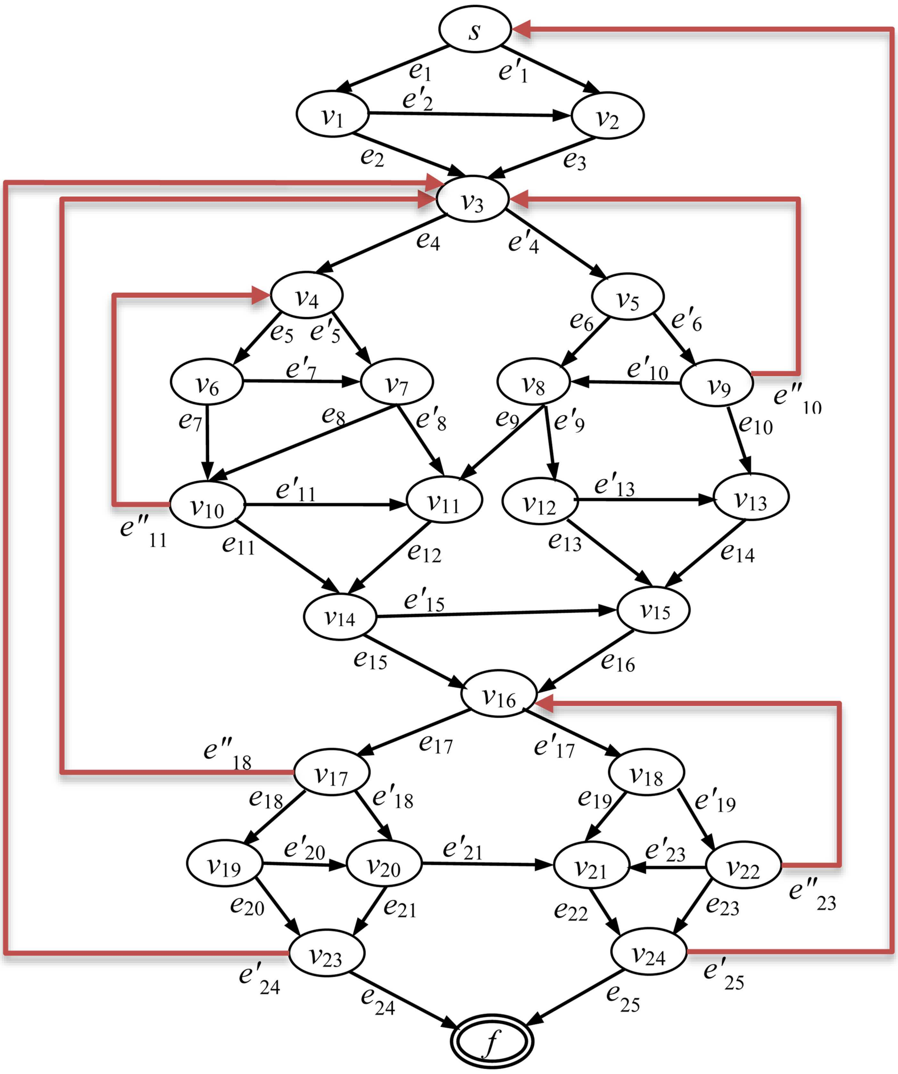

Figure 5. The flow graph of a program with start and final vertices s and f, respectively. The flow graph has 6 back edges.

the reachability condition is specified by a set of edges to be marked with T and the reachability cost is specified by the number of the edges to be marked with T. Note that none of the RA approaches has unique solutions. Hence, in Table 5, an arbitrary solution to these approaches has been considered.

According to Table 5, the total cost of the pathfinding approach is 137, whereas the total cost of the marking approach is 39, which implies that the reachability cost of the pathfinding approach is $(137/39 \simeq 3.51)$ times greater than that of the marking approach in the flow graph of Figure 5.

A Java program was developed to compare the reachability cost of both RA approaches on 5000 random flow graphs, each containing 50 vertices and 88 edges. The results show that the reachability cost of the pathfinding approach is 3.47 times greater than that of the marking approach.

To verify the correctness of a solution to the marking problem, it is sufficient to remove all sibling edges of every T-marked edge of the solution and, then, check whether every path starting from s finally reaches the target vertex. In Table 5, it is interesting to note that the maximum reachability cost of the marking approach is 3, while that of the pathfinding approach is 10. Note that, in this example, the proposed heuristic algorithm provides an optimal solution to the marking problem for all vertices of the digraph.

Definition 3 (disconnection ratio). *Let G be a flow graph with start vertex s and $v \in V$ be a cut vertex of G. The disconnection ratio of G on v is $|V(G_1)|/|V(G_2)|$ such that G_1 and G_2 are the disconnected components of G after the removal of v and $s \in V(G_1)$.*

Lemma 6. *Let G be a flow graph with start vertex s, n be the number of vertices of G, $v \in V$ be a cut vertex, and the disconnection ratio of G on v be $1/p$.*

Table 5. The reachability cost of every vertex of the flow graph of Figure 5 in both RA approaches, namely the SP and MP.

Vertex	The reachability condition in the SP	The reachability condition in the MP	The reachability cost in the SP	The reachability cost in the MP
v_1	$\{e_1\}$	$\{e_1\}$	1	1
v_2	$\{e'_1\}$	$\{e'_1\}$	1	1
v_3	$\{e_1, e_2\}$	ϕ	2	0
v_4	$\{e_1, e_2, e_4\}$	$\{e_4\}$	3	1
v_5	$\{e_1, e_2, e'_4\}$	$\{e'4\}$	3	1
v_6	$\{e_1, e_2, e_4, e_5\}$	$\{e_4, e_5\}$	4	2
v_7	$\{e_1, e_2, e_4, e'_5\}$	$\{e_4, e'_5\}$	4	2
v_8	$\{e_1, e_2, e'_4, e_6\}$	$\{e'_4, e_6\}$	4	2
v_9	$\{e_1, e_2, e'_4, e'_6\}$	$\{e'_4, e'_6\}$	4	2
v_{10}	$\{e_1, e_2, e_4, e_5, e_7\}$	$\{e_4, e_8\}$	5	2
v_{11}	$\{e_1, e_2, e_4, e'_5, e'_8\}$	$\{e_4, e'_5, e'_8\}$	5	3
v_{12}	$\{e_1, e_2, e'_4, e_6, e'_9\}$	$\{e'_4, e_6, e'_9\}$	5	3
v_{13}	$\{e_1, e_2, e'_4, e'_6, e_{10}\}$	$\{e'_4, e'_6, e_{10}\}$	5	3
v_{14}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}\}$	$\{e_4\}$	6	1
v_{15}	$\{e_1, e_2, e'_4, e_6, e'_9, e_{13}\}$	$\{e'_4, e'_9\}$	6	2
v_{16}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}\}$	ϕ	7	0
v_{17}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e_{17}\}$	$\{e_{17}\}$	8	1
v_{18}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e'_{17}\}$	$\{e'_{17}\}$	8	1
v_{19}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e_{17}, e_{18}\}$	$\{e_{17}, e_{18}\}$	9	2
v_{20}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e_{17}, e'_{18}\}$	$\{e_{17}, e'_{18}\}$	9	2
v_{21}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e'_{17}, e_{19}\}$	$\{e'_{17}, e_{19}\}$	9	2
v_{22}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e'_{17}, e'_{19}\}$	$\{e'_{17}, e'_{19}\}$	9	2
v_{23}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e_{17}, e_{18}, e_{20}\}$	$\{e_{17}, e_{21}\}$	10	2
v_{24}	$\{e_1, e_2, e_4, e_5, e_7, e_{11}, e_{15}, e'_{17}, e_{19}, e_{22}\}$	$\{e'_{17}\}$	10	1
			Sum = 137	Sum = 39

Moreover, let l be the length of the shortest path from s to v. The lower bound of the saved cost of reachability is $l * (n - 1) * p/(1 + p)$ when the MP approach is used instead of the SP.

Proof. Because v is a cut vertex, every path starting at s passes through v. Hence, in order to reach v from s using the MP approach, there is no need to mark any edge of G. This implies that the cost of reachability from s to v in the MP approach is zero. In contrast, the minimal cost of reachability from s to v in the SP approach is l. As $|V(G_1)|/|V(G_2)| = 1/p$ and $|V(G_1)| + |V(G_2)| + 1 = n$, we have $|V(G_2)| = (n - 1) * p/(1 + p)$. In order to reach any vertex of G_2 from s in G using the SP approach, v should be reached first from s at cost l. Hence, when using the SP approach, the extra cost of $C = l*(n-1)*p/(1+p)$ will be accrued. The saved cost of reachability is greater than C because only the cost of reachability from s to v has been considered and not from v to the vertices of G_2. Moreover, only the reachability cost of the vertices of G_2 has been considered and not the vertices of G_1.□

A comparison of the current and proposed approaches to the reachability assurance problem is depicted in Table 6. The computational complexity of the single-source shortest path problem in a non-negative edge-weighted digraph results from applying Dijkstra's algorithm with a Fibonacci heap [19]. The complexity of the marking problem and its heuristic version is described in Section 4. The "reachability cost in theory" is provided by Lemma 6. The "reachability cost in practice" is provided by a benchmark performed on 5000 random flow graphs. By assuming the reachability cost of the MP-Heuristic to be k, the reachability cost is computed for the two other approaches in theory and practice. The pathfinding approach specifies all the edges to be followed in order to reach the target vertex; therefore, it is said that its reachability type is total. However, the marking approach does not specify the exact path; hence, it is said that its reachability type is partial.

Table 6. Comparison of the RA approaches in flow graph G with n vertices and m edges. To compute the reachability cost in theory, it is supposed that G has a cut vertex with disconnection ratio $1/p$ where the shortest path length from s to v is l.

Reachability assurance approach	Computational complexity	Reachability cost (in theory)	Reachability cost (in practice)	Reachability type
Shortest Path (SP)	$m + n^* \log n$	$k + l^*(n-1)^*p/(1+p)$	3.5^*k	Total
Marking Problem (MP)	NP-complete	$\leq k$	$\leq k$	Partial
MP-Heuristic (MPH)	$n^{2^*}m$ in reducible flow graphs	k	k	Partial

6. Conclusion and future work

The marking problem was presented as an optimization problem that used minimal marks T and F to assure the reachability of t from s in a digraph G. If G is unweighted, the minimal number of marked edges will be desired; otherwise, the minimal sum of weights of the marked edges will be desired. We showed that the reachability cost of the pathfinding approach is 3.5 times greater than that of the MP approach in practice.

Fundamental properties of the marking problem were presented; then, it was proved that the marking problem was NP-complete in an arbitrary unweighted DAG as well as in an arbitrary weighted binary DAG. An appropriate heuristic algorithm was provided to the marking problem in a given DAG and demonstrated its high performance and quality by evaluating it on thousands of digraphs. It was shown that the provided algorithm could also be used for cyclic flow graphs that are reducible. In practice, most program flow graphs are reducible [20]. Given the results presented in this paper, new areas for further works have been identified, including:

- To prove whether or not the marking problem is NP-complete in an unweighted binary DAG;
- To present an approximation algorithm to compute a near-optimal solution to the marking problem;
- To compute the reachability cost of the marking problem in general flow graphs;
- To study the infeasibility problem in the context of the marking problem. If the underlying digraph of a marking problem indicates the flow graph of a computer program, some edge sets of the digraph cannot be marked with $\{T, F\}$. Indeed, a set of edges marked with T and F can lead to the generation of Boolean equations, which have no solution. The infeasibility problem is shown to be undecidable in general [21,22].

Acknowledgements

The authors would like to thank Hassan Mirian, Amir Daneshgar, Reza Sadraei, Batool Mokhtarshahi, AND Hossein Valizadeh for their help and encouragement throughout the paper writing process.

Abbreviations

RA	Reachability Assurance
MP	Marking Problem
SP	Shortest Path
HS	Hitting Set
DAG	Directed Acyclic Digraph

References

1. Yu, J.X. and Cheng J. "Graph reachability queries: A survey", In *Managing and Mining Graph Data*, Springer, pp. 181-215 (2010).
2. Agrawal, R., Borgida, A., and Jagadish, H.V. "Efficient management of transitive relationships in large data and knowledge bases", In *SIGMOD,* **18**(2), pp. 253-262 (1989).
3. Nuutila, E. "Efficient transitive closure computation in large digraphs", PhD Thesis, Finnish Academy of Technology (1995).
4. van Schaik, S.J. and de Moor, O. "A memory efficient reachability data structure through bit vector compression", *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011,* Athens, Greece, pp. 913-924 (12-16 June 2011)
5. Trißl, S. and Leser, U. "Fast and practical indexing and querying of very large graphs", In *SIGMOD'07* (2007).
6. Yildirim, H., Chaoji, V., and Zaki, M.J. "Grail: Scalable reachability index for large graphs", *PVLDB,* **3**(1), pp. 276-284 (2010).
7. Juping, W. "Discussion of graph reachability query with keyword and distance constraint", *IDEAL-2016*, pp. 293-301 (2016).
8. Ammann, P. and Offutt, J., *Introduction to Software Testing*, First Ed., Cambridge University Press, 120 p. (2008).
9. Sharma, P. and Khurana, N. "Study of optimal path finding techniques", *Int. J. Adv. Technol.*, **4**(2), pp. 124-130 (2013)

10. Dellin, C. and Srinivasa, S. "A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors", *ICAPS-2016*, London, UK (2016).

11. Anand, S., Burke, E., Chen, T.Y., Clark, J., Cohen, M.B., Grieskamp, W., Harman, M., Harrold, M.J., and McMinn, P. "An orchestrated survey on automated software test case generation", *Journal of Systems and Software,* **86**(8), pp. 1978-2001 (2013).

12. Do, T., Khoo, S.C., Fong, A.C.M., Pears, R., and Quan, T.T. "Goal-oriented dynamic test generation", *Information and Software Technology*, **66**, pp. 40-57 (2015).

13. Saito, N. and Nishizeki, T. "Graph theory and algorithms", *17th Symposium of Research Institute of Electrical Communication,* Tohoku University, Sendai, Japan (1980).

14. Prosser, R.T. "Applications of Boolean matrices to the analysis of flow diagrams", *AFIPS Joint Computer Conferences, Eastern Joint IRE-AIEE-ACM Computer Conference* (Boston, MA: ACM), pp. 133-138 (1959).

15. Aho, A.V., Lam, M.S., Sethi, R., and Ullman, J.D., *Compilers, Principles, Techniques, and Tools*, the 2nd Ed., Pearson Addison Wesley, pp. 659-666 (2007).

16. Karp, R.M. "Reducibility among combinatorial problems", In R.E. Miller and J.W. Thatcher (Eds.), *Complexity of Computer Computations*, New York: Plenum, pp. 85-103 (1972).

17. Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., *Introduction to Algorithms*, 2nd Ed., MIT Press and McGraw-Hill, ISBN 0-262-03293-7 (2001).

18. Fraczak, W., Georgiadis, L., Miller, A., and Tarjan, R.E. "Finding dominators via disjoint set union", *Journal of Discrete Algorithms,* **23**, pp. 2-20 (2013).

19. Fredman, M.L. and Tarjan, R.E. "Fibonacci heaps and their uses in improved network optimisation algorithms", *Journal of the ACM*, **34**(3), pp. 596-615 (1987).

20. Hecht, M.S., *Flow Analysis of Computer Programs*, Amsterdam: Elsevier North-Holland (1977).

21. DeMillo, R.A. and Offutt, J. "Constraint-based automatic test data generation", *IEEE Transactions on Software Engineering,* **17**(9), pp. 900-910 (1991).

22. Goldberg, A., Wang, T.C., and Zimmerman, D. "Applications of feasible path analysis to program testing", In *1994 International Symposium on Software Testing and Analysis*, Seattle, Washington, USA, pp. 80-94 (1994).

Biographies

Mohammad Valizadeh received his BSc and MSc degrees in Software Engineering from Shahid Beheshti University, Tehran, Iran. He is now a PhD student in Software Engineering at the Iran Telecommunication Research Center. His research interests include software testing, security testing, and graph theory. He has fifteen years of experience in the software testing and quality assurance industry.

Mohammad Hesam Tadayon received the MSc degree in Mathematics from the University of Tarbiat Modares, Tehran, Iran in 1997, and the PhD degree in Applied Mathematics (coding and cryptography) from the University of Tarbiat Moallem of Tehran (Kharazmi), Tehran, Iran in 2008. Currently, he is an Associate Professor at Iran Telecommunication Research Center (ITRC). He is the Director of IT Systems group at ITRC and also a member of national councils in the Iranian Ministry of Science and Technology. He has served in many research and industrial projects. His research interests include information theory, error-control coding, and data security.

Alireza Bagheri received his BSc and MSc degrees in Computer Engineering from Sharif University of Technology (SUT), Tehran, Iran. He received his PhD degree in Computer Science from Amirkabir University of Technology (AUT), Tehran, Iran. Currently, he is an Assistant Professor at the Computer Engineering and IT Department at Amirkabir University of Technology (AUT). His research interests include computational geometry, graph algorithms, big data and social network analysis.