Search-tree Based SDN Candidate Selection in Hybrid IP/SDN Network

Ning Li, Yue Shi, Zhaoxin Zhang, Jose Fernan Martinez, Xin Yuan

***Abstract*—The link failure recovery is important to the Internet. For improving the performance of link failure recovery in the IP network, the software defined networking (SDN) is applied to achieve this target. The SDN is effective on solving this kind of issue. However, considering the deployment cost, only a few IP routers can be replaced by the SDN switches. Thus, to minimize the number of SDN switches, the greedy-based approach is proposed to select the most appropriate deployment locations. But the greedy-based approach has disadvantages. For addressing these disadvantages, in this paper, we proposed the search-tree based SDN candidate selection (SCS) algorithm. In this algorithm, for achieving better performance than the greedy-based approach, three algorithms are proposed, which are the search-tree based feasible solutions calculation algorithm, the most appropriate feasible solution selection algorithm, and the most appropriate designated SDN switch selection algorithm. Based on these algorithms, the performance of the search-tree based SCS algorithm is improved greatly compared with the greedy-based algorithms.**

Keywords—SDN, link failure recovery, hybrid IP/SDN network

I. Introduction

Link failure recovery is important to the Internet since it guarantees to provide reliable and secure service for the user. Link failure recovery in the IP network means to reroute the traffic flows to the affected destinations without using the failed link. For recovering the link failure in the IP network, many IP fast rerouting mechanisms have been proposed in the past decades, such as [1-9].

The most widely used and simple link failure recovery approach is the loop free alternates [4]. When there is a link failure, the loop free alternates will reroute the affected packets to the pre-established repair paths. The main disadvantage of the loop free alternates is that it cannot protect all link failures, because the next hop of the loop free alternate may not exist. For addressing this problem, the tunneling/encapsulation based approach is proposed [5-6]. In this approach, the encapsulated packets are transmitted to the affected destination by using the shortest path through the tunnel. Unfortunately, even with both the loop free alternate [4] and the tunneling/encapsulation based approach [5], the existence of the repair path has not been solved thoroughly [7]. Moreover, the cost of this method is very high, since the directed forwarding that used in [5] and [6] needs the upgrade of all routers to support the directed forwarding mechanism. Therefore, the tunneling/encapsulation based approach with directed forwarding is not widely adopted. More similar approaches can be found in [8-9].

Recently, some efforts have been done by using the SDN to improve the performance of link failure recovery in the IP network [10]. As introduced in [10] and [11], by deploying a few SDN switches in the IP network, a resilient hybrid IP/SDN network can be established and the network performance can be improved [11]. In this approach, by setting up tunnels between traditional IP routers and SDN switches, once a link failure is detected, the IP routers can redirect the affected traffic flows to SDN switches immediately. The SDN switches can then forward the traffic to the affected destination without using the failed link based on the routing decision made by the SDN controller. Since the SDN controller has the knowledge of the entire network, it can make optimal routing decision for the post-recovery network [11]. However, even SDN controller is effective on recovering the link failure, considering the cost and the manpower needed to replace the original network devices with SDN switches, the number of SDN switches deployed in the IP network should be minimized, especially when the network size is huge [10][11]. This may result in a network where traditional IP routers and SDN switches co-exist at the same time, i.e., the hybrid IP/SDN network.

In the hybrid IP/SDN network, the first and most important problem is to decide which IP routers should be replaced by SDN switches to minimize the number of SDN switches. This is called the SDN candidate selection (SCS) issue [11]. The SCS issues have been investigated in [10] and [11] preliminary, and the greedy-based approaches are proposed. To the best of our knowledge, these are the only two algorithms to solve this problem. However, the greedy-based approaches proposed in [10] and [11] have disadvantages. First, in the greed-based algorithms, only one feasible solution can be found each time and it cannot find all the feasible solutions, so the selected SDN candidates maybe not the most appropriate one. Second, the greedy-based approach cannot guarantee to find the optimal solution always, i.e., the number of SDN switches calculated by the greedy-based approach maybe not the minimum. Finally, for a certain link, more than one candidate SDN switches can protect the failure of this link, how to choose the most appropriate one as the final designated SDN switch has not been investigated.

Based on the disadvantages introduced above, in this paper, we propose the search-tree based SDN candidate selection algorithm. The search-tree based SDN candidate selection algorithm can address the disadvantages in greedy-based approach. The main contributions of this paper are summarized as follows.

1. In this paper, we propose the search-tree based SCS algorithm. Based on this algorithm, all the feasible solutions can be found. Moreover, according to the branch and bound, the complexity of the search tree is reduced;

2. Since more than one feasible solution can be found, we proposed the most appropriate feasible solution selection algorithm. In this algorithm, the reliability degree of each feasible solution is defined and the most appropriate feasible solution is chosen based on it;

3. Considering that for each link, there is more than one SDN switch can protect it, we propose the most appropriate

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

designated SDN switch selection algorithm to select SDN switch for each link. In this algorithm, the average repair path length and the average link utilization of each SDN switch are considered;

4. We compare the search-tree based algorithm with the greedy-based algorithm, the simulation results show that the search-tree based algorithm can improve the performance greatly.

II. Network Model And Problem Statement

A. Network Model

The network model used in this paper is an undirected graph $G = (V, E)$, where V represents the number of nodes and E represents the number of bidirectional links. The nodes in the network include both the IP routers and the SDN switches. The communication link between node *i* and node *j* is bidirectional, denoted as (i, j). The bidirectional link (i, j) includes two directed links, i.e., link $< i, j >$ means data flows from node *i* to node *j* and link $< j, i >$ means data flows from node *j* to node *i*, respectively. When link (i, j) fails, both node *i* and node *j* can detect this event. For recovering the failure traffic, node *i* and node *j* will reroute the data flow in both link $< i, j >$ and link $< j, i >$. For simplifying the expression, each communication link is represented by a link number *e*, where $e \in [1, 2E]$.

The purpose of the SCS in hybrid IP/SDN network is to find the most appropriate positions to deploy SDN switches in the IP network so that the SDN switches can recover any single link failure in the network. In hybrid IP/SDN network, the conditions that a designated SDN switch can protect the failure of a directed link are defined in [10] and [11]: (1) the shortest path from router *i* to SDN switch *k* does not include link $< i, j >$ (where $< i, j >$ is the failed link); (2) for each affected destination, there exists at least one next-hop *h* of switch *k*, and the shortest path from *h* to the affected destination does not include $< i, j >$.

B. Greedy-based Approach

For achieving this purpose, the greedy-based SCS approach has been applied in [10] and [11]. The greedy-based approach includes two phases: (1) candidate table construction and (2) column selection. In this section, we use the example in [11] to introduce these two phases briefly. The network model is shown in Fig. 1 and the candidate table is shown in Table 1.

For the candidate table construction, according to the shortest path protocol in the IP network and for a certain destination, there is a sink tree that represents all the shortest paths from the rest nodes to this destination node. When a link fails, a set of destinations will be affected; all the affected destinations form the affected destination set. So, when link failure happens, the nodes which meet the conditions introduced in Section II.A will be labeled with “1” in the candidate table; otherwise, it is labeled by “0”. Based on this principle, the candidate table of the network shown in Fig.1 is presented in Table 1. After the candidate table construction, the purpose is to select the columns in the candidate table which are able to protect all the possible link failures, i.e., each element in the sum of the selected columns should be larger than 0.

For the column selection, first, the weight of each column is calculated. The weight of the column is the sum of all the elements in this column. The larger weight indicates that the more fail links can be protected by this SDN switch. Then, the column with the largest weight will be chosen and the corresponding rows of this column are all removed from the candidate table. After that, the weights of the remaining columns are calculated again, and the column with the largest value will be chosen and the corresponding rows will be removed. This process will be repeated until all the rows are removed from the candidate table, which means that all the failure links can be protected by at least one SDN switch. These selected columns are the locations that should be replaced by SDN switches.

An example is shown as follows. According to Table 1, column 8 and column 9 have the largest weight; based on the greedy-based approach, these two columns are chosen randomly. Assuming that column 8 is chosen, when removing all the corresponding rows of column 8, the column 3 has the largest weight, then column 3 will be chosen. This process will be repeated, and the column 9, column 1, and column 7 are selected one after the other. Thus, the locations of candidate SDN switches are $(8,3,9,1,7)$. For the greedy-based approach in [11], when there is a tie, the candidate through which the repair path length is the smallest will be chosen. For instance, the column 8 and column 9 have the same weight; however, through column 9, the repair path length is smaller than column 8, so column 9 is chosen. Based on this principle, the column 3, column 8, column 2, and column 7 are chosen one after the other. Therefore, the candidate SDN switches' locations are $(9,3,8,2,7)$. As demonstrated in [11], the $(9,3,8,2,7)$ is better than $(8,3,9,1,7)$ when considering the repair path length while the number of SDN switches in these two solutions are the same.

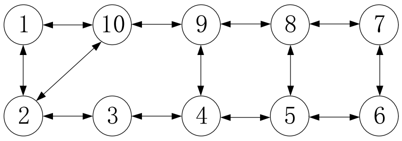

Fig.1. Network Model

Table 1. Candidate Table

Link Failure	SDN candidate									
	1	2	3	4	5	6	7	8	9	10
1, <1,2>	1	0	0	0	0	0	1	1	1	1
2, <1,10>	1	1	1	1	1	1	0	0	0	0
3, <2,1>	0	1	0	1	1	1	1	1	1	1
4, <2,10>	1	1	1	1	1	1	0	0	0	0
5, <2,3>	0	0	0	0	0	0	1	1	1	1
6, <3,2>	0	0	0	1	1	1	1	1	1	0
7, <3,4>	1	1	0	0	0	0	0	0	0	1
8, <4,3>	0	0	0	0	0	0	0	0	1	1
9, <4,5>	0	0	0	0	0	0	0	0	1	0
10, <4,9>	1	1	1	0	1	1	1	1	0	0
11, <5,4>	0	0	0	0	0	0	0	1	0	0
12, <5,6>	0	0	0	0	0	0	0	1	0	0
13, <5,8>	1	1	1	1	0	1	1	0	1	1
14, <6,5>	0	0	0	0	0	0	1	0	0	0
15, <6,7>	1	1	1	1	1	0	0	1	1	1
16, <7,6>	1	1	0	0	0	0	1	1	1	1
17, <7,8>	0	0	1	1	1	1	1	0	0	0
18, <8,5>	1	1	0	0	0	0	1	1	1	1
19, <8,7>	0	1	1	1	1	1	0	1	1	0
20, <8,9>	0	0	1	1	1	1	1	1	0	0
21, <9,4>	1	1	0	0	0	0	1	1	1	1
22, <9,8>	0	1	1	1	1	1	0	0	1	0
23, <9,10>	0	0	1	1	0	0	0	0	0	0
24, <10,1>	0	1	1	1	1	1	1	1	1	1

25, <10,2>	1	0	0	1	1	1	1	1	1	1
26, <10,9>	0	1	1	0	0	0	0	0	0	0

C. *Problem Statement*

To the best of our knowledge, the greed-based approach is the first-and-only effective algorithm to address the SCS issues. However, it has disadvantages. *First*, as the example shown in Section II.B, except for (8,3,9,1,7) and (9,3,8,2,7), there are still many other feasible solutions, such as (7,8,9,3,10) and (7,8,9,2,4). Even the (8,3,9,1,7), (9,3,8,2,7) and (7,8,9,3,10) can be gotten by executing the greedy-based approach repeatedly, the (7,8,9,2,4) cannot be found based on both the approaches proposed in [10] and [11]. *Second*, the greedy-based approach cannot guarantee to get the optimal solution always. For instance, as the example shown in Table 2, based on the greedy-based approach, the feasible solution should be (3,2,1). However, intuitively, the (2,1) is better than (3,2,1) because: on one hand, it can meet the constraints shown in Section II.A, on the other hand, the number of SDN switches is smaller than (3,2,1). *Third*, for each link, there may be more than one SDN switches can protect the failure of this link. For instance, as shown in Fig. 2, for the solution (8,3,9,1,7) calculated by the greedy-based approach, the link $< 5,8 >$ can be protected by four SDN switches, i.e., SDN_1, SDN_7, SDN_3, and SDN_9. In previous works, when link $< 5,8 >$ fails, which SDN switch is used to reroute the data flows in this link has not been investigated. So, the properties of the SDN switches, such as the repair path length and the load balancing, are not considered. Even the repair path length from the failed link to the specific destination through SDN switch *k* is calculated in [11], they use the average repair path length of each SDN switch to choose candidate column when there is a tie.

Table 2 An Example

Link failure		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SDN candidate	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	2	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	4	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
	5	0	1	0	0	0	1	0	0	0	1	0	0	1	0	1
	6	1	0	0	0	0	0	0	1	0	0	0	1	1	1	1

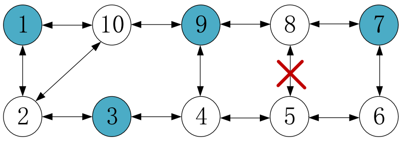

Fig. 2 An example of link failure

Based on the disadvantages introduced above, the problems will be investigated in this paper can be summarized as: (1) how to find all the feasible solutions with low computation complexity; (2) how to choose the most appropriate solution as the final SDN switches deployment locations; (3) how to guarantee the proposed approach can always find the optimal solution; (4) how to choose the most appropriate designated SDN switch for each link failure. These issues are not investigated by the previous works.

III. Proposed Search-Tree Based Approach

In this section, we propose the search-tree based SCS algorithm. Moreover, the best feasible solution selection algorithm and the most appropriate designated SDN switch selection algorithm are also proposed in this section.

A. *Search-Tree based SCS algorithm*

The search-tree based approach is different from the greedy-based approach. The greedy-based approach uses the information of the total number of links that each SDN switch can protect to choose the optimal SDN switch deployment location. However, the search-tree based approach uses the information of the number of SDN switches which can protect the failure of a certain link to choose the optimal deployment location for each SDN switch. The search-tree based SCS algorithm includes two phases: (1) information collection and (2) search-tree construction. In the following, we will introduce these two phases in detail.

In the information collection phase, based on the candidate table, we calculate the cover set for each link. The cover set is defined as the set of SDN switches which can protect the failure of a certain link (such as link *e*), denoted as C_e. For example, as the candidate table that shown in Table 1, the cover set of link_11 is $C_{11} = (8)$, the cover set of link_1 is $C_1 = (1,7,8,9,10)$, etc. The number of elements in the cover set of link *e* is defined as the reliability degree of link *e*, denoted as d_e. For instance, in Table 1, the reliability degrees of the link_11, link_12, and link_14 are all 1; the reliability degrees of the link_8, link_23, and link_26 are all 2; the reliability degree of link_7 is $d_7 = 3$, etc. The cover set and reliability degree of each link is constructed in Table 3.

Table 3. Cover Set and Reliability Degree

Link failure	Cover set	Reliability degree
1, <1,2>	1, 7, 8, 9, 10	5
2, <1,10>	1, 2, 3, 4, 5, 6	6
3, <2,1>	2, 4, 5, 6, 7, 8, 9, 10	8
4, <2,10>	1, 2, 3, 4, 5, 6	6
5, <2,3>	7, 8, 9, 10	4
6, <3,2>	4, 5, 6, 7, 8, 9	6
7, <3,4>	1, 2, 10	3
8, <4,3>	9, 10	2
9, <4,5>	9	1
10, <4,9>	1, 2, 3, 5, 6, 7, 8	7
11, <5,4>	8	1
12, <5,6>	8	1
13, <5,8>	1, 2, 3, 4, 6, 7, 9, 10	8
14, <6,5>	7	1
15, <6,7>	1, 2, 3, 4, 5, 8, 9, 10	8
16, <7,6>	1, 2, 7, 8, 9, 10	6
17, <7,8>	3, 4, 5, 6, 7	5
18, <8,5>	1, 2, 7, 8, 9, 10	6
19, <8,7>	2, 3, 4, 5, 6, 8, 9	7
20, <8,9>	3, 4, 5, 6, 7, 8	6
21, <9,4>	1, 2, 7, 8, 9, 10	6
22, <9,8>	2, 3, 4, 5, 6, 9	6
23, <9,10>	3, 4	2
24, <10,1>	2, 3, 4, 5, 6, 7, 8, 9, 10	9
25, <10,2>	1, 4, 5, 6, 7, 8, 9, 10	8
26, <10,9>	2, 3	2

After the construction of the cover set and reliability degree of each link, the next step is to construct the search tree based on these two parameters. When constructing the search tree, the most important thing is to choose the starting point, i.e., which candidate deployment locations should be the elements in the first level of the search tree. In this paper, considering the purpose of deploying the SDN switches is to protect all links in the network, and different links have different cover set and reliability degree, we give the starting point selection principle as: the link whose reliability degree is the smallest has the highest priority as the starting point, and the smaller reliability degree, the higher priority is. For instance, since the reliability degree of link_11 is $d_{11} = 1$ and $C_{11} = (8)$, column (8) must be chosen. This is because link_11 is protected by only one SDN switch (8), if column (8) is not chosen, when link_11 fails, it cannot be recovered by the hybrid IP/SDN network.

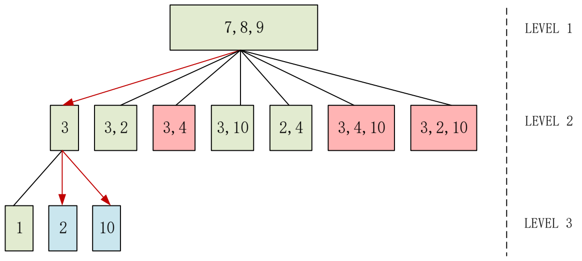

Fig. 3 Search tree of SCS

As shown in Table 3, the reliability degrees of link_9, link_11, link_12, and link_14 are all 1, the cover sets of link_9 is (9), the cover sets of link_11 and link_12 are (8), and the cover set of link_14 is (7), based on the principle introduced above, (7,8,9) should be the elements in the first level of the search tree. This can be found in Fig. 3. When the elements in the first level of the search tree are selected, we need to judge whether these nodes are already can protect all link failure in the network. Since (7,8,9) cannot protect all the link failure, the second level of the search tree is needed.

Since (7,8,9) is selected, the link_9, link_11, link_12, and link_14 have been protected. However, since the (7,8,9) cannot protect all link failures, the next step is to analyze the link whose reliability degree is 2, i.e., the link_8, link_23, and link_26. As shown in Table 3, the cover sets of these three links are $C_8 = (9{,}10)$, $C_{23} = (2{,}3)$, and $C_{26} = (3{,}4)$, respectively.

Based on the cover sets of these three links, the column 2, 3, 4, 9, 10 will be selected to construct the second level of the search tree. Note that not all these five columns are selected at the same time, our purpose is to choose the minimum number of columns from column 2, 3, 4, 9, 10 to protect the failure of link_8, link_23, and link_26. So, the selection principle in the second level should be: the selected columns should be able to protect all link failures of link_8, link_23, and link_26. Since node 9 is already chosen in the first level, it will be not considered in the second level. Thus, based on the node selection principle, the elements in the second level are: (3), (3,2), (3,4), (3,10), (2,4), (3,4,10), (3,2,10) and (2,4,10), which are shown in Fig. 3. The calculation of the elements in the second level is also the search-tree based approach. Since the reliability degrees of link_8, link_23, and link_26 are all 2, we can start from any link. As shown in Fig.4, assuming that we start from link_8, two search trees starting from column 9 and column 10 are presented. In Fig. 4, the red element means this element will be deleted; the blue element means repetition.

Similarly to the first level, when the elements in the second level are calculated, we will judge whether the selected columns (both in level 1 and level 2) can protect all link failures or not. For the elements in the second level which are gotten from the search tree shown in Fig.4(a), the feasible solutions are (7,8,9,2,3) and (7,8,9,2,4). The solution (7,8,9,3) and (7,8,9,3,4) cannot protect all link failures. Intuitively, we need to investigate the third level of solution (7,8,9,3) and (7,8,9,3,4). However, considering (7,8,9,2,3) and (7,8,9,2,4) can protect all link failures, and the number of SDN switches is 5, so (7,8,9,3,4) will be not considered anymore; because the number of SDN switches of solution (7,8,9,3,4) must be larger than 5 when takes the third level into consideration. Similarly, the feasible solutions in Fig.4(b) are (7,8,9,3,10), (7,8,9,3,4,10), (7,8,9,2,3,10), and (7,8,9,2,4,10). However, since (7,8,9,2,3) and (7,8,9,2,4) are already the feasible solutions, the (7,8,9,2,3,10) and (7,8,9,2,4,10) are not considered any more. The (7,8,9,3,10) and (7,8,9,3,4,10) are the feasible solutions. However, even (7,8,9,3,4,10) is the feasible solution, it is not considered due to the number of SDN switches. As shown in Fig. 3, the green elements mean the reserved feasible solutions; the red elements have the same meaning as Fig. 4. Therefore, only (7,8,9,3) need to be investigated in the third level.

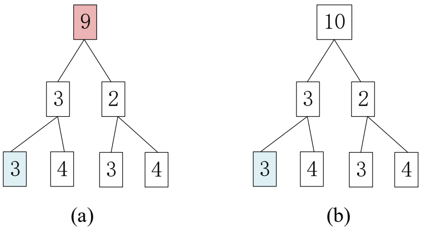

Fig.4 Search tree of the elements in the second level

For the third level, only the reliability degree of link_7 is 3, and the cover set of link_7 is (1,2,10). Therefore, the feasible solutions takes the third level into consideration are: (7,8,9,3,1), (7,8,9,3,2), and (7,8,9,3,10). Since (7,8,9,3,10) and (7,8,9,3,2) are repeated with the solutions in level_2, it will be deleted. Similarly, (7,8,9,3,1) can cover all link failures and the number of SDN switches is also 5, so (7,8,9,3,1) is the feasible solution. This can be found in Fig. 3.

Since to level_3, all the leaf nodes in the search tree (except the red nodes) can protect all link failures, for reducing computation complexity, we do not need to investigate the fourth level anymore. Even the solutions in the higher level also can protect all link failures, the numbers of SDN switches is larger than 5. Therefore, we can give the stop condition of the searching as: if all the leaf nodes in the search tree can protect the link failure in the network, the searching will be stopped. All the feasible solutions in the search tree form the feasible solution set; the number of SDN switches in the feasible solution set is the smallest and the SDN switches in the feasible solution set can protect all link failures in the network.

B. *Most Appropriate Solution Selection Algorithm*

In Section III.A, we construct the feasible solution set based on the search tree; in this section, we will choose the most appropriate solution from the feasible solution set. The selection is based on the reliability degree of each solution. Firstly, we define the reliability degree of feasible solution as follows.

Definition 1. The reliability degree of feasible solution i is defined as the number of SDN switches which can protect more than one link failures in i, denoted as rd_i.

For instance, as shown in Table 1 and Table 3, based on the feasible solution and the reliability degree of each link, we can construct the reliability degree vectors for each feasible solution as below:

$v_1 = (4,2,3,2,3,3,1,1,1,4,1,1,4,1,4,3,2,3,3,3,4,2,1,4,4,1)$
$v_2 = (3,2,4,2,3,3,1,1,1,4,1,1,4,1,4,3,2,3,4,3,4,3,1,5,3,2)$
$v_3 = (4,1,4,1,4,3,1,2,1,3,1,1,4,1,4,4,2,3,3,3,4,2,1,5,4,1)$
$v_4 = (3,2,5,2,3,4,1,1,1,3,1,1,4,1,4,4,2,3,4,3,4,3,1,5,4,1)$

where v_1 is the reliability degree vector of $(7,8,9,3,1)$, v_2 is the reliability degree of $(7,8,9,3,2)$, v_3 is the reliability degree of $(7,8,9,3,10)$, and v_4 is the reliability degree of $(7,8,9,2,4)$, respectively. Thus, according to Definition 1, the reliability degree of these four feasible solutions are $rd_1 = 18$, $rd_2 = 19$, $rd_3 = 17$, and $rd_4 = 18$, respectively. When there is a tie, such as rd_1 and rd_4, the average reliability degree of link will be used to decided which solution is better than another. For instance, for rd_1 and rd_4, since the reliability degrees of these two feasible solutions are the same and the average reliability degrees of link are 2.54 and 2.7, respectively, rd_4 has higher priority to be chosen than rd_1.

C. Most Appropriate SDN Switch Selection Algorithm

In Section III.A, we construct the feasible solution set based on the search tree; in Section III.B, we choose the most appropriate solution from the feasible solution set. In this section, we will investigate how to choose the most appropriate designated SDN switch for each link.

As presented in the reliability degree vectors v_1, v_2, v_3, and v_4, for each feasible solution, many links' reliability degrees are larger than 1. This means that for each link, there are more than one designated SDN switches can be chosen, which is shown in Fig.2. The properties of these SDN switches are different, so we need to choose the most appropriate SDN switch for each link. In this paper, we use the average repair path length and the load balancing degree of SDN switch as the performance metrics. These two performance metrics are defined as follows.

Average repair path length. For link *e*, we assume that there are *n* SDN switches that can protect it, and the number of affected destinations is *m*. So, for SDN switch *i*, the repair path length is the number of hops from the failed link *e* to its affected destination *j* through SDN *i*. Since there are more than one affected destinations for link *e*, the average repair path length of these affected destination related to SDN switch *i* is calculated as:

$$d_e^i = \sum_{j=1}^{m} d_{e,j}^i / m \tag{1}$$

where $d_{e,j}^i$ is the repair path length when link *e* fails and reroutes the packet data to the affected destination *j* through SDN switch *i*; moreover, $0 < i \le n$ and $0 < j \le m$.

Load balancing degree. When link *e* fails and the traffic load in link *e* is *k*, for a certain affected destination *j* and a certain SDN switch *i*, the path which has the lowest link utilization can be found based on the tree based load balancing approach [10], which is denoted as $r_{e,j}^i$. Similarly, we assume that there are *n* SDN switches can protect link *e* and the number of affected destination is *m*. Therefore, the average minimum link utilization of SDN switch *i* can be calculated as:

$$r_e^i = \sum_{j=1}^{m} r_{e,j}^i / m \tag{2}$$

where $0 < i \le n$ and $0 < j \le m$.

When the average repair path length and the average link utilization of SDN switch are calculated, the principle is that the SDN switch which has the smallest average repair path length and the average link utilization should be selected as the most appropriate designated SDN switch. However, this target is hard to be achieved in practice. The most common situation in practice is that for one SDN switch, its average repair path length is small while its average link utilization is high, vice versa. Thus, we need to achieve a tradeoff between these two performance metrics, i.e., the selected SDN switch should have high quality of performance on both average repair path length and link utilization.

In this paper, we propose to use the weight based multi-attribute decision making (MADM) approach to achieve this purpose. The MADM approach is effective on dealing with this kind of issue [12]. For the weight based MADM approach, the first important thing is to calculate the weight of each parameter. Considering the fact that the performance metric whose variance is larger has a higher effect on selection than the parameter whose variance is smaller [12], we propose to use the variance of each parameter to calculate the weights of performance metrics.

However, on one hand, due to the order-of-magnitude of these two performance metrics are different, it is not appropriate to use the values of d_e^i and r_e^i directly in MADM [13]. Therefore, the values of d_e^i and r_e^i should be normalized before used. For the MADM approach, there are many effective parameter normalized algorithms, such as SAW and WP [13]. Assuming that d_e^{i*} and r_e^{i*} are the two parameters after normalization, then the variances of these two parameters are v_d^* and v_r^*, respectively. On the other hand, in the weight-based MADM approach, the sum of all the weights of different parameters should equal to 1 [13]. Thus, based on the variances, the weights can be calculated as: $\omega_d^* = \frac{v_d^*}{v_d^* + v_r^*}$ and $\omega_r^* = \frac{v_r^*}{v_d^* + v_r^*}$, respectively. Therefore, the utility of each feasible SDN switch is:

$$p_i = \omega_d^* d_e^{i*} + \omega_r^* r_e^{i*} \tag{3}$$

where ω_d^* is the weight of the average repair path length, d_e^{i*} is the average repair path length of *ith* SDN switch, ω_r^* is the weight of average link utilization, r_e^{i*} is the average link utilization of *ith* SDN switch. Based on the utilities of different SDN switches, the SDN switch whose utility is the largest will be chosen as the most appropriate designated SDN switch.

IV. Performance Evaluation

In this section, we compare the proposed search-tree based SCS algorithm with the greedy-based algorithm (i.e., the algorithm presented in [10]) by simulation. In this simulation, four performance metrics are investigated under different network topologies: the number of SDN switches, the reliability degree, the average repair path length, and the link utilization. The network topologies used in this paper are similar to that in [10] and [11], which are presented in Table 4.

Table 4. The different network topologies

Topology	Number of nodes	Number of links
NSFNet	14	21
Abilene	11	14
Internet2	10	13

A. Number of SDN Switches

Table 5. The number of SDN switches under different network topologies

	NSFNet	Abilene	Internet2	40-node ER (0.1)
Greedy-based	3	5	5	4.4
Search-tree based	3	5	5	4.1

The simulation results are presented in Table 5. In this simulation, except for the three topologies in Table 4, we also investigate the performance under random topologies that obtained by the Erdos-Renyi (ER) generator, in which the probability of two arbitrary nodes are directly connected is 0.1. From Table 5, we can conclude that the numbers of SDN switches calculated by greedy-based approach and search-tree based approach are similar. Even in random topology, the number of SDN switches calculated by the search-tree based approach is slightly smaller than that in greedy-based approach. This is because the purpose of the search-tree based approach is not to minimize the number of SDN switches further compared with the greedy-based approach. The purpose is to find all the feasible solutions and select the most appropriate one based on the feasible solutions' properties, such as reliability degree, link utilization, etc.

B. Reliability Degree

Table 6. The reliability degree under different network topologies

	NSFNet	Abilene	Internet2	40-node ER (0.1)
Greedy-based	13	6	5	11
Search-tree based	15	9	9	14

The simulation results of the reliability degree are presented in Table 6. From Table 6, we can find that the reliability degree calculated by the search-tree based approach is better than the greedy-based approach. For instance, in the NSFNet, the reliability degree calculated by the greedy-based approach is 13, while this value is 15 in the search-tree based approach; the value of the search-tree based approach is 15.4% larger than the greedy-based approach. These values are 50%, 80%, and 27.3% in Abilene, Internet2, and 40-node ER, respectively. This is because during the most feasible solution selection, the search-tree based approach takes the reliability degree into account, while the greedy-based approach is not.

C. Average repair path length and Link Utilization

Table 7. Average repair path length and link utilization under different network topologies

	NSFNet		Abilene		Internet2		40-node ER (0.1)	
	d_e^i	r_e^i	d_e^i	r_e^i	d_e^i	r_e^i	d_e^i	r_e^i
Greedy-based	4.42	0.897	4.22	0.92	4.24	0.915	4.5	0.886
Search-tree based	4.18	0.82	3.89	0.837	3.91	0.741	4.23	0.802

The simulation results of link utilization and average repair path length are presented in Table 7. From Table 7 we can find that the search-tree based approach has better performance on repair path length and link utilization than the greedy-based approach. For instance, the average repair path length and the link utilization in NSFNet are 4.42 and 0.897 based on the greedy-based approach, while these two values are 4.18 and 0.82 based on the search-tree based approach. Even the average repair path length is considered in [11] and the link utilization is considered in [10], on one hand, these two parameters are considered separately; on the other hand, the average repair path length calculated in [11] relates to all the affected destinations of the selected column in Table 1 and the link utilization calculated in [10] is the minimum link utilization by randomly selecting SDN switch. Therefore, they cannot reflect the properties of each candidate SDN switch.

V. Conclusion

In this paper, considering the disadvantages of the greedy-based approach, we proposed the search-tree based SDN candidate selection (SCS) algorithm. In this algorithm, for achieving better performance than the greedy-based approach, the search-tree based feasible solutions calculation algorithm, the most appropriate feasible solution selection algorithm, and the most appropriate designated SDN switch selection algorithm are proposed. Based on these algorithms, the performance of the search-tree based SCS algorithm is improved greatly compared with the greedy-based algorithms. Due to the limitation of space, the computation complexity will be analyzed in future work.

References

[1] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast ip network recovery using multiple routing configurations," in Proceedings of IEEE INFOCOM. IEEE, 2006.

[2] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," ACM SIGMETRICS Performance Evaluation Review, 2011.

[3] M. Reitblatt, M. Canini, A. Guha, and N. Foster, "Fattire: declarative fault tolerance for software-defined networks," in Proceedings of ACM SIGCOMM workshop on HotSDN, 2013.

[4] A. Atlas and A. Zinin, Basic Specification for IP Fast Reroute: Loop-Free Alternates, document RFC 5286, Sep. 2008.

[5] S. Bryant, C. Filsfils, S. Previdi, and M. Shands, IP Fast Reroute Using Tunnels, document Draft-Bryant-Ipfrr-Tunnels-03, IETF Draft, Sep. 2007.

[6] C. Perkins, IP Encapsulation Within IP, document RFC 2003, Oct. 1996.

[7] M. Shand and S. Bryant, IP Fast Reroute Framework, document RFC 5714, Jan. 2010.

[8] S. Nelakuditi, S. Lee, Y. Yu, Z. L. Zhang, and C. N. Chuah, "Fast local rerouting for handling transient link failures," IEEE/ACM Transactons on Networking, vol. 15, no. 2, 2007, pp: 359–372.

[9] S. Antonakopoulos, Y. Bejerano, and P. Koppol, "Full protection made easy: The DisPath IP fast reroute scheme," IEEE/ACM Transactons on Networking, vol. 23, no. 4, 2015 pp: 1229–1242.

[10] C.Y. Chu, K. Xi, M. Luo, H.J. Chao, "Congestion-Aware Single Link Failure Revovery in Hybrid SDN Networks," in Proceedings of IEEE INFOCOM, 2015.

[11] Z. Yang, K.L. Yeung, "SDN Candidate Selection in Hybrid IP/SDN Networks for Single Linke Failure Protection," IEEE/ACM Transactions on Networking, vol.28, no.1, 2020, pp: 312-321.

[12] N. Li, J.F. Martinez, V.H. Diaz, J.A. Sanchez, "Probability Prediction-Based Reliable and Efficient Opportunistic Routing Algorithm for VANETs," IEEE/ACM Transactions on Networking, vol. 26, no. 4, 2018, pp: 1933-1947.

[13] K. Yoon, C.L. Hwang, "Multiple Attritube Decision Making Introduction," Sage Publication, 1995.