PAPER *Special Section on Mathematical Systems Science and its Applications*

Two-Phase Approach to Finding the Most Critical Entities in Interdependent Systems

Daichi MINAMIDE[†], ***Nonmember and*** **Tatsuhiro TSUCHIYA**[†a)], ***Member***

SUMMARY In interdependent systems, such as electric power systems, entities or components mutually depend on each other. Due to these interdependencies, a small number of initial failures can propagate throughout the system, resulting in catastrophic system failures. This paper addresses the problem of finding the set of entities whose failures will have the worst effects on the system. To this end, a two-phase algorithm is developed. In the first phase, the tight bound on failure propagation steps is computed using a Boolean Satisfiablility (SAT) solver. In the second phase, the problem is formulated as an Integer Linear Programming (ILP) problem using the obtained step bound and solved with an ILP solver. Experimental results show that the algorithm scales to large problem instances and outperforms a single-phase algorithm that uses a loose step bound.

key words: *interdependent networks, cascading failures, satisfiability, integer programming, critical entities*

1. Introduction

Cyber-physical systems, typically electric power systems, can often be regarded as *interdependent systems* [1]–[3]. In an interdependent system, entities of the system mutually depend on each other in a complex manner. For example, the control center of an electric power system requires electricity, which is provided by the power network. On the other hand, the power network needs the control center together with the communication network. In such a system, a small number of initial failures can lead to a catastrophic system failure as a result of cascading failure propagation. This paper considers failures in interdependent systems and addresses the problem of identifying the most vulnerable entities whose failures result in the worst-case scenario.

As a model of interdependent systems, we adopt the *implicative interdependency model* proposed and adopted by a series of works [4]–[7]. The authors of this series of work showed that many practical problems under this model are NP-hard, which means that it is highly unlikely that polynomial-time algorithms exist for these problems.

The problem that we address in this paper is one of these problems. Specifically, we consider the *$\mathcal{K}$-most vulnerable node problem* [4]. The goal of this problem is to find the $\mathcal{K}$ entities whose initial failures cause the maximum number of failures in the end. This problem is important because identifying the most vulnerable part of a system is critical to cope with the worst case scenario and to design plans to strengthen the system.

Although the problem is NP-hard, recent advances in mathematical programming solvers enable us to deal with many instances of NP-hard problems in practice, and thus we base our approach on modern mathematical programming solvers. Specifically, we propose a two-phase algorithm that uses a Boolean Satisfiability (SAT) solver and an Integer Linear Programming (ILP) solver. In the first phase, the tight bound on failure propagation steps is computed using an SAT solver. In the second phase, the problem is formulated as an ILP problem using the obtained step bound and solved with an ILP solver.

The use of ILP solving was already proposed in the previous work [4]; but the previous approach did not consider the bound on the failure propagation steps that can occur. Instead, this previous study used a trivial upper bound in reducing the $\mathcal{K}$-most vulnerable node problem to ILP. This requires the IPL solver to analyze an unnecessarily large number of steps, resulting in a large computation time. Our approach makes use of an SAT solver to obtain the maximum possible failure propagation steps, which leads to a much more compact ILP formulation. As a result, a significant speedup is achieved for ILP solving which easily offsets the computation cost for SAT solving.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 describes the implicative interdependency model for interdependent systems. Section 4 provides the definition of the problem that is addressed and an overview of the proposed algorithm. Section 5 and Sect. 6 describe Phase 1 and Phase 2 of the algorithm, respectively. Section 7 shows experimental results that we obtained by applying the algorithm to some problem instances. Section 8 concludes this paper.

2. Related Work

Cascading failures have sometimes been observed in cyber-physical systems, such as electric power systems. The approach of treating such systems as interdependent systems has received recent attention, since it enables us to account for cascading failures that would be otherwise impossible to explain. Many models of interdependent systems have been proposed so far. A well-known model proposed by [2] treats an interdependent system as a pair of networks which mutually depend on each other and provides a graph-theoretic interpretation of cascading failures. For example, a power

Manuscript received April 17, 2023.
Manuscript revised August 16, 2023.
Manuscript publicized September 20, 2023.

[†]The authors are with Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: t-tutiya@ist.osaka-u.ac.jp

DOI: 10.1587/transfun.2023MAP0003

system can be regarded as a composition of a power network and a Supervisory Control And Data Acquisition (SCADA) network. Later studies that extended or generalized this model include [3], [8], [9].

Problems similar to the $\mathcal{K}$-most vulnerable node problem can be found in several studies. For example, the work [10] assumed the model of [2] and addressed a problem they named the interdependent power network disruptor problem. The goal of this problem is to find k nodes whose removal minimizes the size of the largest connected component remaining in the network after cascading failures. Other studies include [11] and [12].

The application of an SAT solver to the analysis of cascading failures is studied in [13], [14]. In particular, [14] is similar to this study in that [14] proposes a two-phase approach using SAT and IPL solvers to find the most critical nodes in interdepenent networks. These previous studies are built on the model of [2], instead of the implicative interdependency model which we adopted here. Because the two underlying models are substantially different, the formulations of SAT and IPL are totally different from those presented in this paper.

The studies most related to this paper are [4]–[7] where the implicative interdependency model is adopted. In particular, [4] introduced the $\mathcal{K}$-most vulnerable node problem and proposed an ILP approach to the problem. (The ILP formulation is described in [7] in more detail.) Our ILP formulation is different from that of the previous study not only in that ours makes use of the value of m, but also in that the constraint formulas are slightly different. We made some modifications to them because it was difficult for us to see that the original ILP formulation faithfully captures the implicative interdependency model.

3. Model of Interdependent Systems

An interdependent system consists of n entities that depend on each other. We denote the i-th entity simply by $i \in \{1, 2, \ldots, n\}$ unless otherwise mentioned. The state of an entity is either *operational* or *failed* at a given time. Each entity is associated with a logic formula, which we call a *dependency formula*, that represents such dependency[†]. The dependency formula is in the form of a sum of products of symbols, where symbols represent entities. These products are called *min-terms*. The dependency formula for entity i is

$$e_{i,1,1}e_{i,1,2}\ldots e_{i,1,p_{i,1}} + \cdots + e_{i,t_i,1}e_{i,t_i,2}\ldots e_{i,t_i,p_{i,t_i}}, \quad (1)$$

where t_i is the number of min-terms and $p_{i,l}$ is the number of symbols in the l-th min-term.

The formula means that the entity i relies on those that appear in the formula and that i requires that all entities of at least one of the min-terms be operational. We assume that failures propagate in steps; an entity will *fail* in step j, that is, transit from the operational state to the failed state if, for every min-term, at least one entity in the min-term failed at the end of step $j-1$.

For example, suppose that an entity a is associated with formula $b1b2b3 + b4b5 + b6$, then a will fail in step j if in step $j-1$: 1) $b1$, $b2$, or $b3$ failed, 2) $b4$ or $b5$ failed, and 3) $b6$ failed.

For another example, consider a system which is taken from [4]. The entities of the example are: $a1, a2, a3, b1, b2, b3, b4$. The dependency formulas for these entities are as follows.

$$\begin{array}{lcc} a1 & :: & b2 \\ a2 & :: & b2 \\ a3 & :: & b4 \\ b1 & :: & a1 + a2 \\ b2 & :: & a1a2 \\ b3 & :: & a2 + a1a3 \\ b4 & :: & a3 \end{array}$$

	0	1	2	3	4	5
$a1$	O	O	F	F	F	F
$a2$	F	F	F	F	F	F
$a3$	F	F	F	F	F	F
$b1$	O	O	O	F	F	F
$b2$	O	F	F	F	F	F
$b3$	O	F	F	F	F	F
$b4$	O	F	F	F	F	F

Fig. 1 Failure propagation when $a2$ and $a3$ are initially failed. An O or F represents that an entity is operational or failed, respectively.

Figure 1 shows how failures propagate in the system when $a2$ and $a3$ initially failed, that is, failed in step 0. These failures cause the failures of $b2$, $b3$, and $b4$ in step 1. In step 2, $a1$ fails because $b2$ failed. In step 3, $b1$ fails because both $a1$ and $a2$ failed. In step 4 or later steps, no entity changes its state because all entities have already failed.

We say that the system reaches the steady state in step j if the last failure occurs in step j. In the above example, the system reaches the steady state in step 3.

4. Problem and Our Approach

As mentioned earlier, we consider the problem of determining the most vulnerable entities in a system. In [4], this problem is called the $\mathcal{K}$-most vulnerable node problem.

The $\mathcal{K}$-most vulnerable node problem: Given an integer $\mathcal{K}$, find a set of $\mathcal{K}$ entities such that the failures of these entities at step 0 result in the greatest number of failed entities when the steady state is reached.

In [4], the decision version of the $\mathcal{K}$-most vulnerable node problem is proved to be NP-complete, which means that its optimization version, that is, the $\mathcal{K}$-most vulnerable node problem is NP-hard. The NP-hardness implies that it is unlikely that an polynomial-time algorithm exists for the problem. Generally, two common approaches are available to tackle NP-hard problems. One approach is to use a heuristic algorithm that is efficient but cannot compute optimal solutions in general. In this paper, we take the other

[†]The formula is called a *live equation* in [4] and a *dependency equation* in [7].

approach: We devise an algorithm that can always find optimal solutions. Such an algorithm cannot avoid exponential time complexity in the worst case; therefore the goal of the proposed algorithm is to compute optimal solutions in reasonable time for practical problem instances.

The proposed approach consists of two phases. In the first phase, we determine the maximum possible number of the step in which the steady state is reached when the number of initially failed entities is $\mathcal{K}$. Let m denote this maximum number. Our algorithm computes this value by repeatedly executing a Boolean satisfiability (SAT) solver.

In the second phase, the $\mathcal{K}$ most vulnerable node problem is formulated into an ILP problem and in turn solved by an ILP solver. The value of m is used in the ILP formulation.

SAT and the decision version of ILP are both NP-complete; thus, any solvers for these problems have exponential time complexity. However, modern solvers integrate many optimization techniques and are usually able to handle very large problems.

5. Phase 1: Computing the Maximum Steps of Failure Propagation

Phase 1 of the proposed approach is used to compute m, the maximum possible step number in which failure propagation stops when there are initially exactly $\mathcal{K}$ failed entities in step 0. We address this problem using a Boolean satisfiability (SAT) solver. SAT is the problem of determining if a given Boolean formula is *satisfiable*, i.e., if there is at least one truth value assignment to the Boolean variables that makes the formula evaluate to true. For example, $(x \vee y) \wedge (\neg x \vee \neg z)$ is satisfiable because it has a satisfying value assignment to the variables, such as $x = true, y = true, z = false$.

SAT is an NP-complete problem and, thus no polynomial-time algorithm exists unless P = NP. However, recent SAT solvers can often solve large problem instances very fast, thanks to advanced optimization techniques. We will show below the problem of deciding if the steady state is reached by step j can be reduced to SAT. Repeatedly solving SAT instances with j being increased, we can obtain m, i.e., the possible maximum number of failure propagation steps.

Now we show how to construct a Boolean formula that is satisfiable if and only if the steady state may NOT be reached by step j. The formula is obtained by unwinding the state transition relation of the given interdependent system. The Boolean variables that appear in the formula are $x_i^{(j)}$, where i corresponds to an entity and j represents the step number. The truth value of $x_i^{(j)}$ represents the state of entity i at the end of step j: if it is true, then entity i is operational at the end of step j; otherwise i is failed. The formula, denoted as $M^{(j)}$, is in the form as follows:

$$M^{(j)} := \begin{cases} I & j = 0 \\ M^{(j-1)} \wedge T^{(j-1,j)} \wedge P^{(j)} & j \geq 1 \end{cases} \tag{2}$$

The ingredients of this formula are explained below.

I is used to represent the condition for initial failures.

$$I := Card(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}, n - \mathcal{K}) \tag{3}$$

where $Card()$ is a *cardinality constraint* [15] and evaluates to true if and only if exactly $n - \mathcal{K}$ variables are assigned true. For example, $Card(x, y, z, 2)$ could be $xy\neg z \vee x\neg yz \vee \neg xyz$. Cardinality constraints have been widely studied in the field of SAT theories, and many different encodings are known. Cardinality constraints play an important role in many SAT instances (i.e., Boolean formulas) arising from practical applications [15].

$T^{(j-1,j)}$ represents the state transition from step $j-1$ to step j. $T^{(j-1,j)}$ is a conjunction of Boolean formulas each of which determines the state of each entity. Note that the dependency formula for entity i is of the form:

$$e_{i,1,1}e_{i,1,2} \ldots e_{i,1,p_{i,1}} + \cdots + e_{i,t_i,1}e_{i,t_i,2} \ldots e_{i,t_i,p_{i,t_i}} \tag{4}$$

The state of an entity i is represented as follows.

$$\begin{aligned} &x_i^{(j)} \Leftrightarrow \\ &\Big(x_i^{(j-1)} \wedge \\ &(x_{e_{i,1,1}}^{(j-1)} \ldots x_{e_{i,1,p_{i,1}}}^{(j-1)} \vee \cdots \vee x_{e_{i,t_i,1}}^{(j-1)} \ldots x_{e_{i,t_i,p_{i,t_i}}}^{(j-1)})\Big) \end{aligned} \tag{5}$$

This formula directly represents the dependency formula: it means that i is operational at the end of step j if and only if i is operational in step $j-1$, and all entities in one of the min-terms of the dependency formula are also operational. $T_i^{(j-1,j)}$ is a conjunction of the formulas each corresponding to an entity.

$P^{(j)}$ is a Boolean formula that evaluates to true if at least one entity changes its state from operational to failed in step j. The formula is as follows:

$$P^{(j)} := \bigvee_{1 \leq i \leq n} \neg x_i^{(j)} x_i^{(j-1)} \tag{6}$$

The conjunct $\neg x_i^{(j)} x_i^{(j-1)}$ for each entity i represents that the entity is operational at the end of step $j-1$ (represented by $x_i^{(j-1)}$) but is failed at the end of step j (represented by $\neg x_i^{(j)}$).

As a result, it can be seen that $M^{(j)}$ is satisfiable if and only if a set of $\mathcal{K}$ entities exists whose initial failures cause some entity to fail in step j.

Shown below are the formulas I, $T^{(0,1)}$, and $P^{(1)}$ for the running example presented in Sect. 3.

$$\begin{aligned} &I := Card(x_{a1}^{(0)}, x_{a2}^{(0)}, x_{a3}^{(0)}, x_{b1}^{(0)}, x_{b2}^{(0)}, x_{b3}^{(0)}, x_{b4}^{(0)}, 5) \\ &T^{(0,1)} := \Big(x_{a1}^{(1)} \Leftrightarrow (x_{a1}^{(0)} \wedge x_{b2}^{(0)})\Big) \\ &\wedge \Big(x_{a2}^{(1)} \Leftrightarrow (x_{a2}^{(0)} \wedge x_{b2}^{(0)})\Big) \\ &\wedge \Big(x_{a3}^{(1)} \Leftrightarrow (x_{a3}^{(0)} \wedge x_{b4}^{(0)})\Big) \\ &\wedge \Big(x_{b1}^{(1)} \Leftrightarrow (x_{b1}^{(0)} \wedge (x_{a1}^{(0)} \vee x_{a2}^{(0)}))\Big) \\ &\wedge \Big(x_{b2}^{(1)} \Leftrightarrow (x_{b2}^{(0)} \wedge (x_{a1}^{(0)} x_{a2}^{(0)}))\Big) \end{aligned} \tag{7}$$

```
$M^{(0)} \leftarrow I$
for $j = 1, 2, \cdots n$ do
    $M^{(j)} \leftarrow M^{(j-1)} \wedge P^{(j-1)}$
    Check satisfiability of $M^{(j)}$
    if $M^{(j)}$ is unsatisfiable then
        Break the for loop
    end if
end for
return $j - 1$
```

Fig. 2 Algorithm for computing m.

$$\wedge\Big(x^{(1)}_{b3} \Leftrightarrow (x^{(0)}_{b3} \wedge (x^{(0)}_{a2} \vee x^{(0)}_{a1}x^{(0)}_{a2}))\Big)$$
$$\wedge\Big(x^{(1)}_{b4} \Leftrightarrow (x^{(0)}_{b4} \wedge x^{(0)}_{a3})\Big) \tag{8}$$
$$P := \neg x^{(1)}_{a1}x^{(0)}_{a1} \vee \neg x^{(1)}_{a2}x^{(0)}_{a2} \vee \neg x^{(1)}_{a3}x^{(0)}_{a3}$$
$$\vee\neg x^{(1)}_{b1}x^{(0)}_{b1} \vee \neg x^{(1)}_{b2}x^{(0)}_{b2} \vee \neg x^{(1)}_{b3}x^{(0)}_{b3} \vee \neg x^{(1)}_{b4}x^{(0)}_{b4} \tag{9}$$

Here, $\mathcal{K}$ is set to two. For this example, $M^{(1)}$ is satisfiable. One of the satisfying value assignments to the variables is:

$$\begin{array}{lll} x^{(0)}_{a1} = true & x^{(0)}_{a2} = false & x^{(0)}_{a3} = false \\ x^{(0)}_{b1} = true & x^{(0)}_{b2} = true & x^{(0)}_{b3} = true \\ x^{(0)}_{b4} = true & & \end{array}$$

$$\begin{array}{lll} x^{(1)}_{a1} = true & x^{(1)}_{a2} = false & x^{(1)}_{a3} = false \\ x^{(1)}_{b1} = true & x^{(1)}_{b2} = false & x^{(1)}_{b3} = false \\ x^{(1)}_{b4} = false & & \end{array}$$

This assignment corresponds to the failure propagation pattern shown in Fig. 1.

Because of this property, the greatest step number when failure propagation stops can be obtained by repeatedly checking the satisfiability of $M^{(j)}$ with j being gradually increased. This algorithm is shown in Fig. 2. It repeats the satisfability check for $M^{(j)}$ with j being gradually increased. When $M^{(j)}$ turns out to be unsatisfiable, the algorithm will return $j-1$ as the value of m because in that case some entity can fail in step $j-1$, but none can in step j.

6. Phase 2: Finding the Most Vulnerable Entities

Phase 2 computes the most vulnerable entities that lead to the maximum number of failed entities in the steady state. This problem is directly reduced to a 0-1 ILP problem.

The idea behind this ILP formulation is similar to the SAT encoding. We represent step-wise state transitions using variables that represent entities' state in different steps. The most significant differences are that all variables in the ILP problem are of integer type, not Boolean, and that only a conjunction of linear arithmetic constraints can be used to define the feasible solution space. Below we show how to formulate the problem using only linear arithmetic constraints over 0-1 integer variables.

We let $X^{(j)}_i$ denote these 0-1 integer variables. The variable $X^{(j)}_i$ represents the state of entity i at the end of step j. The state is failed if the value of the variable is 1; operational, otherwise. (Note that in the SAT encoding described in Sect. 5, we use the values $true$ and $false$ to represent the state of being operational and failed, respectively.)

The objective function is:

$$\text{Maximize} \quad X^{(m)}_1 + X^{(m)}_2 + \cdots + X^{(m)}_n \tag{10}$$

Clearly, the value of the objective function represents the number of failed entities at the end of step m.

Once the optimal solution to the ILP problem was obtained, the solution to the $\mathcal{K}$-most vulnerable node problem can be directly obtained from it. When the values of $X^{(j)}_i$ represent the optimal solution to the ILP problem, the optimal solution to the original problem is as follows:

$$\{i \mid X^{(0)}_i = 1\} \tag{11}$$

That is, if entity i is failed in step 0, that is, $X^{(0)}_i = 1$, then i is one of the $\mathcal{K}$ most vulnerable entities.

The constraints of the ILP problem are used to define: 1) initial failures and 2) failure propagation (state transition). The former is straightforward. The initial state of the system is represented by the constraint:

$$X^{(0)}_1 + X^{(0)}_2 + \cdots + X^{(0)}_n = \mathcal{K} \tag{12}$$

This can be regarded as a liner arithmetic form of I of the SAT encoding shown in the previous section. Note again that entity i is initially failed when $X^{(0)}_i = 1$.

Failure propagation is specified by a collection of linear arithmetic constraints that represent the state of each entity i at the end of each step j. In other words, these constrains are used to have the value of X^j_i conform to the system model. The constrains use as many additional 0-1 variables as the min-terms in the dependency formula for an entity i. Let $C^{(j)}_{i,1}, C^{(j)}_{i,2}, \ldots, C^{(j)}_{i,t_i}$ be these 0-1 variables, where t_i is the number of min-terms.

Note again that the dependency formula for entity i is of the form:

$$e_{i,1,1}e_{i,1,2}\ldots e_{i,1,p_{i,1}} + \cdots + e_{i,t_i,1}e_{i,t_i,2}\ldots e_{i,t_i,p_{i,t_i}} \tag{13}$$

The variable $C^{(j)}_{i,l}$ is used to represent that at least one entity in the l-th min-term has failed. For each $C^{(j)}_{i,l}$, this property can be enforced with the two constraints shown below.

$$C^{(j)}_{i,l} \geq \frac{X^{(j-1)}_{e_{i,l,1}} + X^{(j-1)}_{e_{i,l,2}} + \cdots + X^{(j-1)}_{e_{i,l,p_{i,l}}}}{p_{i,l}} \tag{14}$$
$$C^{(j)}_{i,l} \leq X^{(j-1)}_{e_{i,l,1}} + X^{(j-1)}_{e_{i,l,2}} + \cdots + X^{(j-1)}_{e_{i,l,p_{i,l}}} \tag{15}$$

$C^{(j)}_{i,l}$ can be replaced with $X^{(j-1)}_e$ if the l-th min-term consists of a single entity e. For presentation sake, however, we use $C^{(j)}_{i,l}$, instead of $X^{(j-1)}_e$, in the rest of this section.

Using $C^{(j)}_{i,k}$, the value of $X^{(j)}_i$ is specified by the three constraints as follows.

$$X_i^{(j)} \leq \frac{C_{i,1}^{(j)} + C_{i,2}^{(j)} + \cdots + C_{i,t_i}^{(j)}}{t_i} + X_i^{(j-1)} \tag{16}$$

$$X_i^{(j)} \geq C_{i,1}^{(j)} + C_{i,2}^{(j)} + \cdots + C_{i,t_i}^{(j)} - t_i + 1 \tag{17}$$

$$X_i^{(j)} \geq X_i^{(j-1)} \tag{18}$$

Because of these constraints, $X_i^{(j)} = 1$ if and only if: 1) $X_i^{(j-1)} = 1$ or 2) $C_{i,1}^{(j)}, \dots, C_{i,t_i}^{(j)}$ are all 1. This faithfully represents the dependency formula.

For instance, consider entity $b3$ in the example of the system described in Sect. 3. Its dependency formula is $a2 + a1a3$. The value of $X_{b3}^{(1)}$ is determined by the following constraints.

$$C_{b3,1}^{(1)} \geq \frac{X_{a2}^{(0)}}{1} \tag{19}$$

$$C_{b3,1}^{(1)} \leq X_{a2}^{(0)} \tag{20}$$

$$C_{b3,2}^{(1)} \geq \frac{X_{a1}^{(0)} + X_{a3}^{(0)}}{2} \tag{21}$$

$$C_{b3,2}^{(1)} \leq X_{a1}^{(0)} + X_{a3}^{(0)} \tag{22}$$

$$X_{b3}^{(1)} \leq \frac{C_{b3,1}^{(1)} + C_{b3,2}^{(1)}}{2} + X_{b3}^{(0)} \tag{23}$$

$$X_{b3}^{(1)} \geq C_{b3,1}^{(1)} + C_{b3,2}^{(1)} - 1 \tag{24}$$

$$X_{b3}^{(1)} \geq X_{b3}^{(0)} \tag{25}$$

In this example, the first min-term, namely $a2$, contains only one entity; thus, as stated above, $C_{b3,1}^{(1)}$ can be replaced with $X_{a2}^{(0)}$.

7. Experimental Results

We wrote a Python program that implements the algorithm. In the implementation of Phase 1, Z3 [16] is used as an SAT solver. Although Z3 is actually an SMT (Satisfiability Modulo Theories) solver which supports many background theories, only pure Boolean algebra is used in the program. We utilized the built-in cardinality constraints of Z3, rather than explicitly implementing them. Calling Z3 from the Python program is implemented via the APIs provided by the Z3Py package. All files are available at the project's repository[†].

Phase 2 of the algorithm uses an external solver. The program outputs a file in the LP format. This format is a de-facto standard for description of LP and ILP problems. We used SCIP [17] to solve the ILP program output by our program.

A total of eight problem instances are taken from [7]. The authors of [7] obtained these instances by analyzing the power system in Maricopa county in Arizona. The power system is considered to be comprised of a power network and a communication network. The entities of the power network include load buses, generator buses, neutral buses, and transmission lines, whereas the entities of the communication network include cell towers, fiber-lit buildings, and fiber links. The $\mathcal{K}$ values are also taken from [7]. Table 1 summarizes these problem instances.

For comparison purposes, we ran only Phase 2 of the algorithm with m being replaced with $n - 1$. As n is the total number of entities, clearly the steady state is reached by step $n - 1$ at the latest. This approach is the same as the one proposed by [4], [7], except that some linear constraints are different.

For each problem instance, we ran the proposed two-phase algorithm and the Phase 2-only algorithm with a time-out set to 1800 seconds. All runs were performed on a laptop PC running Windows10 Home 64 bits with a Core i5-6200U CPU and 8 GB memory.

The experimental results are summarized in Table 2. The second leftmost column shows the number of failed entities at the end of failure propagation when $\mathcal{K}$ is set to the value shown in Table 1. The column "m" shows the possible maximum number of the step in which the steady state is reached. The remaining columns show computation time in seconds. The total computation time of the proposed

[†]https://github.com/tatsuhirotsuchiya/k-most-vulnerble-nodes

Table 1 Problem instances.

Problem	n (# entities)	$\mathcal{K}$
24 bus	58	8
30 bus	71	13
39 bus	84	17
57 bus	135	26
89 bus	295	78
118 bus	297	89
145 bus	567	191
300 bus	709	145

Table 2 Experimental results.

Problem	max # failed entities	m	Phase 1 (SAT)	Phase 2 (ILP)	total	ILP only
24 bus	21	3	0.44	0.00	0.2	16
30 bus	36	5	0.55	1.00	1.3	32
39 bus	41	5	0.4	0	0.4	12
57 bus	67	9	1.2	0	1.2	321
89 bus	147	17	23.4	352	375.4	TO
118 bus	148	4	1.5	3	4.5	TO
145 bus	283	11	6.5	47	53.5	TO
300 bus	354	14	48.7	1298	1346.7	TO

algorithm is the sum of the computation times of Phase 1 and Phase 2. The column "ILP only" shows the computation time required when ILP is only used without running Phase 1. As stated above, this is basically the same approach as the existing one [4], [7]. TO stands for timeout.

The results clearly show that the proposed approach substantially outperforms the ILP-only approach. The ILP-only approach timed out for four of the eight problem instances. By contrast, the proposed two-phase approach was able to solve all the problem instances within the timeout period.

The computation time for the proposed approach is comprised of two parts: Phase 1 using the SAT solver and Phase 2 using the IPL solver. The relative ratios between these two parts varied for different instances. But the total time was always smaller than the time used by the IPL-only approach.

For small problems, Phase 1 consumed a longer time than did Phase 2; however the total computation time for these small problems was very small in the first place. By contrast, larger problems, namely, 89 bus, 118 bus, 145 bus, and 300 bus, the computation time of Phase 2 was dominant in the total computation time. Note that a single run of Phase 1 involves solving a total of $m + 1$ SAT instances with different j ($1 \leq j \leq m+1$). Therefore, the computation time for each of the SAT instances was even much smaller than the ILP instance. The ILP-only approach failed to solve these four problems. Although Phase 1 consumed a small part of the total computation time, it made a significant contribution to the reduction of computation time.

8. Conclusion

In the paper, we proposed a two-phase algorithm that computes the most vulnerable entities in interdependent systems where entities rely on each other to function. In such systems, a small number of entities can lead to a system-wide failure as a result of failure propagation. More specifically, the problem is to find the set of $\mathcal{K}$ entities whose initial failures maximize the number of induced failed entities. The proposed algorithm solves the problem by making use of a Boolean satisfiability (SAT) solver and an integer linear programming (ILP) solver. In the first phase, an SAT solver is used to compute the maximum number of failure propagation steps. In the second phase, an ILP solver is used to find the set of the most vulnerable entities. Empirical results show that the maximum step number obtained by a SAT solver was critical in reducing computation time and that as a result, the proposed algorithm is able to solve large problem instances.

As future work, we plan to adapt the proposed algorithm to other related problems. These problems include, for example, the hardening problem [7], which is the problem of determining the entities that should be hardened to minimize the damage of the worst case scenario.

References

[1] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, and R. Setola, "Modelling interdependent infrastructures using interacting dynamical models," International Journal of Critical Infrastructures, vol.4, no.1–2, 2008.

[2] S.V. Buldyrev, R. Parshani, H. Gerald Paul, E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," Nature, vol.464, no.7291, pp.1025–1028, April 2010.

[3] O. Yagan, D. Qian, J. Zhang, and D. Cochran, "Optimal allocation of interconnecting links in cyber-physical systems: Interdependence, cascading failures, and robustness," IEEE Trans. Parallel Distrib. Syst., vol.23, no.9, pp.1708–1720, Sept. 2012.

[4] A. Sen, A. Mazumder, J. Banerjee, A. Das, and R. Compton, "Identification of K most vulnerable nodes in multi-layered network using a new model of interdependency," Proc. IEEE Conference on Computer Communications Workshops, pp.831–836, April 2014.

[5] A. Das, J. Banerjee, and A. Sen, "Root cause analysis of failures in interdependent power-communication networks," 2014 IEEE Military Communications Conference, pp.910–915, 2014.

[6] A. Das, C. Zhou, J. Banerjee, A. Sen, and L. Greenwald, "On the smallest pseudo target set identification problem for targeted attack on interdependent power-communication networks," MILCOM 2015 - 2015 IEEE Military Communications Conference, pp.1015–1020, 2015.

[7] J. Banerjee, K. Basu, and A. Sen, "On hardening problems in critical infrastructure systems," International Journal of Critical Infrastructure Protection, vol.23, pp.49–67, 2018.

[8] A. Sturaro, S. Silvestri, M. Conti, and S.K. Das, "A realistic model for failure propagation in interdependent cyber-physical systems," IEEE Trans. Netw. Sci. Eng., vol.7, no.2, pp.817–831, 2020.

[9] Z. Huang, C. Wang, M. Stojmenovic, and A. Nayak, "Characterization of cascading failures in interdependent cyber-physical systems," IEEE Trans. Comput., vol.64, no.8, pp.2158–2168, Aug. 2015.

[10] D.T. Nguyen, Y. Shen, and M.T. Thai, "Detecting critical nodes in interdependent power networks for vulnerability assessment," IEEE Trans. Smart Grid, vol.4, no.1, pp.151–159, March 2013.

[11] Y. Shen and M.T. Thai, "Network vulnerability assessment under cascading failures," 2013 IEEE Global Communications Conference (GLOBECOM), pp.1526–1531, 2013.

[12] A. Veremyev, K. Pavlikov, E. Pasiliao, M. Thai, and V. Boginski, "Critical nodes in interdependent networks with deterministic and probabilistic cascading failures," J. Glob. Optim., vol.74, pp.803–838, Sept. 2019.

[13] K. Hanada, T. Tsuchiya, and Y. Fujisaki, "Satisfiability-based analysis of cascading failures in systems of interdependent networks," 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), pp.105–1058, 2019.

[14] K. Hida and T. Tsuchiya, "Finding critical nodes in interdependent networks with SAT and ILP solvers," CoRR, vol.abs/2211.05659, 2022.

[15] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications, IOS Press, 2009.

[16] L.M. de Moura and N.S. Bjørner, "Z3: An efficient SMT solver," Proc. 14th International Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), C.R. Ramakrishnan and J. Rehof, eds., Lecture Notes in Computer Science, vol.4963, pp.337–340, Springer, 2008.

[17] G. Gamrath, D. Anderson, K. Bestuzheva, W.K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S.J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M.E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig, "The SCIP Optimization Suite 7.0," ZIB-Report 20-10, Zuse Institute Berlin, March 2020.

Daichi Minamide received his bachelor's degree in engineering and master's degree in information science from Osaka University in 2020 and 2022, respectively. He is currntly with Panasonic Automotive Systems Co., Ltd. The current work was conducted when he was a master's student.

Tatsuhiro Tsuchiya is currently a professor in the Department of Information Systems Engineering at Osaka University. He received the M.E. and Ph.D. degrees from Osaka University in 1995 and 1998, respectively. His research interests are in the areas of model checking, software testing, and dependable systems.