Enhancing the Robustness of Interdependent Cyber-Physical Systems by Designing the Interdependency Relationship

Yangming Zhao and Chunming Qiao
Department of Computer Science and Engineering
State University of New York at Buffalo

***Abstract*—This paper studies how to optimize the interdependencies among the components in the interdependent Cyber-Physical Systems (CPS), in order to enhance the system robustness. In the interdependent CPS, some components may require the resources (or work conditions) provided by other components. Accordingly, small scale initial failure may incur large scale cascading failure as some of the working components may loss the resources (or work conditions) provided by the failed components. Due to this fact, we can optimize the resource allocation and change the interdependencies among components, so as to minimize the impact of cascading failure incurred by single component failure. To this end, we formulate the problem as an Integer Linear Programming (ILP) problem, and design an efficient algorithm based on progressive relaxation and rounding method to solve it. We also propose a greedy algorithm to quickly solve the problems in extreme large scale systems. In addition, we study how to allocate the redundant resources for the backup purpose and further enhance the system robustness. Simulation results show that if 1.3 times of the resources required by all the components are provided, we can eliminate the cascading failure incurred by single component failure.**

I. Introduction

Nowadays, the infrastructure systems are interdependent on each other. To properly work, components in one system may require the resources provided by other systems. For example, the electric power systems need the fuel provided by oil/gas systems, the communication systems need the power provided by the electric power systems, and banking/finance systems need the communication services provided by communication systems. We call such system of systems as *interdependent Cyber-Physical System* (CPS). In such interdependent CPS, the failure in the oil/gas systems may result in the failure in the electric power systems, and then in the communication systems and banking/finance systems. This phenomenon is referred as *cascading failure*. A small scale failure may lead to large scale cascading failure. In the worst case, it can result in the failure of entire system [1].

There have been plenty of such cascading failures. For example, the 2003 Northeast blackout, which is caused by a software bug, affected estimated 55 million people and incurred around $6 billion losses [2, 3]. Also, the 2012 India blackouts due to a high loading caused power outage to over 600 million people for 24 hours [4]. The estimated economic loss was round 12 billion Rupees. Accordingly, enhancing the robustness of interdependent CPS is an important issue.

A lot of works have been proposed to enhance the robustness of interdependent CPS. Some of them focused on how the cascading failure propagates. [5] and [6] were the first works on this issue. They studied when the cascading failure can be restricted to a limited area in a random network with percolation theory. [7] investigated the failure scale that may result in the entire system failure under given interdependency relationship. [8] and [9] were the leading works on how to recover the system from a large scale cascading failure. In addition, [1] proposed to find out the most vulnerable components in the interdependent system, and hence we can pay more attention on protecting them. However, all of these works assumed that the interdependencies among components are fixed. Actually, the interdependencies among components in the system are also an ingredient that can be controlled. A good interdependency relationship can limit the propagation of cascading failure and enhance the system robustness (see detail in Section III). In this paper, we will study how to design the interdependencies among components to enhance the robustness of interdependent CPS.

The interdependencies among components in the CPS are due to the fact that some components/systems require the resources (or working conditions) provided by other components/systems. For example, the electric power systems require the fuel provided by oil/gas systems. Hereby, the failure of oil/gas systems may incur failures in the electric power systems. However, the same resources can be provided by different systems, e.g. there are multiple oil/gas systems that can provide fuel to the same electric power system. Different resource allocation schemes can result in different interdependencies and hence impact the robustness of interdependent CPS. Accordingly, we mainly focus on the resource allocation problem in this paper.

To optimize the resource allocation (i.e. the interdependency) among system components, we first formulate the problem as an Integer Linear Programming (ILP) problem. Since the ILP is intractable in large scale systems, an algorithm based on progressive relaxation and rounding is proposed to solve it. As solving a LP model in large scale system is

This work was supported by NSF grant EFMA No. 1441284.

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

also time consuming, a greedy algorithm is also proposed for the extreme large scale systems. Besides, we also discuss how to assign the redundant resources for the backup purpose which can further enhance the system robustness. To the best of our knowledge, this is the first work that optimizes the interdependency to enhance the robustness of interdependent CPS.

The main contributions of this paper include:

- An ILP formulation for the interdependency design problem
- Problem analysis and algorithms to design interdependency
- Redundant resource allocation for enhancing the system robustness

II. Related Work

[5] and [6] were the first works that revealed the interdependencies among the components in multiple systems and studied how the failure propagates in the interdependent CPS. However, both of them did not focus on how to protect the system from cascading failure and enhance the system robustness.

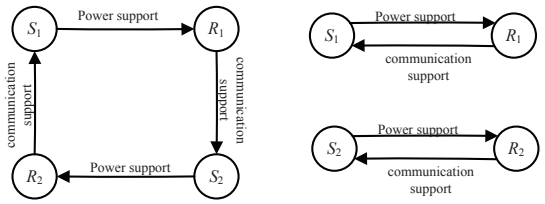
[10] and [1] are the representatives to study how to find the most critical components in the interdependent CPS, and then we can pay more attentions on protecting these components. [11] proposed a metric to evaluate the robustness of interdependent systems and design algorithms to derive such metric. These works are partially related to the interdependent CPS robustness, but they did not explicitly tell us what can be adopted to enhance the system robustness.

[8] and [9] focused on recovering the interdependent CPS from cascading failure. Nevertheless, [8] made an impractical assumption as in [6]: there are two subnetworks such that a component in one subnetworks only depends on the components in the other subnetwork. More importantly, it assumed that each component can be repaired in one stage and did not consider the repair resources at all. [9] removed all these assumptions and derived a more practical solution to recover the interdependent system from a large scale cascading failure. However, both of them assumed the interdependencies are fixed. They are enhancing the system robustness in different dimensions of our work.

III. Motivation Example

In this section, we take a simple cyber-physical system, which consists of a power grid and a Control and Communication Network (CCN), as an example to motivate our work. In the power grid, there are two substations, S_1 and S_2, that provide power support to the routers in the CCN. There are two routers, R_1 and R_2, in the CCN providing communication support to the substations in power grid for sending data and receiving control signals.

Fig. 1 shows two different ways to form the interdependencies among these substations and routers. The first way is shown in Fig. 1(a). Substation S_1 provides power support to router R_1, which provides the communication support to the substation S_2. Substation S_2 provides power support to router R_2, and R_2 is used to send data and receive signals by S_1.



(a) Worse interdependency (b) Better interdependency

Fig. 1. Motivation example.

Following this interdependency, assume substation S_1 fails, then the router R_1 losses power, and hence it cannot provide communication support to substation S_2. Consequently, substation S_2 cannot properly provide power support to the router R_2, and R_2 also fails. In the end, all the components in the CPS fail. Actually, the failure of any component in the CPS will result in the failure of entire system if the interdependency is formed following the way shown in Fig. 1(a).

To alleviate the impact of such cascading failure, we can carefully design the interdependencies among the components in the system. Take the interdependencies shown in Fig. 1(b) as an example, let substation S_1 (S_2) and router R_1 (R_2) provide power support and communication support to each other, the failure of any component will impact at most one another component. In this way, we can limit failure propagation in the system and mitigate the impact of cascading failure.

Motivated by above example, in this paper, we are to study how to design the resource allocation scheme among the components, i.e. design the interdependency, in the CPS in order to minimize the impact of cascading failure, and enhance the system robustness.

IV. System Model and Formulation

A. System Model

We assume there are N components and K types of resources in the CPS. To work properly, component i needs a subset of these K types of resources provided by other components. More specifically, component i needs r_{ik} units of resource k in order to properly work. If component i is working, it can provide a_{ik} units of resource k to other components. In our work, we assume that each component may need multiple types of resources, however, even one type of resources is not enough, the corresponding component cannot work and hence provides nothing to other components.

To enhance the robustness of CPS, we can optimize the amount of each type of resources that every component provides to other ones, such that the impact of cascading failure incurred by arbitrary one (and only one) component failure in the system can be minimized. It means that our work is to provide the worst case performance guarantee under single component failure in the CPS.

We define a concept named *Shared Failure Group* (SFG) to refer a group of components that may fail due to the cascading failure incurred by the same component. The component whose failure will result in the failure of the entire SFG is called the *root* of this SFG. Take the case shown in Fig. 1

as an example, in Fig. 1(a), the failure of R_1 will result in the failure of all the components in the CPS. Accordingly, all the components are in the same SFG with the root R_1. In fact, there are another three SFGs rooted by R_2, S_1 and S_2, respectively. All of these SFGs contains all the components in the CPS. For the case shown in Fig. 1(b), $\{S_1, R_1\}$ forms an SFG rooted at S_1 or R_1, while $\{S_2, R_2\}$ forms another SFG rooted at S_2 or R_2. For simplicity, we say SFG i as the SFG with component i as its root. It is worth noting that one component can be included in multiple SFGs containing different components. For example, component A and B, which are not in the same SFG, provide different types of resources to component X. In this case, component X should be in both SFG A and SFG B. With the concept of SFG, our goal is to minimize the size of the largest SFG.

B. Problem Formulation

As discussed in last subsection, we are to minimize the size of the largest SFG. Suppose the largest size is C, then the objective of our problem is

$$\text{minimize} \qquad C \tag{1}$$

If we use a binary variable x_{ig} to denote if component i belongs to the SFG g, then we have

$$\sum_i x_{ig} \leq C \tag{2}$$

for all SFG g. To ensure each component i working properly, we should provide it enough resources. With y_{ijk} denoting the amount of resource k that is provided by component i to component j, this condition can be formulated as

$$\sum_i y_{ijk} = r_{jk} \tag{3}$$

for all component j and resource k. With this constraint, we provide each component the exactly amount of resources it requires. This is because that providing least resource may involve the least component into each SFG, and hence achieve the best system robustness. However, if we provide each component more resources for the backup purpose, we can further improve the system robustness. We will discuss how to further improve the system robustness by leveraging the redundant resource for backup purpose in Section VI.

The total amount of every type of resources that each component can provide to other components is limited, i.e,

$$\sum_j y_{ijk} \leq a_{ik} \tag{4}$$

for all component i and resource k. Furthermore, if component i provides resources to component j, component j must be in all the SFGs that component i is there. This is because that any component whose failure will result in the failure of component i will further incur the failure of component j as it will lose the resources provided by component i. This relationship can be formulated as

$$x_{jg} - x_{ig} \geq \frac{y_{ijk} - a_{ik}}{a_{ik}} \tag{5}$$

for all component i, j, resource k and SFG g.

Only with constraints (2) – (5), there is a trivial solution $x_{ig} = 0$ for all the component i and SFG g. To avoid this case, we initialize every component to be in the SFG rooted by itself, i.e.

$$x_{ii} = 1 \tag{6}$$

for all component i.

Based on above discussions, the problem we are to solve can be formulated as following ILP model, named CPS Robustness Enhancement Problem (CREP):

maximize (1)
subject to: (2) – (6)
$x_{ig} \in \{0, 1\}$

V. Analysis and Algorithm Design

As the ILP model formulated in last section is intractable in large size systems, we need efficient algorithms to solve the problem. Before we design the algorithms, we first present some analyses in Section V-A. Then, an algorithm based on the progressive relaxation and rounding of ILP model and a pure heuristic algorithm are proposed in Section V-B and Section V-C, respectively.

A. Problem Analysis

To derive some hints on designing the algorithm, We first consider a special case that there is only one type of resources in the system. Some of the components are providing such resource (named *provider*) and the others are requiring such resource to work (named *consumer*). Without loss of generality, we assume the resource amount provided by providers and required by consumers are both integer, and the total amount of required resources equals to the total amount of provided resources. In this case, we have following proposition.

Proposition 1. *If there is a resource assignment scheme such that every consumer gets resource from only one provider, there should not be a consumer getting resources from multiple providers in the optimal solution.*

This proposition is obvious by noting the fact that all the consumers getting resource from the same provider should be in the same SFG with the provider as the root. Based on the assumption in the proposition, we know that there is a solution such that every consumer is only in one SFG. If one consumer gets resource from multiple providers, this consumer should be in multiple SFGs, and hence it degrades the system robustness.

From Proposition 1, we know that if we can find the optimal solution of every specific instance of CREP within polynomial time complexity, we can also solve the subset sum problem [12]. For a given set of numbers $S = \{s_1, s_2, \ldots, s_n\}$, to check if there is a non-empty subset whose sum is P, we can assume there are n consumers in the system and each consumer requires $s_i \in S$ unit of resources. In addition, we suppose there are two providers, one of them can provide P unit of resources, while the other provides $\sum s_i - P$ unit of resources.

Algorithm 1: ILP Based Algorithm

Input: Resource requirement of each component $\{r_{ik}\}$, resource provided by each component $\{a_{ik}\}$
Output: Resource providing scheme $\{Y_{ijk}\}$

1: Initialize the resource already gotten by each component $z_{ik} \leftarrow 0$, and the resource already provided out by each components $p_{ik} \leftarrow 0$
2: Formulate CREP model based on the input
3: **while** Not all the components get enough resource **do**
4: Solve relaxed CREP, and assume the solution is $\{y^*_{ijk}\}$
5: Find $(i, j, k) = \arg\max \frac{y^*_{ijk}}{r_{jk} - z_{ik}}$
6: **if** $r_{jk} - z_{ik} \geq a_{ik} - p_{ik}$ **then**
7: $Y_{ijk} \leftarrow a_{ik} - p_{ik}$, $p_{ik} \leftarrow a_{ik}$, $z_{ik} \leftarrow z_{ik} + a_{ik}$
8: **else**
9: $Y_{ijk} \leftarrow r_{jk} - z_{ik}$, $p_{ik} \leftarrow p_{ik} + r_{jk} - z_{ik}$, $z_{ik} \leftarrow 2z_{ik} - r_{jk}$
10: **end if**
11: Add constraint $x_{jg} \geq x_{ig}$ and $y_{ijk} = Y_{ijk}$ in to the relaxed CREP model
12: **end while**
13: **return** $\{Y_{ijk}\}$

If this problem has a solution without any consumer asks resources from both providers, the answer is "yes", i.e. the subset sum problem has a feasible solution. Otherwise, there is no feasible solution. Since subset sum problem is a well-known NP-complete problem, we have following theorem:

Theorem 1. *CREP is NP-complete.*

Due to the hardness of CREP, we would like to design a heuristic to solve it efficiently. Since the failure of one component will result in the failure of all the components it is supporting, we should reduce the number of components supported by each provider. Accordingly, we propose:

Guideline 1. *When component i provides resource k to component j, it should either give all the resource k it can provide to component j, or fulfill the component i's requirement of resource k.*

B. ILP Based Algorithm

The ingredient that makes CREP difficult to solve is the binary variable x_{ig}. Accordingly, we can try to leverage the relaxation and rounding method to design the algorithm. The main idea of our algorithm can be summarized as: 1) relax the binary constraint of x_{ig} in CREP model; 2) fix part of the resource assignment based on the solution of relaxed-CREP model; 3) repeat 1) and 2) till all the resource assignments are fixed. When we fix the resource assignment, we should follow the hints from Guideline 1 to enhance the CPS robustness.

Following the above idea, we can design Algorithm 1 to solve the CREP. In this algorithm, we first initialize the amount of each type of resources obtained and provided out by every component to be 0, and formulate the CREP model based on the input. After that, we iteratively assign resources to every component from Line 3 to 12.

In each iteration, the relaxed CREP model is solved to get some hints for resource assignment (Line 4). According to the solution of relaxed CREP, we find out the resource assignment that can provide the largest fraction of the remaining required amount of resources (Line 5), and concrete it, as such assignment scheme has a higher probability to appear in the optimal solution. However, before the resource assignment, the amount of assigned resource should be determined based on the Guideline 1. If component i cannot fulfill the remaining requirement of component j on resource k, it should provide all its remaining resource k to component j (Lines 6–8). Otherwise, it fulfills the component j's requirement on resource k (Lines 8–10). After we determine the resource assignment in each iteration, we should update the total obtained and provided resources for each component. At the end of each iteration, more constraints should be added into the CREP model to reflect the resource assignment (Line 11).

C. Greedy Heuristic

Solving LP model is also time consuming when there are thousands of components in the CPS. Accordingly, a faster algorithm is needed in extreme large scale systems. To this end, we also propose a greedy heuristic to solve the CREP.

The main idea of our greedy heuristic is that we sequentially fulfill the resource requirement of each component. When finding a component to provide the required resource, the component that minimizes the SFG size increase has the highest priority to be selected. The amount of resources that the selected component should provide is determined based on the Guideline 1. Following this greedy mind, we propose Algorithm 2 to solve CREP.

In Algorithm 2, we sequentially find resource provider for every component and every type of resource it requires. To find a resource provider in Line 5, we should traverse all the possible providers and select the one that brings the minimum SFG size increase. It is worth noting that the minimum SFG size increase is defined in the lexicographic order. In other words, if two components would not enlarge the largest SFG, we would compare their impact on the size of the second largest SFG, and then the third largest SFG, and so on. From Line 6 to Line 10, we determine the amount of resource that the selected provider should supply following the Guideline 1 as in the Lines 6–10 in Algorithm 1. When all the components get enough resources it requires, the algorithm ends.

VI. Redundant Resource Assignment

When there are redundant resources, we can provide them to the consumers in case some of the providers fail, and hence further enhance the CPS robustness. To this end, we greedy assign the redundant resources to the components in the largest SFG. Every component in SFG i is directly or indirectly supported by components i. All the components in SFG i will fail due to the failure of component i. Accordingly, if SFG i is the largest SFG, we can enhance the system robustness

Algorithm 2: Greedy Heuristic

Input: Resource requirement of each component $\{r_{ik}\}$, resource provided by each component $\{a_{ik}\}$
Output: Resource providing scheme $\{Y_{ijk}\}$

1: Initialize the resource already gotten by each component $z_{ik} \leftarrow 0$, the resource provided by each component $p_{ik} \leftarrow 0$, and the SFG $S_i \leftarrow \{i\}$
2: **for** All component j in the system **do**
3: **for** All resource type k required by component i **do**
4: **while** $z_{ik} < r_{ik}$ **do**
5: Find the component i, such that it can bring the minimum SFG size increase if it provides resource k to component j
6: **if** $r_{jk} - z_{ik} \geq a_{ik} - p_{ik}$ **then**
7: $Y_{ijk} \leftarrow a_{ik} - p_{ik}$, $p_{ik} \leftarrow a_{ik}$, $z_{ik} \leftarrow z_{ik} + a_{ik}$
8: **else**
9: $Y_{ijk} \leftarrow r_{jk} - z_{ik}$, $p_{ik} \leftarrow p_{ik} + r_{jk} - z_{ik}$, $z_{ik} \leftarrow 2z_{ik} - r_{jk}$
10: **end if**
11: **end while**
12: **end for**
13: **end for**
14: **return** $\{Y_{ijk}\}$

Algorithm 3: Redundant Resources Assignment

Input: Resource providing scheme $\{Y_{ijk}\}$, remaining resources $\{b_{ik}\}$
Output: Remaining resource assignment $\{B_{ijk}\}$

1: Get the SFG information based on $\{Y_{ijk}\}$, and sort all SFGs in the decreasing order of their size
2: **for** all SFG g **do**
3: **if** $\sum_{i \neq g} b_{ik} \geq \sum_i Y_{gik}$ for all k **then**
4: **for** all i, k such that $Y_{gik} > 0$ **do**
5: Assign $\min\{Y_{gik} - \sum_{u \neq g} B_{uik}, 0\}$ unit of resources to component i in arbitrary way, and update $\{B_{ijk}\}$
6: **end for**
7: **end if**
8: **end for**
9: **return** $\{B_{ijk}\}$

by assigning more resources to all the components directly supported by component i. If all the resources provided by component i is backed up, the SFG i can be removed.

Based on above discussions, we design Algorithm 3 to allocate the redundant resources. In this algorithm, we first derive the SFG information based on the input, and sort all the SFGs in the decreasing order of their size (Line 1). Then, we sequentially backup the resources provided by the root of each SFG (Lines 2–8). To minimize the size of largest SFG, we should first backup the resources provided by the root of the largest SFG. To backup the resources provided by component g, we first check if there are enough resources for the backup purpose. If so, we assign resources to the components which component g provides resources to (Lines 4–6). When we assign the redundant resources for the backup purpose, since we are only focusing on the single failure scenario, how to assign the backup resources, or in other word, which components provide the required resources for backup, does not impact the resource backup effect.

There is an important issue that should be noted when we assign the redundant resources for the backup purpose. Suppose component j gets resource k from both components g_1 and g_2, and when we back up the resources provided by component g_1, component u has already assigned B_{ujk} unit of resource k to component j, in this case, when we back up the resources provided by component g_2, it is not necessary to supply the amount of Y_{g_2jk} to component j, as some of the resources are provided by component u. We only need to assign $\min\{Y_{g_2jk} - \sum_{u \neq g_2} B_{ujk}, 0\}$ unit of resources to component j in case of the failure of component g_2.

VII. Performance Evaluation

In this section, we investigate the performance of our proposed algorithms.

A. Impact of System Size

To investigate how the algorithm performance is impacted by the system size, we change the component number in the system from 10 to 100. In addition, we assume there are 7 types of resources in the system, each component provides 4 of them, but requires 2 of them in order to properly work. The amount of each type of resources required by every component is evenly distributed on [10,20]. We also assume that the amount of each type of resources provided by each component is on average 1.2 times of the amount required by each component. For simplicity, we refer this ratio as *redundant ratio*. The larger redundant ratio means more resources can be utilized by the system to improve its robustness.

The simulation results are shown in Fig. 2. From this figure, we can observe that the size of the largest SFG does not have a strict trend with the system size, though it largely increases with the number of components in the system. The increase of the largest SFG size may be due to the way we generate the simulation input. When generating the input, we ensure the problem is feasible, and hence there must be a way to form a small scale subsystem. In a small scale CPS, the subsystem must be with smaller size. However, the subsystem scale should be larger in the large scale CPS.

The second observation is that the ILP based algorithm always outperforms the greedy algorithm. This is intuitive as the ILP based algorithm optimizes the system in a global view, while the greedy algorithm optimizes it in a myopic manner.

B. Impact of resource number provided by each component

In this section, we assume there are 50 components and 20 types of resources in the system. Every component requires 2 types of these resources for properly working and

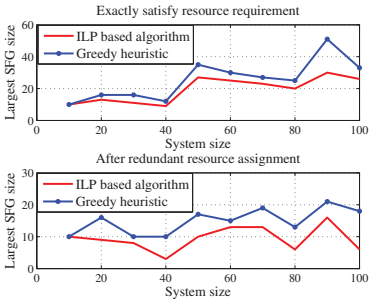


Fig. 2. Performance changes with system size.

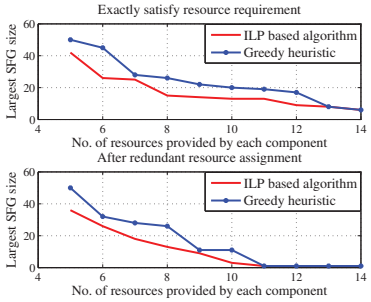


Fig. 3. Performance changes with resource number provided by each component.

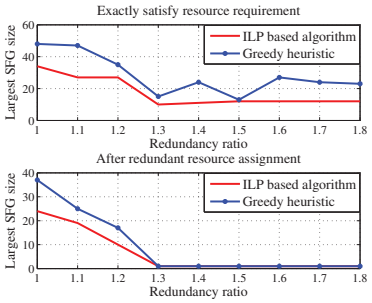


Fig. 4. Performance changes with redundant ratio.

the redundant ratio is also set to be 1.2. Then we change the type number of resources provided by each component and investigate how the algorithm performance changes. The simulation results are shown in Fig. 3.

From this figure, we can see that with the increase of type number of resources provided by each component, the largest SFG size decreases. For a given group of components, each component provides more resources brings more opportunity for them to fulfill their resource requirements by themselves. Accordingly, the SFG size correspondingly reduces.

When each component can provide most types of the resources, the performance of greedy algorithm approaches that of the ILP based algorithm. This is because that when each component can provide most types of the resources, there may be a lot of ways to form small scale subsystems and hence the greedy algorithm can derive a good solution. When the number of resources provided by each component exceed 13 in our simulation, ILP based algorithm and greedy heuristic derive the same performance.

C. Impact of redundant ratio

In practical system, we may also be interested in how many resources provided by each component is reasonable. We answer this question by simulation in this subsection. We we assume there are 50 components and 20 types of resources in the system. Every component requires 2 types of these resources for properly working but provides 10 of them. Then we test the algorithm performance under different redundant ratio. Fig. 4 shows the simulation results.

When the redundant ratio increases, the largest SFG size decreases as it provides more SFG formation options. However, when the redundant ratio exceeds a certain threshold, the largest SFG size will not significantly change any more. Especially for the ILP based algorithm, the largest SFG size fixes when the redundant ratio exceeds 1.3.

The same phenomenon appears after the redundant resource assignment. Larger redundant ratio results in smaller largest SFG size after redundant resource assignment. In our simulation, when the redundant ratio exceeds 1.3, the largest SFG size reduces to be 1, i.e. single component failure may not incur any cascading failure.

VIII. Conclusion

In this paper, we studied how to optimize the resource allocation among components in an interdependent cyber-physical system, in order to minimize the impact of cascading failure incurred by single component failure. To this end, we formulated the problem as an Integer Linear Programming (ILP) model, and then proposed an efficient algorithm based on progressive relaxation and rounding of the ILP model to solve it. We also proposed a greedy algorithm for the extreme large scale systems where the ILP based algorithm is time consuming. Furthermore, we discussed how to allocate the redundant resources for the backup purpose, and further enhance the system robustness. Simulation results show that ILP based algorithm outperforms greedy algorithm, and redundant resource can be used to greatly enhance the system robustness. More importantly, providing 1.3 times of resources required by all the components can eliminate the cascading failure incurred by single component failure.

References

[1] 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," in *INFOCOM WKSHPS*, 2014.

[2] "Power blackout risks," *CRO Forum*, 2011. [Online]. Available: https://www.allianz.com/v_1339677769000/media/responsibility/documents/position_paper_power_blackout_risks.pdf

[3] J. Glotfelty, "Transforming the grid to revolutionize electric power in north america," Tech. Rep., 2003.

[4] L. L. Lai, H. T. Zhang, C. S. Lai, F. Y. Xu, and S. Mishra, "Investigation on july 2012 indian blackout," in *2013 International Conference on Machine Learning and Cybernetics*.

[5] S. Buldyrev, R. Parshani, G. Paul, and S. Stanley, H.and Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025–1028, 2010.

[6] J. Gao, S. Buldyrev, and S. Stanley, H.and Havlin, "Networks formed from interdependent networks," *Nat. Phy.*, vol. 8, no. 1, pp. 40–48, 2011.

[7] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *IEEE GLOBECOM*, Dec 2013, pp. 2164–2169.

[8] A. Mazumder, C. Zhou, A. Das, and A. Sen, "Progressive recovery from failure in multi-layered interdependent network using a new model of interdependency," in *CRITIS*, 2014.

[9] Y. Zhao, M. Pithapur, and C. Qiao, "On progressive recovery in interdependent cyber physical systems," in *IEEE GLOBECOM*, Dec 2016.

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

[11] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *IEEE Globecom*, Dec 2013, pp. 2164–2169.

[12] "Subset sum problem." [Online]. Available: https://en.wikipedia.org/wiki/Subset_sum_problem