On Progressive Recovery in Interdependent Cyber Physical Systems

[1]Yangming Zhao, [2]Mohammed Pithapur and [1]Chunming Qiao

[1]Department of Computer Science and Engineering, [2]Department of Electrical Engineering
State University of New York at Buffalo

***Abstract*—This paper studies how to determine an optimal order of recovering interdependent Cyber Physical Systems (CPS) after a large scale failure. In such a CPS, some failed devices must be repaired first before others can. In addition, such failed devices require a certain amount of repair resources and may take multiple stages to repair. We consider two scenarios: 1) reserved model where all the required repair resources should be prepared at the beginning of repairing a device; and 2) opportunistic model where we can partially repair a device with only part of the required resources. For each scenario, we model it using an Integer Linear Programming (ILP) and use a relaxation and rounding method to design an ILP based algorithm. In addition, we also design a Dynamic Programming (DP) based algorithm. Simulation results show that ILP based algorithm outperforms DP based algorithm by 10%–20% in systems with less than 200 failed devices, but DP based algorithm can support extreme large size systems with more than 5000 failed devices.**

I. Introduction

Background: After a large scale failure of a Cyber Physical Systems (CPS), it usually takes multiple stages to repair all the failed components and restore the system functions and utilities. This multi-stage recovery, or progressive recovery, is necessary due to the fact that there is a limited amount of repair resources, such as man-power and materials, available at a time. Given an initial failure, and the amount of limited resources available at each stage, a progressive recovery strategy determines what to repair during each stage. An optimal progressive recovery strategy aims to maximize certain system utilities during the recovery.

Optimal progressive recovery is already a challenging problem in communication networks when considering communication link failures only [1]. This problem is even more challenging in interdependent CPS such as a modern power grid consisting of devices for sensing (e.g., PMUs), communications (e.g., Internet), and control (e.g., SCADA systems) as well as transmissions lines, circuit breakers, substations and power generators. In such an interdependent CPS, the tripping of a critical line may lead to tripping of more lines because of overloading or the SCADA network is unable to shed load automatically, then a cascaded failure occurs and even results in the collapse of the entire CPS.

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

There have been several such large scale failures due to cascaded failures. For example, the 2003 Northeast blackout, which is caused by a software bug, affected an 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 ($180 million). Accordingly, how to effectively recover from a large scale failure in such a CPS is an important issue.

Problem Statement: For simplicity, we refer the components, e.g. nodes, links, as devices in this paper. For arbitrary device, a set of devices that provides it the necessary condition to work is called a *support* of this device. For example, with the IIM [5], in boolean expression $C_i \leftarrow C_1 * C_2 + C_3$, $\{C_1, C_2\}$ is a support of C_i and $\{C_3\}$ is another support of C_i. If a device is alive at the beginning or fully repaired, we say this device *works*. A support works if and only if all the devices in this support work.

Given 1) the initial list of failed devices and their dependencies according to IIM; 2) the total amount of repair resources needed for each failed devices; 3) the amount of repair resources available at the beginning of each stage; and 4) a constant utilities (representing the the number of customers that can be served for example) associated with each device, we will consider two models referred as *reserved (repair resources) model* and *opportunistic model*, respectively.

The reserved model is similar to a "prepaid" model, in which all the required resources have to be committed upfront before a device can start its repair process, even though the repair process may still take multiple stages, and in addition, during each stage, only some of the total resources committed will actually be used. This is applicable to some repair process which cannot be paused/stopped such as when a generator is being started (starting a generator may require a couple of staffs to work continuously for a few hours). In this model, we are mainly interested in determining which devices to repair at each stage.

The opportunistic model is similar to a "pay-as-you-go" model. where a device can start its repair process as soon as some repair resources are available (and allocated). The device will not be completely repaired unless all the required resources are allocated. In addition, there is also no guarantee that the repair process will be a continuous one. In other words, if there are no more resources available in the next

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

TABLE I
NOTATION LIST

Notation	Description
u_i	Constant. The utility of device i.
x_i^k	Binary variable. It indicates if the device i starts repairing at stage k.
w_i^k	Binary variable. It indicates if the device i completes repairing at stage k.
$G_i^{m,k}$	Binary variable. It indicates if the m^{th} support of device i works at stage k.
$\boldsymbol{S}_{im}$	The devices set of the m^{th} support of device i.
$\boldsymbol{N}$	The set of all devices in the system.
$\boldsymbol{A}$	The set of all the alive devices at the beginning of the recovery procedure.
$\boldsymbol{D}$	The set of all the dead/failed devices at the beginning of the recovery procedure.
K	The number of stages used for the recovery procedure.
M_i	The number of supports for device i.
r_i	The amount of resources available to repair devices at stage i.
a_i^k	The amount of resources allocated to device i at stage k.
e_i	The maximum amount of resources that can be consumed to repair device i in each stage.
t_i	The number of stages that is required to repair device i.
m_i	The amount of resource that are required to repair device i. $m_i = 0$ for all the devices that are alive at the beginning of the recover procedure.

stage, the repair process has to be paused. Note that even if all the required resources are available, the device may still take multiple stages to complete as the amount of resources used in each stage is limited. In this opportunistic model, our major focus is to determine how much resources to allocate to each device at every stage. In both models, we will maximize the accumulated system utility over K repair stages, where K is assumed to be the maximum number of stages needed to repair all the failed devices.

Proposed Approach: We first formulate the recovery problem as an Integer Linear Programming (ILP) model. However, we cannot solve them directly as it is intractable even in small size systems (about 10 devices and 10 dependence relationships). To solve this problem, for each model, we design an algorithm based on the relaxation and rounding of the ILP formulation, which can support up to 200 failed devices, and in addition, another algorithm based on Dynamic Programming (DP) which can easily support more than 5000 failed devices in the systems. Simulation results show that the ILP based algorithm may outperform the DP based algorithm by about 10% – 20%, with the cost of more computation time.

For clear presentation, we summarize the notations used in this paper in Tab. I.

II. RELATED WORK

Significant efforts have been made on revealing the interdependcies between system devices in the past few years. [6] and [7] studied the failure propogation in the interdependent networks/systems. However, [6] assumed every device only depended on exactly one other devices in the system. Though this limitation was mitigated in [7], it supposed that there were two subnetworks such that a device in one subnetworks only depended on the devices in the other network.

[8] and [5] are the representatives to study how to find the most critical devices in the interdependent systems, and then we can pay more attentions on protecting these devices. However, they did not study how to recover the systems once the critical devices fail.

[1] is the first work studying the network progressive recovery after large scale failure, but it did not consider the device interdependence. As far as we know, [9] is the only existing work that focused on recovering the interdependent CPS from cascaded failure. Nevertheless, [9] made an impractical assumption as in [7], there are two subnetworks such that a device in one subnetworks only depends on the devices in the other subnetwork. More importantly, it assumed that each device can be repaired in one stage and did not consider the repair resources at all. In this paper, we will study how to determine the order to repair devices in the interdependent CPS under more reasonable assumption, i.e. a device may take multiple stages to recover, and any device may depend on arbitrary combination of other devices in the systems.

III. RESERVED (REPAIR RESOURCE) MODEL

In this section, we discuss the reserved model where the repair resources must be spent before we start repairing a device, although each device needs multiple stages to repair.

A. ILP Formulation

The objective of our work is to maximize the accumulated system utility over all repaired devices over the entire repair procedure (e.g. K stages), i.e.

$$\text{maximize} \qquad \sum_{i\in\boldsymbol{D}}\sum_{k} u_i w_i^k \tag{1}$$

There are three series of constraints for this problem. The first one is the device dependency, i.e., a device can be repaired only if at least one of its supports works. This dependency can be modeled as

$$x_i^k \le \sum_m G_i^{m,k-1} \tag{2}$$

for all $i \in \boldsymbol{N}$ and $k = 1, \ldots, K$. In addition, a support works when all the devices in this support work, i.e. the variable $G_i^{m,k}$ should satisfy

$$\frac{\sum_{j\in\boldsymbol{S}_{im}} w_j^k - |\boldsymbol{S}_{im}| + 1}{|\boldsymbol{S}_{im}|} \le G_i^{m,k} \le \frac{\sum_{j\in\boldsymbol{S}_{im}} w_j^k}{|\boldsymbol{S}_{im}|} \tag{3}$$

for all m and $i \in \boldsymbol{N}$ and $k = 0, \ldots, K$

The second series of constraints are for the time to repair a device. A device will work after t_i stages of repair. This is

$$w_i^k \le x_i^{k-t_i} \tag{4}$$

for all $i \in \boldsymbol{D}$. However, for the devices that are alive at the beginning, they always work, i.e. we have

$$x_i^0 = 1, w_i^0 = 1 \tag{5}$$

for all the device $i \in \boldsymbol{A}$. For a repaired device, it is also repaired in the later stages, and so are the working devices. Therefore, there is

$$x_i^{k-1} \le x_i^k, w_i^{k-1} \le w_i^k \tag{6}$$

for any device $i \in \boldsymbol{N}$.

The last constraint is to ensure that there should be enough resources for the devices that are to be repaired in each stage. This can be formulated as

$$\sum_{i} m_i x_i^k \leq \sum_{j \leq k} r_j \tag{7}$$

for $k = 1, \ldots, K$

In summary, the formulation of Reserved Model (RM) is

maximize (1)
subject to: (2) — (7)

B. ILP Based Algorithm

Though we have formulated the problem as an ILP model, it cannot be directly applied to a practical problem due to the time complexity. The main reason that results in the intractability of the ILP model in Section III-A is the binary variables x_i^k and $G_i^{m,k}$. Accordingly, we propose to leverage the relaxation and rounding method to solve this challenge.

Assuming that after relaxing the binary constraints, the relaxed variables are $\hat{x}_i^k$, $\hat{w}_i^k$ and $\hat{G}_i^{m,k}$ corresponding to x_i^k, w_i^k and $G_i^{m,k}$, respectively. These three relaxed variables present how much device i can be repaired, repair percentage of device i and the availability of the m^{th} support of device i, which is determined by the minimal repair percentage of devices in this support, respectively. Then, the constraint (2) can be modified as

$$\hat{x}_i^k \leq \max_m \hat{G}_i^{m,k-1} \tag{8}$$

for all $i \in \boldsymbol{N}$ and $k = 1, \ldots, K$, while the constraint (3) becomes

$$\hat{G}_i^{m,k} \leq \min_{j \in \boldsymbol{S}_{im}} \hat{w}_j^k \tag{9}$$

Constraint (8) ensures that a device cannot be repaired more than the availability of all its supports, and constraint (9) means that the availability of a support is determined by the device with minimum repair percentage in this support. Though constraint (9) can be represented by a group of linear constraints

$$\hat{G}_i^{m,k} \leq \hat{w}_j^k \tag{10}$$

for all $j \in \boldsymbol{S}_{im}$, constraint (8) is not a convex form. In other words, the relaxed formulation is still difficult to solve. To make the problem tractable, we use

$$\hat{x}_i^k \leq \frac{\sum_m \hat{G}_i^{m,k-1}}{M_i} \tag{11}$$

for all device i and $k = 1, \ldots, K$ to approximate (8). Actually, constraint (11) is tighter than (8), which can compensate for the relaxation of binary variables.

After the above relaxation and reformulation, we obtain the following Linear Programming (LP) model (Relaxed Reserved Model, R2M), which can be solved:

minimize (1)
Subject to: (4) — (7), (10), (11)

Once we obtain the solution of R2M, we can round the partial repair scheme that tells us which device to repair "partially" first and which device repair partially next, and so on, and derive an integer scheme. In order to get a better solution, we adopt a progressive rounding method as shown in Algorithm 1. In this algorithm, we first solve the R2M model (Line 3) and then repair the device with maximum recovery fraction among all the devices that are ready for repair according to the R2M solution (Line 4). To check if a device is ready for repair at stage k, we should ensure that at least one of its supports works before stage k. To start repairing device n at stage l, we should add two more constraints $x_n^l = 1$ and $w_n^{l+t_n} = 1$ into the R2M (Line 5), to indicate that we start repairing device n at stage l and it will work from stage $l + t_n$. In addition, when all the devices in one of the supports for a device (say device n) works at stage k, we should remove the constraint (11) for device n to enable device n to be repaired at any stage after k (Line 6–8). Otherwise, a device i cannot be fully repaired until all its supports work, which is obvious not a good solution. Then, we re-solve the R2M, and find the next device to repair. This procedure will end till all the devices are repaired.

Algorithm 1: ILP Based Algorithm for RM

Input: a set of failed devices, set of supports of each such device, utility of each such device, resource available in each stage, resource requirement to recover each such device, time to repair each such device

Output: Recovery scheme

1: Formulate R2M model according to the input
2: **while** not all devices are alive **do**
3: Solve R2M model
4: $(n, l) \leftarrow \arg_{i,k} \max x_i^k$, such that device n can be repaired at stage k
5: $x_n^l \leftarrow 1$, $w_n^{l+t_n} \leftarrow 1$ and add this constraint into R2M
6: **if** all the devices in one of the supports of device n ($n \in \boldsymbol{N}$) starts working at stage k **then**
7: remove constraint (11) for device n in R2M from stage $k + 1$
8: **end if**
9: **end while**
10: **return** $\{x_i^k\}$

C. Dynamic Programming Based Algorithm

Although the above Algorithm 1 addresses the intractability of the original ILP model (RM), it still needs to solve LP problem many times, which is time consuming in large scale systems. For example, we can solve the system recovery problem in a system with 200 devices in about 3 hours on a desktop, while more than 10 hours in a system with 300 devices. Accordingly, a heuristic that is not based the above LP model is also necessary for extremely large size systems.

Generally, there are two main lines of thought to design a heuristic algorithm. The first one is to determine which devices to repair first, and which next and so on. This method can be called as a *forward algorithm*. However, it is difficult to

determine at stage k which devices to repair result in a higher overall utility gain in a future stage $j > k$. Accordingly, we adopt the second method, called *backward algorithm*, which stage by stage kill devices till all the failed devices are killed. The reverse order of killing the devices is exactly the desired recovery order. When killing a device, the algorithm uses the following rules:

- A device can be killed if it is not in the only working support of another working devices
- A device with the lower utility should be killed first

The first rule is to ensure the (reversed) repair order follows the dependencies (as a device cannot repair if none of its supports work). The second rule is to maximize the system utility during recovery.

Note that this is a backward Dynamic Programing (DP) scheme. To maximize the system utility during recovery, it is to minimize the loss of device utility. If a device has not worked at one stage, we loss the utility of this device in such stage. To kill a device is exactly to give up the utility of a device in order to maintain the utility of other devices. Accordingly, the DP can be formulated as

$$L(s-1) = \min_{i \in \boldsymbol{R}(s)} \{L(s) - u_i\} \tag{12}$$

where $L(s)$ is the loss of device utility before stage s, while $\boldsymbol{R}(s)$ is the set of devices that still work at stage s.

Following this idea, we can design a DP based algorithm as shown in Algorithm 2. From Line 2 to 6, we sequentially kill the devices to derive the recovery order. In Line 3, we get all the devices that can be killed and kill the device with minimum utility in Line 4. When all the failed devices are killed, the recover order is stored in L. After getting the device recover order, we assign resources to repair them from Line 9 to 18. We sequentially assign the repair resources the devices in L (Lines 10–12). For a given device, we first check if the dependency is satisfied. If not, we jump to repair the next devices (Line 14), as this case happens only when some repaired devices have not worked and it is not necessary to wait for the repair of these devices. If the dependency requirement is satisfied but there are not enough resources, we wait to the next stage to accumulate more resources (Line 16), since the killing procedure gives out the near optimal repair order. When all devices are repaired, the algorithm finishes and the devices to repair in each stage are recorded in $R[k]$.

Algorithm 2: DP based Algorithm for RM

Input: a set of failed devices $\boldsymbol{D}$, set of supports of each such device, utility of each such device $\{u_i\}$, resource available in each stage $\{r_k\}$, resource requirement to recover each such device $\{m_i\}$, time to repair each such device $\{t_i\}$

Output: Recovery scheme

1: Initialize $L \leftarrow \Phi$
2: **while** $\boldsymbol{D} \neq \Phi$ **do**
3: Get all the devices that are not in the last working support for arbitrary device, and form set $\boldsymbol{X}$
4: $i \leftarrow \arg\min_{i \in \boldsymbol{X}} u_i$
5: Push i to the front of L, $\boldsymbol{D} \leftarrow \boldsymbol{D} - i$
6: **end while**
7: $k \leftarrow 1$, $R[k] \leftarrow \Phi$ for all k, $r \leftarrow r_1$
8: $n \leftarrow L.front()$
9: **while** L is not empty **do**
10: **if** none of n's supports works at stage k **then**
11: $R[k] \leftarrow R[k] \cup n$, $L \leftarrow L - n$, $r \leftarrow r - m_n$
12: $L \leftarrow L.front()$
13: **else if** $r \geq m_n$ **then**
14: $n \leftarrow n.next()$
15: **else**
16: $k \leftarrow k + 1$, $r \leftarrow r + r_k$, $L \leftarrow L.front()$
17: **end if**
18: **end while**
19: **return** R

IV. Opportunistic (Repair Resource) Model

In this opportunistic model (OM), the devices in the system can start their repair with only a part of the required resources. The amount of resources that can be used to repair each device during each stage is limited, so the minimum number of stages needed to repair a device can also be the same as that in the reserved model. To formulate this scenario, we redefine the notation x_i^k to denote that device i can start to repair, i.e. allocate resources to it, in stage k. Then, compared with RM, constraint (4) is not required any more, as the repair procedure is formulated by the resource allocation. Nevertheless, we have to add two more constraints into the ILP model:

$$a_i^k \leq x_i^k e_i \tag{13}$$

and

$$\sum_{j \leq k} a_i^j \geq m_i w_i^k \tag{14}$$

In addition, constraint (7) should be reformulated as

$$\sum_i \sum_{j \leq k} a_i^j \leq \sum_{j \leq k} r_j \tag{15}$$

for $k = 1, \ldots, K$. Accordingly, the ILP for the OM can be summarized as:

maximize (1)
Subject to: (2), (3), (5), (6), (13) — (15)

A. ILP Based Algorithm for OM

Similar to Section III-B, we can relax above ILP and derive Relaxed Opportunistic Model (ROM):

maximize (1)
Subject to: (5), (6), (10), (11), (13) — (15)

Different from Algorithm 1, we cannot determine which devices to repair in each stage as we can only partially recover

some devices due to the per stage resource utilization constraint of each device. To maximize the system utility during recovery, the only decision variable that we can change is the amount of resource allocated to different devices in every stage. Accordingly, we cannot slightly modify Algorithm 1 to adopt to this scenario. Before presenting an ILP based Algorithm 3 for solving OM, we first give out a theorem and a guideline, which guide our algorithm design:

Theorem 1. *In the OM, the optimal solution must fully utilize all the available resources in each stage, unless the inter-dependency constraints prevent us from using the remaining resources.*

We will omit the formal proof, but intuitively, the reason this holds is that one can allocate the available repair resources to a device that can be repaired, which will accelerate its recovery without delaying the repair of other devices. Based on the Theorem 1, we also have the following design guideline:

Guideline 1. *In each stage, we should aggregate the resources to fewer devices such that more devices can be fully repaired, rather than distribute them among as more devices as possible.*

This guideline can be explained by an example. Say there are two devices C_1 and C_2 can be repaired in stage k. Each of them needs 2 units of resources to repair, but there is only 2 units of resources available in each stage. The first method to repair these two devices is first repair one of them, and then repair the other one in the next stage. The second method is to allocate 1 unit of resources to each of the devices in successive two stages. Obviously, the first method which repairs a device first can derive more utility during the recovery procedure.

Based on above theorem and guideline, we design Algorithm 3 to determine the amount of resources that should be allocated to repair each device stage by stage. In Line 2, r is used to record the remaining resources that can be used to repair devices. Therefore, it should be initialized as 0 at the beginning. Lines 4–6 are used to prevent resources being allocated on the devices that cannot be repaired in stage k. After that, we solve the ROM model in Line 8. In Line 9, we sort all the devices that should get more resource to recover, i.e. $a_i^k > 0$ in the ROM solution, based on the amount of remaining resources that each device needs to recover. The reason behind this operation is given by Guideline 1. At last, motivated by Theorem 1, we assign as many resources as possible to the devices that can be repaired in Lines 10–16.

B. DP Based Algorithm for OM

Though the ILP based algorithm for OM is drastically different from Algorithm 1 for RM, the DP also for OM can be similar to Algorithm 2 for RM with the following modifications:

- Set the repair duration of each device to be 0.
- Use the "for" loop at Lines 2–6 in Algorithm 2, we can get the device repair order and store them in L.

Algorithm 3: ILP Based Scheme for OM

Input: a set of failed devices $\boldsymbol{D}$, set of supports of each such device, utility of each device, resource available in each stage $\{r_k\}$, resource requirement to recover each device $\{m_i\}$, resource utilization constraint $\{e_i\}$

Output: Recovery scheme

1: Formulate ROM model according to the input
2: Initialize $r \leftarrow 0$
3: **for** $k = 1$ to K **do**
4: **for** all the devices i cannot be repaired at stage k **do**
5: add constraint $a_i^k = 0$ to ROM model
6: **end for**
7: $r \leftarrow r + r_k$
8: Solve ROM model
9: Sort all the devices i with $a_i^k > 0$ according to the m_i
10: **for** all the devices i with $a_i^k > 0$ **do**
11: add constraint $a_i^k = \min\{e_i, m_i, r\}$ to ROM model
12: $r \leftarrow r - a_i^k$, $m_i \leftarrow m_i - a_i^k$
13: **if** $r \leq 0$ **then**
14: **break**
15: **end if**
16: **end for**
17: **end for**
18: **return** $\{a_i^k\}$

- Use Lines 9–18 in Algorithm 2 to allocate the resources to devices. Note that now we can allocate part of the resources to a device.

V. Performance Evaluation

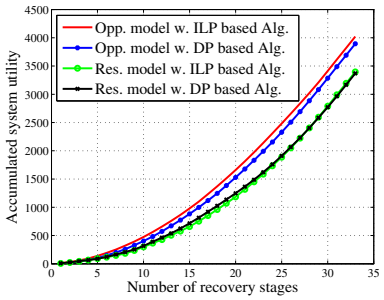
In this section, we investigate the performance of our proposed algorithms. We first use a small, representative system with default settings to compare different algorithms. Then, we study how parameters, such as the available resources and system size, impact the algorithm performance.

A. Default System

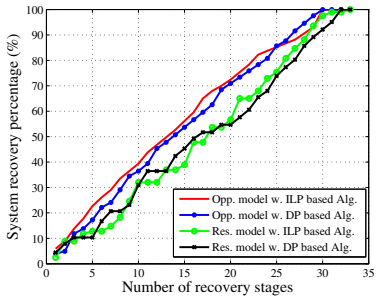
In the simulations, we assume there are 50 failed devices in the system, each device has at most 5 supports and its utility is evenly distributed [1,7]. For the reserved model, we assume the repair time, t, of each device is evenly distributed [1,4], while the maximum resources that can be consumed in each stage by each device, r, is evenly distributed [1,5] in the opportunistic model. Then, the repair resources required by a device is $t \times r$, i.e. a device in different models needs the same amount of resources to repair.

Fig. 1 shows how the accumulated system utility (Fig. 1(a)) and the system recover percentage change (Fig. 1(b)) during the recovery. From this figure, we make four observations:

Firstly, from Fig. 1(a), we can see that ILP based algorithm slightly outperforms DP based algorithm in both models, while this is not surprising, after a given stage, the accumulated system utility obtained by ILP based algorithm is not always larger than that obtained by DP based algorithm in the reserved model. This is due to the fact that the ILP based algorithm may



(a) Stage vs. accumulated utility



(b) Stage vs. recovery percentage

Fig. 1. How algorithms recover system

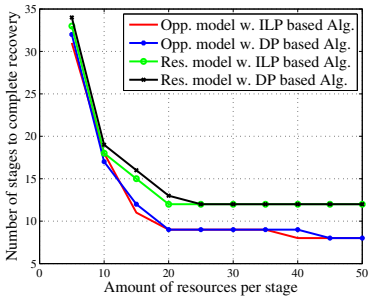


Fig. 2. How the amount of resources per stage impact the recovery process

TABLE II
ALGORITHM PERFORMANCE AND RUNNING TIME

Sys.size		100		200		300	
		Time (s)	Sys. uti.	Time (s)	Sys. uti.	Time (s)	Sys. uti.
Res. mod.	ILP based	602.413	19221	11743.37	68362	$> 10h$	NA
	DP based	0.33	16585	1.474	59428	2.907	128168
Opp. mod.	ILP based	5432.85	22237	$> 10h$	NA	$> 10h$	NA
	DP based	1.742	18107	4.38	61115	7.098	130496
# of stages		39		119		211	

save some repair resources for the latter stages to maximize accumulated system utility in the long term, though it may result in a lower accumulated utility in the short term.

Secondly, also from Fig. 1(a), we can see that opportunistic model results in a higher system utility during the system recovery. This is because that the opportunistic model is more flexible as it allows a device to be partially repaired in non-continuous stages, which is not allowed in the reserved model.

Thirdly, from Fig. 1(b), for a given type of algorithm, the recovery speed in the opportunistic model is faster than that in the reserved model. This is also intuitive as a device in reserved model may need to wait for a few more stages to have all the repair resources before being repaired.

Fourthly, however, even in the OM, the ILP based algorithm may result in a lower system recovery percentage than its DP based counterpart after certain stages, because the former tries to maximize the accumulated utility in the long term.

B. Impact of Available Resources Per Stage

Fig. 2 shows how the number of required recovery stages changes with the available repair resources per stage. Initially, this number reduces approximately linearly with the amount of available resource per stage. However, with more and more amount of available resources, the decreasing rate reduces. Eventually, the number of required stages is constraint by the device dependencies, and different algorithms need the same number of stages for complete recovery.

C. Impact of System Size

To study how our algorithms scale, we test the running time and algorithm performance in different size systems. The simulation results are summarized in Tab. II. From this table, we can see that the ILP based algorithm can support a system with ~ 100 failed devices for the opportunistic model, while supports more than 200 failed devices for the reserved model. This is because the opportunistic model has to deal with a larger decision space for partial repair.

When the number of failed devices is larger than 300, neither ILP based algorithms can obtain a solution within 10 hours. In this case, we leverage DP based algorithms can derive a solution within a few seconds on a laptop with an AMD 2.70 GHz Duo-Core CPU and 4 GB RAM. In fact, we have tested that the DP based algorithms can derive a solution in a system with 5000 failed devices within several minutes. In terms of performance, the DP based algorithms suffers 10% – 20% performance loss.

VI. Conclusion

In this paper, we have studied how to progressively recover from a large scale (cascased) failure in an interdependent CPS. Depending on when we can start repairing a device, we have considered two models: the reserved model and the opportunistic model. For both models, we have presented the ILP formulation, designed a relaxation and rounding method algorithm for the systems with less than 200 failed devices, as well as a DP based algorithm for the extreme large size systems with over 5000 failed devices. Simulations have shown than ILP based algorithm can derive a better solution, while DP based algorithms can support much larger systems.

References

[1] J. Wang, C. Qiao, and H. Yu, "On progressive network recovery after a major disruption," in *INFOCOM, 2011 Proceedings IEEE*.

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.