2019

Uncertainty Theory Based Reliability-Centric Cyber-Physical System Design

Houbing Song
Embry-Riddle Aeronautical University, songh4@erau.edu

Ya Jiang
Tsinghua University

Mingzhe Wang
Tsinghua University

Xun Jiao
University of California - San Diego

Hui Kong
Tsinghua University

See next page for additional authors

Authors
Houbing Song, Ya Jiang, Mingzhe Wang, Xun Jiao, Hui Kong, Rui Wang, Yongxin Liu, Jian Wang, and Jiaguang Sun

Uncertainty Theory Based Reliability-Centric Cyber-Physical System Design

Yu Jiang*, Mingzhe Wang*, Xun Jiao†, Houbing Song‡, Hui Kong*, Rui Wang*, Yongxin Liu‡, Jian Wang‡ and Jiaguang Sun*
*School of Software, Tsinghua university, Beijing
†School of Computer Science and Technology, UCSD, USA.
§ Department of ECSSE, Embry-Riddle Aeronautical University, USA.

***Abstract*—Cyber-physical systems (CPSs) are built from, and depend upon, the seamless integration of software and hardware components. The most important challenge in CPS design and verification is to design CPS to be reliable in a variety of uncertainties, i.e., unanticipated and rapidly evolving environments and disturbances. The costs, delays and reliability of the designed CPS are highly dependent on software-hardware partitioning in the design. The key challenges in partitioning CPSs is that it is difficult to formalize reliability characterization in the same way as the uncertain cost and time delay.**

In this paper, we propose a new CPS design paradigm for reliability assurance while coping with uncertainty. To be specific, we develop an uncertain programming model for partitioning based on the uncertainty theory, to support the assured reliability. The uncertainty effect of the cost and delay time of components to be implemented can be modeled by the uncertainty variables with uncertainty distributions, and the reliability characterization is recursively derived. We convert the uncertain programming model and customize an improved heuristic to solve the converted model. Experiment results on some benchmarks and random graphs show that the uncertain method produces the design with higher reliability. Besides, in order to demonstrate the effectiveness of our model for in coping with uncertainty in design stage, we apply this uncertain framework and existing deterministic models in the design process of a sub-system that is used in real world subway control. The system implemented based on the uncertain model works better than the result of deterministic models. The proposed design paradigm has the potential to be generalized to the design of CPSs for greater assurances of safety and security under a variety of uncertainties.

***Index Terms*—cyber-physical system; hardware-software partitioning; uncertain programming; reliability-centric.**

I. Introduction

Cyber-physical systems (CPSs) are built from, and depend upon, the seamless integration of software and hardware components with embedded sensors, processors and actuators [1], [2], [3]. In [4] a significant cloud-edge computing framework is proposed for cyber-Physical-Social System services including the cyber-physical systems design. Furthermore, a systematic big data-as-a-service CPS framework was presented in [5], where there are many challenges about data processing were discussed. The most important challenge in CPS design and verification is to design CPS to be reliable in a variety of uncertainties, i.e., unanticipated and rapidly evolving environments and disturbances. How to find an efficient partition for the hardware implementation and software implementation of the CPS remains challenging.

Recently, many research efforts have been undertaken to automate this task. Those efforts include exact partitioning [6], [7], [8] and heuristic partitioning models[9], [10], [11], [12], [13]. But few algorithms address the issue that we cannot accurately determine the cost and time of system components, and few works take the reliability characterization into account. Reliability analysis is complex but has significant benefits in terms of design quality of complex safety-critical CPS. These drawbacks limit the application to many real system designs. Hence, there has been a recent surge for methods to handle those uncertainty effects with reliability in consideration.

In this paper, we propose a new CPS design paradigm for reliability assurance while coping with uncertainty. To be specific, we develop an uncertain programming model to cope with uncertainty and to characterize reliability in partitioning. In our model, the partitioning problem is formulated as a mathematical optimization equation, where the delay related constraints and the cost related objective are defined on uncertain variables, and the reliability characterization is recursively derived from the task graph of the system. The proposed design model has the potential to be generalized to the design of CPSs for greater assurances of safety and security under a variety of uncertainties. Experiments on benchmarks with all parameters deterministic demonstrate the compatibility and reliability improvement of the uncertain model with the existing models, and experiments on a real-world subway control system design with all parameters unknown demonstrates the effectiveness of our model for the uncertainty effects in design.

The paper is organized as follows: related work is presented in II. The proposed uncertainty model and the model conversion are presented in Section III. Section III-C presents the solution of the proposed model. Empirical results on some benchmarks and a real system design are given in Section IV, and we conclude in Section V.

II. Related work

There are many existing models and algorithms for partitioning, typically can be classified as exact partitioning and heuristic partitioning models. The family of exact algorithm includes branch-and bound [6], integer linear programming [7] and dynamic programming [14], [8], and the heuristic algorithm includes genetic algorithm and simulated annealing [15], [16], [17], [18].

All these algorithms presented above work perfectly within their own co-design environments. But their parameters of components are all deterministic, while in the design stage, the cost and time of those components cannot be determined in the design stage, especially for the software components. Some people think that they are subjective probability, and make use of this theory in system level partitioning [19], [20]. However, they focus on the project management, about the probabilistic implementation cost and delay time of different develop teams for each system module. They do not consider the actual implemented system. They may consider the communication time between two develop team, and do not consider the actual communication time between two task modules.

Besides, few prior studies take the reliability [21], [22], [23] in account in partitioning. A lot of surveys presented in [24] showed that those imprecise quantities behave neither like randomness nor like fuzziness. Hence, based on some preliminary idea presented in our previous work [25], we will conduct the system partitioning with the reliability in consideration. Based on the uncertainty theory [24], [26], [27], we can measure the belief degree of uncertain events. Other factors like balanced performance or social utility of CPS are also considered for the system design. A comprehensive framework for data process is proposed in [28], where the balanced benefits for multiple stakeholders are guaranteed in multiple scenarios. Meanwhile, the study in [29], [30], [31] presents an efficient heuristic method for physical data processing, providing a thorough consideration between the physical and social aspects of CPS.

III. UNCERTAINTY MODEL

This section presents the CPS reliability-centric partitioning problem statement and the proposed model.

A. Problem Definition

Based on the uncertainty theory [24], the system is formalized as $G(V, E)$, where V is the set of nodes $\{v_1, v_2 \cdots v_n\}$ and E is the set of edges $\{e_{ij} | 1 \leq i, j \leq n\}$:

1) ξ_i^h denotes the cost of node i in the hardware implementation ways that in hardware or software.
2) Φ_{ci}^h is the linear uncertainty distribution of uncertain variables ξ_i^h, denoted by $\zeta(a_{ci}^h, b_{ci}^h)$, where a_{ci}^h, b_{ci}^h are nonnegative real numbers.
3) t_i^h denotes the execution time of node i in hardware implementation, and t_i^s denotes the execution time of node i in software implementation.
4) Φ_{ti}^h, Φ_{ti}^s are uncertain distributions of uncertain variables t_i^h, t_i^s, denoted by $\zeta(a_{ti}^h, b_{ti}^h)$, $\zeta(a_{ti}^s, b_{ti}^s)$, where a_{ti}^h, b_{ti}^h, a_{ti}^s, b_{ti}^s are real numbers.
5) c_{ij} denotes the communication time between node i and j. The value of c_{ij} is given in the context of the two nodes implemented differently.
6) r_i^h denotes the reliability of node i in hardware implementation, and r_i^s denotes the reliability of node i in software implementation.

The partitioning problem is to find a bipartition P, where $\mathrm{P} = (V_h, V_s)$ such that $V_h \bigcup V_s = V$ and $V_h \bigcap V_s = \emptyset$. In this paper, partitioning is scaled as: T_0 is the given execution time constraint, and R_0 is the given system reliability constraint. The objective is to find a hardware-software partition P such that $T_X \leq T_0$, $R_X \leq R_0$ and H_X is the minimal hardware cost.

B. Problem Formalization

A partition is characterized and scaled to three metrics: cost, time and reliability. The cost includes hardware cost and software cost. It represents the resource consumption to achieve the hardware and software implementation of each task module. The time delay includes the execution time of each node and the communication time between nodes. The reliability is the probability that the system will perform its intended function correctly during a specified period of time.

First, let us consider the **cost metric**. If a given node is partitioned to be in hardware implementation, the hardware cost of the node is considered. Otherwise, the software cost of the node is considered. Then, the total hardware cost with respect to a dedicated partition can be calculated as the sum of nodes in hardware implementation. The additive calculation rule for resource consumption cost is reasonable in most cases of the computation model. Hardware cost $H(\mathbf{x})$ of the partition $P(\mathbf{x})$ can be formalized as follow:

$$H(\mathbf{x}) \quad = \quad \sum_{i=1}^{n} \xi_i^h (1 - x_i)$$

Then, let us consider the **time metric**. It consists of two parts: execution time of each node, and the communication time between nodes. We give a reasonable assumption that the communication cost between node i and j is 0 when the two nodes are partitioned into the same implementation way. Based on the assumption, the execution time $T_e(\mathbf{x})$, communication time $T_c(\mathbf{x})$, and the total time metric $T(\mathbf{x})$ can be formalized as:

$$\begin{aligned} T_e(\mathbf{x}) &= \sum_{i=1}^{n} t_i^s x_i + t_i^h (1 - x_i) \\ T_c(\mathbf{x}) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} [(x_i - x_j)^2] \\ T(\mathbf{x}) &= T_e(\mathbf{x}) + T_c(\mathbf{x}) \end{aligned}$$

Finally, let us see the **reliability metric**. The node that has no outgoing arcs is regarded as the destination node and the output of the system, while the nodes that have no incoming arcs are regarded as the start nodes. The adjacency matrix $Relation[n][n]$ is used to represent the dependency relations of parent tasks and child tasks in the directed acyclic graph. Based on the theory of fault tree and reliability block diagram, the reliability of the task node x_i

Algorithm 1: *System_Reliabiliby*

```
Input: The directed acyclic task graph of the system.
Output: The reliability formalization of the system
Int Relation[n][n] /* Relation matrix */ ;
for i ← 1 to n do          /* initialize the relation matrix */
    for j ← 1 to n do
        if there is an edge between node x_i and x_j then
            Relation[i][j] ← 1
        end
    end
end
R ← Recursive(n)            /* Reliability derivation */;

Recursive(m)        /* Definition of the recursive function */ ;
for i ← 1 to n do
    if Relation[i][m] == 1 then
        R_m ← R_m·(r^s_m x_m + r^h_m (1−x_i))·Recursive(i)
    end
end

return R
```

is the product of its parent nodes and itself. The system reliability R is derived in a recursive manner as presented in the algorithm.

Based on the formalization, the given constraint M on execution time, and R_0 on the reliability, the partitioning problem can be modeled as :

$$P_0 : \begin{cases} minimize & H(\mathbf{x}) \\ subject\ to & T(\mathbf{x}) \leq M \\ & R(\mathbf{x}) \geq R_0 \\ & \mathbf{x} \in \{0,1\}^n \end{cases}$$

We note that minimizing the value of $H(\mathbf{x})$ is equivalent to maximizing the value of $\sum_{i=1}^{n} h_i x_i$. Hence, the solution of the problem P_0 is equal to that of the problem P_1:

$$P_1 : \begin{cases} maximize & \sum\limits_{i=1}^{n} \xi_i^h x_i \\ subject\ to & \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} c_{ij}[(x_i - x_j)^2)] + \\ & \sum\limits_{i=1}^{n} (t_i^s - t_i^h) x_i \leq M - \sum\limits_{i=1}^{n} t_i^h \\ & 1 - Recursive(n) \leq 1 - R_0 \\ & \mathbf{x} \in \{0,1\}^n \end{cases}$$

We convert the uncertain objective function first. It has been proved that: $\xi_1, \xi_2 \cdots \xi_n$ are uncertain variables with uncertain distributions $\Phi_1, \Phi_2 \cdots \Phi_n$. The function $f(\mathbf{x}, \xi_1, \xi_2\ \cdots \xi_n)$ is strictly increasing with respect to $(\xi_1, \xi_2 \cdots \xi_m)$ and strictly decreasing with respect to $\xi_{m+1}, \xi_{m+2} \cdots \xi_n$. Then, the converted expected objective function can be calculated as:

$$E[f\big(\mathbf{x}, \xi_1, \xi_2 \cdots \xi_n)] = \int_0^1 f\Big(\mathbf{x}, \Phi_1^{-1}(\alpha), \Phi_2^{-1}(\alpha) \\ \cdots \Phi_m^{-1}(\alpha), \Phi_{m+1}^{-1}(1-\alpha) \cdots \Phi_n^{-1}(1-\alpha)\Big) d\alpha$$

Hence, the objective function of the problem P_2 can be converted to:

$$\begin{aligned} & max \quad \sum_{i=1}^{n} \xi_i x_i \\ \Rightarrow \quad & max \quad \sum_{i=1}^{n} [\Big(\int_0^1 \Phi_i^{-1}(\alpha) d\alpha\Big) x_i] \end{aligned}$$

where $\Phi_i^{-1}(\alpha)$ is equal to $(1-\alpha)(a_{ci}^h - b_{ci}^s) + \alpha(b_{ci}^h - a_{ci}^s)$.

Then, we convert the uncertain constraint. It has been proved that: $h_0(x), h_1(x) \cdots h_n(x)$ are real-valued functions, $h_i^+(x)$ is defined as $(|h_i(x)| + h_i(x))/2$ and $h_i^-(x)$ is defined as $(-|h_i(x)| + h_i(x))/2$. The converted constraint of $M\{\sum_{i=1}^{n} h_i(x)\xi_i < h_0(x)\} \geq a$ can be calculated as:

$$\sum_{i=1}^{n} h_i^+(x) \Phi_i^{-1}(a) - \sum_{i=1}^{n} h_i^-(x) \Phi_i^{-1}(1-a) \leq h_0(x)$$

Hence, the uncertain constraint of the problem P_1 can be converted as follows:

$$\begin{aligned} & \sum_{i=1}^{n} T_i x_i \leq M' \\ \Rightarrow \quad & M\{\sum_{i=1}^{n} T_i x_i \leq M'\} \geq 1 \\ \Rightarrow \quad & \sum_{i=1}^{n} (b_i) x_i + \leq a_0 \end{aligned}$$

where b_i is equal to $b_{ti}^s - a_{ti}^h$ and a_0 is equal to $M - \sum_{i=1}^{n} a_{ti}^h - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij}[(x_i - x_j)^2)]$. The calculation process is the same as the proof process of theorem 1.

Then, the final version of the converted problem is:

$$P_2 : \begin{cases} maximize & \sum\limits_{i=1}^{n} [\Big(\int_0^1 \Phi_i^{-1}(a) d\alpha\Big) x_i] \\ subject\ to & \sum\limits_{i=1}^{n} (b_{ti}^s - a_{ti}^h) x_i + \\ & \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} C_{ij}(x_i - x_j)^2 \leq M - \sum\limits_{i=1}^{n} a_{ti}^h \\ & 1 - Recursive(n) \leq 1 - R_0 \\ & x_i \in \{0,1\}; i = 1, 2 \cdots n. \end{cases}$$

where b_i is equal to $b_{ti}^s - a_{ti}^h$ and a_0 is equal to $M - \sum_{i=1}^{n} a_{ti}^h - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij}[(x_i - x_j)^2)]$.

For safety-critical applications, the partition problems can also be formalized to maximize the reliability of the system, with the limited hardware resource and time constraints. The converted formalization is as follow:

$$P_3 : \begin{cases} maximize & Recursive(n) \\ subject\ to & \sum_{i=1}^{n}(b_{ti}^{s} - a_{ti}^{h})x_i + \\ & \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C_{ij}(x_i - x_j)^2 \leq M - \sum_{i=1}^{n} a_{ti}^{h} \\ & \sum_{i=1}^{n}[\Big(\int_0^1 \Phi_i^{-1}(a)d\alpha\Big)(1-x_i)] \leq N \\ & x_i \in \{0,1\}; i = 1, 2 \cdots n. \end{cases}$$

Algorithm 2: *Customized_ALG2*

```
      /* iteration-block */;
while termination conditions do
    while number of individuals ≤ number of the
    generation size do
        Select (g1, g2) from the current generation;
        Perform crossover on (g1, g2) to produce two
          new individuals (g1', g2') ;

          /* annealing-crossover */;
        if (max{fitness(g1'), fitness(g2')} ≤
          max{fitness(g1), fitness(g2)}) then
            △ C = max{fitness(g1'), fitness(g2')} -
              max{fitness(g1), fitness(g2)};
            if min{1, exp(-△C/Tk)} ≥ random[1, 0)}
              then
                Accept the crossover;
            else
                g1' = g1, g2' = g2;
            end
        else
            Accept the crossover;
        end
          /* annealing-mutation */;
        Perform mutation on g1' to produce ng1;
        if (fitness(ng1) ≤ fitness(g1')) then
            △ C = (fitness(ng1) - fitness(g1'));
            if min{1, exp(-△C/Tk)} ≥ random[1, 0)}
              then
                Accept the mutation;
            else
                ng1 = g1';
            end
        else
            Accept the mutation;
        end
        Perform step 19-29 on g2' to produce ng2;
          /* individual-selection */;
        if the highest fitness of the current generation
          ≥ fitness(solution) then
            Copy the individual to the solution;
        end
    end
end
return solution: x[i], i ∈ [1, n];
```

C. Problem Solution

After we formalize the partitioning problem as P_2 and P_3, many general-purpose heuristic algorithms presented in the related work section can be applied to solve the problems. Because the reliability formalization results in a more complex optimization problem, which cannot be simplified to the knaspack problem, the domain-specific based acceleration algorithms cannot be applied. In this paper, the general purpose algorithm [32], [9] based on the genetic algorithm and simulated annealing algorithm is customized to get the final result of the system design. First, we apply the original genetic algorithm (ALG1) presented in [33] to the uncertain partitioning problem P_2. Then, the enhanced algorithm (ALG2) presented in our previous work [32], [9] is also customized to solve the problem.

IV. Performance Evaluation

In order to demonstrate the performance of the uncertain model, we have implemented the two algorithms in C, and test the algorithms on Intel i5 2.27GHZ PC. First, we apply these two algorithms and the uncertain model to some benchmarks and random graphs, to show that the proposed uncertain model (P2) and algorithm 2 (ALG2) produce quality partitions, which are compatible and more reliable than the existing deterministic model (PE) and algorithm 1 (ALG1). In order to demonstrate the reliability effect on the partitioning, we conduct some experiments on the model (P3) and the deterministic model (PE) without reliability constraint. The results show that the reliability is improved significantly. Finally, we also apply our uncertain model (P2) and existing deterministic model (PE) in the design process of a sub-system that is used in real world subway control. We implement two different versions of the control system according to the partitioning results, and find that the sub-system implemented according to the partitioning results of P2 works better than the results of the deterministic model (PE).

A. Experiments on Benchmarks

The test cases are some benchmarks from [34], [35], and several random instances with different nodes and metrics. The TGFF is used to generate the general-purpose graphs. Some descriptions are listed in the Fig 1. The second column is the number of task modules, and the third column is the the number of edges denoting the communication.

The technology library [35] provides different values of time, cost, and reliability for the deterministic model (PE). While for the hardware cost value, the software execution time, the hardware execution time, the communication edges and the reliability of the uncertain model, more efforts are needed to initialize. Because we use the uncertain distribution to convert the uncertain model P_1 to the final deterministic model P_2, we need to initialize two variables to model the intervals for the uncertain distribution. For all $i \in [1, n]$, they are generated with the following rules:

Name	Node	Edge	Description
crc32	25	34	32-bit cyclic redundancy check. From the Telecommunications category of MiBench
patricia	21	50	Routine to insert values into Patricia tries. From the Network category of MiBench.
dijikstra	26	71	Computes shortest paths in a graph. From the Network category of MiBench.
clustering	150	333	Image segmentation algorithm. From a medical application.
rc6	329	448	RC6 cryptographic algorithm.
random1	500	1000	random generated graph
random2	1000	2000	random generated graph
random3	1500	3000	random generated graph
random4	2000	4000	random generated graph
random5	2500	5000	random generated graph
random6	3000	6000	random generated graph

Fig. 1. Test cases descriptions, including some benchmarks and some random dependency graphs.

- When the hardware cost is available, $a_{ci}^h = b_{ci}^h = value_1$. Otherwise, a_{ci}^h is generated as uniform random numbers in [0, 100] and b_{ci}^h is set as $a_{ci}^h + \beta_{ci}^h$. β_{ci}^h is a constant real number.
- When the execution time is available, $a_{ti}^h = b_{ti}^h = value_2$ and $a_{ti}^s = b_{ti}^s = value_3$. Otherwise, a_{ti}^h is generated as uniform random numbers in [0, 10] and b_{ti}^h is set as $a_{ti}^h + \beta_{ti}^h$. (In most cases, the hardware execution is so fast that the time is usually set as 0). The software execution time a_{ti}^s is generated as uniform random numbers in [b_{ti}^h, 100] and b_{ti}^s is set as $a_{ti}^s + \beta_{ti}^s$. β_{ti}^h is a constant.
- When the communication time is available, $c_{ij} = value_4$. Otherwise, c_{ij} is generated as uniform random numbers in [0, $2 \cdot \rho \cdot \max(b_{ti}^s)$]. We can find that the communication time has an expected value of $\rho \cdot \max(b_{ti}^s)$, where ρ is the so-called communication to computation ratio in the general partitioning problem. We conduct our experiment with two values $\rho = 0.1, 1$. The reliability parameters r_i^h, r_i^s and constraint R_0 are initialized in the similar way.
- M is the execution time constraint. It is generated as uniform random numbers in $[\sum_1^n a_{ti}^h,\ \sum_1^n b_{ti}^s]$. We test two constraints (strict time constraint and loose real-time constraint) for each partitioning instance. The first M_1 is chosen from $[\sum_1^n a_{ti}^h,\ \frac{1}{2}\sum_1^n b_{ti}^s]$, and the second M_2 is chosen from $[\frac{1}{2}\sum_1^n b_{ti}^s,\ \frac{1}{2}\sum_1^n b_{ti}^s]$.

Then, we simulate the two algorithms on the benchmarks and the random graphs, for different values of ρ, M and R_0. For the benchmarks, the values can be initialized as available values as described above. For the random graphs, the value can be initialized in the random generated value as the rules. In order to demonstrate the difference between deterministic and uncertain theory, we build the deterministic model using the method described in the section III-B for comparison. We just need to replace the uncertain variables with deterministic variables in P_1 to get P_1^d, and initialize these deterministic parameters of PE_1^d with the expectation or the values in the technology library. The deterministic variables fit the probabilistic distribution among the random generated interval described in the rules. Then, PE_1^d can also be solved by the customized algorithm. Each instance is tested for 100 times. The averaged values of the object function $E[\mathbf{f}(\mathbf{x}, \xi)]$ of P_2 and the object function of PE_1^d are denoted by P_2_ALG1, P_2_ALG2, and PE_ALG2, respectively.

The results are presented in Fig. 3, 4, 5, 6 for different parameters configurations. We find that for the benchmarks with the parameters available, the results of the P_2_ALG2, and PE_ALG2 are almost the same, because the initialization rules such as $a_{ci}^h = b_{ci}^h = value_1$ make the P_2 and PE_1^d the same. That means that the uncertain model can also deal with the deterministic partitioning problem with the strategy of the parameters initialization rules described above. For the random graphs, the results of the P_2_ALG2, and PE_ALG2 are different with 2% deviation, because the uniform random generated parameters make the P_2 and PE_1^d different. We convert the P_1 to P_2 according to the uncertain theory with the uncertain phenomenon in consideration, while PE_1^d is just the expectation of each interval. For bigger size of nodes, ALG2 outperforms ALG1. ALG2 can generate smaller values than ALG1. With the increase in the size, the deviation between the two algorithms grows bigger. The values of the ρ and M have no effect on the performance of the two algorithms.

The results presented in the Figures 7, 8 store the convergence track of the two algorithms. At the beginning of the iteration procedure, ALG1 drops faster than ALG2. But ALG2 can find the near optimal solution faster than ALG1 in the convergence process. The iteration number grows with the size of the nodes, which means more time to go into the stable state. The time consumption of those algorithms is concluded in Figure 9.

From the above results, we can draw the conclusion that the proposed uncertain model (P2) and the customized algorithm 2(ALG2) produce quality partitions, which are not only compatible to but also more reliable than the existing deterministic model(PE) and algorithm 1(ALG1).

For the effect of reliability on partitioning, the result of the proposed model P_3 is compared with the result of the traditional model PE_1^d. Based on the parameters described in the technology library [35], [21], the partitioning results for systems are presented in the TABLE I. From the last column of the table, we can see that the reliability of the system has been improved through the partitioning.

B. Experiments on Real System Design

We conduct some experiments on a train communication control system that is used in real world subway systems to show the effectiveness of our model for the uncertainty effects in the design stage. The train control system is a safety-critical CPS described in the standard IEC 61375 [36], [37], [38]. The system consists of two controllers: multifunction vehicle bus(MVB) controller which interconnects devices within a vehicle, and wire train bus(WTB) controller which interconnects the vehicles of a train. The controllers connected in the train will transport two classes of data: (1)Time-critical, short Process_Data (used for

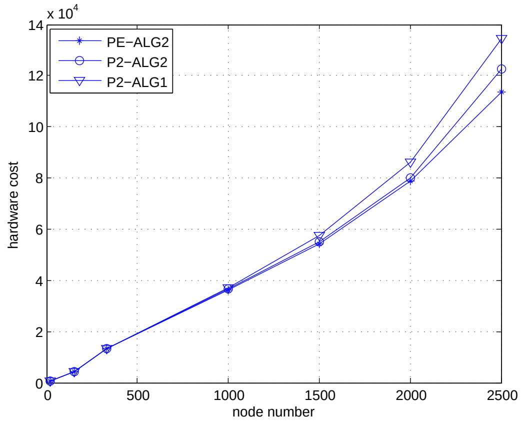

Fig. 2. $\rho = 0.1,\ M = M_1$

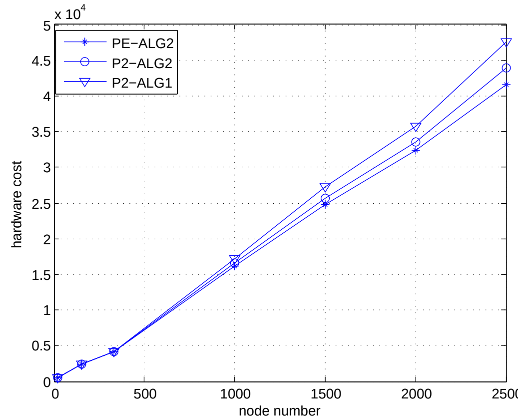

Fig. 3. $\rho = 0.1,\ M = M_2$

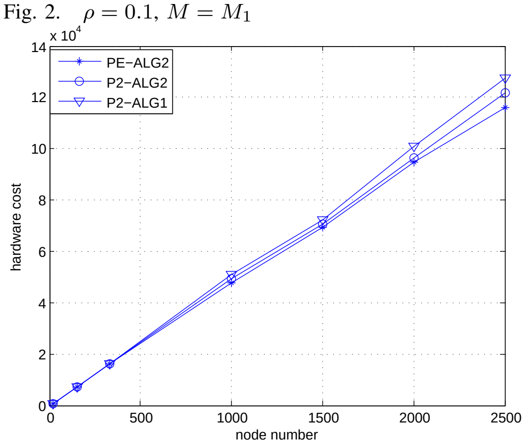

Fig. 4. $\rho = 1,\ M = M_2$

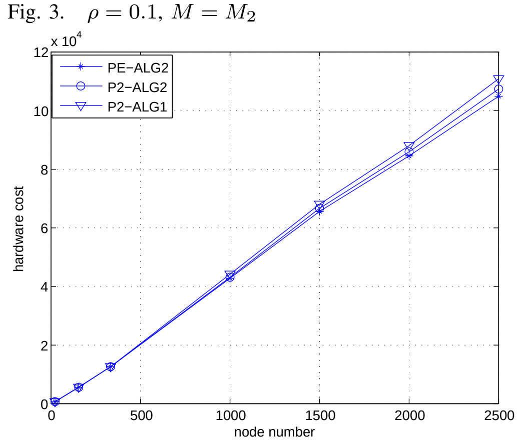

Fig. 5. $\rho = 1,\ M = M_2$

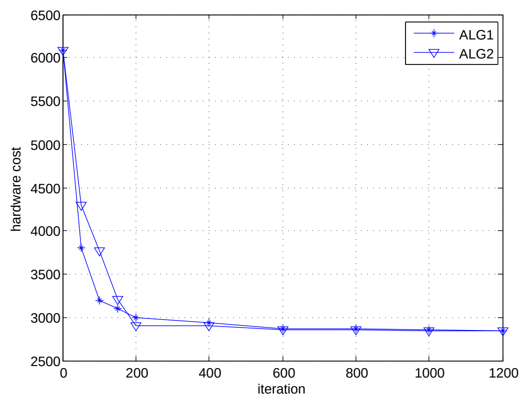

Fig. 6. convergence track for node number equals 100

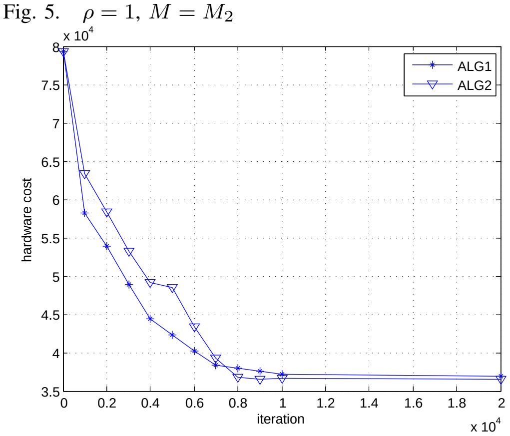

Fig. 7. convergence track for node number equals 1000

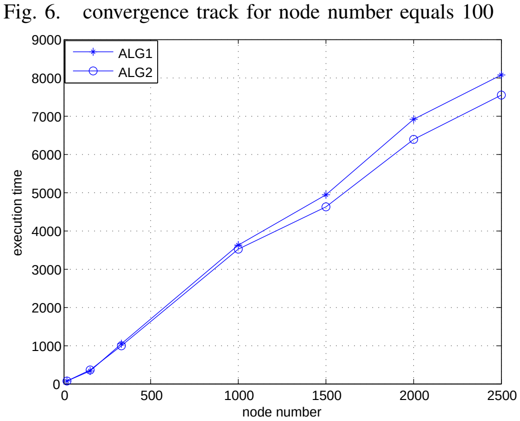

Fig. 8. Run time of the algorithm

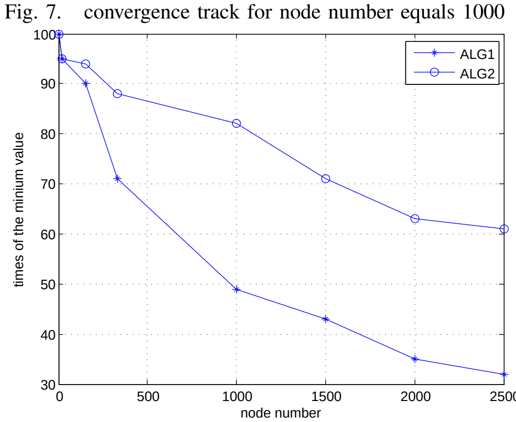

Fig. 9. Frequency of the lowest value

TABLE I
CPS COMPONENTS DESIGN WITH CONSIDERING RELIABILITY OR NOT.

Node	Bounds		Considering R			Out of considering R			Improving %		
	T	Cost	T	Cost	R	T	Cost	R	T	Cost	R
6	150	200	142	160	0.824981	122	180	0.720446	-14.0	12.5	12.7
	140	200	135	177	0.799978	122	180	0.720446	-9.6	1.7	11.1
	130	220	127	210	0.816563	94	205	0.713019	-25.9	2.5	12.7
7	150	250	149	220	0.752544	129	223	0.650345	-13.4	5.9	13.5
	145	265	121	245	0.744786	101	265	0.664037	-16.5	8.2	10.9
	140	270	134	270	0.744865	101	265	0.664037	-16.4	1.9	10.9
11	270	350	269	336	0.470365	253	350	0.437090	-6.3	4.2	7.6
	250	370	239	361	0.465516	249	351	0.410719	4.2	-2.9	11.8
	245	365	239	361	0.465516	240	353	0.385313	0.5	-2.3	17.0
14	350	430	349	405	0.391591	323	429	0.252508	-8.1	5.9	34.2
	330	430	328	426	0.383728	323	429	0.252508	-1.6	0.7	34.2
	350	410	349	405	0.391591	337	409	0.255511	-3.6	1.0	33.4
22	570	650	570	589	0.144126	544	619	0.110657	4.8	-5.1	23.2
	560	630	559	602	0.130903	528	629	0.117756	-5.9	4.5	10.1
	535	635	529	627	0.122806	524	634	0.078451	-1.0	1.1	36.1
25	670	670	627	629	0.132518	625	669	0.073722	-6.7	6.4	44.4
	680	650	680	622	0.127109	650	647	0.069255	-4.6	4.0	53.3
	660	660	669	641	0.125744	641	659	0.067871	-3.0	2.8	46.0
58	1850	1850	1675	1593	0.006149	1828	1446	0.003227	-8.4	-10.2	47.5
	1650	1650	1643	1613	0.006408	1646	1558	0.003496	0.2	-3.5	45.4
	1750	1750	1737	1509	0.006213	1748	1451	0.003220	0.6	-4.0	48.2

traction control including speed information, etc.); (2)Less urgent, but possibly lengthy Message_Data (used for diagnostics including device status information, etc.).

We conduct some experiments on the MVB controllers. They are controlled by one master, which controls the sending of the Process_Data and Message_data. All the data are transferred as frame on the link layer bus. The Master broadcasts a Master_Frame, which carries the identifier of a Process_Data frame. The device which sends these Process_Data and Message_Date responds by broadcasting a Slave_Frame, which is received by all other devices. If several devices respond to General_Event_Request, a collision occurs. The Master starts an Event_Arbitration to single out a device. In order to accomplish the functions above, the MVB controller can be designed as eighteen modules. For example, the Event_Arbitration function can be mapped to the Event arbitration module. Each module can be implemented by hardware and software. The hardware implementation can be described by VHDL and synthesized, and the software implementation can be described by C. The synthesized VHDL code can be loaded into an FPGA processor, and the C code can be loaded into an ARM processor. The two processors communicate with each other through GPIO.

As defined in the standard IEC 61375, the controller should send a slave frame when receiving a master frame, within 2-6us. The suggested time interval should be 3us. We set the time limit of the system as 3us. Then, the problem occurs. In order to satisfy the time requirement, some module should be implemented by hardware, and some module should be implemented by software. We do not know the parameters of each module exactly, and can only describe these parameters according to the pseudo code of the standard IEC 61375. The frequency of the FPGA processor is 24MHZ, and the frequency of the ARM processor is 18MHZ. Usually, the sender module and receiver module are implemented in hardware. We implement these two modules and test the time of the two modules, 4 periods and 5 periods, respectively. Then, we use these data to estimate the possible parameters according the pseudo code size described in the standard. For example, the send judgement module deciding which kind of slave frame needs to be sent, is supposed to run 3 periods. The initial process data module is supposed to take 9 periods to finish. All the other modules can be estimated. The communication between the FPGA processor and the ARM processor is supposed to take on period of ARM processor. Then, we can take these values to initialize the partitioning problem PE directly. The parameters initialization for uncertain model P2 is the same as the method described in the previous subsection, except that the distribution of these uncertain time and cost variables are normal uncertainty distribution.

Then, we can solve the two models. In the deterministic model, the Event arbitration, Device Scan, Master transfer, Device synchronization, System initialization, Message service, and Initial process data modules are supposed to be implemented in software. In the uncertain model, the Event arbitration, Device Scan, Master transfer, Device synchronization, and System initialization modules are supposed to be implemented in software. We implement the system according to the two different partitions, and connect the two implemented MVB controllers. The two controller can communicate with each other through the MVB bus. We use an oscilloscope to sample the data from the serial port that is connected to the MVB bus. Both of them receive and send the correct frame, but the time interval is different. The implemented system according to the deterministic model does not satisfy the time requirements, while the one according to the uncertain model works well. The system implemented according to the uncertain model has been deployed in a real subway control and run 20,000 miles without time-out error.

V. CONCLUSION

In this paper, we propose a new CPS design paradigm for reliability assurance while coping with uncertainty. To be specific, we develop an uncertain programming model for partitioning based on the uncertainty theory, to support the assured reliability. We also conduct some experiments on benchmarks and real complex system design to demonstrate the effectiveness of the proposed model and algorithm, especially for the significant improvement of the reliability. The results show potential usage of our model to improve the dependability of system. In the future, we plan to focus on modeling the relationship and interaction between the hardware and software, for greater assurance of reliability. Furthermore, we plan to consider the safety as well as the security in CPS design.

REFERENCES

[1] S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, "Industrial internet of things," pp. 1–715, 2017.

[2] H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, Cyber-Physical Systems: Foundations, Principles and Applications. Boston, MA: Academic Press, 2016.

[3] Y. Jiang, H. Song, R. Wang, M. Gu, J. Sun, and L. Sha, "Data-centered runtime verification of wireless medical cyber-physical system," IEEE Transactions on Industrial Informatics, vol. 13, no. 4, pp. 1900–1909, Aug 2017.

[4] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," IEEE Communications Magazine, vol. 55, no. 11, pp. 80–85, 2017.

[5] X. Wang, L. T. Yang, H. Liu, and M. J. Deen, "A big data-as-a-service framework: State-of-the-art and perspectives," IEEE Transactions on Big Data, vol. 4, no. 3, pp. 325–340, 2017.

[6] K. Chatha and R. Vemuri, "Hardware-software partitioning and pipelined scheduling of transformative applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 3, pp. 193–208, 2002.

[7] R. Niemann and P. Marwedel, "An algorithm for hardware/software partitioning using mixed integer linear programming," Design Automation for Embedded Systems, vol. 2, no. 2, pp. 165–193, 1997.

[8] J. Madsen, J. Grode, P. Knudsen, M. Petersen, and A. Haxthausen, "Lycos: The lyngby co-synthesis system," Design Automation for Embedded Systems, vol. 2, no. 2, pp. 195–235, 1997.

[9] X. Zhao, H. Zhang, Y. Jiang, S. Song, X. Jiao, and M. Gu, "An effective heuristic-based approach for partitioning," Journal of Applied Mathematics, vol. 2013, 2013.

[10] R. Dick and N. Jha, "Mogac: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 10, pp. 920–935, 1998.

[11] J. Hidalgo and J. Lanchares, "Functional partitioning for hardware-software codesign using genetic algorithms," in Proceedings of the 23rd EUROMICRO Conference EUROMICRO. IEEE, pp. 631–638.

[12] P. Eles, Z. Peng, K. Kuchcinski, and A. Doboli, "System level hardware/software partitioning based on simulated annealing and tabu search," Design Automation for Embedded Systems, vol. 2, no. 1, pp. 5–32, 1997.

[13] T. Wiangtong, P. Cheung, and W. Luk, "Comparing three heuristic search methods for functional partitioning in hardware–software codesign," Design Automation for Embedded Systems, vol. 6, no. 4, pp. 425–449, 2002.

[14] P. Knudsen and J. Madsen, "Pace: A dynamic programming algorithm for hardware/software partitioning," in Proceedings of the 4th International Workshop on Hardware/Software Co-Design. IEEE Computer Society, 1996, p. 85.

[15] R. Gupta and G. De Micheli, "Hardware-software cosynthesis for digital systems," IEEE Design and Test of Computers, pp. 29–41, 1993.

[16] R. Niemann and P. Marwedel, "Hardware/software partitioning using integer programming," in Proceedings of the 1996 European conference on Design and Test. IEEE Computer Society, 1996, p. 473.

[17] F. Vahid and D. Gajski, "Clustering for improved system-level functional partitioning," in Proceedings of the 8th international symposium on System synthesis. ACM, 1995, pp. 28–35.

[18] F. Vahid, D. Gajski, and J. Gong, "A binary-constraint search algorithm for minimizing hardware during hardware/software partitioning," in Proceedings of the conference on European design automation. IEEE Computer Society Press, 1994, pp. 214–219.

[19] J. Albuquerque, C. Coelho Jr, C. Cavalcanti, D. Cecilio da Silva Jr, and A. Fernandes, "System-level partitioning with uncertainty," in Hardware/Software Codesign, 1999.(CODES'99) Proceedings of the Seventh International Workshop on. IEEE, 1999, pp. 198–202.

[20] J. Albuquerque, "Solving hw sw partitioning by stochastic linear programming with management of teams uncertainty."

[21] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, Y. Xie, and W.-L. Hung, "Reliability-centric hardware/software co-design," in Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on. IEEE, 2005, pp. 375–380.

[22] Y. Jiang, M. Wang, H. Liu, M. Hosseini, and J. Sun, "Dependable integrated clinical system architecture with runtime verification," in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2017, pp. 951–956.

[23] Y. Jiang, H. Song, Y. Yang, H. Liu, M. Gu, Y. Guan, J. Sun, and L. Sha, "Dependable model-driven development of cps: From stateflow simulation to verified implementation," ACM Transactions on Cyber-Physical Systems, vol. 3, no. 1, p. 12, 2018.

[24] B. Liu, "Uncertainty theory," Uncertainty Theory, pp. 1–79, 2010.

[25] J. Yu, Z. Hehua, J. Xun, S. Xiaoyu, W. N. N. Hung, G. Ming, and S. Jiaguang, "Uncertain model and algorithm for hardware/software partitioning," IEEE Computer Society Annual Symposium on VLSI, 2012, Amherst, MA, USA, August 19-21, 2012, vol. 2013, pp. 243–248.

[26] B. Liu and Y. Liu, "Expected value of fuzzy variable and fuzzy expected value models," IEEE Transactions on Fuzzy Systems, vol. 10, no. 4, pp. 445–450, 2002.

[27] B. Liu, "Fuzzy random dependent-chance programming," IEEE Transactions on Fuzzy Systems, vol. 9, no. 5, pp. 721–726, 2001.

[28] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," IEEE Transactions on Industrial Informatics, 2019.

[29] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: Privacy preserved profile publishing in cyber-physical social systems," IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1868–1878, 2017.

[30] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," IEEE Transactions on Network Science and Engineering, 2018.

[31] Z. Cai and H. Zaobo, "Trading private range counting over big iot data," ICDCS, 2019.

[32] X. J. X. S. Yu Jiang, Hehua Zhang, "Uncertain model and algorithm for hardware/software partitioning," 2012 IEEE Computer Society Annual Symposium on VLSI, vol. 11, no. 4, pp. 243 – 248, 2012.

[33] P. Arató, S. Juhász, Z. Mann, A. Orbán, and D. Papp, "Hardware-software partitioning in embedded system design," in Intelligent Signal Processing, 2003 IEEE International Symposium on. IEEE, 2003, pp. 197–202.

[34] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on. Ieee, 2001, pp. 3–14.

[35] A. Orailoglu and R. Karri, "A design methodology for the high-level synthesis of fault-tolerant asics," in VLSI Signal Processing, V, 1992.,[Workshop on]. IEEE, 1992, pp. 417–426.

[36] Y. Jiang, H. Zhang, X. Song, W. N. Hung, M. Gu, and J. Sun, "Verification and implementation of the protocol standard in train control system," in Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual. IEEE, 2013, pp. 549–558.

[37] H. Song and H. Kong, "Safety-assured formal model-driven design of the multifunction vehicle bus controller," in 21st International Symposium Formal Methods. Springer, 2016, pp. 757–763.

[38] H. Zhang, Y. Jiang, X. Song, W. N. Hung, M. Gu, and J. Sun, "Tsmart-galsblock: A toolkit for modeling, validation, and synthesis of multi-clocked embedded systems," in Proceedings of the 2014 Foundations of Software Engineering. ACM, 2014.