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*

Follow this and additional works at: https://commons.erau.edu/publication

Part of the **Hardware Systems Commons**

Scholarly Commons Citation  

This Article is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in Publications by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu, wolfe309@erau.edu.
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) $\xi_{ij}^h$ denotes the cost of node $i$ in the hardware implementation ways that in hardware or software.
2) $\Phi_{c_{ij}^h}$ is the linear uncertainty distribution of uncertain variables $\xi_{ij}^h$, denoted by $\zeta(a_{c_{ij}^h}, b_{c_{ij}^h})$, where $a_{c_{ij}^h}$, $b_{c_{ij}^h}$ are nonnegative real numbers.
3) $t_{ij}^h$ denotes the execution time of node $i$ in hardware implementation, and $t_{ij}^s$ denotes the execution time of node $i$ in software implementation.
4) $\Phi_{t_{ij}^h}$, $\Phi_{t_{ij}^s}$ are uncertain distributions of uncertain variables $t_{ij}^h$, $t_{ij}^s$, denoted by $\zeta(a_{t_{ij}^h}, b_{t_{ij}^h})$, $\zeta(a_{t_{ij}^s}, b_{t_{ij}^s})$, where $a_{t_{ij}^h}$, $b_{t_{ij}^h}$, $a_{t_{ij}^s}$, $b_{t_{ij}^s}$ 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) $r_{ij}^h$ denotes the reliability of node $i$ in hardware implementation, and $r_{ij}^s$ denotes the reliability of node $i$ in software implementation.

The partitioning problem is to find a bipartition $P$, where $P = (V_h, V_s)$ such that $V_h \cup V_s = V$ and $V_h \cap V_s = \emptyset$.

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 $H(x)$ of the partition $P(x)$ can be formalized as follow:

$$H(x) = \sum_{i=1}^{n} \xi_{ij}^h(1-x_i)$$

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 $T_e(x)$, communication time $T_c(x)$, and the total time metric $T(x)$ can be formalized as:

$$T_e(x) = \sum_{i=1}^{n} t_{ij}^e x_i + t_{ij}^h (1-x_i)$$

$$T_c(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}[(x_i - x_j)^2]$$

$$T(x) = T_e(x) + T_c(x)$$

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 Reliability

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_j and x_i
            Relation[i][j] ← 1
        end
    end
R ← Recursive(n) /* Reliability derivation */;

Recursive(n) /* Definition of the recursive function */:
for i ← 1 to n do
    if Relation[i][m] == 1 then
        R_m ← R_m * (r_m * x_m + r(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_ on the reliability, the partitioning problem can be modeled as:

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

We note that minimizing the value of _H_(_x_) is equivalent to maximizing the value of _n_ \(h_i x_i\). Hence, the solution of the problem _P_0 is equal to that of the problem _P_1:

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

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:

\[
E[f(\mathbf{x}, \xi_1, \xi_2 \cdots \xi_n)] = \int_0^1 f(\mathbf{x}, \Phi_1^{-1}(\alpha), \Phi_2^{-1}(\alpha) \cdots \Phi_n^{-1}(\alpha)) d\alpha
\]

Hence, the objective function of the problem _P_2 can be converted to:

\[
\max \sum_{i=1}^{n} \xi_i x_i \\
\Rightarrow \max \sum_{i=1}^{n} \left(\int_0^1 \Phi_i^{-1}(\alpha) d\alpha\right) x_i
\]

where \(\Phi_i^{-1}(\alpha)\) is equal to \((1 - \alpha)(a_i^h - b_i^h) + \alpha(b_i^h - a_i^h)\).

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, \(h_i^t(x)\) is defined as \(\|h_i(x) + h_i(x)/2\) and \(h_i^c(x)\) 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:

\[
\sum_{i=1}^{n} h_i^t(x)\Phi_i^{-1}(a) - \sum_{i=1}^{n} h_i^c(x)\Phi_i^{-1}(1 - a) \leq h_0(x)
\]

Hence, the uncertain constraint of the problem _P_1 can be converted as follows:

\[
\sum_{i=1}^{n} T_i x_i \leq M' \\
\Rightarrow M' \sum_{i=1}^{n} T_i x_i \leq M' \geq 1 \\
\Rightarrow \sum_{i=1}^{n} (b_i) x_i + \leq a_0
\]

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

Then, the final version of the converted problem is:

\[
P_2: \begin{cases}
\text{maximize} & \sum_{i=1}^{n} \left(\int_0^1 \Phi_i^{-1}(\alpha) d\alpha\right) x_i \\
\text{subject to} & \sum_{i=1}^{n} (b_i - a_i^h)x_i + \\
& \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij}(x_i - x_j)^2 \leq M - \sum_{i=1}^{n} a_i^h \\
& 1 - \text{Recursive}(n) \leq 1 - R_0 \\
& x_i \in \{0, 1\}; i = 1, 2 \cdots n
\end{cases}
\]
constraints. The converted formalization is as follow:

\[
\begin{aligned}
P_3 : & \left\{ \begin{array}{l}
\text{maximize} & \text{Recursive}(n) \\
\text{subject to} & \sum_{i=1}^{n} (b_{i} - a_{i})x_{i} + \\
& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij}(x_{i} - x_{j})^2 \leq M - \sum_{i=1}^{n} a_{i} \\
& \sum_{i=1}^{n} \left( \int_{0}^{1} \Phi_{i}^{-1}(a)d\alpha \right)(1 - x_{i}) \leq N \\
& x_{i} \in [0, 1]; i = 1, 2 \cdots n.
\end{array} \right.
\end{aligned}
\]

**Algorithm 2: Customized ALG2**

/* iteration-block */

while termination conditions do

while number of individuals ≤ number of the generation size do

Select \((g_1, g_2)\) from the current generation;

Perform crossover on \((g_1, g_2)\) to produce two new individuals \((g_1', g_2')\);

/* annealing-crossover */

if \((\max\{\text{fitness}(g_1'), \text{fitness}(g_2')\} \leq \\
\max\{\text{fitness}(g_1), \text{fitness}(g_2)\})\) then

\[\Delta C = \max\{\text{fitness}(g_1), \text{fitness}(g_2)\} - \max\{\text{fitness}(g_1), \text{fitness}(g_2)\};\]

if \(\min\{1, \exp(-\Delta C/T_k)\} \geq \text{random}[1, 0]\) then

| Accept the crossover;
else

| \(g_1' = g_1, g_2' = g_2;\)
end
end

/* annealing-mutation */

Perform mutation on \(g_1\) to produce \(ng_1;\)

if \((\text{fitness}(ng_1) \leq \text{fitness}(g_1'))\) then

\[\Delta C = (\text{fitness}(ng_1) - \text{fitness}(g_1));\]

if \(\min\{1, \exp(-\Delta C/T_k)\} \geq \text{random}[1, 0]\) then

| Accept the mutation;
else

| \(ng_1 = g_1';\)
end
else

| Accept the mutation;
end

Perform step 19-29 on \(g_2;\) to produce \(ng_2;\)

/* individual-selection */

if the highest fitness of the current generation

\[\geq \text{fitness(solution)}\] then

Copy the individual to the solution;
end
end
return solution: \(x[i], i \in [1, n];\)

**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 knapsack 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 \((P_2)\) 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 \((P_3)\) and the deterministic model \((PE)\) without reliability constraint. The results show that the reliability is improved significantly. Finally, we also apply our uncertain model \((P_2)\) 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 \(P_2\) 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:
**Table 1: Test cases description**

<table>
<thead>
<tr>
<th>Name</th>
<th>Node</th>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc32</td>
<td>25</td>
<td>34</td>
<td>32-bit cyclic redundancy check.</td>
</tr>
<tr>
<td>patricia</td>
<td>21</td>
<td>50</td>
<td>Routine to insert values into Patricia tries.</td>
</tr>
<tr>
<td>dijistra</td>
<td>26</td>
<td>71</td>
<td>Computes shortest paths in a graph.</td>
</tr>
<tr>
<td>clustering</td>
<td>156</td>
<td>333</td>
<td>Image segmentation algorithm.</td>
</tr>
<tr>
<td>rc0</td>
<td>329</td>
<td>448</td>
<td>RUC cryptographic algorithm.</td>
</tr>
<tr>
<td>random1</td>
<td>500</td>
<td>1000</td>
<td>Random generated graph</td>
</tr>
<tr>
<td>random2</td>
<td>1000</td>
<td>2000</td>
<td>Random generated graph</td>
</tr>
<tr>
<td>random3</td>
<td>1500</td>
<td>3000</td>
<td>Random generated graph</td>
</tr>
<tr>
<td>random4</td>
<td>2000</td>
<td>4000</td>
<td>Random generated graph</td>
</tr>
<tr>
<td>random5</td>
<td>2500</td>
<td>5000</td>
<td>Random generated graph</td>
</tr>
<tr>
<td>random6</td>
<td>3000</td>
<td>6000</td>
<td>Random generated graph</td>
</tr>
</tbody>
</table>

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

- When the hardware cost is available, $a_i^h = b_i^h = value_1$. Otherwise, $a_i^h$ is generated as uniform random numbers in [0, 100] and $b_i^h$ is set as $a_i^h + \beta_i^h$. $\beta_i^h$ is a constant real number.
- When the execution time is available, $a_i^h = b_i^h = value_2$ and $a_i^h = b_i^h = value_3$. Otherwise, $a_i^h$ is generated as uniform random numbers in [0, 10] and $b_i^h$ is set as $a_i^h + \beta_i^h$. (In most cases, the hardware execution is so fast that the time is usually set as 0). The software execution time $a_i^s$ is generated as uniform random numbers in [0, 100] and $b_i^s$ is set as $a_i^s + \beta_i^s$. $\beta_i^s$ 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 · $\rho$ · max($b_i^s$)]. We can find that the communication time has an expected value of $\rho$ · max($b_i^s$), where $\rho$ is the so-called communication to computation ratio in the general partitioning problem. We conduct our two experiments with two values $\rho = 0.1, 1$. The reliability parameters $r_i^s, r_i^s$ and constraint $R_0$ are initialized in the similar way.

**M** is the execution time constraint. It is generated as uniform random numbers in [$\sum_i^{n} a_i^h$, $\sum_i^{n} b_i^h$]. We test two constraints (strict time constraint and loose real-time constraint) for each partitioning instance. The first $M_1$ is chosen from [$\sum_i^{n} a_i^h$, $\frac{1}{2} \sum_i^{n} b_i^h$], and the second $M_2$ is chosen from [$\frac{1}{2} \sum_i^{n} b_i^h$, $\sum_i^{n} b_i^h$].

Then, we simulate the two algorithms on the benchmarks and the random graphs, for different values of $\rho$, $M$, and $R_0$. For the benchmarks, the values can be initialized as available values as described above. For the random graphs, the values 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 $E[|f(x, \xi)|]$ of $P_2$ and the object function of $PE_1^d$ are denoted by $P_2$ 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 $P_2$ and $PE_1^d$ 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, $ALG_2$ outperforms $ALG_1$. $ALG_2$ can generate smaller values than $ALG_1$. With the increase in the size, the deviation between the two algorithms grows bigger. The values of the $\rho$ 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, $ALG_1$ drops faster than $ALG_2$. But $ALG_2$ can find the near optimal solution faster than $ALG_1$ 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 ($P_2$) and the customized algorithm 2($ALG_2$) produce quality partitions, which are not only compatible to but also more reliable than the existing deterministic model($PE_1^d$) and algorithm 1($ALG_1$).

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
Arbitration function can be implemented in software. 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 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