TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2015; **26**:1039–1049
Published online 11 July 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.2682

RESEARCH ARTICLE

Improving phasor data concentrators reliability for smart grid

Zhi Chen[1], Meikang Qiu[2]*, Yongxin Zhu[3], Xiao Qin[4] and Gang Quan[5]

[1] Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506, USA
[2] Department of Computer Engineering, San Jose State University, San Jose, CA 95192, USA
[3] School of Microelectronics, Shanghai Jiao Tong University, Shanghai 200240, China
[4] Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849, USA
[5] Department of Electrical & Computer Engineering, Florida International University, Miami, FL 33174, USA

ABSTRACT

Phasor data concentrators (PDCs) are the critical components of a *wide area monitoring system* (WAMS) in the smart grid. They are multi-processor devices employed to collect, concentrate, and synchronize phasor data that is from various *phasor measurement units* and from other PDCs. Because of their present significance and potential criticality for WAMS, reliability of PDCs has become a high priority problem. In this paper, we first formally model the reliability problem on PDCs. Then, we propose a novel task scheduling algorithm, *two-phase heuristic task scheduling*, to enhance the reliability of a PDC with real-time constraints of applications. This algorithm seamlessly integrates both tasks scheduling and reliability optimization techniques via appropriately allocate and reallocate tasks to different processors of PDCs. Experimental results show that the proposed algorithm can efficiently guarantee reliability for PDCs with real-time constraints. Copyright © 2013 John Wiley & Sons, Ltd.

***Correspondence**
Meikang Qiu, Department of Computer Engineering, San Jose State University, San Jose, CA 95192, USA.
E-mail: qiumeikang@yahoo.com

Received 26 September 2012; Revised 27 November 2012; Accepted 9 June 2013

1. INTRODUCTION

Continuous application demands coupled with competitive markets are forcing the current power grid systems to approximate their operating limits, while satisfying real-time constraints for applications. To efficiently support ever increasing demands for maintaining reliability of smart grid is a vital but challenging issue [1]. Reliability of a smart grid system relies on the extensive applications of real-time communications, monitoring, and control systems [2]. Generally, phasor techniques are widely adopted in a *wide area monitoring system* (WAMS), the key components in smart grid, to address this challenge.

WAMS is intrinsically an integrated sensor network to guarantee functionalities of the smart grid and complement the grids' *supervisory control and data acquisition* system. The key components in WAMS are *phasor measurement units* (PMU) and *phasor data concentrators* (PDC) [3, 4]. PMUs are fundamental devices to perform real-time monitoring of voltage stability, frequency stability, and dynamic load balance of a power grid. PDCs are multiprocessor computing platforms, such as Multilin P30, employed to collect, concentrate, and synchronize phasor data from various PMUs and from other PDCs [5, 6]. For example, Figure 1 shows the architecture of a typical WAMS system, where PMUs collect data and submit them to a PDC. Then, a phasor data manager is able to obtain these data and analyze them. Finally, the detection and analysis results are sent to a data manager to store as historical information [7, 8]. To facilitate the execution of various tasks with different performance metrics such as time, energy, and chip size, PDCs tend to be designed as heterogeneous devices with multiple *processor elements* (PEs) that have different capacities.

In realistic deployment, however, PDCs are easily prone to failures incurred by internal and external noises, unexpectedly power malfunction, and aging of devices. The failure of a PDC will dramatically impair the connectivity and aggravate the service quality of a WAMS. Therefore, to efficiently monitor and control the power grid in real-time, it is critical to enhance reliability of PDCs. In this paper, we focus on the improvement of reliability of PDCs while ensuring real-time requirements of grid applications through sophisticated task scheduling techniques. The involved tasks are modeled as *directed acyclic graphs* (DAGs) on the basis of their precedence relationships. We mainly consider the failure caused by

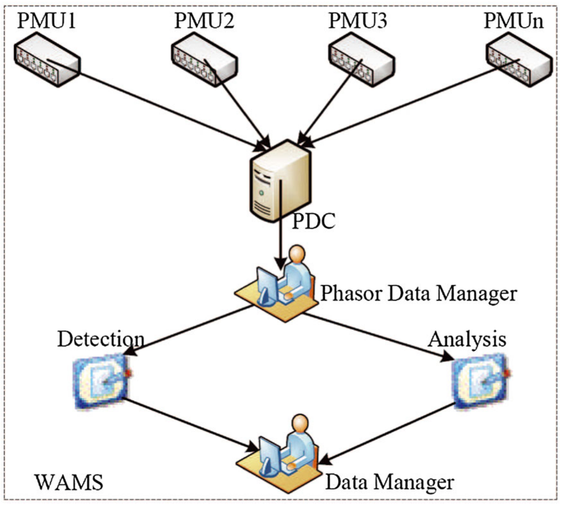


Figure 1. An overview of a typical wide area monitoring system (WAMS) system.

processor cores and communication links between processors. Therefore, the term *processing (PE) element failure* can be referred to as the processor failure or the communication link failure in this work. In addition, because the hardware implementation of reliability strategies lead to low flexibility and inconvenient maintenance, we use a software scheme to improve the reliability of PDCs. For example, our scheduling approach can be implemented by modification of real-time schedulers in operating systems.

Aiming to promote reliability of a PDC with the real-time constraint, we propose a *two-phase heuristic task scheduling* (THETAS) algorithm to seamlessly integrate both task scheduling and reliability optimization. In the first stage, we use a slack time conscious method to appropriately allocate tasks to different PEs. In the second phase, we reschedule the tasks with the aid of a novel matching scheduling strategy to reduce the reliability cost of the target PDC system. We adopt *reliability cost (RC)* as a metric to evaluate reliability of a given system. The lower *RC* a system achieves, the more reliable the system is. The objective of this algorithm is to reduce the RC of a PDC system while satisfying the given real-time constraints.

To the best of our knowledge, this is the first work to address the reliability problem of PDCs for the smart grid through task scheduling techniques. Experimental results across a suite of benchmarks show that the THETAS strategy can reduce the RC for the target system with four PEs by 24.04% compared with a heuristic algorithm, *highest levels first with estimated times* (HLFET), when the time constraint is relax.

The major contributions of this work are twofold.

(1) Targeting the reliability optimization problem, we use probability to formulate it. In this reliability model, we will consider the RC incurred by failures of devices and execution times of tasks for applications in PDCs.
(2) To minimize the RC, we propose a THETAS algorithm to appropriately schedule tasks to different processor cores of PDCs. The first phase is a traditional *as soon as possible* (ASAP) scheduling, which does not take account of RC. In the second phase, we will reschedule the tasks with the aid of a matching scheduling strategy to reduce the RC of a PDC system.

The organization of the remainder of this paper is as follows. Section 2 briefly overviews the related work of reliability optimization problem. Section 3 describes the fundamentals of the reliability optimization models. Section 4 discusses the details of the proposed THETAS algorithm. Experimental results are shown in Section 5, and conclusions are presented in Section 6.

2. RELATED WORK

2.1. Optimization of reliability for smart grid

Reliability of the smart grid is a prioritized task. In [9], Moslehi et al. investigated the impacts of reliability on renewable resources, demand response, electric storage, and electric transportation. On the basis of the review, the authors jointly proposed an architectural framework to facilitate the design, development, and integration of various standards and protocols. Liu et al. [10] explored the commonly used models and methods to analyze reliability. They also discussed how smart grid environment complicated these models. On the basis of condition monitoring of circuit breakers and transformers, Eftekharnejad and Vittal et al. [11] devised a method to improve reliability of smart grid systems. Bouhouras et al. [12] proposed alternative methods of selective automation upgrade in power distribution networks. The current studies on reliability of smart grid primarily focus on the infrastructure research and hardware strength. The utilization of software techniques to enhance reliability of PDCs, however, is still insufficient in the literature.

Focusing on the cyber security of WAMSs in the smart grid, including deliberate attacks and inadvertent operations, Metke and Ekl [13] discussed key security technologies for a smart grid system, such as public key infrastructures and trusted computing. Qiu et al. [14] designed micro-power measurement circuit to measure energy consumptions of various security algorithms. They also proposed algorithms to minimize energy consumption of various security algorithm by using code optimization methods. Some researchers focus on the physical measurement of security algorithm to get their energy characteristics. For example, in [15], Wander et al. measured energy consumption of RAS and ECC algorithms on MICA2DOT nodes.

In this paper, we focus on using task scheduling algorithms to reduce the RC of PDCs while satisfying real-time constraints of applications. Our algorithm can be implemented by modifying the real-time scheduler and tasks management services in other areas, such as embedded operating systems.

2.2. Optimization of reliability for distributed heterogeneous computing systems

Numerous efforts have been invested in optimization of reliability for distributed heterogeneous computing systems. In [16], the authors proposed a matching and scheduling task scheduling algorithm by integrating RC for tasks with precedence constraints. However, this algorithm did not consider real-time characteristics of applications. Liberato et al. [17] presented a feasibility-check algorithm to enhance the fault-tolerant ability for homogeneous single processor systems. This algorithm is not supportable to heterogeneous systems. In [18], an algorithm to improve reliability and fault-tolerance of heterogeneous systems for real-time applications was introduced. Instead of concentrating on distributed computing platforms, we mainly focus on the heterogeneous devices, PDCs, in smart grid.

Dongarra et al. [19] proposed an optimal scheduling strategy for independent unitary tasks to enhance system reliability within the minimal makespan. Sun et al. [20] formally modeled the impacts of failure, checkpointing, and recovery mechanisms on high performance computing systems in separate. In [21], Oh and Son investigated a fault-tolerant scheduling algorithm that statically schedules independent real-time tasks for applications. They also proposed a fault-aware task partition and scheduling algorithm to boost system performance and reduce the application execution time. Aiming to improve the system reliability, the authors in [22, 23] also studied independent real-time tasks.

2.3. Algorithms to minimize the failure of systems

Qiu et al. [24] proposed algorithms to minimize the total failure rate of wireless sensor network while satisfying timing constraints for applications. But [24] did not consider the link failure problem among processors, in this paper, we sufficiently consider both the failure of processing components and the communication links between them. In addition, we model communication links as communication nodes during the task scheduling for smart grid applications.

However, these studies listed previously are either designed for independent tasks or for homogenous systems. We propose a heuristic algorithm, THETAS, to enhance the reliability for heterogeneous multiprocessor PDC systems while satisfying given time constraints. In this algorithm, we sufficiently take into account precedence dependencies among real-time tasks and use DAGs to model their execution sequences. This algorithm contains two phases. In the first phase,we use an *as soon as possible* (ASAP) algorithm and an *as last as possible* (ALAP) algorithm to get an initial scheduling for the tasks profiled from an application. Then, a bipartite matching algorithm is utilized in the second phase to compact the scheduling and further reduce the RC.

Table I. Major notations used in this paper and their definition

Notation	Definition
PE	Processing element
N_P	Number of computation nodes
N_C	Number of communication nodes
M	$N_P + N_C$
DAG	Directed acyclic graph
T_{vi}	Execution time set of task *vi*
$T(i,j)$	Computation time of task *vi* on component *j*
X_{ij}	Indicator of task *i* to PE *j*
λ	Failure rate of a component
Re	Reliability of a target system
RC_{ij}	Reliability cost of allocating task *i* on PE *j*
EST	Earliest start time
LST	Latest start time
$Pre(v_i)$	Predecessor set of task v_i
$succ(v_i)$	Successor set of task v_i
$FT(v_i)$	Finish time of task v_i
$SL(j)$	Schedule length of PE *j*

3. DEFINITIONS AND MODELS

We first give a Table I to indicate the acronyms will be used in this paper.

3.1. System Model

For a heterogeneous multiprocessor PDC, we assume it has N_P computation elements and N_C communication links between these computation elements. We use a PE set $P = \langle P_1, P_2, \ldots, P_M \rangle$ to represent all *processing elements* (PEs) in the target system, where $M = N_P + N_C$. Figure 2(a) shows the architecture of a PDC with four heterogeneous PEs. The edges among these PEs indicate the point-to-point communication links. We regard each communication link to be a communication node (see Figure 2(b)), when there is data transfer between any two PEs, and the weight of the virtual communication node equals to the delay of the communication link. For example, the node C_{01} in Figure 2(b) represents the communication link between PE P_0 and PE P_1. However, for simplicity, we assume that the communication link failure is similar to that of the PE failure, although they may or may not have the same failure rate. Therefore, the notation M in the later sections refers to the total number of PEs

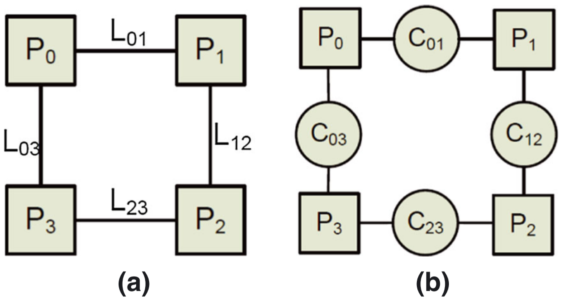


Figure 2. The architecture of phasor data concentrators with four heterogenous processing elements (PEs) P_1, P_2, P_3, and P_4. (a) The communication links between these PEs are indicated. (b) The communication links are viewed as communication nodes among these four PEs.

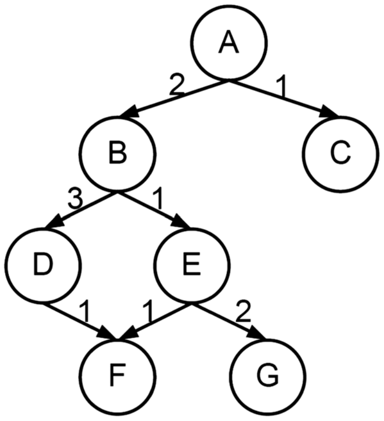


Figure 3. An example of a directed acyclic graph that consists of seven tasks.

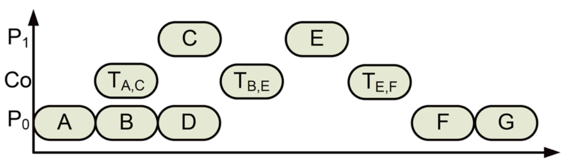


Figure 4. A simple mapping of the tasks on processing elements of the target system. Inter-processor communications are implemented by communication links that are represented as communication components.

and communication links of a target system. For the reliability optimization problem of PDCs, the reliability of each component is not known exactly due to uncertainties of the failure rate for each component. We address this problem by assuming that the failure rates are independent to each other in the system.

3.2. Application Model

To represent the dependencies between different tasks of a smart grid application, a DAG is employed. Specifically, a DAG $G = \langle V, E, T\rangle$ is a directed graph with weights on nodes and edges. In a DAG G, the element $V = \langle v_1, v_2, \cdots, v_N\rangle$ is set of N tasks. $E \subseteq V \times V$ is a set of edges that represent data dependencies among different tasks in the task set V. For instance, an edge $E(u \rightarrow v)$ represents the execution of task v needs to wait for the completion of tasks u, because there is a data flow from task u to task v. $T = \langle T_{v_1}, T_{v_2}, \cdots, T_{v_N}\rangle$ represents an execution time set of tasks in the task set V. Each element T_{v_i} is a set of execution time of task v_i on each PE, which is defined as $T_{v_i} = \langle T_{i1}, T_{i2}, \cdots, T_{iM}\rangle$, where T_{ij} indicates the execution time of task v_i on PE j. Figure 3 shows an example of a DAG consisting of seven tasks.

For example, Figure 4 shows a simple instance of scheduling the tasks in Figure 3 to a target system with two processors and a communication link between these two processors. In this figure, the three nodes in the row of C_O represent that there are three communications between these two processors when the allocation of tasks is as follows: tasks A, B, D, F, and G to processor 0, tasks C and E to processor 1. For simplicity, we only consider the cost caused by inter-communication while ignoring the intra-communication. This is mainly because that the communication overhead and failure rate between two tasks on a same PE are much lower than when they are on two different PEs. Therefore, the weight of an edge in Figure 3 indicates the communication overhead of two tasks when they are allocated to different PEs.

In this paper, we mainly focus on coarse-grain granularity tasks, which means that each task in our model would be a function or basic block obtained from a specific program. There are an array of ways to get the coarse-grain granularity tasks from an application, but the most commonly used method is with the help of profiling tools. We will also use this approach to partition tasks and derive their dependencies, then construct a corresponding DAG. One striking characteristic of coarse-grained tasks is that they have high computation-to-communication ratio. Therefore, the computational overhead is always much higher than that of the communication overhead, so that the later is usually negligible.

3.3. Reliability Model

This paper focuses on maximizing reliability of the target PDC systems. More specifically, we will develop novel task scheduling techniques to improve reliability of target systems and make sure that they can stand with some unpredictable failures. To benefit from stability of different processors in a target system, we reasonably schedule each task on these processors. Essentially, reliability is a probability which indicates how possible a system can process a set of tasks without any failure.

We assume that the target system has M PEs (including processing units and communication nodes) and N

tasks can be profiled from a smart grid application. A random variable λ_i is introduced to represent the failure rate of a component in the system. Also, we define a variable $T(i, j)$ as the computation time of task v_i on component j. Note that, the failure of a PE usually consists of two parts: one is processor failure and another is interconnection link failure. Commonly, the failure of each component conform to a Poisson distribution [16, 18].

Define a binary variable $X = \{X_{ij} | \ \forall i \in 1, 2, \cdots, N, \ j \in 1, 2, \cdots, M\}$ to denote whether task i is mapped on PE j.

$$X_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned PE } j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Define a variable ζ_j to represent the total execution time of PE j for all tasks on it. Therefore, we have the equation $\zeta_j = \sum_{i=1}^{N} X_{ij} T_{ij}$, and the schedule length of all tasks is $\max(\zeta_1, \ \zeta_2, \cdots, \zeta_M)$.

Let a variable A_j represent the failure of PE j and Re_{ij} be the reliability of PE j with allocating task i on it.

$$\begin{aligned} Re_{ij} &= 1 - \int_0^{T_{i,j}} f(A_j) dt \\ &= 1 - \int_0^{T_{ij}} \lambda_j e^{-\lambda_i t} dt = e^{-\lambda_j T_{ij}} \end{aligned} \tag{2}$$

We have the reliability of PE j as the equation $Re_j = \prod_{i=1}^{N} Re_{ij} X_{ij}$. Therefore, the reliability of the target system, *Re*, can be expressed as Equation (3)

$$Re = \prod_{j=1}^{N} Re_j = e^{\sum_{i=1}^{M} -\lambda_j \zeta_j} = e^{-\sum_{i=1}^{N} \sum_{j=1}^{M} \lambda_j X_{ij} T_{ij}} \tag{3}$$

However, the expression of reliability in Equation (3) is nonlinear. We can obtain the maximum of it by using an equivalent linear form as Equation (4) through a logarithmic operation. From Taylor's theorem, we can simplify the computation of *Re* by replacing e^x with its small value approximation, $1+x$. Then, the reliability expression approximates

$$Re = 1 - \sum_{i=1}^{N} \sum_{j=1}^{M} \lambda_j X_{ij} T_{ij} \tag{4}$$

Hence, to maximize the overall system reliability *Re*, we only need to maximize the value of Equation (4). Because *Re* is a decreasing function, we can get the maximal reliability, *Re*, by minimizing the following:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \lambda_j X_{ij} T_{ij} \tag{5}$$

We call this expression *RC* of the target PDC system. We introduce a variable RC_{ij} to denote the RC of allocating task v_i to PE j.

$$RC_{ij} = \lambda_j T_{ij} \tag{6}$$

The overall system *RC* for a PDC is the summation of the reliability cost incurred by allocating tasks to each PE in the system.

$$RC = \sum_{i=1}^{N} \sum_{j=1}^{M} RC_{ij} \tag{7}$$

Therefore, the reliability optimization problem becomes minimizing the RC of a system. In this work, we use RC to indicate how unreliable a given PDC device is when a set of real-time tasks are assigned on it. The lower RC produced by a PDC in the WAMS, the higher reliability of the PDC can be achieved.

4. TWO-PHASE HEURISTIC TASK SCHEDULING

Based on the constructed models, we propose a *THETAS* algorithm to reduce the RC for PDC system in the smart grid. The rationale behind the THETAS strategy is as follows.

Phase 1: We use a *modified ASAP* (MASAP, see Algorithm 4.1) and a *modified ALAP* (MALAP, see Algorithm 4.2) algorithm to schedule a given DAG and get an initial scheduling. The MASAP algorithm, presented in Algorithm 4.1, picks a task T from the task set to calculate its *earliest start time* (EST). Note that the tasks in the task set are obtained from a DAG by using *bread first search*. For each task v_i, the algorithm computes its EST on each processor P_x on the basis of the allocation of its predecessors and execution time of v_i on P_x. Then it selects the processor that is able to execute the task with the earliest time. This scheduling is repeated until all tasks in the task set are allocated. The MALAP is similar to the MALSP algorithm. However, it determines the EST of a task by checking execution time of its successors. By using these two algorithms, we can obtain the EST and the *latest start time* (LST) for each task. Then, based on the EST and LST, we can compute the mobility (slack time), *Mo*, of each task i as: $Mo(v_i) = LST(v_i) - EST(v_i)$. According to the ascending slack time of tasks, we get an initial schedule for an application on PDCs without violating the dependencies among tasks. In this work, we call this initial scheduling as *slack time aware scheduling* (STAS). The scheduling in this phase provides an initialization for further optimization of the RC. The details of initialization are illustrated in Algorithm 4.3.

Phase 2: Based on the initial scheduling, STAS, from Phase 1, we construct a bipartite matching graph by putting all currently schedulable tasks in a ready task set and all PEs in a PE set, as shown in Algorithm 4.4. A tasks can

Algorithm 4.1 Modified as soon as possible scheduling

```
Require: A DAG $G = \langle V, E \rangle$.
Ensure: An initial scheduling of the DAG $G$.
1: $TaskSet \leftarrow V$
2: Schedule $v_0$ by setting $EST(v_0) = 0$;
3: repeat
4:     Select a task $v_i, \{v_i | Pre(v_i) = \emptyset\}$; /* $Pre(v_i)$ is
       the predecessor set of task $v_i$;*/
5:     $EST(v_i) = max(EST(v_j))+$
       $min(T(v_j, P_x)) + weight(e(v_j \leftarrow v_i))$;        /*
       $v_j \in Pre(v_i), P_x \in P$; */
6:     Remove $v_i$ from $TaskSet$;
7: until $TaskSet = \emptyset$
8: $SLtmp \leftarrow EST(v_n)$;
9: return A scheduling of MASAP;
```

Algorithm 4.2 Modified as last as possible scheduling

```
Require: A DAG $G = \langle V, E \rangle$, the time of the lasted task
    in MASAP $SL\_ASAP$.
Ensure: An initial scheduling of the DAG $G$.
1: $TaskSet \leftarrow V$
2: Schedule $v_n$ by setting $LST(v_n) = SL\_ASAP$;
3: repeat
4:     Select a task $v_i, \{v_i | Succ(v_i) = \emptyset\}$; /* $Succ(v_i)$ is
       the successor set of task $v_i$; */
5:     $LST(v_i) = min(LST(v_j))$ -
       $min(T(v_i), P_x) - weight(e(v_j \leftarrow v_i))$;/*
       $v_j \in Succ(v_i), P_x \in P$; */
6:     Remove $v_i$ from $TaskSet$;
7: until $TaskSet = \emptyset$
8: return A scheduling of MALAP;
```

be scheduled only when all its predecessors are complete. Therefore, all the tasks in the ready task set can be scheduled without violating precedence constraints. We reschedule the tasks to different PEs by considering the RC of the task and a PE pair. The objective of this phase is to minimize the RC heuristically at each single step. Algorithm 4.5 describes the pseudo code of the THETAS strategy.

Therefore, the execution of the whole algorithm is as following.

(1) We use MASAP and MALAP algorithms to calculate the earliest and latest start time of each task. Based on the earliest and latest time of each task, we can get the slack time of each task by using the STAS algorithm. Meanwhile, we can obtain an initial scheduling and schedule length for all tasks.
(2) We push the tasks without a predecessor into a task set V_T and push all the PEs in the target system into the PE set V_P. On the basis of these two sets, the Algorithm 4.4 will be called to construct a bipartite match graph.
(3) Repeatedly taking a $< task, processor >$ pair from the task set V_T of the bipartite matching graph until all tasks are removed from this set. Whenever we remove a task from V_T, we need to update the total RC of the system and the schedule length of the processor that will execute this task.
(4) Repeat to execute step 2 until the tasks are visited.

Algorithm 4.3 Slack time aware scheduling

```
Require: A DAG $G = \langle V, E \rangle$, a processor set $P$, and the
    execution times of each task on $P$.
Ensure: An initial scheduling of the DAG $G$.
1: Add two dummy nodes, $v_0$ and $v_n$, as the entry node
   and exit node of the DAG.
2: Get the earliest start time by Algorithm 4.1;
3: Get the latest start time by Algorithm 4.2;
4: for all $v_i \in V$ do
5:     $Mo(v_i) \leftarrow LST(v_i) - EST(v_i)$;
6: end for
7: Schedule the tasks according to the ascending
   order of $Mo(v_i)$ while satisfying their precedence
   relationships;
8: for all $v_i \in V$ do
9:     $FT(v_i) \leftarrow$ the finish time of task $v_i$;
10: end for
11: for all $P_j \in P$ do
12:     $SL(P_j) \leftarrow 0$;
13: end for
14: $RC \leftarrow 0$; /*$RC$ is the total reliability cost*/
15: return The initial scheduling;
```

Algorithm 4.4 Build bipartite matching graph (BBMG)

```
Require: A DAG $G = \langle V, E \rangle$, a PE set $V_P$, and a task set
    $V_T$.
Ensure: A task and processor matching graph $GB =$
    $\langle VB, EB \rangle$.
1: for all $v_i \in V_T$ do
2:     for all $P_j \in V_P$ do
3:         $TSL_j \leftarrow SL(P_j)$; /*Get the schedule length of
           processor $(P_j)$*/
4:         Calculate $RC_{ij}$ by using Equation (6);
5:         Add an edge in $EB$ by connecting from task $v_i$
           to processor $P_j$;
6:         if $max(TSL_j, EST(v_i)) + T(i, j) \leqslant FT(v_i)$
           then
7:             $W(EB(v_i, P_j)) \leftarrow RC_{ij}$; /*set the weight of
               the edge to be equal to the reliability cost*/
8:         else
9:             $W(EB(v_i, P_j)) \leftarrow \infty$;
10:        end if
11:    end for
12: end for
13: $VB = V_T \cup V_P$;
14: return $GB$;
```

Figure 6 illustrates the procedures of our THETAS algorithm for the example shown in Figure 3. For the

Algorithm 4.5 Two-phase heuristic task scheduling algorithm

```
Require: A DAG $G = \langle V, E\rangle$, a task set $V_T$, a PE set $V_P$,
    and a time constraint $TC$.
Ensure: A schedule with the overall reliability cost $RC$.
 1: Apply Algorithm 4.3 to get an initial scheduling
 2: repeat
 3:   Insert all schedulable tasks in $V_T$ and all PEs in $V_P$;
 4:   /* Build a matching graph $GB = \langle VB, EB\rangle$ by
      using Algorithm 4.4;*/
 5:   $GB \leftarrow BBMG(G, V_T, V_P)$
 6:   repeat
 7:     $e(v_i, P_j) \leftarrow min\{W(EB)\}$;
 8:     $TSL \leftarrow max(EST(v_i), SL(P_j))+$
        $T(i, P(v_i))$;
 9:     if $TSL \geqslant FT(v_i)$ then
10:       $W(EB(v_i, P_j)) \leftarrow \infty$;
11:     end if
12:     Set $v_i$ as visited and delete it from $V_T$;
13:     $v_i \rightarrow P_j$; /*allocate task $i$ to processor $j$;*/
14:     $RC \leftarrow W(EB(v_i, P_j)) + RC$; /*Increase the
        total reliability cost;*/
15:     $SL(P(v_i)) \leftarrow TSL$;
16:     for all $v_m \in Succ(v_i)$ do
17:       if $EST(v_m) < SL(P(v_i))$ then
18:         $EST(v_m) \leftarrow SL(P(v_i))$;
19:       end if
20:     end for
21:   until $V_T = \emptyset$
22: until All tasks are "visited"
23: return $RC$;
```

demonstration purpose, we assume: (1) there are two PEs, namely P_0 and P_1, in the target PDC; (2) the failure rates of P_0, P_1, and the communication link are 0.5, 0.2, and 0.3, respectively. In the real systems, failure rate can be obtained through observing historical data of a target system, provided by commercial sources or through tests; (3) the execution time of tasks A, B, C, D, E, F, and G on P_0 and P_1 are (3, 3), (1, 3), (4, 2), (5, 3), (1, 3), (2, 2), and (3, 4), respectively; and (4) the communication costs among tasks are shown on the edges of Figure 3.

Based on the slack time aware scheduling, we can get an initial allocation shown in Figure 5. From this figure,

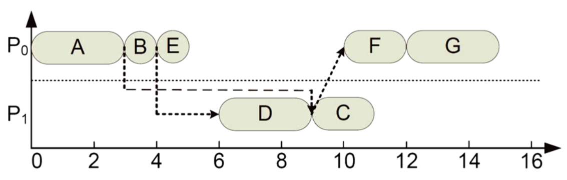


Figure 5. Applying the slack time conscious scheduling algorithm on the directed acyclic graph shown in Figure 3. In this case, the reliability cost is $7.5 = 5 * 0.2 + 10 * 0.5 + 5 * 0.3$.

we can see that only tasks C and D are allocated to PE P_1, whereas the others are allocated to PE P_0. After the initialization, the finish times of each task are determined. On the basis of this scheduling, we can compute the failure of each task by allocating them on different processors. For example, the weight on the edge (A, P_0) is obtained by using $3 * 0.5 = 1.5$, and the weight on the edge (B, P_0) is calculated by using $1 * 0.5 + 0.3 = 0.8$. The reason for the infinite weight of edge (B, P_1) is because task B has already scheduled on P_0 at step (b). Then, we perform the phase 2 scheduling to compact the schedule and reduce the RC for the initial scheduling. Therefore, with the help of the these two phases, we can get a more reliable schedule for the target PDC systems. Figure 6 illustrates the processes to reschedule the initial schedule shown in Figure 5.

At the very beginning, task A is the only schedulable task. On the basis of the Algorithm 4.4, we can build the matching graph $GB = \langle VB, EB\rangle$ as Figure 6(a), where the left is the task set V_T and the right is the PE set V_P. For example, the edge $e(A \rightarrow P_1)$ indicates the RC of allocating task A to PE P_1 is 0.6. The calculation of the weight of each edge for the matching graph is given by the lines 6–10 of Algorithm 4.4. For each edge $e(v_x, P_y) \subseteq EB$, if the sum of the earliest available time of PE P_y and the computation time of task v_x on P_y is less than the finish time of task v_x in the SATS scheduling, we set the weight of $e(v_x, P_y)$ to be equal to the RC of task v_x on PE P_y ($RC_{x,y}$). Otherwise, the weight of the edge $e(v_x, P_y)$ is set to be infinity.

From Figure 6(a), we can observe that the cost of allocating task A to P_0 and P_1 are 1.5 and 0.6, respectively. Obviously, the pair of (A, P_1) incurs lower RC, therefore we reschedule task A to P_1. After the scheduling, we set

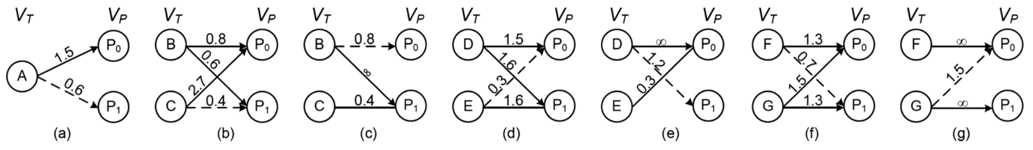


Figure 6. The procedures of rescheduling the initial task scheduling for the directed acyclic graph given in Figure 3. The whole rescheduling process consists of seven steps. (a) Schedule task A to P_1; (b) schedule task C to P_1; (c) schedule task B to P_0; (d) schedule task E to P_0; (e) schedule task D to P_1; (f) schedule task F to P_1; and (g) schedule task G to P_0. The dotted lines represent the allocation of tasks to the corresponding PEs. The solid lines without arrows indicate the matched task and PE pairs. The solid lines with arrows show the possible scheduling schemes. The datum on the lines represent the RC of this allocation.

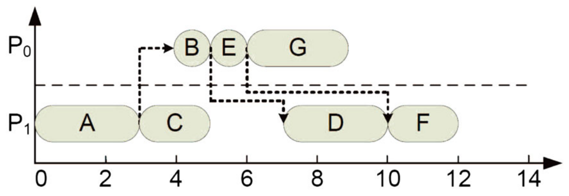


Figure 7. Applying the slack time conscious scheduling algorithm on the directed acyclic graph shown in Figure 3. The reliability cost is 6.3 in this case.

task A to be *visited* and updated the EST of its direct successors (tasks B and C). For example, the EST of task B on P_0 and P_1 are 4 and 3, respectively.

By the completion of task A, the numbers of unscheduled predecessors of tasks B and C are both 0. Consequently, we can schedule tasks B and C and put them in to task set V_T. Similarly, we build the matching graph for these two tasks as Figure 6(b).

After scheduling task A, tasks B and C are schedulable. Therefore, they can be put into the ready task set V_T. By repeating the same approach in the previous text, we build a bipartite matching graph for task B and task C as Figure 6(b). First, based on the constructed bipartite matching graph, it is convenient to know that the edge $e(C, P_1)$ has the minimal RC (which is 0.4). Therefore, we schedule task C to P_1, mark it as $visited$, and update the schedule length of P_1 to 4 time units. In the next scheduling, although the edge $e(B, P_1)$ has the minimal RC, if we allocate task B to P_1, the finish time of task B on P_1 is 8, which exceeds its ALAP finish time ($LST = 6$). Hence, we must schedule task B to P_0 to guarantee the valid execution.

By the completion of this step, tasks D, E, and G are ready to execute. We repeat the previous procedures until all tasks are marked as visited. Rescheduling of the remaining tasks are shown in Figures 6(c) and 6(d). The final scheduling result of our THETAS algorithm for the DAG shown in Figure 3 is shown in Figure 7. We can calculate that the RC of this scheduling is 6.3. Compared with the initial scheduling, the RC is reduced by 16%. which approximates the optimal solution of 7.5 (Schedule tasks A, B, D, E, and F to P_1, and schedule tasks C and G to P_0). Therefore, this algorithm is efficient in reducing the RC for PDC systems.

5. EXPERIMENTS AND RESULTS

This section illustrates the simulations results of our proposed algorithms, by comparing with a heuristic algorithm, *highest levels first with estimated times* (HLFET) [25]. In the HLFET algorithm, the priority of each tasks is determined by a defined ‘level’, which is the maximum sum of the execution time of all tasks along the path from a task to an exit task. The exit task is the one that does not have outgoing edges in a DAG.

To explore the reflexibility of the proposed algorithms, a *time constraint* (TC) is defined to bound the total execution time of an application. Therefore, to guarantee the successful execution of an application, all tasks should be finished in this bound.

$$\min T(i) = \min\{T(i,1), T(i,2), \cdots, T(i,M)\} \tag{8}$$

$$TTC = \frac{\sum_{i=1}^{N} \min T(i)}{M} \tag{9}$$

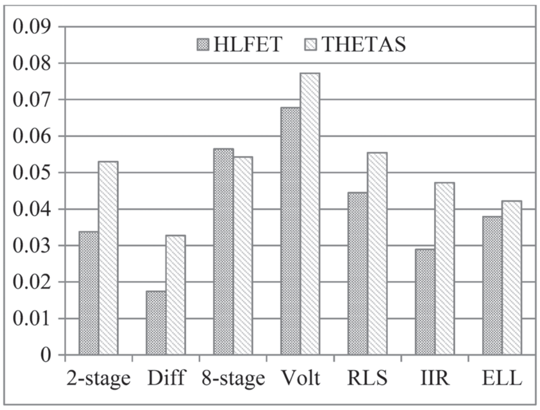
where $\min\{T(i,1), T(i,2), \cdots, T(i,M)\}$ is the minimal execution time of a task on M processors.

Equation (8) donates a tightest time constraints of an application, which is almost impossible to satisfy unless all the tasks are perfectly scheduled. For example, we have to miss a number of tasks with this time constraints for most of cases, which might result in serious consequences including power outage. We make this time constraint adjustable by introducing a relaxed coefficient β: $TC_{new} = \beta \times TC$. Therefore, we can get a more relaxed or tighter time constraint via modifying the relaxed coefficient from 1.3 to 1.8.

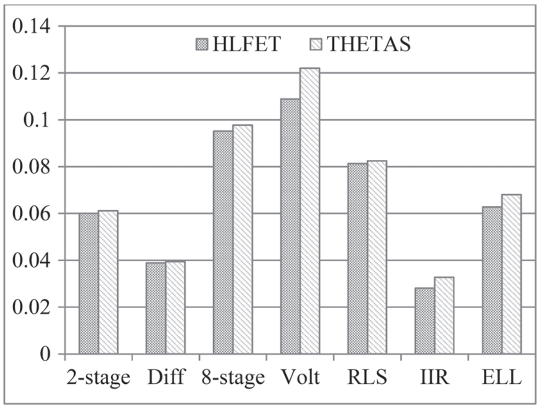
In our experiments, we conduct simulations on the heterogeneous systems with two PEs and four PEs, respectively. Without loss of generality, we also assume that the failure rates of all communication and processing components are conformed to a Possion Law. A variety of benchmarks are selected to simulate and verify the effectiveness of our THETAS algorithm by comparing it with the HLFET algorithm. The benchmarks include two-stage/eight-stage lattice filter, the differential equation solver, the voltera filter, RLS-Laguerre, infinite impulse response filter, and elliptic filter, which are shown as two-state, eight-stage, Diff, Volt, RLS, IIR, and ELL in the experiments, respectively. Both the THETAS algorithm and the HLFET algorithm are implemented as stand-alone programs.

Through adjusting the relax coefficient, the effects of different time constraints on RC of these two algorithms are explored. In Figure 8, the results are obtained from a heterogeneous system with two PEs by applying the THETAS and HLFET algorithm to schedule the tasks. Figures 8(a) and 8(b) depict the RCs of the target system when $\beta = 1.3$ and $\beta = 1.5$, respectively. We can observe from these two figures that the HLFET algorithm outstrips the THETAS algorithm when the time constraints very tight. The primary reason for this phenomena is that most of tasks cannot be schedule in this case by using HLFET algorithm. Therefore, there will be much fewer tasks executed on the target system. That is, less RC will be incurred.

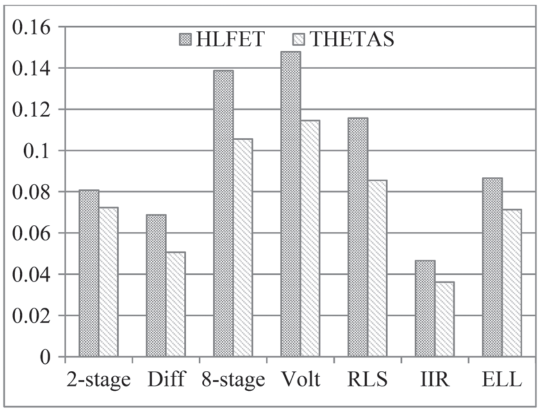
However, the THETAS algorithm will exhibit its advantages when we make the relaxed coefficient more relax. For example, we can see from Figure 8(c) RC caused by the THETAS algorithm will be much less than that of the HLFET algorithm when the relaxed coefficient is 1.8.



(a) $\beta = 1.3$

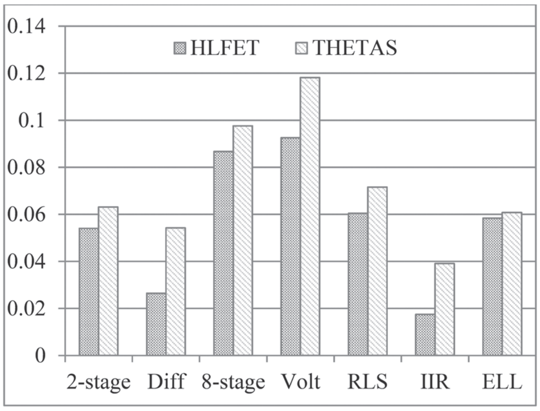


(b) $\beta = 1.5$

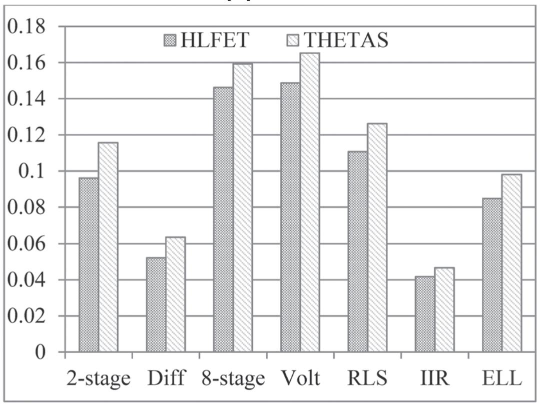


(c) $\beta = 1.8$

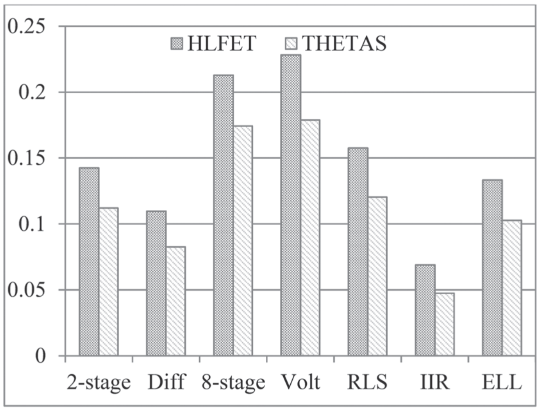
Figure 8. The comparison of reliability cost for our two-phase heuristic task scheduling (THETAS) algorithm to the highest levels first with estimated times (HLFET) algorithm, when the relax coefficient satisfies (a) $\beta = 1.3$, (b) $\beta = 1.5$, and (c) $\beta = 1.8$. In these experiments, the target system has two PEs.



(a) $\beta = 1.3$



(b) $\beta = 1.5$



(c) $\beta = 1.8$

Figure 9. The comparison of reliability cost for our two-phase heuristic task scheduling (THETAS) algorithm to the highest levels first with estimated times (HLFET) algorithm, when the relaxed coefficient satisfies (a) $\beta = 1.3$, (b) $\beta = 1.5$, and (c) $\beta = 1.8$. In these experiments, the target system has four PEs.

This is mainly because the THETAS algorithm can more wisely schedule the tasks in an application to different PEs. Specifically, the THETAS algorithm is able to cut the RC by 21.68% in this case compared to the HLFET algorithm.

The reliability of the target system with four PEs according to a variety of time constraints are shown in Figure 9. In this case, the RC regarding different time constraints exhibits a lot of similarity to that of presented in system having two PEs. For example, the RC can be reduced by 24.04% when the THETAS algorithm is applied on the system when the time constraints is relax (say $\beta = 1.8$), compared with that of the HLFET algorithm. A prominent disparity between these two scenarios with the same time constraint is that the RC incurred by the system with four PEs exceed the RC caused by the system with two PEs. This is mainly because that more communication between tasks on different PEs will be needed when the same number of tasks distributed on more PEs, which usually leads to higher RC. The reduction of the RC will contribute to the enhancement of reliability and stability of PDCs in smart grid systems.

6. CONCLUSION

Phasor technologies such as PDCs have been developed to take real-time monitoring missions of the smart grid. PDCs are multiprocessor devices, whose reliability significantly affects real-time data collection, concentration, and synchronization of the power grid. In this paper, we proposed a THETAS algorithm to seamlessly integrate both task scheduling and reliability optimization of for heterogeneous PDCs. Experimental results showed that the THETAS algorithm can prominently reduce the RC of PDCs.

For our future works, first, we will consider more types of failure and implement these algorithms on a real PDC system to further verify its effectiveness. Second, more accurate models for the communication links will be investigated and used to reinforce the practice of this work. Third, besides the reliability of PDC systems, the security problem of them are also critical, hence this will be one of our future works.

ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation (NSF) CNS-1249223; the National High Technology Research and Development Program of China (863 Program) 2009AA012201; Shanghai International Science and Technology Collaboration Program 13430710400; and NSF CNS-0915762 (CSR), CCF-08452578 (CAREER), CNS-0917137 (CSR).

REFERENCES

1. Liu C-W, Lin T-C, Yu C-S, Yang J-Z. A Fault Location Technique for Two-Terminal Multisection Compound Transmission Lines Using Synchronized Phasor Measurements. *IEEE transaction on smart grid* 2012; **3**(1): 113–121.
2. Reliability considerations from integration of smart grid, Dec. 2010. http://www.nerc.com/files.
3. Ree J, Centeno V, Thorp J, Phadke A. Synchronized phasor measurement applications in power systems. *IEEE Transactions On Smart Grid* 2010; **1**(1): 20–27.
4. Zhang P, Li F, Bhatt N. Next-generation monitoring, analysis, and control for the future smart control center. *IEEE Transactions On Smart Grid* 2010; **1**(2): 186–192.
5. Akke M, Karlsson D. Phasor measurement applications in scandinavia, In *TDC '02*, Yokohama, Japan, 2002; 480–484.
6. Ming Y. Phasor measurement applications in China, In *TDC '02*, Yokohama, Japan, 2002; 485–489.
7. Zhang Y, Markham P, Xia T, Chen L, Ye Y, Wu Z. et al. Wide-area frequency monitoring network (FNET) architecture and applications. *IEEE Transcations on Smart Grid* 2010; **1**(2): 159–167.
8. Armenia A, Chow J. A flexible phasor data concentrator design leveraging existing software technologies. *IEEE Transcations on Smart Grid* 2010; **1**(1): 73–80.
9. Moslehi K, Kumar R. A reliability perspective of the smart grid. *IEEE Transactions On Smart Grid* 2010; **1**(1): 57–64.
10. Liu Y, Ning P, Reiter MK. False data injection attacks against state estimation in electric power grids, In *CCS '09*, Chicago, Illinois, USA, 2009; 21–32.
11. Eftekharnejad S, Heydt GT, Vittal V. Implications of smart grid technology on transmission system reliability, In *PSCE '11*, Phoenix, AZ, USA, 2011; 1–8.
12. Bouhouras AS, Andreou GT, Labridis DP, Bakirtzis AG. Selective automation upgrade in distribution networks towards a smarter grid. *IEEE Transcations on Smart Grid* 2010; **1**(3): 278–285.
13. Metke AR, Ekl RL. Security technology for smart grid networks. *IEEE Transcations on Smart Grid* 2010; **1**(1): 99–107.
14. Qiu M, Gao W, Chen M, Niu J, Zhang L. Energy efficient security algorithm for power grid wide area monitoring system. *IEEE Transactions On Smart Grid* 2011; **2**(4): 715–723.
15. Wander AS, Gura N, Eberle H, Gupta V, Shantz SC. Energy analysis of public-key cryptography for wireless sensor networks, In *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Hawaii, 2005; 324–328.
16. Dogan A, *Ö*zg*ü*ner F. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. *IEEE Transactions on Parallel Distributed Systems* 2002; **13**(3): 308–323.
17. Liberato F, Melhem R, Moss D. Tolerance to multiple transient faults for aperiodic tasks in hard real-time systems. *IEEE Transcations on Computers* 2000; **49**(9): 906–914.
18. Qin X, Jiang H, Swanson DR. An efficient fault-tolerant scheduling algorithms for real-time tasks with precedence constraints in heterogeneous systems, In *ICPP '02*, Vancouver, Canada, 2002Aug.; 360–368.
19. Dongarra J, Jeannot E, Saule E, Shi Z. Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems, In *SPAA '07*, San Diego, CA, USA, 2007; 280–288.

20. Wu M, Sun XH, Jin H. Performance under failures of high-end computing, In *SC '07*, Reno, NV, USA, 2007; 48:1–48:11.
21. Oh Y, Son SH. Scheduling real-time tasks for dependability. *Journal of Operational Research Society* 1997; **48**(6): 238–251.
22. Palencia JC, Gonzalez HM. Schedulability analysis for tasks with static and dynamic offsets, In *RTSS '98*, Madrid, Spain, 1998; 26–37.
23. Thomadakis ME, Liu J-C. On the efficient scheduling of non-periodic tasks in hard real-time systems, In *RTSS '99*, Phoenix, AZ, USA, 1999; 148–151.
24. Qiu M, Deng J, Sha EH-M. Failure rate minimization with multi-fu scheduling for heterogeneous wsn, In *GlobeCom*, New Orleans, LA, USA, 2008; 5213–5217.
25. Adam TL, Chandy KM, Dickson JR. A comparison of list schedules for parallel processing systems. *Communications of the ACM* 1974; **17**(12): 685–690.