Optimal Scheduling of Precedence-constrained Task Graphs on Heterogeneous Distributed Systems with Shared Buses

Sanjit Kumar Roy[1], Sayani Sinha[2], Kankana Maji[2], Rajesh Devaraj[1], and Arnab Sarkar[1]

IEEE ISORC 2019

May 9, 2019

[1]Indian Institute of Technology Guwahati
[2]Jadavpur University

Table of Contents

Introduction

- Applications in many time-critical cyber-physical systems are often represented as *Precedence-constrained Task Graphs* (PTGs)
- There is an increasing trend towards their implementation on distributed heterogeneous platforms
  - consisting of heterogeneous processing elements
  - shared buses (CAN, LIN, FlexRay etc.) [1]
- On a distributed platform consisting of heterogeneous processing and communication resources,
  - execution of a task may require different amounts of time on different processing elements.
  - transmission of a message may require different amounts of time on different communication resources

Introduction Contd.

Given a PTG representing a real-time application and a heterogeneous platform, successful execution/transmission of the task/message nodes while satisfying all timing, precedence and resource related specifications, is ultimately a *scheduling* problem

- Scheduler design schemes for PTGs can be broadly classified as *static* (offline) and *dynamic* (online) [2]
- In safety-critical systems such as automotive/avionic systems [3], it is often advisable that all timing requirements be guaranteed off-line, before putting the system in operation [4]
- Hence, static off-line scheduling schemes are preferred in such systems to provide a high degree of timing predictability [5]

Introduction Contd.

- Most existing real-time static scheduling approaches for PTGs are *list scheduling* based heuristic schemes [2, 6, 7]
- A majority of them attempt to minimize the overall schedule length (*makespan* minimization)
- Such an objective allows maximization of the spare computation bandwidth in the system, which may be used to perform other useful activities
- Many of them assume that the underlying execution platform consists of a fully connected system of processing elements
- There exists a significant class of cyber-physical systems with bus based shared communication links among processors

Introduction: Heuristic vs Optimal

- Heuristic schedules
  - typically based on the satisfaction of a set of sufficiency conditions
  - cannot take into consideration all necessary schedulability requirements
  - schedules are sub-optimal in nature
- Optimal solutions
  - can make a fundamental difference in resource-constrained time-critical systems with respect to performance, reliability and other non-functional metrics like cost, power, space etc
  - Optimal schedules can act as benchmarks allowing accurate comparison and evaluation of heuristic solutions [8]

We design an *Integer Linear Programming (ILP)* based static optimal real-time scheduling strategy for PTGs executing on a distributed platform consisting of heterogeneous processing nodes and inter-connected through a set of heterogeneous shared buses

Platform Model

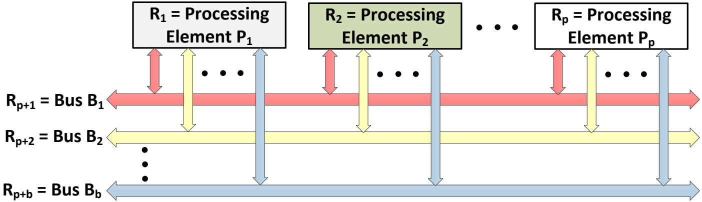
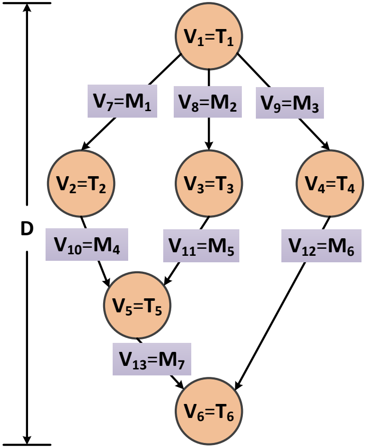


Figure: Platform Model

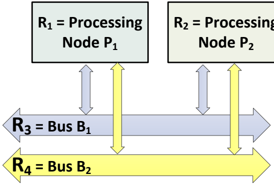
A set of resources $\{R_1, R_2, \ldots, R_{p+b}\}$ among which,

- $\{R_1, R_2, \ldots, R_p\}$ denote a set $P = \{P_1, P_2, \ldots, P_p\}$ of p heterogeneous processing elements
- $\{R_{p+1}, R_{p+2}, \ldots, R_{p+b}\}$ denote a set $B = \{B_1, B_2, \ldots, B_b\}$ of b heterogeneous shared buses
- Each processing node P_i is connected to all b buses

Computation Model



(a)



(b)

ID	P_1	P_2
T_1	12	7
T_2	6	11
T_3	8	5
T_4	14	10
T_5	12	14
T_6	6	15

(c)

ID	B_1	B_2
M_1	8	4
M_2	4	5
M_3	3	3
M_4	6	9
M_5	4	3
M_6	9	7
M_7	7	10

(d)

Figure: (a) PTG G, (b) Platform Model ρ, (c) Computation-time Matrix (CT) and (d) Communication-time Matrix (CM).

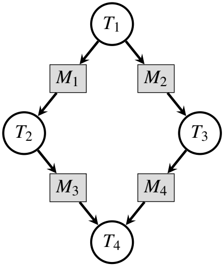
Computation Model Contd.

A *Precedence-constrained Task Graph* (PTG) G is described by a quadruple $G = (V, E, CT, CM)$ where,

- $V = \{V_1, V_2, \ldots, V_{n+m}\}$ represents a set of nodes
- $\{V_1, V_2, \ldots, V_n\}$ represent a set $T = \{T_1, T_2, \ldots, T_n\}$ of n task nodes
- $\{V_{n+1}, V_{n+2}, \ldots, V_{n+m}\}$ denote a set $M = \{M_1, M_2, \ldots, M_m\}$ of m message nodes
- $E \subseteq V \times V$ is a set of edges that describe the *precedence-constraints* among nodes in V.
- CT is a $n \times p$ *computation-time matrix*
- CM is a $m \times b$ *communication-time matrix*

Assumptions

- Single source node T_1
- Single sink node T_n
- Both source (T_1) and sink (T_n) nodes are tasks.
- Each task node T_i is preceded/succeeded by one or more message nodes.
- Each message node M_k is preceded/succeeded by a single task node.
- The communication time for M_k is negligible if both preceding and succeeding task nodes are mapped to same processing element.



Problem Formulation

Given a PTG $G = (V, E, CT, CM)$ with end-to-end deadline D, p processing elements and b buses, find:

- A task node assignment $V_i \mapsto R_j$; $1 \leq i \leq n$ and $1 \leq j \leq p$
- A message node assignment $V_i \mapsto R_j$; $n+1 \leq i \leq n+m$ and $p+1 \leq j \leq p+b$
  - If both the preceding and succeeding task nodes of message node M_i are mapped to the same processing element then, $V_i \rightarrow \emptyset$
- A start time for each task node and message node, such that
  - length of the total schedule is minimized and
  - meets the deadline D

Earliest/Latest Start Times for PTG Nodes

Let, t_i^s and t_i^l be the ASAP and ALAP time of node V_i, respectively

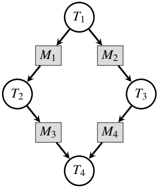


Figure: PTG with message nodes

- **ASAP time computation of task nodes:**
  - Ignore message nodes in the PTG
  - Set ASAP time of the source task node, $t_1^s = 1$
  - Compute ASAP times of the remaining task nodes recursively (downward) as follows:

$$t_i^s = \max_{T_j \in pred(T_i)} (t_j^s + \min_{r \in [1,p]} CT_{jr})$$

where, $pred(T_i)$ is the set of immediate predecessors of task node T_i

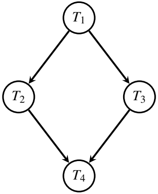


Figure: PTG without message nodes

Earliest/Latest Start Times for PTG Nodes

- **ALAP time computation of task nodes:**
  - Ignore message nodes in the PTG
  - Set ALAP time for the sink task node as,

$$t_n^l = D - \min_{r \in [1,p]} CT_{nr}$$

  - Compute ALAP times of the remaining task nodes recursively (upward) as follows:

$$t_i^l = \min_{T_j \in succ(T_i)} (t_j^l - \min_{r \in [1,p]} CT_{ir})$$

where, $succ(T_i)$ is the set of immediate successors of task node T_i

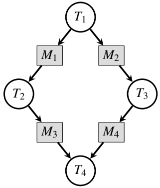


Figure: PTG with message nodes

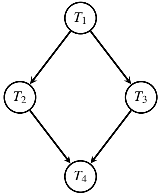


Figure: PTG without message nodes

Earliest/Latest Start Times for PTG Nodes

- **ASAP/ALAP computation procedure for message nodes:**
  - ASAP time of a message node M_k is,

$$t^s_{n+k} = t^s_i + \min_{r \in [1,p]} CT_{ir}$$

where, T_i is the predecessor task node of M_k

  - ALAP time of a message node M_k is,

$$t^l_{n+k} = t^l_j - \min_{r \in [1,b]} CM_{kr}$$

where, T_j is the successor task node of M_k

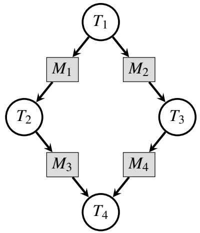


Figure: PTG with message nodes

ILP Formulation: ILP1

We define binary decision variable,

$$X_{irt} = \begin{cases} 1 & \text{if node } i \text{ starts its execution/transmission} \\ & \text{on } r^{th} \text{ resource at time step } t \\ 0 & \text{Otherwise} \end{cases}$$

where, $i = 1, 2, \ldots, n+m;\ r = 1, 2, \ldots, p+b;\ t = 1, 2, \ldots, D$

ILP1

Unique Start Time Constraints:

Start time of each task node should be unique,

$$\forall i \in [1, n] \quad \sum_{r=1}^{p} \sum_{t=t_i^s}^{t_i^l} X_{irt} = 1 \qquad (1)$$

Start time of each message node should be unique,

$$\forall M_k |\ T_i = pred(M_k) \text{ and } T_j = succ(M_k),$$

$$\sum_{r=p+1}^{p+b} \sum_{t=t_{k'}^s}^{t_{k'}^l} X_{k'rt} = 1 - Y_k \qquad (2)$$

where,

$$k' = n + k \text{ and } Y_k = \sum_{r=1}^{p} \sum_{t_1=t_i^s}^{t_i^l} \sum_{t_2=t_j^s}^{t_j^l} X_{irt_1} * X_{jrt_2}$$

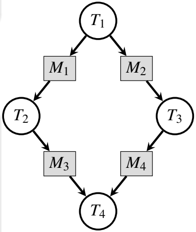


Figure: PTG

ILP1

We introduce another binary decision variable $U_{krt_1t_2}$ $(= X_{irt_1} * X_{jrt_2})$ to linearize the non-linear term,

$$Y_k = \sum_{r=1}^{p} \sum_{t_1=t_i^s}^{t_i^l} \sum_{t_2=t_j^s}^{t_j^l} U_{krt_1t_2} \quad (3)$$

Now, the non-linear variables $U_{krt_1t_2}$ can be linearized using the following three inequalities,

$$X_{irt_1} \geqslant U_{krt_1t_2} \quad (4)$$

$$X_{jrt_2} \geqslant U_{krt_1t_2} \quad (5)$$

$$U_{krt_1t_2} \geqslant X_{irt_1} + X_{jrt_2} - 1 \quad (6)$$

ILP1

Resource Constraints:

A resource can execute at most one task/message node at a given time.

For processing element:

$$\forall t \in [1, D] \text{ and } \forall r \in [1, p] \quad \sum_{i=1}^{n} \sum_{t'=\psi}^{t} X_{irt'} \leqslant 1 \tag{7}$$

where, $\psi = t - CT_{ir} + 1$.

For bus element:

$$\forall t \in [1, D] \text{ and } \forall r \in [1, b] \sum_{i=1}^{m} \sum_{t'=\psi}^{t} X_{i'r't'} \leqslant 1 \tag{8}$$

where, $i' = i + n$, $r' = r + p$ and $\psi = t - CM_{ir} + 1$.

ILP1

Dependency Constraints:

Dependencies between nodes must be satisfied,

$$\forall M_k |\ T_i = pred(M_k) \text{ and } T_j = succ(M_k),$$

$$\sum_{r=1}^{p} \sum_{t=t_i^s}^{t_i^l} (t + CT_{ir}) * X_{irt} \leqslant \sum_{r=p+1}^{p+b} \sum_{t=t_{k'}^s}^{t_{k'}^l} t * X_{k'rt} + \sum_{r=1}^{p} \sum_{t=t_j^s}^{t_j^l} t * X_{jrt} * Y_k \quad (9)$$

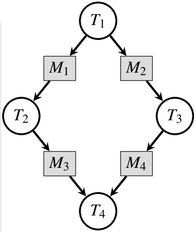


Figure: PTG

where, $k' = n + k$.

ILP1

Dependency Constraints Contd.

We, replace the non-linear term $Y_k * X_{jrt}$ by Z_{krt} and linearize by,

$$Z_{krt} \leqslant X_{jrt} \tag{10}$$

$$Z_{krt} \leqslant Y_k \tag{11}$$

$$Z_{krt} \geqslant Y_k + X_{jrt} - 1 \tag{12}$$

$\forall M_k | \; T_j = succ(M_k)$,

$$\sum_{r=p+1}^{p+b} \sum_{t=t^s_{k'}}^{t^l_{k'}} (t + CM_{kr}) * X_{k'rt} \leqslant \sum_{r=1}^{p} \sum_{t=t^s_j}^{t^l_j} t * X_{jrt} \tag{13}$$

where, $k' = n + k$.

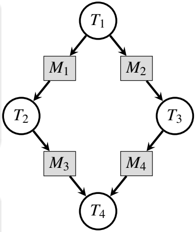


Figure: PTG

ILP1

Objective function: Minimize schedule length of the PTG.

$$Minimize \sum_{r=1}^{p} \sum_{t=t_n^s}^{t_n^l} X_{nrt}(t + CT_{nr}) \tag{14}$$

subject to constraints presented in equations 1 - 13.

ILP2

Linearization in equations 10 to 12 may be avoided by replacing equation 9 with the following two equations.

$$\forall M_k |\ T_i = pred(M_k) \text{ and } T_j = succ(M_k),$$

$$\sum_{r=1}^{p} \sum_{t=t_i^s}^{t_i^l} (t + CT_{ir}) * X_{irt} \leqslant \sum_{r=p+1}^{p+b} \sum_{t=t_{k'}^s}^{t_{k'}^l} t * X_{k'rt} + \sum_{r=1}^{p} \sum_{t=t_j^s}^{t_j^l} t * X_{jrt} * Y_k$$

where, $k' = n + k$.

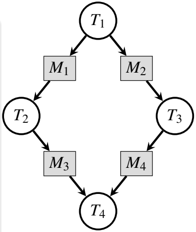


Figure: PTG

ILP2

$\forall M_k |\ T_i = pred(M_k)$ and $T_j = succ(M_k)$,

$$\sum_{r=1}^{p} \sum_{t=t_i^s}^{t_i^l} (t + CT_{ir}) * X_{irt} \leqslant \sum_{r=1}^{p} \sum_{t=t_j^s}^{t_j^l} t * X_{jrt} \quad (15)$$

$$\sum_{r=1}^{p} \sum_{t=t_i^s}^{t_i^l} (t + CT_{ir}) * X_{irt} \leqslant \sum_{r=p+1}^{p+b} \sum_{t=t_{k'}^s}^{t_{k'}^l} t * X_{k'rt} + C * Y_k \quad (16)$$

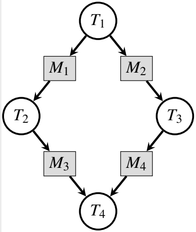


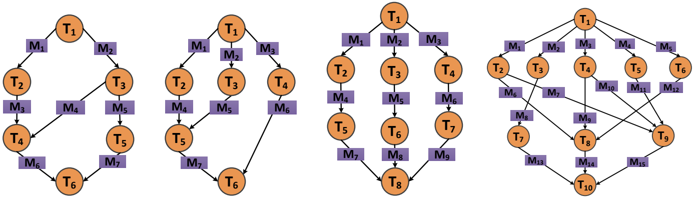
Figure: PTG

where, $k' = n + k$ and C is a sufficiently large constant.

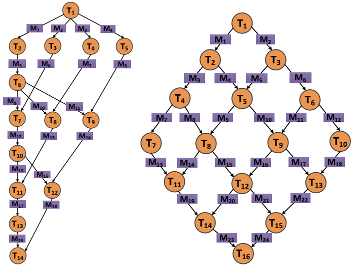
Experimental Setup

- We evaluate and compare the performance of ILP1 and ILP2
- Performance metrics
  - #Constraints generated
  - Time required to generate a solution
- Experiments have been conducted using six standard PTGs
- The scenarios considered differ in terms of,
  - Number of processing elements (p)
  - Number of buses (b)
  - Communication to Computation Ratio (CCR)
  - Deadline (D)
- All experiments are carried out using the CPLEX optimizer [9] version 12.6.2.0, executing on a system having Intel(R) Xeon(R) CPU running Linux Kernel 2.6.32-042stab123.1

Experimental Setup



(a) PTG1 [10] (b) PTG2 [11] (c) PTG3 [11] (d) PTG4 [6]



(e) PTG5 [6] (f) PTG6 [11]

Figure: Benchmark PTGs from [6, 10, 11]

Experiment-1

Compared ILP1 and ILP2

- #processing elements (p) = 4
- #buses (b) = 2
- *Communication to Computation Ratio* (CCR) = 0.5
- Execution/transmission times generated from a uniform random distribution within the range 5 *ms* to 15 *ms* and scaled properly

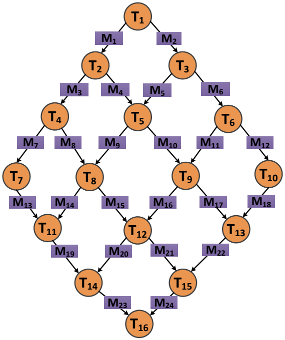
PTG	*n*	*m*	*D*	*SL*	Running Time		#Constraints	
					*ILP*1	*ILP*2	*ILP*1	*ILP*2
*PTG*1	6	7	32	32	0.19	0.07	4681	4112
*PTG*2	6	7	37	37	0.34	0.10	8734	7925
*PTG*3	8	9	46	42	7.68	3.72	28601	26918
*PTG*4	10	15	42	38	35.57	5.73	48905	46064
*PTG*5	14	19	80	72	111.44	24.73	147313	141512
*PTG*6	16	24	72	67	1577.40	171.98	226443	218535

Table: Running time (seconds) and #constraints for PTGs

Experiment-2

Compared ILP1 & ILP2 (varying number of task and message nodes)

- PTG6a: Eliminate message nodes M_{11}, M_{16}, M_{17} and task node T_9 from PTG6
- PTG6b: Eliminate message nodes M_9, M_{14}, M_{15} and task node T_8 from PTG6a



(a) PTG6 [11]

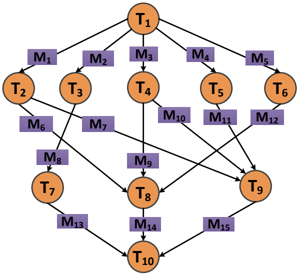
PTG	*n*	*m*	*D*	*SL*	Running Time		#Constraints	
					*ILP*1	*ILP*2	*ILP*1	*ILP*2
*PTG*6	16	24	72	67	1577.40	171.98	226443	218535
*PTG*6*a*	15	21	69	64	208.07	35.81	160889	154682
*PTG*6*b*	14	18	63	58	53.56	10.17	95857	91711

Table: Performance comparison w.r.t PTGs 6, 6a and 6b (second)

Experiment-3

This experiment compares run time overheads incurred by ILP2

- Parameters are,
  - $p \in \{2, 4\}$
  - $b \in \{1, 2\}$
  - $CCR \in \{0.25, 0.5, 0.75\}$
  - $DR \in \{1.0, 1.1, 1.2\})$
  - DR refers to the ratio $(D : SL)$



(b) PTG4 [6]

		$CCR = 0.25$				$CCR = 0.5$				$CCR = 0.75$			
		SL	DR 1	DR 1.1	DR 1.2	SL	DR 1	DR 1.1	DR 1.2	SL	DR 1	DR 1.1	DR 1.2
$p = 2$	$b = 1$	57	10.93	17.46	131.25	55	38.15	117.27	85.65	58	595.45	514.54	1702.12
	$b = 2$	57	9.61	21.61	79.76	54	21.27	33.16	71.15	52	35.35	109.02	85.31
$p = 4$	$b = 1$	42	37.30	109.53	168.96	45	27.30	186.67	173.46	56	27024.51	3925.43	9331.68
	$b = 2$	37	1.53	14.76	23.71	38	1.83	5.64	20.11	45	66.82	59.31	150.42

Table: Running time of ILP2 (in seconds) w.r.t PTG4 for different #resources, *DR* and *CCR*

Case Study: *Adaptive Cruise Controller*

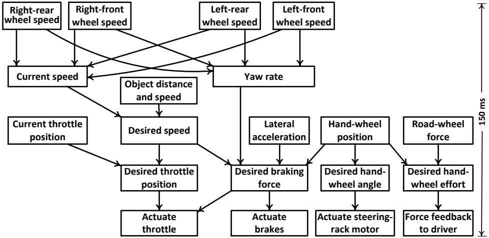


Figure: ACC Block Diagram [12]

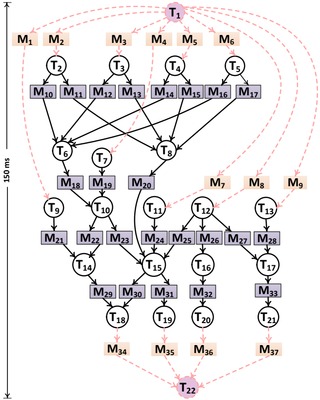


Figure: PTG for ACC

Case Study

	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}	T_{21}
P_1	29	29	26	29	21	43	36	21	37	25	21	20	36	29	43	36	21	21	21	21
P_2	25	27	29	35	23	45	43	25	43	28	25	30	30	27	40	40	18	17	25	22
P_3	32	21	27	27	20	37	45	24	45	26	19	25	40	31	45	30	23	24	20	24
P_4	30	35	34	26	17	40	40	29	40	20	18	26	32	28	42	34	20	18	19	25

Table: Computation time (in *ms*) of task nodes

	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}	M_{33}
B_1	1	1	1	1	2	2	2	2	3	1	1	2	1	1	1	1	1	1	3	3	3	3	2	1
B_2	2	2	1	1	1	1	1	1	2	2	3	3	2	2	2	3	3	3	1	2	2	2	1	1

Table: Transmission time (in *ms*) of message nodes

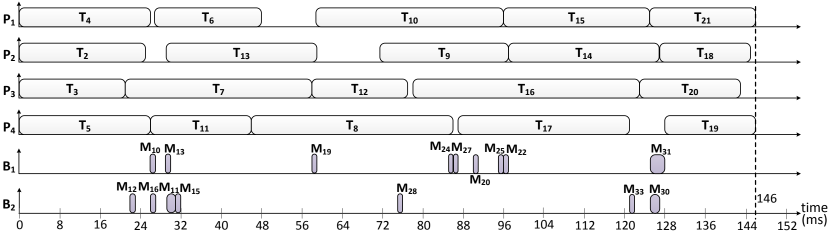


Figure: Gantt chart representation of the schedule

Case Study

Observations:

- ILP2 takes approximately 21872 secs (∼6 hours)
- Makespan is 146 *ms*
- Message nodes M_{14}, M_{17}, M_{18}, M_{21}, M_{23}, M_{26}, M_{29} and M_{32} are absent in the schedule
- All scheduling constraints are satisfied

Conclusion

- This work considers the problem of computing optimal schedules for PTGs executing on distributed systems consisting of heterogeneous processing nodes and inter-connected via a limited number of shared buses
- The first version of the proposed ILP formulation requires two sets of computationally expensive linearizations
- Proposed an improved version of the ILP which reduces computational overheads by elegantly avoiding a sub-set of linearizations that are required to handle dependency constraints
- Experimental analysis using standard benchmark PTGs reveal the practical efficacy of our scheme
- Finally, a case study on a cruise control application has been presented

[1] Z. Guo, R. Liu, X. Xu, and K. Yang, "A survey of real-time automotive systems," 2017.

[2] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2014.

[3] G. Xie, R. Li, and K. Li, "Distributed computing for functional safety of automotive embedded systems," 2016.

[4] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer, 2011, vol. 24.

[5] N. Kandasamy, J. P. Hayes, and B. T. Murray, "Transparent recovery from intermittent faults in time-triggered distributed systems," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 113–125, 2003.

[6] H. Topcuoglu *et al.*, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[7] G. Xie *et al.*, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems," *Journal of Parallel and Dist. Comp.*, 2015.

[8] S. Venugopalan *et al.*, "ILP formulations for optimal task scheduling with communication delays on parallel systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 142–151, 2015.

[9] "CPLEX Optimizer: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer." [Online]. Available: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer

[10] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment." *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, 2016.

[11] A. Olteanu and A. Marin, "Generation and evaluation of scheduling DAGs: How to provide similar evaluation conditions," *Computer Science Master Research*, vol. 1, no. 1, pp. 57–66, 2011.

[12] C. Bolchini and A. Miele, "Reliability-driven system-level synthesis for mixed-critical embedded systems," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2489–2502, 2013.

Thank You