2014 IEEE 28th International Conference on Advanced Information Networking and Applications

A High-performance DAG Task Scheduling Algorithm for Heterogeneous Networked Embedded Systems

Guoqi Xie, Renfa Li,Xiongren Xiao,Yuekun Chen
Key Laboratory for Embedded and Network Computing of Hunan Province
Hunan University
Changsha, China
{xgqman@126.com, lirenfa@vip.sina.com, xxr@hnu.edu.cn, chenyuekun@126.com}

***Abstract*—A high-performance scheduling for a DAG (Directed Acyclic Graph) task graph on heterogeneous networked embedded systems or parallel and distributed systems is to maximize concurrency and minimize inter-processor communication. Most of the algorithms using upward rank value for task prioritizing and earliest finish time for processor assignment. But both approaches ignored the heterogeneity of system and could not create accurate and efficient schedules. Yet no one has doubled about and recognized that. A fully heterogeneous task scheduling algorithm is proposed to address the above problems in this paper. The fundamentals of DAG model and corresponding algorithms are investigated. New concepts called Heterogeneous Upward Rank Value (HURV) and Heterogeneous Priority Rank Value (HPRV) are defined. An algorithm called Heterogeneous Select Value (HSV) is proposed in paper. Both benchmark and extensive experimental evaluation demonstrate the significant improvements in proposed algorithm.**

Keywords-heterogeneous networked embedded systems; heterogeneous upward rank value; heterogeneous select value; DAG

I. INTRODUCTION

With higher demands on vehicle safety and reliability, new vehicle heterogeneous buses, e.g. FlexRay, MOST and Ethernet are integrated into automotive electronic systems, resulting in quantities of heterogeneous Electronic Control Units (ECUs) which are massively surging [1, 3]. Constraints and dependencies between functions in systems are more complex; the proportion of non-periodic tasks with precedence constraints is growing rapidly [2]; heterogeneity and networking become more evident. Automotive electronic systems have evolved into heterogeneous networked embedded systems. Directed acyclic graph (DAG) is a classic parallel and distributed computing model [7], in which tasks are interconnected via heterogeneous links (buses) and have obvious dependence and constraint relationships [4]. Hence, heterogeneous networked automotive electronic systems can be molded with DAG to schedule.

Many functions, e.g. anti-lock brakes, brake-by-wire system in automotive electronic systems are highly performance-critical and safety-critical applications. These applications must be completed in shortest time and produce correct results to ensure the safety of vehicle and occupants. In automotive electronic systems, analyzing message and task scheduling is mainly based on Worst-Case Response Time (WCRT) [17], by comparing with its deadline to determine whether it can meet the real-time requirements. The WCRT is defined as the time-span between the times when the first task allocated ECU for execution and the times when the last task is completed.

Scheduling such a task graph on a set of processors (ECUs) for fastest execution is a well-known NP-hard optimization problem [5, 6] and many heuristics algorithms have been proposed in real-time embedded systems or parallel and distributed systems [7-14]. The heuristics have to find the trade-off between maximizing concurrency and minimizing inter-processor communication. The core idea of list scheduling includes two phases, where the first phase is to order tasks in a list in descending order of priorities and the second phase is to assign each task to a proper processor. Most of algorithms use upward rank value for ordering tasks and using earliest finish time for assigning processors. It is increasingly recognized that the upward rank value and earliest finish time are given by homogeneous environments and does not permit creating accurate and efficient schedules in heterogeneous computing systems. Yet no such algorithm has been proposed to a fully consideration of the heterogeneity. This paper proposes a heterogeneous task scheduling algorithm. We investigate the fundamentals for task scheduling in heterogeneous computing systems and propose an algorithm which is based on heterogeneity.

II. SCHEDULING MODEL

The application can be represented by a directed acyclic graph $G=(V,E,w,c)$,where V represents the set of v nodes(tasks), each node $v_i \in V$ represents an task, which has variable value in different processor. E is the set of communication edges. The directed edge $e_{i,j}$ which joins nodes v_i and v_j represents the precedence constraint which means task v_i should complete its execution before task v_j starts its execution and the weight of $e_{i,j}$ is represented by $c_{i,j}$,which is the average communication cost from v_i to v_j. If v_i and vj are assigned to the same processor, the $c_{i,j}$ is zero .

$pred(v_i)$ represents the set of v_i's immediately predecessor tasks, $ind(v_i)$ represents v_i's in-degree which means the number of immediately predecessor tasks. The

1550-445X/14 $31.00 © 2014 IEEE
DOI 10.1109/AINA.2014.123

task is triggered to execute only if its all predecessor tasks have been executed. *succ*(v_i) represents the set of v_i's immediately successor tasks; *outd*(v_i) represents v_i's out-degree which means the number of immediately successor tasks. The task which has no predecessor tasks is called v_{entry} and the task which has no successor tasks is called v_{exit}.

W is a *v* x *p* computation costs matrix in which each $w_{i,k}$ is denoted by the execution time to run task v_i on processor p_k, Fig.1 shows an example DAG with assigned nodes and edge weights and Table Ⅰ is corresponding computation costs matrix. The example shows number of 10 tasks and 3 processors, e.g. the weight 18 of edge between v_1 and v_2 in in Fig.1 represents the communication cost and is denoted with $c_{1,2}$=12. The weight 14 of v_1 and p_1 in Table Ⅰ represents the computation cost and is denoted with $w_{1,1}$=14.

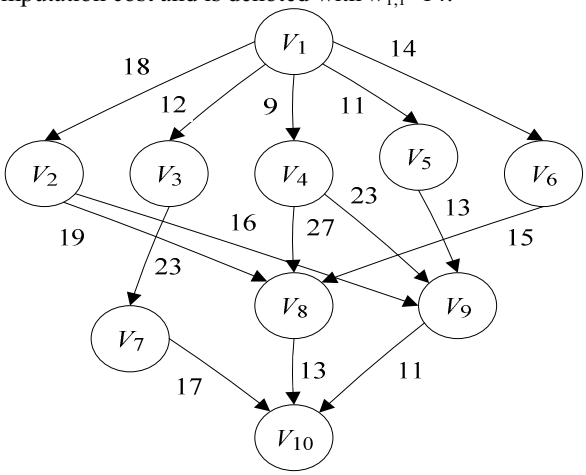

Figure 1. Example of DAG task model

TABLE I. DAG COMPUTATION COST MATRIX (W) IN FIG.1

Task	P_1	P_2	P_3
v_1	14	16	9
v_2	13	19	18
v_3	11	13	19
v_4	13	8	17
v_5	12	13	10
v_6	13	16	9
v_7	7	15	11
v_8	5	11	14
v_9	18	12	20
v_{10}	21	7	16

III. RELATED WORK

Most of scheduling algorithms are categorized into list scheduling, cluster scheduling, duplication based scheduling and random search based scheduling [14]. Because list scheduling is simple and has good performance with low complexity, this section summarizes the approaches adopted by different list scheduling strategies.

CPOP (Critical Path On a Processor) algorithm has two phases [7]: task prioritizing and processor assignment. In the task prioritizing phase, each task has two attributes which are the upward rank value (the upward rank value of task v_i is the longest path from v_i to exit task, the computation of v_i is included) and downward rank value (the download rank value of task v_i is the longest path from entry task to v_i, the computation of v_i is excluded). The sum of upward rank value and download rank value is as the priority value. The entry task's priority value is the length of critical path (LCP). Hence, any tasks whose priority value is equal to LCP could be labeled with critical path tasks. The ready task with the highest priority is selected for processor assignment. In the processor assignment phase, the critical path processor is defined as the processor that minimizes the cumulative computation cost of the tasks on the critical path. If the selected task is on the critical path, it is assigned to the critical path processor; otherwise, it is assigned to a processor that minimizing the finishing execution time of t he task. The complexity of the CPOP algorithm is O (pv^2). The priorities ordered by critical path could lead to a failure for start and completion timely of some nodes which have earliest opportunities. All critical path tasks are assigned to the same processor, this approach could cause load unbalance of processors and increase the schedule length.

HEFT (Heterogeneous Earliest Finish Time) algorithm made some progress in two phases [7]. The upward rank value is computed in task prioritizing phase. Task priority queue is generated by ordering the tasks in decreasing order of the upward rank value which is considered the ordering criteria. In processor assignment phase, tasks are selected in order of their priorities and scheduled to the best processor that has the earliest finish time of the task based on the insertion manner. The complexity of the HEFT algorithm is O (pv^2). HEFT algorithm computes the upward rank value with the average computation cost. This approach has two problems: (1) it does not consider the differences on heterogeneous processors; (2) the processor assignment which is based on earliest finish time does not consider the entire topology of DAG. Nevertheless, because of its low algorithm complexity and producing relative shorter scheduling length, HEFT becomes the most popular and widely used algorithm.

CEFT (Constrained Earliest Finish Time) algorithm is based on the concept of constrained critical paths (CCPs) [14], a notion of ready queues at an instance of scheduling and take the collection of tasks at one instance into account. All the tasks in a CCP are assigned the same processor to eliminate and reduce the communication cost, but cause other tasks to be delayed. What is more important is that this approach has high complexity due to a consideration of a broader view of the task graph and many times computing on CCP.

On the basis of the above algorithm strategies, task duplication algorithms are used to reduce scheduling length, e.g. HCNF (Heterogeneous Critical Node First) [10], HCPFD (Heterogeneous Critical Parents with Fast Duplicator) [11] and TANH (Task duplication-based scheduling Algorithm for Network of Heterogeneous

systems) [13] were proposed. Task duplication technologies can reduce communication cost and scheduling length, but in turn may cause number of unnecessary redundant tasks reducing utilization of processors. Sometimes, complexity of eliminating redundant tasks is even higher than the scheduling algorithm itself.

Upward rank value and earliest finish time are employed in above algorithms. But most of them have not been realized that the heterogeneity could affect the scheduling length. In next section, we will propose a novel task algorithm which fully takes the heterogeneity into account and has shorter scheduling length.

IV. PROPOSED ALGORITHM

This section presents an algorithm for list scheduling called HSV algorithm, which aims to achieve high performance and low complexity. As the general algorithmic approach, list scheduling, our proposed algorithm also consists of two phases. The benchmark which is used in CPOP, HEFT, PETS [9], HCNF and HCPFD is to be used to explain our proposed algorithm.

A. Task Prioritizing

As we know the upward rank value of task given by (1) is employed in many algorithms[7,8,10,12] and is considered as the common task prioritizing, where the nodes are ordered according to their non-increasing.

$$rank_u(v_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_u(v_j)\} \quad (1)$$

We can find that (1) used the average computation cost $\overline{w_i}$. But in heterogeneous systems, each task has variable computation cost in different processors, hence, each task must has corresponding upward rank value in different processors. Equation (1) did not consider the heterogeneity of processor and we should define a new upward rank value of homogeneous computing environments actually. In the following, we define a more accurate value of upward rank value for heterogeneous networked systems.

Definition 1: Heterogeneous Upward Rank Value (HURV) of task v_i means the longest path from task v_i to exit task in processor p_k, the computation of v_i is included.

$$\begin{cases} hurv(v_i, p_k) = \max\limits_{v_j \in succ(v_i)} \{hurv(v_j, p_k) + w_{i,k} + c_{i,j}\} \\ hurv(v_{exit}) = w_{eixt,k} \end{cases} \quad (2)$$

We have determined the average heterogeneous upward rank value for Heterogeneous computing environments on the basis of (2) and defined it as follows:

Definition 2: Average Heterogeneous Upward Rank Value (AHURV) of task v_i represents an average value which is the sum of heterogeneous upward rank value of v_i on each processor.

$$ahurv(v_i) = \frac{1}{|P|} \times \sum_{p_k \in P} hurv(v_i, p_k) \quad (3)$$

The *hranku*(v_i, p_k) and *ahranku*(v_i) of benchmark are given in (2) and (3) as showing in Table Ⅱ.

TABLE II. UPWARD RANK VALUES OF TASKS

function	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	n_{10}
hurv (v_i,p_1)	111	79	79	86	75	67	45	39	50	21
hurv (v_i,p_2)	103	69	75	66	56	62	39	31	30	7
hurv (v_i,p_3)	108	81	86	87	70	67	44	43	47	16
ahurv (v_i)	107	76	80	80	67	65	43	28	42	15

Communication cost is a major bottleneck which affects the performance of the scheduling. As analyzed above, the approaches like collection of tasks and task duplication are not the fundamentally solutions. Here, we give a new definition of task.

Definition 3: Out-degree Communication Cost Weight (OCCW) of task v_i means the possible max sum of communication cost generated by v_i with it's immediately successors.

$$\begin{cases} occw(v_i) = \sum\limits_{v_j \in succ(v_i)} c_{i,j} \\ occw(v_{exit}) = 0 \end{cases} \quad (4)$$

Out-degree communication cost weight also affects the task priorities ordering, the task which has larger out-degree communication cost weight is not executed and may could result in all the successors not being ready, it is easy to yield a large number of communication cost which could increase the length of the DAG scheduling directly or indirectly.

Definition 4: Heterogeneous Priority Rank Value (HPRV) of task v_i represents the sum of out-degree communication cost weight and average upward rank value of task v_i:

$$hprv(v_i) = occw(v_i) + ahurv(v_i) \quad (5)$$

Descending order of priority rank weight *hprv*(v_i) is as the task prioritizing in our approach, because HPRV is more accurate than upward rank value not only on homogeneous environments, but also on communication cost. The out-degree communication cost weight and priority rank weight of each task for instance provided by Fg.1 and table.1 is shown in Table Ⅲ.

TABLE III. PRIORITY WEIGHTS OF TASKS

function	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	n_{10}
occw(v_i)	64	35	23	50	13	15	17	13	11	0
ahurv (v_i)	105	76	80	80	67	65	43	28	42	15
hprv(v_i)	169	111	103	130	80	80	60	41	53	15

B. Processor Selection

As we known the attributes $est(v_i, p_k)$ and $eft(v_i, p_k)$ represents the earliest start time(EST) and the earliest finish time(EFT) of task v_i on processor p_k respectively. The Equations of EST and EFT are defined as

$$\begin{cases} est(v_{enry}, p_k) = 0 \\ est(v_i, p_k) = max(avail[k], \max\limits_{v_j \in pred(v_i)} \{aft(v_j^k) + c'_{i,j}\} \end{cases} \quad (6)$$

$$eft(v_i, p_k) = est(v_i, p_k) + w_{i,k} \quad (7)$$

Where $avail[k]$ is the earliest time at which processor p_k is ready for task executions; $c'_{i,j}$ has two cases, if v_i and v_j is in the same processor, $c'_{i,j}$ =0 ,else $c'_{i,j} = c_{i,j}$; $aft(v_j^k)$ is the actual finish time of task v_j , which is assigned to processor p_k, if v_i is the exit task, $aft(v_j^k)$ is the schedule length(also called makespan) of DAG which is defined as follows:

$$makespan \quad eft(v_{exit}) \quad (8)$$

CPOP, HEFT and CEFT only consider the earliest finish time as the processor selection criteria; this is a typical greedy strategy. However, this approach neglects one more important effective factor, that is, the topology of DAG. Then a new concept is defined as Least Distance Exit Time (LDET) and is given by

$$ldet(v_i, p_k) = hurv(v_i, p_k) - w_{i,k} \quad (9)$$

A task has minimum EFT based on downward, but if it has max LDET which based on upward, the schedule length of DAG may not be reduced. Thus we must optimize the processor criteria not only from "downward" but also from "upward".

Definition 5: Heterogeneous Select Value (HSV) of task v_i means the multiply of EFT and LEDT of task v_i which is executed on process p_k,

$$hsv(v_i, p_k) = eft(v_i, p_k) \times ldet(v_i, p_k) \quad (10)$$

For processor selection, classical DAG algorithms not take the perspective of heterogeneity into the account of processor selection criteria, in which the LDET value of task is the same in heterogeneous processors and needn't consider it. In our approach, LDET is different and as one of the factors for processor selection which is HSV.

C. HSV Algorithm

We proposed the algorithm called Heterogeneous Select Value and we listed out the steps as follows.

Algorithm HSV (Heterogeneous Select Value)

1: Init: compute each task's $hrank_u(v_i,p_k)$, $occw(v_i)$, $hprv(v_i)$, and put all tasks to Queue according to non-increasing $hprv(v_i)$
2: **while** there are tasks to be scheduled in Queue **do**
3: Select task v_i with max $hprv(v_i)$
4: Compute the $eft(v_i,p_k)$, $ldet(v_i,p_k)$ and $hsv(v_i,p_k)$ respectively
5: Assign task v_i to the processor p_k that has the minimum HSV using insertion-based scheduling policy
6: Mark v_i with scheduled task
7: **end while**

The time complexity of scheduling algorithms for parallel application DAG is usually expressed in terms of number of task v and number of processors p. The time-complexity of HSV algorithm is analyzed as follows. Computing $hrank_u$ (v_i,p_k),$occw(v_i)$ and $hprv(v_i)$ must traverse all tasks and compare processors which could be done within O(pv) in initialization phase. Scheduling all tasks must traverse all task which could be done in O(v) . Computing the HSV of all task which could be done in O(pv). Thus, the complexity of the algorithm HSV is O(pv^2) which is the same as the complexity of algorithm HEFT and CPOP.

We list the Gantt graph of some algorithms based on benchmark, as shown in Fig. 2. The task ordering of HSV algorithm is $\{n_1,n_4,n_2,n_3,n_6,n_5,n_7,n_9,n_8,n_{10}\}$ and the schedule length is 73.The corresponding result is shown in Table Ⅳ,and we can see that the makespan of CPOP,HEFT and CEFT are 86,80 and 81 respectively, HSV algorithm has shorter makespan and less communication costs. The makespan of HCNF and HCPFD which employed task duplication are 73[10,11].HSV algorithm which has not employed task duplication and it not increased in time complexity achieve the same schedule length with HCNF and HCPFD. HSV algorithm shows better scheduling performance with no task duplication.

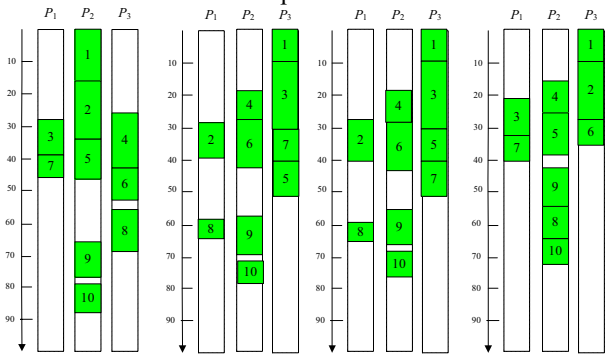

Figure 2. Gannt chart of scheduling DAG of 4 algorithms

TABLE IV. RESULTS OF SCHEDULING DAG OF 4 ALGORITHMS

results	CPOP	CEFT	HEFT	HSV
Task Prioritizing	$n_1,n_2,n_3,n_7,n_4,n_5,n_9,n_6,n_8,n_{10}$	$n_1,n_3,n_7,n_4,n_2,n_6,n_5,n_9,n_8,n_{10}$	$n_1,n_3,n_4,n_2,n_5,n_6,n_9,n_7,n_8,n_{10}$	$n_1,n_4,n_2,n_3,n_5,n_6,n_7,n_9,n_8,n_{10}$
makespan	86	81	80	73

I. EXPERIMENTAL EVALUATION

A. Performance Metrics

The performance metrics chosen for the comparison are the schedule length ratio (SLR) and the speedup. SLR is computed by dividing the real schedule length by the minimum execution time of all tasks in critical path as shown in (11)[12]:

Speedup is computed by dividing the sequential execution time (i.e., cumulative execution time) by the parallel execution time (i.e., the makespan of the output schedule) as shown in (12):

Meanwhile, we also consider the real computing averments and chose WCRT which is the core performance evaluation of automotive electronic systems as one of the metrics [16].

$$SLR = \frac{makespan}{\min_{p_k \in P}\{\sum_{n_i \in CP} w_{i,k}\}} \tag{11}$$

$$Speedup = \frac{\min_{p_k \in P}\{\sum_{v_i \in V} w_{i,k}\}}{makespan} \tag{12}$$

B. Randomly Generated Graphs

Random task graphs have been generated by TGFF 3.5 (Task Graphs for Free) [17] and programmed by Java and Highcharts to compare the results.

The TGFF allows the user to generate a variety of test DAGs with various characteristics that depends on several input parameters and they are number of tasks set v={30,40,50,60,70,80,90,100}, out degree set β={1,2,3,4,5}, in-degree set γ={1,2,3,4,5}, shape parameter set α={0.5,1.0,2.0}, Communication to Computation Ratio (CCR) set is{0.1,0.5,1.0,5.0,10.0}and Range percentage of computation cost set η={0.1,0.5,1.0}[15].

Fig.3 and Fig.4 show the average SLR and average Speedup for varying number of tasks respectively, each data node is the average value given by the number 1125 experiments.

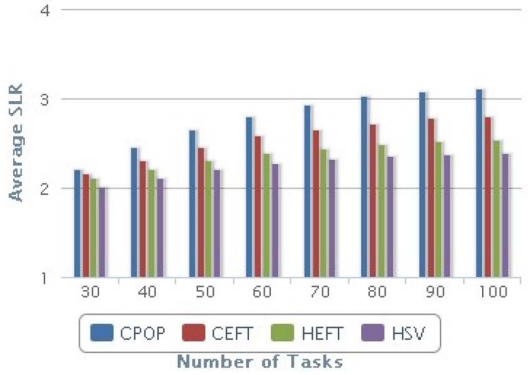

Figure 3. Average SLR for varying number of tasks

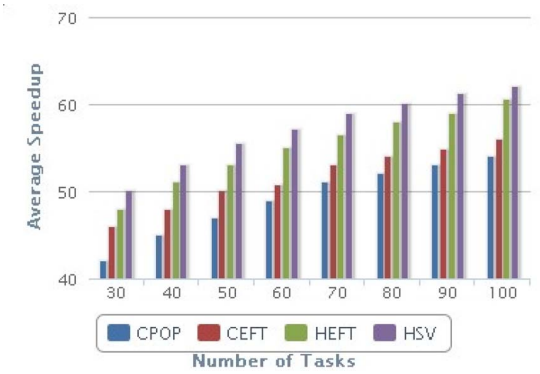

Figure 4. Average Speedup for different number of tasks

HSV algorithm has higher performance compared other algorithms; CPOP is the worst because all the critical path tasks are in the same processors.

η is basically the heterogeneity factor for processors speeds. A high percentage value of η causes a significant difference in a task's computation cost among the processors and a low percentage indicates that the expected execution time of a task is almost equal to any given processor in the system and then, and the computation cost of each task v_i on each processor p_k in the system is randomly set from the following range:

$$w_i * (1-\eta/2) <= w_{i,k} <= w_i * (1+\eta/2) \tag{13}$$

The larger η, the more heterogeneous[15], HSV algorithm fully consider the heterogeneous characteristics with both task priority ordering phase and processor select phase and it has higher performance than other algorithms.

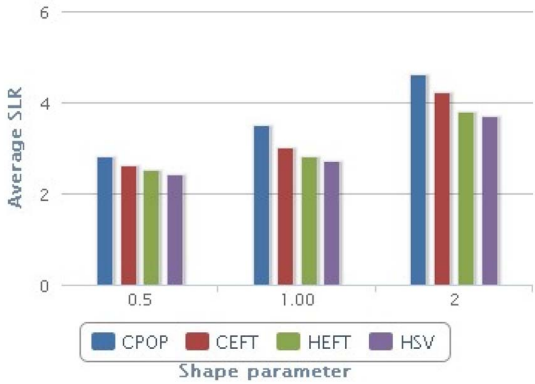

Figure 5. Average SLR for different Shape parameter

C. Real Automobile Electronic Environments

HSV algorithm is proposed based on the background of heterogeneous networked embedded systems. We have analyzed the WRCT for different number of messages in real automobile electronic environments. We employed the real message set based on single CAN network which is provided by embedded system research center, Nagoya university, Japan. The message set contains 65 messages and is assigned

in 14 ECUs. The results show that the HSV algorithm has shortest WCRT and outperform others by 20%.

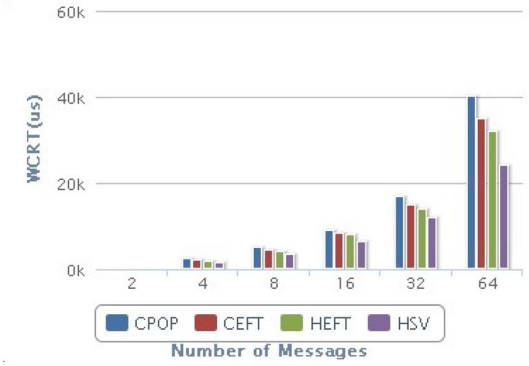

Figure 6. WCRT for different number of messages

II. CONCLUSION

This paper proposed a novel and high-performance DAG task scheduling algorithm called HSV algorithm for heterogeneous networked automobile electronic systems. we found that exist efficient DAG task scheduling algorithms did not fully consider the heterogeneity and we proposed the concept of heterogeneous upward rank value , heterogeneous priority rank value and heterogeneous select value for task priorities ordering and processor selection respectively. Based on this, the high performance HSV algorithm was proposed to reduce scheduling length and has minimum SLR and WCRT by comparing with other algorithms.

ACKNOWLEDGMENT

This work is partially supported by The National High-Tech Research and Development Plan of China under Grant No.2012AA01A301-01 and The National Natural Science Foundation of China under Grant No.61173036.

REFERENCES

[1] Buckl C, Camek A, and Kainz G. “The software car: Building ICT architectures for future electric vehicles”.Electric Vehicle Conference (IEVC), 2012 IEEE International. IEEE, 2012,pp.1-8.

[2] Furst S. “Challenges in the design of automotive software”.Proceedings of the Conference on Design, Automation and Test in Europe. European Design and Automation Association, 2010,pp. 256-258.

[3] Konik D. “Development of the Dynamic Drive for the new 7 Series of the BMW Group”. International journal of vehicle design, 2002, 28(1), pp. 131-149.

[4] Klobedanz K, Koenig A, and Mueller W. “A reconfiguration approach for fault-tolerant flexray networks”.Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. IEEE, 2011, pp.1-6.

[5] V. Sarkar. “Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors”. MIT Press, 1989.

[6] J.D. Ullman. “NP-complete scheduling problems, Journal of Computer and System Sciences”. 1975, pp.384–393.

[7] Topcuoglu H, Hariri S, and Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3) ,pp.260-274.

[8] Sinnen O, To A, and Kaur M. “Contention-aware scheduling with task duplication”. Journal of Parallel and Distributed Computing, 2011, 71(1) ,pp.77-86.

[9] Ilavarasan E, Thambidurai P. “Low complexity performance effective task scheduling algorithm for heterogeneous computing environments”. Journal of Computer Science, 2007, 3 (2) ,pp.94-103.

[10] Baskiyar S, SaiRanga PC. “Scheduling directed a-cyclic task graphs on heterogeneous network of workstations to minimize schedule length”.IEEE Conference on Parallel Processing Workshops, 2003.,pp.97- 103.

[11] Hagras T, Janeček J. “A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems”. Parallel Computing, 2005, 31(7) ,pp.653-670.

[12] Tang X, Li K, and Li R. “Reliability-aware scheduling strategy for heterogeneous distributed computing systems”. Journal of Parallel and Distributed Computing, 2010, 70(9) ,pp.941-952.

[13] Agrawal DP. “Improving scheduling of tasks in a heterogeneous environment”. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(2) ,pp.107-118.

[14] Khan M A. “Scheduling for heterogeneous systems using constrained critical paths”. Parallel Computing, 2012, 38(4) ,pp.175-193.

[15] O.Sinnen. “Task Scheduling for Parallel Systems”. Wiley,2007.

[16] Chen Y, Zeng G, and Ryo K. “Effects of queueing jitter on worst-case response times of CAN messages with offsets”. In Proc. of the Embedded System Symposium in Japan, 2012,pp.119-126.

[17] Dick R P, Rhodes D L, and Wolf W. “TGFF: task graphs for free”.Proceedings of the 6th international workshop on Hardware/software codesign. IEEE Computer Society, 1998,pp.97-101.