处理顺序约束的信息物理融合系统静态任务表调度算法

王小乐 [1] 黄宏斌 [1] 邓 苏 [1]

摘 要 针对异构环境并行计算的静态任务调度问题, 以最小化有向无环图 (Directed acyclic graph, DAG) 的执行跨度为目标, 改变 HEFT (Heterogeneous earliest finish time) 算法中任务上行权重的计算方法, 获得更加合理的任务顺序排列, 提出了一种最早完成时间优先的表调度算法 IHEFT (Improvement heterogeneous earliest finish time). 该算法在计算任务的上行权重时, 分别计算该任务分配给不同资源的上行权重, 取其最小值, 比使用所有资源对该任务的平均处理时间进行计算的 HEFT 算法更为准确. 确定任务的处理顺序后采用最早完成时间越小越优先的策略将任务分配给最优资源, 并使得任务的开始执行时间和结束时间满足 DAG 中有向边的通讯时间约束. 通过使用部分文献中的算例数据以及随机生成满足一定结构要求的 DAG 进行算法测试, 将 IHEFT 与 HEFT, CPOP (Critical-path-on-a-processor) 和 LDCP (Longest dynamic critical path) 进行了比较, 结果显示 IHEFT 算法更有效, 而且时间复杂度较低.

关键词 异构计算环境, 信息物理融合系统, 有向无环图, 任务调度, 表调度, 静态任务

引用格式 王小乐, 黄宏斌, 邓苏. 处理顺序约束的信息物理融合系统静态任务表调度算法. 自动化学报, 2012, **38**(11): 1870−1879

DOI 10.3724/SP.J.1004.2012.01870

List Scheduling Algorithm for Static Task with Precedence Constraints for Cyber-physical Systems

WANG Xiao-Le[1] HUANG Hong-Bin[1] DENG Su[1]

Abstract In this paper, we study the static task scheduling problems in distribution heterogeneous computing environment such as cyber-physical system (CPS). We present a list scheduling algorithm named improvement heterogeneous earliest finish time (IHEFT) algorithm for minimizing makespan of the precedence constrained applications which can be modelled as a directed acyclic graph (DAG). We acquire a better task list by changing the task′s upward rank weight calculation method in heterogeneous earliest finish time (HEFT). In IHEFT, the different computing time not the average computing time in each resource is considered for each task when calculating the upward rank weight. After the priority list is determined, tasks are assigned to the resource based on the earliest finish time first policy and the precedence constraints. Comparison on performance evaluation using both the case data in the recent literature and randomly generated DAGs show that the IHEFT list scheduling algorithm has outperformed the HEFT, CPOP (Critical-path-on-a-processor) and LDCP (Longest dynamic critical path) algorithms both in makespan and scheduling time consumed.

Key words Heterogeneous computing environment, cyber-physical systems (CPS), directed acyclic graph (DAG), task scheduling, list scheduling, static task

Citation Wang Xiao-Le, Huang Hong-Bin, Deng Su. List scheduling algorithm for static task with precedence constraints for cyber-physical systems. *Acta Automatica Sinica*, 2012, **38**(11): 1870−1879

随着网络技术的不断发展, 越来越多单个能力有限的设备通过高速网络连接, 构成异构处理环境, 这种集成异构计算资源的系统为任务调度带来了新的挑战, 例如信息物理融合系统 (Cyber-physical system, CPS)[1] . CPS 是近些年出现的一个新概念, 是网络技术、传感器技术和微处理系统发展的结果, 它强调计算过程和物理过程密切融合[2]. 网络化的 CPS 将具有不同处理能力的异构资源通过有线/无线网络相互连接构成分布式异构处理环境. CPS 中包含了大量能力有限的资源, 在处理用户应用请求时将应用分解为若干相互依赖任务进行并行处理, 这样不仅可以提高系统的性能, 同时也可以保证任务执行的可靠性[3].

用户请求应用一般可以分解为若干相互依赖的任务, 通过有向无环图 (Directed acyclic graph,

收稿日期 2011-06-27 录用日期 2012-07-09
Manuscript received June 27, 2011; accepted July 9, 2012
国家高技术研究发展计划 (863 计划) (2011AA010106), 国家自然科学基金 (71071160) 资助
Supported by National High Technology Research and Development Program of China (863 Program) (2011AA010106) and National Natural Science Foundation of China (71071160)
本文责任编委 刘民
Recommended by Associate Editor LIU Min
1. 国防科学技术大学信息系统工程重点实验室 长沙 410073
1. Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073

DAG) 来描述, 其中每个节点表示一个任务, 节点之间的有向边表示任务之间的依赖关系. 一般情况下的多机调度问题是 NP 完全问题[4], 分布式异构处理环境中的任务调度问题复杂度更高. 任务调度分为静态调度和动态调度. 根据 DAG 所提供的信息, 可以计算每个任务在各个资源上的处理时间以及任务之间的依赖, 采用非抢占式调度, 称为静态调度[5]; 反之, 则为抢占式调度[6], 或称动态调度[7]. 静态任务调度的方法主要有两种: 基于启发式的调度和智能搜索调度, 其中启发式调度主要有表调度算法、基于分簇的调度算法和基于复制策略的调度算法[8], 静态任务调度算法分类如图 1 所示.

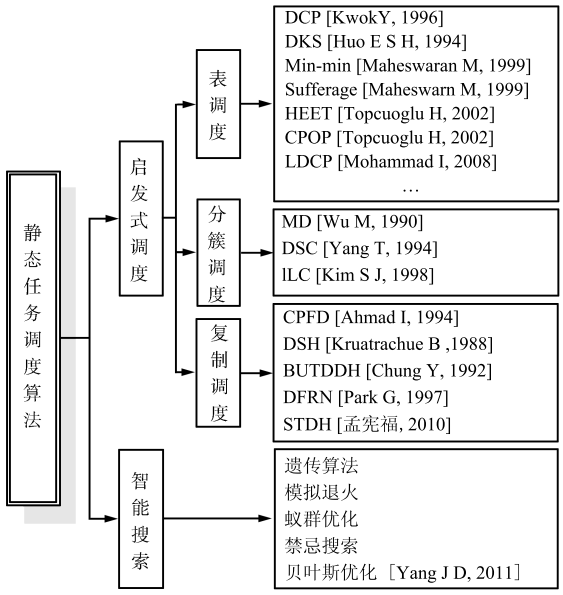


图 1　静态任务调度算法分类

Fig. 1　Taxonomy of the DAG static task scheduling algorithm

表调度以其简单的思想和较高的调度效率受到广泛的研究, 到目前为止已经出现了很多算法, 如: HEFT (Heterogeneous earliest finish time)[8], CPOP (Critical-path-on-a-processor)[8], LDCP (Longest dynamic critical path)[9] 等. 表调度算法的主要思想为: 首先, 计算任务的优先级并对任务排序; 然后, 按照任务处理顺序, 将任务分配给合适的资源. 此处假设一个资源在同一时刻只能处理一个任务, 并且每个任务只能由一个资源来处理. 文献 [8] 提出了两种性能有效的低复杂度的表任务调度算法 HEFT 和 CPOP, 算法处理异构环境下任务之间有依赖关系的调度问题, 算法可以将任务插入相应资源的空闲时间, 而且任务在资源之间的处理时间可以是非单调的. LDCP[9] 算法也是处理异构环境下的任务具有依赖关系的调度问题, 该算法对每个资源都定义一个 DAGP (Directed acyclic graph that corresponds to a processor), 在关键 DAGP 上进行关键路径调度完成任务排序, 但是该算法假设任务在资源之间的处理时间是单调的, 这在 CPS 的异构处理环境中不一定成立. 除了上述较经典的基于表的调度算法外, 文献 [10] 研究分布式表调度问题, 不采用传统的只用一个表的集中式调度, 而是在整个系统中建立多个任务表. 文献 [11] 研究网格环境下的任务表调度算法, 提出了六种启发式算法: MM-STFT (Minimizing both makespan and task finish time), TMWD (Matching tasks with data), Slack, LSlack, LevelU, NLPETS (Non-leveling of performance effective task scheduling).

除表调度算法外, 基于复制的调度算法也得到了很大的发展[5,12−13], 文献 [12] 通过定义关键任务和空闲时间片, 提出了一种异构环境下关键任务调度算法 HCT (Heterogeneous critical task), 该算法将关键任务复制到具有空闲时间片的资源上, 从而提早任务的开始时间. 此外, 面向 DAG 调度的方法还有: 表调度和复制结合的方法[14]、基于状态转移的调度[15]、遗传算法求解调度问题[16]、贝叶斯优化算法[17]、免疫克隆选择[18]、随机调度算法[19] 等.

本文针对资源对任务处理时间具有非单调性的异构处理环境下任务调度问题, 提出了一种表调度算法 IHEFT (Improvement heterogeneous earliest finish time). 该算法通过计算更加合理的节点上行权重获得一个更好的调度序列, 然后采用最早完成时间优先策略[8] 进行调度, 通过对文献 [8] 中的算例进行调度, 说明算法流程和效果. 最后, 随机生成 DAG 进行大量实验对比和分析, 算法比 HEFT, CPOP 和 LDCP 在各种结构的 DAG 中都有较好表现.

1　问题描述

一般情况下将 DAG 描述为二元组: $G(T,E)$, 其中 T 表示 DAG 中节点的集合, 描述的是任务集 $T=\{t_1,t_2,\cdots,t_n\}$, $n=|T|$ 表示任务总数; E 是图中有向边, 每个边连接两个任务节点, $E=\{e_{ij}\,|e_{ij}=\langle t_i,t_j\rangle,e_{ij}\in T\times T\}$, 如果 $e_{ij}\in E$ 表示任务 t_i 和 t_j 之间有依赖关系, t_j 必须在 t_i 完成后才能开始处理, 如图 2 所示. 假设 CPS 中资源之间的网络是高速同构网络, 任意两个资源之间的通讯速率相同, 那么 DAG 中每条边对应一个通讯开销, 用 $c_{i,j}$ 描述任务 t_i 和任务 t_j 不在同一资源上处理所需要的通讯开销, $C=\{c_{i,j}\,|e_{ij}\in E,c_{ij}>0\}$ 表示整个任务图的通讯开销集合. 设 CPS 所拥有的异构资源集合为 $R=\{r_1,r_2,\cdots,r_m\}$, 资源之间的处理能力不同, 而且存在偏好, 用 $w_{i,j}$ 表示任

务 t_i 在资源 r_j 上的处理开销 (一般指时间), 那么 $W = [w_{i,j}]_{n\times m}$ 是任务集合在资源上的处理时间矩阵, 如表 1 所示.

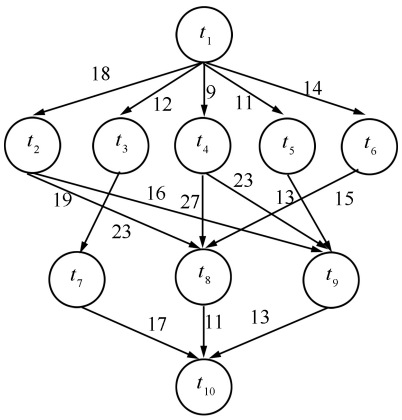


图 2 DAG 结构示意图

Fig. 2 Example of directed acyclic graph

表 1 图 2 中任务在不同资源上的处理时间

Table 1 Execution time of tasks on different resource in Fig. 2

Task	r_1	r_2	r_3
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

定义 1. 处理时间单调性, 资源对任务的处理时间满足如下条件的称为处理时间具有单调性:

$$\forall r_j \forall r_k \left(\exists t_i, (w_{i,j} < w_{i,k}) \rightarrow \forall t_i, (w_{i,j} < w_{i,k})\right)$$

本文中 t_i 的前驱和后继任务分别描述为

$$pred(t_i) = \{t_j \,|t_j \in T, e_{ji} \in E\}$$

$$succ(t_i) = \{t_j \,|t_j \in T, e_{ij} \in E\}$$

入口任务描述为 $t_{\rm entry}$, 出口任务描述为 $t_{\rm exit}$, 那么: $pred(t_{\rm entry}) = \emptyset$, $succ(t_{\rm exit}) = \emptyset$.

如果一个节点既没有前驱也没有后继, 则该任务称为元任务. 如果一个 DAG 存在多个入口节点, 那么可以构建一个虚拟入口节点将其设为 DAG 的多个入口节点的前驱, 它们之间边的通讯开销设为零; 同样, 如果存在多个出口节点也可以构建一个虚拟出口节点, 并将其设置为 DAG 原来的出口节点的后继节点, 它们之间边的通讯开销设为零. 这样就能保证 DAG 只有一个出口节点和一个入口节点, 便于处理.

任务 t_i 在资源 r_j 上的最早开始时间描述为

$$EST(t_i, r_j) = \min\Big\{\tau \,|\tau \geq \max_{t_m \in pred(t_i)} (AFT(t_m) + c'_{m,i}),\ idle^{(r_j)}[\tau, \tau + w_{i,j}]\Big\} \quad (1)$$

其中, $AFT(t_m)$ 表示任务 t_m 的实际完成时间, 当 t_m 实际调度资源为 r_j 时, $c'_{m,i} = 0$, 否则 $c'_{m,i} = c_{m,i}$, $idle^{(r_j)}[\tau, \tau + w_{i,j}]$ 表示资源 r_j 在时间段 $[\tau, \tau + w_{i,j}]$ 中空闲. 最早完成时间为

$$EFT(t_i, r_j) = EST(t_i, r_j) + w_{i,j}$$
$$EST(t_{\rm entry}, r_j) = 0,\ r_j \in R$$

基于表调度的算法中每个 DAG 节点都需要一个权重来计算节点的优先级, 从而建立调度表. 在 HEFT 算法中上行权重[5, 8] 的计算公式为

$$rank_u(t_i) = \bar{w}_i + \max_{n_j \in succ(n_i)} (c_{ij} + rank_u(n_j)) \quad (2)$$

其中, $\bar{w}_i = \sum_{j=1}^{m} w_{i,j}/m$ 表示平均处理成本. 这里使用平均处理时间计算上行权重存在一定的不精确性, 本文给出效果更好的计算公式如下:

$$\begin{aligned} &rank_{\rm up}(t_i) = \\ &\quad \min_{r_j \in R} \max_{t_k \in succ(t_i)} \left\{rank_{\rm up}(t_k) + w_{i,j} + c'_{i,k}\right\} \\ &rank_{\rm up}(t_{\rm exit}) = \min_{r_j \in R} w_{{\rm exit},j} \end{aligned} \quad (3)$$

其中, $c'_{i,k} = \begin{cases} 0, & r_j = pred\_R(t_k) \\ c_{i,k}, & \text{其他} \end{cases}$, $pred\_R(t_k)$ 的含义见定义 2.

定义 2. 在计算上行权重时, 使得权重最小的资源称为任务的预分配资源. 用 $pred\_R(t_i)$ 描述, 如果用 $rank_{\rm up}(t_i, r_j) = \max_{t_k \in succ(t_i)} \left\{rank_{\rm up}(t_k) + w_{i,j} + c'_{i,k}\right\}$ 描述任务 t_i 分配给 r_j 的上行权重, 那么 $pred\_R(t_i)$ 满足: $\forall r \in R$ 有 $rank_{\rm up}(t_i, pred\_R(t_i)) \leq rank_{\rm up}(t_i, r)$ 成立.

定义 3. 任务实际处理时所选择的资源称为该任务的调度资源, 在该资源上处理的起止时间称为调度时间.

整个 DAG 的调度总长度为 $makespan$, 它等于

最后一个任务的完成时间, 有多个出口节点时

$$makespan = \max\{AFT(t_{\text{exit}})\}$$

调度算法的目标是将任务分配到资源上并确定任务的开始执行时间, 使得 $makespan$ 最小.

2　IHEFT 调度算法

2.1　构造调度顺序表

IHEFT 是一种基于表的调度算法, 算法首先建立调度的优先级队列, 然后根据队列顺序逐个任务进行调度, 基本流程和思路如图 3 所示.

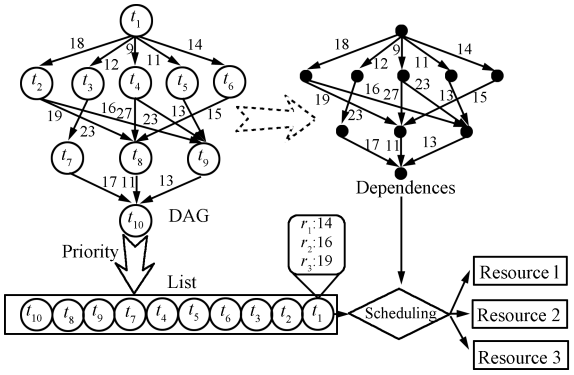


图 3　表调度思路图

Fig. 3　Procedure of list scheduling

算法通过计算每个节点的上行权重, 然后按照权重的降序排列确定任务调度的优先级序列. 上行权重通过式 (3) 计算.

计算每个节点的上行权重时可以从入口节点开始, 使用嵌套函数来计算, 但是嵌套函数的计算复杂性比较高, 因此 IHEFT 算法中通过采用队列和栈两种数据结构配合建立节点的初始优先级队列. 该队列从出口任务开始, 满足每个任务的后继任务必在该任务的前面. 按照该初始优先级队列根据式 (3) 计算每个任务节点的上行权重. 最后, 根据上行权重的降序排列构建任务的优先级队列, 如表 2 中的算法.

对图 2 和表 1 对应的 DAG 计算每个节点的上行权重如表 3 所示. 获得节点的上行权重后, 按照上行权重的降序安排每个节点的调度顺序, 如果权重相同则后继节点多的优先, 如果后继节点也相同, 则随机确定顺序.

表 3 中预分配资源相同, 主要是因为算例通讯开销以及计算开销之间的差异较小, 通讯开销与计算开销之比 (Communication to computation cost ratio, CCR) 大于等于 1, 为了节约通信开销, 任务趋向于分配到同一资源. 如果出口节点的预分配资源确定后很容易使得其他资源的预分配资源与其相同. 根据上行权重的计算公式可知: 任何一个节点的上行权重必大于它的所有后继节点的上行权重. 因此, 按照如上计算的调度顺序表中, 任何一个节点必在其所有前驱节点之后进行调度.

表 2　IHEFT 算法核心代码

Table 2　Key code of the IHEFT

```
Algorithm of list scheduling
for each task t in the pQueue
    t.Ft = maxValue; //Ft is the finish time of t;
    for each resource r ∈ R
        st = 0;
        ft = 0;
        if t = t_entry //t_entry is the entry of DAG;
            st = 0; ft = st + w(t, r);
        else
            for each task tk ∈ pred(t)
                if tk.sR = r; //sR is the selected resource;
                    stt = tk.Ft; ftt = stt + w(t, r);
                else
                    stt = tk.Ft + c(tk, t); ftt = stt + w(t, r);
                end if
                for each task tr if tr.sR = r
                    if (stt < tr.st — ftt > tr.Ft) = false
                        stt = tr.Ft; ftt = stt + w(t, r);
                    end if
                end for each
                if st < stt
                    st = stt; ft = stt + ft;
                end if
            end for each
        end if
        if t.Ft > ft
            t.St = st;
            t.Ft = ft;
            t.resource = r;
        end if
    end for each
end for each
```

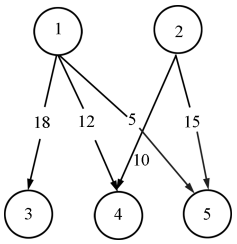
表 3　IHEFT 计算的节点上行权重及调度顺序

Table 3　Task′s $rank_{\text{up}}$ weight and sequence of IHEFT

任务编号	$rank_{\text{up}}$	预分配资源	调度顺序
1	52	2	1
2	37	2	2
3	35	2	3
4	27	2	6
5	32	2	5
6	34	2	4
7	22	2	7
8	18	2	9
9	19	2	8
10	7	2	10

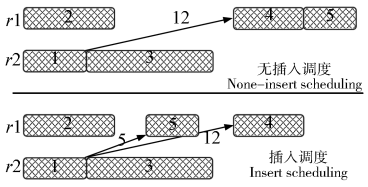
2.2 资源选择

当任务调度的优先级对列构建完成后, 按照队列的顺序, 根据式 (1) 和式 (2) 计算任务在每个资源上的最早开始时间和完成时间, 选择完成时间最早的资源为该任务的调度资源, 在该资源上的最早开始时间和完成时间是该任务的调度时间. 大多数经典的调度算法中计算任务在某个资源上的最早开始时间都是从已分配给该资源的最后一个任务的完成时间开始的, 而在实际中可能存在如图 4 所示的情况: 任务 5 调度时, 计算在资源 r_1 上的最早开始时间, 传统的方法只能在任务 4 结束后, 同时也满足任务 1 与任务 5 之间存在 5 个单位的通讯时间的约束, 如图 4 (a) 所示. 近年出现的启发式表调度算法中采用插入调度方法, 即 r_1 在任务 4 处理之前有足够的空闲时间可以执行任务 5, 同时满足 5 个单位的通讯时间约束, 如图 4 (b) 所示.



(a) 参考图像

(a) Example graph



(b) 真实视差

(b) Comparison of scheduling result

图 4 插入调度示意图

Fig. 4 Diagram of insert scheduling

IHEFT 算法步骤如下:

步骤 1. 初始化优先级队列 pQueue, 保持队列中每个任务的后继任务都在该任务的前面;

步骤 2. 根据式 (3), 按照 pQueue 中的顺序计算每个任务的上行权重;

步骤 3. 按照上行权重的降序重新更新优先级队列;

步骤 4. 根据优先级对列和 DAG 图中的约束依次计算每个任务的调度资源和调度时间.

算法中的重点是步骤 2 和步骤 4, 其伪代码见表 2 和表 4.

表 4 IHEFT 计算优先级算法

Table 4 Algorithm of priority computing in IHEFT

Algorithm of computing *rank*_up

```
for each task t in the pQueue
    t.rank_up = maxValue;//maxValue is plus INF
    for each resource r ∈ R
        rank_up = 0;
        if t = t_exit //t_exit is the exit of DAG
            rank_up1 = w(t, r);
        else
            for each task tk ∈ succ(t)
                if pred_R(tk) = r
                    rank_up1 = tk.rank_up + w(t, r);
                else
                    rank_up1 = tk.rank_up + w(t, r) + c(t, tk)
                end if
            end for each
            if rank_up < rank_up1
                rank_up = rank_up1;
            end if
        end if
        if t.rank_up > rank_up
            t.rank_up = rank_up;
            pred_R(t) = r;
        end if
    end for each
end for each
```

使用 IHEFT 算法对图 2 和表 1 中的数据进行调度产生的调度结果与 LDCP 算法、HEFT 算法获得的调度结果如图 5, 三种算法产生的调度顺序不同, 根据不同的顺序产生的调度结果也不同, 本文所提算法调度结果的 $makespan$ 最小, 说明算法效果最好.

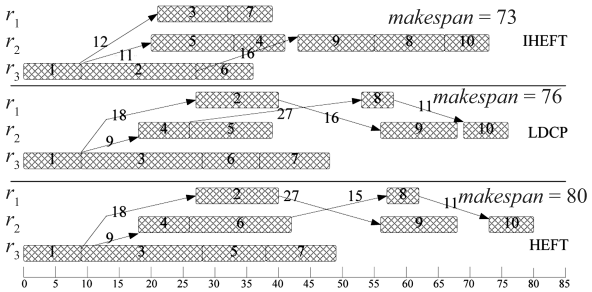


图 5 表调度结果甘特图

Fig. 5 Gant chart of scheduling result with different algorithms

2.3 IHEFT 算法性能分析

基于表的调度算法复杂度比较低, HEFT 算法和 CPOP 算法的时间复杂度均为 $O(m \times e)$[8], 其中 e 为 DAG 中边的数目, m 为资源数[8], 当 DAG 中边密集时, HEFT 算法的时间复杂度为 $O(m \times n^2)$[9]. LDCP 的时间复杂度为 $O(m \times n^3)$. IHEFT 算法的时间复杂度主要是计算上行权重和资源选择两个

步骤中, 两者均为 O($m \times e$), IHEFT 算法的时间复杂度为 O($m \times e$).

DAG 分布式任务调度问题是 NP 问题. 表调度算法在排序的阶段考虑全局信息, 但是由于任务没有分配给具体资源, 因此排序是不准确的, 而按照排序将任务分配给资源时, 由于不能确定先分配任务对后面任务的定量影响, 要获得最优调度顺序通过启发式信息是不现实的. 算法 HEFT 中计算上行权重的过程中, 首先, 没有考虑前驱节点和后继节点是否会在同一资源上处理的情况, 而是直接增加通信开销; 其次增加的计算开销也没考虑具体分配给那个资源, 而是使用任务在资源上的平均处理时间来代替; 如式 (2) 所示, 其中的 $\bar{w}_i = \sum_{j=1}^{m} w_{i,j}/m$, 因此在 IHEFT 中对其进行了更周密的考虑, 见式 (3), 即通过预分配资源确定资源的上行权重, 因此更精确. 但是无论是 HEFT 或 IHEFT 以及现有的其他启发式方法中均不能精确地获得最优的调度顺序.

2.4　不同类算法对比

启发式表调度算法是分布式异构处理环境调度问题多种算法中的一种, 因其简单高效受到广泛关注. 基于 DAG 的调度是 NP 问题, 近些年除了启发式算法, 智能搜索算法也成为解决该问题的重要研究方向, 如 GVNS[20], H2GS[21], PN[22], BOA[17], AIS[23], ICSA[18] 等. 本节根据分析和经验对几类算法进行了定性比较, 详见表 5.

通过表 5 中的比较可见, 启发式表调度算法简单效率高, 调度算法本身不会占用太多的计算成本, 而智能搜索算法一般需要较大的计算成本, 虽然可能获得较好的结果. 启发式表调度算法不需要输入参数, 而智能搜索算法一般都需要用户参数, 如遗传算法中的变异概率、种群规模等. 在 CPS 处理环境中节点的能力一般是有限的, 基于智能搜索算法的调度算法本身会消耗大量的计算成本, 而启发式表调度算法所需计算量仅仅相当于对任务进行一次排序, 计算效率要高很多. 总而言之, 启发式表调度算法在 CPS 应用中具有较大优势, 虽然获得的调度不一定是最好的, 但是效率较高. CPS 环境中资源的接入退出是不确定的, 资源对任务的处理能力并非固定, 而是随着状态改变, 因此算法具有较好的鲁棒性也是很重要的, 鲁棒性是指算法能够接受计算和通讯中存在不确定性而保持解稳定的程度[24].

3　算法实验分析

3.1　实验环境

目前 DAG 算法的公用仿真平台较少, 文献 [26] 中提到名为 EasyGrid 的网格任务调度仿真工具, 但是该工具目前并未对外开放. 根据研究需要, 我们在.net 平台下用 C# 语言开发了一个基于表的 DAG 调度仿真平台 DAGScheduler, 该平台目前有 4 种表调度算法, 如图 6 所示.

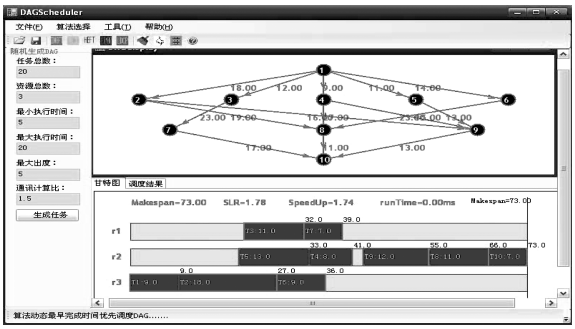


图 6　DAGScheduler 界面

Fig. 6　GUI of DAGScheduler

DAGScheduler 首先融合了 DAG 生成模块, 可以根据用户的定制要求随机产生符合要求的 DAG. 该平台同时包含了算法的性能对比分析模块, 可以通过随机产生大量的 DAG 同时运行多种算法, 将其结果进行对比分析. 同时可以将结果通过文本和甘特图进行可视化展现, 本文通过该平台将 IHEFT 算法与 HEFT、CPOP 和 LDCP 算法进行了对比分析.

3.2　参数及度量标准

本文通过将四种算法在相同的 DAG 上产生的调度结果进行效果比较及算法性能分析. 描述任务数据集和 DAG 的参数有:

表 5　DAG 调度算法比较

Table 5　Comparison of different scheduling algorithms

算法法分类	代表算法	计算步骤	算法成本	最优性	鲁棒性	输入参数	适合的 DAG
启发式表调度算法	HEFT[8], CPOP[8], IHEFT, LDCP[9]	先排序再分配	小	低	强	无	各种
基于复制调度	HEFD[14], MDM-TD[25]	先排序在分配	中	低	差	无	CCR 较大
基于遗传算法	GVNS[20], H2GS[21], PN[22]	先分配再排序	大	高	强	多个	各种
其他智能搜索算法	BOA[17], AIS [23], ICSA[18] 等	先分配再排序	大	高	强	多个	各种

1) 任务个数 n, 即构成 DAG 的节点个数;

2) 资源个数 m, 即处理环境中存在的资源数;

3) 每个节点的最大出度 $maxOutDegree$;

4) 任务在资源上的最小处理时间和最大处理时间 $minprocTime$, $maxprocTime$;

5) 通讯计算比例 CCR, 即平均通讯时间和平均计算时间之比.

DAG 调度算法常用的性能比较参数有 SLR (Schedule length ratio), $speedup$ 以及算法的运行时间. 其中 SLR 的计算如下:

$$SLR = \frac{makespan}{\sum\limits_{t_i \in CP_{\min}} \min\limits_{r_j \in R} \{w_{i,j}\}} \tag{4}$$

其中, $CP_{\min}$ 指当每个任务节点都使用最小计算计算时间的资源时所确定的关键路径上任务节点的集合. $speedup$ 计算如下:

$$speedup = \frac{\min\limits_{r_j \in R} \left\{ \sum\limits_{t_i \in T} w_{i,j} \right\}}{makespan} \tag{5}$$

除此之外, 本文还定义了一个最优率 (BestRatio) 的度量指标. 最优率是指进行 N 次测试后某种算法获得最小 $makespan$ 的次数除以 N.

3.3 实验与结果对比

本文将 IHEFT、HEFT、CPOP 和 LDCP 四种算法分别进行了四种不同的实验.

实验 1. $n = 100$, $m = 10$, $minprocTime = 5$, $maxprocTime = 50$, $maxOutDegree = 4$, $CCR \in \{0.25, 0.5, 1, 2, 4, 8\}$, 进行 100 次实验, 即随机生成 100 个 DAG 进行测试, 获得的 SLR、$speedup$ 和最优次数如图 7 (a) 所示. 通过实验对比分析 IHEFT 获得的平均 SLR 比 HEFT 小 1.2 %、比 CPOP 小 8.6 %、比 LDCP 小 1.7 %; 平均 $speedup$ 比 HEFT 大 2.4 %、比 CPOP 大 9.3 %、比 LDCP 大 2.8 %; 无论 CCR 取值如何, 最小 $makespan$ 次数始终是最多的, CCR 越小, HEFT 与 IHEFT 同时达到最优的情况越多.

实验 2. $m = 8$, $CCR = 2.5$, $minprocTime = 5$, $maxprocTime = 50$, $maxOutDegree = 4, n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, 进行 100 次实验, 获得的 SLR、$speedup$ 和最优次数如图 7 (b) 所示. 通过实验对比分析 IHEFT 获得的平均 SLR 比 HEFT 小 1.2 %、比 CPOP 小 5.3 %、比 LDCP 小 1.0 %; 平均 $speedup$ 比 HEFT 大 1.6 %、比 CPOP 大 5.2 %、比 LDCP 大 1.4 %, 由最优率图可见 IHEFT 优势不随 n 的增大而改变.

实验 3. $n = 100$, $CCR = 2.5$, $minprocTime = 5$, $maxprocTime = 50$, $maxOutDegree = 4$, $m \in \{2, 4, 8, 16, 32\}$, 进行 100 次实获得的 SLR、$speedup$ 和最优次数如图 7 (c) 所示. 通过实验对比分析在资源数较小时, IHEFT 与 HEFT、LDCP 算法的效果差不多, 随着资源数的增多, 任务执行的并行性的增强 IHEFT 算法的优势就比较明显, 当资源超过一定数时几种算法的效果又比较相近, 如图 7(c) 的第 3 个子图所示.

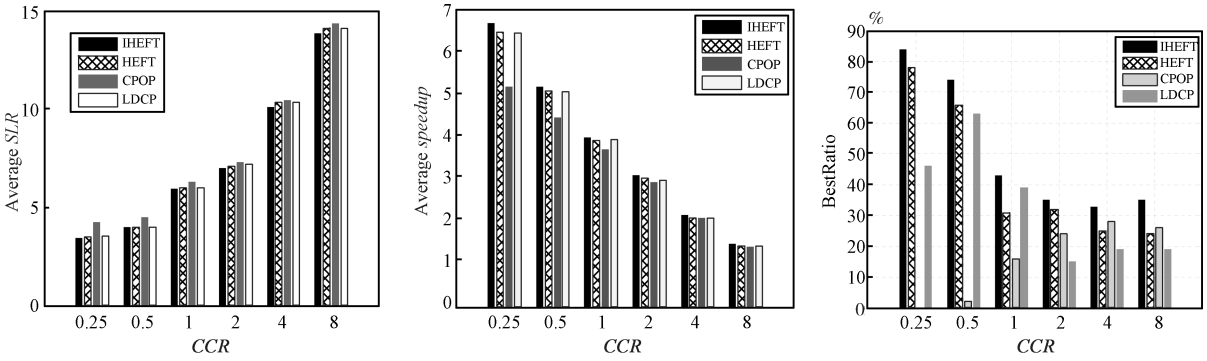
实验 4. $n = 100$, $m = 8$, $CCR = 16$, $minprocTime = 5$, $maxprocTime = 50$, $maxOutDegre \in \{2, 4, 6, 8, 10\}$, 进行 100 次实获得的 SLR、$speedup$ 和最优次数如图 7 (d) 所示, 可见当节点的最大出度较小时几种算法效果相近, 但是随着节点最大出度的增加, IHEFT 算法明显优于其他算法, 说明在 DAG 并行性比较强时 IHEFT 算法更有效.

通过实验可知 IHEFT 算法比 HEFT 算法更有效, 本文对比实验中 LDCP 与 HEFT 算法表现相当, 主要是因为 LDCP 算法强调任务处理时间在资源上是单调的情况[9], 而此处都是非单调的.

3.4 时间复杂度对比

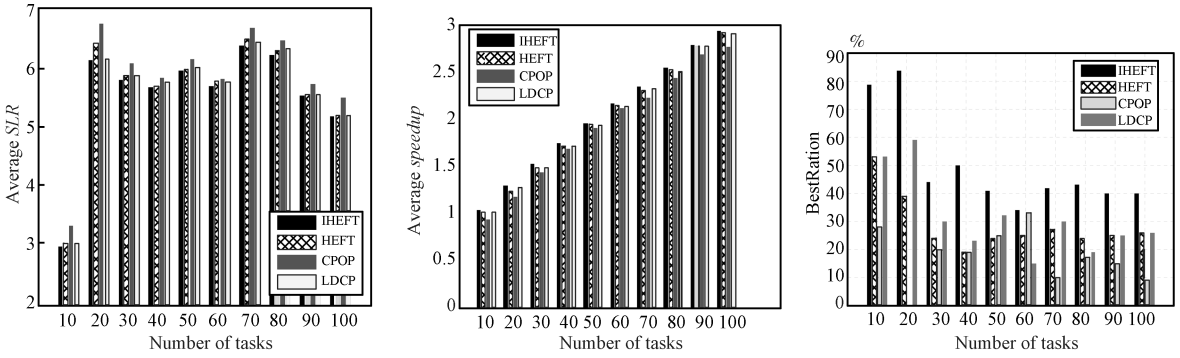
IHEFT 算法与 HEFT 算法的理论时间复杂度相同, 同时进行大量实验估计算法的运行时间, 确定资源数为 8、最大出度为 4、最小处理时间为 5、最大出处理时间为 30、CCR 为 1.5, 通过将任务数从 10 增加到 100, 每次增加 10 个任务, 进行测试, 每种任务数随机生成 5 000 个 DAG, 进行 5 000 次, 此时取平均运行时间, 运行时间与任务数之间的关系如图 8.

由图 8 可以看出, IHEFT、HEFT 和 CPOP 三种算法趋势比较相似, 近似为线性的关系; LDCP 趋势要大一些, 近似为 $O(n^3)$ 关系. 在最大出度确定的情况下, 任务图中边的数目与节点数大致上是一种线性关系, 边数 $e < 4n$, 但是由于在随机生成 DAG 的算法中为了保证只有一个出口节点, 因此节点的出度并非是严格的控制参数, 一般随着任务的增加, 最大出度也会略有增加, 因此边数和节点数大致满足 $e \approx \alpha \cdot n$, 其中 α 为常数, 所以 IHEFT、HEFT 和 CPOP 的运行时间与节点数呈线性关系. 由于个人计算机在算法运行时还有其他应用占用了 CPU, 算法时间的测算并非完全精确, 因此出现了图 8 中的不平滑现象. IHEFT 算法的运行时间略大于 CPOP 和 HEFT, 主要原因是 IHEFT 在计算上行权重的过程中需要计算每个资源上的上行权重而非平均值, 因此计算量略大.



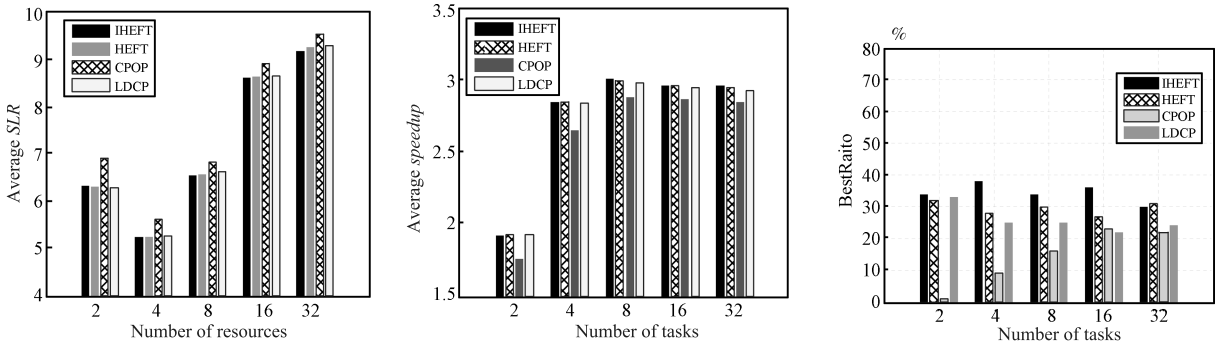
(a) 不同通讯计算比对各算法性能的影响

(a) Performance for each algorithm with changing CCR



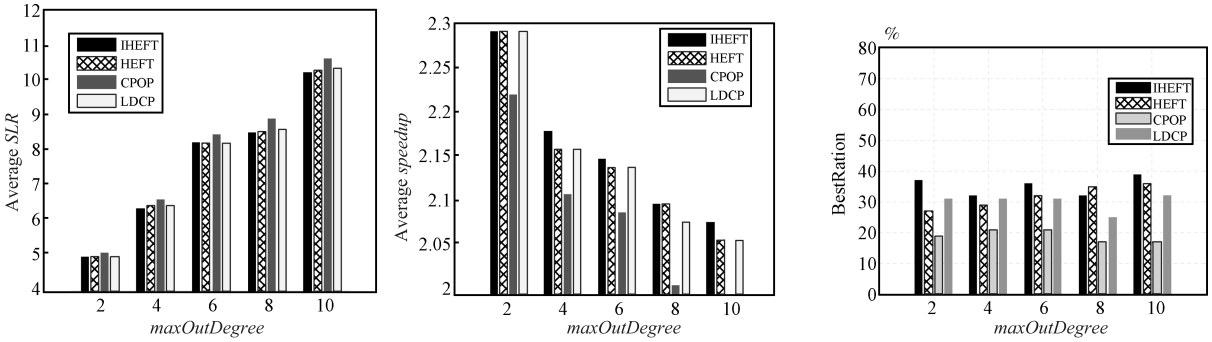
(b) 任务个数对各算法性能的影响

(b) Performance for each algorithm with changing tasks′ number



(c) 资源个数对各算法性能的影响

(c) Performance for each algorithm with changing resources′ number



(d) DAG 节点最大出度对各算法性能的影响

(d) Performance for each algorithm with changing DAG node′s max out degree

图 7　实验结果对比图

Fig. 7　Diagrams of experiment for performance comparison

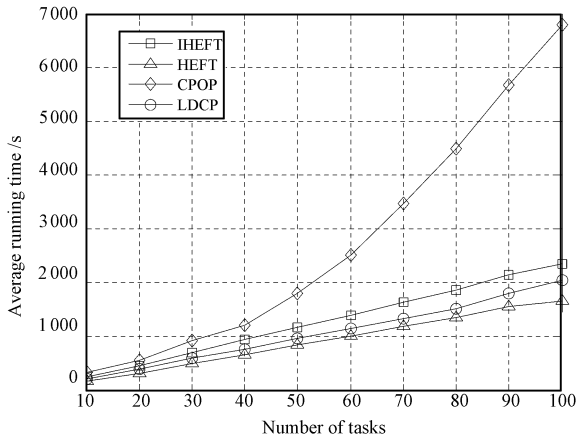


图 8 算法运行时间与任务数的关系

Fig. 8 Relationship between time consumption and task′s number

4 结束语

CPS 是一种典型的分布式异构处理环境, 高效的调度技术能够提高任务处理的并行性[27], 从而增强系统性能. 表调度算法具有较低的复杂性, 本文提出了一种异构环境下的具有顺序依赖任务的表调度算法 IHEFT, 该算法在计算每个任务节点的上行权重时, 对任务分配到不同资源上的情况逐一考虑, 不同于 HEFT 算法中使用平均计算时间, 因此算法更合理. 实验分析比较说明 IHEFT 具有较好的调度效果, 其平均 SLR 比 HEFT、LDCP 和 CPOP 都小.

本文研究发现 DAG 中存在少数重要任务, 如果能够采用复制策略复制这些重要任务到必要的资源上以节约通信开销, 其他任务仍采用表调度, 可能获得更好的调度效果, 这将是下一步研究的内容. 另外, 在不确定环境中由于资源可用性造成调度不可靠, 且算法对数据存在依赖, 因此除了任务执行跨度, 在调度过程中还必须考虑任务的可靠性调度问题以及算法的鲁棒性问题.

References

1 Wen Jing-Rong, Wu Mu-Qing, Su Jing-Fang. Cyber-physical system. *Acta Automatica Sinica*, 2012, **38**(4): 507−517
(温景容, 武穆清, 宿景芳. 信息物理融合系统. 自动化学报, 2012, **38**(4): 507−517)

2 Lee E A. Cyber physical systems: design challenges. In: Proceedings of the 11th IEEE International Symposium on Object Component Oriented Real-Time Distributed Computing. Orlando, USA: IEEE, 2008. 363−369

3 Tang X Y, Li K L, Li R F, Veeravalli B. Reliability-aware scheduling strategy for heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 2010, **70**(9): 941−952

4 Ullman J D. NP-complete scheduling problems. *Journal of Computer and System Sciences*, 1975, **10**(3): 384−393

5 Meng Xian-Fu, Liu Wei-Wei. A DAG scheduling algorithm based on selected duplication of precedent tasks. *Journal of Computer-Aided Design and Computer Graphics*, 2010, **22**(6): 1056−1062
(孟宪福, 刘伟伟. 基于选择性复制前驱任务的 DAG 调度算法. 计算机辅助设计与图形学学报, 2010, **22**(6): 1056−1062)

6 Xie Zhi-Qiang, Xin Yu, Yang Jing. Machine-driven integrated scheduling algorithm with rollback-preemptive. *Acta Automatica Sinica*, 2011, **37**(11): 1332−1343
(谢志强, 辛宇, 杨静. 可回退抢占的设备驱动综合调度算法. 自动化学报, 2011, **37**(11): 1332−1343)

7 Yin Jin-Yong, Gu Guo-Chang, Zhao Jing. Dynamic scheduling algorithm for hybrid real-time tasks with precedence constraints. *Computer Integrated Manufacturing Systems*, 2010, **16**(2): 411−416, 422
(殷进勇, 顾国昌, 赵靖. 优先约束的混合实时任务动态调度算法. 计算机集成制造系统, 2010, **16**(2): 411−416, 422)

8 Topcuoglu H, Hariri S, Wu M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 2002, **13**(3): 260−274

9 Daoud M I, Kharma N. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 2008, **68**(4): 399−409

10 Tchiboukdjian M, Trystram D, Roch J L, Bernard J. List Scheduling: the Price of Distribution, Technical Report inria-00458133, Centre de recherche INRIA Grenoble, France, 2010

11 Falzon G, Li M Z. Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments. *The Journal of Supercomputing*, 2012, **59**(1): 104−130

12 Lan Z, Sun S X. Scheduling algorithm based on critical tasks in heterogeneous environments. *Journal of Systems Engineering and Electronics*, 2008, **19**(2): 398−404

13 Gacias B, Artigues C, Lopez P. Parallel machine scheduling with precedence constraints and setup times. *Computers and Operations Research*, 2010, **37**(12): 2141−2151

14 Tang X Y, Li K L, Liao G P, Li R F. List scheduling with duplication for heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 2010, **70**(4): 323−329

15 Lee Y C, Zomaya A Y. A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 2008, **19**(9): 1215−1223

16 Omara F A, Arafa M M. Genetic algorithms for task scheduling problem. *Journal of Parallel and Distributed Computing*, 2010, **70**(1): 13−22

17 Yang J D, Xu H, Pan L, Jia P F, Long F, Jie M. Task scheduling using Bayesian optimization algorithm for heterogeneous computing environments. *Applied Soft Computing*, 2011, **11**(4): 3297−3310

18 Meng Xian-Fu, Wang Min. Peer to peer task scheduling based on improved immune clonal selection algorithm. *Computer Integrated Manufacturing Systems*, 2009, **15**(9): 1795−1802
(孟宪福, 王敏. 基于改进免疫克隆选择的对等网络任务调度机制. 计算机集成制造系统, 2009, **15**(9): 1795−1802)

19 Tang X Y, Li K L, Liao G P, Fang K, Wu F. A stochastic scheduling algorithm for precedence constrained tasks on Grid. *Future Generation Computer Systems*, 2011, **27**(8): 1083−1091

20 Wen Y, Xu H, Yang J D. A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system. *Information Sciences*, 2011, **181**(3): 567−581

21 Daoud M I, Kharma N. A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous processor networks. *Journal of Parallel and Distributed Computing*, 2011, **71**(11): 1518−1531

22 Page A J, Keane T M, Naughton T J. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *Journal of Parallel and Distributed Computing*, 2010, **70**(7): 758−766

23 Swiecicka A, Seredynski F, Zomaya A Y. Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. *IEEE Transactions on Parallel and Distributed Systems*, 2006, **17**(3): 253−262

24 Canon L C, Jeannot E. Evaluation and optimization of the robustness of DAG schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, 2010, **21**(4): 532−546

25 Chaudhuri P, Elcock J. Scheduling DAG-based applications in multicluster environments with background workload using task duplication. *International Journal of Computer Mathematics*, 2010, **87**(11): 2387−2397

26 Vianna B A, Fonseca A A, Moura N T, Menezes L T, Mendes H A, Silva J A, Boeres C, Rebello V E F. A tool for the design and evaluation of hybrid scheduling algorithms for computational grids. In: Proceedings of the 2nd Workshop on Middleware for Grid Computing. Toronto, Canada: ACM, 2004. 41−46

27 Niu Jian-Jun, Deng Zhi-Dong, Li Chao. Distributed scheduling approaches in wireless sensor network. *Acta Automatica Sinica*, 2011, **37**(5): 517−528
(牛建军, 邓志东, 李超. 无线传感器网络分布式调度方法研究. 自动化学报, 2011, **37**(5): 517−528)

王小乐 国防科学技术大学信息系统与管理学院博士研究生. 主要研究方向为信息物理系统最优观测与控制技术, 分布式系统并行计算. 本文通信作者.
E-mail: shaulor@yeah.net
(**WANG Xiao-Le** Ph. D. candidate at the College of Information System and Management, National University of Defense Technology. His research interest covers optimal observation and control for cyber-physical systems, and distributed systems parallel computing technolgy. Corresponding author of this paper.)

黄宏斌 国防科学技术大学信息系统与管理学院信息系统工程重点实验室副教授. 主要研究方向为信息系统与智能决策技术, Web 语义网与本体技术, 信息物理融合系统.
E-mail: huanghongbin@yahoo.com.cn
(**HUANG Hong-Bin** Associate professor at the Laboratory of Science and Technology on Information Systems Engineering, College of Information System and Management, National University of Defense Technology. His research interest covers information system and intelligent decision, semantic web and ontology technology, and cyber-physical systems.)

邓 苏 国防科学技术大学信息系统与管理学院信息系统工程重点实验室教授. 主要研究方向为信息系统与智能决策技术, 信息物理融合系统.
E-mail: sudeng@sohu.com
(**DENG Su** Professor at the Laboratory of Science and Technology on Information Systems Engineering, College of Information System and Management, National University of Defense Technology. His research interest covers information system and intelligent decision, and cyber-physical systems.)