doi:10.3788/gzxb20144302.0206001

面向光子网格任务调度的迭代列表算法

刘暾东,张春宇,陈俊仁,陈耿,陶继平

(厦门大学 云计算与大数据研究中心;信息科学与技术学院,福建 厦门 361005)

摘 要:光子网格中任务和通信的联合调度是一个非确定性多项式难题.为了进一步优化调度长度,本文在扩展列表算法的基础上,提出一种迭代列表调度算法.该算法通过扩展列表算法产生一个初始调度序列,并通过迭代的方式不断估计调度过程中子任务之间的通信时间;然后重新计算子任务的权重,调整子任务的调度序列,达到改善调度长度的目的.仿真实验表明,迭代调度算法对于大部分的实例能够有效地减少任务的调度长度,并且更加适用于数据密集型的任务调度.

关键词:光子网格;通信时间;任务调度;有向无圈图;列表算法;迭代算法;随机算法

中图分类号:TN929.11 **文献标识码**:A **文章编号**:1004-4213(2014)02-0206001-6

An Iterative List Algorithm for Task Scheduling Based on Optical Grids

LIU Tun-dong, ZHANG Chun-yu, CHEN Jun-ren, CHEN Geng, TAO Ji-ping

(*Research Center for Cloud Computing and Big Data; College of Information Science and Technology, Xiamen University, Xiamen, Fujian* 361005, *China*)

Abstract: The joint scheduling problem of tasks and network communication is a non-deterministic polynomial hard(NP-hard) problem in the optical grid. In order to further extend the list scheduling, an iterative scheduling algorithm was proposed to optimize the scheduling length. Firstly, an initial task list was sorted according to the priority called bottom level of each task. Then the scheduling order was modified according to the updated time-weight of tasks by estimating the actual communication cost between tasks. The simulation results show that the proposed iterative scheduling algorithm can effectively improve the scheduling length in the majority of the cases, especially for data-intensive applications in task scheduling.

Key words: Optical grid; Communication cost; Task scheduling; Directed Acyclic Graph(CAG); List algorithm; Iterative algorithm; Random algorithm

OCIS Codes: 060.1155; 060.1660; 060.4256; 060.4258

0 引言

光子网格通过光纤及其他光网络器件将各种分布在不同物理位置的、远距离的计算资源连接起来,为生物、军事、航天等大规模数据密集型应用提供计算服务[1-3].光子网格的计算资源包括超级计算机、数据中心、显示设备、虚拟现实设备等[4].对于复杂的光子网格系统,如何有效地调度这些计算资源以满足用户所提交的任务请求,一直是光子网格中一项重要的研究内容[5].

光子网格中的任务调度问题可以抽象为存在数据通信的工作流调度问题,一般通过有向无圈图(Directed Acyclic Graph,DAG)来进行建模,常使用最小化调度长度作为目标函数[6].文献[7]证明了该类问题是一个非确定性多项式(Non-deterministic Polynomial, NP)难题,并设计了一种考虑通信延迟的列表算法,文献[8]针对光子网格中的任务调度问题,在文献[7]的基础上考虑了光网络中的路由选择对通信延迟的影响,提出了一个综合考虑任务与通信的联合调度模型,并设计了扩展列表算法.文献[9]在文献[8]的基础上证明了理论调度算法与实际调度存在偏差,并提出一种基于数学模型的精确调度算法.文献

基金项目:国家自然科学基金青年项目(No. 11201391)资助

第一作者:刘暾东(1970-),男,教授,博士,主要研究方向为云计算和网络信息技术. Email:ltd@xmu.edu.cn

通讯作者:陶继平(1980-),男,助理教授,博士,主要研究方向为复杂系统调度与优化. Email:jipingtao@gmail.com

收稿日期:2013-05-31;**录用日期**:2013-08-16

[10]提出一种基于工作流中关键任务优先调度的算法,进一步优化了任务的调度长度.文献[11]说明了启发式调度算法[7-10]在某些工作流实例情况下得不到较好调度结果.文献[12]提出了一种随机调度算法,该算法通过不断随机调整工作流中任务的排序可以找到优于已有算法[7-11]所得到的结果,并且适用于所有工作流实例,但是该算法存在着搜索次数多、结果随机性大的问题.

本文针对光子网格中的任务和通信的联合调度问题,在随机搜索调度算法思想的基础上设计了一种迭代列表调度算法.该算法通过不断估计调度过程中任务之间的通信时间来计算任务的权重,从而调整工作流中任务的排序,起到优化调度长度的目的,仿真实验表明该算法能够有效地减少任务的调度长度,并且更适用于数据密集型的任务调度.

1 光子网格资源调度模型

光子网格资源调度模型包括资源模型、任务模型和模型约束条件.

1.1 资源模型

光子网格的资源包括把各种计算资源连接起来的光链路资源以及光交换节点资源.本文使用无向图 $G_n(N,L,t,bw)$来表示光子网格.$|N|$表示光网格中的节点集合,即系统中计算资源以及光交换节点的集合,其中$|N|=|P|+|S|$,节点 $N\in P$ 表示系统中的计算资源,节点 $N\in S$ 表示系统中的光交换节点资源,type(r)表示资源 r 的类型.$|L|$代表光网络中的光链路集合,其中 $L=L_A+L_T$,光链 L_A 表示光交换节点与计算资源节点之间的接入链路,链路 L_T 表示光交换节点与光交换节点之间的传输链路.bw(l)代表光链路 l 的带宽.$h(n)$代表计算资源 n 的处理能力,使用单位数据在计算资源 n 上进行处理所需的时间进行表示.例如,一个可执行子任务有 3 个单位的数据量,那么这个可执行子任务在计算资源 n 上需要 $3h(n)$的时间才能处理完毕.

1.2 任务模型

用户的任务包括若干个具有依赖关系的可执行子任务以及子任务之间数据通信的任务.使用 DAG 表示一个到达系统的用户应用请求.在 DAG 图中,包含了若干个节点和有向边,其中每个节点表示一个可执行子任务,每一条有向边则用来表示两个有依赖关系的子任务之间的数据通信任务.其中本文使用 $G_n(N,L,t,c)$表示一个 DAG 任务,$|V|$代表子任务的集合,$|E|$代表两个有依赖关系的子任务之间数据通信任务的集合.$e_{ij}\in E$ 则表示子任务 v_i 与子任务 v_j 之间的数据通信任务.type(v)表示子任务 v 的类型,pred(v)表示子任务 v 的所有父任务的集合,succ(v)表示任务 v 的所有子任务的集合,$d(v)$表示子任务 v 的数据量,$c(e)$表示数据通信边 e_{ij} 在具有单位传输能力的光链路上进行传输时的时间,对于一个带宽为 bw(l)的光链路 l 来讲,传输时间为 $c(e)$/bw(l).

1.3 资源模目标函数及约束条件

为了完成将用户的任务调度到光子网格中,表 1 定义以下参量和约束条件.

表 1 光子网格资源调度模型参量

Table 1 Optical grid resource model parameters

Parameter	Meaning
rsv(v)	The resource that task v is assigned to.
$t_s(v)$	Start time of task v on resource.
$t_f(v)$	Finish time of task v on resource.
Rt(e_{ij})	Lightpath for e_{ij} which includes access links and a series of optical links.
bw(Rt)	Bandwidth of lightpath Rt bw(Rt) = min{bw(l_1),…,bw(l_k)}$l_k\in Rt$,$k=1,2,\cdots n$;
$t_s(e_{ij})$	The start time of data communication e_{ij}.
$t_f(e_{ij})$	The finish time of data communication e_{ij} $t_f(e_{ij})=\begin{cases}t_s(e_{ij}),\text{rsv}(v_i)=\text{rsv}(v_j)\\ t_s(e_{ij})+c(e_{ij})/\text{bw(Rt)},\text{otherwise}\end{cases}$.
$t_{dr}(v_j,r)$	Ready time of task v_j on resource $rt_{dr}(v_j,r)=\max\{t_f(e_{ij})\},v_i\in\text{pred}(v_j)$.

目标函数:调度长度最短,即求最后一个子任务的完成时间最小.

$$\min\{\text{SL}\}=\min\{\max_{v_i\in V}[t_f(v_i)]\}$$

类型约束:子任务只能被调度到与自己本身类型相同的机器上执行.

$$\text{type}(v)=\text{type}(r),\ \text{if rsc}(v)=r$$

任务依赖约束:子任务 v_j 要在父任务 v_i 执行结束后才能开始执行,即父任务 v_i 执行结束后,才能开始进行通信任务,所以子任务 v_j 的开始时间一定要大于或等于父任务 v_i 的结束时间.

$$t_f(v_i)\leqslant t_s(v_j),v_i\in\text{pre}(v_j)$$

通信约束:子任务 v_j 必须在所有通信数据都到达后才能开始执行,如果子任务 v_j 和父任务 v_i 在同一台机器运行,则可以忽略通信时间.

$$t_s(v_j)\geqslant\max_{e_{ij}\in E}\{t_f(e_{ij})\}$$

通信依赖约束:对于通信任务 e_{ij} 必须在其父任务 v_i 执行结束之后才能开始执行,其中 $v_i\in\text{pre}(v_j)$.

$$t_f(v_i)\leqslant t_s(e_{ij})$$

2 迭代列表算法

DAG 任务调度问题是一个 NP 难问题,目前许多做法都是通过设计启发式算法在多项式时间内寻找问题的次优解[13].其中列表调度算法是 DAG 任务调度最常用的调度算法,文献[8]提出了一种扩展列表调度算法(Extended List Scheduling, ELS)并将其应用于光子网格

中的任务调度问题中.下文将先介绍扩展列表调度算法，然后重点描述本文设计的迭代列表调度算法.

2.1 扩展列表算法

扩展列表调度算法是在列表算法的基础上提出一种结合自适应路由(Earliest Start Route First，ESRF)的资源选择策略[8]，其算法伪代码见表2.

表2 扩展列表算法伪代码

Table 2 Expanded list algorithm pseudo code

	ELS algorithm
1	Sort each task into a List by decreasing order of their bottom levels
2	While List is not empty
3	Select the first unfinished task v_i
4	For all resource
5	For all task $v_k \in \mathrm{pred}(v_i)$
6	Select path and resource for task by ESRF
7	Calculate $t_f(e_{ki})$
8	End For
9	Record $t_f(v_i)$on every resource
10	Removev_i from List
11	End For
12	Assign task v_i to resource r that minimizes the value of $t_f(v_i)$
13	End while

扩展列表算法主要思想就是为工作流中的每个子任务计算底度(bottom level)，然后按照底度从高到低的顺序将子任务进行排序，每个子任务的底度由式(1)来计算.

$$\mathrm{bl}(i)=c(i)+\max_{j\in \mathrm{succ}(i)}\{\mathrm{bl}(j)+d(e_{ij})\} \tag{1}$$

如图1所示，V_1 的底度为28，V_2 的底度的计算式为 $\mathrm{bl}(v_2)=c(v_2)+\mathrm{bl}(v_5)+d(e_{25})$，得到 V_2 的底度为 $\mathrm{bl}(v_2)=8+28+13=49$，这样依次计算出每个子任务的底度，根据底度大小由高到低将子任务进行排序，并按照排序结果依次将每一个子任务按ESRF算法调度到光子网格中.其中基于自适应路由的资源选择策略是将每个子任务试探性的调度到与之类型相同的可用资源上，每次选择能够使该子任务最早完成的资源及链路进行传输和调度.

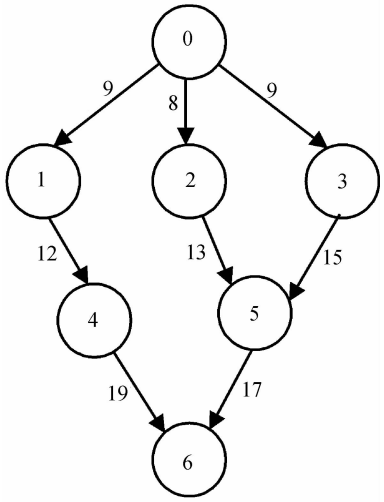

图1 DAG任务

Fig. 1 DAG

2.2 迭代列表算法

由于扩展列表调度算法利用DAG中的通信时间进行底度的计算，但实际调度过程中会出现光链路被占用而产生通信延迟，从而导致实际通信时间与DAG中的通信时间不一致.因此扩展列表调度直接利用DAG中的通信时间计算底度所得到的排队序列并不是最优的调度序列，而随机调度算法通过多次随机调整排队序列可以找到优于扩展列表算法所得到的调度序列，但却存在循环次数多、结果随机性大的问题.根据以上考虑，本文提出一种迭代列表调度算法，该算法是在扩展列表调度的基础上，利用扩展列表调度算法得到初始的调度序列，然后通过估计子任务调度过程中的通信时间，重新计算子任务的底度，从而调整子任务的排队序列得到第二代的调度序列，算法不断迭代重复上述过程，最后选择迭代过程中最优的调度序列进行调度，其迭代列表算法的伪代码见表3.

表3 迭代列表算法伪代码

Table 3 Iterative list algorithm pseudo code

	ILS algorithm
1	$s=0$
2	BestSL$=\infty$
3	While $s<$max do
4	Sort each task into a List by decreasing order of their bottom levels
5	While List is not empty
6	Select the first unfinished task v_i
7	Select path and resource for v_i by ESRF
8	Record actual time of communication
9	Remove v_i from List
10	ScheduleLength$=\max\{t_f(v_i)\}$
11	End while
12	if ScheduleLength$<$BestSL
13	BestSL$=$ ScheduleLength
14	Record ScheduleLength as a best schedule
15	End if
16	Estimate communication times of tasks, sort by recalculating bottomlevel
17	$s=s+1$
18	End while

基于以上思想给出式(2)和式(3)，具体过程描述为

$$d^s(e_{ij})=t_f^{s-1}(e_{ij})-t_f^{s-1}(v_i) \tag{2}$$

$$\mathrm{bl}^s(i)=c(i)+\max_{j\in \mathrm{succ}(i)}\{\mathrm{bl}^s(j)+d^s(e_{i,j})\} \tag{3}$$

式中 $d^s(e_{i,j})$ 代表第 s 代估计的通信时间，文中资源和

路径的选择采用 ESRF.

2.3 算法举例

如图 1,该 DAG 任务图中包含 7 个子任务和 8 条通信边,其中每条通信边上的权重为通信量的值,每个子任务的执行时间和具体类型见表 4,将该子任务调度到图 2 所示的网络中,利用式(1)通过计算每个节点的底度可以得到子任务的执行顺序为:V_0、V_1、V_2、V_3、V_4、V_5、V_6,按照该顺序通过自适应路由算法将子任务逐个调度到网络中,可得到第一代调度长度为 115,其调度过程见图 3(a).

表 4 任务量和类型表

Table 4 Execution time and type of a task

Task	V_0	V_1	V_2	V_3	V_4	V_5	V_6
$D(v)$	9	16	8	5	15	3	8
Type(v)	2	3	2	1	1	1	2

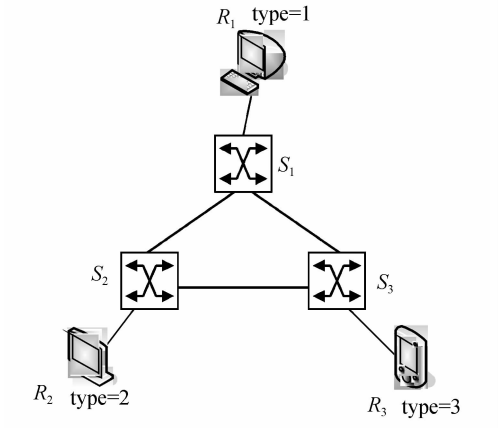

图 2 网络拓扑

Fig. 2 Network topology

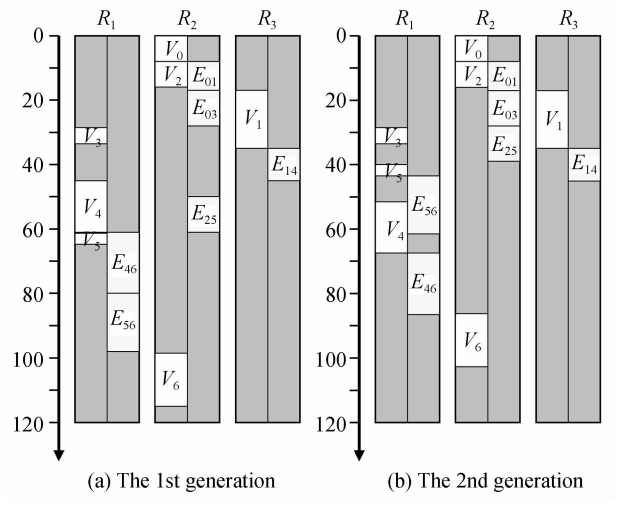

图 3 第一代和第二代调度甘特图

Fig. 3 The first and second generation scheduling gantt chart

第二代根据第一代的结果利用式(2)和式(3)估计调度过程中实际的通信时间,得到表 5. 根据估计的通信时间,通过重新计算每个子任务的底度,可以得到第二代的调度顺序为:V_0、V_2、V_1、V_3、V_5、V_4、V_6,按照该顺序对子任务进行调度可得到新的调度长度为 104,其具体调度过程见图 3(b).

表 5 第二代通信时间表

Table 5 The second generation communication schedule

Edge	E_{01}	E_{02}	E_{03}	E_{14}	E_{25}	E_{35}	E_{46}	E_{56}
$D(E_{ij})$	9	8	9	12	13	15	19	17
$D^2(E_{ij})$	9	8	18	12	42	15	19	35

2.4 算法复杂度分析

光子网格任务调度算法的时间复杂度与子任务节点数量 v、通信边数量 e、光网络交换节点数量 p 以及光网络传输链路数量 l 有关. 迭代列表调度算法在每一次循环中的时间复杂度计算为:更新通信边的值时间复杂度 $O(e)$,计算底度的时间复杂度 $O(vp)$,任务按底度排序的时间复杂度 $O(\text{vlogv})$,资源选择及其路由选择的时间复杂度 $O(v(\text{logp}+1))$,所以迭代列表调度算法每一次循环中总的时间复杂度为 $O(e+vp+\text{vlogv}+v(\text{logp}+1))$.

3 仿真分析

利用 JAVA 开发平台对迭代列表调度算法进行仿真实验. 通过 16 个节点 NSFNET 网络拓扑[8],并且假设网络中每个光交换节点均有一个计算资源与之相连,光交换节点和计算资源节点之间的接入链路的带宽为 1 个单位,异构因子为 1,网络中的计算资源类型共有 3 种,包括计算类型、存储类型、显示类型,值分别为 1、2、3.

实验中采取文献[14]随机生成子任务的方式. 为了满足实验的要求,定义参量:CCR:通信计算比(Communication Computation Ratio,CCR),即 DAG 中所有边的通信时间代价与所有子任节点的执行时间代价的比值. 对于 CCR 的取值,本文采取 0.1、1 和 10 分别仿真低、中、高不同通信密集程度的 DAG 应用. Outdrgee:代表每个子任务节点的平均出度,即每个可执行子任务与其他子任务之间的平均通信代价,本文节点的平均出度取 2.

产生 DAG 的方法:DAG 任务中的每条边以相同的概率产生(基于每个节点的平均出边的数量),DAG 任务中每个节点的任务量服从区间[1,19]的均匀分布,均值为 10,边的权值的选取与设定的 CCR 值和边的平均权值有关,其具体过程见文献[14]并且边的权值也服从均匀分布.

3.1 算法对比实验

实验中为了进行算法对比,采取文献[15]所定义的调度长度改善率 SL_impr 作为评价算法性能的指标,即

$$\text{SL\_impr}=\frac{\text{SL}_{\text{i}}-\text{SL}_{\text{f}}}{\text{SL}_{\text{i}}} \tag{4}$$

式中 SL_i 代表第一次调度得到的调度长度，SL_f 代表最终调度得到的调度长度.

为了比较迭代列表调度算法和扩展列表调度算法的性能，本文采取如下实验方案：通过 16 个节点的 NSFNET 网络拓扑，子任务数量分别为 10、20、50、100、250、500、1 000，CCR 值分别为 0.1、1、10，且针对每种子任务数量以及 CCR 值随机生成 1 000 个实例，仿真实验的总数为 21 000 次，测得仿真实例中平均 74.39%的实例调度长度与扩展列表调度算法相比得到改善，并且调度长度的平均改善率为 4.85%，其结果如表 6 和表 7.

表 6　平均调度实例改善率表

Table 6　Average percentage of improved cases

Task number	Average percentage of improved cases/(%)
10	64.2
20	68.8
50	79.6
100	75.0
250	70.9
500	80.2
1 000	77.3

表 7　平均调度长度改善率表

Table 7　Average improvement ratio

Task number	Average improvement ratio/(%)
10	3.24
20	5.34
50	4.61
100	6.32
250	4.81
500	5.32
1 000	4.2

为了比较迭代列表调度算法和随机调度算法的性能，在相同实例情况下，将随机调度算法和迭代调度算法分别与扩展列表调度算法做比较，通过 16 个节点的 NSFNET 网络拓扑，子任务的数量分别为 10、20、50、100、250、500、1 000，CCR 值分别为 0.1、1、10，且针对每种子任务数量以及 CCR 值随机生成 1 000 个实例，并对比两种算法得到的调度实例改善率和调度长度改善率，其结果如图 4 和图 5.

从图 4 的仿真结果可以看出随机调度算法和迭代列表调度算法随着搜索次数的增加得到的调度实例改善率和调度长度改善率都呈上升趋势；而在循环次数相同的情况下，迭代列表调度算法得到的改善率优于随机调度算法. 可见，迭代列表调度算法在每次循环中利用上一代循环得到的实际通信时间来修改 DAG 图中通信边的值，从而重新修改子任务权重，调整子任务的调度序列的做法要优于随机调度算法在每次循环中都独立的随机调整子任务调度序列的做法. 另外，根据

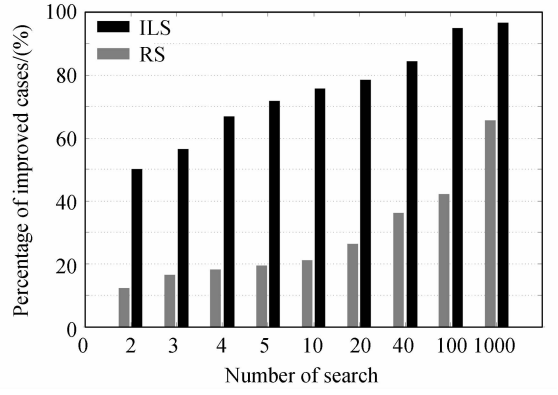

图 4　迭代列表调度算法和随机调度算法平均调度实例改善率对比

Fig. 4　Average percentage of improved cases between iterative list scheduling algorithm and random scheduling algorithm

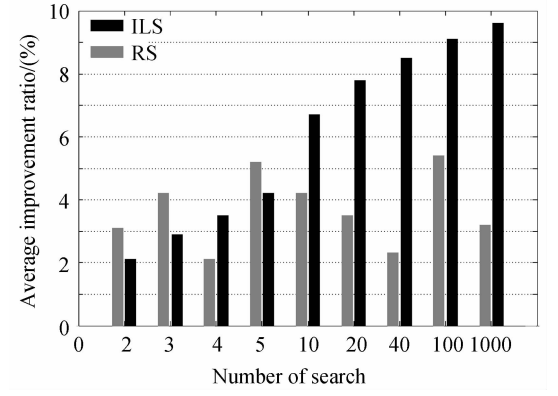

图 5　迭代列表调度算法和随机调度算法平均调度长度改善率对比图

Fig. 5　Average improvement ratio between iterative list scheduling algorithm and random scheduling algorithm

仿真结果还可以看出在循环次数为 5 时，迭代列表调度算法就可以得到较满意的调度结果，因此本文仿真实验过程中采取的循环次数为 5.

从图 5 中可以看出迭代列表调度算法的平均改善率随着循环次数的增加呈上升趋势，而随机调度算法的平均改善率随着循环次数的增加未呈现规律，可见，迭代列表调度算法随着循环次数的不断增加，会逐步改善任务的调度长度，也可以看出随机调度算法的随机性大. 在仿真过程中，随机调度算法在 DAG 图中子任务数较大的情况下，需用通过大量的循环次数才能获得优于扩展列表算法得到的调度长度；而这种情况下，迭代列表调度算法使用较少的循环次数就能获得优于扩展列表算法得到的调度长度.

3.2　CCR 的影响

为了研究 CCR 的不同取值对调度实例改善率和调度长度改善率的影响，本文采取如下实验方案，通过 16 个节点的 NSFNET 网络拓扑，子任务的个数为 10、20、50、100、250、500、1 000，CCR 值分别为 0.1、0.25、

0.5、0.75、1、2.5、5、7.5、10，且针对每种子任务数量以及 CCR 值随机生成 1 000 个实例，测得平均调度实例改善率和平均调度长度改善率如图 6.

从图 6(a)可以看出随着 CCR 的增加调度实例改善率并没有出现明显变化，而从图 6(b)可以看出迭代列表调度算法的调度长度改善比率随着 CCR 值的增加呈上升趋势. 这是因为在随着 CCR 值的增加，数据通信在占用链路的时间也将增加，从而导致子任务的实际通信时间与 DAG 中的通信时间差别变大. 而迭代列表调度算法是按照更加准确的实际通信时间进行子任务排序并调度，因此该算法可以得到更好的调度长度改善率. 可见，迭代列表调度算法更加适用于数据密集型的任务调度.

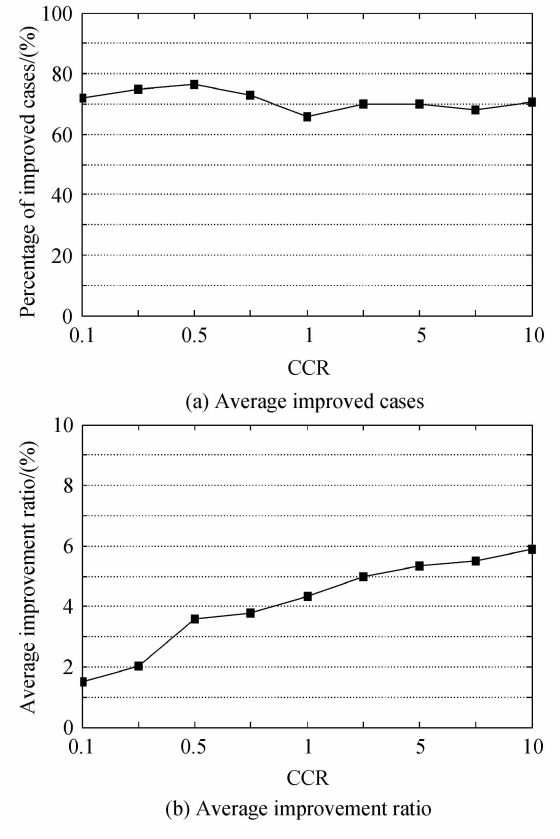

图 6 不同 CCR 值下的平均调，度实例与平均调度实例改善率

Fig. 6 Average improved cases and average improved ratios for different CCR

4 结论

针对光子网格中任务和通信的联合调度问题，在扩展列表算法的基础上提出了一种迭代列表调度算法. 该算法通过迭代的方式不断估计调度过程中子任务之间的通信时间，从而重新计算子任务的权重，调整子任务的调度序列，达到优化调度长度的目的. 在不同子任务数、CCR 值下进行了大量的仿真实验，并将调度实例改善率和调度长度改善率与随机调度算法相对比，仿真结果表明迭代调度算法在大部分的工作流实例中得到的调度长度都优于扩展列表算法，相比于随机调度算法得到了更好的调度实例改善率和调度长度改善率，且使用了更少的循环次数，同时迭代列表调度算法更加适用于通信密集型的任务调度.

参考文献

[1] FOSTER I, GROSSMAN R L. Data integration in a bandwidth-rich world [J]. *Communications of the ACM*, 2003, **46**(11): 50-57.

[2] GUO Wei, SUN Wei-qiang, JIN Yao-hui, *et al*. Demonstration of joint resource scheduling in an optical network integrated computing environment [J]. *Communications Magazine*, 2010, **48**(5): 76-83.

[3] HE Jian-wu, MEI Jie, GU Wan-yi, *et al*. A novel distributed restoration method in intelligent otical networks [J]. *Acta Photonica Sinica*, 2003, **32**(12): 1464-1469.
何建吾，梅杰，顾畹仪，等. 智能光网络中的一种新型的分布式恢复方法[J]. 光子学报，2003, **32**(12)：1464-1469.

[4] VEERARAGHAVAN M, ZHENG X, HUANG Z. On the use of connection-oriented networks to support grid computing [J]. *Communications Magazine*, 2006, **44**(3): 118-123.

[5] GUO Yan-tao, LIU Zeng-ji. QoS policies for core nodes in optical burst switching networks[J]. *Acta Photonica Sinica*, 2005, **34**(11): 1706-1709.
郭彦涛，刘增基. 光突发交换网络核心节点中 QoS 策略研究[J]. 光子学报，2005, **34**(11)：1706-1709.

[6] WU M Y, SHU W, GU J. Local search for DAG scheduling and task assignment[C]. IEEE Proceedings of the 1997 International Conference on Parallel Processing, 1997, 174-180.

[7] SINNEN O, SOUSA L. Communication contention in task scheduling [J]. *IEEE Transactions on Parallel and Distributed Systems*, 2005, **16**(6): 503-515.

[8] WANG Yan, JIN Yao-hui, GUO Wei, *et al*. Joint scheduling for optical grid applications [J]. *Journal of Optical Networking*, 2007, **6**(3): 304-318.

[9] GUO Wei, WANG Zheng-yu, SUN Zhen-yi, *et al*. Task scheduling accuracy analysis in optical grid environments[J]. *Photonic Network Communications*, 2009, **17**(3): 209-217.

[10] SUN Zhen-yu, GUO Wei, WANG Zheng-yu, *et al*. Scheduling algorithm for workflow-based applications in optical grid[J]. *Journal of Lightwave Technology*, 2008, **26**(17): 3011-3020.

[11] KIM S, LEE S, HAHM J. Push-pull: Deterministic search-based dag scheduling for heterogeneous cluster systems[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2007, **18**(11): 1489-1502.

[12] BOYER W, HURA G. Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments [J]. *Journal of Parallel and Distributed Computing*, 2005, **65**(9): 1035-1046.

[13] FOULDS L, GRAHAM R. The steiner problem in phylogeny is NP-complete[J]. *Advances in Applied Mathematic*, 1982, **3**(2): 43-49.

[14] SINNEN O, SOUSA L. List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures[J]. *Parallel Computing*, 2004, **30**(1): 81-101.

[15] LIU G Q, POH K L, XIE M. Iterative list scheduling for heterogeneous computing [J]. *Journal of Parallel and Distributed Computing*, 2005, **65**(5): 654-665.