文章编号：1000-8608(2008)02-0298-06

经济网格中基于剪枝策略的时间-费用优化任务调度

黄飞雪*[1]， 姜新娜[2]， 李志洁[3]， 侯铁珊[1]

（1.大连理工大学 经济系，辽宁 大连 116024；
2.大连理工大学 软件学院，辽宁 大连 116024；
3.大连民族学院 计算机科学与工程学院，辽宁 大连 116600）

摘要：由于在经济网格环境下，存在着资源异构和分布的特征，网格任务调度变成了一个复杂的问题.为此，针对独立任务，在考虑用户的服务质量经济需求偏好的基础上，提出了一个优化用户时间和费用的任务调度方案选择算法.该算法首先将网格中兼顾时间和费用的任务调度方案形式化为一个 n 层 m 叉树，然后将调度方案的选择问题转化为树的遍历问题，最后利用剪枝方法避免无效路径的搜索，降低了时间复杂度，实现了任务的优化调度.结果表明该算法能按照用户的时限和费用需求偏好选择优化的调度方案，且在性能上优于传统的未剪枝算法.所以该算法是一种可行的任务调度算法.

关键词：费用约束；任务调度；剪枝；时限；网格计算
中图分类号：TP393 **文献标志码**：A

0 引 言

对于大规模的资源共享和分布式系统集成，网格计算[1,2]正成为一个主流技术.针对不同的应用领域，已经开发出了一些网格计算系统，主要包括：计算网格[3,4]、数据网格[5,6]、知识网格[7,8]、服务网格[9~11]、经济网格[12,13]和网格工具[14~17]等.但是作为一个新兴的领域，网格计算仍然面临着许多挑战[18]，资源调度[19,20]就是其中一个核心问题，其主要目标是要找到将一组任务按先后顺序条件安排到处理器上使得任务可被“最优”完成的一种方案，以满足各种服务质量(quality of services，QoS)需求.由于网格是一个异构、动态的分布式环境，当供需双方的规模变大时，资源的优化调度将变得十分复杂.目前主要的资源调度方法有：

①基于群体智能(swarm intelligence)[21,22]技术的资源调度，如粒子群优化(particle swarm optimization)算法[23]、蚁群算法[24,25]等.此类算法使用智能的、分布式的方法解决组合优化问题，但往往参数选取复杂，而且实现起来效率不高.

②基于启发式算法的资源调度[26]，代表性的方法有 GA(genetic algorithms)算法[27]、Min-Min 算法等.这类方法的特征是效率较高，但不一定能找到最优的分配方案，而且大多数情况下得到的是主观满意解.

③经济网格环境下，借鉴经济学原理[28~31]，引入经济模型中的买和卖机制，试图通过“价格”这只无形的“手”自动调节理性选择(个人效用最大化)的供需双方均衡，最终达到帕累托(Pareto)最优，从而实现网格资源的动态优化调度.

为此，本文提出将任务调度算法问题转化成树的搜索问题，通过引进剪枝策略思想，减少大量的无效搜索路径，以最终实现时间与费用的网格资源的优化调度.

1 构建时间-费用任务调度模型

约定 网格中有 n 个独立任务，且分别运行在 m 个不同的 CPU 上，其中 $m \geqslant n$. 每个任务都有自己的运行时间权重 w_t 和所需支付运算费用权重 w_m. 对于每个任务，当确定了被分配给网格中的某个 CPU 之后，它的运行时间是可以预先

收稿日期：2006-10-20； **修回日期**：2008-01-16.
基金项目：辽宁省社会科学规划基金资助项目(L07DJY065)；大连理工大学人文社会科学研究基金资助项目(DUTHS2007321).
作者简介：黄飞雪*(1971-)，男，博士生，副教授.

获知的.同样地,把某一个任务传输到网格中指定的CPU所需的时间亦为已知,它可由数据量大小和当地的网络速度得到.

(1) 定义耗费时间矩阵 $\boldsymbol{T}_{mn}$

T_{ij} = 任务 X_i 被分配给 CPU S_j 后的运行时间 + 将任务 X_i 传输到 S_j 所需要的时间 ($i \in \{1,2,3,\cdots,n\}$; $j \in \{1,2,3,\cdots,m\}$) (1)

由约定知 T_{ij} 均为已知.

(2) 定义一个数组 $\boldsymbol{M}_n$

用来存贮每个任务所需支付费用权重和时间权重之比,即费用相对时间的重要性,如方程(2)

$$M_i = W_{mi}/W_{ti}(i \in \{1,2,3,\cdots,n\}) \quad (2)$$

W_{mi} 为 i 个任务的运行费用权重;W_{ti} 为 i 个任务的运行所需支付运算时间权重.

(3) 定义矩阵 $\boldsymbol{W}_{mn}$

W_{ij} = CPU S_j 每单位的费用 × 任务 X_i 被分配到 CPU S_j 后的运行时间 ($i \in \{1,2,3,\cdots,n\}$; $j \in \{1,2,3,\cdots,m\}$) (3)

(4) 定义矩阵 $\boldsymbol{R}_n$

调度方案可以表示成一个0-1矩阵 $\boldsymbol{R}_{mn}$.其中 $R_{ij}=1$ 表示任务 X_i 在CPU S_j 上运行,否则取0.由约定知,本调度方案的表示方法可进一步简化为数组 $\boldsymbol{R}_n$.其中 $R_i = j$ 表示任务 X_i 在机器 S_j 上运行($i \in \{1,2,3,\cdots,n\}$; $j \in \{1,2,3,\cdots,m\}$).

(5) 定义一条路径上的总耗费

$$cost = \sum_{i=1}^{n} M_i \times W_{ij} + \max T_{ij} \quad (4)$$

式中:$j \in \{1,2,3,\cdots,m\}$, j 为第 i 层时所对应选择的边;W_{ij} 表示费用;T_{ij} 表示时间.用户对程序最终完成的时间和费用存在某个边界要求,本文的调度方案要使程序在用户规定的时间内完成,并且所耗费用不能超过用户的最大预算.

本文将问题的解空间定义成一棵 n 层 m 叉树.第 i 层第 j 条边表示任务 X_i 在 CPU S_j 上运行.这样从根节点到一个叶节点的一条路径就对应着一个调度方案.接着问题就转换为 n 层 m 叉树的遍历问题.本文的目的就是找到一条具有最小 *cost* 值的路径,该路径就是本文要采用的优化调度方案.

2　时间-费用任务调度模型的优化剪枝算法

为了减小时间复杂度,提高搜索效率,使用以下技术来避免无效搜索:①用约束函数在扩展节点处剪去不满足约束的子树;②用界限函数剪去不能得到最优解的子树.本文定义了 A、B、C、D 四个剪枝函数.

(1)剪枝函数 A

由于每个任务必须指派给一个计算机来处理,任两个任务都不在同一个CPU上处理,那么当搜索到第 i 层时,如果在第1到 $i-1$ 层已经搜索过第 j 条边,那么第 i 层的第 j 条边及其子树都可以剪掉,即不需要再向下进行搜索.

(2)剪枝函数 B

用户对程序最终完成的时间存在边界要求,即用户有权设定运行时间的最大值 T_0.本文中的模型进行调度时必须使用户的要求得到满足,如果不存在满足要求的方案,用户将被告知他的要求无法满足,任务调度将被放弃.因此定义剪枝函数 B.当搜索到第 i 层第 j 条边时,如果它的权值 $T_{ij} > T_0$,那么这条边及其子树都可以剪掉.

(3)剪枝函数 C

用户对程序最终完成的CPU耗费存在边界要求,即用户有权设定运行的最大CPU耗费 W_0.因此当搜索到第 i 层第 j 条边时,如果

$$M_i \cdot W_{ij} > W_0 \quad (5)$$

$j \in \{1,2,3,\cdots,m\}$,j 为第 i 层时所对应选择的边,即还未搜索到叶节点就已经超出了用户的费用界限,那么这条边及其子树都可以剪掉.

(4)剪枝函数 D

存储已经搜索过的最优路径的耗费,如果另一条路径从根节点到中间节点的耗费已经大于所存储的路径的最小耗费,那么该节点对应的边及其子树也可以剪掉.

采用递归回溯法对解空间树进行深度优先搜索的算法如下:

(1) Pruning function

```
Pruning A:
when reach Layer(i) then
  if Path(j) had been searched when search
    Layer(1) to Layer(i-1) then
      Cut Path(j) and its children
      return false
  else return true
  end if
Pruning B:
  when reach Path(j) in Layer(i) then
    if (T_ij > T) then
      Cut Path(j) and its children
      return false
    else return true
    end if
```

```
Pruning C:
  when reach Path(j) in Layer(i) then
    if (M_i · W_ij > W_0) then
      Cut Path(j) and its children
      return false
    else return true
    end if
Pruning D:
  when reach Path(j) in Layer(i) then
    if (cost >= best cost) then
      Cut Path(j) and its children
      return false
    else return true
    end if
```

(2) Traversal algorithm

```
Backtrack(int i):
if reach leaf node (i>n) then
  if cost<best cost then
    store this full path to best[] as best schedule path
    best cost= cost
  end if
    Search next path
end if
if reach middle node then
  Get the max T of this path
  if Pruning A, Pruning B, Pruning C, Pruning D all
return true then
T=max T
    W=W+W_ij
    cost= cost+ M_i W_ij +(T-old best T)
    Backtrack(i+1)
  end if
end if
```

(3) Output algorithm

```
for each TASK_i do
{
  schedule TASK_i to best_i CPU
}
```

3 算法复杂度分析

在传统的 n 层 m 叉树搜索中，需要搜索 m^n 条路径. 本算法中，在最好的情况下只需搜索一条路径，就可得到最优解；即使在最差情况下，由于有剪枝函数 A，也只需搜索 $m\times(m-1)\times(m-2)\times(m-3)\times\cdots\times(m-n+1)$ 条路径，搜索路径小于等于 $m!$. 因此本算法优于传统的算法.

本算法具有可以容忍的时间复杂度. 剪枝函数 A 的时间复杂度为 $O(i)$，剪枝函数 B、C、D 的时间复杂度显然为常数 $O(1)$，所以进行剪枝时并不浪费时间. 在最好的情况下本算法的时间复杂度为 $O(n^2)$，即只搜索一条路径其余的都被剪掉，剪枝的时间为

$$\begin{aligned}O(n^2) = {} & O(1+2+3+\cdots+n-1)+\\ & O(m-1+m-2+m-3+\cdots+\\ & m-n)+O(n)\end{aligned}\tag{6}$$

其中 $O(1+2+3+\cdots+n-1)$ 是用剪枝函数 A 剪掉树枝的时间复杂度，$O(m-1+m-2+m-3+\cdots+m-n)$ 是用剪枝函数 B、C、D 剪掉树枝的时间复杂度，$O(n)$ 是遍历剩余的那条路径的时间. 最好情况下的路径如图 1 所示，在图中细线表示由剪枝函数 A 剪掉树枝及其子树，粗线表示最佳搜索路径，虚线表示由剪枝函数 B、C、D 剪掉树枝及其子树.

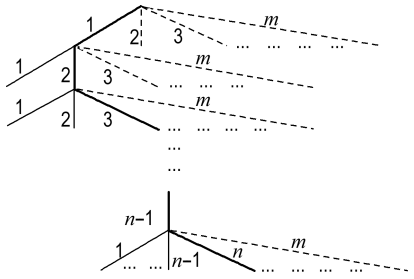

图 1 最好情况下的搜索路径

Fig. 1 The search path under the best condition

在最坏的情况下，本算法的时间复杂度为

$$\begin{aligned}&O(m^2+2(m-1)^3+\cdots+\\ &(n-1)(m-n)^n)+O(mn!)\ll O(m^n)\end{aligned}\tag{7}$$

其中 $O(m^2+2(m-1)^3+\cdots+(n-1)(m-n)^n)$ 是用剪枝函数 A 剪掉树枝的时间复杂度，$O(mn!)$ 是遍历剩余 n 条路径的时间复杂度. 所以从复杂度分析上可看出剪枝后的算法明显优于剪枝前的算法.

4 实验结果与讨论

4.1 实验目的与环境设置

实验目的是为了将本文提出的算法与传统算法进行比较、评价.

实验环境是用 JAVA 语言开发了一个模拟器. 输入数据如下：时间矩阵 $\boldsymbol{T}_{mn}$ 是随机生成的 $n\times m$ 个 1～100 的整数，费用矩阵 $\boldsymbol{W}_{mn}$ 是随机生成的 $n\times m$ 个 1～100 的整数，权重比矩阵 $\boldsymbol{M}_n$ 是随机生成的 n 个 1～10 的整数或其倒数，时间上限 T_0 为随机生成的 1～100 的一个整数，为避免

CPU耗费上限过大或过小，W_0采用的是$n\times100$到$n\times100+100$之间的随机生成的整数.

4.2 结果与讨论

4.2.1 *任务数n和CPU数m的变化对调度时间的影响* 实验(1)是在m、n均匀变化，$\mathbf{W}_{mn}$、$\mathbf{T}_{mn}$、$\mathbf{M}_n$随机生成的情况下进行的，结果如图2所示.

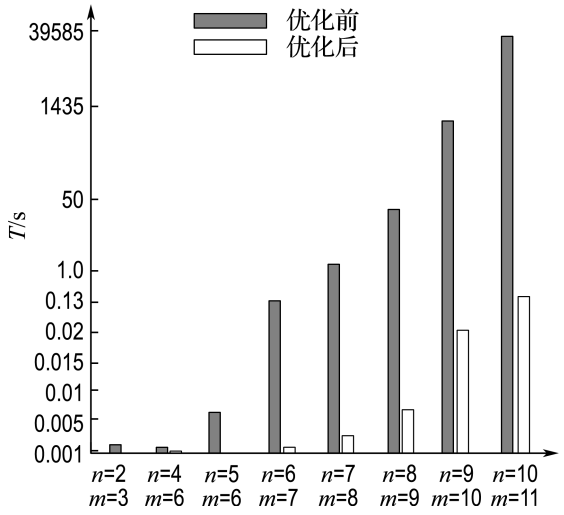

图2　任务数和CPU数的变化对调度时间的影响

Fig. 2　Effect of scheduling time with variety of CPU and task number

从图2可以看出在n、m都很小的情况下，优化前的时间和优化后的时间对比并不明显，这是因为n、m都很小时，路径很少，能剪掉的枝叶也就很少，所以效果并不明显，但随着n、m个数的增加，路径的增加，剪掉的枝会越来越多，效果会越来越明显，优化后的剪枝算法比优化前的未剪枝算法大幅度节省时间.

4.2.2 *根据用户对时间和费用的偏好不同来进行调度* 实验(2)假设网格中有4个独立的任务和6个CPU，每个任务在每个CPU上所需的运行时间如表1所示. 每个任务在每个CPU上运行所需花费的费用如表2所示.

表1　每个任务在每个CPU上所需的运行时间

Tab. 1　Task execution time on different CPUs

任务	时间/s					
	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6
Task1	5	2	4	1	8	6
Task2	13	3	6	7	2	9
Task3	8	3	4	7	11	9
Task4	5	9	15	2	1	6

表2　每个任务在每个CPU上所需的费用

Tab. 2　Task execution cost on different CPUs

任务	费用					
	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6
Task1	13	26	6	30	17	1
Task2	7	9	23	35	27	11
Task3	8	10	12	2	24	6
Task4	33	7	11	5	19	31

假设时间上限为12，费用上限为40，本文设用户有3种调度需求偏好，第1种认为费用比时间重要，第2种认为时间和费用同等重要，第3种认为时间比费用重要，其权值M_i，即所需支付运算费用权重与运行时间权重之比，如表3所示.

剪枝后的算法给出3种调度情况实验结果，如表4所示.

表3　各个用户的权值

Tab. 3　Weights of different users

用户	权值			
	Task1	Task2	Task3	Task4
User1	2	1	4	3
User2	1	1	1	1
User3	1/2	1	1/4	1/3

表4　剪枝后的算法给出3种调度情况结果

Tab. 4　Scheduling results of algorithms with different user requests

用户	Task1	Task2	Task3	Task4	T/s	*cost*
User1	CPU3	CPU6	CPU4	CPU2	9	26
User2	CPU6	CPU2	CPU1	CPU4	8	23
User3	CPU6	CPU2	CPU3	CPU4	6	27

从表4可看出，剪枝后的算法可根据用户的需求偏好将用户的任务映射到合适的CPU上. User1偏好费用，所以其牺牲了CPU时间换来了费用；User3偏好时间，所以他节省了时间而消耗了CPU费用；而User2认为任务执行时间与所耗费用同等重要，所以选择的是一种比较均衡的值. 系统需要满足各种需求偏好之间的一个平衡，只有在总耗费最小的情况下，才是最优解. 3种调度结果都是在当前权重下，总耗费最小的解.

4.2.3 *优化前后的性能比较* 实验(3)是在7个任务8个CPU和9个任务10个CPU的环境下对不同情况下优化前后的调度时间进行的比较结果，如表5所示(时间单位为s).

表5所指的最好情况、一般情况、最差情况是根据优化后的情况进行的分类，而优化后的最好情况，恰恰是优化前的最坏情况，这是因为，本文

的算法在最好的情况下只需搜索一条路径，就可得到最优解，所以需要时间很少，而在这种情况下优化前的算法在搜索完这条路径之后，对剩下的路径都要进行一定的处理，所以略费时间.

表5 不同情况下的性能比较

Tab. 5 Comparison of performance under different conditions

情况	7个任务8个CPU		9个任务10个CPU	
	优化前	优化后	优化前	优化后
最好情况	1.639	0.001	955	0.002
一般情况	1.613	0.004	945	0.826
最坏情况	1.590	0.179	917	21.400

本文的算法在最差情况下要搜索除剪枝函数A以外的所有路径，搜索量大大增加，于是时间也大大增加，而优化前的算法，直到最后的路径才进行处理，所以耗费时间较少.可以看出，随着问题规模的扩大，最好情况下和最坏情况下的对比也会愈加明显，而优化前的算法在不同情况下性能并没有明显的变化.

5 结 论

(1) 针对资源共享的经济网格环境中用户的QoS(预算和时限)要求的调度方案问题，本文提出将上述问题形式化为搜索 n 层 m 叉树的遍历路径上 $cost$ 值最小的问题.

(2) 通过引进树的剪枝思想，提出了基于剪枝的时间-费用优化的任务调度算法，该剪枝算法使用2个关键技术来避免无效搜索：用约束函数在扩展节点处剪去不满足约束的子树；用界限函数剪去不能得到最优解的子树.

(3) 采用模拟实验评估了基于剪枝策略的时间-费用优化任务调度算法的性能并验证了该算法相对于未剪枝算法的优越性，即可以有效地提高任务调度效率.

参考文献：

[1] YANG Guang-wen, JIN Hai, LI Ming-lu, *et al*. Grid computing in China [J]. **Journal of Grid Computing**, 2004, **2**(2): 193-206

[2] FOSTER I. A new infrastructure for 21st century science [J]. **Physics Today**, 2002, **55**(2): 42-47

[3] FOX F, GANNON D. Computational grids [J]. **IEEE Computational Science and Engineering**, 2001, **3**(4): 74-77

[4] BERTEN V, GOOSSENS J, JEANNOT E. On the distribution of sequential jobs in random brokering for heterogeneous computational grids [J]. **IEEE Transactions on Parallel and Distributed Systems**, 2006, **17**(2): 113-124

[5] JOHNSTON W E. Computational and data grids in large-scale science and engineering [J]. **Future Generation Computer Systems**, 2002, **18**(8): 1085-1100

[6] OLESZKIEWICZ J, LI X, LIU Y H. Effectively utilizing global cluster memory for large data-intensive parallel programs [J]. **IEEE Transactions on Parallel and Distributed Systems**, 2006, **17**(1): 66-77

[7] BERMAN F. From TeraGrid to knowledge grid [J]. **Communications of the ACM**, 2001, **44**(11): 27-28

[8] ZHUGE H. China's e-science knowledge grid environment [J]. **IEEE Intelligent Systems**, 2004, **19**(1): 13-17

[9] BERNERS-LEE T, HENDLER J, LASSILA O. The Semantic web [J]. **Scientific American**, 2001, **284**(5): 34-43

[10] FOSTER I. Service-oriented science [J]. **Science**, 2005, **308**(5723): 814-817

[11] HEY T, TREFETHEN A E. Cyber infrastructure for e-science [J]. **Science**, 2005, **308**(5723): 817-821

[12] KWIAT K. Using markets to engineer resource management for the information grid [J]. **Information Systems Frontiers**, 2002, **4**(1): 55-62

[13] GOMOLUCH J, SCHROEDER M. Performance evaluation of market-based resource allocation for grid computing [J]. **Concurrency and Computation: Practice & Experience**, 2004, **16**(5): 469-475

[14] ZHU Y, HU Y. Efficient proximity-aware load balancing for DHT-based P2P systems [J]. **IEEE Transactions on Parallel and Distributed Systems**, 2005, **16**(4): 349-361

[15] FOSTER I, KESSELMAN C. Globus: a metacomputing infrastructure toolkit [J]. **International Journal of Supercomputer Applications**, 1997, **11**(2): 115-128

[16] KARONIS N, TOONEN B, FOSTER I. MPICH-G2: a grid-enabled implementation of the message passing interface [J]. **Journal of Parallel and Distributed Computing**, 2003, **63**(5): 551-563

[17] BUYYA R, MURSHED M. GridSim: a toolkit for modeling and simulation of grid resource management and scheduling [J]. **Concurrency and Computation: Practice and Experience**, 2002, **14**(13-15): 1175-1220

[18] JIN Hai. Challenges of grid computing [C] // **Advances in Web-Age Information Management: 6th International Conference, LNCS 3739**. Hangzhou: Springer-Verlag, 2005

[19] KRAUTER K, BUYYA R, MAHESWARAN M. A taxonomy and survey of grid resource management system [J]. **Software: Practice and Experience**,

2002，**32**(2)：135-164
[20] 李志洁，程春田，黄飞雪，等. 一种基于序贯博弈的网格资源分配策略 [J]. 软件学报，2006，**17**(11)：2373-2383
[21] BONABEAU E，DORIGO M，THERAULAZ G. Inspiration for optimization from social insect behaviour [J]. **Nature**，2000，**406**(6791)：39-42
[22] LIAN Zhi-gang，GU Xing-sheng，JIAO Bin. A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan [J]. **Applied Mathematics and Computation**，2006，**175**(1)：773-785
[23] BOERINGER D W，WERNER D H. Particle swarm optimization versus genetic algorithms for phased array synthesis [J]. **IEEE Transactions on Antennas and Propagation**，2004，**52**(3)：771-779
[24] DORIGO M，BONABEAU E，THERALULAZ G. Ant algorithms and stigmergy [J]. **Future Generation Computer Systems**，2000，**16**(8)：851-871
[25] CHANDRASEKHARAN R，HANS Z. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs [J]. **European Journal of Operational Research**，2000，**155**(2)：426-438
[26] BRAUN T D，SIEGEL H J，BECK N. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems [J]. **Journal of Parallel and Distributed Computing**，2001，**61**(6)：810-837
[27] BOERINGER D W，WERNER D H. Particle swarm optimization versus genetic algorithms for phased array synthesis [J]. **IEEE Transactions on Antennas and Propagation**，2004，**52**(3)：771-779
[28] GAING Zwe-lee. Particle swarm optimization to solving the economic dispatch considering the generator constraints [J]. **IEEE Transactions on Power Systems**，2003，**18**(3)：1187-1195.
[29] BUYYA R，ABRAMSON D，GIDDY J S，*et al*. Economic models for resource management and scheduling in grid computing [J]. **Concurrency and Computation：Practice & Experience**，2002，**14**(13-15)：1507-1542
[30] WOLSKI R，PLANK J S，BREVIK J，*et al*. Analyzing market-based resource allocation strategies for the computational grid [J]. **International Journal of High Performance Computing Applications**，2001，**15**(3)：258-281
[31] ABRAMSON D，BUYYA R，GIDDY J. A computational economy for grid computing and its implementation in the nimrod-g resource broker [J]. **Future Generation Computer Systems**，2002，**18**(8)：1061-1074

Task scheduling based on cost-time optimization of pruning strategy in economic grid

HUANG Fei-xue[*1]，JIANG Xin-na[2]，LI Zhi-jie[3]，HOU Tie-shan[1]

（1. Department of Economics，Dalian University of Technology，Dalian 116024，China；
2. School of Software，Dalian University of Technology，Dalian 116024，China；
3. School of Computer Science and Engineering，Dalian Nationalities University，Dalian 116600，China）

Abstract：Grid task scheduling is very complex issue due to the heterogeneity and distributed nature of economic grid environment. To solve independent tasks scheduling under user quality of service economic demand preference，a scenario selection algorithm for task scheduling is proposed. The task scheduling scenarios are formulated as *n*-level and *m*-ary tree. Then，the scenario selection for task scheduling is translated into a traversal problem of tree. Next，invalid path searching is avoided using pruning method. Finally，a simulative example is also provided to analyze the characteristics of the algorithm. The results show that this algorithm can allocate tasks to appropriate CPU resources based on the user′s requirement of deadline and budget constraint，and it has a low space complexity and a tolerable time complexity，which outperforms the conventional scheduling algorithm. The conclusions indicate that it is a feasible task scheduling algorithm.

Key words：cost constraint；task scheduling；pruning；deadline；grid computing