

## 自适应邻域的多目标网格任务调度算法

杨明<sup>1,2\*</sup>, 薛胜军<sup>2</sup>, 陈亮<sup>1</sup>, 刘永生<sup>1</sup>

(1. 浙江省气象信息网络中心, 杭州 310017; 2. 南京信息工程大学 计算机与软件学院, 南京 210044)

(\* 通信作者电子邮箱 yangmingstudy@126.com)

**摘要:** 针对网格计算中的多目标网格任务调度问题, 提出了一种基于自适应邻域的多目标网格任务调度算法。该算法通过求解多个网格任务调度目标函数的非劣解集, 采用自适应邻域的方法来保持网格任务调度多目标解集的分布性, 尝试解决网格任务调度中多目标协同优化问题。实验结果证明, 该算法能够有效地平衡时间维度和费用维度目标, 提高了资源的利用率和任务的执行效率, 与 Min-min 和 Max-min 算法相比具有较好的性能。

**关键词:** 网格任务调度算法; 多目标进化算法; 自适应邻域; 任务调度

**中图分类号:** TP301.6; TP393.028; TP18 **文献标志码:** A

### Multi-objective evolutionary algorithm for grid job scheduling based on adaptive neighborhood

YANG Ming<sup>1,2\*</sup>, XUE Sheng-jun<sup>2</sup>, CHEN Liang<sup>1</sup>, LIU Yong-sheng<sup>1</sup>

(1. Zhejiang Meteorological Information Network Center, Hangzhou Zhejiang 310017, China;

2. College of Computer and Software, Nanjing University of Information Science and Technology, Nanjing Jiangsu 210044, China)

**Abstract:** A new adaptive neighborhood Multi-Objective Grid Task Scheduling Algorithm (ANMO-GTSA) was proposed in this paper for the multi-objective job scheduling collaborative optimization problem in grid computing. In the ANMO-GTSA, an adaptive neighborhood method was applied to find the non-inferior set of solutions and maintain the diversity of the multi-objective job scheduling population. The experimental results indicate that the algorithm proposed in this paper can not only balance the multi-objective job scheduling, but also improve the resource utilization and efficiency of task execution. Moreover, the proposed algorithm can achieve better performance on time-dimension and cost-dimension than the traditional Min-min and Max-min algorithms.

**Key words:** grid job scheduling algorithm; multi-objective evolutionary algorithm; adaptive neighborhood; job scheduling

## 0 引言

在资源异构高效的网格计算中, 网格环境涉及网格用户、网格资源管理者、虚拟组织管理者等多个实体, 不同实体间对管理机制、安全策略、调度时间和费用等目标都不同, 有的甚至相互冲突。因此, 如何通过资源管理与调度算法协调这些不同实体间和同类实体内部的不同目标要求是当前网格计算中的热点问题<sup>[1]</sup>。

近年来, 国内外研究人员提出了不少高效的网格任务调度算法, 其中比较典型的有 Min-min、Max-min 和 Sufferage 等<sup>[2-5]</sup>, 这些调度算法主要以任务完成时间作为单一调度目标。针对多目标任务的调度算法有 Min-min 启发式调度算法<sup>[6]</sup>和多目标冲突度遗传算法<sup>[7]</sup>等, 这些算法主要采用权重法将多个目标函数聚合成单一的目标函数, 进行单目标求解。权重法具有时间复杂度低、便于实现的特点。但是, 这些算法主要通过调整权重来平衡多个目标, 而权重的取值具有随意性, 影响了调度算法的性能, 这些算法自身也存在一些固有的缺陷, 很难满足网格环境用户同时对多个目标的需求。另外, 单目标优化算法仅能根据聚合函数来得到决策空间中一个可行解, 降低了最终解的质量, 缺乏灵活性与可扩展性。而多目

标优化问题的解并非唯一的, 它存在一个最优解集或非支配解集 (Non-Dominated solutions Set, NDSets), 集合中元素称为 Pareto 最优解或非劣最优解<sup>[8-9]</sup>。因此, 更为合理的途径是采用多目标最优化算法来解决多目标网格调度问题。

鉴于此, 本文通过网格任务调度的多个效用函数指标, 运用多目标最优化理论及其算法来解决网格任务调度中多目标协同问题, 提出了一种基于自适应邻域的多目标网格任务调度算法 (A new adaptive Neighborhood Multi-Objective Grid Task Scheduling Algorithm, ANMO-GTSA)。该算法采用一种新的基于自适应邻域修剪策略, 不仅可以获得多个相互冲突的调度目标的最优解, 还可以改善负载均衡的问题。在通常情况下, 最短任务完成时间目标和最低费用目标在一般情况下是冲突的, 无法在这两个目标上同时获得最优的调度结果, 通过该算法可以平衡这两个目标, 使得计算节点的分配更均匀, 并且将其与传统算法 Min-min 和 Max-min 进行了对比实验, 实验结果验证了新算法的优越性。

## 1 调度模型和问题定义

从用户的角度来说, 总是希望分配到的资源能够用最少的费用、最小的时间完成所有提交的任务; 而从资源提供者角

收稿日期: 2011-08-24; 修回日期: 2011-11-18。

**作者简介:** 杨明 (1983-), 男, 江苏泗阳人, 助理工程师, 主要研究方向: 网格计算、进化算法; 薛胜军 (1956-), 男, 山东青岛人, 教授, 博士生导师, 主要研究方向: 人工智能、计算机网络、高性能计算; 陈亮 (1981-), 男, 江苏泰兴人, 工程师, 主要研究方向: 高性能计算、计算机网络; 刘永生 (1981-), 男, 江苏宿迁人, 工程师, 主要研究方向: 网格计算。

度看,完成时间越少、资源性能越高,定价费用自然也会越高。资源的可用性越高,它完成单位计算量的费用越高;请求资源的用户数量越多,资源的价格越高。资源的价格和任务完成时间对网格用户的成本起决定作用,都是评估模型中重要的指标。

网络任务调度相关定义如下:

- 1) 用集合  $R = \{r_1, r_2, \dots, r_m\}$  表示  $m$  个计算资源,其中  $r_j$  表示第  $j$  个资源。
- 2) 用集合  $T = \{t_1, t_2, \dots, t_n\}$  表示  $n$  个任务,其中  $t_i$  表示第  $i$  个任务。
- 3) 任务的执行时间用  $m \times n$  阶矩阵  $ET$  表示; $ET_{ij}$  表示任务  $t_i$  在处理节点  $r_j$  上的预期执行时间, $ET_{ij} = \infty$  表示任务  $t_i$  不能在处理节点  $r_j$  上执行。

下面给出网络任务调度方面的最优跨度和费用效能这两个问题的描述。

**最优跨度问题** 假设不考虑计算任务的跨节点执行,用户向系统提交的任务就是网络调度单元进行任务分配的最小单位,大型计算任务的分解由用户来完成,并且任务之间是独立的。

任务调度问题是将  $n$  个任务集  $T$  以合理的方式调度到  $m$  个处理节点集  $R$  上执行,以尽可能地达到预定目标。任务调度的结果用  $n \times m$  的二维矩阵  $W$  来表示,当  $w_{ij} = 1 (w_{ij} \in W)$  表示将任务  $t_i$  调度到处理节点  $r_j$  上执行;否则  $w_{ij} = 0$ 。

为了便于描述,定义以下参数和符号: $makespan$  表示最优跨度; $b_j$  表示处理节点  $r_j$  的最早可用时间; $CT_{ij}$  表示任务  $t_i$  在处理节点  $r_j$  上的预期完成时间。

一般有  $CT_{ij} = b_j + ET_{ij}$ 。当任务  $t_i$  被调度到处理节点  $r_j$  上执行时,令  $CT_i = CT_{ij}$ ,表示任务  $t_i$  执行完成的时间,则任务集  $T$  的执行完成时间  $makespan(T) = \max_{i \in T} (CT_i)$ ,调度目标是要获得尽量小的  $makespan$ ,即  $\min(makespan)$ 。

**费用效能问题** 当需要缩短运行时间时,需要牺牲更多的 CPU 使用费用来补偿,即处理能力越高的 CPU,其使用费用通常越高。因此,如何能使 CPU 的使用费用和运行时间同时达到最小值,是一个复杂的问题。

假设  $Cost$  表示应该执行完所有任务所需的总费用; $u_j$  表示资源  $r_j$  在单位时间内的价格; $C_{ij}$  表示任务  $t_i$  在资源  $r_j$  上的费用,则任务  $t_i$  的执行费用  $C_{ij}$  和总费用  $Cost$  可表示为:

$$C_{ij} = ET_{ij} \times u_j \quad (1)$$

$$Cost = \sum_{i=1}^n \sum_{j=1}^m C_{ij} \quad (2)$$

则调度求解目标是尽量小的  $Cost$ ,即  $\min(Cost)$ 。

基于上面提出的最优跨度和费用效能两个问题的模型,多目标网络任务调度的总目标函数可以表示为:

$$\min f(makespan, Cost) \quad (3)$$

式(3)表示任务调度的目标是在执行完所有任务后,取得尽量少的总执行时间和尽量小的总花费,使该两个目标同时达到最优。

## 2 自适应邻域多目标网络任务调度算法

### 2.1 自适应邻域策略

为了避免算法的分布性在进化中受到破坏的现象以及解

决邻域半径的取值问题,首先,应使半径变化率的自适应调整曲线在  $d_{avg}$  处缓慢改变,从而提高距离接近平均距离的个体的邻域半径。其次,当  $d_{avg}$  和  $d_{min}$  相差较大时,确保自适应调整曲线不会趋于直线型。最后,保证种群中较优个体仍具有一定的分布性。同时,为了尽可能地保留较好的分布性,应更平滑  $d_{avg}$  处的自适应调整曲线。如图 1 所示。

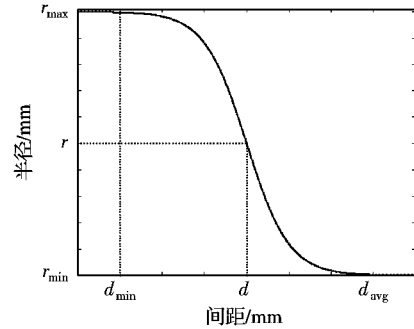


图1 自适应调整曲线

由于 sigmoid 函数在线性和非线性行为之间显现出较好的平衡<sup>[9]</sup>,所以采用 sigmoid 函数作为自适应调整曲线。求解邻域半径的自适应调整公式如式(4)所示。

$$r = \begin{cases} \frac{r_{\max} - r_{\min}}{1 + \exp\left(\frac{a(r - d_{\min})}{d_{\text{avg}} - d_{\min}}\right)} + r_{\min}, & d \geq d_{\min} \\ r_{\min}, & d < d_{\min} \end{cases} \quad (4)$$

其中: $a$  表示任意常数, $r$  表示当前代的邻域半径, $r_{\max}$  表示邻域半径的最大值, $r_{\min}$  表示邻域半径的最小值, $d$  表示前一代的邻域半径, $d_{\min}$  表示当前个体之间的最小间距, $d_{\text{avg}}$  表示当前个体之间间距的平均值。

分析式(4)可知,邻域半径的变化率按照个体之间的距离在最小距离和平均距离之间随 sigmoid 曲线进行非线性调整。显然,当种群中的大部分个体拥有相近的距离且最小距离与平均距离相接近时,种群的分布性得到了改善。

### 2.2 染色体编码

传统的多目标算法染色体编码形式是将决策空间备选方案编码成一个 0 和 1 的串。对于网络任务调度问题,采用这个编码方式比较困难。因为编码方式既要表示任务的分配,又要表示其计算资源对任务的调度信息,采用二进制编码的方式很难包含映射的全部信息。

本文采用实数编码方式来对染色体进行编码。如图 2 所示,将染色体(即调度  $x_i$  方案)编码成多维矩阵,表中的每项代表一个资源  $r_j$  和分配在其上的任务  $t_i$ 。同时,一个染色体  $x_h$  还记录了任务集在资源队列中执行顺序。所有染色体的集合  $X = \{x_1, x_2, \dots, x_P\}$  即为一个种群, $P$  表示群体规模。

	$r_1$		$r_2$		$\dots$		$r_j$		$\dots$		$r_m$	
$x_h$	$t_{11}^1$	$t_{12}^2$	$t_{21}^3$	$\dots$	$t_{j1}^i$	$\dots$	$t_{jk}^i$	$\dots$	$t_{m1}^n$	$\dots$	$t_{mn}^n$	

图2 网络任务调度染色体编码结构

图 2 中: $x_h$  表示第  $h$  个染色体; $r_j$  表示第  $j$  个网络资源; $t_{jk}^i$  表示分配到第  $j$  个网络资源  $r_j$  上的第  $k$  个任务的编号; $i$  表示第  $i$  位染色体; $m$  表示计算资源数; $n$  表示染色体的长度。

### 2.3 ANMO-GTSA 算法描述

初始种群由  $P$  个随机生成的调度方案组成。种群随机生成调度方案如算法 1 所示。

算法1 初始化种群算法。

输入  $R, T, m, n, P$ 。

输出  $X$ 。

此过程中的参数  $P$  表示种群的大小,  $n$  表示任务数,  $m$  表示网格资源数,  $X$  表示调度方案集合,  $R$  表示资源集合,  $T$  表示任务集合,  $r_{\text{random}}$  表示随机产生的资源节点。

```

Begin
  For  $i = 1$  to  $P$ 
    For  $j = 1$  to  $n$ 
      随机选择一个任务  $t_i$  分配到网格资源  $r_{\text{random}}$  上 (其中  $r_{\text{random}} \in R, t_i \in T$ );
    End for
    产生一个任务调度方案  $x_i$ ;
  End for
  得到一个初始种群  $X$ ;
End

```

算法2 计算个体的分布适应度。

输入  $Q(t)$ 。

输出 归档集中个体的密度值、距离值和邻域关系矩阵。

此过程中的参数  $Q(t)$  表示当前的归档集,  $|Q[i], Q[j]|$  表示个体  $i$  与个体  $j$  之间的间距,  $r$  表示邻域半径的值,  $n$  表示当前归档集的个体数。

```

Begin
  统计当前归档集的个体数;
  For  $i = 1$  to  $n$ 
    初始化个体的密度变量和距离变量;
    For  $j = 1$  to  $n$ 
      If  $(i \neq j, \text{ and. } |Q[i], Q[j]| < r)$  then
        统计当前个体的邻域密度;
        统计当前个体到其邻域内其他个体的距离;
        记录各个体之间的邻域关系;
      End if
    End for
  End for
  返回归档集中个体的密度值、距离值和邻域关系值;
End

```

算法3 种群维护。

输入  $Q(t)$ 。

输出  $Q(t)$ 。

此过程中的参数  $Q(t)$  表示当前的归档集,  $\text{delete-index}$  表示将被删除的个体序号的集合,  $n$  表示当前归档集的个体数。

```

Begin
  统计当前归档集的个体数;
   $\text{delete-index} = \emptyset$ ;
  While 当非支配个体数大于归档集时 do
    按个体的邻域密度进行降序排序;
    If 前两个个体密度不同 then
      密度大的个体并入  $\text{delete-index}$  中;
    Else if 前两个个体密度相同, and. 前两个个体到其邻域其他个体的距离不同 then
      距离小的个体并入  $\text{delete-index}$  中;
    End if
  End while
  For  $i = 1$  to  $n$ 
    If  $\text{delete-index}$  中个体与个体  $i$  存在邻域关系 then
      个体  $i$  邻域密度减1;
      个体  $i$  邻域距离减去  $\text{delete-index}$  中个体到个体  $i$  的距离;
    End if
  End for

```

```

End if
End for
End while
删除  $Q(t)$  对应  $\text{delete-index}$  集合中的所有个体;
End

```

## 2.4 ANMO-GTSA 算法流程

ANMO-GTSA 的算法流程如下:

第1步 任务调度编码。根据2.2节的任务调度编码方案,采用实数编码方式,取值的上、下限由资源数和任务数确定。

第2步 产生初始任务调度种群。随机生成一个初始任务调度群体  $X(g)$ , 同时将任务调度归档集  $Q(g)$  置空, 并设置进化代数  $g = 0$ , 其中  $X(g)$  和  $Q(g)$  规模等于任务数  $N$ 。

第3步 进化操作。使用二元锦标赛法则选择  $X(g)$  中的资源个体进入交配池, 然后对交配池中资源个体执行交叉和变异操作, 产生的新资源个体赋给  $Q(g)$ 。

第4步 适应度分配。采用  $R(g) = X(g) \cup Q(g)$ , 其中  $R(g)$  规模是任务数的2倍, 即  $2N$ , 然后利用式(3)定义的目标函数, 评价  $R(g)$  中所有个体的适应度。

第5步 精英策略。将  $R(g)$  中的所有非支配个体保存到非支配解集  $NDS\text{Set}$  中, 若  $|NDS\text{Set}| > N$ , 则利用算法3进行修剪操作, 降低其大小, 直到其规模等于任务数  $N$ ; 若  $|NDS\text{Set}| < N$ , 则从  $R(g)$  中选取支配个体填满, 直到其规模等于任务数  $N$ 。

第6步 结束条件。以进化代数作为终止条件, 若满足终止条件, 则将  $NDS\text{Set}$  中的所有非支配个体作为最终的输出群体; 否则,  $NDS\text{Set}$  保存到  $X(g+1)$  中, 令  $g = g+1$ , 转第3步。

## 3 实验分析

为了检验所提自适应邻域多目标网格任务调度算法模型的性能, 首先讨论了实验的参数设置, 然后通过与其他网格任务调度算法, 如 Max-min 和 Min-min 算法进行了比较分析, 表明了 ANMO-GTSA 算法的优越性。

### 3.1 实验参数设置

在网格环境中, 由于任务到达时间的随机性以及计算资源的动态性, 所以网格环境中一般需要动态调度算法<sup>[10]</sup>。本节在批处理模式下分别针对不同的任务数和处理节点数进行了实验。模拟实验的参数设置如下:

1) ANMO-GTSA 算法采用实数编码, 交叉概率 0.85, 变异概率 0.01, 种群规模为 100, 进化代数为 300, 适应度评价次数为 30 000, 算法的运行代数为评价个体的数目除以种群规模。

2) 每个任务在各个处理节点上的执行时间满足均匀分布; 任务的到达时间服从泊松分布。

3) 处理节点的失效率在区间  $1 \times 10^{-4} \sim 1 \times 10^{-3}$  内均匀选择。

4) 在不同数目的任务集进行实验时, 处理节点数为 10, 任务数的范围是 100 ~ 1 000, 间隔为 100; 在不同数目的处理节点进行实验时, 任务数为 500, 处理节点数为 10 ~ 100, 间隔为 10。

每个实验都分别选择了不同任务集和计算节点集进行了 100 次, 将 100 次实验结果取平均得到最终的实验结果。



### 3.2 实验结果与性能分析

仿真实验分为两部分:1) 网络系统是由 10 个网格资源节点组成,任务数为 100~1000 的任务集合组成,间隔为 100; 2) 网络系统是由 500 个任务数组成,网格资源数为 10~100 的网格资源节点集合组成,间隔为 10。为了使实验数据具有稳定性,每组实验均进行多次,用其平均值进行性能分析。

图 3 为在不同任务数情况下,各种调度算法得到的执行时间和费用性能比较曲线图。图 4 为在不同资源节点数情况下,各种调度算法得到的执行时间和费用性能比较曲线图。时间维度定义为调度方案中所有的任务所获得的时间性能效用总和。从图 3(a)和图 4(a)可看出,ANMO-GTSA 时间性能效用优于 Max-min 和 Min-min 算法。费用维度定义调度方案中所有的任务所获得的费用性能效用总和。从图 3(b)和图 4(b)可看出,ANMO-GTSA 费用性能效用优于 Max-min 和 Min-min 算法。从实验结果数据可看出,由于 ANMO-GTSA 算法综合考虑了执行时间和费用性能效用,所以其执行时间和费用性能效用都优于 Max-min 和 Min-min 算法。

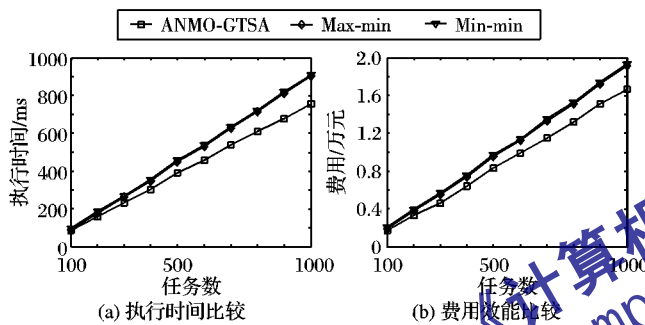


图 3 在不同任务数下的性能比较

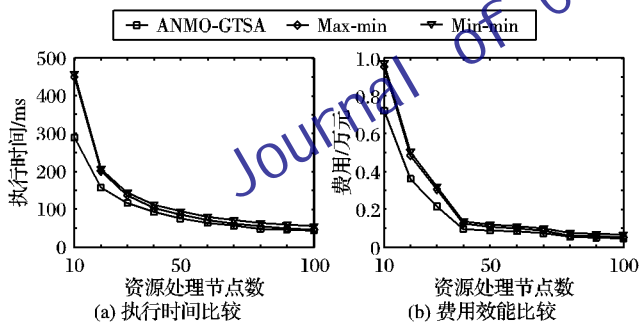


图 4 在不同网格资源节点数下的性能比较

为了分析算法在负载均衡方面的性能,从第一部分实验中,选择由 100 个任务数组成,网格资源节点数为 10 的网格资源节点集合组成的网络系统。图 5 显示了各种调度算法的映射能力。从图中可看出,ANMO-GTSA 算法在负载均衡性能方面比 Max-min 和 Min-min 算法有较大改善。主要由于以下原因:ANMO-GTSA 算法具有解决多个任务调度目标的最优解的能力,能在一定程度上平衡负载。而 Min-min 算法中每个资源的负载极不平衡,主要原因在于该算法将执行时间最短的任务分配在负载最小的资源处理节点上,这导致执行时间长的任务分配到负载较大的资源处理节点上。

综上所述,本文提出的 ANMO-GTSA 算法在不同的网络系统下都表现得较好,在处理执行时间和费用这两个任务调度目标时,能使这两个目标同时达到最优,验证了 ANMO-GTSA 算法的可行性和有效性。

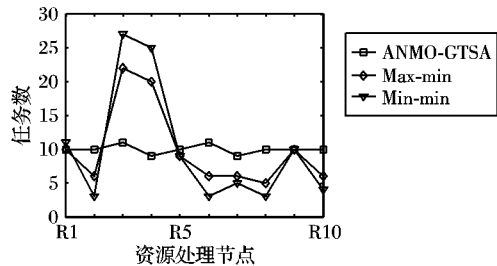


图 5 在 10 个网格资源节点和 100 个任务数下的映射图

## 4 结语

网络任务调度问题是网络计算中一个至关重要的问题,它的调度算法将直接影响到网络环境中任务执行的效率与结果。另外,传统的单目标任务调度方式无法有效地满足网络环境个体的多目标需求,必须有合理的管理模式和多目标调度算法才能充分地发挥网络的特性。

本文针对网络任务调度的特点提出了一个基于自适应邻域多目标网络任务调度模型;根据当代种群本身的情况,设计了一种基于自适应邻域多目标网络调度算法,该算法采用的自适应邻域的策略,改善了算法的性能;通过与 Max-min 和 Min-min 算法的对比实验表明,该算法在处理网络任务调度的多目标问题,表现出较优秀的效果,使用该算法处理网络任务调度问题,是一个有效的方法。

### 参考文献

- [1] 都志辉,陈渝,刘鹏. 网络计算[M]. 北京:清华大学出版社,2002:3-16.
- [2] MAHESWARAN M, ALI S, SIEGEL H J, *et al.* Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems [C]// HCW'99: Proceedings of the 8th Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 1999: 30-44.
- [3] MAHESWARAN M, ALI S, SIEGEL H J, *et al.* A comparison of dynamic strategies for mapping a class of independent tasks onto heterogeneous computing systems [R]. West Lafayette, Indiana: Purdue University, School of Electrical and Computer Engineering, 1999.
- [4] WU M Y, SHU W, ZHANG H. Segmented Min-Min: A static mapping algorithm for meta-tasks on heterogeneous computing systems [C]// Proceedings of the 9th IEEE Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 2000: 375-385.
- [5] CASANOVA H, LEGRAND A, ZAGORODNOV D, *et al.* Heuristics for scheduling parameter sweep applications in grid environments [C]// Proceedings of the 9th Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 2000: 349-363.
- [6] 王树鹏,云晓春,余翔湛. 基于生存性和 Makespan 的多目标网络任务调度算法研究[J]. 通信学报, 2006, 27(2): 43-45.
- [7] 乔付,张国印,刘忠艳. 基于多目标冲突度网络任务调度策略[J]. 计算机应用研究, 2009, 26(4): 1491-1493.
- [8] DEB K, PRATAP A, AGARWAL S, *et al.* A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.
- [9] 薛胜军,杨明. 一种新的自适应邻域的多目标进化算法[J]. 计算机工程与应用, 2011, 47(25): 49-53.
- [10] MAHESWARAN M, ALI S, SIEGEL H J, *et al.* Dynamic mapping of a class of independent tasks onto heterogeneous computing systems [J]. Journal of Parallel and Distributed Computing, 1999, 59(2): 107-131.