文章编号: 1001-0920 (2013) 06-0808-07

基于均匀离散PSO算法的多QoS网格任务调度策略

蒲 汛[1,2], 彭喜化[2], 于显平[2], 卢显良[1]

(1. 电子科技大学 计算机学院，成都 410073；2. 西南大学 计算机与信息科学学院，重庆 400716)

摘 要: 针对网格环境中多服务质量(QoS)约束条件下独立任务调度问题, 提出一种融合配方均匀设计与离散粒子群优化算法(UDPSO)的任务调度策略, 以实现对独立任务优化调度的快速生成. 该算法采用类似DPSO算法的速度和位置更新方法, 结合配方均匀设计, 快速衡量各QoS约束条件的适应度, 以产生分布均匀且较优的Pareto解集, 最终为系统提供一组较优的任务调度方案. 仿真实验表明, 该算法更符合网格调度的复杂环境, 能够得到较短的任务执行时间和较均衡的QoS保障.

关键词: 离散粒子群优化算法；Pareto最优；均匀设计；服务质量约束；任务分配

中图分类号: TP273 **文献标志码:** A

Jobs scheduling policy for gird with multi-QoS constraints using uniform-design discrete particle swarm optimization

PU Xun[1,2], *PENG Xi-hua*[2], *YU Xian-ping*[2], *LU Xian-liang*[1]

(1. College of Computer Science，University of Electronic Science and Technology of China，Chengdu 410073, China；2. College of Computer and Information Science，South-west University，Chongqing 400716，China. Correspondent：PU Xun，E-mail：puxun@swu.edu.cn)

Abstract: One of the key technologies to improve the efficiency of grid computing is to solve the independent job scheduling problem under the QoS constraints. Therefore, this paper designes an algorithm named uniform-design discrete particle swarm optimization(UDPSO) to find a sufficient number of uniformly distributed and representative Pareto optimal solution for the problem, which can make the performance of gird system to be more efficient. To solve the problem, the velocity and position of particles are refined, and a new method is used to optimize those two parameters. Then, the uniform design method is employed to get distributed Pareto front in the objective space efficiently. Finally, the global convergence of the algorithm is proved. Simulation results show that this algorithm is more effective and practical.

Key words: discrete particle swarm optimization；Pareto optimal；uniform design；quality of service constrains；job scheduling

0 引 言

网格计算[1]作为一种整合网络异构资源的系统, 在动态、多制度的虚拟组织中协调各种资源的共享, 以解决大规模挑战性计算问题. 尽管当前云计算[2]作为一种新兴的分布式服务模式被众多大型IT企业(IBM, Amazon, Google)所关注和研究, 但其所采用的系统架构、资源管理、程序设计模型等核心技术仍然与网格计算有千丝万缕的联系[3]. 因此, 对于网格计算的研究, 尤其是对网格环境中资源管理等核心技术的研究显得尤为重要.

网格环境涉及网格用户、虚拟组织管理者和网格资源管理者等多个实体, 不同实体间对资源分配策略、任务调度机制等用户服务质量(QoS)的目标都不尽相同, 甚至相互矛盾. 如资源提供商希望资源集群以可控的方式安全、高效地工作, 而网格用户希望以尽可能低的价格高效地完成相关任务. 因此在这种多用户QoS约束的网格环境中, 网格任务管理的实质是将任务分配到合适的资源上, 使得在满足用户各种QoS需求的前提下, 任务完成时间尽量小且资源利用率尽量高. 基于以上目的, 通常情况下对网格计算系统而言, 只要求尽快找出一个满足用户QoS条件的优化任务调度方案即可, 而并非是全局最优服务.

收稿日期: 2012-02-22；**修回日期:** 2012-11-29.

基金项目: 国家重大专项项目(2011ZX03002-003-02)；中央高校基本科研业务费专项项目(XDJK2012C020).

作者简介: 蒲汛(1977−), 男, 讲师, 博士, 从事计算机体系结构的研究；卢显良(1944−), 男, 教授, 博士生导师, 从事计算机网络、操作系统等研究.

针对网格环境中多QoS任务调度这一NP完全问题[4], 通常采用启发式算法来解决, 如遗传算法、蚁群算法和粒子群算法等. 所有的启发式算法均面临一个共同的问题, 即多用户QoS需求中如何衡量解的优劣. 目前常用的衡量方法一般分为两类[5]: 1) 将多个QoS的目标函数聚合成单一目标函数, 这类算法具有时间复杂度低、便于实现的特点, 如Max-min和Genitor-style GA等调度策略; 2) 采用多目标优化算法, 使用Pareto占优指导搜索, 并返回一个非占优解集作为任务调度结果, 以满足用户QoS需求. 在实际网格应用环境中, 由于用户QoS的需求各有不同, 网格的任务调度策略期望在可接受时间内计算出全部的Pareto最优解几乎不可行.

针对这一现象, 若调度策略能够快速产生一组均匀分布且数量合理的Pareto最优解的代表解, 将具有重要意义, 这也是算法用于实际网格环境的一个关键问题. 本文将网格环境中多QoS约束的任务调度问题规约为多目标组合最优化问题, 针对尽量保持非劣解多样性的问题提出了基于配方均匀设计的离散粒子群优化算法(UDPSO). 通过新定义的速度和位置计算公式, 结合配方均匀设计产生的权重值组合参数, 最终获得分布均匀的Pareto最优解集, 以协调资源负载均衡和用户的QoS要求之间的矛盾, 并最大化网格用户在多个QoS维度下的效用值.

1　多QoS参数约束的网格任务调度问题

1.1　网格任务调度模型

网格任务调度是指将所有等待队列中的任务分配到各服务节点, 并产生一个相对较优的执行序列, 它不但要满足用户的QoS需求, 还要尽量使系统资源达到负载均衡. 因此, 网格计算任务调度的目标是对用户提交的任务实现最优调度, 并设法提高网格系统的总体吞吐率.

本文所讨论的网格系统基于3个基本层次, 由上至下依次为: 应用层、中间层和资源层. 应用层作为用户界面, 是用户需求的具体体现, 在网格中间层的支持下, 网格用户可以使用应用层提供的工具进行各种应用开发; 资源层由各种物理资源构成, 是网格系统的硬件基础, 可分为处理资源和通信资源, 处理资源完成任务的处理, 通信资源负责信息的传送; 中间层包括一系列工具和协议软件, 主要是为了屏蔽资源层中计算资源的分布、异构等特性, 并向应用层提供透明、一致的使用接口, 为网格用户提供服务. 本文主要通过中间层来实现任务调度机制, 模型由上至下分为3层, 分别是任务管理层、映射层和资源管理层, 如图1所示.

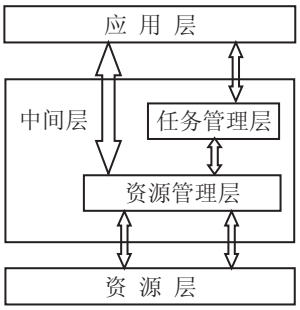


图1　分层任务调度模型

1.2　网格任务调度问题定义

为了减少问题的复杂性, 在不影响一般性的情况下, 针对如下网格环境进行研究: 1) 网格任务之间相互独立, 且各任务已分解为系统执行的最小单位; 2) 网格任务采用统一调度方式, 即在任务积累到一定数值时由任务调度服务进行分配, 或者在每次任务分配完成后, 开始调度新到的和执行失败的任务; 3) 各服务节点只能同时处理一个独立的网格任务, 且运算时间能够被估计. 基于以上假设, 提出如下符号定义.

定义1　集合

$$R=\{(r_1,c_1,a_1),(r_2,c_2,a_2),\cdots,(r_i,c_i,a_i)\}$$

表示i个网格服务节点. 其中: r_i为第i服务节点所提供的计算能力或存储能力; c_i为使用该节点单位时间内的花费; a_i为任务节点的可信度, 用来衡量节点的稳定性.

定义2　集合

$$T=\{(t_1,p_1),(t_2,p_2),\cdots,(t_t,p_t)\}$$

表示由t个独立作业构成的作业集合. 其中: t_t为作业t所需要的计算或存储能力; p_t为用户为作业所定义的优先级, 优先级越高的用户所需支付的费用越多.

定义3　集合$Q=\prod\limits_{1\leqslant k\leqslant d}Q_k$表示$d$维QoS约束空间, 每个作业$t_k$均有自己的QoS限制维度. 本文选择3个常用的QoS约束来描述用户需求, 分别为$Q_{MC}\leqslant$ Deadline(任务执行时间小于用户时限)、$Q_{CC}\leqslant$ Budget(任务总的花费小于用户预算)、$Q_{AC}\geqslant$ AvailabilityLevel(服务节点可信度应大于用户限定阈值).

定义4　集合$M=\{\mathrm{map}^1,\mathrm{map}^2,\cdots,\mathrm{map}^n\}$表示由算法产生的映射方案集合, 即结果集, 其中每个映射方案由下式表示:

$$\mathrm{map}^y=\begin{bmatrix}O^t_{1,1} & O^t_{1,2} & \cdots & O^t_{1,m}\\ O^t_{2,1} & O^t_{2,2} & \cdots & O^t_{2,m}\\ \vdots & \vdots & \ddots & \vdots\\ O^t_{n,1} & O^t_{n,2} & \cdots & O^t_{n,m}\end{bmatrix},\qquad(1)$$

$O^t_{n,m} \in T$ 为第 n 个服务节点的第 m 个操作实现任务 O, 其服务质量记为 $Q^t_{i,j}$, 有

$$Q^t_{i,j} = \mathrm{Quality}(O^t_{i,j}), \tag{2}$$

$\mathrm{Quality}(X)$ 为对服务 X 的质量评估, 一般根据定义3的相关参数采用权重方式整合为一个函数值, 或者直接使用Pareto占优进行判断.

定义5 (Pareto占优)[6] 向量 $U=(u_1,u_2,\cdots,u_m)$ Pareto占优向量 $V=(v_1,v_2,\cdots,v_m)$(本文记为 $U \prec V$) 表示当且仅当 $\forall i \in \{1,2,\cdots,m\}$ 满足 $u_i \leqslant v_i$, 同时 $\exists j \in \{1,2,\cdots,m\}$ 使得 $u_i < v_i$ 成立. 向量 $x_u \in \Omega$ 是Pareto最优的, 当且仅当不存在 $x_v \in \Omega$ 满足 $F(x_v) \preceq F(x_u)$. Pareto最优解向量的集合称为问题的Pareto最优集, 对应的目标向量集合称为非占优集或Pareto前沿.

根据以上定义, 用户多QoS约束的网格任务调度问题可以定义为, 根据给定的 R,T,Q 求一个映射方案 M, 使得

$$\min F = (Q_{MC}, Q_{CC}, f(Q_{AC})). \tag{3}$$

其中: 任务Deadline的限定应分别满足总任务时限和子任务时限的要求; $f(Q_{AC})$ 为用户对服务节点稳定性的期望值, 其值属于区间[0, 1], Q_{AC} 越高表示机器越稳定.

2 基于UDPSO网格任务调度算法的设计

2.1 算法原理

粒子群优化(PSO)[7]算法是由Kennedy等提出的基于群体的群智能进化算法, 它将种群中每个个体看作搜索空间中的一个没有体积和质量的粒子, 这些粒子在搜索空间中以一定的速度飞行, 根据环境的适应度, 调整自身速度和方向向着好的区域移动, 因此在进化过程中保留并利用了位置与速度信息, 使得算法收敛度大幅提高. 同时, PSO算法还因其概念简单、实现容易、参数较少、能有效解决复杂优化任务等特点, 在模式识别、多目标优化和图形处理等领域得到广泛应用.

Coello等[8]对基本PSO算法进行改进, 提出了Multi-objectives PSO算法, 通过采用自适应网格机制来保存外部种群, 并引入新的变异策略来保证最终解的多样. 在此基础之上, 文献[9]提出了基于拥挤距离和 ε 占优机制的粒子群多目标算法, 这些算法在连续域问题的求解中取得了较好的结果. 针对任务调度离散问题, 文献[10-11]提出了一些离散粒子群算法, 但没有考虑用户QoS约束的多目标优化问题. 本文将这类离散问题与Pareto占优机制相结合, 利用配方均匀设计方法, 提出了针对离散问题求解的Pareto占优的优化机制, 以提高DPSO算法的运行效率.

2.2 UDPSO针对网格任务调度的编码

将离散PSO算法应用到网格任务调度中的一个关键问题是在问题解和粒子之间建立合适的映射关系. 采用整数编码机制, 各粒子根据定义4采用三元组矩阵来表示网格任务的调度方案, 即

$$\left\{\begin{array}{ccc} (k_{1,1},s_{1,1},c_{1,1}) & \cdots & (k_{1,m},s_{1,m},c_{1,m}) \\ (k_{2,1},s_{2,1},c_{2,1}) & \cdots & (k_{2,m},s_{2,m},c_{2,m}) \\ \vdots & \ddots & \vdots \\ (k_{n,1},s_{n,1},c_{n,1}) & \cdots & (k_{n,m},s_{n,m},c_{n,m}) \end{array}\right\}. \tag{4}$$

其中: 行号 i 为服务节点; 列号 j 为在服务节点上任务将被第 j 个操作实现; 假设有 n 个任务被调度, 由于所有任务可能被分配到同一服务节点上, 矩阵的列数 $m \in (0,n)$; $k_{i,j}$ 为任务的编号且 $k_{i,j} \in (0,n)$, 因为任务在服务节点上是顺序执行, 如果当前位置的前一位没有分配任务, 则当前位置也不可能分配任务, 所以有

$$k_{i,j-1} = 0 \rightarrow k_{i,j} = 0; \tag{5}$$

$s_{i,j}$ 和 $c_{i,j}$ 分别为任务 $k_{i,j}$ 在服务节点 r_i 上的估计起始执行时间和估计结束时间. 用户使用服务节点的花费为

$$Q_{CC} = \lambda(c_{i,j} - s_{i,j}) + \mathrm{RND}, \tag{6}$$

其中RND为扰动系数. 因此, 整个解空间将由 N 个map矩阵粒子组成, 这种设计适合本问题的求解, 并拥有较少的解空间.

2.3 粒子速度和位置的更新方法

微粒群算法的实质是利用当前微粒与个体极值和全局极值之间的差距信息来调整微粒的速度, 并改变微粒下一步的迭代位置. 在求解组合优化问题时, 由于当前解与极值解之间的差距难以用定量的方式表达, 导致微粒的速度和位置不能按照基础PSO算法进行更新. 为了使PSO算法能够用于离散域的优化问题, 借鉴遗传算法中交叉操作的思想, 结合文献[12]速度和位置的公式, 首先定义对于矩阵位置的变换子操作 $\mathrm{ch}(a_{11},a_{22})$ 为矩阵第 a_{11} 位与 a_{22} 位交换, 如 $A' = A \oplus \mathrm{ch}(a_{11},a_{22})$ 表示如下交换操作:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}.$$

由一个或多个交换子组成的序列称为交换序列, 记作 $\mathrm{CH} = (\mathrm{ch}_1, \mathrm{ch}_2, \cdots, \mathrm{ch}_m)$. 两个矩阵相减产生一个交换序列, 表示矩阵 A 按序列变换后能得到矩阵 A'. 由于交换子的先后顺序不同, 产生一个等价集来表示矩阵 A 到矩阵 A' 的变换, 将变换次数最少的交换序列称为基本交换序列集.

根据以上定义, 对传统的PSO速度和位置公式进行如下改进:

$$V_{id}^{k+1} = V_{id}^{k} + \alpha \otimes (p\mathrm{Best}_{id}^{k} - p_{id}^{k}) + \beta \otimes (p\mathrm{Best}_{id}^{k} - p_{id}^{k}). \quad (7)$$

其中: V_{id}^{k} 保留了基础交换序列中没有发生交换的交换子 ch_i 及其执行概率; $(p\mathrm{Best}_{id}^{k} - p_{id}^{k})$ 将产生一个从矩阵 $p\mathrm{Best}_{id}^{k}$ 到矩阵 p_{id}^{k} 的基本交换序列集CH; 符号 "$\otimes$" 表示变异乘法, $\alpha \otimes k$ 表示矩阵 k 中所有项都乘以实数 α; $\alpha, \beta \in [0,1]$ 为随机数, 称为学习因子, 即粒子向当前最优解学习的概率; $\alpha \otimes (p\mathrm{Best}_{id}^{k} - p_{id}^{k})$ 表示以概率 α 保留CH中的交换子, α 越大表示 $(p\mathrm{Best}_{id}^{k} - p_{id}^{k})$ 产生的交换序列集保留的变换子越多, 对 X_{id} 的影响越大,即

$$X_{id}^{'} = X_{id} \oplus \mathrm{CH}, \quad (8)$$

"$\oplus$" 表示矩阵按基本交换序列集进行交换.

如前文所述, 在多目标优化问题中, 虽然罚函数法能够有效地将多个目标转化为单个目标进行求解, 但容易造成群体多样性较差、由于先验知识不足使得权重难以衡量等缺点, 本文将配方均匀设计与Pareto多目标优化方法相结合, 通过逼近Pareto前沿获取调度方案的较优解集. 所提出算法采用文献[8]使用的类似策略, 利用外部存档Ar保存当前Pareto最优解的集合. 对于如何选取外部存档中的非劣解作为粒子学习样本, 学者们提出了不同策略[8,9,11,13]. 由于本文所产生的Pareto解集分布较均匀, 同时为了有效地减小计算复杂度, 采用随机选取一个非劣解的方式获得 $g\mathrm{Best}_{id}^{k}$ 和 $p\mathrm{Best}_{id}^{k}$.

2.4 基于均匀设计的适应值函数

在多QoS约束的网格任务调度问题的离散PSO算法中, 为了快速评价解的适应度, 通常定义个体的适应值函数为

$$F_{\mathrm{fit}} = \sum_{i=1}^{m} w_i f_i(x), \quad (9)$$

其中 $w_i \geqslant 0 (i = 1, 2, \cdots, m)$ 且 $\sum_{i=1}^{m} w_i = 1$. 为了便于评价适应值, 先将各目标函数作如下归一化操作: 设当前解集合为

$$\mathrm{Map} = \{\mathrm{map}_1, \mathrm{map}_2, \cdots, \mathrm{map}_n\},$$

令

$$a_i(\mathrm{DL}) = \{f_i(\mathrm{DL}_1), f_i(\mathrm{DL}_2), \cdots, f_i(\mathrm{DL}_n)\}, \quad i = 1, 2, \cdots, m,$$

其中 $f_i(\mathrm{DL})$ 为各类QoS需求的Deadline, 则 $f_i(\mathrm{map})$ 可以被规范化为

$$S_i(\mathrm{map}) = f_i(\mathrm{map}) / a_i(\mathrm{DL}),\ i = 1, 2, \cdots, m. \quad (10)$$

网格任务调度问题(3)可以转化为如下多目标问题:

$$\min F = (S_{\mathrm{MC}}(\mathrm{map}), S_{\mathrm{CC}}(\mathrm{map}), S_{\mathrm{AC}}(\mathrm{map})). \quad (11)$$

同时, 适应度函数(9)可以转化为

$$F_{\mathrm{fit}} = \sum_{i=1}^{m} w_j S_j(x). \quad (12)$$

如果式(12)采用固定权值集合 $W = (w_1, w_2, \cdots, w_n)^{\mathrm{T}}$, 则其产生的适应度函数可能只找到部分Pareto最优解, 一些具有重要特性的Pareto最优解可能被忽略[14-15]. 针对这一现象, 文献[15]提出了应用每次随机产生的若干组权重参数来计算适应值函数, 并取得了一定的成果. 但是将随机权重 W 代入式(12)容易产生以下问题: 1) 为了求出尽可能均匀的Pareto最优解, 必须随机产生很多组权向量, 使得算法计算量增大; 2) 如果Pareto最优解的个数为无穷多, 则即使产生了大量的权向量, 也可能使生成解集中分布在目标函数空间中的一部分, 而其他部分分布较为稀疏, 造成一些重要的Pareto最优解依然不能被有效发现[16].

为了避免上述两类问题, 本文算法利用文献[16-17]所描述的配方均匀设计法, 产生若干个在目标函数空间均匀分布的权向量组, 使其均匀指向不同的Pareto最优解. 本文涉及3个QoS需求, 设权值 $w_i \in [0.1, 0.9]$, 则均匀设计中的因素 $n = 3$, 量子化区间 $[0.1, 0.9]$, 取 $s = 11$ 个测试点作为因素的水平, 每组水平满足 $\sum_{i=1}^{m} w_i = 1$. 权值 w_1 可以由公式 $w_1 = 1 - w_2 - w_3$ 直接得出, 因此选用因素 $n = 2$ 的均匀设计表即可. 最终算法将产生在 s^n 上均匀分布的点集

$$P^* = \{X_1, X_2, \cdots, X_m\}.$$

本文参考文献[17]中类似的算法产生均匀设计的方案, 令其为算法1, 执行步骤如下.

Step 1: 根据文献[17]选取适当均匀设计表 $U_s^*(s^{n-1})$, 并用 $\{q_{kj}\}$ 表示 $U_s^*(s^{n-1})$ 的每一个元素.

Step 2: 对于每个 k 和 j 计算

$$c_{kj} = \frac{q_{kj} - 0.5}{s},$$

$$k = 1, 2, \cdots, s,\ j = 1, 2, \cdots, n-1.$$

Step 3: 下式所得 x_{kj} 即为 s 次实验 n 种因素的配方均匀设计, 记为 $UM_s^*(s^{n-1})$:

$$x_{k1} = 1 - \sqrt[s-j]{c_{k1}},\ k = 1, 2, \cdots, s;$$

$$x_{ks} = \prod_{j=1}^{s-1} c_{kj}^{\frac{1}{s-j}},\ k = 1, 2, \cdots, s. \quad (13)$$

2.5 UDPSO算法描述

结合配方均匀设计算法, 本文提出网格任务调度策略的UDPSO算法, 具体执行步骤描述如下.

输入: R, T, Ar, $p_{\text{粒子}}$, $M_{\text{进化代数}}$;

输出: Pareto调度方案集 P^*.

Step 1: 初始化粒子群数目 $p_{粒子}$ 和学习因子, 总迭代次数设为 $M_{进化代数}$, 当前迭代次数 $t=0$, 初始化外部存档文件 Ar 和粒子速度等参数. 由算法 1 产生 P^* 个权向量组, 每个粒子选择自身的权向量参数, 最佳粒子位置 $g\mathrm{Best}_{id}^k = p\mathrm{Best}_{id}^k = \phi$, 随机产生服务节点和任务参数, 使用贪心算法生成初始种群 P_0.

Step 2: 判定算法是否满足最大迭代次数的停止条件, 若不满足则转至 Step 3.

Step 3: 根据各粒子的当前位置 X_{id} 计算下一个位置 X_{id}^*, 即新解.

Step 3.1: 从 Pareto 解集中随机选取一个解作为 $g\mathrm{Best}_{id}^k$ 和 $p\mathrm{Best}_{id}^k$, 根据式 (7) 更新每个粒子的速度, 即产生一个基础交换序列;

Step 3.2: 根据式 (9) 计算新解的位置;

Step 3.3: 更新任务在新服务节点上的完成时间和用户费用等信息, 根据定义 4 规范化各解的结构, 以保证式 (5) 成立.

Step 4: 利用 Pareto 占优选择新解中的非劣解, 对于不能使用 Pareto 占优判断的解, 按照粒子最初选用的权向量 $W_j(j=1,2,\cdots,n)$, 根据式 (12) 计算每个新粒子的适应度, 更新粒子自身最优集的外部存档, 产生新的 $p\mathrm{Best}_{id}^k$. 利用所选取的各类权向量的前 2 个最小值来更新全局 Pareto 集Ar, 如果 Pareto 解中有粒子适应度小于新解, 则被新解替换, 否则丢弃新解, 转至 Step 2.

Step 5: 如果满足约束条件则输出结果, 否则提示没找到满足约束条件的解.

3 算法收敛性

Veldhuizen 等[18]证明了若一个进化算法满足以下两个条件, 则序列 p_k 以概率 1 收敛到全局最优解集: 1) 对于可行域中任意两个解 X 和 X^*, 满足 X^* 是 X 通过杂交和变异可达的; 2) 序列 $p_1,p_2,\cdots,p_k$ 是单调的, 即 $\forall k$, p_{k+1} 中任意解非劣于 p_k 中的对应解.

文献 [16] 将条件 1 的“可达”扩展为“ ε- 精度可达”, 即满足上述两个条件的进化算法将以概率 1 收敛到具有 ε- 精度的最优解, “ε- 精度可达”表示

$$\mathrm{Prob}\{\lim_{t\to\infty}(P_{\mathrm{true}}-P_t)\leqslant\varepsilon\}=1,$$

其中 P_{true} 为真实的最优解. 运算符 “$\oplus$” 类似于将任务调度方案结果集 Map_i 作适当的交换操作和变异操作, 即按概率 α,β 将 Map_i 与 $p\mathrm{Best}_{id}^k$ 和 $g\mathrm{Best}_{id}^k$ 进行交换. 由 UDPSO 算法的 Step 2 ~ Step 4 可以得到, 算法产生的结果集 Map_{k+1} 至少改进 Map_k 中的一个解, 因此 $\mathrm{Map}_1,\mathrm{Map}_2,\cdots,\mathrm{Map}_k$ 是向 Pareto 最优单调递增的. 注意到, 本算法中粒子的状态可能因交换操作的概率问题出现扰动, 即出现劣解, 但外部存档中的 Pareto 解集由于该类解一直采用相同权重向量, 其运行过程可以看作是一个包含两种状态的 Markov 链:

1) 使得 $\{\mathrm{Map}_k\}$ 满足 $\forall A^*\in\mathrm{Map}_k$, $\exists X\in\mathrm{Map}_{\mathrm{true}}$ 使得 $\|X^*-X\|\leqslant\varepsilon$, 其中 $\mathrm{Map}_{\mathrm{true}}$ 为真实的最优分配方案;

2) 解集 $\{\mathrm{Map}_k\}$ 不满足 1).

从状态 1) 到状态 2) 的转移概率为 0, 即状态 1) 为吸收态. 本算法将使得从状态 2) 到状态 1) 至少 ε- 精度可达, 即状态 2) 到状态 1) 的概率大于 0, 表明此状态是瞬时的. 在这两种情况下, UDPSO 完全满足 Markov 定理[19], 即 UDPSO 以概率 1 收敛.

4 性能评估

本文实验基于 GridSim Toolkit 环境, 结合 Alea[20] 系统进行. 考察了 50 ~ 300 个独立作业在 10 个服务节点组成的网格系统的调度性能, 并假设每个任务在所有计算节点上的执行时间是可预测的. 各任务的 QoS 参数均在一定范围内随机生成, 服务费用按式 (6) 计算, 节点处理能力 MIPS 在区间 [200, 600] 中随机选取, 实验运行在 AMD X4 640 3.0 G, 物理内存 2 G 的 PC 机上.

在用 UDPSO 寻找任务调度的最优可行解方案时, 种群规模为 30, 外部存档 Ar 也为 30, 最大允许迭代次数为 300. NSGA-II 算法的相关参数根据文献 [21] 中的实验数据进行设置, DPSO 算法的相关参数和 UDPSO 算法的部分参数根据文献 [22] 进行设置. 本文所有仿真实验均重复进行 30 次独立实验, 并对实验得到的各项数据求平均值.

图 2 为任务调度算法的执行时间. 由图 2 可见, 本文所提出的算法在执行时间上低于 NSGA-II 和 DPSO 算法, 但与 NSGA-II 相比, 随着任务数量的增加, 在生成基础交换序列时占用的时间也越多, 造成算法随任务增加执行时间增长较快, 因此对大任务量处理的算法有待进一步优化.

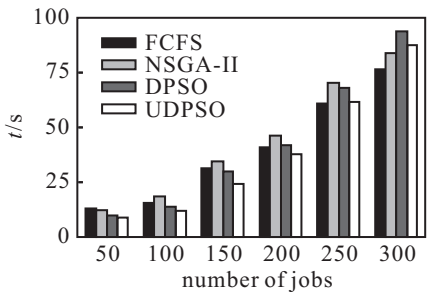


图 2 任务分配算法的执行时间

表 1 描述了在不同任务规模中, NSGA-II, DPSO 和 UDPSO 算法的平均收敛迭代次数. 由表 1 可见, 由于 UDPSO 算法基于均匀设计的权重参数, 算法的收

敛性较 DPSO 更快, 且保证了解集中各项 QoS 参数性能明显优于 DPSO 算法.

表 1　算法收敛次数对比

(任务规模, 节点)	NSGA-II	DPSO	UDPSO
(50, 10)	49	49.1	44.3
(100, 10)	95.8	82.4	77.5
(150, 10)	151	138	115.7
(200, 10)	245.4	182.4	173.3

为了衡量粒子群规模对 UDPSO 算法性能的影响, 实验分别考察了 50 个任务在粒子数为 20, 30 和 50 时, 算法的收敛速度和解的情况, 结果如图 3 所示. 由图 3 可见, 粒子群规模越大, 算法的收敛速度和解的性能越好, 但规模过大会显著增加算法的执行时间, 因此在有较大可能找到较优解的情况下, 应保持粒子的规模较小, 本文实验选择粒子数量为 30.

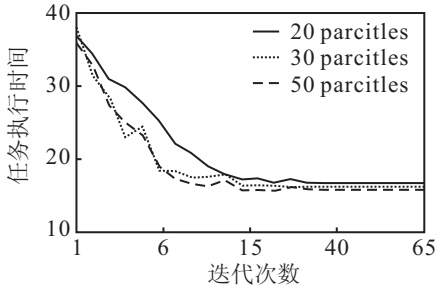


图 3　不同群体规模对算法的影响

图 4 为 100 个任务在 10 个可用服务节点的网格环境中运行后的节点负载情况. 由图 4 可见, 虽然 DPSO 算法多次实验后的平均值趋于均衡, 但每次实际的节点负载量差异较大; 而 UDPSO 算法每次的负载情况都相对均衡. 如: 系统中服务节点 4 和节点 5 的整体性能相对较弱, DPSO 算法指派给这些节点的任务便少得多, 节点 3 和节点 7 性能较优, 被分配的任务较多, 造成任务执行缓慢, 影响了系统性能. UDPSO 算法既体现出服务节点性能的优劣, 又在任务调度时考虑了负载的均衡问题, 使系统的整体性能明显优于 DPSO 算法.

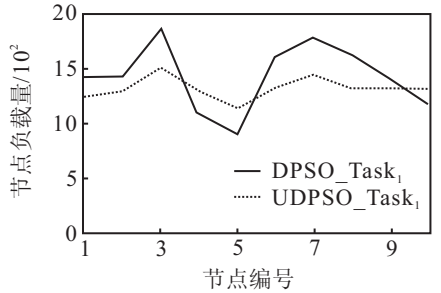
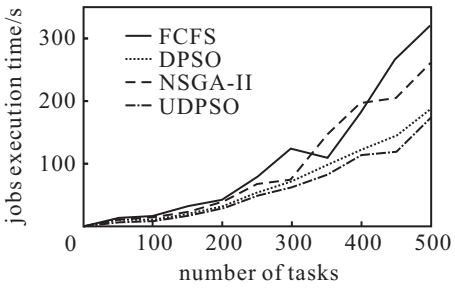
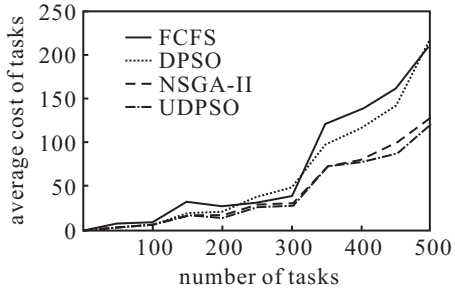


图 4　服务节点负载情况

Pareto 最优集的平均任务完成时间和用户花费比较如图 5 所示. 由图 5 可见, UDPSO 产生的任务执行时间跨度和用户平均花费分别优于 NSGA-II 计算结果的 6% 和 8%. 与 DPSO 相比, UDPSO 的执行时间提高近 12.2%, 主要原因是使用配方均匀设计指导 Pareto 最优解集的产生, 提高了算法的收敛性. 同时, 结果集中任务的平均执行时间和用户费用的分布也更为均匀, 便于用户根据偏好选择调度方案. 比较而言, UDPSO 比 NSGA-II 的执行效率和 Pareto 解集分布都有优势, 且 UDPSO 算法涉及的参数较少, 算法实现相对简单, 便于在网格环境或分布式环境中执行.



(a)　平均执行时间



(b)　平均任务花费

图 5　Pareto 最优集中 Q_{mc} 和 Q_{cc} 的平均值

5　结　论

用户多 QoS 网格任务调度问题通常以多个可能相互冲突的需求作为目标函数, 是 NP-hard 的组合优化问题, 尽管文献中已有许多启发式算法试图获取最优解, 但解的质量仍然不能令人满意. PSO 算法实现简单, 具有并行搜索、可调参数少等特点, 主要用于连续问题的优化求解. 鉴于此, 本文利用基于离散空间的 PSO 算法, 通过自定义的速度和位置更新方法, 结合配方均匀设计思想, 得到适于求解多 QoS 约束网格任务调度问题的 UDPSO 算法. 既保留了 PSO 算法的优点, 又将其应用于离散空间的求解中. 仿真实验表明, 本文提出的 UDPSO 算法在计算复杂性上优于 NSGA-II 算法, 并能达到比 NSGA-II 和 DPSO 算法更优的性能指标, 完全能够满足小规模网格环境中任务调度时用户多 QoS 约束的需求.

参考文献(References)

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

[2] Geelan J. Twenty one experts define cloud computing[DB/OL]. (2008-08-25). http://cloudcomputing.sys-con.com/read/612375p.htm.

[3] Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-Degree Compared[C]. Grid Computing Environments Workshop. Austin, 2008: 1-10.

[4] Andronikos T, Koziris N. Optimal scheduling for UET-UCT grids into fixed number of processors[C]. The 8th Euromicro Workshop on Parallel and Distributed. Cancun, 2000: 237-243.

[5] 蒲汛, 何为, 卢显良. 基于改进遗传算法的多QoS约束网格任务调度[J]. 电子科技大学学报, 2010, 39(增): 54-56.
(Pu X, He W, Lu X L. Improved genetic algorithm for grid job scheduling of multi-QoS constraints[J]. J of University of Electronic Science and Technology of China, 2010, 39(S): 54-56.)

[6] Geilen M, Basten T, Theelen B, et al. An algebra of pareto points[J]. Fundamenta Informaticae, 2007, 78(1): 35-74.

[7] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway, 1995: 1942-1948.

[8] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 256-279.

[9] Margarita Reyes Sierra, Carlos A Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance[C]. The 3rd Int Conf on Evolutionary Multi-Criterion Optimization. Guanajuato: Springer-Verlag, 2005: 505-519.

[10] 卜艳萍，俞金寿. 离散微粒群优化算法在网格任务调度中的应用[J]. 计算机仿真, 2008, 25(4): 175-178.
(Bu Y P, Yu J S. Application of discrete particle swarm optimization algorithm to grid task scheduling[J]. Computer Simulation, 2008, 25(4): 175-178.)

[11] Salman A, Ahmad I, Al-Madani S. Particle swarm optimization for task assignment problem[J]. Microprocessors and Microsystems, 2002, 26(8): 363-371.

[12] Clerc M. Discrete particle swarm optimization, illustrated by the traveling salesman problem[C]. New Optimization Techniques in Engineering. Berlin: Springer-Verlag, 2004: 219-240.

[13] Hernández Díaz A G, Santana-Quintero L V, Coello C C A, et al. Pareto-adaptive ε-dominance[J]. Evolutionary Computation, 2007, 15(4): 493-517.

[14] Fonseca C M, Fleming P J. An overview of evolutionary algorithms in multiobjective optimization[J]. Evolutionary computation, 1995, 3(1): 1-16.

[15] Ishibuchi H, Murata T. A multi-objective genetic local search algorithm and its application to flowshop scheduling[J]. IEEE Trans on Systems, Man and Cybernetics, 1998, 28(3): 392-403.

[16] 王宇平, 焦永昌, 张福顺. 解多目标优化的均匀正交遗传算法[J]. 系统工程学报, 2003, 18(6): 481-486.
(Wang Y P, Jiao Y C, Zhang F S. Uniform and orthogonal genetic algorithm for multiobjective optimization[J]. J of Systems Engineering, 2003, 18(6): 481-486.)

[17] Fang K T, Wang Y. Number theoretic methods in statistics[M]. London: Capman and Hall, 1994: 68-81.

[18] Veldhuizen D A V, Lamont G B. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art[J]. Evolutionary Computation, 2000, 8(2): 125-147.

[19] Trivedi K S. Probability and statistics with reliability, queuing and computer science applications[M]. The 2nd ed. Boston: Academic Press, 1990: 78-85.

[20] Klusáăaek D, Rudová H. Alea 2: Job scheduling simulator[C]. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. Malaga, 2010.

[21] Yang Y, Wu G, Chen J, et al. Multi-objective optimization based on ant colony optimization in grid over optical burst switching networks[J]. Expert Systems with Applications, 2010, 37(2): 1769-1775.

[22] 范小芹, 蒋昌俊, 方贤文, 等. 基于离散微粒群算法的动态Web服务选择[J]. 计算机研究与发展, 2010, (1): 147-156.
(Fan X Q, Jiang C J, Fang X W, et al. Dynamic web service selection based on discrete particle swarm optimization[J]. J of Computer Research and Development, 2010, (1): 147-156.)