多 QoS 约束下网格工作流调度的克隆选择算法*

赵建峰[1,2] 曾文华[2,3] 刘 敏[1,2] 张 雪[2,3]

[1](厦门大学 信息科学与技术学院 智能科学与技术系 厦门 361005)
[2](厦门大学 福建省仿脑智能系统重点实验室 厦门 361005)
[3](厦门大学 软件学院 厦门 361005)

摘 要 多 QoS 约束下的工作流调度是网格计算中难以求解的问题. 在深入剖析该问题难解性基础上,采用克隆选择算法求解该问题. 首先通过增加网格服务的唯一标识,简化工作流调度的编码方式. 其次,提出 QoS 偏好的概念,将调度问题的目标函数转换为适应值函数. 该算法具有 QoS 属性的可扩展性. 最后通过大量实验,优化算法参数,与基于遗传算法、蚁群算法的调度算法对比,克隆选择算法求解效率较优. 在扩展情况下,与单一 QoS 约束下的时间、费用贪婪算法对比,克隆选择算法能进行最优调度.

关键词 工作流调度,克隆选择算法,多 QoS 约束,网格计算
中图法分类号 TP 393

Grid Computing Workflow Scheduling Clonal Selection Algorithm with Multi-QoS Constraints

ZHAO Jian-Feng[1,2], ZENG Wen-Hua[2,3], LIU Min[1,2], ZHANG Xue[2,3]

[1](*Cognitive Science Department*, *School of Information Science and Technology*, *Xiamen University*, *Xiamen* 361005)
[2](*Fujian Key Laboratory of the Brain-Like Intelligent Systems*, *Xiamen University*, *Xiamen* 361005)
[3](*School of Software*, *Xiamen University*, *Xiamen* 361005)

ABSTRACT

Workflow scheduling with multi-QoS constraints is hard to be solved under the grid computing environment. A clonal selection algorithm, named EvoWF, is proposed to solve workflow scheduling problem based on deep analysis on the difficulty of this problem. The encoding of working scheduling is simplified by adding the grid service identification. The concept of QoS preference is proposed, which converts object function of workflow scheduling to fitness function, and QoS attributes can be extended. Compared to genetic algorithm and ant colony optimization, EvoWF is more efficient. In extension, EvoWF gets the same optimum scheduling results compared with the single-QoS constraint greed time or cost algorithm. Moreover, the effect of parameters is analyzed by experiments.

* 国家自然科学基金项目(No. 60672018,40774065)、国家 863 计划项目(No. 2006AA01Z129)资助
收稿日期: 2011 - 04 - 07
作者简介 赵建峰,男,1982 年生,博士研究生,主要研究方向为网格计算、多目标优化、机器视觉. E-mail: zjfhi2008g@gmail.com. 曾文华,男,1964 年生,教授,博士生导师,主要研究方向为网格计算、嵌入式系统、嵌入式软件、并行演化计算、机器视觉. E-mail: whzeng@xmu.edu.cn. 刘敏,男,1974 年生,博士研究生,讲师,主要研究方向为多目标优化算法. 张雪,女,1987 年生,硕士研究生,主要研究方向为网格计算.

Key Words Workflow Scheduling, Clonal Selection Algorithm, Multi-QoS Constraints, Grid Computing

1 引 言

网格计算作为一个新的领域出现,相对于传统的分布式计算,网格计算更加注重大规模的资源共享、创新性应用和高性能[1]. 在网格中,大型的计算任务往往是由一个有依赖关系任务集组成,如何将这些任务有效分配到服务集上称为工作流调度问题,该问题由于其难解性,成为学者们研究的热点问题[2-4]. 经典算法有时间贪婪算法、费用贪婪算法、Min-Min 、Max-Min 和 Heterogeneous Earliest Finish Time (HEFT) 等. 苑迎春等[5]提出基于串归约的网格工作流费用优化方法,能较好较快地完成工作流的调度. 但上面提到的算法都是在单一 QoS 约束进行调度,已经无法满足多 QoS 约束的需求.

在多 QoS 约束下,各个管理实体对网格的 QoS 要求是不一样的,甚至是相互冲突的,这更加剧了调度问题的难解性. 墨尔本大学的 Yu 等[6]提出时间和预算约束下的基于遗传算法的工作流调度算法,该算法只能在时间和费用两个 QoS 约束下进行且效率不高. 金海等[7]提出具有扩展 QoS 约束的CGSP (China Grid Supporting Platform) 管理器,但其只针对 China Grid 支撑平台. 张伟哲等[8]使用多目标演化算法进行求解,但使用多目标算法求解该问题较为复杂,且没有必要给出 Pareto 最优解集. Chen 等[9]采用蚁群算法求解该问题,同时具有可扩展的 QoS,具有较大的借鉴意义.

综观以上算法,网格中多 QoS 约束下如何高效进行工作流调度依然是一个研究难点. 本文采用克隆选择算法求解多 QoS 约束下的网格工作流调度问题,简称 EvoWF 算法. 通过深入分析多 QoS 约束下调度问题的难解性,指出采用进化算法进行求解的必要性. 通过提出 QoS 偏好的概念,将调度问题的目标函数转换为适应值函数,使得算法更加符合实际的应用,同时在迭代上借鉴克隆选择算法的思想,增强了算法的鲁棒性.

2 问题描述及分析

大规模网格计算工作流调度包含两个部分: 具有依赖关系的任务集和服务节点集. 任务集是指网格中完成一次科学计算所包含的所有任务,记为 $T = \{t_i\}, i = 1,2,\cdots,n$,其中 t_i 为一个任务. 任务集常抽象为一个有向无环图(Directed Acyclic Graph, DAG): $G = (V,E)$, V 为任务节点,E 为连接边. 如图 1 所示,图中 $n = 7$.

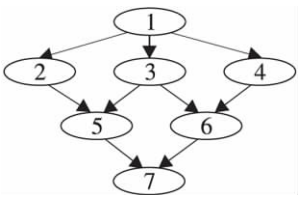


图 1 具有依赖关系的任务集

Fig. 1 Task set with dependency

服务节点集是指完成任务集 T 的所有服务的集合,记为 $S = \{s_j\}, j = 1,2,\cdots,m$,其中 s_j 为一个服务. 一个服务可通过其 QoS 来衡量,虽然 EvoWF 具有 QoS 扩展性,本文为说明将采用时间、费用及可靠性 3 个最常见的 QoS 来描述一个服务,因此,一个服务 s_j 表示为 $s_j(\theta_{ij},\omega_{ij},\varepsilon_{ij})$,其中 θ_{ij}、ω_{ij}、ε_{ij} 是指任务 t_i 使用服务 s_j 所花费的时间、费用及获得的可靠性. 实际中,可根据需要选择 QoS 属性.

对应于图 1,一个服务节点列表集如表 1 所示. 表中第 1 列表示任务编号,第 2 列表示针对于每一个任务的服务节点,每个服务节点中包含其唯一标识和服务节点完成该任务所需要的时间、费用与完成该任务的可靠性. 示例中,每个任务有 2 个服务节点,实际上每个任务可有一个或者多个服务节点.

表 1 具有 QoS 属性的服务列表

Table 1 Service list with QoS attributes

任务	服务节点
1	{1,3,100,0.980} ,{2,4,90,0.961}
2	{3,6,200,0.973} ,{4,4,300,0.99}
3	{5,4,70,0.954} ,{6,8,10,0.965}
4	{7,2,60,0.872} ,{8,9,60,0.943}
5	{9,10,110,0.942} ,{10,110,10,0.974}
6	{11,20,70,0.769} ,{12,70,10,0.932}
7	{13,80,55,0.992} ,{14,40,150,0.984}

在网格中,用户关心的是任务能否在限定的 QoS 下完成. 因此,调度问题实际上是在限定 QoS 的情况下,给出最优的调度方案. 在给出调度问题前,先给出关键路径的定义.

定义 1 关键路径是指有向无环图 G 中部分任

务的集合,该任务集具有最长的执行时间并决定总任务的最短完成时间,记为 $K(S)$.

因此,工作流调度问题 $P(1)$ 描述如下.

问题 $P(1)$　对于调度序列 $L(s_{x1},s_{x2},\ldots,s_{xn})$,有

$$f(s_{x1},s_{x2},\cdots,s_{xn}) = \left(\min\left(\sum_{\theta_{xi}\in K(S)}\theta_{xi}\right),\min\left(\sum_{i=1}^{n}\omega_{xi}\right),\max\left(\prod_{i=1}^{n}\varepsilon_{xi}\right)\right),$$

$$s.t.\ \sum_{\theta_{xi}\in K(S)}\theta_{xi}\leqslant T_{\text{use}},\sum_{i=1}^{n}\omega_{xi}\leqslant C_{\text{use}},\prod_{i=1}^{n}\varepsilon_{xi}\leqslant P_{\text{use}},$$

其中,s_{xi} 表示任务 t_i 分配的服务,T_{use},C_{use},P_{use} 分别表示用户给出的时间、费用和可靠性的 QoS 约束.

下面证明该问题是难求解的问题及存在最优解的条件.

定理 1　若采用穷举算法求解问题 $P(1)$,问题 $P(1)$ 是难求解问题.

证明　设问题 $P(1)$ 中包含的任务个数为 n,对于每个任务有 m 个服务,因此,对于每个任务有 m 种选择,采用穷举算法,则算法的时间复杂度为 $O(m^n)$. 证毕.

定理 2　若在 $P(1)$ 的解空间上存在最优解,则对于关键路径上任意任务,其服务列表中至少存在一个服务,其费用和可靠性是最优的.

证明　设 $L^*(s_{x1},s_{x2},\ldots,s_{xn})$ 是任务集 T 的最优解,因此有

$$f^*(s_{x1},s_{x2},\ldots,s_{xn}) = \left(\min\left(\sum_{\theta_{xi}\in K(S)}\theta_{xi}\right),\min\left(\sum_{i=1}^{n}\omega_{xi}\right),\max\left(\prod_{i=1}^{n}\varepsilon_{xi}\right)\right).$$

若存在一个在关键路径上的任务 t_i,其选定的服务是 $s_i(\theta_{ii},\omega_{ii},\varepsilon_{ii})$,若该服务的费用或可靠性不是最优的,那么对于 t_i 一定存在这样一个服务 $s_j(\theta_{ij},\omega_{ij},\varepsilon_{ij})$,有 $\theta_{ii}<\theta_{ij}$,而 $\omega_{ii}>\omega_{ij}$ 或 $\varepsilon_{ii}<\varepsilon_{ij}$. 那么将序列 L^* 中的服务 s_i 替换为 s_j,一定使得总费用减少或总可靠性增加,这与 $\min\sum_{i=1}^{n}\omega_{xi}$ 或 $\max\sum_{i=1}^{n}\varepsilon_{xi}$ 矛盾. 证毕.

推论 1　若对于关键路径上的每个任务,其服务列表不存在一个服务在所有的 QoS 属性上是最优的,则该任务集不存在最优解.

定理 3　若对于 T 中每个任务,其服务列表存在一个在所有 QoS 属性上都是最优的服务,那么 T 存在最优解.

证明　因为对于 T 中每个任务,其服务列表存在一个在所有 QoS 属性上都是最优的服务,那么选定一个序列 $L^*(s_{x1},s_{x2},\ldots,s_{xn})$,其中 s_{x1} 是任务 t_i 上的那个最优服务,一定有 L^* 是 $P(1)$ 的最优解.

若 L^* 不是最优解,则有 L_1^* 不同与 L^*,使得 L_1^* 是在某一 QoS 属性上优于 L^*,这与条件矛盾,因为对某一 QoS 属性来说,L^* 都是最优的.

因此存在最优解 L^*. 证毕.

若定理 3 的条件成立,那么求解 $P(1)$ 将简化为选择具有最优 QoS 的服务,但在网格计算中多个 QoS 属性相互制约导致定理 2 的条件难以成立,因此,如推论 1 所示,调度不存在最优解. 又由定理 1 知,目前采用穷举算法求解该问题是不可行的,因此问题 $P(1)$ 需要进化算法来求解该问题的近似解.

3　多 QoS 约束下工作流调度的克隆选择算法

针对问题 $P(1)$,本文采用克隆选择算法来求解. 下面介绍 EvoWF 的 3 个核心部分: 编码,个体适应值函数和搜索算法.

3.1　算法的编码

由 $P(1)$ 知,网格环境下的任务调度是要寻找一个符合约束条件的调度方案. 通过第 2 节的分析知道,一个调度方案可看作一个服务排列. 因此本文通过增加服务列表的唯一标识来表示任务的调度方案,即每个任务选取一个服务,其服务标识组成一个实数序列,如(1 3 5 8 9 11 14),表示任务 1 在服务 1 执行,任务 2 在服务 3 执行等. 该方案的优点是简单,将调度方案转换为服务标识的排列,所需的只是在资源管理器返回服务列表时对服务进行唯一标识.

3.2　算法的个体适应值函数

从 $P(1)$ 看出,该问题是典型的多目标优化问题,可使用多目标优化算法来对 $P(1)$ 求解,但多目标优化算法存在实现复杂、搜索效率不高的问题. 因此,本文未采用多目标优化算法来求解. 受实际使用中,用户对 QoS 具体值不敏感,但对 QoS 属性却有明确偏好的启迪,本文提出 QoS 偏好的概念.

定义 2　QoS 偏好 Q_i 是指用户对某一 QoS 属性的限定程度,取值范围为 0 ~ 1,且有 $\sum_{i=1}^{c}Q_i=1$,其中 c 为 QoS 的属性个数.

算法实际使用中,若用户没有明确的 QoS 偏好,则此时用户对 QoS 属性无要求. 本算法采用 (0.1,0.8,0.1) 作为默认的 QoS 偏好,此时算法给出的结果有较好的经济性,可在兼顾时间和稳定性的情况下,减少用户开销.

根据定义 2,对于解序列 $L(s_{x1},s_{x2},\cdots,s_{xn})$,则有

EvoWF 的个体适应值函数:

$$t = \sum_{\theta_{xi} \in K(S)} \theta_{xi},\ c = \sum_{i=1}^{n} \omega_{xi},\ p = \prod_{i=1}^{n} \varepsilon_{xi},$$

$$f(t) = \frac{t_{\max} - t}{t_{\max} - t_{\min}},\ f(c) = \frac{c_{\max} - c}{c_{\max} - c_{\min}},$$

$$f(p) = \frac{p - p_{\min}}{p_{\max} - p_{min}},\ f = Q_t f(t) + Q_c f(c) + Q_p f(p),$$

其中,$t_{\max}$,$t_{\min}$,$c_{\max}$,$c_{\min}$,$p_{\max}$,$p_{\min}$ 指任务集的最大,最小执行时间,最大,最小费用,最大,最小可靠性; Q_t, Q_c,Q_p 分别指用户对时间,费用和可靠性的偏好.

3.3 算法步骤

本文采用克隆选择算法来进行搜索. 依据 Garrett[10] 在调度问题上相对于经典遗传算法,克隆选择算法更加鲁棒. 其主要步骤包括克隆和变异. 克隆是指在种群中选择优良的个体进行克隆,同时淘汰最差的个体,然后进行变异. 下面给出克隆与变异的伪码.

算法 1 EvoWF 中的克隆算法

```
Clone(种群){
    种群中的个体按照其适应值进行排序;
    选择 Pc·Pn 个最优个体;
    for(选择出的个体){
        Mutation(个体);
        将变异个体添加到新种群;
        克隆个体;
        Mutation(个体);
        将克隆后变异个体添加到新种群;
    }
    生成 Pe·Pn 个个体添加到新种群;
    返回新种群;
}
```

算法 2 EvoWF 中的变异算法

```
Mutation(个体){
    变异次数 n = 个体长度 l·Mp;
    while(i < n){
        在个体中随机选择一个位置;
        查找该位置对应的服务;
        随机选择一个服务替换现在的值;
    }
    返回替换后的个体;
}
```

3.4 算法的伪码

为了优化算法效率,EvoWF 中引入最优个体无变化的机制. 在算法迭代过程中,记录算法的最优个体,若最优个体连续一定的次数没有变化,则停止算法. 同时算法若找到最优解,则适应值为 1 的个体也会停止算法. 下面给出 EvoWF 算法的伪码.

算法 3 EvoWF 算法

```
main(){
    依据定理 3,生成一个个体,其中,对于每个任务
    if(存在最优服务)
        选择最优服务;
    else
        选择用户 QoS 偏好中最大值的所对应的 QoS 属性最优的服务;
    在初始种群中添加 Pn·Pc 个该个体;
    在初始种群中添加 Pn·Pe 个随机生成的个体;
    while(达到最大迭代次数 Ln){
        按照式(1) 计算个体适应值;
        if(个体适应值 > 最优个体适应值){
            设置最优个体适应值;
            迭代计数 c 清零;
        } else
            迭代计数 c 加 1;
        if(迭代计数 c 达到预定次数 Ls)
            结束迭代;
        if(满足最优个体适应值 Bf)
            结束迭代;
        Clone(种群);
    }
}
```

算法中的参数及取值如表 2 所示,其中,c 为常数,取值为 10,sp 为求解空间的大小,f_i 为要进行变异的个体适应值,按照式(1) 计算. 算法中容易确定的参数有: L_n、B_f、M_p、P_e. 因为 EvoWF 的主要停止条件是 L_s,所以 L_n 只作为最坏情况下的停止条件,算法中取值为: $c \cdot \log_2(sp)$.

表 2 算法参数

Table 2 Parameter setting

参数	意义	取值
L_n	最大迭代次数	$c \cdot \log_2(sp)$
L_s	最优个体无变化次数	10
B_f	最佳个体适应值	1
P_n	种群中个体数	10
P_c	克隆比例	0.175
P_e	淘汰比例	$1 - 2 \cdot 0.175$
M_p	变异率	$1 - f_i$

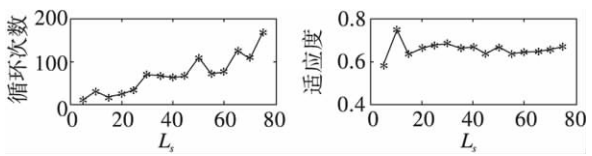
对于不容易确定的参数,本文通过实验的方法给出参数的最佳取值. 其基本思想是,选取某一个参数,固定其他参数的取值,分析该参数取值对算法求解速度和精度的影响. 详细的实验方法及结果参见实验部分.

4　实　验

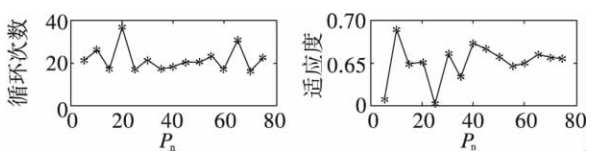
为体现 EvoWF 算法的通用性,更加可靠地测试算法的性能同时便于与其他算法进行比较,本文采用仿真环境进行实验. 实验的主要目的有: 1) 确定算法参数的取值; 2) 验证算法的正确性; 3) 与遗传算法及蚁群算法进行比较.

4.1　不易确定参数的实验方法及结果

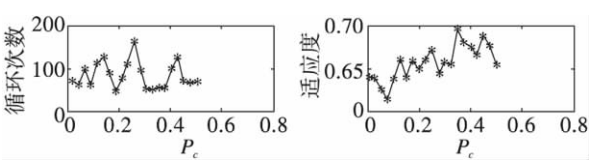
对测试参数的每个取值,随机生成 1 000 个任务集,每个任务随机生成低于 5 个服务节点,同时随机生成用户的 QoS 偏好. 对每个任务集进行求解,观察参数对算法求解速度和求解精度的影响. 为不与具体的机器相关,本文通过迭代的次数来表示算法运行的速度,算法运行结果如图 2 所示.



(a) L_s



(b) P_n



(c) P_c

图 2　不同参数对算法求解速度与精度的影响

Fig. 2　Effect of different parameters on algorithm's fitness and iteration

从图中 L_s 的曲线看出,随着 L_s 的增大,算法运行时间增加,精度增加. 从 P_n 的曲线看出,随着 P_n 的增多,迭代次数基本没有变化,但算法的运行时间在增加,因为 P_n 增多,单次迭代时间增加,同时,求解精度增加. 从图 2 中看出,P_c 的取值为 0. 175 的时候,算法迭代次数较少,精度较高.

4.2　本文算法的可行性

下面对图 1 的工作流进行求解,表 3 给出不同用户偏好的求解结果. 用户偏好为(1,0,0) 表示用户只关注运行时间,对费用和可靠性无要求. 该情况下 EvoWF 可与时间贪婪算法进行对比,实验表明 EvoWF 给出的解与时间贪婪算法的解一致. 同理,若偏好为(0,1,0) ,则与费用贪婪算法进行对比,实验结果表明,两者给出的解是一致的.

从表 3 看出,随着用户偏好的变化,EvoWF 给出不同的解,解的各个 QoS 属性值也在做相应的变化. 从迭代次数上看,EvoWF 效率较高. 另外,用户偏好对迭代次数有 2 种影响: 对于单一偏好,算法能够较早停止,因为算法能找到满足 B_f 条件的解; 对于多 QoS 偏好,算法运行速度无明显变化.

对于求解结果,本文通过对本例中的 128 个解穷举验证,算法给出的是正确解,因此,EvoWF 是可以对工作流进行调度的.

表 3　不同的 QoS 偏好下对图 1 的求解结果

Table 3　Solving results of Fig. 1 with different QoS preferences

偏好	节点调度序列	时间	费用	可靠性	迭代次数
(1 0 0) / 时间贪婪	{ 1 3 5 7 9 11 14}	67	860	0. 5753	5
(0 1 0) / 费用贪婪	{ 2 3 6 8 10 12 13}	202	435	0. 7662	6
(0 0 1)	{ 1 4 6 8 10 12 13}	201	545	0. 7950	1
$(\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3})$	{ 1 3 6 8 10 12 14}	161	540	0. 7750	18
(0. 8 0. 1 0. 1)	{ 1 3 6 8 9 11 14}	71	700	0. 5719	11
(0. 1 0. 8 0. 1)	{ 2 3 6 8 10 12 13}	202	435	0. 7662	10
(0. 1 0. 1 0. 8)	{ 1 3 6 8 10 12 13}	201	445	0. 7813	13

4.3　本文算法分析及与其他算法对比

4.3.1　遗传算法及蚁群算法的设计

本文算法与文献 [6] 采用的遗传算法(Genetic Algorithm, GA) 及文献 [10] 的蚁群算法(Ant Colony Optimization, ACO) 进行对比. 遗传算法与蚁群算法按照各自文献的描述设计,3 点说明如下.

1) 实验中的遗传算法所采用的适应值函数是基于 QoS 偏好的适应值函数,因为除了基于 QoS 偏好的适应值函数更加符合用户实际需要外,本文对文献 [6] 中的适应值函数进行分析,得出该函数和本文提出的适应值函数是按比例正相关的,即采用文献 [6] 的适应值函数,个体适应值若为大,则本文

的适应值亦为大,在诱导进化上是一致的. 另外,引入最大个体无变化机制来控制迭代的次数.

2) 蚁群算法中没有 QoS 偏好的概念,但需要用户选择一种 QoS 作为偏好,在不同的 QoS 偏好下,采用不同的启发式信息进行求解. 本文设计的蚁群算法,为利用 QoS 偏好的概念,采用对原有启发式算法进行加权的方式,扩展文献[10]的算法,即原来的算法只能求解偏好为(1,0,0),(0,1,0),(0,0,1)这 3 种情况,现在则可求解所有的 QoS 偏好. 另外,对蚁群算法给出的结果采用式(1)重新计算后与本文算法进行对比.

3) 算法中主要参数取值按照两篇文献进行取值,即对于遗传算法交叉和变异率是 0.5,对于蚁群算法选择阈值 $q_0 = 0.9$ 且启发式和激素影响参数 $\beta = 1.2$. 另外,算法中个体(蚂蚁)的个数都是 10,最大迭代次数与 EvoWF 相等.

4.3.2 算法分析

为了解算法的详细求解过程,本文着重研究算法迭代过程中种群平均适应度与最佳个体适应度的变化过程. 为了图片易于观察,本文采用适中数据量来进行测试. 实验中,随机生成节点个数为 15 的工作流,每个任务节点的服务个数在[1,10]间随机选取,用户偏好设置为(1/3,1/3,1/3).

图 3 显示 EvoWF 与遗传算法的一次运行结果. 从图中看出,EvoWF 种群的平均适应度小于遗传算法的平均适应度,说明 EvoWF 具有较强的搜索性,而遗传算法则随着迭代次数的增加,种群的平均适应度不断上升,说明一个适应值较高的个体的多个拷贝分布在种群中,不利于新的最优解的发现.

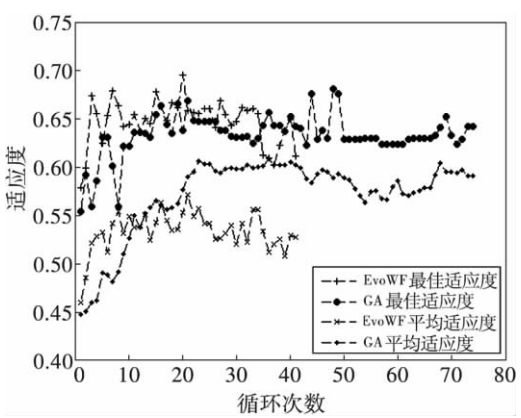


图 3 本文算法与遗传算法种群平均适应度和最佳个体适应度随迭代次数变化的对比

Fig. 3 Comparison of average fitness and best individual fitness between EvoWF and GA during the iteration

图 4 为 EvoWF 与蚁群算法的平均适应度和最佳个体适应度对比,图中看出,EvoWF 的种群平均适应度和最优个体适应度波动都比蚁群算法大,说明迭代一定次数后 EvoWF 比蚁群算法陷入局部极值的可能性小.

从图 3 和图 4 知,EvoWF 相比其它两种算法,发现最优个体的能力较强,同时陷入局部极值的风险较小.

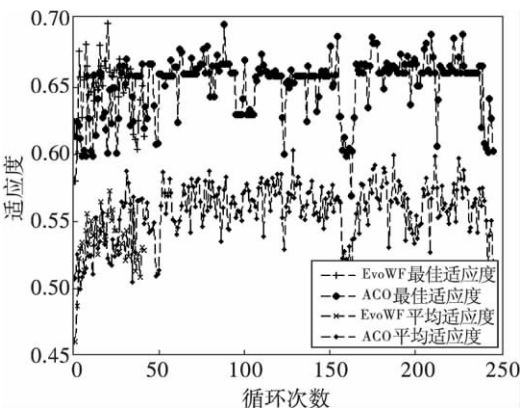


图 4 本文算法与蚁群算法种群平均适应度和最佳个体适应度随迭代次数变化的对比

Fig. 4 Comparison of average fitness and best individual fitness between EvoWF and ACO during the iteration

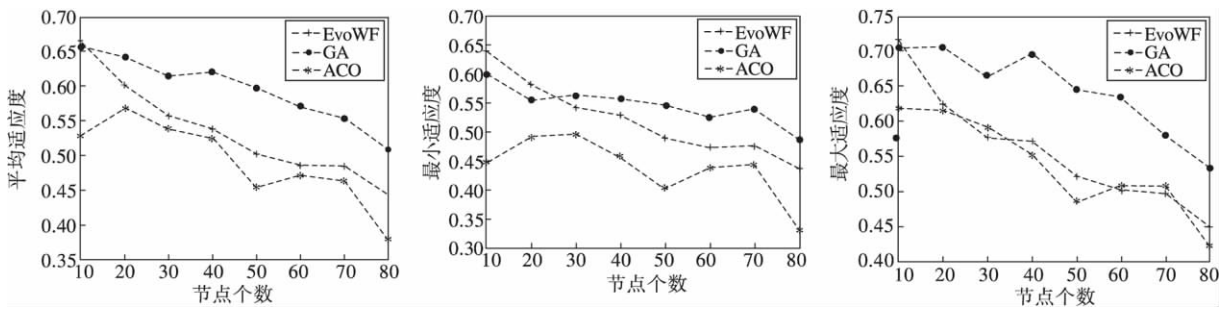
4.3.3 大规模数据上的对比

通过一次的运行结果并不能说明 EvoWF 算法优于其他两种算法,因此本文对任务节点分别为 10,20,…,80 的任务集进行求解,随机生成用户偏好. 每次运行,随机生成任务集,同时为每个任务随机生成一个小于 10 个服务的列表. 针对每个任务算法运行 100 次,图 5 显示随着节点个数的变化,3 种算法给出的求解结果的平均适应值、最佳适应值,最差适应值. 从图中看出,3 种算法的平均求解精度较为接近,在较小空间时,EvoWF 在精度上具有优势. 同时在最差适应值上,EvoWF 的精度较高,最佳适应值上,遗传算法则比较好. 图 6 显示随着节点个数的变化,算法迭代次数的变化. 从图中看出蚁群算法迭代次数较多,因为蚁群算法中没有最优个体无变化停止机制,而 EvoWF 在迭代次数上全面优于其他两种算法. 因此,得出结论 EvoWF 在求解速度上优于其他两种算法,同时在解空间较小时(实验表明小于 10^{15} 次),在精度上也具有优势.

4.3.4 较小求解空间下的详细求解结果

从上面的实验中知道 EvoWF 性能较佳,即与其

它两种算法相比迭代次数较少，在解空间较小时(实验表明小于 10^{15} 时) EvoWF 在性能和精度两个方面俱佳，但是文献[6]、[10]都没有使用 QoS 偏好概念，存在由于本文在两个算法中引入 QoS 偏好的概念所导致算法性能的下降. 故为研究算法在没有 QoS 偏好的情况下的性能，进一步详细了解算法的求解结果，本文给出解空间大小为 10^5 ~ 10^{15} 的求解结果. 表 4 ~ 表 6 分别展示 QoS 偏好为(1,0,0)，(0,1,0)，(0,0,1) 时算法运行 100 次的求解结果. 限于篇幅，表 7 给出 3 种 QoS 偏好下算法的平均迭代次数. 在这 3 种偏好下，遗传算法和蚁群算法都是文献本身的算法，不受本文的定义 2 影响.

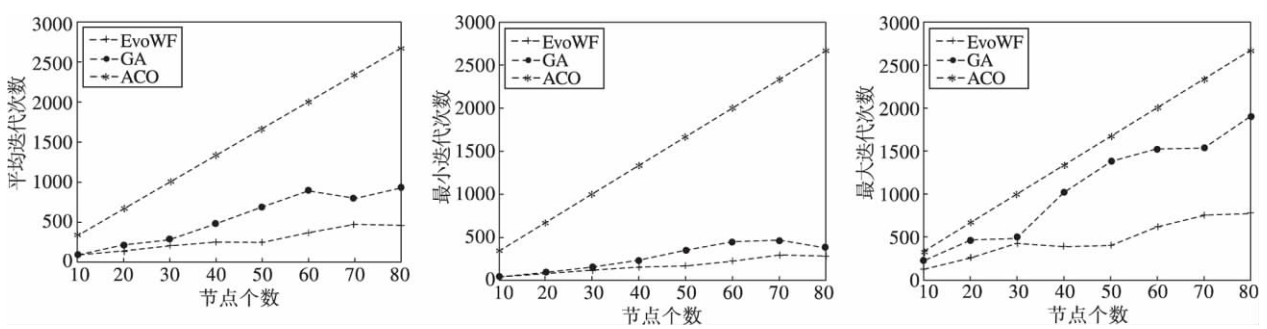


(a) 平均适应度　(b) 最小适应度　(c) 最大适应度

(a) Mean fitness　(b) Min fitness　(c) Max fitness

图 5　3 种算法求解精度对比

Fig. 5　Fitness comparison of EvoWF, GA and ACO



(a) 平均迭代次数　(b) 最小迭代次数　(c) 最大迭代次数

(a) Mean Loop number　(b) Min loop number　(c) Max loop number

图 6　3 种算法迭代次数对比

Fig. 6　Iteration comparison of EvoWF, GA and ACO

表 4　3 种算法的最佳、平均、最差运行时间对比(QoS 偏好为(1,0,0))

Table 4　Comparison of best, average and worst running time of 3 algorithms (QoS preference (1,0,0))

节点数	最佳运行时间			平均运行时间			最差运行时间		
	EvoWF	GA	ACO	EvoWF	GA	ACO	EvoWF	GA	ACO
5	266	266	429	292.6	487.2	1022.5	398	742	2192
6	1099	1099	1099	1116.7	1328.4	1524.0	1229	1583	2062
7	657	657	1165	729.9	1150.4	1417.0	898	1994	1727
8	812	876	1151	847.5	1157.6	1682.8	1011	1634	2244
9	532	533	779	585.6	872.7	1053.3	732	1163	1272
10	520	638	659	571.6	957.2	911.4	745	1178	1077
11	629	693	1079	654.9	1001.4	1539.8	729	1552	1820
12	641	806	1409	887.1	1040.2	1943.6	1223	1269	2532
13	1324	1495	2297	1349.0	1754.6	3177.9	1379	2417	3768
14	683	688	1154	726.2	1134.5	1902.0	834	1449	2672
15	845	845	1354	920.5	1134.2	1784.7	1213	1509	1977
平均	728.00	781.45	1143.18	789.24	1092.58	1632.63	944.63	1499.09	2122.09

表 5 3 种算法的最佳、平均、最差运行费用对数(QoS 偏好为(0,1,0))

Table 5 Comparison of best, average and worst cost of 3 algorithms (QoS preference (0,1,0))

节点数	最佳运行费用			平均运行费用			最差运行费用		
	EvoWF	GA	ACO	EvoWF	GA	ACO	EvoWF	GA	ACO
5	450	450	450	457.4	610.6	512.4	499	1188	992
6	403	403	672	425.9	600.6	1085.5	494	942	1832
7	267	308	267	308.8	573.0	761.2	440	872	1701
8	663	699	922	701.6	950.7	1452.8	771	1238	1959
9	750	774	1065	789.5	1092.4	1883.5	840	1787	2497
10	658	189	934	709.8	1066.7	1242.0	753	1712	1793
11	1211	1318	1961	1284.6	1748.1	2465.6	1671	2569	2946
12	2362	2458	2818	3069.2	3078.9	3428.9	3619	3755	4537
13	1719	1749	2621	1814.6	2355.2	3240.0	2547	3056	3630
14	1396	1459	2640	1530.1	2033.0	3300.4	2331	2922	3925
15	1186	1491	2293	1978.4	2032.1	3801.0	2521	2599	4718
平均	1005.91	1027.09	1513.00	1188.17	1467.39	2106.66	1498.73	2058.18	2775.45

表 6 3 种算法的最佳、平均、最差运行稳定性对比(QoS 偏好为(0,0,1))

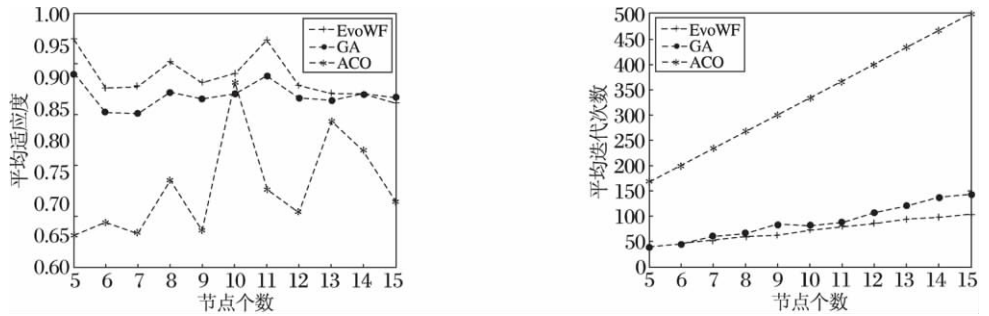
Table 6 Comparison of best, average and worst reliability of 3 algorithms (QoS preference (0,0,1))

节点数	最佳运行稳定性			平均运行稳定性			最差运行稳定性		
	EvoWF	GA	ACO	EvoWF	GA	ACO	EvoWF	GA	ACO
5	0.9794	0.9745	0.9497	0.9754	0.9590	0.8873	0.9610	0.9422	0.8295
6	0.9533	0.9533	0.9198	0.9507	0.9302	0.8659	0.9410	0.9022	0.8046
7	0.9652	0.9652	0.8789	0.9579	0.9373	0.8410	0.9330	0.8863	0.8040
8	0.9271	0.9271	0.8878	0.9143	0.8834	0.7858	0.8697	0.8274	0.7302
9	0.9326	0.9221	0.9080	0.9254	0.8949	0.8719	0.8955	0.8332	0.8488
10	0.9254	0.9202	0.8626	0.9104	0.8907	0.8133	0.8760	0.8533	0.7747
11	0.8957	0.8871	0.8375	0.8745	0.8623	0.7792	0.8342	0.8290	0.7394
12	0.8523	0.8953	0.8545	0.8232	0.8513	0.7973	0.7760	0.7996	0.7642
13	0.8749	0.8836	0.8514	0.8226	0.8510	0.7908	0.7692	0.8155	0.7194
14	0.8599	0.8779	0.7815	0.8243	0.8438	0.7400	0.7806	0.8003	0.7021
15	0.8230	0.8884	0.7222	0.7381	0.8509	0.6798	0.7017	0.7846	0.6200
平均	0.9081	0.9177	0.8594	0.8833	0.8868	0.8048	0.8489	0.8431	0.7579

表 7 3 种算法的最佳、平均、最差迭代次数对比(QoS 偏好(1,0,0)(0,1,0)(0,0,1))

Table 7 Comparison of average best, average and worst iteration time of 3 algorithms (QoS preferences (0,0,1),(0,1,0),(0,0,1))

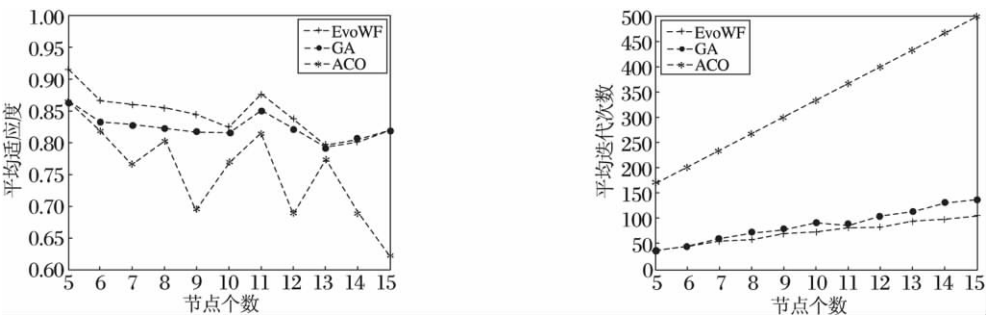
节点数	最佳迭代次数			平均迭代次数			最差迭代次数		
	EvoWF	GA	ACO	EvoWF	GA	ACO	EvoWF	GA	ACO
5	9.00	13.67	167	29.20	37.40	167	56.00	70.33	167
6	15.33	24.67	200	41.49	46.85	200	78.33	86.33	200
7	20.33	30.67	233.00	4974.00	59.17	233	88.00	112.67	233
8	26.67	30.00	266	63.36	67.48	266	133.00	132.67	266
9	31.67	40.00	299	72.48	73.10	299	146.33	140.67	299
10	33.00	41.33	333	89.60	81.25	333	142.00	195.67	333
11	39.33	48.00	366	88.48	101.00	366	171.33	184.67	366
12	51.00	55.00	366	89.88	109.60	366	174.00	205.67	366
13	57.67	62.67	432	107.38	129.41	432	186.33	267.67	432
14	57.00	59.67	466	119.10	130.98	466	209.33	302.67	466
15	54.33	74.00	499	90.65	148.96	499	148.67	308.00	499
平均	35.94	43.61	329.72	524.15	89.56	329.72	139.39	182.46	329.72



(a1) 平均适应度　　(a2) 平均迭代次数

(a1) Mean fitness　　(a2) Mean loop number

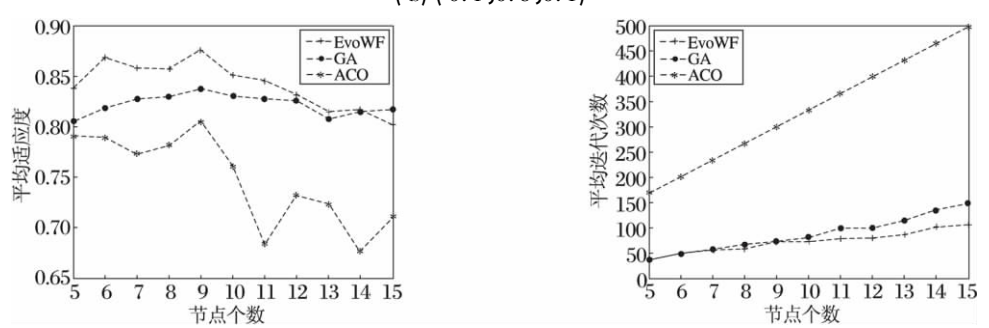
(a) (0.8,0.1,0.1)



(b1) 平均适应度　　(b2) 平均迭代次数

(b1) Mean fitness　　(b2) Mean Loop number

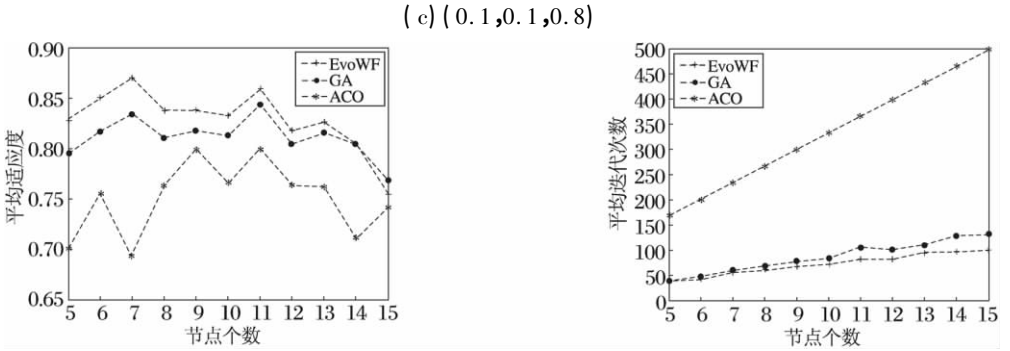
(b) (0.1,0.8,0.1)



(c1) 平均适应度　　(c2) 平均迭代次数

(c1) Mean fitness　　(c2) Mean Loop number

(c) (0.1,0.1,0.8)



(d1) 平均适应度　　(d2) 平均迭代次数

(d1) Mean fitness　　(d2) Mean Loop number

(d) $(\frac{1}{3},\frac{1}{3},\frac{1}{3})$

图 7　3 种算法在不同偏好下的求解精度与迭代次数对比

Fig. 7　Comparison of fitness and iteration of 3 algorithms with different preferences

从表4、表5中看出，EvoWF全面优于其他两种算法，表6中解空间大于10^{12}次时，遗传算法的求解结果要略优于EvoWF. 表7中，EvoWF的迭代次数小于其它两种算法. 因此，在解空间小于10^{15}时，EvoWF在求解性能和精度上优于其他两种算法.

4.3.5 不同QoS偏好下的算法运行结果

本文研究QoS偏好为(0.8,0.1.0.1)，(0.1,0.8,0.1)，(0.1,0.8,0.1)，(1/3,1/3,1/3)下的算法求解结果. 随机生成节点个数为5,6,…,15的DAG，每个节点的服务个数为10. 图7显示3种算法的运行结果，从图中看出，在解空间大于10^{14}时，EvoWF在求解精度上会出现稍微弱于GA的情况，而在其他情况下，EvoWF则全面优于其他两种算法. 另外，本文引入的最优个体无变化机制能使算法有效收敛，蚁群算法中没有采用该机制，因此该算法迭代次数较多.

5 结束语

本文采用克隆选择算法求解网格环境下的工作流调度问题. 实验表明，在单一QoS约束下，EvoWF能获得最优解. 在多QoS约束下，EvoWF在效率上优于基于遗传算法和蚁群算法的调度算法，并且在解空间小于10^{15}次时，在精度上也具有优势. 同时，算法提出的QoS偏好概念，使得算法更符合用户的实际需求. 通过大量实验优化算法参数.

下一步将对算法求解结果的精度进行更深入的分析和改进，并在实际使用中进行更详细的测试，以进一步改进算法的求解精度.

参 考 文 献

[1] Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001, 15(3): 200-223

[2] Yu J, Buyya R. A Taxonomy of Workflow Management Systems for Grid Computing. Journal of Grid Computing, 2006, 3(3): 171-200

[3] Deelman E, Gannon D, Shields M, *et al*. Workflows and E-Science: An Overview of Workflow System Features and Capabilities. Future Generation Computer Systems, 2009, 25(5): 528-540

[4] Wieczorek M, HoheiseL A, Prodan R. Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid. Future Generation Computer Systems, 2009, 25(3): 237-256

[5] Yuan Yingchun, Li Xiaoping, Wang Qian, *et al*. Cost Optimization Heuristics for Grid Workflows Scheduling Based on Serial Reduction. Journal of Computer Research and Development, 2008, 45(2): 246-253 (in Chinese)
(苑迎春,李小平,王 茜,等. 基于串归约的网格工作流费用优化方法. 计算机研究与发展, 2008, 45(2): 246-253)

[6] Yu Jia, Buyya R. Scheduling Scientific Workflow Applications with Deadline and Budget Constraints Using Genetic Algorithms. Scientific Programming, 2006, 4(3): 217-230

[7] Jin Hai, Chen Hanhua, Lü Zhipeng, *et al*. QoS Optimizing Model and Solving for Composite Service in CGSP Job Manager. Chinese Journal of Computers, 2005, 28(4): 578-587 (in Chinese)
(金 海,陈汉华,吕志鹏,等. CGSP作业管理器合成服务的QoS优化模型及求解. 计算机学报, 2005, 28(4): 578-587)

[8] Zhang Weizhe, Hu Mingzeng, Zhang Hongli, *et al*. A Multiobjective Evolutionary Algorithm for Grid Job Scheduling of Multi-QoS Constraints. Journal of Computer Research and Development, 2006, 43(11): 1855-1862 (in Chinese)
(张伟哲,胡铭曾,张宏莉,等. 多QoS约束网格作业调度问题的多目标演化算法. 计算机研究与发展, 2006, 43(11): 1855-1862)

[9] Chen Weineng, Zhang Jun. An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem with Various QoS Requirements. IEEE Trans on Systems, Man and Cybernetics, 2009, 39(1): 29-43

[10] Garrett S. How Do We Evaluate Artificial Immune Systems? Evolutionary Computation, 2005, 13(2): 145-177