云计算中基于改进的布谷鸟算法的资源调度①

陈海涛

(中国食品药品检定研究院, 北京 100050)

摘 要: 云计算的资源调度一直以来都是研究的重点, 引入布谷鸟算法来解决资源分配问题, 首先描述云计算资源模型, 其次针对该算法存在局部收敛速度快, 容易造成局部最优值的问题, 采用三个方面来改进, 其一采用变长因子进行调整, 减小探索求解质量之间的差别; 其二使用差分变异策略更新鸟窝位置; 其三使用基于Coelho的混沌全局搜素和局部搜索避免了Levy的随意扰动. 通过测试函数说明表明本文算法的性能优于基本布谷鸟算法, Cloudsim仿真平台说明本文的算法在消耗时间, 成本和用户满意度方面具有明显的优势.

关键词: 云计算; 布谷鸟算法; 变长因子; 差分策略

Resource Scheduling Based on Improved Cuckoo Algorithm in Cloud Computing

CHEN Hai-Tao

(National Institutes for Food and Drug Control, Beijing 100050, China)

Abstract: Resource scheduling in cloud computing has always been the focus of research, thus the cuckoo algorithm is introduced in this paper to solve the problem of resource allocation. The resource model of cloud computing is described at first. Then, aiming at the problem that this algorithm is easy to cause local optimal value with fast local convergence speed, this model is improved from the following three aspects. Firstly, the variable-length factor is adopted to make the adjustment and reduce difference between the quality of solutions. Secondly, differential mutation strategy is used to update the bird nest's location. Thirdly, chaotic global search and local search are used based on Coelho to avoid the random disturbance of Levy. It is shown through the test functions that the algorithm proposed in this paper has superior performance than the basic cuckoo algorithm, and the Cloudsim simulation platform shows that algorithm in this paper has obvious advantages in the consumption of time, costs and users' satisfaction.

Key words: cloud computing; cuckoo algorithm; variable-length factor; differential strategy

云计算是近年来一种利用互联网可以随地, 随时, 按需, 便捷, 快速的访问共享资源的一种计算模式, 能够有效的实现软硬件的资源共享[1]. 由于在实际过程中资源数目远远小于任务的数量, 因此如何合理的分配资源是云计算中分配主要问题. 大量的研究表明, 云计算的资源分配问题其实一种多任务目标的 NP 问题. 为了解决这个问题, 国内外学者将人工智能算法引入到资源分配中起到了非常好的效果. 文献[2]提出一种基于膜计算的蝙蝠算法, 通过在辅助膜内进行蝙蝠的个体局部寻优, 将优化后的个体传送到主膜间进行全局优化, 从而达到了云计算资源优化分配要求. 本算法能够有效的提高了云计算环境下的系统处理时间和效率. 文献[3]设计了一种基于Q学习和双向ACO算法的云计算任务资源分配模型, 设计一种结合前向蚂蚁和后向蚂蚁的双向 ACO 算法实现任务资源的最终分配, 仿真实验表明, 本文算法能够有效实现云计算异构环境下的任务资源分配. 文献[4]提出一种基于资源状态蚁群算法, 加入虚拟机实时状态, 更精确地表达云计算任务分配的问题, 仿真实验表明本文算法的稳定性得到了有效的提高. 文献[5] 提出一种基于改进人工蜂群算法的云计算资源调度模型(IABC).采用人工蜂群算法求解云计算模型, 并将个体当前最优

① 收稿时间:2015-04-13;收到修改稿时间:2015-06-08

值及随机向量引入到蜂群搜索过程中, 加快搜索速度, 提高搜索能力, 仿真结果表明, IABC 算法提高了云计算资源利用率, 而且大幅度减少了任务的完成时间. 文献[6]构建了一种基于动态平衡点动态变化的云资源调度算法. 通过 CloudSim 平台下的仿真实验表明, 该方法比基于服务驱动的调度方法具有更快的响应、使用更少的虚拟机、能获得更大的收益. 文献[7,8]分别将改进的遗传算法和粒子群算法运用到资源分配中, 起到了一定的效果.

本文首先描述了云计算的资源调度模型, 在云计算资源中引入布谷鸟算法来进行解决, 针对该算法存在局部收敛速度快, 容易造成局部最优值的问题进行改进, 首先针对变长因子进行调整, 减少探索求解质量之间的差别, 其次引入差分变异策略更新鸟窝位置, 最后采用基于 Coelho 的混沌全局搜素和局部搜索, 避免了 Levy 的随意扰动. 仿真实验表明本文算法在解决云计算资源分配具有比较好的效果.

1 云计算资源调度描述

云计算中的任务调度可以描述为将 n 个相互独立的子任务分配给 m 个资源($m<n$), 其中任务表示为 $Task=\{task_1,task_2,\cdots task_n\}$, $task_i$ 表示第 i 个子任务, 资源数目为 $\mathrm{Re}source=\{resource_1,resource_2,\cdots resource_n\}$, $resource_i$ 表示第 i 个资源, 因此每一个子任务只能在一个虚拟资源上执行, 任务 $Task$ 和 $\mathrm{Re}source$ 之间的关系适用矩阵来进行表示:

$$X=\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \tag{1}$$

公式(1)中 x_{ij} 表示子任务 i 与第 j 个资源的关系, 其值为 0 或者 1, 当值为 0 的时候, 表示子任务 i 没有占据第 j 个资源, 反之则占据. ETC_{ij} 表示子任务 i 在第 j 个资源上的理论上所需要的时间, 因此 ETC 矩阵表示如下:

$$ETC=\begin{pmatrix} ETC_{11} & \cdots & ETC_{1n} \\ \vdots & \ddots & \vdots \\ ETC_{m1} & \cdots & ETC_{mn} \end{pmatrix} \tag{2}$$

同理, 可以使用 RCU_{ij} 来表示子任务 i 在第 j 个资源上的理论上所需要的执行成本, 因此, RCU 矩阵表示如下:

$$RCU=\begin{pmatrix} RCU_{11} & \cdots & RCU_{1n} \\ \vdots & \ddots & \vdots \\ RCU_{m1} & \cdots & RCU_{mn} \end{pmatrix} \tag{3}$$

根据 ETC 矩阵和 RCU 矩阵可以得到各个资源节点在执行完所分配的子任务花费的时间和成本.

$$sumTime(i)=\max\sum_{i=1}^{resouce}\sum_{j=1}^{n}T(i,j)\times ETC(i,j) \tag{4}$$

$$sumCost(i,j)=\sum_{i=1}^{resource}\sum_{j=1}^{n}resource(i,j)\times RCU(i,j) \tag{5}$$

式中, $T(i,j)$ 表示子任务 i 中的第 j 个子任务的所需要时间, $resource(i,j)$ 表示子任务 i 中的第 j 个资源的所需要成本, 公式(4)和公式(5)分别表示任务完成的总时间和消耗的总成本. 本文利用加权模型来实现对总时间和总成本的统一权衡, 综合以上考虑因素, 得到云资源下的目标适应度函数:

$$F=\min(\alpha\cdot sumTime+\beta\cdot sumCost) \tag{6}$$

式中, α 和 β 分别表示任务完成总时间, 任务完成总成本的加权系数, $\alpha+\beta=1$.

2 基本知识描述

2.1 基本布谷鸟算法

布谷鸟算法(Cuckoo Search, 简称 CS 算法)中的布谷鸟在飞行的过程中主要采用随机方式的来选择新的地方作为鸟窝, 但是这种随机方式主要通过 Levy 飞行路径的过程中产生候选鸟窝和采用最佳保留策略不断更新当前的鸟窝位置, 最终使得鸟窝位置能够的达到或者接近全局最优解, Yang[9]提出了一种多目标的布谷鸟算法能够适应满足多目标优化问题, 定义布谷鸟算法三种理想状态:

1) 第 i 只布谷鸟随机选定一个寄宿鸟窝, 产生 1 个蛋.

2) 随机选择一组宿主鸟窝, 将最好的宿主鸟窝保留到下一代

3) 寄宿鸟发现鸟窝中外来的蛋的概率是 P , 倘若发现会丢弃外来的蛋或者舍弃该鸟窝继续搭建另一个鸟窝.

因此, 在这三个理想的状态的基础上, 布谷鸟寻找鸟窝的路径和位置更新公式如下:

$$x_i^{(t+1)}=x_i^{(t)}+\alpha\otimes L(\lambda) \tag{7}$$

其中 $x_i^{(t)}$ 表示第 i 个鸟窝在 t 次迭代中的鸟窝的位置; $\otimes$ 为点对点乘法, α 为步长, $L(\lambda)$ 为 Levy 随机搜索路径. 通过位置更新后, 使用随机数 r 与 P 进行比较, 如果 $r > P$, 则采用 $x_i^{(t+1)}$ 代替 $x_i^{(t)}$ 进行改变, 反之不变, 并保留目前为止的鸟窝位置记作 $x_i^{(t+1)} \to y_i^{(t+1)}$.

2.2 云计算资源下的基本布谷鸟算法局限性

布谷鸟算法中的步长的设定 0 由于具有一定的随机性且不易控制, 因此容易导致算法陷入局部最优, 加快整个算法的收敛速度, 这样算法的精度受到严重的影响, 无法产生最优的蛋. 在云计算中, 最优的蛋对应的最优的解, 无法产生最优的蛋导致云计算中无法得到目标函数的最小值, 因此对布谷鸟算法进行改进使得算法的性能得到提高, 提高算法分配效率.

3 基于改进的布谷鸟算法在云计算中资源分配

3.1 步长改进

步长决定了 CS 算法中下一个布谷鸟的选择探索的位置的长度, 为了很好的调整探索求解质量之间的差别, 本文设置如下:

$$\alpha=\alpha_0\cdot(\min\{(x_{j1}^t-x_{j0}^t),(x_{j2}^t-x_{j0}^t),\cdots;(x_{jk}^t-x_{j0}^t)\}) \quad (8)$$

式中, α_0 为步长初始因子, x_{j0}^t 和 x_{jk}^t 分别表示在 t 次迭代中的当前解和第 k 个解, $\min\{(x_{j1}^t-x_{j0}^t),(x_{j2}^t-x_{j0}^t),\cdots,(x_{jk}^t-x_{j0}^t)\}$ 表示这些 k 个解与初始值之间的距离的最小值来作为步长参考. 这样的优点是可以从最小的距离中来进行考虑, 最大限度的避免因为步长过大而产生的算法最优解的遗漏.

3.2 变异策略

为了进一步增加 CS 算法的探索能力, 本文使用经典的 DE 变异策略来更新鸟窝的位置.

$$V_i=x_{r1}+\lambda_1\cdot(x_{r2}-x_{r3})+\lambda_2\cdot(x_{r4}-x_{r5}) \quad (9)$$

在第 t 次迭代中, 随便选取 4 个与第 i 个鸟窝不同的其他鸟窝, 即 x_{r1}^t, x_{r2}^t, x_{r3}^t, x_{r4}^t, 用来新建第 i 个鸟窝的位置, 结合公式(9)得到如下

$$V_i=x_{r1}^t+\lambda_1\cdot(y_c^t-x_{r2}^t)+\lambda_2\cdot(x_{r3}^t-x_{r4}^t) \quad (10)$$

式中, $y_c^t=(y_c^t,\alpha_i^t)^T$, y_c^t 表示当前的最优鸟窝的位置, $V_i=(v_{i1},v_{i2},\cdots v_{in})^T$, v_{in} 作为步长因子, 然后通过概率选择鸟窝是否作为备选鸟窝, 如公式(11).

$$w_{ij}=\begin{cases}v_{ij}, r_2\in[0,1]\\ x_{ij}^t, else\end{cases} \quad (11)$$

令 $W_i=(t_{i1},t_{i2},\cdots t_{in})$, 通过评选备选的鸟窝的位置, 来决定是否替换原来的鸟窝, 公式如下.

$$x_i^{(t+1)}=\begin{cases}W_i & finess(W_i)<finess(x_i^t)\\ x_i^t & else\end{cases} \quad (12)$$

3.3 基于 Coelho 的混沌全局搜素和局部搜索

CS 算法中的搜素主要是基于 Levy 行为的扰动搜索策略, 具有最大的特点就是使得搜索具有很好的搜索性, 防止算法陷入局部最优, 但存在搜索区域大的同时求解精度低的问题. 对 Levy 行为采用基于 Coelho 的全局搜索和局部搜搜, 这样可以能够使得算法的性能得到平衡下的最大化.

设定全局搜索最大迭代次数为 L_G, 局部搜索最大迭代次数为 L_M, N 为搜索的停止标准, α 为局部搜索步长. X^* 为搜索当前的最优解, f^* 为目标函数值, 初始化迭代次数 k.

全局混沌搜索算法

```
Begin
  while k ≤ L_G
    x_i(k) = M_i + z_i(k)·(U_i − M_i), i = 1,... n
    if f(X(k)) < f* then
      X* = X(k)
      f* = f(X(k))
    endif
    k = k + 1
endwhile
End
```

局部混沌搜索算法

```
Begin
  while k ≤ L_M
    for i = 1 to n
      if rand < 0.5 then
        x_i(k) = x_i* + α · z_i(k)·| U_i − X_i* |
      else if
        x_i(k) = x_i* − α · z_i(k)·| X_i* − M_i |
      endif
    Endfor
    if f(X(k)) < f* then
      X* = X(k)
      f* = f(X(k))
    endif
    k = k + 1
  Endwhile
```

以上这两种混沌搜索方法同时在两个空间中进行搜索, 这是一种相对有效的混沌搜索算法, 可以保证在搜索区域大的时候算法的精度保持稳定.

3.4 算法的步骤

步骤 1: 初始化 CS 算法中的相关参数, 确定宿主鸟窝的数目, 发现概率鸟蛋的概率 P, 设定全局最大迭代次数 Max, 布局迭代次数小于 Max, 步长初始 α_0, 随机选择鸟窝的位置 $x_i=(x_{i1},x_{i2},\cdots x_{in})^T$, 设定 $t=0$, 鸟窝的最佳位置就是云计算资源的最佳分配方案, 鸟蛋对应的就是云计算的最优解.

步骤 2: 初步使用公式(8)来对步长因子进行改进, 得到新的步长因子 α';

步骤 3: 在局部搜索中结合公式(12)获得鸟窝的位置, 结合公式(8), 得到 $x_i^{(t+1)}$ 和 α_i^{t+1}, 从局部范围了得到暂时最优鸟窝的位置;

步骤 4: 将局部搜索的暂时最佳的鸟窝位置结合 3.3 节全局和局部搜索算法得到最佳的鸟窝位置 $x_{ibest}^{(t+1)}$,

步骤 5: 计算当前最佳鸟窝的适应度值, $t=t+1$;

步骤 6: 如果 $t<Max$, 则转到步骤 2, 否则, 继续执行

步骤 7: 对应的最佳鸟窝的位置中的鸟蛋就是云计算下资源分配的最优解.

4 仿真实验和分析

本实验采用运行环境为处理器为酷睿 i3, 内存为 4GDDR3, Window7 操作系统, 仿真软件为 Matlab2012, 云计算仿真平台为 Cloudsim.

4.1 与其他改进的 CS 算法性能比较

基本 CS 算法中 P_a 设置为 0.25, α 为 1, 文献[11]的 ICS 算法设置 $P_a\in[0.005,1]$, $\alpha\in[0.05,0.5]$, 本文算法中设置 $\alpha_0=0.1$. 以上三种算法设置鸟窝的数量 N 为 30, 设定迭代次数最大为 200, 空间维数最大为 100. 采用表 1 所示的测试函数, 表 2 到表 4 描述了三种算法的优化结果. 图 1 到图 4 是三个算法的收敛效果.

表 1 测试函数

函数	搜索空间	维数
$f_1(x)=\sum_{i=1}^{n}x_i^4+random(0,1)$	[-1.28, 1.28]	100
$f_2(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	[-5.12, 5.12]	100
$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	[-600, 600]	100
$f_4(x)=-(\sum_{i=1}^{n}100(x_{i+1}-x_i^2)^2+(1-x_i)^2)$	[-50, 50]	100

表 2 三种算法对 f_1 函数的优化比较结果

维数	算法	最优值	最差值
30	CS	1.347e-13	5.432e-15
	ICS	2.783e-27	9.143e-23
	本文算法	3.912e-40	7.134e-36
50	CS	1.214e-06	7.323e-07
	ICS	2.417e-04	3.172e-05
	本文算法	4.217e-02	7.182e-01
100	CS	6.362e+01	7.126e+04
	ICS	8.416e+02	9.162e+03
	本文算法	34.1542	1.381e+02

表 3 三种算法对 f_2 函数的优化比较结果

维数	算法	最优值	最差值
30	CS	2.147e-13	4.132e-15
	ICS	3.983e-26	7.743e-23
	本文算法	7.912e-36	9.174e-32
50	CS	2.614e-07	7.323e-09
	ICS	7.427e-03	7.172e-02
	本文算法	5.237e-02	6.192e-01
100	CS	2.962e+02	3.726e+03
	ICS	2.576e+02	3.372e+03
	本文算法	1.127e+02	1.851e+02

表 4 三种算法对 f_3 函数的优化比较结果

维数	算法	最优值	最差值
30	CS	7.947e-08	5.432e-09
	ICS	8.763e-18	9.273e-17
	本文算法	4.912e-25	3.334e-24
50	CS	5.234e-02	6.323e-03
	ICS	9.127e-04	5.782e-05
	本文算法	1.1927e-12	2.182e-11
100	CS	5.6812	6.1452
	ICS	7.2871	17.3036
	本文算法	3.154e-03	4.38e-04

表 5 三种算法对 f_4 函数的优化比较结果

维数	算法	最优值	最差值
30	CS	5.547e-12	5.432e-11
	ICS	3.483e-09	6.273e-10

	本文算法	2.123e-03	3.134e-06
	CS	7.237e-07	9.173e-08
50	ICS	4.159e-04	5.372e-05
	本文算法	2.237e-02	7.182e-01
	CS	6.362e+04	7.126e+05
100	ICS	5.436e+02	9.162e+03
	本文算法	3.1982	9.3192

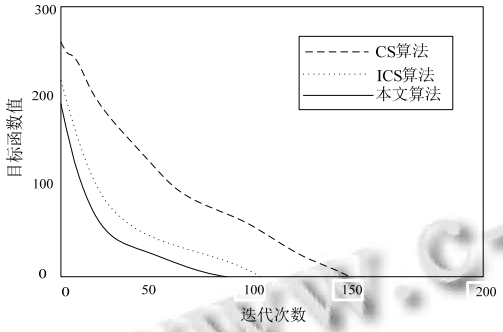

图 1　三种算法在 f_1 的测试结果

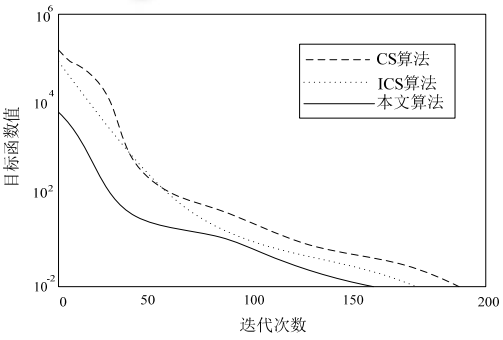

图 2　三种算法在 f_2 的测试结果

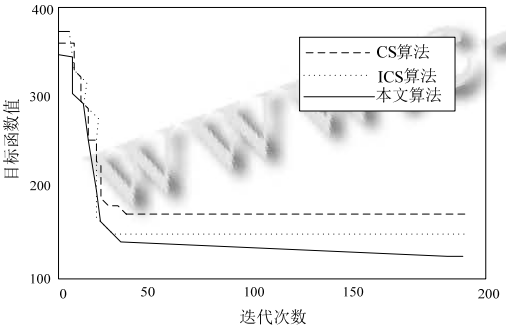

图 3　三种算法在 f_3 的测试结果

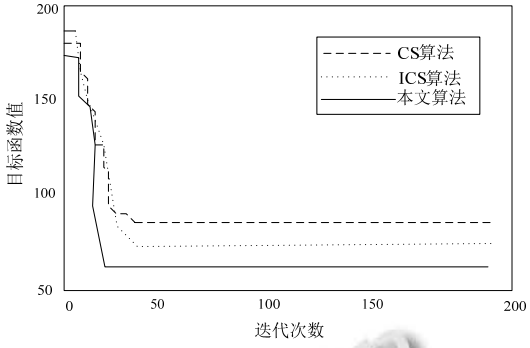

图 4　三种算法在 f_4 的测试结果

4.2 与其他智能算法在云计算下的比较

1.与 CS 算法在云计算下的比较

设定虚拟任务为 100 个，虚拟资源为 10 个，通过比较两种算法下的时间和成本上的消耗可以发现，本文的算法时间花费低于基本 CS 算法，主要是本文算法引入了对步长的改进以及使用全局搜索和局部搜索的方法，使得算法的不必要消耗更多的时间在寻找无效的空间上，使得算法减少了产生最优解的时间，进一步解节约了成本，如图 5 和图 6 所示.

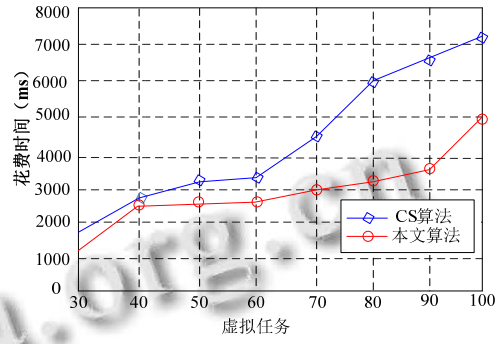

图 5　2 种算法时间消耗比较

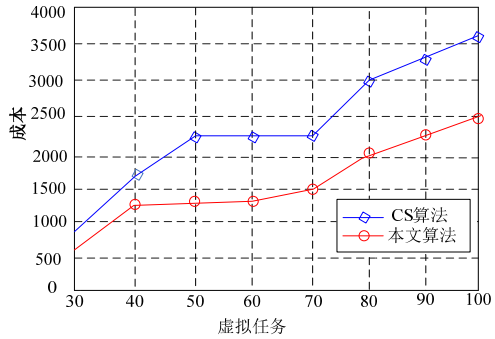

图 6　2 种算法成本消耗比较

2.与其他智能算法进行比较

文献[2], 文献[3]和文献[6]算法是近年来比较新颖的云计算资源调度算法, 选择这些算法与本文的算法进行比较, 设定虚拟任务为 800 个, 虚拟资源为 10 个, 设置迭代次数为 100. 比较结果如下

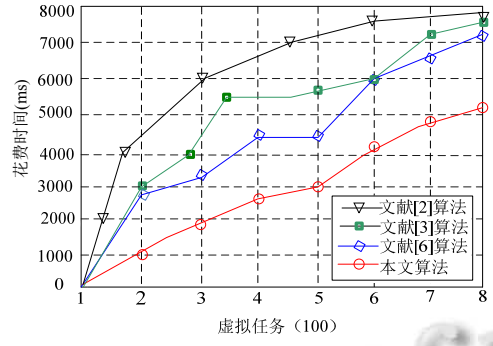

图 7 4 种算法的任务完成时间比较

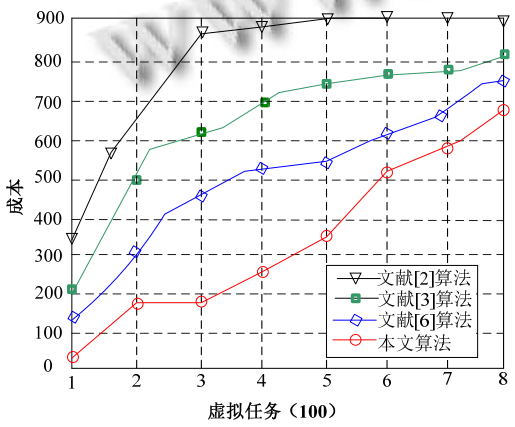

图 8 4 种算法的成本消耗比较

从图 7 中发现, 本文算法在完成时间方面消耗时间是低于其他三种参考文献算法, 同时伴随着任务数目不断的增多, 消耗的时间震荡幅度是最小的, 说明本文的算法能够有效的平衡资源. 从图 8 中发现经过改进后的本文算法在云计算环境中的成本消耗持续方面优于其他的三种参考文献算法. 从而能够更好地满足云计算中的资源调度的要求.

4.3 客户满意度分析

云计算中的客户的满足度反应了云计算资源分配效率的标准, 本文选取了文献[6]和文献[8]的算法来与本文的算法进行比较, 如图 9 所示. 选取表 5 中的不同任务来进行比较, 从图中可以发现本文算法在用户满意度方面优于两种参考算法, 这说明算法在性能上能够对云计算的资源分配起到一定的促进作用, 取得了不错的效果.

表 6 不同任务对应的资源数目

任务类别	任务个数	资源个数
任务 1	100	15
任务 2	1000	20
任务 3	2000	30
任务 4	5000	50
任务 5	10000	75

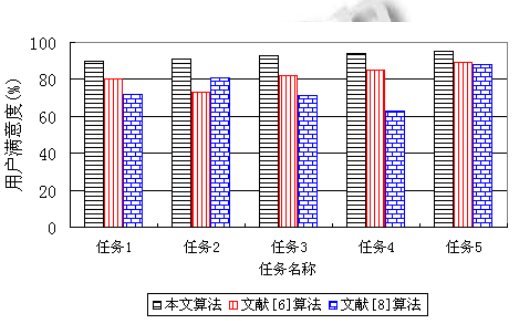

图 9 资源分配效果

5 结语

云计算的资源调度一直以来都是研究的重点, 本文引入布谷鸟算法来进行解决云计算资源分配问题, 针对该算法存在局部收敛速度快, 容易造成局部最优值的问题进行改进, 首先针对变长因子进行调整, 调整探索求解质量之间的差别, 其次引入差分变异策略更新鸟窝位置, 最后采用基于 Coelho 的混沌全局搜素和局部搜索, 避免了 Levy 的随意扰动. 仿真实验表明本文算法的性能优于基本 CS 算法, 同时通过与参考文献算法的比较说明本文算法在任务完成时间, 任务成本消耗方面和用户满意度方面具有一定优势.

参考文献

1 林伟伟,齐德昱.云计算资源调度研究综述.计算机科学,2012,39(10):1–5.

2 宁彬,谷琼,吴钊,等.基于膜计算的蝙蝠算法在云计算资源调度的研究.计算机应用研究,2015,32(3):830–833.

3 孙花,朱锦新.基于 Q 学习和双向 ACO 算法的云计算任务资源分配模型设计.计算机测量与控制,2014,22(10):2243–3346.

4 黄俊,王庆凤,刘志勤.基于资源状态蚁群算法的云计算任务分配.计算机工程与设计,2014,35(9):3305–3309.

5 卓涛,詹颖.改进人工蜂群算法的云计算资源调度模型.微电子学与计算机,2014,31(7):147–150.

6 张爱科,符保龙.基于最大收益平衡点动态变化的云资源调度算法.重庆邮电大学学报(自然科学版),2014,26(5):706–710.

7 朱宗斌,杜中军.基于改进的 GA 的云计算任务调度算法.计算机工程与应用.2013,49,(5):77–80.

8 丁燕艳,等.云计算环境下的 PSO 可信资源调度.计算机工程与应用,2013,49(18):78–81.

9 Yang X, Suash D. Cuckoo search via Levy flight. Proc. of the 2009 World Congress on Nature & Biologically Inspired Computing. Piscataway, NJ. IEEE Press. 2009. 210–214.

10 Valian E, Mohanna S, Tavakoli S. Improved cuckoo search algorithm for global optimizaiton. Int. J. Communciatons and Information Technology, 2011, 1(1): 31–44.

11 王李进,尹义龙,钟一文.逐维改进的布谷鸟搜索算法.软件学报,2013,24(11):2687–2698.