文章编号: 1001-0920(2019)12-2537-10 DOI: 10.13195/j.kzyjc.2018.1675

泛集群环境中计算密集型任务流调度策略

张可佳, 胡亚楠, 李春生†, 富 宇, 李盼池

(1. 东北石油大学 计算机信息与技术学院，黑龙江 大庆 163318；
2. 黑龙江省石油大数据与智能分析重点实验室，黑龙江 大庆 163318)

摘 要: 针对计算节点较多的泛集群环境下难以快速、合理地制定计算密集型任务流调度方案的问题,提出一种基于多目标连续竞买博弈的任务调度策略. 建立多目标优化调度模型,降低多目标优化函数维度,并采用线性加权和法将其转化为总和目标函数,以保证最优解的合理性. 为提高最优解搜索速度,引入ETC矩阵作为最优解表达形式,设计连续竞买博弈算法. 模拟真实场景并通过与同类算法的对比,表明了调度策略在泛集群环境下的响应速度、资源性价比和总成本支出等方面具有明显优势.

关键词: 泛集群环境；计算密集型；任务调度；多目标优化；连续竞买博弈

中图分类号: TP13 **文献标志码:** A

Scheduling strategy of compute-intensive task-flow in generalized cluster

ZHANG Ke-jia, HU Ya-nan, LI Chun-sheng†, FU Yu, LI Pan-chi

(1. College of Computer & Information Technology，Northeast Petroleum University，Daqing163318，China；
2. Heilongjiang Provincial Key Laboratory of Oil Big Data & Intelligent Analisys，Daqing 163318，China)

Abstract: A scheduling strategy based on multi-objective continuous bidding game is proposed to solve the problem that it is difficult to make a quick and reasonable scheduling plan for compute-intensive task-flow in a generalized-cluster with many computing nodes. To ensure the rationality of the optimal solution, a multi-objective optimal scheduling model is established, the dimensions of multi-objective optimization function are reduced, and the multi-objective optimization function is converted into a sum-objective function using the linear weighting method. For improving the search speed of the optimal solution, the ETC matrix is introduced for expressing the form of optimal solution, and continuous bidding game algorithm is designed. By simulating real scenarios and comparing with similar algorithms, it is proved that the scheduling strategy has obvious advantages regarding response speed, resource cost performance and total cost expenditure in the generalized-cluster.

Keywords: generalized-cluster；compute-intensive；task scheduling；multi-objective optimization；continuous bidding game

0 引 言

大规模科学计算(large-scale scientific computing, LSC)任务的调度问题一直是国际上的研究热点,目的是探索如何生成快速、合理的“调度方案”[1],而研究重点侧重于“云计算”“雾计算”等相对封闭的计算环境. 随着边缘计算的发展,泛集群(generalized-cluster, GC)得到了广泛应用. 与封闭式计算环境不同[2],泛集群是一种边缘计算的表现形式[3],是由众多独立的、闲置的计算节点通过网络构成的大型计算环境,计算节点可能是个人电脑、工作站或手机等计算单元,也可能是多个计算单元组成的松散的“计算群体”,目的是提供高性价比的计算能力[4]. 泛集群最基本的特征是不稳定性,同时具有可扩展性、自主性和社会性,以及由多智能体系统理论(multi-agent systems, MASs)的不断发展而延伸出来的自组织能力、学习能力和推理能力.

泛集群突出地强调了可扩展性和廉价性,可扩展性体现在其松散的结构上:任意计算节点都可以加入泛集群并为其效力;而高效利用闲置的资源保证了泛集群低廉的使用成本[5]. 因此,泛集群逐渐成为处理LSC任务的新兴手段,尤其适用于计算密集型LSC任务,这也使越来越多的科研人员开始关

收稿日期: 2018-12-06；**修回日期:** 2019-02-10.
基金项目: 国家自然科学基金面上项目(51774090, 61502094)；黑龙江省教育厅科研专项创新基金项目(2017YDL-12)；东北石油大学青年科学基金项目(2017PYQZL-11).
责任编委: 王凌.
†通讯作者. E-mail: csli_dmis@nepu.edu.cn.

注于研究泛集群环境下的任务调度问题. 这一问题可以描述为: 如何在所有可行的调度方案中, 搜索多目标优化调度问题(multi-objective optimization schedulingproblem,MOSP) 的最优解[6]. 传统的解决方法包括博弈算法和启发式算法.

博弈算法的一种经典应用是将调度问题转化为资源分配问题, 建立博弈模型, 并通过内核、核仁和谈判集等概念反向验证所有可行的调度方案进而取得“最满意解”[7]. 一些研究人员为降低支出成本开始考虑性价比因素的影响, 并引入局部非合作博弈理论[8-9], 论证了竞买博弈在验证性价比因素的有效性, 这也是泛集群环境下研究调度问题的雏形. 近年来, 混合单次博弈算法[10-11] 解决了很多封闭计算环境下的MOSP问题, 实现了理论研究到生产应用的转换. 采用博弈算法解决MOSP问题的方法确实可以取得令人信服的调度方案, 但多数方法为提高验证速度而设计为单次博弈, 其计算复杂度会随着计算节点(成员)数量的增加而激增, 因此在有大量成员参与的泛集群环境下, 其可行性始终受到质疑.

一种采用启发式算法[12-13] 求解MOSP问题的原理如下: 解析多目标函数, 通过线性加权和法将MOSP问题转化为总和目标函数; 改进并采用启发式算法求解总和目标函数的最优解, 启发式算法包括模拟退火、遗传算法、禁忌搜索、粒子群算法、涡流搜索算法等. 近年来, 越来越多的改进型启发式算法和混合型启发式算法[14-15] 逐渐用于解决真实场景下的MOSP问题, 并取得了较好的应用效果, 但现有方法在大量可行解中搜索速度依然较慢, 易陷入局部最优, 因此不完全适用于泛集群环境.

本文针对“如何在成员数目较多时, 快速、合理地搜索MOSP问题的最优解”问题, 提出了一种基于多目标连续竞买博弈算法(multi-objective bidding cooperation game scheduling algorithm, MSA)的任务调度策略. 选用Lattice模型抽象地表示计算密集型LSC任务, 通过泛集群调度系统将Lattice模型生成任务包集合; 建立多目标调度优化模型, 降低多目标优化函数的维度, 并采用线性加权和法转化为总和目标函数, 达到兼顾“顾客期望”和“供应商要求”[7], 满足合理性要求; 引入ETC矩阵作为调度方案的表达形式, 设计MSA算法以提高最优解搜索速度. MSA算法将任务和资源分配过程表示为一个连续竞买博弈模型, 并将总和目标函数作为子博弈的优胜条件, 全部任务的调度过程将在所有子博弈结束时完成. 在油田边缘计算系统中, 根据真实测试和应用效果分析, 验证了调度策略在响应速度、性价比及成本支出等方面具有明显优势, 论证了其有效性和应用价值.

1 准备工作

泛集群的控制和管理是无中心化或局部去中心化的, 一项任务的调度过程是由“临时决策者”或“裁决集团”在分析任务流的基本属性后, 通过泛集群调度系统完成的. 本节总结任务数据流模型, 采用Lattice模型表示计算密集型LSC任务, 并提出泛集群调度系统的框架, 以保证生成为独立任务包集合的可行性.

1.1 任务数据流模型

任务数据流模型可以将一项大规模科学计算任务切割为独立的子任务或任务微粒[16], 常见的任务数据流模型包括: Montage用于建筑学的风洞试验; Ligo用于分析宇宙深处的引力波; Cybershake用于地质地震的数据分析; Epigenomics用于生物学的DNA结构重构; SIPHT用于ERP系统的监控和追踪.

Lattice是一种典型的任务数据流模型框架, 可以表示计算密集型LSC任务[17], 它具有一个确定的高(H)和分支因子(B), H用于表示分支任务的复杂度, B用于表示子任务的层级关系, 也可以表示为(H-B) Lattice模型. 举例说明: 一项油田开发领域的大规模科学计算任务可简述为计算某含油区当月总产油量, 该任务可以分解为, 分别计算该含油区下所有小区块的月总产油量, 并进一步分解为计算所有小区块下每口产油井的月总产油量, 其中每口产油井的月总产油量是一项独立的、包含多次计算的任务微粒, 产油井之间的计算任务不存在交叉. 于是这项大规模科学计算任务可以采用(H-B) Lattice模型表示, 分解后的任务微粒数量与产油井数量相同.

1.2 泛集群调度系统框架

一个完整的泛集群调度系统包括数据流解释器、任务流生成器和任务调度策略. 数据流解释器负责按照业务需要粗粒度划分和切割计算密集型LSC任务, 以构成(H-B) Lattice型的任务数据流结构; 任务流生成器根据任务数据流结构生成任务包列表; 任务调度策略制定出合理的任务调度方案, 其核心是MSA算法和一些辅助组件. 在泛集群环境下, 一次完整的计算密集型LSC任务分布式执行过程可以描述为如下步骤.

step 1: 将计算密集型LSC任务以DAG模型[18]的形式输入到数据流解释器中.

step 2: 初始化DAG模型并提取核心元数据信息, 解释并获得Lattice模型的任务数据流.

step 3: 向任务流生成器中输入Lattice模型的任务数据流.

step 4: 由任务流生成器生成任务包集合.

step 5: MSA算法搜索最优的任务调度方案.

step 6: 泛集群执行任务调度方案并实时监测运行情况,对异常情况作出及时调整和处理.

2　多目标连续竞买博弈算法

本节从“顾客期望”和“供应商要求”的双重角度归纳多目标期望函数,给出MSA算法的设计思想、工作原理和执行过程. 论述过程中出现的符号及解释如下:

M: 任务包的数量;

T: 任务包集合;

S: 泛集群的成员集合;

T_m: 任务包m;

S_i: 泛集群的成员i;

$\overline{p}_i$: S_i的最优支出;

p'_{mi}: T_m消耗的计算资源;

t_{mi}: $S_i(T_m)$的预期完工时间;

e_{mi}: $S_i\,(T_m)$的支出成本;

v_{mi}: T_m的回报/奖励;

o_{mi}: T_m 的工作强度(工作量);

t_i: S_i的预期完工时间;

δ_i: S_i承担的任务包数量;

λ_i: S_i可提供的带宽;

p'_i: S_i已消耗的计算资源;

$p(\Delta,\Pi)$: 性价比目标函数;

$f(\Delta,\Pi)$: 时间目标函数;

$c(\Delta,\Pi)$: 成本目标函数;

$v(\Delta,\Pi)$: 收益目标函数;

w_i: 权重矩阵;

pw_i: 性价比权重;

fw_i: 时间权重;

cw_i: 成本权重;

vw_i: 收益权重;

Δ: 任务分布矩阵;

Π: 计算资源分布矩阵.

2.1　问题描述

定义场景H:一个由C名成员组成的泛集群环境,其调度系统的数据流解释器和任务流生成器已经将一项计算密集型LSC任务转化为任务包集合T. 于是,H可简要描述为制定由“S处理T”的调度方案的过程. 在调度问题中,ETC矩阵是一种表示最优调度方案的常用方法[19-20],任务的分布矩阵和资源分布矩阵分别为

$$\Delta=\begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{array}\!\!\begin{array}{c} \begin{array}{cccc} S_1 & S_2 & \cdots & S_i \end{array} \\ \begin{vmatrix} q_{11} & q_{12} & \cdots & q_{1i} \\ q_{21} & q_{22} & \cdots & q_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mi} \end{vmatrix} \end{array},$$

$$\Pi=\begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{array}\!\!\begin{array}{c} \begin{array}{cccc} S_1 & S_2 & \cdots & S_i \end{array} \\ \begin{vmatrix} v_{11} & v_{12} & \cdots & v_{1i} \\ v_{21} & v_{22} & \cdots & v_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mi} \end{vmatrix} \end{array}. \tag{1}$$

其中: q_{mi}等于0或1,表示T_m是否由S_i处理,如$q_{11}=0$表示S_1不处理T_1; $v_{mi}=p'_{mi}/p_i$,表示S_i处理T_m所支出的资源比率,显然$v_{mi}\in[0,1]$. 给出Δ和Π的一种简化表达: $\Delta=|q_{mi}|_{M\cdot C}$, $\Pi=|v_{mi}|_{M\cdot C}$. 泛集群环境下任务调度的合理性体现在同时满足“顾客期望”和“供应商要求”. “顾客期望”包括“尽可能高的性价比”和“尽可能短的完工时间”,可表示为$\arg\max(p(\Delta,\Pi))$和$\arg\min(f(\Delta,\Pi))$; “供应商要求”是指“尽可能低的支出成本”和“尽可能高的成员个体收益”,可表示为$\arg\min(c(\Delta,\Pi))$和$\arg\max(v(\Delta,\Pi))$. 需要注意的是,虽然泛集群环境具有极强的可扩展性,但是在某一时刻也将受到来自于环境的刚性限制(不能超过计算能力、带宽和存储能力的总上限). 于是,泛集群环境下的MOSP问题可以描述为: 以上述4项函数为目标求Δ和Π的最优解.

2.2　目标函数的转化

以$e_{mi}=E(p'_{mi})$表示支出成本与计算能力之间的关系,有

成员个体的收益奖励 =

成员承担的所有任务的回报 − 成员的总支出,

形式化表示为

$$B(S_i)=\sum_{m=1}^{\delta_i}(v_m-e_{mi})=\sum_{m=1}^{\delta_i}(v_m-E(p'_{mi})), \tag{2}$$

其中v_m已知并满足$v_m\propto o_m$. “最大化成员的收益奖励”等价于“尽可能提高所有成员中预期收入最少的成员的收益”,可表示为

$$\arg\max(v(\Delta,\Pi))=\arg\max(v(\min(B(S_i),c))), \tag{3}$$

其中$\min(B(S_i),c)$表示从泛集群成员中找到预期收入最少的成员.

由式(2)和(3)给出“尽可能高的成员个体收益”

的最终描述,有

$$\arg\max(v(\Delta,\Pi))=$$
$$\arg\max\Big(v\Big(\min\Big(\sum_{m=1}^{\delta_i}(v_m-E(p'_{mi}),c)\Big)\Big)\Big).\quad(4)$$

以$t_{mi}=F(p'_{mi})$表示预期完工时间与计算能力之间的关系,则泛集群所有成员的总支出成本为

$$c(\Delta,\Pi)=\sum_{i=1}^{c}c(S_i)=\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}t_{mi}\cdot e_{mi}=$$
$$\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi}).\quad(5)$$

由于$F(p'_{mi})\cdot E(p'_{mi})>0$,有

$$\arg\min(c(\Delta,\Pi))\equiv\arg\max(1/c(\Delta,\Pi)),$$

即

$$\arg\min(c(\Delta,\Pi))=\arg\max(1/c(\Delta,\Pi))=$$
$$\arg\max\Big(\frac{1}{\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi})}\Big).\quad(6)$$

根据性价比=总性能支出/总支出成本,由式(5)给出性价比计算方法为

$$p(\Delta,\Pi)=\Big(\sum_{i=1}^{c}p_i\Big)\Big/c(\Delta,\Pi)=$$
$$\frac{\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}p'_{mi}}{\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi})}.\quad(7)$$

进而给出“尽可能高的性价比”描述

$$\arg\max(p(\Delta,\Pi))=$$
$$\arg\max\Big(\frac{\sum_{i=1}^{c}\sum_{m=1}^{\delta_i}p'_{mi}}{\sum_{i=1}^{c}\sum_{m=i}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi})}\Big).\quad(8)$$

总完工时间取决于完工时间最长的成员,可表示为

$$f(\Delta,\Pi)=\max(t_i)=$$
$$\max\Big(\sum_{m=1}^{\delta_i}t_{mi}\Big)=\max\Big(\sum_{m=1}^{\delta_i}F(p'_{mi})\Big),\quad(9)$$

其中$\max(t_i)$表示取所有成员中最长的完工时间.

由于$\sum_{m=1}^{\delta_i}F(p'_{mi})>0$,有

$$\arg\min(f(\Delta,\Pi))\equiv\arg\max(1/f(\Delta,\Pi)),$$

即“尽可能短的完工时间”可表示为

$$\arg\min(f(\Delta,\Pi))=$$
$$\arg\max(1/f(\Delta,\Pi))=$$
$$\arg\max\Big(1/\max\Big(\sum_{m=1}^{\delta_i}F(p'_{mi})\Big)\Big).\quad(10)$$

式(4)、(6)、(8)、(10)分别体现了“顾客期望”和“供应商要求”所表达的4个目标,并且均期望目标函数能够取得最大值. 所有目标函数尽可能用p'_{mi}表示,这也保证了后续算法的复杂度不会过高.

定义优化权重

$$v_{mi}=\{pw_i,fw_i,cw_i,vw_i\},$$
$$pw_i+fw_i+cw_i+vw_i=1.$$

计算每名成员的计算能力投入,有

$$p'_i=p_i\cdot\frac{c_i}{c}=$$
$$p_i\cdot\frac{\sum_{m=1}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi})\cdot v_{mi}}{\sum_{i=1}^{c}\Big(\sum_{m=1}^{\delta_i}F(p'_{mi})\cdot E(p'_{mi})\Big)\cdot\Big(\sum_{m=1}^{\delta_i}v_{mi}\Big)},\quad(11)$$

其中c_i/c表示成员个体的资源占用率. 由式(4)、(6)、(8)、(10)给出总和目标函数为

$$R(\Delta,\Pi)=pw_i\cdot\arg\max(\Delta,\Pi)+$$
$$fw_i\cdot\arg\max(1/f(\Delta,\Pi))+$$
$$cw_i\cdot\arg\max(1/c(\Delta,\Pi))+$$
$$vw_i\cdot\arg\max(v(\Delta,\Pi)).\quad(12)$$

2.3 MSA算法描述

在真实场景中,成员的计算能力可由CPU频率表示. 为完整地表述MSA算法计算成员能力的过程,给出一种简化的测距算法[21]作为衡量成员通信能力(带宽)的依据,如算法1所示.

算法1 range-finding algorithm.

input: the physical medium distance from S_i to communication objective S_z is dp_{iz},the count of switches is W;

output: λ_i.

```
1 begin
2   d_p = 0, k = 0;
3   while(k < W)
4     d_k = pd_{k,k+1} + e^k · d_p;
5     d_p = d_p + d_k/(k + 1);
6     k++;
7   end while
8 end
```

连续竞买博弈的思想是按o_m、v_m将任务包集合重新排序(优先级$o_m>v_m$),并逐一将任务包分配给能够保证$R(\Delta,\Pi)$收益最高的成员. 定义在1个M

次的连续竞买博弈中,l_m^T表示根据o_m、v_m对T重新排序后的任务包序列. 在第j次子博弈中($j < M$),T_j是当前未分配的任务包集合中v_m/o_m最大的元素,对成员的收益也最高. 因此在优先保证“$R(\Delta, \Pi)$不低于第$j-1$次子博弈的回报”这一前提下,成员们试图通过支出尽可能少的计算能力获得T_j(期望少量支出换取高额回报). 在竞买到T_j后,成员们会将剩余的计算能力作为下一次博弈的“筹码”. 在连续的竞买博弈结束后,所有任务都将分配完毕,资源的分配也随之完成. MSA 算法的执行过程如算法 2 所示,详细步骤描述如下.

step 1: 计算l_m^T,初始化Δ, Π.

step 2: 开始第j次子博弈,找到获得T_j后预期完工时间小于$\max\left(\sum_{m=1}^{\delta_i} F(p'_{mi})\right)$的成员集合$SC_j$.

step 3: 在SC_j中找到满足$\sum_{m=1}^{\delta_i}(p'_{mi}) < \overline{p}_i$的成员集合$SC'_j$.

step 4: 若SC'_j存在,则根据$R(\Delta, \Pi)$在SC'_j中搜索$\overline{S_j}, \overline{S_j}$为完成$T_j$所需要支出计算能力$p'_{ij}$最小的成员;若$SC'_j$不存在,则将范围扩大到$SC_j$中搜索$\overline{S_j}$,并由$\overline{S_j}$处理$T_j$.

step 5: 循环 step 2 ~ step 4,直到所有任务都已经被分配.

step 6: 统计S_i的胜利次数δ_i,记录所有计算结果并给出Δ, Π.

算法 2　MSA.

input: list_$t, S, T, w, R(\Delta, \Pi)$;

output: Δ, Π.

1 begin
2 compute l_m^T, zerosΔ, Π;
3 while $j < M$ do
4 $SC_j = \varnothing$;
5 for each member S_i in S
6 if $t_{mi} < \max\left(\sum_{m=1}^{\delta_i} F(p'_{mi})\right)$, then
7 $S_i \mapsto SC_j$;
8 end if
9 for each member S_i in SC_j
10 if $\sum_{m=1}^{\delta_i} p'_{mi} < \overline{p}_i$, then
11 $S_i \mapsto SC'_j$;
12 end if
13 end for
14 end for
15 if $SC'_j = \varnothing$, then
16 find $\overline{S_j}$ In $SC'_j$by $R(\Delta, \Pi)$;
17 $T_j \mapsto \overline{S_j}$;
18 compute p'_{ij};
19 else
20 find $\overline{S_j}$ in SC_j by $R(\Delta, \Pi)$;
21 $T_j \mapsto \overline{S_j}$;
22 compute p'_{ij};
23 end if
24 δ_i++;
25 update Δ, Π;
26 end while
27 countδ_i, record the results and get Δ, Π;
28 end

3　实例测试

准纳米晶体结构(QNC)是一种由同一类原子或分子经过层级堆叠形成的正多面体结构,通常以正二十面体和正十四面体形式出现. QNC的弱互斥力计算包括空间距离计算和空间标准误计算. 以9层正二十面体晶体结构为测试背景,实验过程包括实验准备、性能测试和任务分配的合理性分析,通过与同类算法的对比验证了调度策略在响应速度、资源性价比和总成本支出具有明显优势.

3.1　实验准备

1) 单个晶体由2 533个分子组成,因此累积计算次数将达到48.66×109次. 将该任务分解为由任务包T构成的Lattice模型,这是一个典型的计算密集型LSC任务.

2) 为清晰完整地体现实验过程,基于一种将复杂的科学工作流映射到分布式系统的框架,前期自行设计开发了一套软件,命名为“Pegasus”. 开发Pegasus的目的是将LSC任务定制为一系列独立的任务包,并将它们封装到样本或样本组中. 在实验过程中,Pegasus将作为数据流处理器. 为提高实验的对比效果,实验过程中可能会出现以下操作.

operation 1:　通过 Pegasus 生成 sample1 ~ sample(Mp). 每个样本包含与样本号相同数量的任务包,例如sample(10)包含10个任务包,每个任务包的计算次数为Mc次;

operation 2: 通过Pegasus生成包含Mp个任务包的sample(Mp);

operation 3:　通过 Pegasus 生成 sample(1) ~ sample(6),任务包的数量基本相同,但总计算次数递增,增量为sample(1)总计算量的一半;

operation 4:　通过 Pegasus 生成 sample(1) ~

sample(6),任务包的数量和样本的总计算量都没有变化.

3) 真实场景的物理环境包括两组用于控制和管理的服务器(型号:Think Centre RD630)、4组标准配置的虚拟机服务器(型号:Think Centre RD630)以及两组高配置的虚拟机服务器(型号:Think Centre R850G7). 所有设备都由一台交换机(型号:Cisco 7340)连接,并形成一个共享的资源池.

场景H的搭建:采用虚拟机模拟泛集群成员,虚拟机由VMware(Version 3.0)创建[22]. 通过改变虚拟机的处理器性能、硬盘控件和内存容量模拟泛集群成员的性能差异. 表1给出了这些成员的性能参数. 其中:C5表示在单位时间内,成员每提供5%的计算能力所能完成的计算次数;E5表示成员每提供5%的计算能力所需要的成本支出基数. 根据$E(p'_i) = \sum E(p'_{mi}) = \sum (p'_{mi}Z(p'_{mi}))$可知

$$\begin{aligned}&\text{成员的成本支出} =\\&\text{已支出的计算能力} \times \text{成本支出基数}\times\\&\text{成本支出系数}(Z(p'_{mi})),\end{aligned}$$

这表明成员的计算能力支出越多,成本支出的增量越高. $Z(p'_{mi})$通常可由一组分段函数表示,如图1所示.

表1 成员的性能参数

No.	dominant frequency / GHz	communication distance	internal storage / G	C5 / times	E5
TC1	2.66	102	4	4 764.82	7 498.10
TC2	1.33	308	8	3 394.68	1 769.10
TC3	2.66	577	8	4 904.97	1 365.31
TC4	1.33	1 476	16	3 162.73	347.05
TC5	3.4	320	8	6 543.96	3 282.43
TC6	5.7	70	32	11 616.44	26 728.36
TC7	1.33	102	4	2 453.98	3 861.68
TC8	2.8	847	16	5 531.47	1 071.65
TC9	2.8	1 426	16	5 112.47	576.46
TC10	1.8	192	8	3 680.98	3 077.28
TC11	1.5	211	4	3 067.48	2 403.48
TC12	2.1	600	2	4 274.47	1 148.85
TC13	4.1	699	32	8 384.45	1 890.16
TC14	4.0	902	8	4 083.97	727.81
TC15	3.3	303	16	6 748.46	3 584.92
TC16	3.6	502	8	7 354.96	2 353.93
TC17	2.66	487	4	5 439.67	1 792.87
TC18	4.6	812	4	9 406.95	1 859.50

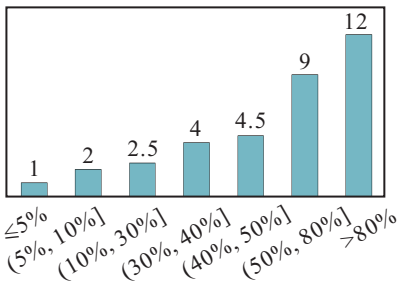


图1 成本支出系数的分段函数表示

在上述构建的场景中,由于虚拟机使用同样的网络通信设备,成员之间的通信性能是相同的. 为了模拟真实场景,采用路由器限制成员的上行/下行带宽,以测量一个10 mb文件在两点间的传输时间模拟算法1中的物理距离.

综上所述,环境配置过程可以描述为:以多组服务器为物理环境,采用VMware以创建虚拟机的方式模拟泛集群,实验中涉及的样本由Pegasus定制.

4) 对比算法包括PBACO[14]、DBMO-SA[15]和HPSO[13]. PBACO算法是一种改进的蚁群算法,用于解决云计算中的多目标优化调度问题. DBMO-SA算法是一种模拟退火算法. HPSO算法是一种混合粒子群算法. 对比算法都有真实的应用场景. PBACO和DBMO-SA算法最初均期望成本最低,HPSO算法优先期望最短的完成时间.

3.2 性能测试

在封闭计算环境中的调度过程只需要执行一次,而泛集群是不稳定的,调度策略可能伴随着成员的各种异常情况而实时改变,这要求调度策略本身需要有较好的响应速度,因此性能测试的目标是比较4种算法在不同场景下的响应速度,进而反映4种算法的适用场景. 测试方法是测量最优调度方案达到95%置信度时,4种算法的平均运行时间(mean run time, MRT)[23]. 性能测试共进行3组实验,具体实验项目包括:1)任务包数量相同,泛集群成员数量增加;2)泛集群成员数量相同,任务包数量增加;3)泛集群成员数量和任务包数量同时增加.

表2　不同任务数量包下4种算法搜索最优解的平均运行时间　ms

	S-a				S-b			
	MSA	PBASO	DBMO-SA	HPSO	MSA	PBASO	DBMO-SA	HPSO
sample(1)	15±1.2	8±1.3	9±1.3	4±1.3	16±1.3	9±1.1	11±1.3	6±1.3
sample(2)	23±1.3	11±1.3	12±1.3	6±1.2	25±1.1	12±1.3	13±1.3	7±1.2
sample(3)	29±1.5	16±1.3	15±1.3	9±1.1	30±1.3	15±1.3	16±1.3	11±1.4
sample(4)	35±1.4	23±1.3	22±1.3	11±1.2	36±1.4	22±1.1	23±1.7	13±1.6
sample(5)	43±1.2	31±1.2	28±1.2	16±1.6	45±1.3	30±1.3	29±1.3	17±1.8
sample(6)	51±1.4	42±1.3	31±1.3	21±1.5	53±1.5	42±1.2	32±1.3	22±1.4
sample(7)	57±1.6	53±1.6	42±1.2	25±1.4	59±1.3	54±1.3	44±1.6	26±1.9
sample(8)	63±1.5	69±1.3	51±1.3	29±1.3	66±1.5	71±1.4	52±1.3	32±1.7
sample(9)	70±1.7	88±1.1	62±1.3	31±1.9	71±1.8	89±1.4	64±1.8	33±1.8
sample(10)	78±1.4	107±1.3	72±1.3	38±1.6	80±1.9	110±1.3	73±1.3	39±1.4
sample(11)	84±1.3	136±1.5	87±1.7	42±1.3	87±1.7	136±1.3	89±1.1	43±1.2
sample(12)	92±1.4	163±1.3	98±1.3	59±1.3	94±1.6	163±1.2	101±1.3	61±1.1
sample(13)	99±1.5	194±1.7	115±1.1	82±1.7	100±1.5	193±1.3	117±1.2	83±1.5
sample(14)	106±1.6	230±1.3	133±1.1	108±1.3	107±1.4	231±1.5	136±1.3	113±1.7
sample(15)	113±1.7	268±1.9	157±1.3	135±1.3	109±1.3	269±1.3	159±1.1	138±1.5
sample(16)	121±1.8	309±1.3	190±1.5	192±1.3	121±1.3	316±1.3	196±1.3	198±1.5
sample(17)	127±1.6	356±1.2	221±1.3	264±1.3	129±1.1	358±1.3	224±1.8	269±1.6
sample(18)	134±1.4	412±1.3	252±1.3	325±1.3	135±1.3	413±1.3	256±1.3	327±1.5
sample(19)	142±1.3	476±1.3	290±1.3	429±1.3	143±1.1	478±1.3	298±1.3	431±1.6
sample(20)	150±1.3	538±1.3	319±1.7	638±1.8	152±1.3	549±1.3	323±1.3	645±1.5

实验1　不同数量任务包的MRT对比.

通过operation 1生产两个样本组S-a和S-b,两个样本组都包含20个样本sample(1)～sample(20),每个样本的任务包数量与其编号相同,S-a的总计算次数为6×10^3,S-b的总计算次数为6×10^7,实验共进行20次. 表2给出了4种算法的MRT测量结果.

由表2得到如下结论:

1) 随着任务包数量的增长,4种算法的平均运行时间也在不断提高.

2) 每种算法在不同样本组中的平均运行时间几乎相同,这表明任务的总计算次数不会影响调度过程本身.

3) HPSO算法在样本数量较少时表现出非常出色的响应速度,但当任务包数量超过临界值后,HPSO算法的平均运行时间持续激增,响应速度不断降低.

4) 随着任务包数量的不断增加,MSA算法的平均运行时间增速最慢.

实验2　不同数量泛集群成员的MRT对比. 通过operation 2生成sample(1)～sample(10)十个样本,由VMware调整泛集群的成员数量(11～20个),实验共进行10次. 表3给出了4种算法的MRT的测量结果.

表3　不同数量泛集群成员下4种算法搜索最优解的平均运行时间　ms

numbers of agents	MSA	PBASO	DBMO-SA	HPSO
11	73±1.3	42±2.1	39±1.5	17±0.8
12	74±1.1	51±2.2	41±1.3	19±1.1
13	73±1.3	57±2.1	45±1.4	22±1.2
14	74±1.3	64±2.2	49±1.3	22±1.9
15	74±1.2	69±2.2	53±1.1	25±1.0
16	75±1.5	75±1.9	57±1.2	28±1.4
17	76±1.5	80±2.7	61±1.7	30±1.2
18	76±1.6	89±2.5	67±1.7	34±0.9
19	77±1.6	97±2.3	68±1.9	36±1.2
20	78±1.7	109±2.4	72±1.3	38±1.3

由表3得到如下结论:

1) 泛集群成员数量将影响4种算法求解最优调度方案的平均运行时间.

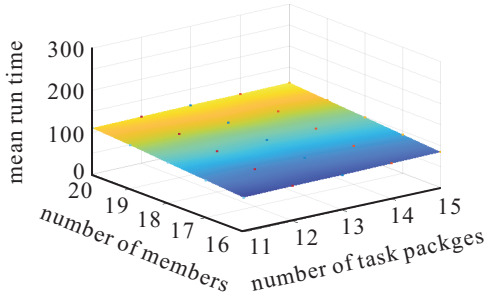
2) 每加入一名成员,MSA的搜索次数仅提高10次,因此平均运行时间的波动不会超过7%.

3) 每加入一名成员,其他对比算法的种群基数将提高$M^c(M-1)$,因此平均运行时间幅度将超过100%,响应速度明显下降. 随着成员数量的持续增长,平均运行时间也将不断激增.

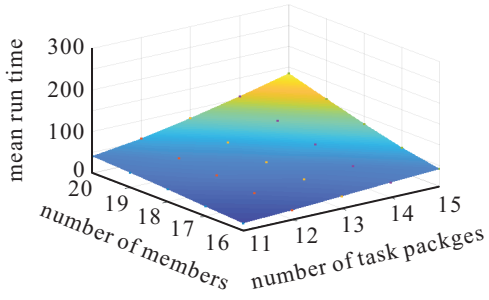
实验3　泛集群成员数量和任务包数量持续增

加的MRT对比.

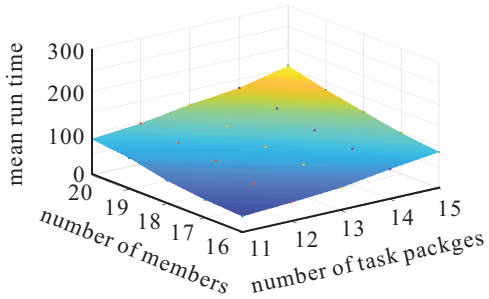
通过 operation 2 生成 sample(1)～sample(5) 五个样本,样本的任务包数量 11～15 个,由 VMware 控制泛集群的成员数量 (16～20 个). 每种算法总计进行 25 次实验,实验结果如图 2 所示.



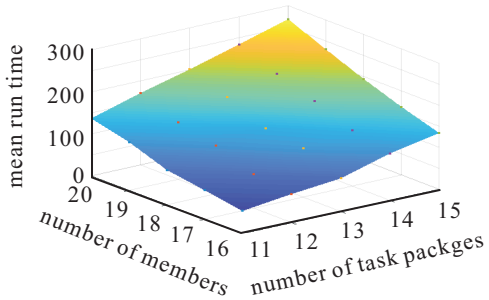
(a) MSA



(b) HPSO



(c) DBMO-SA



(d) PBASO

图 2 4种算法在不同任务包和成员数量的情况下搜索最优解的平均运行时间

由图 2 可见:

1) 当成员和任务包数量较少时,HPSO 算法的平均运行时间最短,因此HPSO 算法更适合少量成员和任务包的情况.

2) 随着成员和任务包数量的增加,所有算法的平均运行时间都在不断增加. 对比算法的种群数量呈指数增长,因此当成员或任务包的数量较大时,平均运行时间将迅速提高. MSA 算法按顺序逐一分配任务包,每次分配过程相对独立,因此 MSA 算法的平均运行时间的增长速度明显低于其他对比算法. 而平均运行时间反映了计算复杂度,从实验效果可知,MSA 算法的计算复杂度明显小于其他算法.

在现实世界中,成员和任务包的数量通常远高于本文构建的实验场景,因此MSA 算法更适合于解决在泛集群环境下的任务调度问题.

3.3 任务分配的合理性分析

任务分配的合理性体现在调度方案实施后的效果,本次测试的目标是比较 4 种算法取得的调度方案的实施效果. 测试过程分为 4 组,分别记录并对比总完工时间、总成本、计算资源占用率和性价比,保持泛集群成员的生存状态和计算能力不变,以前期研究成果[24]给出一种参考权重值 $v_i = \{pw = 0.4, fw = 0.3, cw = 0.2, vw = 0.1\}$. 实验过程描述如下:

1) 总完工时间 (running time) 对比实验. 由 operation 4 生成样本,记录 4 种算法的总完工时间如图 3(a) 所示.

2) 总支出成本对比实验. 由 operation 3 生成样本,计算 4 种算法的总成本支出实验结果如图 3(b) 所示.

3) 计算资源占用率对比实验. 重复实验 2),记录 4 种算法的峰值支出 (可以由所有成员中 CPU 的占有率最高值模拟),实验结果如图 3(c) 所示.

4) 性价比对比实验. 重复实验 2,计算 4 种算法的性价比,实验结果如图 3(d) 所示.

由上述实验结果给出以下分析结果:

1) MSA 算法相比其他算法时间延长了 12 %～26 %.

2) MSA 算法期望的是性价比最高,由图 1 可知,在计算能力的投入大于 50 % 时,支出将提高 100 %,因此 MSA 算法放弃了更高的计算能力支出,限制阈值不超过 50 %(如图 3(c) 所示).

3) 受到 v_{mi}、$E(p'_{mi})$、t_{mi} 的共同影响,MSA 算法的总支出成本最低. 在连续的竞买博弈过程中,ϖ_{mi} 直观地反映出某项子约束在全部约束条件中的重要程度,$E(p'_{mi})$ 的加入也间接体现了 MSA 算法对性价比的要求,这也是泛集群廉价性的表现.

4) HPSO 算法的目的是寻求最短的完工时间,期望以时间弥补由于高计算能力投入导致的低性价比缺陷,这种做法确实降低了一定的支出成本,但始终是 4 种算法中性价比最低的.

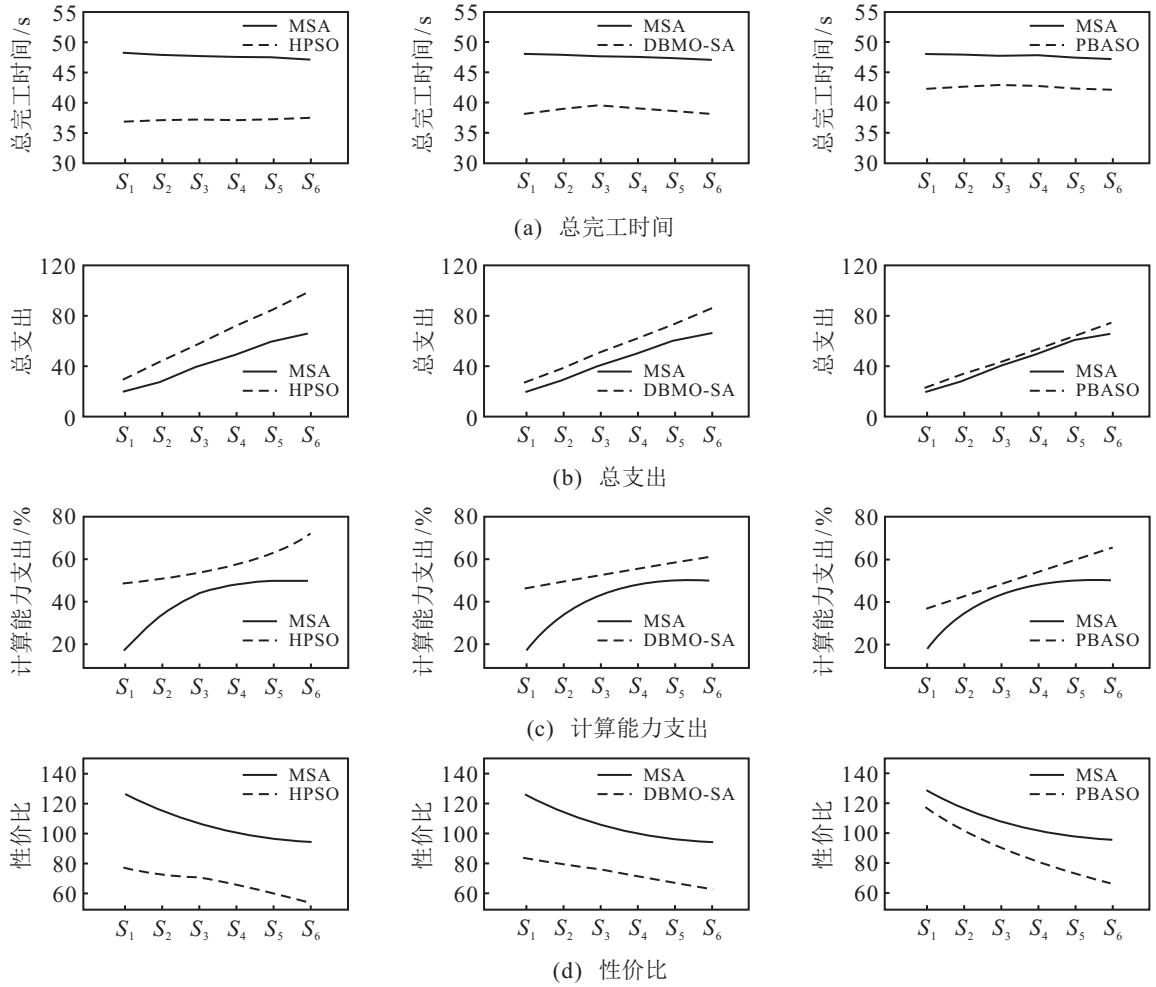


图 3　4种算法的合理性分析实验数据对比

5) PBACO和DBMO-SA算法试图在众多约束间寻找一个平衡点,考虑因素过多将导致算法复杂度过高,考虑因素过少将导致调度方案不合理,因此较难解决泛集群环境下的MOSP问题.

4　结　论

本文针对“计算节点较多的泛集群环境,计算密集型大规模科学计算任务的多目标优化调度问题”,提出了一种基于连续竞买博弈的调度策略. 采用Lattice模型并通过任务调度系统将LSC任务转化为任务包集合;综合考虑高性价比、总完工时间、总成本支出和个体收益等目标保证了MOSP问题求解的合理性;降低目标函数维度,采用线性加权和法将多目标函数转为总和目标函数,并以ETC矩阵作为调度方案的表达形式设计连续的竞买博弈,实现了最优解的快速搜索. 通过响应时间、方案实施效果等实验对比,表明了MSA算法在计算量大(任务包数量多)、成员较多(泛集群环境)的条件下,优于其他同类算法.

边缘计算是未来处理大规模计算任务的必然发展趋势,泛集群作为一种边缘计算的表现形式,具有长远的研究意义. 未来的研究将从以下两方面出发:泛集群环境下动态调度机制的研究和泛集群成员主观稳定性的研究.

参考文献(References)

[1] Gonzalez N M, Miers C C. Cloud resource management: towards efficient execution of large-scale scientific applications and workflows on complex infrastructures[J]. Journal of Cloud Computing, 2017, 6(1): 13-15.

[2] Papa G. Comparison of multi-objective approaches to the real-world production scheduling[J]. Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences, 2018, 48(3): 457-471.

[3] Satyanarayanan Mahadev. Edge computing[J]. Computer, 2017, 50(10): 36-38.

[4] Shi Weisong, Cao J, Zhang Q, et al. Edge Computing: Vision and challenges[J]. IEEE IoT Journal, 2016, 3(5): 637-646.

[5] 周彦伟, 杨波, 张文政. 普适计算环境下的安全访问模型[J]. 电子学报, 2017, 45(4): 959-965.
(Zhou Y W, Yang B, Zhang W Z. Security access model in pervasive computing environment[J]. Acta Electronica sinica, 2017, 45(4): 959-965.)

[6] 王凌, 王晶晶, 吴楚格. 绿色车间调度优化研究进展[J]. 控制与决策, 2018, 33(3): 385-391.
(Wang L, Wang J J, Wu C G. Research progress of green workshop scheduling optimization[J]. Control and Decision, 2018, 33(3): 385-391.)

[7] 沈尧, 秦小麟, 鲍芝峰. 一种云环境中数据流的高效多目标调度方法[J]. 软件学报, 2017, 28(3): 579-597.
(Shen Y, Qin X L, Bao Z F. An efficient multi-objective scheduling method for data flow in cloud environment[J]. Journal of software, 2017, 28(3): 579-597.)

[8] Zheng X, Zhang J, Gao Q. Application of non-cooperative game theory to multi-objective scheduling problem in the automated manufacturing system[C]. International Conference on Automatic Control & Artificial Intelligence. Xiamen: IET, 2013: 554-557.

[9] Zhang Y, Wang J, Liu S, et al. Game theory based real-time shop floor scheduling strategy and method for cloud manufacturing[J]. International Journal of Intelligent Systems, 2016, 32(4): 934-948.

[10] Ni Q, Zarakovitis C C. Nash bargaining game theoretic scheduling for joint channel and power allocation in cognitive radio systems[J]. IEEE Journal on Selected Areas in Communications, 2011, 30(1): 70-81.

[11] Heikkinen T. A potential game approach to distributed power control and scheduling[J]. Computer Networks, 2006, 50(13): 2295-2311.

[12] 徐俊刚, 戴国忠, 王宏安. 生产调度理论和方法研究综述[J]. 计算机研究与发展, 2004, 41(2): 257-267.
(Xu J G, Dai G Z, Wang H A. Research review on production scheduling theory and methods[J]. Computer Research and Development, 2004, 41(2): 257-267.)

[13] Fard H M, Prodan R, Barrionuevo J J D, et al. A multi-objective approach for workflow scheduling in heterogeneous environments[C]. The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Ottawa: ACM, 2012: 300-309.

[14] 王凌, 郑洁, 王晶晶. 求解区间数分布式流水线调度的混合离散果蝇优化算法[J]. 控制与决策, 2018, DOI: 10.13195/j.kzyjc.2018.1274.
(Wang L, Zheng J, Wang J J. Hybrid discrete drosophila optimization algorithm for interval number distributed pipeline scheduling[J]. Control and Decision, 2018, DOI: 10.13195/j.kzyjc.2018.1274.)

[15] Possel B, Wismans L J J, Van Berkum E C, et al. The multi-objective network design problem using minimizing externalities as objectives: Comparison of a genetic algorithm and simulated annealing framework[J]. Transportation, 2016, 4(1): 39-52.

[16] Sarkar S, Maitra S. Some applications of lattice based root finding techniques[J]. Advances in Mathematics of Communications (AMC), 2012, 4(4): 519-531.

[17] Zhang G Q, Zhu W, Sun M, et al. MaPLE: A map reduce pipeline for lattice-based evaluation and its application to SNOMED CT[C]. IEEE International Conference on Big Data. Washington, DC: IEEE, 2014: 754-759.

[18] Deelman E, Singh G, Su M, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems[J]. Scientific Programming, 2005, 13(3): 219-237.

[19] Panda S K, Jana P K. Efficient task scheduling algorithms for heterogeneous multi-cloud environment[J]. The Journal of Supercomputing, 2015, 71(4): 1505-1533.

[20] Braun T D, Siegel H J, Beck N, et al. 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.

[21] Jing G, Zheng Y, Wang L, et al. Consensus of multiagent systems with distance-dependent communication networks[J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, 28(11): 2712-2726.

[22] Eswaraprasad R, Raja L. A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment[J]. Journal of Statistics and Management Systems, 2017, 20(4): 703-711.

[23] Anderson L M. Statistics with confidence-confidence intervals and statistical guidelines[J]. Journal of Clinical Pathology, 1989, 42(12): 1315-1317.

[24] 张可佳, 李春生, 胡亚楠, 等. 智能体自分裂数量最优解的计算方法研究[J]. 控制与决策, 2019, 34(8): 1789-1796.
(Zhang K J, Li C S, Hu Y N, et al. Study on the calculation method of the optimal solution of the number of self-splitting agents[J]. Control and decision, 2019, 34(8): 1789-1796.)

作者简介

张可佳(1986−), 男, 讲师, 博士, 从事多智能体系统(协同机制设计)、算法博弈等研究, E-mail: zkj_3537@163.com;

胡亚楠(1991−), 女, 博士生, 从事多智能体系统与群体智能的研究, E-mail: hyn20160423@163.com;

李春生(1960−), 男, 教授, 博士生导师, 从事多智能体系统方法论、模式挖掘技术等研究, E-mail: csli_dmis@nepu.edu.cn;

富宇(1972−), 男, 副教授, 博士, 从事算法博弈论、群体智能与多智能体系统等研究, E-mail: fy@nepu.edu.cn;

李盼池(1969−), 男, 教授, 博士生导师, 从事量子衍生计算、量子图像处理、深度学习等研究, E-mail: panchili@nepu.edu.cn.

(责任编辑：郑晓蕾)