多任务并发边缘计算环境中最优联盟结构生成策略

赵庶旭，韦萍，王小龙
（兰州交通大学电子与信息工程学院，甘肃 兰州 730071）

摘 要：针对求解最优联盟结构时搜索空间大、效用低等问题，提出了一种基于离散最近过去位置更新策略的多进制离散粒子群优化（MDPSO-DRPPUS）算法。首先，使用基于索引的编码方式编码联盟结构。其次，将多目标优化问题转化为联盟结构的特征值函数。最后，使用 MDPSO-DRPPUS 算法进行最优联盟结构的搜索。实验表明，与多进制离散粒子群优化（MDPSO）算法和遗传算法（GA）相比，所提算法运行时间大幅度降低，联盟结构的效益、均衡性和边缘节点的完成任务效率都有所提高。

关键词：移动边缘计算；资源调度；联盟结构生成；多进制离散粒子群优化；基于离散最近过去更新策略
中图分类号：TN92
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023012

Optimal coalition structure generation strategy in multi-task concurrent edge computing environment

ZHAO Shuxu, WEI Ping, WANG Xiaolong
School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730071, China

Abstract: A discrete recent past-position updating strategy based m-ary discrete particle swarm optimization (MDPSO-DRPPUS) algorithm was proposed for the problem of large search space and low efficiency when solving the optimal coalition structure. First, the coalition structure with index-based was coded. Then, the multi-objective optimization problem was transformed into an eigenvalue function of the coalition structure. Finally, the optimal coalition structure was searched by using the MDPSO-DRPPUS algorithm. Experiments show that compared with the m-ary discrete particle swarm optimization (MDPSO) algorithm and genetic algorithm (GA), the proposed algorithm dramatically reduces the average running time, and improves the efficiency and equilibrium of the coalition structure and task completion efficiency of edge nodes.

Keywords: mobile edge computing, resource scheduling, coalition structure generation, MDPSO, DRPPUS

0 引言

随着物联网的快速发展，其设备数量的爆炸性增长所产生的大量数据发送至云端处理会导致高时延、低带宽等一系列问题。移动边缘计算（MEC, mobile edge computing）[1]提供了一种新的计算范式：在更接近用户或数据源的物理位置上处理和分析数据，以此来降低时延、节省带宽。然而在此环境下，资源受限的边缘节点在面对多任务并发场景时存在以下问题。

1) 单个边缘节点由于自身资源受限无法独立完成任务，或无法满足时延敏感型任务需求。

2) 边缘资源有限且高度分布，现有的资源调度方案并不能最大化其资源利用率。

联盟结构（CS, coalition structure）作为一种用来求解合作问题的模型，广泛应用于传感器融合[2]、无线通信网络融合[3]、蜂窝网络协作[4]和资源协同调度[5]等领域。边缘计算环境中边缘节点资源受限

收稿日期：2022−07−30；**修回日期：**2022−10−28
基金项目：甘肃省重点研发计划基金资助项目（No.20YF8GA123）
Foundation Item: The Key Research and Development Program of Gansu Province (No.20YF8GA123)

条件下的多任务并发资源调度问题[6]可以转化为最优联盟结构生成问题。联盟结构在多任务并发条件下是所有边缘节点集合的划分。其中，边缘联盟是一组平等且通过合作共同完成任务的边缘节点集合。在多任务并发情况下，边缘节点为完成一组并发任务所产生的一组联盟称为联盟结构，其最终目的是通过生成最优联盟结构使社会福利（效用）最大化。然而在边缘计算环境中由于并发任务数量、边缘节点密度、节点计算能力、优化目标和约束条件等因素的影响，最优联盟结构的生成问题较复杂，已成为边缘计算领域的一大挑战。

在有 n 个智能体的系统中，可能的联盟结构的总数为贝尔数，因此无法使用穷尽法搜索得到最优的联盟结构解。胡山立等[7]在最坏条件下提出了一种新的分组方法和给定限界的联盟结构生成算法。Rahwan[8]将所有潜在联盟结构空间划分为包含相似联盟结构的子空间，从而使用分支限界法高效地搜索所选子空间。张新良等[9]针对联盟数量是智能体个数的指数倍的问题，基于智能体合作收益独立性，提出联盟快速动态生成算法，并对联盟结构图进行剪枝，降低了搜索空间大小。徐广斌等[10]利用动态规划原理针对联盟个数约束的特殊性，设计了联盟约束动态规划算法，并证明其算法时间复杂度为 $O(3^n)$。以上算法往往适用于求解小型实例下的联盟结构生成问题。但由于边缘计算环境的特殊性，边缘节点密度相对较高，使用上述算法解决边缘计算环境下的联盟结构生成问题存在一定的局限性。

启发式算法求解最优联盟结构生成问题因其简单直接并能在可接受的时间范围内找到一个相对较优的解而受到广泛关注。对于求解联盟结构，Sen 等[11]首先引入了遗传算法，并采用一维积分编码来搜索最优联盟结构。Yang[12]在 Sen 等[11]工作的基础上，提出了一种基于二维二进制染色体编码及交叉和变异算子的不相交联盟形成算法。Contreras 等[13]使用改进编码的遗传算法求解联盟结构生成问题，其能够在一个合理的计算时间内获得高质量的解。蒋建国等[14]引入蚁群算法解决多任务联盟问题。蚁群基于信息正反馈机制选择协作性能较好的智能体组成联盟，有效减少了联盟生成的时间以及计算量，但该算法只适用于多任务串行计算场景。Lin 等[15]将二进制粒子群优化（BPSO, binary particle swarm optimization）算法扩展为二维二进制编码，并讨论了如何修复无效编码使其有效，但该编码方案效率不高。据 Zhang 等[16]所述，在文献[11-15]的方法中，粒子群优化（PSO, particle swarm optimization）算法可以得到与遗传算法（GA, genetic algorithm）和蚁群优化（ACO, ant clony optimization）算法相似的结果，但计算时间明显快于 GA 与 ACO，尤其在解决较大规模实例方面。与遗传算法相比，粒子群算法没有交叉和变异算子，且算法所需调整的参数少。基于此，本文对性能更优的粒子群算法进行了改进。

在利用粒子群算法对联盟结构进行研究方面，Zhang 等[16]开发了一种一维二进制编码方案，在每次迭代过程中使用编码修复策略确保每个编码都是近似有效的。许金友[17]对传统的基于多任务并发的联盟问题进行了分析，指出联盟资源利用方面的不足，并使用离散粒子群优化算法求解多任务联盟结构生成问题。Hu 等[18]借鉴堆智能离散粒子群优化算法解决资源分配问题的思想[19]，构造了一种描述资源调度方案的联盟结构表达式。将边缘计算环境中的资源调度问题转化为优化问题模型，设计了适应联盟结构编码方式的多进制离散粒子群优化算法。Zhang 等[20]在 Hu 等[18]的基础上结合合作博弈与启发式算法的优点，引入讨价还价集的概念。通过判断非讨价还价联盟来消除一部分不满足条件的联盟结构，从而达到缩小搜索策略空间的目的。最后使用改进的多进制离散粒子群优化（MDPSO, m-ary discrete particle swarm optimization）算法进行联盟结构的搜索，找到近似最优的联盟结构解。

综上所述，PSO 算法[21]在求解边缘计算环境下的多任务并发问题上已经取得了一定的成果。但在任务数和边缘节点数量多的情况下，算法运行时间较长、稳定性不能保证且容易陷入局部最优解。为了解决上述问题，本文从多进制离散粒子群的位置更新部分出发，提出一种新的位置更新方式——基于离散最近过去的位置更新策略（DRPPUS, discrete recent past-based position updating strategy）。在每一次迭代过程中，将粒子的更新区域确定到一个可能出现最优解的区域，以此来提高搜索效率。

本文主要研究工作及贡献如下。

1) 将资源受限边缘计算环境下的节点调度问题转化为优化问题模型，并使用联盟结构来表示节点的资源调度方案。

2) 为优化粒子群算法，提出一种新的更新策略——基于离散最近过去的位置更新策略，并使用

改进的基于离散最近过去位置更新策略的多进制粒子群优化（MDPSO-DRPPUS, m-ary discrete particle swarm optimization discrete recent past-based position updating strategy）算法进行最优联盟结构的搜索。

3) 通过将 MDPSO-DRPPUS 与 MDPSO 和 GA 进行比较可知，与 MDPSO 相比，MDPSO-DRPPUS 的优化速度与最优解质量都得到了提高；与 GA 相比，MDPSO-DRPPUS 的运行时间大幅度降低，联盟结构效益、均衡性和节点完成任务效率都有所提高。

1 问题描述

在边缘计算环境中，当多个大规模科学计算任务同时到达时，边缘节点由于自身资源受限无法独立完成任务或无法满足时延敏感型任务需求。为高效完成大规模科学计算的并发任务，边缘节点选择相互协作组成联盟结构处理这批任务成为一种有效的解决方案。本文方法的工作流程如图 1 所示。

首先，边缘节点生成能够处理并发任务的联盟结构，并使用基于索引的编码方式将其编码，形成策略空间。

其次，分析多目标函数，通过指定不同权重，将多目标问题转化为单目标函数。

最后，使用 MDPSO-DRPPUS 搜索策略空间，找到目标函数的高质量解。

设边缘节点的集合为 $N=\{n_1,n_2,n_3,\cdots,n_i,\cdots,n_{|N|}\}$，并发任务的集合为 $M=\{m_1,m_2,m_3,\cdots,m_i,\cdots,m_{|M|}\}$。边缘节点可以自发地组成联盟来处理一个任务，当多个任务同时到达时，所有边缘节点组成多个联盟同时完成任务，即联盟结构 $\mathrm{CS}=\{A_1,A_2,A_3,\cdots,A_i,\cdots,A_{|\mathrm{CS}|}\}$。在联盟结构生成（CSG, coalition structure generation）问题中，联盟 A_i 定义为集合 N 的任意一个子集，CS 为集合 N 的一个完全划分。$\forall A_i\in\mathrm{CS}$，$A_i\neq\varnothing,\bigcup_{i=1}^{|\mathrm{CS}|}A_i=N$，对于所有的 $i,j\in\{1,2,\cdots,|\mathrm{CS}|\}$，当 $i\neq j$ 时，$A_i\cap A_j=\varnothing$。

典型的联盟生成方法是基于特征值函数[22]的联盟生成，即一个联盟的值由一个特征函数给出，其表示联盟中成员的完成任务所能获得的最大收益。CSG 考虑非超加性环境，随着联盟新成员的加入，联盟的成本也随之增加。当系统中多个任务并行处理时，求解目标就是寻找使系统总效益最大的联盟结构。该场景需同时满足以下 3 个条件。

1) 参与任务的边缘节点数不小于任务数。

2) 每个边缘节点在相同的时间里只能参与完成一个任务，或者不参与完成任务。

3) 联盟产生正利润。

用集合分割的方法分析联盟结构，联盟结构其实是系统内集合的一个划分。Hart 等[23]证明了联盟结构的准确数量为 $\sum_{i=1}^{n}Z(n,i)$，其中，$Z(n,i)$ 是由 i 个联盟所能组成的所有联盟结构的数量。$Z(n,i)$ 的数量也称为第二类斯特林数（联盟结构计算复杂性的具体证明参考文献[24]），其值为

$$Z(n,i)=iZ(n-1,i)+Z(n-1,i-1) \tag{1}$$

其中，$Z(n,n)=Z(n,1)=1$。式(1)右边第一项表示新成员加入现有的联盟形成的联盟结构的数量；式(1)右边第二项表示将新的成员加入自己的联盟中。

例如，在边缘节点集合 $n=\{1,2,3,4\}$ 中，联盟的个数为 2^n-1（不包含空集），共计 15 个联盟，所以联盟结构的个数为 15。4 个智能体（Agent）的联盟结构如图 2 所示。

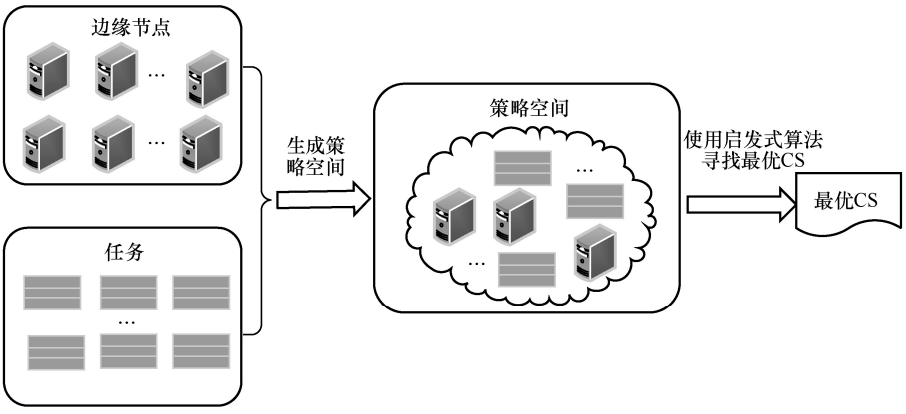


图 1 本文方法的工作流程

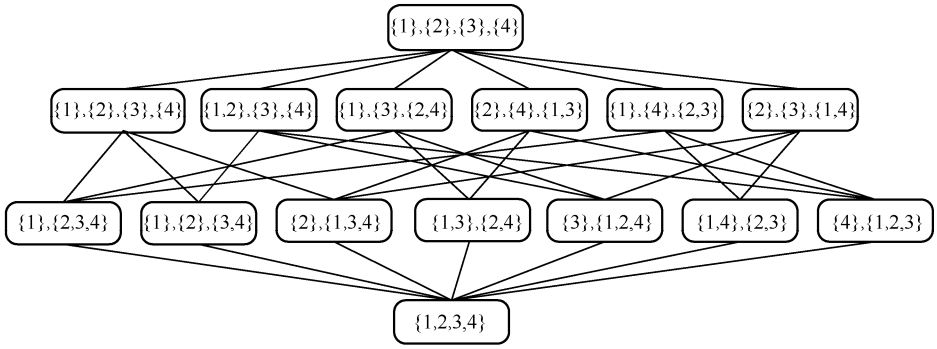


图 2　4 个 Agent 的联盟结构

1.1　问题建模

多任务并发边缘计算环境下的边缘节点资源调度问题可以建模为多目标优化模型。在该场景中，所有边缘节点组成联盟结构来完成集合为 $M=\{m_1,m_2,m_3,\cdots,m_i,\cdots,m_{|M|}\}$ 的多任务并发计算。联盟结构中的每个联盟 A_i 完成一个任务 m_i，联盟 $A_i=(n_j^i,|n^i|)$，其中，n_j^i 表示节点 n_j 加入联盟 A_i，$|n^i|$ 表示加入联盟 A_i 的成员个数。每个联盟 A_i 都有一个联盟的特征值 $C_i=(m_i,|n^i|)$，表示联盟 A_i 完成任务 m_i。特征值 C_i 由联盟 A_i 的收益、成本、任务完成时间以及节点组成联盟的额外支出共同决定。联盟结构特征值为联盟结构中联盟的总特征值之和，即 $V(\mathrm{CS})=\sum_{i=1}^{|M|}V(C_i)$。

1.1.1　时间约束

记任务完成的截止时间为 DL_i，要求联盟 A_i 完成各任务的时间 t_i 必须小于或等于各任务完成的截止时间，即 $t_i \leqslant \mathrm{DL}_i$。联盟结构中的所有联盟都按时完成任务，形成一个有效的联盟结构。由木桶效应可知，联盟结构的任务完成时间取决于有效联盟结构中完成任务时间最长的联盟，即 $t(\mathrm{CS}_i)=\max(t(A_i))$。

$$\mathrm{P1}:\ \min_{n_j^i\in A_i} t(\mathrm{CS}_i)$$

$$\text{s.t. } t_i \leqslant \mathrm{DL}_i$$

1.1.2　效用函数

联盟完成任务后会获得一定的收益，定义 E_i 为联盟 A_i 完成任务 m_i 的收益，CP_i 为边缘节点 n_i 的计算能力。联盟的计算能力支出即组成联盟的边缘节点的计算能力支出总和，即 $\mathrm{EP}(A_i)=\sum_{i=1}^{|n^i|}(\mathrm{CP}_i t(A_i))$。定义各边缘节点组成联盟的损耗函数为 $L(A_i)$，损耗函数是指节点之间组成联盟引起的开销，其包括联盟间的通信损耗、计算冗余等。由于损耗是由节点之间协同组成联盟所造成的，因此为单调递增函数，且随着联盟内成员的增加而增加，即 $L(A_i)=\mathrm{num}(n_i)=|n^i|$。

一个联盟完成任务 m_i 的利润可以用联盟的收益与成本进行计算。设 $P(C_i)$ 表示一个联盟完成任务 m_i 的利润，其计算方式为收益–成本–额外损耗，即

$$P(C_i)=E_i-\mathrm{EP}(A_i)-L(A_i) \tag{2}$$

$$\mathrm{P2}:\ \max_{n_i\in N} P(C_i)$$

1.1.3　均衡性

基于个体理性原则，成员所做出的决策都是明智且理性的。节点期望加入支出最小、效益最大的联盟，其可以通过性价比衡量。联盟的性价比函数定义为

$$\mathrm{CP}(C_i)=\frac{P(A_i)}{T(A_i)}=\frac{E_i-\mathrm{EP}(A_i)-L(A_i)}{\max(t_j^i)} \tag{3}$$

为保证联盟结构中各联盟的均衡性，即要求联盟结构中各联盟间的性价比差异最小。因此，使用方差来度量其均衡性，即

$$\begin{cases}\mathrm{var}(C_i)=\sum_{i=1}^{|M|}(\mathrm{CP}(A_i)-\mathrm{avg}(\mathrm{CP}(A_i)))^2\\ \mathrm{avg}(\mathrm{CP}(C_i))=\dfrac{\sum_{i=1}^{|M|}\mathrm{CP}(A_i)}{|M|}\end{cases} \tag{4}$$

$$\mathrm{var}(C_i)=\sum_{i=1}^{|M|}\left(\frac{E_i-\mathrm{EP}(A_i)-L(A_i)}{\max(t_j^i)}-\frac{\sum_{i=1}^{|M|}\mathrm{CP}(A_i)}{|M|}\right)^2 \tag{5}$$

$$\mathrm{P3}:\ \min_{n_i\in N}\mathrm{var}(C_i)$$

1.2 解法模型

为了解决该多目标优化问题，本文使用线性加权法将多目标优化问题转化为一个综合的目标函数。对于 P1、P2 和 P3，其权重分别为 ω_1、ω_2和ω_3，且 $\omega_1+\omega_2+\omega_3=1$。

联盟结构的特征值函数为组成联盟结构联盟的特征值总和，即

$$\max_{m_i\in M}(O(\mathrm{CS}_i))=\omega_1\sum_{i=1}^{|M|}P(C_i)-\omega_2\sum_{i=1}^{|M|}\mathrm{var}(C_i)-\omega_3 t(\mathrm{CS}_i) \tag{6}$$

权重系数会影响最优解的求解结果，ω_1 是任务完成时间的权重系数，会使算法倾向于搜索更高效的联盟结构；ω_2 是效用权重系数，会使算法倾向于搜索效用值较大的联盟结构；ω_3 是性价比权重系数，会使算法倾向于搜索满足所有节点性价比的联盟结构。由于大多数场景更关注时延和能耗，因此本文在实验中弱化性价比权重系数 ω_3，并设置 ω_3=0.1。分析不同 ω_1 和 ω_2 条件下优化速度和最优解质量的影响，重复进行多次实验，实验结果表明，ω_1 对算法的平均运行时间影响不大，同样需弱化 ω_1，令 ω_1=0.1。因此，ω_2=0.8。

2 寻找最优联盟结构

启发式算法在求解联盟结构生成这类复杂的组合优化问题时，其共同点是从随机的可行解开始，经过不断的迭代、改进、变异，最终无限趋近于问题的最优解。但是面对大规模的联盟结构搜索空间，原始启发式算法因其运行时间长、易陷入局部最优解等缺点并不能获得较优的解。因此，本文提出了基于离散最近过去位置更新策略的多进制粒子群优化算法。

寻找一种适用于该场景下联盟结构的编码方式是使用启发式算法求解该问题的第一步。因此，首先使用基于索引的联盟结构编码方式对联盟结构进行编码，将编码粒子的长度与参与任务的边缘节点数对应起来，并将粒子每个维度的索引与任务编号对应起来，用以满足本场景下的约束条件。其次，本文对原始粒子群算法的更新方式进行改进，引入政治优化器（PO, political optimizer）中的更新策略——基于最近过去的位置更新策略。再次，为了使其适用于本文场景下联盟结构的编码方式，对该更新策略进行离散化改进。最后，对所提算法的复杂度进行分析。

2.1 联盟结构的编码方式

任何一个有效的联盟结构都以任务的完成为前提，联盟结构中联盟的数量就等于其所要完成任务的数量，即 $\mathrm{num}\left(\sum_{i=1}^{|M|}A_i\right)=|M|$。联盟与联盟的结构关系如图 3 所示。

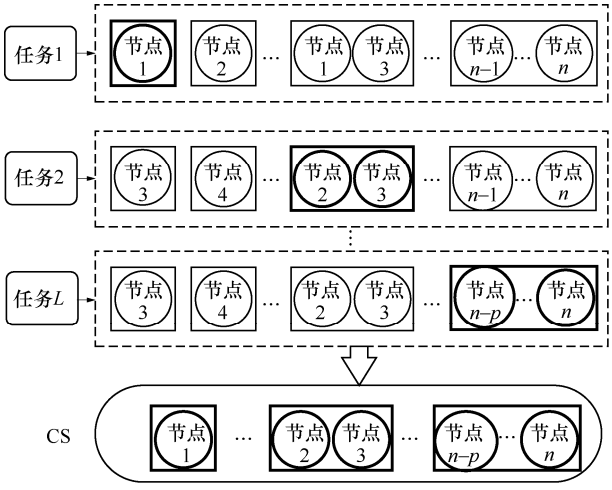


图 3 联盟与联盟的结构关系

在该场景中，所要搜索的最优联盟结构需同时满足以下 4 个约束条件。

1) 联盟实际完成各任务的时间应小于或等于各任务的计划完成时间。

2) 参与联盟结构节点的数量不应超过边缘节点的总数。

3) 每个节点最多只能参与完成一个任务。

4) 每个任务至少需要一个边缘节点完成。

为了使联盟结构的编码能够满足约束 2)~约束 4)，本文选用基于索引的联盟结构编码方式。定义 D_i 代表节点 n_i 参与任务，且 $D_i\in[0,|M|]$。联盟结构的编码方式可记为 $\langle D_1,D_2,D_3,\cdots,D_i,\cdots,D_{|N|}\rangle$。

以 6 个边缘节点完成 3 个任务为例，图 4 表示基于索引的联盟结构编码方式。由图 4 可知，节点 1 和节点 6 组成联盟完成任务 1，节点 4 和节点 5 协同组成联盟完成任务 2，节点 2 完成任务 3，节点 3 不参与完成任何一个任务。该联盟结构的编码为 130221。每个节点都可以选择参与任务或不参与任务，即有 $|M|+1$ 种选择，若边缘节点的总数为 $|N|$，联盟结构的总数就有 $\mathrm{num}(\mathrm{CS}_i)=(|M|+1)^{|N|}$ 种。

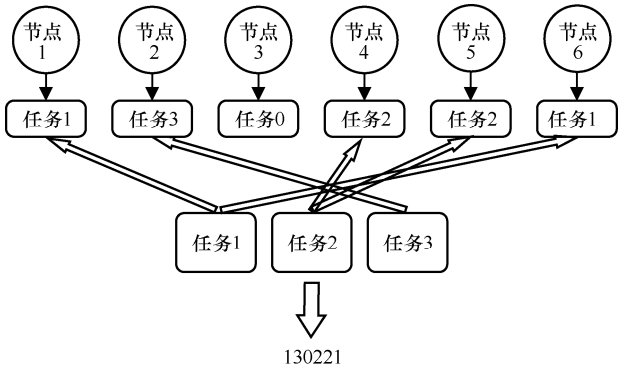


图 4　基于索引的联盟结构编码方式

2.2　基于改进更新策略的 MDPSO

原始的粒子群算法存在过早收敛、容易陷入局部最优值、收敛速度慢等缺点。因此，本文提出一种改进的 DRPPUS 的 MDPSO 算法。

2.2.1　DRPPUS

该更新策略是 Askari[25]在 2020 年提出的政治优化器中的更新机制[26]。

基于最近过去更新策略保存了前一次迭代时算法所学习到的信息，更新每一个成员当前最优解的位置来寻找下一次可能的最优解位置。算法使用式(7)和式(8)来更新其可能的最优解位置，根据成员当前得到的适应度值 $f(p_{i,d}^{j}(t))$ 与前一次适应度值 $f(p_{i,d}^{j}(t-1))$ 确定选择式(7)或式(8)进行位置更新。若特征值函数有所提高，则使用式(7)；反之，则使用式(8)。在这 2 种情况中，位置的更新依据当前可能的最优解 $p_{i,d}^{j}(t)$、变量 r 和可能的参考解 $p_d^*(t)$，其中，随机数 r 的取值范围为 $[0,1]$。

$$\mathrm{C1}: p_{i,d}^{j}(t-1)\leqslant p_{i,d}^{j}(t)\leqslant p_d^*(t) \text{ 或 } p_{i,d}^{j}(t-1)\geqslant p_{i,d}^{j}(t)\geqslant p_d^*(t)$$

$$\mathrm{C2}: p_{i,d}^{j}(t-1)\leqslant p_d^*(t)\leqslant p_{i,d}^{j}(t) \text{ 或 } p_{i,d}^{j}(t-1)\geqslant p_d^*(t)\geqslant p_{i,d}^{j}(t)$$

$$\mathrm{C3}: p_d^*(t)\leqslant p_{i,d}^{j}(t-1)\leqslant p_{i,d}^{j}(t) \text{ 或 } p_d^*(t)\geqslant p_{i,d}^{j}(t-1)\geqslant p_{i,d}^{j}(t)$$

$$p_{i,d}^{j}(t+1)=\begin{cases} p_d^*(t)+r(p_d^*(t)-p_{i,d}^{j}(t)), \mathrm{C1} \\ p_d^*(t)+(2r-1)\left|p_d^*(t)-p_{i,d}^{j}(t)\right|, \mathrm{C2} \\ p_d^*(t)+(2r-1)\left|p_d^*(t)-p_{i,d}^{j}(t-1)\right|, \mathrm{C3} \end{cases} \tag{7}$$

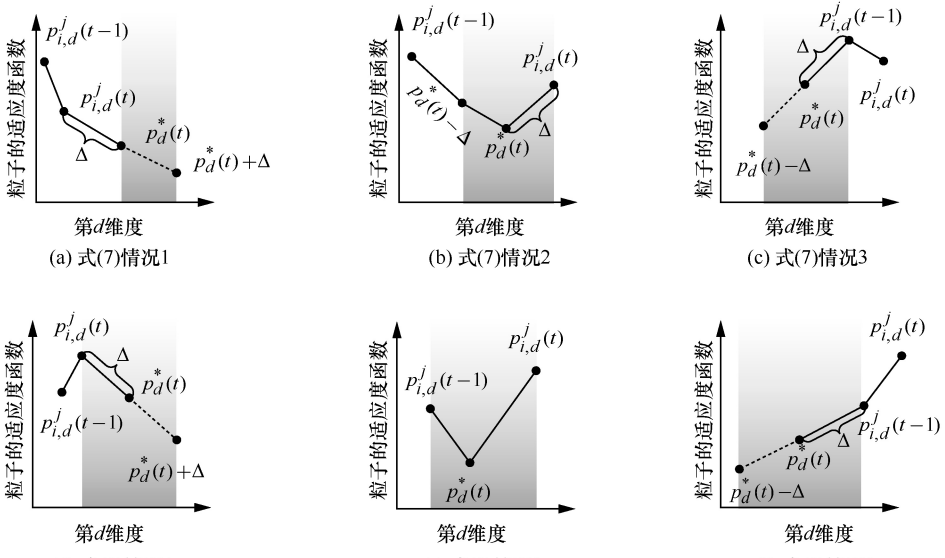
$$p_{i,d}^{j}(t+1)=\begin{cases} p_d^*(t)+(2r+1)\left|p_d^*(t)-p_{i,d}^{j}(t)\right|, \mathrm{C1} \\ p_{i,d}^{j}(t-1)+r(p_{i,d}^{j}(t)-p_{i,d}^{j}(t-1)), \mathrm{C2} \\ p_d^*(t)+(2r+1)\left|p_d^*(t)-p_{i,d}^{j}(t-1)\right|, \mathrm{C3} \end{cases} \tag{8}$$

RPPUS 表示如图 5 所示，图 5(a)~图 5(c)说明了式(7)的 3 种情况，图 5(d)~图 5(f)说明了式(8)的 3 种情况，主要目的就是找到最有可能产生最优解的区域。

情况 1　如图 5(a)所示，成员的当前位置位于可能的参考解和前一次位置之间，可能产生最优解的区域用灰色标出，其范围为 $\Delta=\left|p_d^*(t)-p_{i,d}^{j}(t)\right|$。

情况 2　当成员的参考解位置位于当前位置和前一次位置之间时，可能产生最优解的区域如图 5(b)所示，其范围为 $\Delta=\left|p_d^*(t)-p_{i,d}^{j}(t)\right|$。

情况 3　成员的前一个位置位于参考解与当前位置之间，如图 5(c)所示，可能产生最优解的区域在参考解附近，同理，$\Delta=\left|p_d^*(t)-p_{i,d}^{j}(t-1)\right|$，因为 $f(p_{i,d}^{j}(t-1))>f(p_{i,d}^{j}(t))$。



(a) 式(7)情况1　(b) 式(7)情况2　(c) 式(7)情况3

(d) 式(8)情况1　(e) 式(8)情况2　(f) 式(8)情况3

图 5　RPPUS 表示

2.2.2 MDPSO-DRPPUS

针对最近过去位置更新策略的离散化改进如式(9)~式(22)所示。该更新策略在一个可能产生最优解的区域内进行更新，实际更新有可能超出该区域。所以定义一个整型函数 $R(s,e)$，相关参数的范围为 $[\overline{\sigma}_d,\underline{\sigma}_d]$，算法决策变量的更新范围为 $[\overline{p}_d,\underline{p}_d]$。

$$\begin{cases}\overline{\sigma}_d=\overline{p}_d-p_d^*(t)\\ \underline{\sigma}_d=p_d^*(t)-\underline{p}_d\end{cases} \tag{9}$$

若式(7)的 C1 成立，则有

$$\Delta=\left|p_d^*(t)-p_{i,d}^j(t)\right| \tag{10}$$

位置更新式为

$$p_{i,d}^j(t+1)=\begin{cases}p_d^*(t)+R(s,e),\Delta\geqslant 0\\ p_d^*(t)-R(s,e),\Delta<0\end{cases} \tag{11}$$

其中，$s=0$，e 为

$$e=\begin{cases}\Delta,\Delta\geqslant 0,\Delta<\overline{\sigma}_d\text{或}\Delta<0,\Delta\geqslant\underline{\sigma}_d\\ \overline{\sigma}_d,\Delta\geqslant 0,\Delta\geqslant\overline{\sigma}_d\\ \underline{\sigma}_d,\Delta<0,\Delta>\underline{\sigma}_d\end{cases} \tag{12}$$

针对式(7)的 C2 和 C3，有

$$\Delta=\begin{cases}\left|p_d^*(t)-p_{i,d}^j(t)\right|,\text{C2}\\ \left|p_d^*(t)-p_{i,d}^j(t-1)\right|,\text{C3}\end{cases} \tag{13}$$

位置更新式为

$$p_{i,d}^j(t+1)=p_d^*(t)+R(s,e) \tag{14}$$

$$s=\begin{cases}-\Delta,\Delta<\underline{\sigma}_d\\ -\underline{\sigma}_d,\Delta\geqslant\underline{\sigma}_d\end{cases} \tag{15}$$

$$e=\begin{cases}\Delta,\Delta<\overline{\sigma}_d\\ \overline{\sigma}_d,\Delta\geqslant\overline{\sigma}_d\end{cases} \tag{16}$$

对于式(8)的 C1 和 C3，有

$$\Delta=\begin{cases}\left|p_d^*(t)-p_{i,d}^j(t)\right|,\text{C1}\\ \left|p_d^*(t)-p_{i,d}^j(t-1)\right|,\text{C3}\end{cases} \tag{17}$$

位置更新式为

$$p_{i,d}^j(t+1)=p_d^*(t)+R(s,e) \tag{18}$$

$$s=\begin{cases}-\Delta,\Delta<\underline{\sigma}_d\\ -\underline{\sigma}_d,\Delta\geqslant\underline{\sigma}_d\end{cases} \tag{19}$$

$$e=\begin{cases}\Delta,\Delta<\overline{\sigma}_d\\ \overline{\sigma}_d,\Delta\geqslant\overline{\sigma}_d\end{cases} \tag{20}$$

对于式(8)的 C2，有

$$\Delta=\left|p_d^*-p_{i,d}^j(t)\right| \tag{21}$$

位置更新式为

$$p_{i,d}^j(t+1)=\begin{cases}p_{i,k}^j(t-1)+R(s,e),\Delta\geqslant 0\\ p_{i,k}^j(t-1)-R(s,e),\Delta<0\end{cases} \tag{22}$$

其中，$s=0$，$e=|\Delta|$。

MDPSO-DRPPUS 算法的伪代码如算法 1 所示。

算法 1 MDPSO-DRPPUS 算法

输入 联盟结构信息（联盟结构编码、完成任务所获利润、任务截止时间、成本支出等，该列表构成策略空间），MDPSO-DRPPUS 信息（粒子信息，包括粒子更新位置、适应度值、粒子的全局最优位置），参数设置（种群个数 NP、迭代次数 G、编码长度（节点数 N）、随机数 r、决策变量上限 $\overline{p}_d$、决策变量下限 $\underline{p}_d$）

输出 全局最优解（最优联盟结构）

1) 初始化每一个粒子的随机位置；

2) 计算每一个粒子的适应度；

3) 循环求出粒子当前的全局最优解 $p_d^*(t)$；

4) 设迭代次数 $g=1$；

5) 如果 $g\leqslant G$；

6) 判断粒子前一次适应度值与这次适应度值的大小，若适应度值有所提高，则使用式(7)的 3 种情况进行判断，并按照情况选择相应离散化的位置更新式来更新粒子位置，反之亦然；

7) 根据适应度函数值更新全局最优解 $p_d^*(t)$；

8) 判断粒子适应度值是否提高，若不提高则结束迭代，否则转步骤 9)；

9) $g=g+1$，转步骤 6)；

10) 输出粒子的最优解及对应粒子位置。

2.3 MDPSO-DRPPUS 的算法复杂度分析

MDPSO-DRPPUS 算法的最大计算量是粒子数与迭代次数的乘积，增加粒子数可以扩大搜索范围，降低算法陷入局部最优解的可能性，增加迭代次数则可以提高最优解质量。算法的计算量是算法对特征值函数的求解，在一次算法执行过程中，特征值函数的计算分为 3 个阶段。

1) 计算任务完成时间最长的节点，计算次数为 N。

2) 根据式(2)，将所有形成联盟结构联盟的利润加起来，联盟结构的利润计算次数为$(M+1)(N+1)$。

3) 根据式(3)~式(5)，联盟结构均衡性的计算次数为$(M+1)^2(N+1)$。

综上，MDPSO-DRPPUS 的计算复杂度最低，计算量最大为$\text{sum}=\text{NP}G((M+2)(N+1)(M+1)+N)$。

3　实验与分析

本文的仿真实验是在内存为16 GB、处理器为Inter Core i5-4460、频率为3.2 GHz 的 Windows10 操作系统环境下使用 Python3.7 实现的。通过模拟和仿真，对本文所提算法和对比实验的各项指标进行评估。

3.1　实验准备以及实验数据

创建虚拟机来模拟边缘节点，表 1 给出了实验环境中虚拟机配置，表 2 显示了任务工作量、计划完成时间和任务报酬，表 3 显示了环境参数。

表 1　　实验环境中虚拟机配置

节点 ID	CPU		RAM		计算能力/MIPS	开销/s
	内核/个	频率/MHz	内存/GB	频率/MHz		
VM_1	1	1.33	4	1 600	1.21	1.20
VM_2	1	2.66	4	1 600	2.43	2.50
VM_3	2	1.33	4	1 600	2.62	2.90
VM_4	2	2.66	4	1 600	4.65	3.60
VM_5	1	1.33	4	2 133	1.61	1.70
VM_6	1	2.66	4	2 133	3.25	3.45
VM_7	2	1.33	4	2 133	3.5	3.70
VM_8	2	2.66	4	2 133	6.21	6.30
VM_9	1	1.33	8	1 600	2.51	2.65
VM_{10}	1	2.66	8	1 600	4.92	5.10
VM_{11}	2	1.33	8	1 600	5.38	5.60
VM_{12}	2	2.66	8	1 600	9.54	9.90
VM_{13}	1	1.33	8	2 133	3.34	3.50
VM_{14}	1	2.66	8	2 133	6.31	6.60
VM_{15}	2	1.33	8	2 133	7.19	7.35
VM_{16}	2	2.66	8	2 133	12.98	14.20

表 2　任务工作量、计划完成时间和任务报酬

任务 ID	任务工作量/百万次	计划完成时间/s	任务报酬
M_1	409	110	487
M_2	562	105	610
M_3	713	100	780
M_4	860	120	950
M_5	1 075	110	1 195
M_6	1 204	125	1 395
M_7	1 365	125	1 520
M_8	1 520	130	1 705

表 3　　环境参数

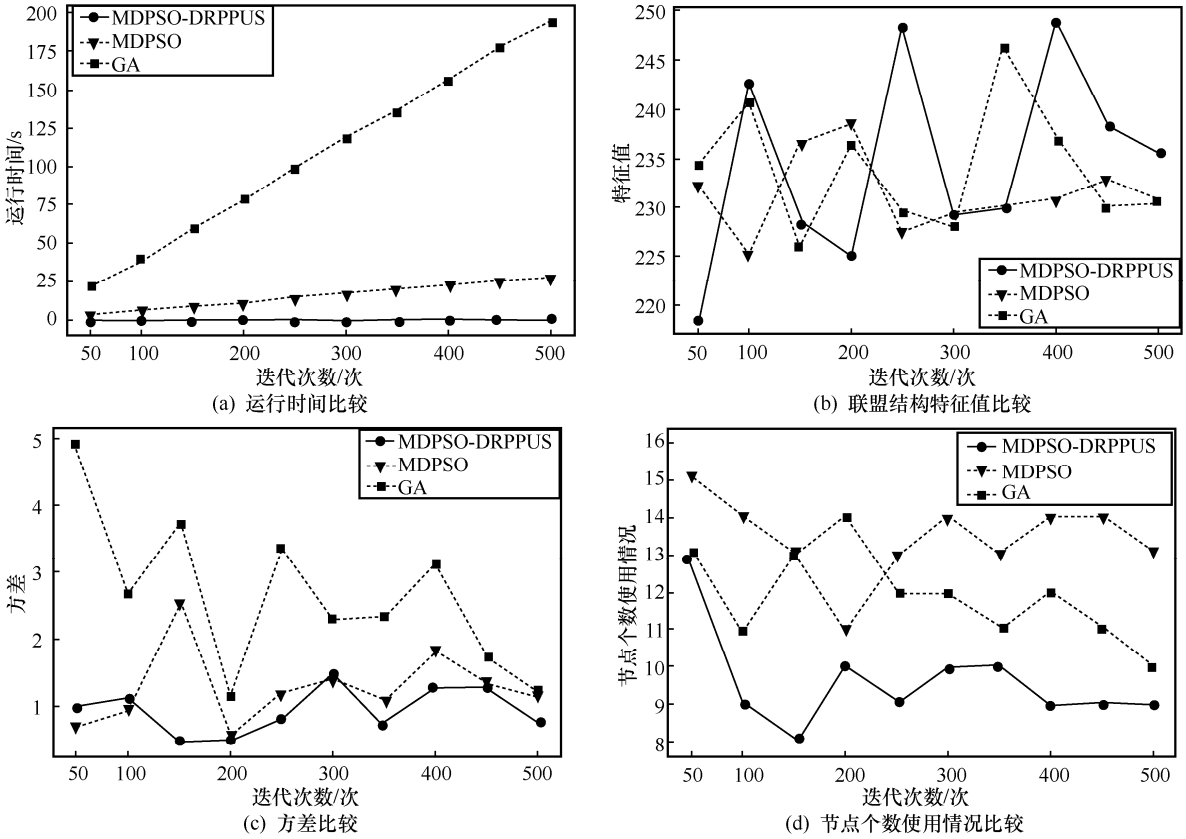
参数	参数值
最大任务数/个	8
最大边缘节点数/个	16
所有联盟结构数	1.85×10^{15}
实验环境总计算能力/MIPS	77.65
实验环境总开销/s	80.25
总计算任务/百万次	7 708

3.2　MDPSO-RPPUS 算法的性能实验

本节在不同迭代次数条件下比较了 3 种算法（MDPSO-RPPUS、MDPSO 和 GA）在 16 个边缘节点上完成 4 个任务（M_1、M_2、M_3 和 M_4）时的算法平均运行时间、最优联盟结构效益、均衡性和节点使用率，并分析了不同迭代次数对算法性能的影响。针对粒子数为 100 个、不同迭代次数进行 10 组实验，迭代次数从 50 增加到 500（每增加 100 次进行一组实验，每组实验进行 5 次，最终结果取平均值）。图 6 是不同迭代次数下 3 种算法性能比较。

由图 6(a)可知，MDPSO 和 GA 的运行时间都随着迭代次数的增加而增加，这是启发式算法求解问题的特点。而 MDPSO-DRPPUS 算法的运行时间随迭代次数的增加变化不大，且算法运行时间在毫秒级，这是因为该算法往往能在迭代 10 次以内收敛。由图 6(b)可知，3 种算法联盟结构的效益随迭代次数增加变化较小，且结果不稳定。由此看来，通过增加迭代次数使 3 种算法获得较好结果的做法意义不大。由图 6(c)可知，与 MDPSO 和 GA 相比，MDPSO-DRPPUS 联盟结构均衡性较优，且 GA 最不稳定。由图 6(d)可知，MDPSO-DRPPUS 算法的节点调度策略对运算量较大的任务分配多个边缘节点进行计算，能够避免多个节点很快完成计算量小的任务，但仍需等待并发任务中计算量较大任务的完成。该算法使用最少数量的节点完成任务，提高了任务完成效率。

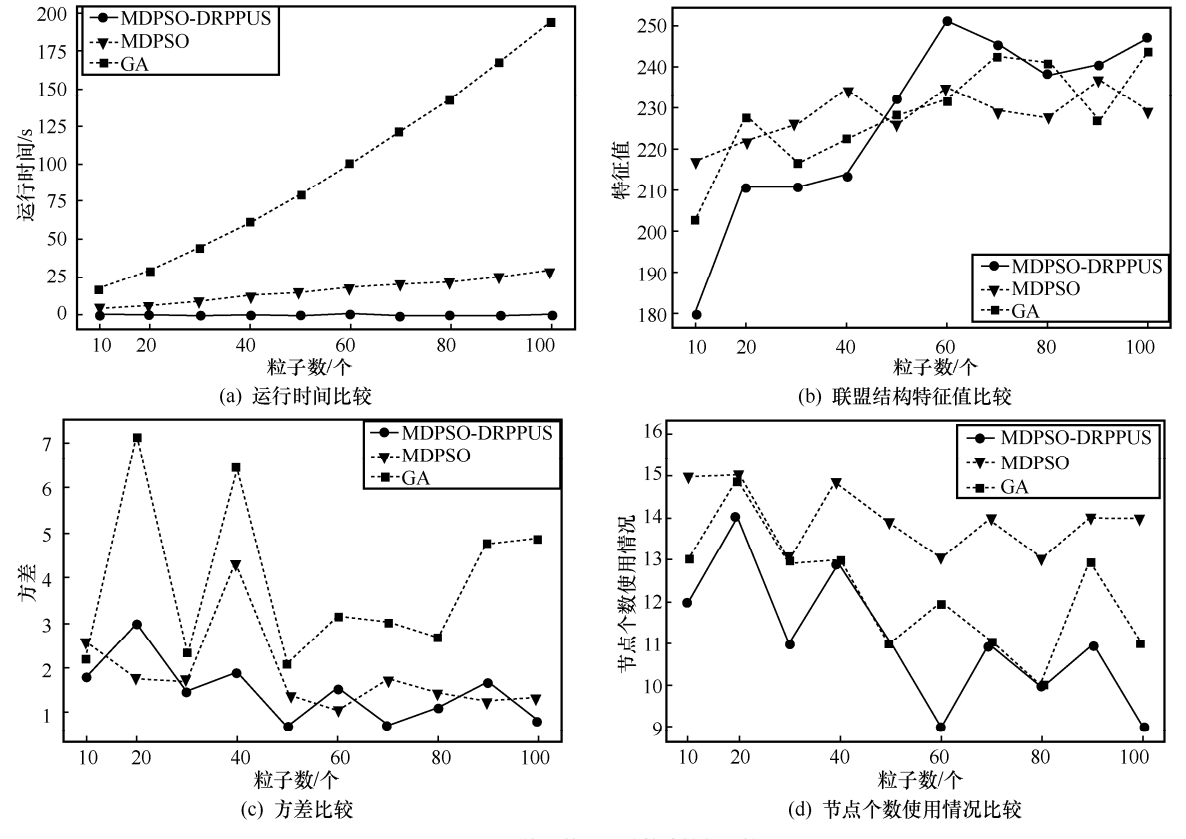
针对不同粒子数，比较 3 种算法在 16 个边缘节点上完成 4 个任务（M_1、M_2、M_3 和 M_4）时在 4 种标准下分析不同粒子数对算法性能的影响。针对迭代次数为 500、不同粒子数进行 10 组实验，粒子数从 10 个增加到 100 个（每次增加 10 个进行一组实验，每组实验进行 5 次，最终结果取平均值）。图 7 是不同粒子数下 3 种算法性能比较。



(a) 运行时间比较　(b) 联盟结构特征值比较

(c) 方差比较　(d) 节点个数使用情况比较

图 6　不同迭代次数下 3 种算法性能比较



(a) 运行时间比较　(b) 联盟结构特征值比较

(c) 方差比较　(d) 节点个数使用情况比较

图 7　不同粒子数下 3 种算法性能比较

图 7(a)、图 7(c)、图 7(d)所示结果与图 6(a)、图 6(c)、图 6(d)相同，此处不再赘述。由图 7(b)可知，MDPSO-DRPPUS 算法随着粒子数的增多联盟结构效益增长较快，这是因为 DRPPUS 极大地提高了算法的开发能力。因此本文推测可以通过大幅度提高算法的粒子数来避免 MDPSO-DRPPUS 陷入局部最优解的可能。

在迭代次数为 10 的情况下大规模增加粒子数，比较 3 种算法在 16 个边缘节点完成 4 个任务（M_1、M_2、M_3 和 M_4）时在 4 种标准下的算法性能。针对不同粒子数进行 6 组实验，粒子数从 200 个增加到 1 200 个（每次增加 200 个进行一组实验，每组实验进行 5 次，最终结果取平均值）。图 8 是迭代次数为 10 时不同粒子数下 3 种算法性能比较。

图 8(c)和图 8(d)与图 6(c)和图 6(d)结果类似。由图 8(a)可知，在迭代次数为 10 的情况下，大规模增加算法的粒子数（GA 大规模增加其基因数），当粒子数增加到 1 200 时，MDPSO-DRPPUS 算法平均运行时间为 2 s 左右；MDPSO 算法平均运行时间为 6 s 左右；GA 平均运行时间已达到 257 s。由图 8(b)可知，MDPSO-DRPPUS 算法随着粒子数的大规模增加联盟结构效益增长较快，进一步证明了之前的推断。

在不同边缘节点（VM_9-VM_{16}、VM_8-VM_{16}、VM_7-VM_{16}、VM_6-VM_{16}、VM_5-VM_{16}、VM_4-VM_{16}、VM_3-VM_{16}、VM_2-VM_{16}、VM_1-VM_{16}）条件下测试 3 种算法在 4 种标准下的性能。为了避免任务量过大导致任何联盟结构都无法完成任务的情况，选择 4 个计算量最小的任务（M_1、M_2、M_3 和 M_4）进行 9 组实验（每增加一个边缘节点进行一组实验，每组实验进行 5 次，最终结果取平均值）。设置 3 种算法（MDPSO-RPPUS、MDPSO 和 GA）的最大迭代次数 $G=50$，最大粒子数 $NP=500$。图 9 显示了不同节点数下 3 种算法性能比较。

图 9(a)、图 9(c)、图 9(d)与图 6(a)、图 6(c)、图 6(d)结果类似，此处不再赘述。由图 9(b)可知，当边缘节点数为 9 和 13 时，3 种算法的最优联盟结构特征值都有了明显的提升，这是因为添加了计算能力较强的 VM_8 和 VM_4，可以使其他节点提供更多的资源去处理运算量较大的任务。

在不同任务数的性能实验中，设置所有节点参与任务，当任务数小于 4（$M<4$）时，策略数小于 4.29×10^9。实验最小任务数设置为 4（每增加一个任务进行一组实验，每组实验进行 5 次，最终结果取平均值）。设置 3 种算法的最大迭代次数 $G=50$，最大粒子数 $NP=500$。图 10 显示了不同任务数下 3 种算法性能比较。

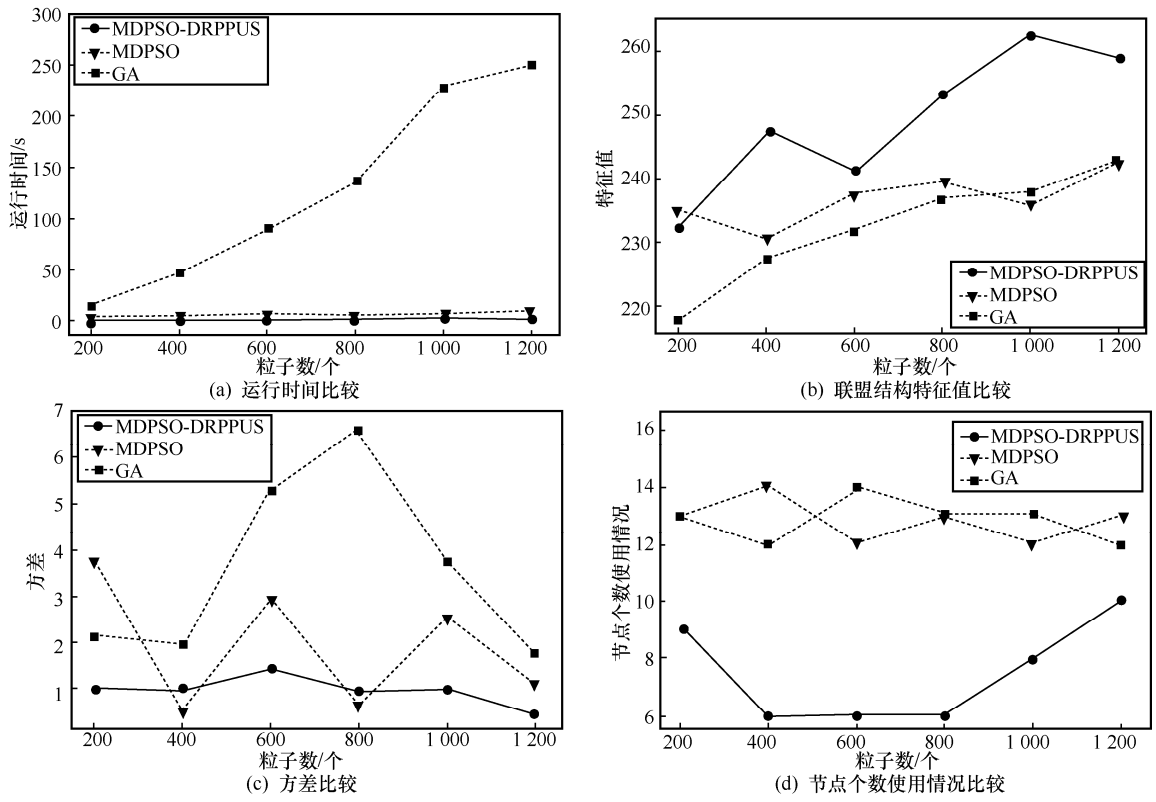
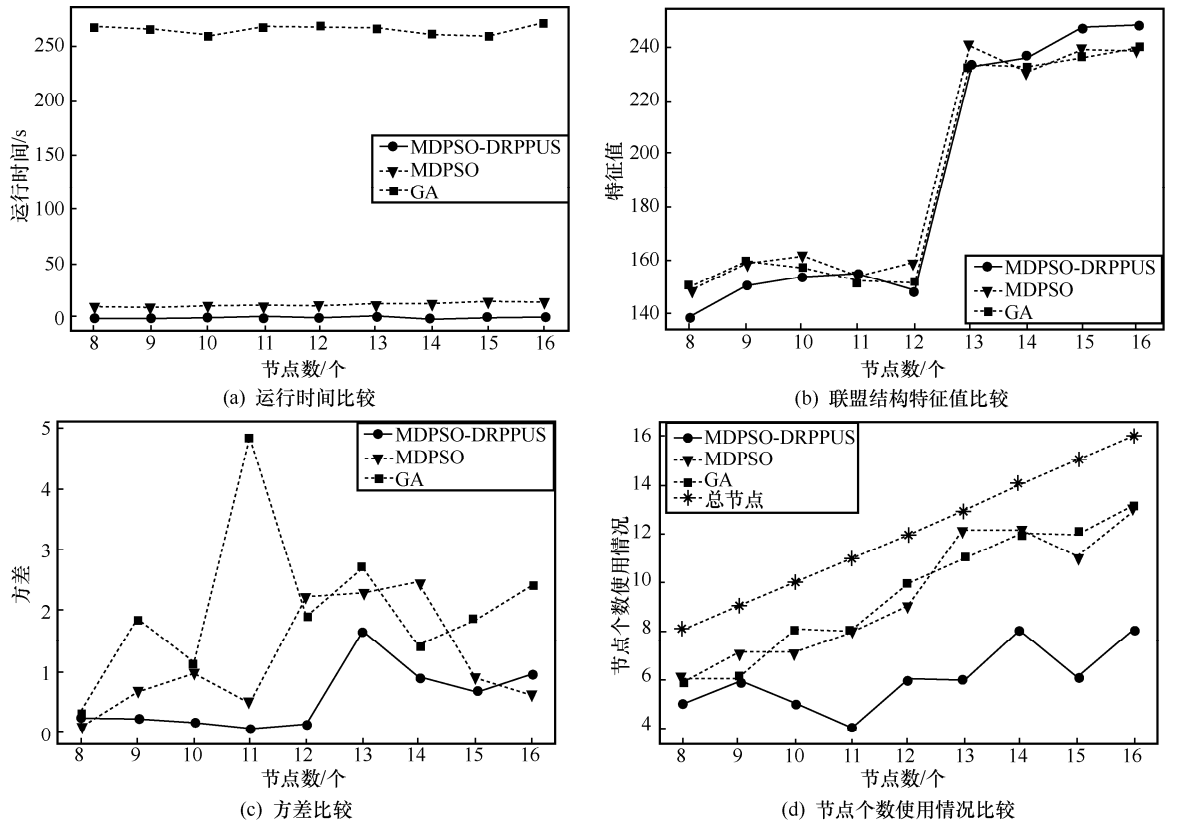


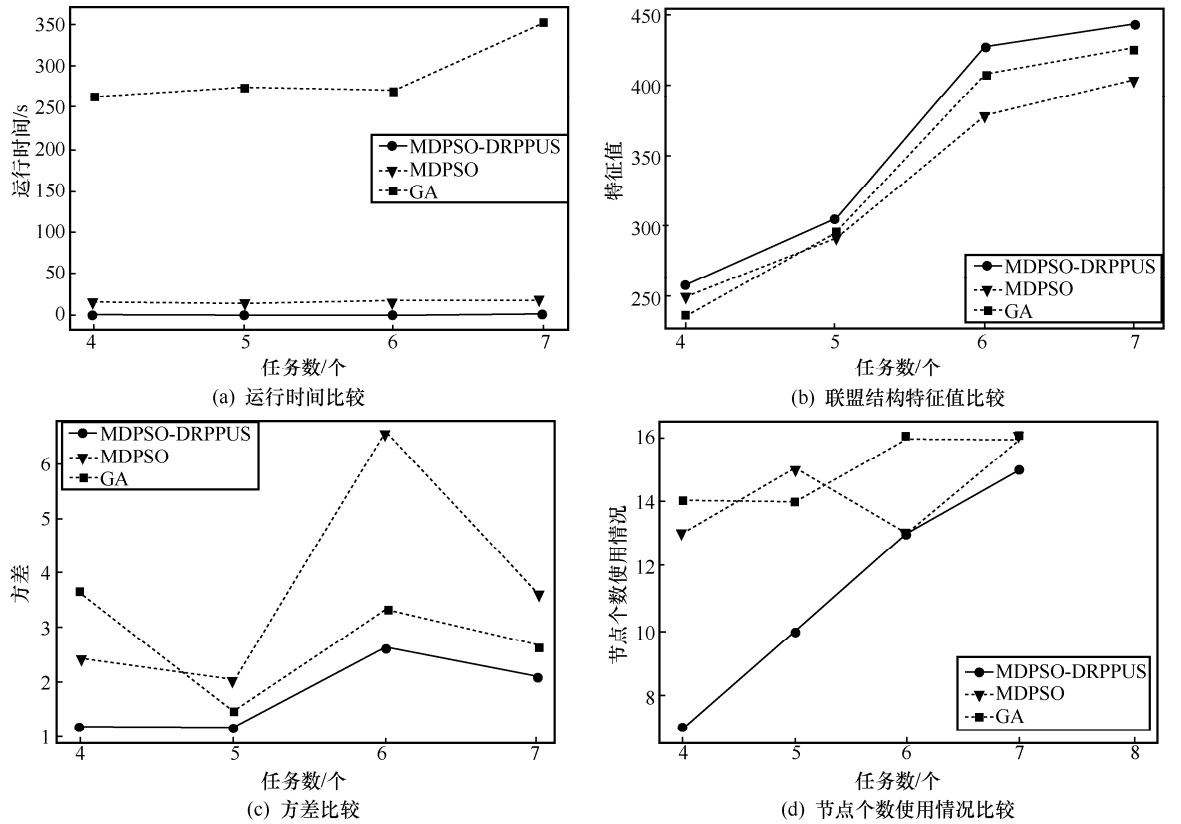
图 8　迭代次数为 10 时不同粒子数下 3 种算法性能比较



(a) 运行时间比较　(b) 联盟结构特征值比较

(c) 方差比较　(d) 节点个数使用情况比较

图 9　不同节点数下 3 种算法性能比较



(a) 运行时间比较　(b) 联盟结构特征值比较

(c) 方差比较　(d) 节点个数使用情况比较

图 10　不同任务数下 3 种算法性能比较

由图 10 可知，当任务数为 8 个时，由于添加了任务量过大的 M_8，3 种算法在当前条件下均不能完成任务，这是因为 M_8 的计算负载过大，从而导致边缘节点无法按时完成任务。

综上所述，相较于 MDPSO 和 GA，本文提出的 MDPSO-DRPPUS 算法运行时间大幅度降低，开发能力也得到了极大提高。在面对此类复杂的组合优化问题时，MDPSO 和 GA 依旧面临陷入局部最优解的困境，但所提算法可以通过大幅度增加粒子数避免这一缺陷。实验结果表明，MDPSO-DRPPUS 的粒子数增加到 5 000 个时算法运行时间为 10 s 左右。增加粒子数后，联盟结构效益也随之得到提升。此外，MDPSO-DRPPUS 所得联盟结构的均衡性和节点使用率均优于 MDPSO 和 GA。

因此，与 MDPSO 和 GA 相比，MDPSO- DRPPUS 在优化速度及最优解质量方面都得到了较好的提升。

4　结束语

本文提出了 MDPSO-DRPPUS 算法进行最优联盟结构搜索，很好地解决了多任务并发边缘计算环境中的最优联盟结构搜索问题。实验结果表明，MDPSO-RPPUS 算法有很强的搜索能力，收敛速度很快（能在迭代 10 次以内收敛），运行时间相对于 MDPSO 和 GA 而言大幅度降低，搜索到的最优联盟结构效益和联盟结构均衡性也相对较优。

在本文场景中，联盟结构的数量随着任务数和边缘节点数的增加呈指数级增长。虽然智能算法对联盟结构的搜索已经取得了一定的成果，但由于策略空间巨大，下一步可否先将庞大的策略空间处理后再进行搜索成为一个值得研究的课题。

参考文献：

[1] KEKKI S, FEATHERSTONE W, FANG Y, et al. MEC in 5G networks[R]. 2018.

[2] DANG V D, DASH R K, ROGERS A, et al. Overlapping coalition formation for efficient data fusion in multi-sensor networks[C]//Proceedings of AAAI. Palo Alto: AAAI Press, 2006: 635-640.

[3] HAN Z, POOR H V. Coalition games with cooperative transmission: a cure for the curse of boundary nodes in selfish packet-forwarding wireless networks[J]. IEEE Transactions on Communications, 2009, 57(1): 203-213.

[4] ZHANG Z F, SONG L Y, HAN Z, et al. Coalitional games with overlapping coalitions for interference management in small cell networks[J]. IEEE Transactions on Wireless Communications, 2014, 13(5): 2659-2669.

[5] ZHAN S C, NIYATO D. A coalition formation game for remote radio head cooperation in cloud radio access network[J]. IEEE Transactions on Vehicular Technology, 2017, 66(2): 1723-1738.

[6] LI X M, WAN J F, DAI H N, et al. A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing[J]. IEEE Transactions on Industrial Informatics, 2019, 15(7): 4225-4234.

[7] 胡山立, 石纯一, 李少芳. 给定限界的势结构分组与联盟结构生成[J]. 计算机学报, 2012, 35(12): 2618-2624.
HU S L, SHI C Y, LI S F. Cardinality structure grouping and coalition structure generation with given required bound[J]. Chinese Journal of Computers, 2012, 35(12): 2618-2624.

[8] RAHWAN T. Algorithms for coalition formation in multi-agent systems[D]. Southampton: University of Southampton, 2007.

[9] 张新良, 石纯一. 多 Agent 联盟结构动态生成算法[J]. 软件学报, 2007, 18(3): 574-581.
ZHANG X L, SHI C Y. A dynamic formation algorithm of multi-agent coalition structure[J]. Journal of Software, 2007, 18(3): 574-581.

[10] 徐广斌, 刘惊雷. 带有联盟个数约束的最优联盟结构生成[J]. 南京大学学报(自然科学), 2015, 51(4): 749-761.
XU G B, LIU J L. The optimal coalition structure generation with the constrained number of coalition[J]. Journal of Nanjing University (Natural Sciences), 2015, 51(4): 749-761.

[11] SEN S, DUTTA P S. Searching for optimal coalition structures[C]// Proceedings of Fourth International Conference on MultiAgent Systems. Piscataway: IEEE Press, 2000: 287-292.

[12] YANG J G. Coalition formation mechanism in multi-agent systems based on genetic algorithms[J]. Applied Soft Computing, 2007, 7(2): 561-568.

[13] CONTRERAS J P, BOSCH P, VARAS M, et al. A new genetic algorithm encoding for coalition structure generation problems[J]. Mathematical Problems in Engineering, 2020, 2020: 1203248.

[14] 蒋建国, 夏娜, 齐美彬, 等. 一种基于蚁群算法的多任务联盟串行生成算法[J]. 电子学报, 2005, 33(12): 2178-2182.
JIANG J G, XIA N, QI M B, et al. An ant colony algorithm based multi-task coalition serial generation algorithm[J]. Acta Electronica Sinica, 2005, 33(12): 2178-2182.

[15] LIN C F, HU S L. Multi-task overlapping coalition parallel formation algorithm[C]//Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems. New York: ACM Press, 2007: 1-3.

[16] ZHANG G F, YANG R Z, SU Z P, et al. Using binary particle swarm optimization to search for maximal successful coalition[J]. Applied Intelligence, 2015, 42(2): 195-209.

[17] 许金友. 基于改进 DPSO 算法的多任务并行联盟生成研究[D]. 大连: 大连理工大学, 2009.
XU J Y. Research on multi-task parallel coalition generation based on modified discrete particle swarm optimization algorithm[D]. Dalian: Dalian University of Technology, 2009.

[18] HU Y N, LI C S, ZHANG K J. A method of searching for optimal coalition structure for solving resource scheduling problem of overall load balancing in edge computing environments[J]. Journal of Physics: Conference Series, 2020, 1550(3): 032080.

[19] ZHANG K J, FU Y, HU Y N, et al. Scheduling strategy for computational-intensive data flow in generalized cluster environments[J]. Applied Soft Computing, 2019, 82: 105571.

[20] ZHANG K J, HU Y N, TIAN F, et al. A coalition-structure's generation method for solving cooperative computing problems in edge computing environments[J]. Information Sciences, 2020, 536: 372-390.

[21] WANG D S, TAN D P, LIU L. Particle swarm optimization algorithm: an overview[J]. Soft Computing, 2018, 22(2): 387-408.

[22] 王炜, 陈渺, 李尚华. 一类联合最大特征值函数优化问题[J]. 吉林师范大学学报(自然科学版), 2014, 35(1): 28-31.
WANG W, CHEN M, LI S H. Solving a minimization problem for a class of joint maximum eigenvalue functions[J]. Jilin Normal University Journal (Natural Science Edition), 2014, 35(1): 28-31.

[23] HART S, KURZ M. Endogenous formation of coalitions[J]. Econometrica, 1983, 51(4): 1047.

[24] SANDHOLM T, et al. Coalition structure generation with worst case guarantees[J]. Artificial Intelligence, 1999, 111(1/2): 209-238.

[25] ASKARI Q. Political optimizer: a novel socio-inspired meta-heuristic for global optimization[J]. Knowledge-Based Systems, 2020, 195: 105709.

[26] WANG X L, DANG J W, ZHAO S X, et al. Coalition structure generation in edge computing environment with multitasking concurrency[J]. IEEE Internet of Things Journal, 2022: doi.org/10.1109/JIOT.2022.3217171.

[作者简介]

赵庶旭（1976– ），男，山东青岛人，博士，兰州交通大学教授，主要研究方向为智能交通、边缘计算等。

韦萍（1996– ），女，甘肃白银人，兰州交通大学硕士生，主要研究方向为边缘计算、边缘联盟等。

王小龙（1989– ），男，甘肃定西人，兰州交通大学博士生，主要研究方向为边缘计算、边缘联盟等。