基于 Spark 框架和 ASPSO 的并行划分聚类算法

毛伊敏[1]，甘德瑾[1]，廖列法[1]，陈志刚[2]
（1. 江西理工大学信息工程学院，江西 赣州 341000；2. 中南大学计算机学院，湖南 长沙 410083）

摘 要：针对划分聚类算法处理海量的数据存在的数据离散系数较大与抗干扰性差、局部簇簇数难以确定、局部簇质心随机性及局部簇并行化合并效率低等问题，提出了一种基于 Spark 框架和粒子群优化自适应策略（ASPSO）的并行划分聚类（PDC-SFASPSO）算法。首先，提出了基于皮尔逊相关系数和方差的网格划分策略获取数据离散系数较小的网格单元并进行离群点过滤，解决了数据离散系数较大与抗干扰性差的问题；其次，提出了基于势函数与高斯函数的网格划分策略，获取局部聚类的簇数，解决了局部簇簇数难以确定的问题；再次，提出了 ASPSO 获取局部簇质心，解决了局部簇质心的随机性问题；最后，提出了基于簇半径与邻居节点的合并策略对相似度大的簇进行并行化合并，提高了局部簇并行化合并的效率。实验结果表明，PDC-SFASPSO 算法在大数据环境下进行数据的划分聚类具有较好的性能表现，适用于对大规模的数据集进行并行化聚类。

关键词：Spark 框架；并行划分聚类；网格划分；粒子群优化自适应策略；并行化合并
中图分类号：TP311
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2022054

Parallel division clustering algorithm based on Spark framework and ASPSO

MAO Yimin[1], GAN Dejin[1], LIAO Liefa[1], CHEN Zhigang[2]
1. School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
2. College of Computer Science and Engineering, Central South University, Changsha 410083, China

Abstract: To deal with the problems that the partition clustering algorithm for processing massive data encountered problems such as large data dispersion coefficient and poor anti-interference, difficulty to determine the number of local clusters, local cluster centroids randomness, and low efficiency of local cluster parallelization and merging, a parallel partition clustering algorithm based on Spark framework and ASPSO (PDC-SFAS PSO) was proposed. Firstly, a meshing strategy was introduced to reduce the data dispersion coefficient of the data division and improve anti-interference. Secondly, to determine the number of clusters, meshing strategy based on potential function and Gaussian function were proposed, which formed an area with different sample points as the core clusters, and obtained the number of local clusters. Then, to avoid local cluster centroids randomness, ASPSO was proposed. Finally, a local cluster merging strategy based on cluster radius and neighbor nodes was introduced to merge clusters with large similarity based on the Spark parallel computing framework, which improved the efficiency of parallel merging of local clusters. Experimental results showed that the PDC-SFASPSO algorithm has good performance in data partitioning and clustering in a big data environment, and it was suitable for parallel clustering of large-scale data sets.

Keywords: Spark framework, parallel division clustering, grid division, ASPSO, parallel merge

收稿日期：2021−09−12；**修回日期**：2021−12−10
通信作者：陈志刚，czg@csu.edu.cn
基金项目：国家自然科学基金资助项目（No.41562019）；科技创新 2030-“新一代人工智能”重大基金资助项目（No.2020AAA0109605）
Foundation Items: The National Natural Science Foundation of China (No.41562019), Technological Innovation 2030-Next-Generation Artificial Intelligence Major Projects (No.2020AAA0109605)

0　引言

聚类算法根据数据的相似特征将数据集划分成不同的类别，同一类别下的对象具有一定的相似性，而不同类别对象之间则差异较大[1]，因其能够发现样本数据潜在的分布模式，被广泛应用于计算机科学、生物信息学、图像处理、社交网络、天文学以及许多其他的领域。在聚类算法中，基于划分的聚类算法，如 K-means 算法[2]、K-中心点算法[3]和 CLARANS（clustering algorithm based on randomized search）[4]，由于聚类思想简单且聚类可行性高，受到人们的广泛关注。

随着 5G 时代的到来，数据规模爆炸式增长。相较于传统数据，大数据拥有数据规模大、数据种类多样化、数据价值密度低、数据增长速度快等基本特征[5-6]。然而，传统的划分聚类算法在处理大数据时，其时间复杂度以几何级数增长。因此，如何使划分聚类算法更快地处理大数据，是国内外重点关注的问题。

随着传统的数据挖掘算法在分布式计算框架的广泛应用，Spark 凭借计算速度快、简洁易用、通用性强和支持多种运行模式等优势深受广大研究人员青睐。其中，Mugdha 等[7]最早提出了 K-means 与 Spark 结合的近似算法，但该算法易受噪声数据影响，划分后的数据网格存在数据离散系数较大与抗干扰性差的问题。针对此问题，王海艳等[8]提出了基于 Spark 与密度区域划分（SP-DAP, Spark and density-area partition）算法，提取数据集中的高密度数据区域，极大地减小了数据划分后的离散系数并增强了算法的抗干扰性。但该算法无法准确获取局部簇的簇数，从而导致对大型数据的聚类存在局部簇的簇数难以确定的问题。为了解决此问题，Wang 等[9]提出基于 Spark 与 K-means（SK-means, Spark and K-means）并行聚类算法，通过将数据划分为几个可重叠子集，并进行预处理得到簇数。徐鹏程等[10]在 Spark 平台上引入 Canopy 算法和最大最小距离方法，通过简单的距离计算把初始数据划分为多个子集，获取局部聚类的簇数，解决了局部簇的簇数难以确定的问题。虽然这些算法对局部簇的簇数难以确定的问题进行了改进，但是并没有避免局部簇质心的随机性对聚类效果的影响。

近年来，群体智能优化算法凭借其寻优效果好、易实现等优势得到了广泛应用。因此，许多学者引入群体智能算法对数据的局部簇质心进行优化。例如，Multazam 等[11]在 Spark 平台上提出遗传算法与 K-means 相结合（SP-GAKMS, Spark and genetic algorithmwith K-means），通过对种群个体的多次选择、交叉以及变异的遗传操作，使种群个体逐渐优化并逐步逼近最优解，最终得到最优的初始质心集。许明杰等[12]在 Spark 环境下提出 PSOK-means（particle swarm optimization and K-means）算法，利用粒子群优化（PSO, particle swarm optimization）算法来提高 K-means 的全局搜索能力，得到初始质心。Gao 等[13]利用最优化问题对 PSO 算法进行改进，提出了 SP-PSOK-means（Spark and PSO with K-means）算法，利用远离个体最差经验和最差群体经验提高局部搜索能力，获取全局最优初始质心。虽然这些算法成功应用了群优化算法来获取局部簇的初始质心，但存在局部搜索能力差、搜索精度不高且易陷入局部最优值等缺点，导致无法获取全局最优初始质心，因此算法的初始质心优化能力有待进一步提升。

此外，基于 Spark 并行计算框架下的划分聚类算法在进行局部簇的并行化合并过程时存在算法局部簇并行化合并效率低的问题。为了解决这个问题，Agrawal 等[14]提出基于 Spark 节点相似度的局部簇合并（SP-LCMNS, Spark and local cluster merging and node similarity）算法，通过合并二次划分算法与群体结构，避免了点与边集的重复计算，极大地提高了局部簇的并行化合并效率；Lai 等[15]提出了基于 Spark 局部聚合的自动迭代聚簇算法（SP-LAICA, Spark and local aggregation and iterative clustering algorithm），通过寻找局部簇数据集中连接紧密的节点集，并迭代合并局部簇，实现了对局部簇的高效率并行化合并。虽然上述算法对提升局部簇的并行化合并的效率有一定提升，但这些算法都存在一定的局限性，导致簇的并行化合并效率降低。因此，局部簇并行化合并效率低的问题仍是亟待解决的问题。

综上，如何有效减小数据离散系数、增强算法抗干扰性、确定局部聚类簇数、避免局部簇质心随机性以及提高局部簇并行化合并效率等仍然是目前亟待解决的问题。针对这些问题，本文提出了一种基于 Spark 框架和粒子群优化自适应策略（ASPSO,

adaptive strategy based on particle swarm optimization）下的并行划分聚类（PDC-SFASPSO, parallel division clustering based on Spark framework and ASPSO）算法。本文的主要工作如下。1) 提出了基于皮尔逊相关系数和方差（PCCV, Pearson's correlation coefficient and variance）的网格划分策略，计算数据网格的皮尔逊相关系数与相关系数阈值，通过与阈值比较，对数据网格进行划分，获取数据离散系数较小的网格单元 $G_1,G_2,G_3,\cdots,G_m$，并设计离群因子对网格单元进行离群点过滤，解决了数据离散系数较大与抗干扰性差的问题。2) 提出了基于势函数与高斯函数（PFGF, potential function and Gaussian function）的网格划分策略，对数据点进行局部区域的有效覆盖，并提出更新函数 FU(x_i, y_j)，更新数据集中的样本点，形成以不同样本点为核心的区域簇，获取局部聚类的簇数，解决了局部簇的簇数难以确定的问题。3) 提出了 ASPSO，计算自适应参数 η，通过自适应参数更新粒子的位置和速度，获取局部簇质心，解决了局部簇质心的随机性问题。4) 提出了基于簇半径与邻居节点（CRNN，cluster radius and neighbor node）的局部簇合并策略，计算出邻居节点，并根据簇的相似性函数 CSM(n_i, n_j)进行相似度判断，结合 Spark 并行计算框架将相似度大的簇进行合并，避免了在并行化运算过程中对所有簇的点与边集同时展开搜索，提高了局部簇并行化合并的效率。

1 相关概念介绍

定义 1 势函数。势函数可以对数据点的作用势进行分析，度量样本空间中 2 个数据点随距离的变化情况[16]。设有一个样本集 S_i，x_1 和 x_2 为数据集中的 2 个样本点，则势函数 $\gamma(x_1,x_2)$ 可以表示为

$$\gamma(x_1,x_2)=\frac{1}{1+Td^2(x_1,x_2)} \tag{1}$$

其中，T 为常数，$d^2(x_1,x_2)$ 为 x_1 和 x_2 之间的距离。

定义 2 高斯核函数。高斯核函数是某种沿径向对称的标量函数，可以将有限维数据映射到高维空间，并将数据集划分成多个不同子空间，从而进行局部计算[17]。设 σ 表示带宽，x 表示样本点，x' 表示核中心，则高斯核函数 $k(x,x')$ 可以表示为

$$k(x,x')=\exp\left(\frac{\|x-x'\|}{\sigma^2}\right) \tag{2}$$

定义 3 邻居节点。邻居节点表示簇与簇之间的交集节点，可以衡量簇与簇之间亲密程度。设有 2 个簇 C_i,C_j 的数据点，对于任意样本点 x_i，如果其到质心的距离小于簇半径 R_i 和 R_j，则此样本点称为邻居节点，其集合称为邻居节点集[18]。

定义 4 PSO 算法。PSO 算法是一种基于群体智能的全局随机搜索算法，可以对粒子的位置进行不断调优，求解粒子最优化问题[19]。该算法主要包括 3 个阶段：粒子位置与速度的初始化、粒子速度更新、粒子位置更新。其具体步骤如下。

1) 粒子位置与速度的初始化

$$x=(x_1,x_2,\cdots,x_i\cdots,x_n)^{\mathrm{T}} \tag{3}$$

$$x_k^t=(x_{k,1}^t,\cdots,x_{k,i}^t,\cdots,x_{k,n}^t)^{\mathrm{T}} \tag{4}$$

$$v_k^t=(v_{k,1}^t,\cdots,v_{k,i}^t,\cdots,v_{k,n}^t)^{\mathrm{T}} \tag{5}$$

2) 粒子速度更新

设 $p_{k,i}^t$ 为粒子的历史最佳位置，$p_{g,i}^t$ 为种群历史最佳位置，f 为粒子的自适应值，则粒子速度更新可表示为

$$v_{k,i}^{t+1}=\omega v_{k,j}^t+c_1\mathrm{rand}(p_{k,i}^t-x_{k,i}^t)+c_2\mathrm{Rand}(p_{g,j}^t-x_{k,j}^t) \tag{6}$$

其中，ω 为惯性权重，表示之前的速度对当前速度的影响；c_1 和 c_2 为学习因子，rand 和 Rand 为 0～1 的随机数。

3) 粒子位置更新

设 $x_{k,i}^t$ 为粒子当前位置，$v_{k,i}^{t+1}$ 为粒子更新的速度，则粒子位置更新可表示为

$$x_{k+1}^{t+1}=x_{k,i}^t+v_{k,i}^{t+1} \tag{7}$$

2 PDC-SFASPSO 算法

2.1 算法思想

PDC-SFASPSO 算法主要包括 3 个阶段：数据划分、局部聚类、局部簇合并。1) 数据划分阶段提出网格划分策略 PCCV，进行数据划分。2) 局部聚类阶段，首先提出 PFGF 策略与更新函数 FU(x_i, y_j)获取局部聚类的簇数；然后提出 ASPSO 初始化局部簇质心；最后结合 Spark 提出并行化局部聚类策

略（SPPLCS, Spark and parallel local clustering strategy）进行局部簇的并行化计算，实现局部聚类。3) 局部簇合并阶段提出局部簇合并策略 CRNN，合并相似度大的局部簇。

2.2 数据划分

目前大数据环境下的划分聚类算法中，在对数据划分时存在数据离散系数较大与抗干扰性差的问题。因此，本文在进行数据划分的过程中提出了网格划分策略 PCCV 来解决此问题。该策略主要包括 3 个步骤：数据集的粗略划分、网格的划分、离群点的过滤。

2.2.1 数据集的粗略划分

对于初始数据集，可以对数据先进行粗略划分，获取数据离散系数较小的网格。其具体过程为：首先获取划分数据集，并将其标记为 G_s；其次提出分割函数 $\mathrm{FD}(x_i)$ 计算出划分阈值，分别与每个数据点比较，将大于阈值的数据放入网格 $G_{\max}$ 中，小于阈值的数据则放入 $G_{\min}$ 中；最后获得 2 个数据网格 $G_{\max}$ 与 $G_{\min}$。

定理 1　分割函数 $\mathrm{FD}(x_i)$。已知空间数据集中第 i 维度的数据的方差为 S_i，数据之和为 d_i，网格中数据点的个数为 num，则分割函数 $\mathrm{FD}(x_i)$ 为

$$k=\left\{\max\left(\frac{S_i}{d_i}\right)\mid i=1,2,\cdots,q\right\} \tag{8}$$

$$\mathrm{FD}(x_i)=\frac{\sum_{i=1}^{\mathrm{num}}x_i^k}{\mathrm{num}} \tag{9}$$

其中，x_i^k 为第 k 维度下的数据值。

证明　由于方差越大，该维度所带的信息量越多。对于不同维度下方差相同的数据，d_i 越大表明数据越离散，d_i 越小表明数据越集中。因此，网格的划分维度 k 可根据 $\frac{S_i}{d_i}$ 确定，并选取 $\frac{S_i}{d_i}$ 的最大值作为网格的划分维度。又由于均值可以反映数据的整体倾向，因此，选取该维度下数据的均值作为数据划分的网格分割函数。证毕。

2.2.2 网格的划分

在获取 $G_{\max}$ 与 $G_{\min}$ 这 2 个数据网格之后，由于分割函数只能对数据集进行粗略划分，无法对相似度较大的数据进行网格划分，导致无法获取网格单元。因此，需要对网格 $G_{\max}$ 与 $G_{\min}$ 进行进一步的数据划分，获取离散系数较小的网格单元，其具体过程如下。

1) 提出数据的皮尔逊相关系数阈值 PCC_k。计算网格中数据点的 PCC_k 值，以 PCC_k 值作为网格划分阈值对数据网格进行划分，通过比较数据的皮尔逊相关系数与 PCC_k 的大小，将系数大于 PCC_k 的数据标记为 core，系数小于 PCC_k 的数据标记为 uncore。

2) 将网格中 2 种数据 core 与 uncore 分别划分为 2 个更小的网格，并取消标记。

3) 对网格数据点个数进行判断，如果数据点的个数大于网格单元的阈值 maxNum，则返回步骤 1)；否则停止对网格进行划分。其中 maxNum 表示数据集的数据个数 n 与并行化节点 Partition 个数的比值。

4) 将划分好的网格单元进行标记，得到网格单元 $G_1,G_2,G_3,\cdots,G_m$。

定理 2　皮尔逊相关系数阈值 PCC_k。设 $\mathrm{PCC}_{i,j}$ 为任意 2 个数据点的皮尔逊相关系数值，G_{num} 为网格单元的数据点个数，sum 为求和函数，ω 为数据点的密度权重，则 PCC_k 为

$$\mathrm{PCC}_k=\frac{\mathrm{sum}(\mathrm{PCC}_{i,j})\omega}{G_{\mathrm{num}}} \tag{10}$$

$$\omega=\frac{G_{\mathrm{num}}}{\sum_{k=1}^{m}\sqrt{(x_{k,i}-x_{k,j})^2}} \tag{11}$$

证明　$\mathrm{PCC}_{i,j}$ 代表数据点之间的关联程度，即 $\mathrm{PCC}_{i,j}$ 越大，数据点之间的相似度越大。将式(11)代入式(10)可得 $\mathrm{PCC}_k=\frac{\mathrm{sum}(\mathrm{PCC}_{i,j})}{\sum_{k=1}^{m}\sqrt{(x_{k,i}-x_{k,j})^2}}$。$\sum_{k=1}^{m}\sqrt{(x_{k,i}-x_{k,j})^2}$ 的大小反映了数据的离散化程度，其值越大，则数据越离散；其值越小，则数据越集中。因此 $\frac{\mathrm{sum}(\mathrm{PCC}_{i,j})}{\sum_{k=1}^{m}\sqrt{(x_{k,i}-x_{k,j})^2}}$ 的大小可以很好地对数据相似程度进行衡量，PCC_k 可以作为网格划分的皮尔逊相关系数阈值。证毕。

2.2.3 离群点的过滤

在获取网格单元 $G_1,G_2,G_3,\cdots,G_m$ 后，由于网格单元存在离群点，会造成算法的抗干扰性差。因此，为了增强算法的可抗干扰性，设计离群因子 GOF 来过滤离群点，具体过程为：计算每个网

格单元中数据的 GOF 值，若 GOF 值远大于 ε（网格单元数据阈值），则将该数据点视为离群点，并对其进行删除。

定理 3 离群因子 GOF 。假设 $d(x_i,x_j)$ 表示网格中 2 个数据点的欧氏距离，$\overline{x}$ 表示网格单元的中心点，则离群因子 GOF 为

$$\text{uncore } d=\sum_{i=1}^{m}\sum_{j=1}^{m}d(x_i,x_j)=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}(x_i-x_j)^2} \tag{12}$$

$$\overline{x}=\frac{1}{m}\sum_{i=1}^{m}x_i \tag{13}$$

$$\text{GOF}=\frac{\sqrt{\sum_{k=1}^{t}(x_{k,i}-\overline{x}_{k,j})^2}}{d} \tag{14}$$

证明 d 表示当前网格中某一数据点与其余的 $m-1$ 个数据点的欧氏距离，其大小可以表示网格的密度。d 越小表明网格的密度越大，d 越大表明网格的密度越小。$\sqrt{\sum_{k=1}^{t}(x_{k,i}-\overline{x}_{k,j})^2}$ 表示当前数据点到网格中心的距离。对于离群点来说，其值会相对于其他数据点更大。如果数据点的 GOF 值远大于 ε，则可以将此数据点过滤，因此，可以用 GOF 来过滤网格的离群点。证毕。

数据划分的伪代码如算法 1 所示。

算法 1 数据划分

输入 初始数据集 S

输出 数据网格单元 $G_1,G_2,G_3,\cdots,G_m$

1) for each x_i in S do

2) $k=\left\{\max\left(\dfrac{S_i}{d_i}\right)\middle| i=1,2,\cdots,m\right\}$

3) $\text{ave}=\text{average}(x_i^k)$

4) $G_{\max}$ and $G_{\min}\leftarrow \text{FD(ave)}$

5) end for

6) $\text{PCC}_k=\dfrac{\text{sum}(\text{PCC}_{i,j})\omega}{G_{\text{num}}}$

7) while $\left(\text{count}>\dfrac{n}{\text{partition}}\right)$

8) for each x_i in $G_{\max}$ do

9) if ($x_i>\text{PCC}_k$)

10) $x_i=\text{core}$

11) else

12) $x_i=\text{uncore}$

13) end if

14) end for

15) for each x_j in $G_{\min}$ do

16) if ($x_j>\text{PCC}_k$)

17) $x_j=\text{core}$

18) else

19) $x_j=\text{uncore}$

20) end if

21) end for

22) end while

23) get $G_1,G_2,G_3,\cdots,G_m$

24) return $G_1,G_2,G_3,\cdots,G_m$

2.3 局部聚类

目前在大数据环境下的划分聚类算法中，对数据的局部聚类需要对网格单元的数据进行质心初始化，再结合并行化计算完成局部聚类。然而，在实现质心初始化的过程中，由于无法确定局部聚类的簇数，导致不能更好地确定初始质心集的个数，无法实现较优的初始质心。因此，为了获取局部聚类的簇数，更好地实现初始化质心，完成并行化局部聚类，提出 LC 策略，其具体过程包括 3 个步骤：局部聚类簇数获取、局部簇质心初始化、局部簇并行化计算。

2.3.1 局部聚类簇数获取

为了有效地实现局部聚类，需要优先获取局部聚类的簇数，因此本文提出了 PFGF 策略，通过势函数与高斯核函数来完成数据的覆盖与搜索，获取局部聚类的簇数。其具体过程为：首先，对数据集中任意一对数据 x_i 和 x_j 通过式(1)计算其作用势 $\gamma(x_i,x_j)$，并以 x_i 为基准样本，将其他的样本点对 x_j 的作用势进行累加，得到每个样本点的作用势集合为 $\rho=\{\rho_1,\rho_2,\cdots,\rho_n\}$；其次，为了对原始数据进行全局搜索，从 ρ 中选择最大作用势 ρ_i 放入一个空的集合 $M\{\}$ 中，并以 ρ_i 为当前的高斯核中心，以给定的核宽 σ 建立相应的高斯核来对原始数据的一个局部区域有效覆盖；最后，为了寻找新的最大势值，需要消除当前高斯核所覆盖的局部区域的样本势值，提出基于高斯核函数的更新函数 $\text{FU}(x_i,y_j)$ 对数据集中的其他样本点进行更新。

定理 4 更新函数 $\text{FU}(x_i,y_j)$。假设当前的高斯核中心为 ρ_i，集合中的样本点为 ρ_j，则其更新函数

$\mathrm{FU}(x_i, y_j)$ 为

$$\mathrm{FU}(x_i, y_j) = \rho_i - \rho_j \exp\left(-\frac{1}{2\sigma_k^2}\| x_i - x_j \|^2\right) \quad (15)$$

其中，σ_k 表示核宽，$\exp\left(-\frac{1}{2\sigma_k^2}\| x_i - x_j \|^2\right)$ 表示高斯内核。

证明　由高斯核函数的衰减特性可知，当样本点距离高斯核中心较远时，x_j 对 x_i 的影响十分小，又由于 $\exp\left(-\frac{1}{2\sigma_k^2}\| x_i - x_j \|^2\right)$ 表示高斯内核，因此 S 中各个样本点的势值都可以有效更新。证毕。

当更新后的势值满足 $\max\{\rho_1', \rho_2', \cdots, \rho_n'\} > \delta$ 时，即可从 ρ' 中选择势值最大的样本点，放入集合 M 中。通过这种方式继续寻找下一个新的样本点，直到数据集 S 被完全覆盖，集合 M 中样本点的个数便是局部聚类的簇数。

2.3.2　局部簇质心初始化

在获取了局部聚类的簇数之后，为了解决局部簇质心随机性的问题，本文提出了策略 ASPSO。该策略主要包括 2 个阶段：自适应参数确定、质心初始化。自适应参数确定阶段提出 AS 策略，引入柯西变异算子，设置粒子平均速度与 η 来作为自适应参数；质心初始化阶段通过 AS 策略与 PSO 算法相结合，并根据自适应参数，对粒子的速度与位置不断更新，跳出局部最优，获取初始化质心。

1) 自适应参数确定

在实现质心初始化的过程中，提出了粒子的收敛性，由此性质可知粒子最终收敛于 $\rho_{k,i}^t = \rho_{g,i}^t$，算法将停止运行，如果算法没有在收敛之前得到全局最优解，就会导致过早收敛，陷入局部最优解。

定理 5　粒子的收敛性。假设 $\varepsilon > 0$, $\max(\|\alpha\|, \|\beta\|) < 1$，存在 $N \geqslant 1$，使对于任意的 $n \geqslant N$，有 $\left\|\rho_{k,i}^t - \rho_{g,i}^t\right\| < \varepsilon$。

证明　$\lim\limits_{t\to+\infty} x_{k,i}^t = \lim\limits_{t\to+\infty}(k_1 + k_2\alpha^t + k_3\beta^t) = \dfrac{c_1\rho_{k,i}^t + c_2\rho_{g,i}^t}{c_1 + c_2}$

由式(8)可知，当 $\max(\|\alpha\|, \|\beta\|) < 1$ 时，有

$$\lim_{t\to+\infty} v_{k,i}^t = \lim_{t\to+\infty}(x_{k,i}^{t+1} - x_{k,i}^t) = \lim_{t\to+\infty} k_2\alpha^t(\alpha - 1) + k_3\beta^t(\beta - 1) = 0$$

即

$$\lim_{t\to+\infty} x_{k,i}^{t+1} = \lim_{t\to+\infty} x_{k,i}^t, x_{k+1,i}^{t+1} = x_{k,i}^t + v_{k,i}^{t+1} = x_{k,i}^t + wv_{k,i}^t - x_{k,i}^t(c_1 + c_2) + c_1\rho_{k,i}^t + c_2\rho_{g,i}^t$$

对两边求极限得 $\lim\limits_{t\to+\infty} x_{k,i}^t = \rho_{k,i}^t = \rho_{g,i}^t$，证毕。

因此，为了实现初始质心的全局最优解，需要设计自适应参数来避免局部最优。为此，PDC-SFASPSO 算法设计了 AS 策略，来确定自适应参数，其具体过程如下。

① 提出粒子群体平均速度 $\overline{v_{k,i}}$ 作为第一个自适应参数，计算出 $\overline{v_{k,i}}$ 值，设置其为控制变异步长的参数。

定理 6　粒子群体平均速度 $\overline{v_{k,i}}$。已知粒子的个数为 n，粒子的速度为 $v_{k,i}$，则粒子的平均速度为

$$\overline{v_{k,i}} = \frac{1}{n}\sum_{i=1}^{n} v_{k,i} \quad (16)$$

证明　由于在初始阶段，粒子的平均速度较大，随着粒子位置的不断更新，由 $v_{k,i}^{t+1} = \omega v_{k,j}^t + c_1\mathrm{rand}(p_{k,i}^t - x_{k,i}^t) + c_2\mathrm{Rand}(p_{g,j}^t - x_{k,j}^t)$ 可知，粒子速度不断减小，导致平均速度相对减小，群体也开始慢慢收敛，即其平均速度的变化的趋势与收敛的趋势是一致的，因此选取平均速度作为控制变异步长的自适应参数。证毕。

② 提出柯西变异算子的离散性。由此性质可知，柯西分布具有比高斯分布更加离散的取值，更有利于算法跳出局部最优。因此，AS 策略引入的变异算子为柯西变异算子，并将其与参数 $\overline{v_{k,i}}$ 结合，根据式(17)对陷入局部最优的粒子进行位置更新，跳出局部最优。

$$x_{k,i}^{t+1} = x_{k,i}^t + \overline{v_{k,i}}C(1) \quad (17)$$

定理 7　柯西变异算子的离散性。假设柯西分布函数为 $f(x) = \dfrac{1}{\pi}\left[\dfrac{\gamma}{(x - x_0)^2 + \gamma^2}\right]$，高斯分布函数为 $g(x) = \dfrac{1}{\sqrt{2\pi}\sigma}\exp\left(-\dfrac{(x-\mu)^2}{2\sigma^2}\right)$，即存在 $N > 0$，使当 $|x| > N$ 时，有 $f(x) > g(x)$。

证明　由于 $f(x)$ 与 $g(x)$ 都是以 $x = \mu$ 对称，因此要证明 $f(x) > g(x)$，只需证明当 $x > N$ 时，

$f(x)>g(x)$ 即可。令 $W(x)=\exp\left(\frac{x^2}{2\sigma^2}\right)-\frac{x^2}{\sigma}-\sigma$，显然存在 $N>0$ 时，使 $W(x)>0$，即 $\exp\left(\frac{x^2}{2\sigma^2}\right)>\frac{x^2+\sigma^2}{\sigma}$，即可知 $f(x)>g(x)$，证毕。

③ 设计边界限制参数 η。由于 $C(1)$ 是引入的柯西算子，是由 t=1 的柯西分布函数产生的随机数，并不能得到有效搜索区域，因此，在进行数据集搜索时，对搜索区域进行边界限制，只对满足边界限制参数 η 的数据区域进行搜索。

定理 8 边界限制参数。假设 x_0 为 x_i 的中位数，γ_i^{L} 和 γ_i^{R} 分别表示 x_i 左侧和右侧的尺度参数，则参数 η 为

$$\eta=\frac{\gamma_i}{(x-x_0)^2+\gamma_i^2}$$

对于任意的 x，满足

$$x_0-\sqrt{\frac{\gamma_i^{\mathrm{L}}}{\eta}-(\gamma_i^{\mathrm{L}})^2}\leqslant x_i\leqslant x_0+\sqrt{\frac{\gamma_i^{\mathrm{R}}}{\eta}-(\gamma_i^{\mathrm{R}})^2}$$

证明 由于 x_0 为 x_i 的中位数，即 $(x-x_0)^2$ 的均值表示粒子位置维度的 2 阶中心矩，减少了粒子的离散化程度和噪声的影响。$\frac{\gamma_i}{(x-x_0)^2+\gamma_i^2}$ 可以有效防止参数 η 过大而影响算法 2 的收敛。而由于 x 满足

$$x_0-\sqrt{\frac{\gamma_i^{\mathrm{L}}}{\eta}-(\gamma_i^{\mathrm{L}})^2}\leqslant x_i\leqslant x_0+\sqrt{\frac{\gamma_i^{\mathrm{R}}}{\eta}-(\gamma_i^{\mathrm{R}})^2}$$

即

$$-\sqrt{\frac{\gamma_i^{\mathrm{L}}}{\eta}-(\gamma_i^{\mathrm{L}})^2}\leqslant x_i-x_0\leqslant\sqrt{\frac{\gamma_i^{\mathrm{R}}}{\eta}-(\gamma_i^{\mathrm{R}})^2}\Rightarrow$$

$$(x_i-x_0)^2\leqslant\frac{\gamma_i^{\mathrm{R}}}{\eta}-(\gamma_i^{\mathrm{R}})^2\text{或}(x_i-x_0)^2\leqslant\frac{\gamma_i^{\mathrm{R}}}{\eta}-(\gamma_i^{\mathrm{R}})^2$$

可得

$$\gamma_i^{\mathrm{L}}\leqslant\frac{\gamma_i}{(x-x_0)^2+\gamma_i^2}\leqslant\gamma_i^{\mathrm{R}}$$

因此，$\gamma_i^{\mathrm{L}}\leqslant\eta\leqslant\gamma_i^{\mathrm{R}}$。即在边界 γ_i^{L} 和 γ_i^{R} 内进行搜索，从而可以得到有效搜索区域。证毕。

2) 质心初始化

在通过 AS 策略进行自适应参数选取，保证不会陷入局部最优解之后，便可以进行质心的初始化。其具体过程如下。

① 将每个网格单元的数据看作一群粒子 S_1，$S_2,\cdots,S_m$，并通过式(3)～式(5)对其进行初始化。

② 计算每个粒子的适应值，并将其与 $p_{k,i}^t$ 和 $p_{g,i}^t$ 进行比较，如果适应值较优，则用适应值代替当前的 $p_{k,i}^t$ 和 $p_{g,i}^t$，进行适应值更新。

③ 计算参数 η 的值，获取有效搜索区域，根据更新的适应值，结合式(6)与式(7)在有效搜索区域中对粒子的速度与粒子的位置进行更新。

④ 将每次更新的 $p_{g,i}^t$ 记录在集合 $W\{\}$ 中，即 $W=\{p_{g,i}^1,p_{g,i}^2,\cdots,p_{g,i}^t\}$，并对集合 $W\{\}$ 中的值进行比较，选取前 K 个较大的值，并找到其对应的粒子点，便是数据集的初始质心。

局部聚类的伪代码如算法 2 所示。

算法 2 局部聚类

输入 初始数据集 S，粒子的初始位置 x 和始速度 v，簇数 K，自适应参数

输出 质心数据集

```
1) while(generation ⩽ MaxGeneration)
2)   for each x_{k,i} ∈ S do
3)     f = Adaptive(x_{k,i})
4)     if ( f > pbest )
5)       pbest = f
6)     end if
7)     if ( pbest > gbest )
8)       gbest = pbest
9)     W{} ← gbest
10)  end if
11)  Get_arverage_speed(v)
12)    η = γ_i / ((x − x_0)^2 + γ_i^2)
13)  end for
14)  for each k do
15)    map{} ← W
16)  end for
17) end while
18) return map
```

2.3.3 局部簇并行化计算

在对网格单元的数据质心初始化之后，便需要对网格单元进行并行化合并，获取局部簇，实现局部并行化聚类。因此，本文提出并行化局部聚类策

略 SPPLCS，实现对局部簇的并行化计算来完成整个局部聚类的过程。其基本步骤如下。

1) 将各个网格单元 $G_1,G_2,G_3,\cdots,G_m$ 分配到 Partition。

2) 通过 mapRartitions 算子计算出各个网格单元的中心点 $\bar{x}_i$，其中 mapRartitions 算子为 $F(x_i,m)=\frac{1}{m\sum_{i=1}^{m}x_i}$。

3) 将各个网格单元中的质心点集 $\{p_{g,i}^1, p_{g,i}^2,\cdots,p_{g,i}^k\}$ 与网格中心点 $\overline{x_i}$ 输入 flatMap 算子，找到质心点所对应的网格单元，并标记为 $C_1,C_2,\cdots,C_P$，计算出网格中心点与质心点之间的欧氏距离 D_i。flatMap 算子为 $F(\rho_{g,i}^j,x_k)=\sqrt{\sum_{k=1}^{m}\sum_{j=1}^{m}(x_k-\rho_{g,i}^j)^2}$，输出 D_i 值。

4) 根据输出的 D_i 值，通过 mapToPair 算子选取值最小的网格单元进行合并，即 $\{G_i, C_i \mid \min\{D_i(p_{(g,i)}^k, \bar{x}_i)\}, p_{(g,i)}^k \in C_i, \bar{x}_i \in G_i\}$，重复循环，直到所有网格单元合并完毕，最后进行 reduceByKey 操作汇总，获得局部簇 $C_1',C_2',\cdots,C_K'$。

2.4　局部簇合并

为了解决局部簇并行化合并效率低的问题，本文提出了局部簇合并策略 CRNN，该策略结合 Spark 计算框架将相似度大的簇进行合并，提高了局部簇并行化合并效率。其主要步骤如下。

1) 对于每一个簇 $C_1',C_2',\cdots,C_K$，分别计算出离质心距离最大的点，以其到质心的距离作为簇半径 R_i。获取每一个簇的簇半径后，计算出各个簇之间的邻居节点。

2) 对于簇 C_i,C_j，通过邻居节点集疏密程度判断 2 个簇之间的亲密程度，并分别计算出 2 个簇的样本点数 n_i,n_j，提出簇的相似性函数 $\mathrm{CSM}(n_i,n_j)$，计算出簇与簇之间的相似度。

定理 9　簇的相似性函数。设 ne_i 和 ne_j 分别为 C_i 和 C_j 之间的邻居节点和非邻居节点的个数，n_i 和 n_j 分别为簇 C_i' 和簇 C_j' 的样本点的个数，则簇的相似性函数表示为

$$\mathrm{CSM}(n_i,n_j)=\frac{\mathrm{ne}_i+\mathrm{ne}_j}{n_i+n_j}\sum_{i=0}^{k}\left(\frac{\mathrm{ne}_i}{2k}\right)^2-\frac{\mathrm{ne}_i}{k} \quad (18)$$

证明　对 $\left(\frac{\mathrm{ne}_i}{2k}\right)^2-\frac{\mathrm{ne}_i}{k}$ 展开可得

$$\left(\frac{\mathrm{ne}_i}{2k}\right)^2-\frac{\mathrm{ne}_i}{k}=\frac{\mathrm{ne}_i^2-4k\mathrm{ne}_i}{4k^2}=\frac{\mathrm{ne}_i(\mathrm{ne}_i-4k)}{4k^2}$$

由此可知，当 2 个簇之间的邻居节点较少时，$\frac{\mathrm{ne}_i(\mathrm{ne}_i-4k)}{4k^2}$ 值较小，如果 2 个簇之间无交集，$\frac{\mathrm{ne}_i(\mathrm{ne}_i-4k)}{4k^2}$ 的结果甚至会小于 0。因此，$\sum_{i=0}^{k}\left(\frac{\mathrm{ne}_i}{2k}\right)^2-\frac{\mathrm{ne}_i}{k}$ 可以很好地衡量簇与簇之间的交集，簇的相似性函数 $\mathrm{SM}(n_i,n_j)$ 可以很好地表示簇的相似度。证毕。

3) 将各个簇 $C_1',C_2',\cdots,C_K'$ 分配到多个 Partition，根据步骤 2) 计算出的相似度值，通过算子 mapRarti-tions 对簇之间的相似度进行比较，将相似度最大的 2 个簇进行合并。对于合并完的 2 个簇，对其中的一个簇进行标记 merged，另一个簇则通过 flatMap 算子对其进行簇半径更新，重复循环，直到不存在 2 个非 merged 的簇还有邻居节点。最后进行 reduceByKey 操作汇总，实现局部簇的并行化合并。

局部簇合并的伪代码如算法 3 所示。

算法 3　局部簇并行化合并

输入　局部簇 $C_1',C_2',\cdots,C_K'$

输出　局部簇的并行化合并

1) for each $C_i^{'} \in C^{'}$
2)　while(j<k)
3)　if (max> $\mathrm{dis}(x_i,p_{g,i}^j)$)
4)　max = $\mathrm{dis}(x_i,p_{g,i}^j)$
5)　end if
6)　end while
7)　R_i = max　//获取簇半径
8)　Map{} ← GetNeighborNode(C_i' ,C_j^i,R_i)
9)

$$\mathrm{CSM}(n_i,n_j)=\frac{(\mathrm{ne}_i+\mathrm{ne}_j)\left(\sum_{i=0}^{k}\left(\frac{\mathrm{ne}_i}{2k}\right)^2-\frac{\mathrm{ne}_i}{k}\right)}{n_i+n_j}$$

10)　while(i<k)
11)　if(CSM_i > CSM_ max)
12)　merged(C_i' ,C_j')
13)　if　(isunmerged(C_p',C_q')　and

NeighborNode == Null)

14) break

15) end if

16) end if

17) end while

18) end for

19) reduceByKey

20) return result

2.5 算法时间复杂度分析

PDC-SFASPSO 算法的时间复杂度主要由网格单元的获取、局部簇的形成以及局部簇的并行化合并三部分构成，时间复杂度分别如下。

1) 网格单元的获取阶段的时间复杂度主要取决于数据集的粗略划分、网格划分、网格单元的离群点过滤。设数据集的样本点数为 n，维度数为 m，网格单元个数为 p， Partition 数为 r，则网格单元的获取阶段的时间复杂度为

$$T_1 = O\left(\frac{2n}{m} + \left(\frac{n}{m}\right)^2 + 2n + pr\right) \tag{19}$$

2) 局部簇的形成阶段的时间复杂度主要取决于簇数的获取、质心初始化、局部并行化聚类。设有 q 个局部区域覆盖， g 个全局最优值， w 个有效搜索区域，即粒子的自适应平均速度只需迭代 w 次，对于参数 η 便只需更新 $\lg w$ 次，令最大迭代次数为 Iter，则局部簇的形成阶段的时间复杂度为

$$T_2 = O\left(n\lg n + \sum_{i=1}^{q} ni + nw + \lg w + g + g^2 + \sum_{i=1}^{\text{Iter}} \frac{nr}{p}\right) \tag{20}$$

3) 局部簇的并行化合并阶段的时间复杂度主要取决于计算簇半径与邻居节点，通过相似度进行簇的并行化合并。假设簇数为 k，分布式节点数为 d，则其时间复杂度为

$$T_3 = O(2n\lg n + d\lg k) \tag{21}$$

综上可知，PDC-SFASPSO 算法的时间复杂度为

$$T_{\text{SP-ASPSO}} = T_1 + T_2 + T_3 = O\left(\frac{2n}{m} + \left(\frac{n}{m}\right)^2 + 2n + pr + \sum_{i=1}^{q} ni + nw + \lg w + g + g^2 + \sum_{i=1}^{\text{Iter}} \frac{nr}{p} + 3n\lg n + d\lg k\right) \tag{22}$$

其中， $w \ll n$，因此 $nw \ll n^2$。由于 $g \ll n$， $r \ll n$， $d \ll n$， $k \ll n$，因此最终的时间复杂度近似为

$$T_{\text{SP-ASPSO}} = O\left(\left(\frac{n}{m}\right)^2 + \sum_{i=1}^{q} ni + \sum_{i=1}^{\text{Iter}} \frac{nr}{p}\right) \tag{23}$$

SP-DAP 算法对划分后的数据进行并行化聚类阶段的时间复杂度为 $O\left(\sum_{i=1}^{\text{Iter}'} \frac{nr}{p} + \sum_{i=1}^{p} ni\right)$。由于该算法并没有计算出局部聚类簇数，导致没有较优的初始化质心，因此并行化聚类的迭代次数要远大于 PDC-SFASPSO 算法。因此， $O\left(\sum_{i=1}^{\text{Iter}'} \frac{nr}{p} + \sum_{i=1}^{p} ni\right) \gg O\left(\sum_{i=1}^{q} ni + \sum_{i=1}^{\text{Iter}} \frac{nr}{p}\right)$，而 $O(n^2 \lg n) \gg O\left(\left(\frac{n}{m}\right)^2\right)$，因此，PDC-SFASPSO 算法的时间复杂度要低于 SP-DAP 算法。

对于 SP-GAKMS 算法，由于该算法在使用群智能算法初始化质心时，会导致陷入局部最优，并不能获取较优的初始质心。因此，其算法的迭代次数也远大于 PDC-SFASPSO 算法，导致并行化聚类阶段的时间复杂度 $O\left(\sum_{i=1}^{q} ni + \sum_{i=1}^{\text{Iter}} \frac{nr}{p}\right)$。所以，PDC-SFASPSO 算法的时间复杂度低于 SP-GAKMS 算法。

对于 SP-LCMNS 算法，设对边的迭代次数为 e，则该算法的数据划分的时间复杂度为 $O(n^2 e)$，在对数据集进行并行化聚类时，仅仅使用了二次划分算法进行优化，并未在并行化聚类之前对数据集进行簇的生成，导致了并行化聚类的迭代次数增多。因此该阶段的时间复杂度为

$$O\left(\sum_{i=1}^{\text{Iter}''} \frac{nr}{p} + \sum_{i=1}^{u} ni + n\lg n\right)$$

其中， $\text{Iter}'' \gg \text{Iter}$， $u \gg q$，故 PDC-SFASPSO 算法的时间复杂度要低于 SP-GAKMS 算法。

综上可知，相较于 SP-DAP 算法、SP-GAKMS 算法与 SP-LAICA 算法，PDC-SFASPSO 算法有更理想的时间复杂度。

3 实验与分析

3.1 实验环境

为了验证 PDC-SFASPSO 算法的性能，本文设计了相关实验。在硬件方面，实验包括 4 个计算节点，即一个 master 节点和 3 个 worker 节点，并修改主机名分别为 master、salve_1、salve_2、salve_3。

所有节点的 CPU 均为 Inter Core i5 ，2.50 GHz 四核 CPU，16 GB 内存，512 GB 存储磁盘。实验环境中的 4 个节点处于同一个局域网中，所有的计算节点通过 1 Gbit/s 以太网相连。在软件方面，使用 CentOS6.0 系统安装 Hadoop 2.7.7，从而配置 Hadoop 集群，对每个节点都安装 JDK1.8.0 并完成节点的集群配置。安装 Hive1.1.2 与 Spark2.2.0，使用 Hive 将 HDFS 中的数据进行分析和管理。在 Spark 并行计算框架的基础上，使用 Scala 编程语言对数据进行处理，并将处理好的数据导入集群。最后使用 Superset 对集群中处理好的数据进行可视化处理。各个节点的具体配置如表 1 所示。

表 1　　实验中节点的配置

节点类型	主机名	IP 地址
master	master	192.168.111.1
worker	slave_1	192.168.111.2
worker	slave_2	192.168.111.3
worker	slave_3	192.168.111.4

3.2　实验数据

本节实验采用 4 个真实的数据集，分别是 Online Retail、N_BaloT、Health News 及 Bag words[20]。其中，Online Retail 数据集包含非商店在线零售的所有交易；N_BaloT 是从 9 台商业物联网设备收集的真实数据；Health News 包含来自超过 15 个主要健康新闻机构的健康新闻；Bag words 包含 5 个文本集合。这些数据集的详细信息如表 2 所示。

3.3　评价指标

3.3.1　加速比

加速比是通过并行计算来降低总体运行时间而获得的性能提升的数值化表示形式，其定义为

$$S_p = \frac{T_1}{T_p} \tag{24}$$

其中，T_1 表示算法在单节点上的运行时间；T_p 表示并行计算的运行时间；S_p 越大，表示算法的并行化效率越高。

3.3.2　NMI

标准互信息（NMI, normalized mutual information）通过对信息进行量化度量，并根据 2 个概率分布的信息熵的差值进行聚类的效果评估，其定义为

$$\mathrm{NMI} = \frac{I(\boldsymbol{X},\boldsymbol{Y})}{\sqrt{H(\boldsymbol{X})H(\boldsymbol{Y})}} \tag{25}$$

其中，$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 为 N 维向量，$I(\boldsymbol{X},\boldsymbol{Y})$ 表示 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 之间的互信息，$H(\boldsymbol{X})$ 和 $H(\boldsymbol{Y})$ 分别表示 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 的熵；NMI 值越大，聚类效果越好。

3.3.3　ARI

调整兰德指数（ARI, adjusted Rand index）是通过将模型的超分布假设为随机模型从而用于聚类模型的性能评估，该评价指标具有更高的区分度，其定义为

$$\mathrm{RI} = \frac{a+b}{\mathrm{C}_2^{n_{\mathrm{samples}}}} \tag{26}$$

$$\mathrm{ARI} = \frac{\mathrm{RI} - E|\mathrm{RI}|}{\max(\mathrm{RI}) - E|\mathrm{RI}|} \tag{27}$$

其中，$\mathrm{C}_2^{n_{\mathrm{samples}}}$ 表示样本中的组合；RI 表示计算正确的比例，取值范围为[0,1]，其值越大代表聚类效果越好。

3.4　PSO 算法的优越性分析

文献[21]中 MSPSO 算法通过保留粒子历史最优位置，重新初始化其他粒子，提高了粒子多样性，扩大了搜索空间，从而获取全局最优解。因此，为验证 ASPSO 算法的优越性，本文选取 MSPSO 算法、PSO 算法、ASPSO 算法在 4 个不同基准函数（Sphere、Schwefel、Ackely、Griewank）进行仿真实验，函数的具体信息如表 3 所示。算法参数设置如下：采用线性下降惯性权重，并且 w 在[0.35,0.95] 内取值随迭代数增加而线性递减。学习因子 c_1 和 c_2 均为 1.5，种群规模为 40，函数维度为 30，收敛阈值 ε 设置为 10^{-5}，边界值 $v_{\max}$ 为 0.5Range，并设置 ASPSO 算法的尺度参数为 $[-100,100]$。为对算法进行统计算分析，将每个函数独立运行 50 次，其结果如表 4 所示。

表 2　　实验数据集

数据集	样本数/个	特征属性	文件大小/MB	数据特点
Online Retail	1 067 371	8	580	样本多，属性少
N_BaloT	7 062 606	115	960.5	样本多，属性相对适中
Health News	580 000	25 000	830.2	样本少，属性多
Bag words	8 000 000	1 000 000	2 687.9	样本多，属性多

表 3 **基准测试函数**

类型	函数名称	函数表达式	取值范围	最优值
单峰	Sphere	$f_1(x)=\sum_{i=1}^{d} x_i^2$	[−10,10]	0
	Schwefel	$f_2(x)=\sum_{i=1}^{d}\lvert x_i\rvert+\prod_{i=1}^{d}\lvert x_i\rvert$	[−100,100]	0
高维多峰	Ackely	$f_3(x)=20+\mathrm{e}-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right)$	[−32,32]	0
	Griewank	$f_4(x)=1+\sum_{i=1}^{d}\frac{x_i^2}{4\,000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	[−600,600]	0

表 4 **ASPSO 算法与 PSO 算法的性能对比**

函数	算法	中间值	平均值	标准差
f_1	PSO	6.487×10^{-14}	2.189×10^{-13}	1.584×10^{-14}
	MSPSO	8.432×10^{-16}	1.345×10^{-13}	5.365×10^{-18}
	ASPSO	3.458×10^{-16}	5.975×10^{-15}	2.858×10^{-26}
f_2	PSO	8.469×10^{-17}	7.486×10^{-22}	5.753×10^{-8}
	MSPSO	2.368×10^{-17}	9.325×10^{-20}	6.225×10^{-12}
	ASPSO	4.457×10^{-18}	6.733×10^{-24}	2.946×10^{-22}
f_3	PSO	6.445×10^{-12}	8.554×10^{-8}	6.332
	MSPSO	5.398×10^{-13}	4.443×10^{-11}	9.352×10^{-4}
	ASPSO	2.331×10^{-14}	9.328×10^{-12}	2.977×10^{-10}
f_4	PSO	4.474×10^{-16}	5.254×10^{-12}	3.289
	MSPSO	9.998×10^{-18}	8.887×10^{-16}	5.877×10^{-0}
	ASPSO	3.649×10^{-22}	3.977×10^{-18}	4.618×10^{-8}

从表 4 可以看出，ASPSO 算法的平均值与标准差始终低于 PSO 算法与 MSPSO 算法，ASPSO 算法在收敛精度和应用在不同类型函数的稳定性方面明显优于 PSO 算法与 MSPSO 算法。尤其是在高维多峰函数 f_3 与 f_4 上，PSO 算法的标准差分别为 6.332 与 3.289，很显然，未优化的 PSO 算法所在的搜索区域有多个极值点，且快速收敛陷入了局部最优解，相较于标准差仍保持理论较优的 ASPSO 算法，其全局搜索能力与稳定性明显远低于 ASPSO 算法。这是因为 ASPSO 算法采用了 AS 策略，设计了控制变异步长参数与边界限制参数，对粒子位置进行更新，

帮助粒子跳出局部最优，获取全局最优解。从函数 f_1 与 f_2 可以看出，ASPSO 算法的标准差接近理论最优值 0，具有较高的收敛精度，极大地体现了 ASPSO 算法较强的跳出局部极值能力和较好的全局搜索能力。实验表明，在全局寻优能力和算法稳定程度方面，ASPSO 算法明显比传统 PSO 算法具有更好的优越性。

3.5 聚类过程性分析

由于 Bag words 数据集具有样本数量多、属性多的特点。因此，选取 Bag words 数据集对算法进行聚类过程性分析。算法首先采用 PCCV 策略对数据集进行划分，获取网格单元。由于数据集包含多维特征属性，聚类效果不易观测，因此将数据网格单元投影到二维空间进行演示。其中图 1 展示了部分网格单元数据点的分布。然后，通过 ASPSO 算法获取各个数据网格单元的质心点集，通过 SPPLCS 策略将质心点与其他网格单元中心距离最近的 2 个数据网格单元进行并行化合并，获取局部簇，图 2 展示了部分局部簇数据点的分布。最后，采用 CRNN 策略计算出各个簇之间的邻居节点数，

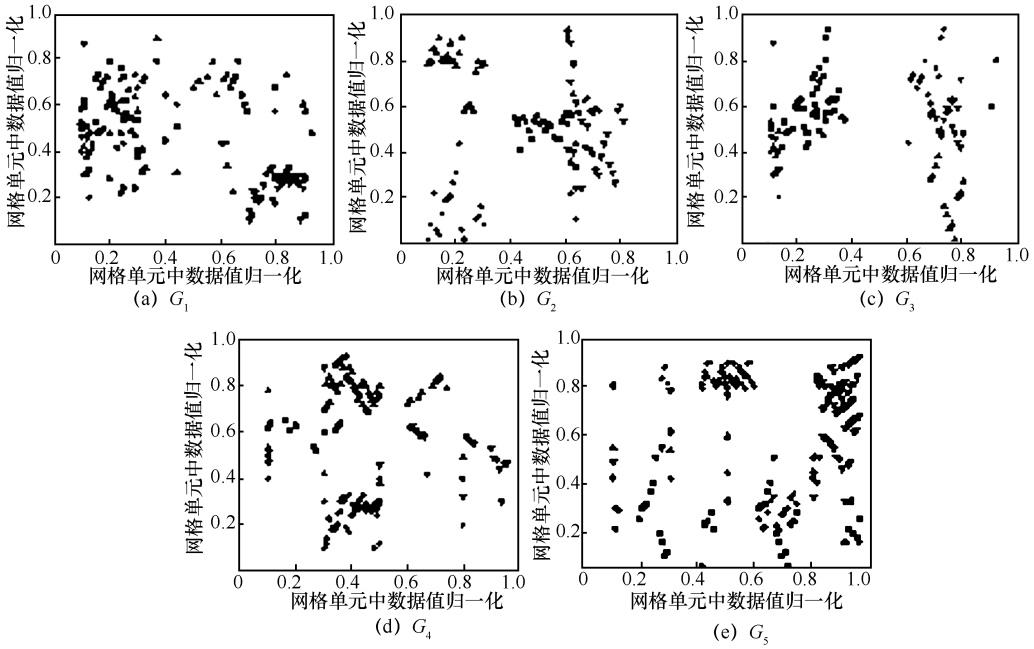


图 1　网格单元数据集分布

依据邻居节点数，并结合 CSM(n_i,n_j) 函数对簇的相似性进行判断，将相似度大的局部簇进行合并，从而完成并行化聚类。图 3 显示了第一轮 2 个簇合并的数据点。

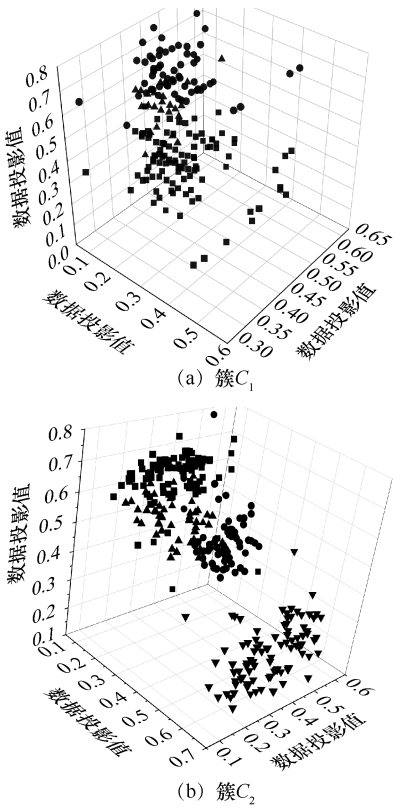


图 2　局部数据点的分布

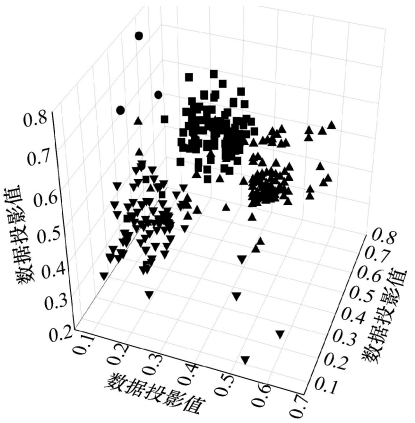


图 3　第一轮 2 个簇合并的数据点

3.6　算法可行性分析与并行化效率比较

为了验证 PDC-SFASPSO 算法的可行性并比较各个算法之间的并行化效率，使用算法的加速比来进行衡量。并对每一个数据集都运行 20 次，取其均值作为实验结果，如图 4 所示。

从图 4 可以看出，PDC-SFASPSO 在节点数达到 4 时，加速比最大，并且在各个数据集上 PDC-SFASPSO 的加速比随着节点数的增多而不断增大，尤其是在 Online Retail 数据集上，PDC-SFASPSO 的加速比的增长趋势更加明显，很好地体现了 PDC-SFASPSO 的并行化可行性。其中，当节点数达到 4 时，在 Online Retail 数据集上，PDC-SFASPSO 的加速比相较于 SP-DAP 算法、SP-GAKMS 算法和

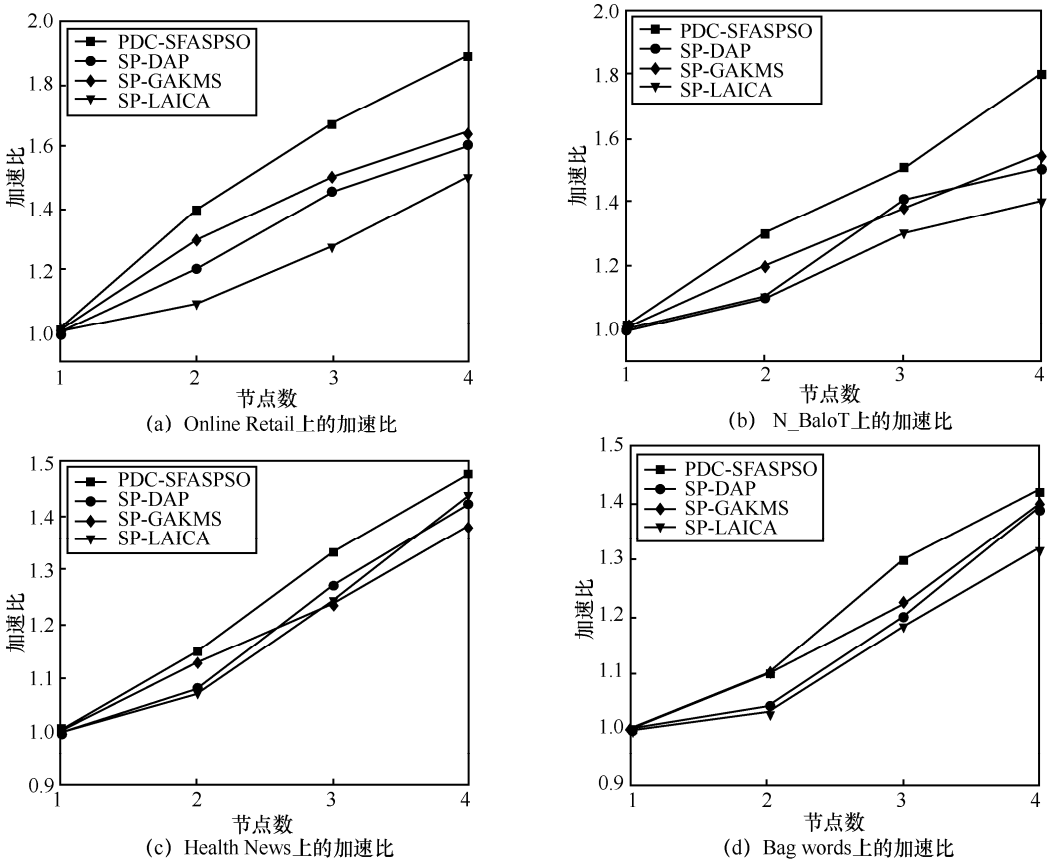


图 4 各算法处理各数据集的加速比

SP-LAICA 算法分别增加了 0.3、0.28、0.4；在 N_BaloT 数据集上，PDC-SFASPSO 的加速比相较于 SP-DAP 算法、SP-GAKMS 算法和 SP-LAICA 算法分别增加了 0.3、0.26、0.4；在 Bag words 数据集上，PDC-SFASPSO 的加速比相较于 SP-DAP 算法、SP-GAKMS 算法和 SP-LAICA 算法分别增加了 0.03、0.02、0.1。产生这些结果的原因如下：一方面，PDC-SFASPSO 采用了 PCCV 策略，解决了数据间离散系数较大与算法抗干扰性差的问题，间接地提高了聚类的并行化效率；另一方面，算法在并行化合并阶段设计了相似性函数 $\mathrm{CSM}(n_i,n_j)$ 进行相似度判断，极大地提高了局部簇并行化合并的效率。因此，在 4 种数据集中，PDC-SFASPSO 相较于其他 3 种算法具有最佳的并行化效率且具有可行性。

3.7 聚类效果与算法性能比较分析

在目前基于 Spark 计算框架的并行聚类算法中，SP-DAP 算法[7]、SP-GAKMS[10]算法和 SP-LAICA 算法[15]在对数据集进行并行化聚类时，分别在数据集划分阶段、局部聚类阶段与局部簇并行化合并阶段进行了算法的优化，解决了数据离散系数较大、局部簇质心随机性以及局部簇并行化合并效率低等问题，从而极大地提高了算法的聚类效果，相较于大多数基于 Spark 的并行聚类算法有着更优的聚类效果、并行效率和时间效率等性能，因此本文选择这 3 种算法与提出的 PDC-SFASPSO 算法进行聚类效果、并行效率与时间效率等比较与分析。

3.7.1 聚类效果比较分析

1) NMI 值比较

使用 NMI 作为衡量指标评价算法的聚类效果，分别比较各个算法在不同数据集上的 NMI 值，从而对各个算法的聚类效果进行分析。因此，为了验证 PDC-SFASPSO 算法的聚类效果，将 4 种算法在上述 4 种数据集上进行比较，通过比较 NMI 值来体现算法的聚类效果，并分别运行 10 次得出聚类结果，取其聚类结果均值并结合一定的误差范围作为实验结果。实验结果如表 5 所示。

表 5 中的数据表明，在聚类效果方面，各算法的 NMI 值随着数据集的特征属性增多而不断减小，PDC-SFASPSO 算法在 4 种数据集上始终表现得最好。其中，在 Online Retail 数据集中，PDC-SFASPSO 算法

表 5　　各算法处理数据集的 NMI 值

算法	Online Retail	N_BaloT	Health News	Bag words
PDC-SFASPSO	0.895（± 0.027）	0.886（± 0.0372）	0.836（± 0.0563）	0.721（± 0.0775）
SP-DAP	0.766（± 0.0372）	0.721（± 0.0197）	0.691（± 0.0754）	0.591（± 0.0674）
SP-GAKMS	0.751（± 0.0196）	0.711（± 0.0542）	0.621（± 0.0621）	0.521（± 0.0788）
SP-LAICA	0.769（± 0.0126）	0.703（± 0.0278）	0.646（± 0.0487）	0.546（± 0.0597）

的 NMI 值相比于 SP-DAP、SP-GAKMS 和 SP-LAICA 算法分别高出了 3.2%、4.9%和 2.9%；在 N_BaloT 数据集上，分别高出了 7.3%、8.4%和 9.3%。产生这种结果的主要原因是 PDC-SFASPSO 采用 PCCV 策略解决了数据离散系数较大与算法抗干扰性差的问题，并且在对局部簇质心进行初始化时，提出了策略 ASPSO，获取了全局最优初始质心，解决了局部簇质心随机性的问题，极大地提高了聚类效果。

2) ARI 值比较

为了进一步验证 PDC-SFASPSO 算法的聚类效果，使用 ARI 作为衡量指标评价算法的聚类效果，将 4 种算法分别在上述 4 种数据集上进行处理，并分别运行 10 次得出聚类结果，取其聚类结果均值作为实验结果，如图 5 所示。

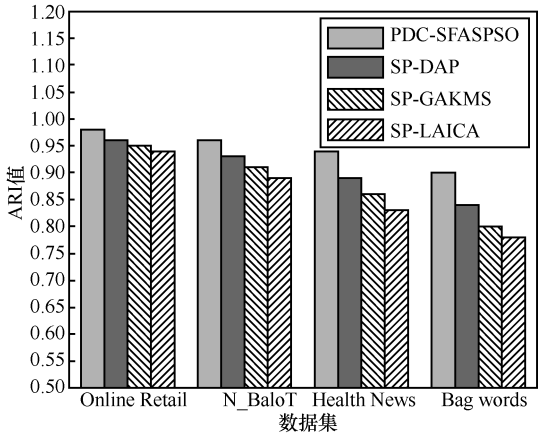


图 5　各算法处理不同数据集的 ARI 值

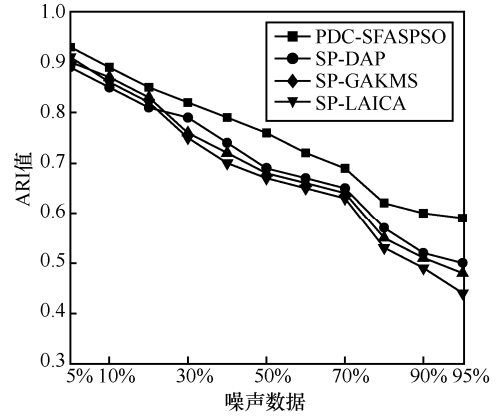
从图 5 可以看出，在对各数据集进行处理时，PDC-SFASPSO 算法的 ARI 值始终保持最高，并且随着数据集的特征属性增多，PDC-SFASPSO 的 ARI 值与其他 3 种算法的 ARI 值相比较，PDC-SFASPSO 算法的优势更加明显。尤其是在 Bag words 数据集上，PDC-SFASPSO 凭借着设计了 PFGF 策略，其 ARI 值远高于 SP-LAICA。在 Online Retail 数据集上 PDC-SFASPSO 算法的 ARI 值相比于 SP-DAP 算法、SP-GAKMS 算法和 SP-LAICA 算法分别高 0.02、0.03 和 0.04，在 Bag words 数据集上分别高 0.06、0.10 和 0.12。产生这些结果的主要原因是 PDC-SFASPSO 算法设计了 ASPSO 策略获取局部簇质心，解决了局部簇质心的随机性问题。并且 PDC-SFASPSO 算法在初始化质心之前，设计了 PFGF 策略来获取局部聚类的簇数，解决了算法局部簇簇数难以确定的问题，从而提高了算法的聚类效果。因此，通过对比算法在 4 种数据集上的 ARI 值可以看出，PDC-SFASPSO 算法具有最佳的聚类效果。

3.7.2　性能比较分析

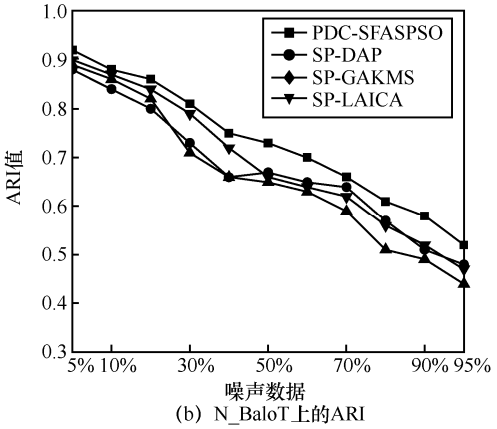
1) 抗干扰性比较

为了验证 PDC-SFASPSO 与其他 3 种算法在大数据集下并行化聚类的抗干扰性，通过利用 Sklearn 中的函数生成一种具有缺失值的噪声数据集，将其分成多份，分别依次加入上述 4 种数据集中。将 ARI 值作为评价指标，对加入含有缺失值噪声数据的算法的聚类效果进行比较。通过比较算法的聚类效果来对算法的可抗干扰性进行分析，算法聚类效果越好，则算法受噪声数据的影响越小，即算法的可抗干扰性越好[14]。实验结果如图 6 所示。

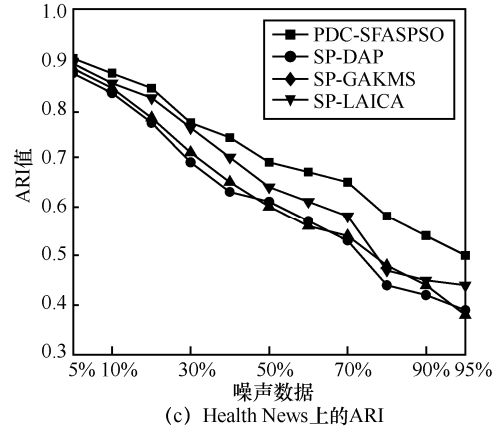
从图 6 可以看出，随着噪声数据的不断增加，各算法的 ARI 值逐渐减小，PDC-SFASPSO 在 4 种数据集中的 ARI 值始终保持最高，具有更好的聚类效果，并且噪声数据越多，PDC-SFASPSO 算法的抗干扰能力相较于其他 3 种算法优势越加明显。其中，当噪声数据达到 50%时，在 Online Retail 数据集上，PDC-SFASPSO 的 ARI 值相较于 SP-DAP、SP-GAKMS 和 SP-LAICA 算法分别高 21%、27%、33.3%；在 Bag words 数据集上，PDC-SFASPSO 的 ARI 值分别高 32.3%、49.1%、40.9%。产生这些结果的主要原因是由于最开始没有加入噪声数据，ARI 值达到最大，当加入噪声数据后，算法的 ARI 值会因为噪声数据的突然加入，呈现快速下降的趋势，因此在噪声数据为 20%～40%时，算法的 ARI 值下降最快。而 PDC-SFASPSO 算法由于设计了 PCCV 策略，提高了算法的聚类效果，并通过设计离群因子 GOF 来对



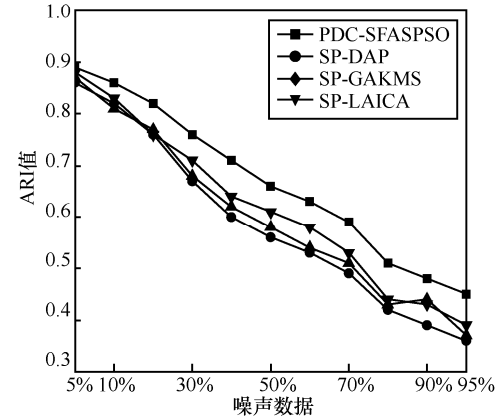
(a) Online Retail上的ARI



(b) N_BaloT上的ARI



(c) Health News上的ARI



(d) Bag words上的ARI

图 6 算法处理各数据集的 ARI

离群点过滤，减小了噪声数据对算法聚类效果的影响，极大地增强了算法的抗干扰能力。同时，通过对比图 6(b)～图 6(d)可以看出，随着噪声数据的不断增加，PDC-SFASPSO 算法的 ARI 值一直保持最优，这也证明了 PDC-SFASPSO 算法具有最佳的抗干扰性。

2) 算法运行时间比较

将 PDC-SFASPSO、SP-DAP、SP-GAKMS 与 SP-LAICA 算法分别在 Online Retail、N_BaloT、Health News 及 Bag words 数据集上进行对比实验，分析各个算法的运行时间，实验结果如图 7 所示。

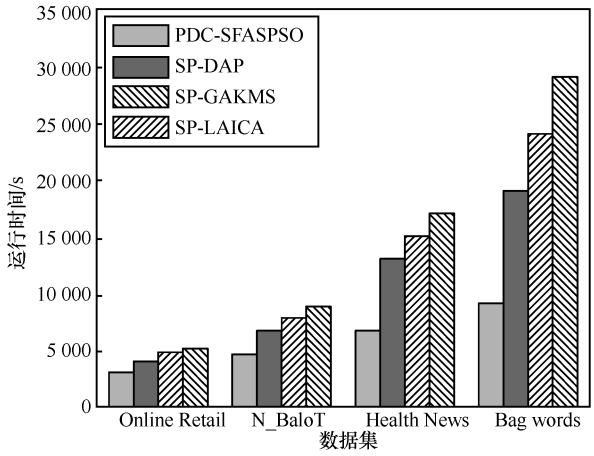


图 7 算法在各个数据集上的运行时间

从图 7 可以看出，随着数据集的属性与数据量的增多，算法运行的时间开销也逐渐增大。并且 PDC-SFASPSO 算法在 4 种数据集上的运行时间始终最低。其中，在 Health News 数据集上，相较于 Online Retail 数据集算法的时间开销产生了明显的增长，分别增加了 38.2%、73.5%、79.6%、85.9%，然而对于 PDC-SFASPSO 算法，其时间消耗增加的始终较为平缓。产生此结果的主要原因是 PDC-SFASPSO 算法使用了网格划分策略 PCCV 解决了数据间离散系数较大的问题与算法抗干扰性差的问题，从而提高了算法的聚类效果，间接地提高了并行化合并效率；此外，在并行化阶段，PDC-SFASPSO 算法设计了 CRNN 策略，极大地减小了聚簇时间消耗，解决了局部簇并行化合并效率低的问题。因此，PDC-SFASPSO 算法在各个数据集上相对于其他 3 种算法时间消耗始终处于最低。

4 结束语

针对传统的划分聚类算法在大数据环境下的不足，本文提出了一种基于 Spark 框架和 ASPSO 策略

下的并行划分聚类算法 PDC-SFASPSO。该算法在数据划分阶段提出了网格划分策略，保证了划分后的数据网格单元数据离散系数较小，并设计了离群因子过滤网格单元数据，增强了数据的抗干扰性；使用基于 PFGF 的搜索策略对数据进行核覆盖搜索获取局部簇簇数，并提出 ASPSO 优化算法，通过自适应参数避免算法陷入局部最优解从而获取全局最优质心；提出了局部簇并行化合并策略 CRNN，通过避免对所有簇的点与边集同时展开搜索而提高算法局部簇的并行化合并效率。为了验证 PDC-SFASPSO 算法的性能，本文设计了相关实验，在 Online Retail、N_BaloT、Health News 和 Bag words 数据集上将 PDC- SFASPSO 算法分别与 SP-DAP、SP-GAKMS 和 SP-LAICA 算法进行比较。实验结果显示，与其他算法相比，PDC-SFASPSO 算法在处理大数据时具有相对较好的性能表现。但是，该算法仍有一些需要改进及完善的地方，如数据的特征约简与负载平衡等问题，这些都是下一步重点研究的问题。

参考文献：

[1] WANG P K, CHEN C H, PUN S H, et al. Parallel architecture to accelerate super paramagnetic clustering algorithm[J]. Electronics Letters, 2020, 56(14): 701-704.

[2] KHAN A, ZUBAIR S. Expansion of regularized kmeans discretization machine learning approach in prognosis of dementia progression[C]//Proceedings of 2020 11th International Conference on Computing, Communication and Networking Technologies. Piscataway: IEEE Press, 2020: 1-6.

[3] MARTANTO, ANWAR S, ROHMAT C L, et al. Clustering of Internet network usage using the K-medoid method[J]. IOP Conference Series: Materials Science and Engineering, 2021, 1088(1): 012036.

[4] SCHUBERT E, ROUSSEEUW P J. Fast and eager k-medoids clustering: *O*(*k*) runtime improvement of the PAM, CLARA, and CLARANS algorithms[J]. Information Systems, 2021, 101: 101804.

[5] LEKHWAR S, YADAV S, SINGH A. Big data analytics in retail[R]. 2019.

[6] WEISSMAN B, VAN D L E. Working with spark in big data clusters[R]. 2020.

[7] MUGDHA S, CHIRAG P, AKASH A. Design and implementation of university network[J]. International Journal of Recent Technology and Engineering, 2019, 8(26): 1199-1214.

[8] 王海艳, 肖亦康. 基于密度峰值聚类的动态群组发现方法[J]. 计算机研究与发展, 2018, 55(2): 391-399.
WANG H Y, XIAO Y K. Dynamic group discovery based on density peaks clustering[J]. Journal of Computer Research and Development, 2018, 55(2): 391-399.

[9] WANG B W, YIN J, HUA Q, et al. Parallelizing K-means-based clustering on spark[C]//Proceedings of 2016 International Conference on Advanced Cloud and Big Data. Piscataway: IEEE Press, 2016: 31-36.

[10] 徐鹏程, 王诚. K-means 算法改进及基于 Spark 计算模型的实现[J]. 南京邮电大学学报(自然科学版), 2017, 37(4): 113-118.
XU P C, WANG C. Improvement of K-means algorithm and implementation based on Spark computing model[J]. Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition), 2017, 37(4): 113-118.

[11] MULTAZAM M T, DIJAYA R, DEVI N S. Index group optimization based on automatic clustering using K-means genetic algorithm[J]. Journal of Physics: Conference Series, 2019, 1402(6): 066028.

[12] 许明杰, 蔚承建, 沈航. 基于 Spark 的并行 K-means 算法研究[J]. 微电子学与计算机, 2018, 35(5): 95-99.
XU M J, WEI C J, SHEN H. Research on K-means algorithm of Spark parallelization[J]. Microelectronics & Computer, 2018, 35(5): 95-99.

[13] GAO H J, LI Y T, KABALYANTS P, et al. A novel hybrid PSO-K-means clustering algorithm using Gaussian estimation of distribution method and Lévy flight[J]. IEEE Access, 2020, 8: 122848-122863.

[14] AGRAWAL S, PATEL A. SAG cluster: an unsupervised graph clustering based on collaborative similarity for community detection in complex networks[J]. Physica A: Statistical Mechanics and Its Applications, 2021, 563: 125459.

[15] LAI M J, MCKENZIE D. Compressive sensing for cut improvement and local clustering[J]. SIAM Journal on Mathematics of Data Science, 2020, 2(2): 368-395.

[16] 裴继红, 谢维信. 势函数聚类自适应多阈值图像分割[J]. 计算机学报, 1999, 22(7): 758-762.
PEI J H, XIE W X. Adaptive multi thresholds image segmentation based on potential function clustering[J]. Chinese Journal of Computers, 1999, 22(7): 758-762.

[17] ZHANG Y L, HAN J. Differential privacy fuzzy C-means clustering algorithm based on Gaussian kernel function[J]. PLoS One, 2021, 16(3): e0248737.

[18] 赵姝, 许显胜, 华波, 等. 收缩邻居节点集方法求解有向网络的最大流问题[J]. 模式识别与人工智能, 2013, 26(5): 425-431.
ZHAO S, XU X S, HUA B, et al. Contracting neighbor-node-set approach for solving maximum flow problem in directed network[J]. Pattern Recognition and Artificial Intelligence, 2013, 26(5): 425-431.

[19] PAULCHAMY B, CHIDAMBARAM S, JAYA J. An energy efficient neighbor node based clustering (EENNC) algorithm for wireless sensor networks[J]. Journal of Xidian University, 2020, 14(6):2483-2493.

[20] SUN C, YUE S H, LI Q. Clustering characteristics of UCI dataset[C]//Proceedings of 2020 39th Chinese Control Conference (CCC). Piscataway: IEEE Press, 2020, 13(5): 428-439.

[21] DASH D R, DASH P K, BISOI R. Short term solar power forecasting using hybrid minimum variance expanded RVFLN and sine-cosine Levy flight PSO algorithm[J]. Renewable Energy, 2021, 174: 513-537.

[作者简介]

毛伊敏（1970− ），女，新疆伊犁人，博士，江西理工大学教授、博士生导师，主要研究方向为数据挖掘、大数据安全与隐私保护。

甘德瑾（1997− ），男，江西抚州人，江西理工大学硕士生，主要研究方向为数据挖掘、大数据。

廖列法（1975− ）,男，江西玉山人，博士，江西理工大学教授、硕士生导师，主要研究方向为人工智能等。

陈志刚（1964− ），男，湖南益阳人，博士，中南大学教授、博士生导师，主要研究方向为网络与分布式计算、机会网络。