PODKNN:面向大数据集的并行离群点检测算法

苟 杰 马自堂 张喆程
(解放军信息工程大学密码工程学院 郑州 450000)

摘 要 针对现有离群点检测算法在运用于大规模数据集时时间效率较低的问题,提出一种基于K近邻的并行离群点检测算法PODKNN(Parallel Outlier Detection Based on K-nearest Neighborhood)。该算法利用划分策略对数据集进行预处理,在规模较小的子集中寻找K近邻并计算离群度,最后合并结果并遴选出离群点,设计算法过程使其符合MapReduce的编程模型,实现并行化,从而提高了离群点检测算法处理大规模数据的计算效率。实验结果表明,PODKNN具有较高的加速比及较好的扩展性。

关键词 数据挖掘,离群点检测,K近邻,MapReduce
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.7.045

PODKNN:A Parallel Outlier Detection Algorithm for Large Dataset

GOU Jie MA Zi-tang ZHANG Zhe-cheng
(Password Engineering Institute,PLA Information Engineering University,Zhengzhou 450000,China)

Abstract In order to improve the outlier detection algorithm's efficiency of dealing with large-scale data set,a parallel outlier detection based on K-nearest neighborhood was put forward. This algorithm can find the K-nearest neighborhood and calculate the degrees of outliers by using partitioning strategy for pretreatment of data sets,and then it merges the results and selects outliers. The algorithm is designed to suit for the MapReduce programming model to implement parallelization and improve the computational efficiency of dealing with large-scale data sets. The experimental results show that the PODKNN has the advantages of high speedup and good scalability.

Keywords Data mining,Outlier detection,K-nearest neighborhood,MapReduce

1 引言

在数据集中通常存在这样一类数据元素,它们可能是因为数据采集时人为失误或者测量设备故障而产生的异常值,也可能是数据变量发生固有的特殊变化而产生偏离数据变化趋势的一些特殊值,这类数据元素就是离群点[1]。对于这些离群点,前者的存在是无意义的,通常需要剔除,以避免在数据分析阶段产生不可预知的影响;而后者的离群在一些情况下是有趣的,它能够反映数据集的分布特征,例如气候的剧烈变化、新型客户的购买习惯、基因的突发变异等[2,3]。

目前,研究人员已经提出了一系列发现离群点的方法,主要包括基于分布、深度、距离、密度和聚类等不同思想的算法[4-8]。基于距离的离群点检测方法具有算法过程简单、易于理解和实现的特点,受到研究人员的广泛使用。文献[9]提出了一种通过比较每一对象与其他对象的分化距离来计算其友邻点密度,从而挖掘出数据集中隐含的离群点,还可以反映离群点的孤立程度;文献[10]在粗糙集中结合边界和距离的方法进行孤立点检测,提出了一种新的解决思路;文献[11]运用了属性约简技术,将离群点的搜索缩小到较小的具有代表性的属性子空间中,降低了属性空间的搜索复杂度。但是,由于基于距离的检测思想通常需要在数据集的属性维上建立相异度矩阵,时间复杂度通常为$O(n^2)$,当处理大规模数据集时,算法的执行效率不佳,传统的串行计算模式已经不能满足数据的爆发式增长带来的计算需求[12]。

Google公司于2004年提出的MapReduce计算模型可以较为容易地实现算法的并行化[13],基于MapReduce模型的Hadoop平台可以在廉价的集群上部署实验环境,这为实现并行离群点检测算法提供了基础[14]。文献[15]提出了利用MapReduce计算模型实现并行离群点检测算法,通过迭代计算来更新全局候选离群点,找出离群度最大的数据点,但是该算法需要启动多次MapReduce过程,耗时较多。

本文提出了一种可以部署在Hadoop平台上的并行离群点检测算法PODKNN,只需要一次MapReduce过程就可以找到离群度最大的几个数据点。首先通过对大规模数据集进行无损划分的预处理过程,将数据集分割为互有重叠的子集,子集在Map节点寻找K近邻[16]并计算离群度,然后通过Reduce节点的汇总规约,得到离群度最大的一些数据点作为离群点。通过实验分析了参数对算法精度的影响,并在集群环境中验证了算法的加速比和扩展性。实验表明,PODKNN在集群环境中具有良好的性能,能够有效处理大规模数据集。

到稿日期:2015-05-25 返修日期:2015-08-16

苟 杰(1990-),男,硕士,主要研究方向为大数据、数据挖掘,E-mail:xdgj0000000@163.com;马自堂(1962-),男,教授,主要研究方向为信息安全;张喆程(1991-),男,硕士,主要研究方向为大数据、大规模图数据挖掘。

2 相关研究

2.1 相关概念

离群点检测是数据挖掘技术中一项重要的工作，最早出现在统计学领域，后来由 Knorr 等引入到数据挖掘领域。关于离群点，目前受到广泛认可的定义是由 Hawkins 提出的：离群点是一个观察点，它偏离其他观察点十分明显，以至引起怀疑其是由不同机制生成的[1]。这是从全局角度对离群点进行的描述。本文从数学角度对离群点检测算法中的离群点进行定义。

定义 1(离群点) 对于数据集合 D 中的任意元素 x，如果它与数据集中至多 pE 个对象的距离小于 pD，则 x 是关于参数 pE 和 pD 的离群点。

从定义 1 中可以看出，如果 x 在 pD 范围内有不多于 pE 个邻居，则 x 是离群点。其中参数 pD 确定邻域，参数 pE 判断对象 x 是否为离群点。

为了描述离群点的离群程度，研究人员提出了离群度的概念。它是指在数据集中，离群点与其邻域的偏离程度。本文基于 K 近邻来进行离群点检测，所以离群度通过计算数据对象到 K 个最近邻的距离之和来表示。距离越大，说明离群度越高。

定义 2(距离函数) 如果将任意元素 x 表示为特征向量 $\langle a_{1(x)}, a_{2(x)}, \cdots, a_{n(x)} \rangle$，其中 $a_{r(x)}$ 表示 x 的第 r 个属性值，那么元素 x_i 和 x_j 之间的欧几里得距离表达式为：

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{1}$$

定义 3(K 近邻) X_i 表示数据集 D 中的任意数据点，它的 K 个最近邻记作 $KNN_k(X_i)$。计算 X_i 到数据集 D 中所有其他数据点间的距离，选取距离点 X_i 最近的 K 个数据点作为 X_i 的 K 个最近邻。

定义 4(离群度) X_i 表示为数据集合 D 中的任意数据点，X_i 的 K 个最近邻表示为 $KNN_K(X_i)$，设 $N_j \in KNN_K(X_i)$，$j=1,2,\cdots,k$，则离群度表达式为：

$$OD_{X_i} = \sum_{j=1}^{k} d(X_i, N_j) \tag{2}$$

通过离群度计算公式，可以得到数据集合中所有元素的离群度量。对于数据对象 X，它的 K 个最近邻的距离之和可以看作 X 的权重，与只计算最近邻的距离比较而言，它能够更好地反映数据点邻域的稀疏程度。通过下列规则得到全局离群点。

定义 5(Top-n Outlier 规则) 如果数据集 D 中有 m 个对象，分别计算每个数据对象的离群度，其中离群度最高的 n 个对象就是离群点。

根据 Top-n Outlier 规则可知，只需要对数据集中的所有对象计算其离群度，然后按照从大到小的顺序进行排序，取前 n 个对象作为离群点。n 的大小需要人为设定，n 的值选择过小，会丢失部分离群点；但若选择过大，则会增加算法的运行时间和通信量；一般而言，需要根据数据的规模和属性维度选择合适的值。

2.2 MapReduce 计算模型

MapReduce 并行编程模型最早是由 Google 公司于 2004 年提出的，Apache 的开源项目 Hadoop 处理平台正是基于这一模型而开发的，它将待处理的任务划分成两个处理阶段，分别为 Map 阶段和 Reduce 阶段，每个阶段都以一组键值对作为输入和输出。编程人员只需要根据需求对 Map 函数和 Reduce 函数进行重写，而不用考虑分布式环境的底层细节，其具有编程简单、实现容易的特点[12]。

1)输入阶段：将分布式文件系统(Hadoop Distributed File System，HDFS)中的待处理数据分片 Split，每一个 Split 的默认大小为 64MB(用户可自行更改大小)，同时创建数据片的副本，这些 Split 就近分配给 Map 节点。

2)Map 阶段：Mapper 遍历所有的 Split，并将其解析成键值对(Key/Value)，然后调用用户编写的 Map 函数对键值对进行处理，输出若干中间键值对(Key_{inter}/$Value_{inter}$)。

3)Combine 阶段：对每个 Map 节点的输出结果进行排序、合并等处理，这个过程可以看作是一个规约的过程，减少了 Map 节点向 Reduce 节点的数据传输通信量。

4)Reduce 阶段：遍历所有的中间结果，执行用户定义的 Reduce 函数，输出新的键值对(Key_{new}/$Value_{new}$)。

5)输出阶段：此阶段将 Reducer 的输出结果写入到 HDFS 中的输出目录。

3 PODKNN 算法介绍

MapReduce 计算模型提供的并行化设计思路需要算法的计算过程具有一定的独立性，可以将算法划分为独立的子任务进行处理，处理完成后还可以将结果合并。PODKNN 算法的基本思想是：为了使基于 KNN 的离群点检测算法能够实现并行化，需要将数据集按照一定的划分策略分块，然后将子块传输给 Map 节点进行离群度的计算，最后将中间结果传输给一个 Reduce 节点进行合并，并通过 Top-n Outlier 规则得到离群点。

已有一些研究人员提出了 KNN 的并行化策略，文献[17]为了实现并行的 KNN 连接查询，利用了泰森多边形法将数据集划分为子集，能够很好地保留邻近点信息，但算法需要事先确定核心点，为此文中提出了 3 种解决方案，分别是随机选择、最远原则法和 K-means 簇心法，但均不能保证近邻信息的完整性，且计算较为复杂；文献[18]提出了近似近邻的 H-zkNNJ 算法，相较于 H-BRJ 算法，H-zkNNJ 算法在牺牲精确度的情况下，大幅减少了 Reduce 节点的数量和算法的运行时间，并且能够有效处理高维数据，但是该算法同样不能保证近邻信息的完整性。为了能够较为完整地保留数据子集中的近邻信息，并且能够针对离群点检测算法进行优化，使算法过程较为简单、容易实现，本文采用子集扩展的方法对数据集进行划分。

3.1 数据集预处理

数据预处理的过程是将较大规模的数据集分割成多个子集。由定义可知数据点的近邻只与它附近的数据有关，与其它较远的数据无关，所以可以将数据集进行分割。与在数据全集中计算近邻相比，在子集中计算数据点的近邻的计算量大幅下降。但是，处于分割边界的数据点则会面临近邻丢失的情况，分割前后边界数据点的近邻会发生变化，如果只进行硬分割，这一部分损失的信息会影响到离群点检测的准确率，需要根据特定的划分策略对数据集进行分割。

本文采用软硬结合的划分策略对数据集进行分割。首先对数据集进行硬分割，采用 K-Means 聚类算法对数据集进行处理，得到 n 个类簇，c_i($i=1,2,\cdots,n$)表示聚类中心。每个簇

作为分割后的核心集，用 $D_i(i=1,2,\cdots,n)$ 表示。由于预处理过程只需要得到数据集的粗略划分，因此对聚类的精度要求不高。

软分割的主要过程是建立核心集的扩展集。扩展集是在核心集外添加一些冗余的数据点，这些点可以保证核心集中的数据点的 K 近邻不丢失。

首先获取每个簇的簇半径，在每个核心集中计算距离 c_i 最远的簇内数据点，这个距离就是簇半径 r_i：

$$r_i=\max\{d(x_1,c_i),d(x_2,c_i),\cdots,d(x_m,c_i)\} \quad (3)$$

其中，$i=1,2,\cdots,n$，x 是 D_i 中的数据点，共有 m 个。对核心集进行扩展，得到扩展集 D_i'：

$$D_i'=\{x=\| x-c_i \| \leqslant r_i+r^{Extend}\},i=1,2,\cdots,n \quad (4)$$

其中，x 是全体数据集中的数据点，r^{Extend} 是扩展系数。可以得到冗余集 D_i^*：

$$D_i^*=D_i'-D_i \quad (5)$$

通过上述步骤可知，核心集是数据集的硬分割，即 $D=D_1\cup D_2\cup\cdots\cup D_n$ 且 $D_i\cap D_j=\emptyset(i\neq j,i,j=1,2,\cdots,n)$；扩展集是数据集的软分割，扩展集包含了核心集，即 $D_i\subset D_i'$，且 $D=D_1'\cup D_2'\cup\cdots\cup D_n'$。这样，数据集被分割成 n 个互有重叠的子集，当取到合适的扩展系数时，核心集近 K 邻信息基本保持完整。

3.2 PODKNN 算法设计

数据的预处理将较大规模的数据集分割成小规模的数据子集，每个子集内数据对象的离群度的计算可以采用并行的方式，这样可以大幅度提高算法的效率。PODKNN 并行化的设计思路具体如下。

按照 3.1 节中介绍的划分方法对数据进行预处理，得到数据集的子集，包含扩展集、核心集和冗余集，各带有相应标记，以便计算。将扩展集 D_i' 分别分配给一定数量的 Map 节点，每个 Map 节点主要完成核心集中数据点离群度 OD_{X_i} 的计算，由于冗余集中的数据点必然是其他某个核心集中的数据点，为了避免重复计算，只需要计算每个子集中核心集数据点的离群度。然后将 OD_{X_i} 传输给 Reduce 节点，Reduce 函数根据 Top-n Outlier 规则得到离群度最高的 n 个数据点，即离群点。算法流程如图 1 所示。

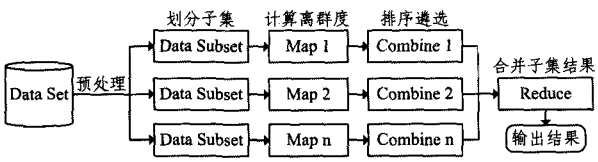

图 1 算法流程图

3.2.1 Map 函数设计

Map 函数以数据点 X_i 在数据集 D 中的全局索引值（如顺序号）和是否为核心集的标记作为输入，函数的输入键值对设置为 $\langle ID_{X_i},Flag\rangle$（其中 $Flag$ 作为标记来区分核心集和冗余集）。函数会根据标记 $Flag$ 来判断是否计算该元素的离群度，函数只选择核心集中的数据点计算离群度，冗余集中的数据点只作为计算核心集中数据点 K 近邻的邻居元素。计算模块计算核心集中数据点的离群度，利用式(1)计算 X_i 与扩展集中所有元素的距离，选择距离最近的 K 个值作为其 K 近邻 $KNN_K(X_i)$，利用式(2)计算 X_i 的离群度 OD_{X_i}。函数输出的键值对设置为 $\langle ID_{X_i},OD_{X_i}\rangle$，只输出 $Flag$ 标记为核心集的数据点。

3.2.2 Combine 函数设计

Combine 函数是对 Map 函数的输出数据进行处理，如排序、合并等，可以减少数据的传输量，从而提高并行效率。Combine 函数的输入键值对设置为 $\langle ID_{X_i},OD_{X_i}\rangle$，函数将 Map 节点的输出数据按照 OD_{X_i} 的大小进行排序，然后选择离群度最大的 n 个数据点，将这 n 个数据输出，输出键值对设置为 $\langle ID'_{X_i},OD'_{X_i}\rangle$。这样做的好处是可以大幅度减少数据的传输量，而且不影响结果的准确性，因为最后的结果混合所有子集数据，取全局离群度最大的 n 个数据点，其值必然在子集中离群度最大的 n 个数据点中选取，所以并不影响最后的结果。

3.2.3 Reduce 函数设计

Reduce 函数主要完成最终结果的输出，只需要一个 Reduce 节点，所有的数据都汇总到该节点。Reduce 函数的输入键值对设置为 $\langle ID'_{X_i},OD'_{X_i}\rangle$，该函数读取到每个 Map 节点中离群度最大的 K 个数据点，然后将这些数据按照离群度大小排序，选择全局最大的 K 个值作为离群点输出。函数的输出键值对设置为 $\langle ID'_{X_i},Rank\rangle$（$Rank$ 代表离群度排序的序列号）。取 $Rank$ 值最大的 K 个值作为离群点。

4 实验与验证

本文通过在实验室环境中搭载 Hadoop 平台进行实验，搭建平台的硬件选取情况如表 1 所列。

表 1 Hadoop 平台节点硬件信息

节点	型号	CPU	内存	硬盘	数量
JobTracker	浪潮 NF5240M3	Xeon E5	16GB DDR3	900GB	1
TaskTracker	Lenovo B4360	赛扬 G1620	2GB DDR3	500GB	9

集群节点之间均以 100M 带宽互联，搭建 Hadoop 平台的软件选取情况如下：Hadoop 版本：2013.10.1 放出的 2.2.0 版本；OS：Ubuntu12.04。实验采用的数据来自于 UCI，为已分类数据集，为了增强离群点检测实验的效果，本文做了适当的处理。

4.1 参数对精度的影响

PODKNN 算法需要设置 3 个参数，分别是 K 近邻中近邻的个数 K、Top-n Outlier 规则中的最值的选取个数 n，以及划分策略中扩展集的扩展系数 r^{Extend}。分别针对这 3 个参数进行实验，采用单一变量原则，即保证其中两个变量不变，通过改变另一个变量的值，观察该变量对算法精度的影响规律。

使用离群点检测准确率衡量算法性能：

算法精度＝找到的正确离群点数目/实际离群点总数

数据采用 1999－2012 年对肺癌患者的调查报告，共有记录 107094 条，数据有 12 个医学指标属性，并根据肺癌的种类分成 4 个类别。在数据集中随机混入 25 名健康人的记录作为离群点，该健康人的记录相对于肺癌患者的簇可以认为是不同机制产生的离群点，实验试图寻找这 25 个离群点。每一轮实验保证其中两个变量不变且都取其最优值，改变剩余的一个变量的值，分别记录运行时间和算法精度。每组实验运行 20 次，结果取平均值，记录结果如表 2 所列。表中“－”表示无法得到结果。

表 2　算法精度实验数据

单一变量	不变量	变量取值	运行时间(s)	算法精度
K	n=25 r^{Extend}=1.8	10	246.65	—
		20	408.43	54.2
		30	459.96	78.6
		40	517.67	97.3
		50	579.32	100.0
		60	625.72	100.0
r^{Extend}	n=25 K=45	0.2	56.92	—
		0.6	113.79	—
		1.0	279.55	71.3
		1.4	354.06	78.9
		1.8	627.27	99.7
		2.2	935.06	100.0
n	r^{Extend}=1.8 K=45	10	428.32	36.6
		15	453.77	60.0
		20	511.62	78.3
		25	565.83	98.7
		30	588.92	100.0
		40	669.74	100.0

关于 K 的取值，需要根据实际中数据集的情况而定，一般以数据集中最大离群簇点的数目为下限，以最小正常簇点数目为上限。从表 2 中可知，随着 K 取值的增加，运行时间也在增长，当 K 的取值在[40,50]之间时，精度可达 100%，随着 K 值的继续增长，运行时间一直在增加且精度维持在 100%。最近邻个数 K 的取值决定了 KNN_K 中元素的数量，影响了离群度的计算，所以过小的 K 值不能完整地反映数据点的近邻信息，而过大的 K 值会增加算法的计算量，因此应当尽量取一个较小的 K 且能够达到较好的检测效果。

当 r^{Extend} 取值较小时，核心集边界的数据点将受到分割线的影响而导致算法精度大幅下降，当 r^{Extend} 取值过大时，扩展集也会随之增加，导致子集中计算 K 近邻时的计算量增加，从而影响了算法效率。本实验中，r^{Extend} 的取值在[1.8,2.2]之间时算法的精度较好，r^{Extend} 的取值与数据的分布情况有关。

n 的取值与离群点的数目相关，当 n 的值接近离群点数目时算法的耗时较少且精度较高；当 n 小于离群点数目时，算法不能检测出所有的离群点；当 n 大于离群点数目时，算法虽然可以检测出所有的离群点，但耗费了更多的时间。

从上述实验数据中可以看出，算法的处理速度与精度是两个矛盾体，较高的精度要求必然要牺牲处理速度，而欲达到较快的处理速度则需要牺牲算法精度。算法参数需要根据实际情况进行相应的变动，以达到最理想的处理结果。

4.2　集群性能实验

加速比和扩展性是衡量算法并行性能的重要指标。加速比是指处理同一份任务时单节点与多节点的运行时间的比值，扩展性体现在算法能够随着节点数目和处理数据规模的增加表现出良好的计算性能。

本实验中采用的是关于气体传感器在动态气体混合物中搜集到的数据，该数据约有 4 千万条记录，包含 19 个数值型属性，大小约为 1647MB。实验中需要 3 种大小的数据集，需将原数据集分成 3 个样本，样本情况如表 3 所列。

表 3　实验样本数据情况

样本	记录数(10^5)	样本大小(MB)
1	125.06427	570
2	202.93420	1045
3	417.85041	1647

实验一　将 9 个子节点依次加入到集群中参与计算，记录每次检测的运行时间，并计算加速比。实验结果如图 2 所示。从图 2 中可以看出，随着节点的增加，加速比呈上升趋势，较大规模的数据比较小规模的数据具有更好的加速比，当数据规模增大时，加速比的上升趋势更加明显。实验表明算法的加速比具有良好的稳定性。

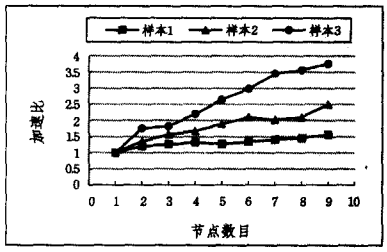

图 2　加速比实验结果

实验二　将 3 种样本分别在 3 个节点、6 个节点、9 个节点上进行离群点检测，运行情况如图 3 所示。从图 3 中可以看出，3 个样本在节点数目增加时运行时间都出现下降趋势，并且数据规模越大，下降趋势越明显。实验表明，当数据规模和节点数目出现增长时，算法的处理性能保持良好，表现出较好的扩展性。

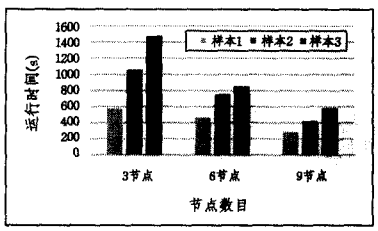

图 3　扩展性实验结果

结束语　随着数据容量的增加，离群点检测算法所消耗的时间也成倍增长，为了应对数据的爆发式增长，本文提出了一种利用 MapReduce 将基于 K 近邻的离群点检测算法并行化的方法，提高了离群点检测算法处理大规模数据集的计算效率。实验表明：基于 MapReduce 的 PODKNN 具有良好的加速比和扩展性，能够有效应对数据规模的急剧增长而带来的算法效率问题。

参 考 文 献

[1] Xue A R, Ju S G, He W H, et al. Study on Algorithms for Local Outlier Detection[J]. Chinese Journal of Computers, 2007, 30(8): 1455-1463(in Chinese)
薛安荣，鞠时光，何伟华，等. 局部离群点挖掘算法研究[J]. 计算机学报，2007，30(8)：1455-1463

[2] Yang M. Research on Algorithms for Outlier Detection[D]. Wuhan: Huazhong University of Science and Technology, 2012(in Chinese)
杨茂林. 离群检测算法研究[D]. 武汉：华中科技大学，2012

[3] Hu T T. Research on Outlier Detection Algorithm in Data Ming [D]. Xiamen: Xiamen University, 2014(in Chinese)
胡婷婷. 数据挖掘中的离群点检测算法研究[D]. 厦门：厦门大学，2014

[4] Angiulli F, Basta S, Pizzuti C. Distance-based detection and prediction of outlier[J]. IEEE Trans. Knowledge and Data Eng., 2006, 2(18): 145-160

(下转第 274 页)

[5] Bhandari A K, Singh V K, Kumar A, et al. Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy[J]. Expert Systems with Applications, 2014, 41(7): 3538-3560

[6] Sun J, Wang X, Huang M, et al. A Cloud Resource Allocation Scheme Based on Microeconomics and Wind Driven Optimization [C]// 2013 8th ChinaGrid Annual Conference (ChinaGrid). IEEE, 2013: 34-39

[7] He Da-kuo, Wang Fu-li, Jia Ming-xing. Uniform design genetic algorithm initial population and operational parameters [J]. Journal of Northeastern University(Natural Science), 2005, 26(9): 828-831(in Chinese)
何大阔,王福利,贾明兴. 遗传算法初始种群与操作参数的均匀设计[J]. 东北大学学报(自然科学版),2005,26(9):828-831

[8] Zhang Ke, Ling Hai-feng. Parameter Turning of Ant Colony Algorithm Based on Uniform Design and Chaos Theory[J]. Computer Engineering, 2012, 38(14): 141-143(in Chinese)
张可,凌海峰. 基于均匀设计和混沌理论的蚁群算法参数调整[J]. 计算机工程,2012,38(14):141-143

[9] Zhang Xu-zhen, Jia Pin-gui. Unequally Spaced Linear Array Synthesis Using Modified DE[J]. Computer Simulation, 2013, 30(6): 226-229(in Chinese)
张旭珍,贾品贵. 基于改进 DE 的非等间距线阵综合[J]. 计算机仿真,2013,30(6):226-229

[10] Chen K, He Z, Han C. A modified real GA for the sparse linear array synthesis with multiple constraints[J]. IEEE Transactions on Antennas & Propagation, 2006, 54(7): 2169-2173

[11] Rani K N A, Malek F. Symmetric linear antenna array geometry synthesis using cuckoo search metaheuristic algorithm [C]// Conference on Communications. IEEE, 2011: 374-379

[12] Khodier M M, Christodoulou C G. Linear Array Geometry Synthesis With Minimum Sidelobe Level and Null Control Using Particle Swarm Optimization[J]. IEEE Transactions on Antennas & Propagation, 2005, 53(8): 2674-2679

[13] Ling S H, Iu H H C, Chan K Y, et al. Hybrid Particle Swarm Optimization With Wavelet Mutation and Its Industrial Applications[J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics, 2008, 38(3): 743-763

[14] Xue Zheng-hui, Li Wei-ming, Ren Wu. Antenna array analysis and synthesis[M]. Beijing: Beihang University Press, 2011 (in Chinese)
薛正辉,李伟明,任武. 阵列天线分析与综合[M]. 北京:北京航空航天大学出版社,2011

[15] Hou Wei, Dong Hong-bin, Yin Gui-sheng. Enhanced Multi-objective Evolutionary Algorithm Based on Decomposition [J]. Computer Science, 2014, 41(2): 114-118(in Chinese)
侯薇,董红斌,印桂生. 一种改进的基于分解的多目标进化算法[J]. 计算机科学,2014,41(2):114-118

[16] Fang Kai-tai. Uniform design and uniform design table[M]. Beijing: Science Press, 1994(in Chinese)
方开泰. 均匀设计与均匀设计表[M]. 北京:科学出版社,1994

(上接第 254 页)

[5] Li C H, Sun Z H. GridOF: An Efficient Outlier Detection Algorithm for very Large Datasets[J]. Journal of Computer Research and Development, 2003, 40(11): 1586-1592(in Chinese)
李存华,孙志挥. GridOF:面向大规模数据集的高效离群点检测算法[J]. 计算机研究与发展,2003,40(11):1586-1592

[6] Yu Dan-tong, Gholambosein S, Zhang Ai-dong. FindOut: Finding Outliers in Very Large Datasets[J]. Knowledge and Information Systems, 2002, 4(4): 387-412

[7] Xue A R, Ju S G. Outlier Mining Based on Spatial Constraint [J]. Computer Science, 2007, 34(6): 207-210(in Chinese)
薛安荣,鞠时光. 基于空间约束的离群点挖掘算法[J]. 计算机科学,2007,34(6):207-210

[8] Wang J C, Zhang J C, Jiang X Y. An Effective and Efficient Approach to Detect and Predict Outliers visually[J]. Computer Science, 2007, 34(6): 200-203(in Chinese)
汪加才,张金城,江效尧. 一种有效的可视化孤立点发现与预测新途径[J]. 计算机科学,2007,34(6):200-203

[9] Liu H, Wu J J, Su J Q. Differentiation Distance-based Outliers Detection Algorithm[J]. Application Research of Computers, 2010, 27(9): 3316-3318(in Chinese)
刘欢,吴介军,苏锦旗. 基于分化距离的离群点检测算法[J]. 计算机应用研究,2010,27(9):3316-3318

[10] Jiang F, Du J W, et al. Outlier Detection Based on Boundary and Distance[J]. Acta Electronica Sinica, 2010, 38(3): 700-705(in Chinese)
江峰,杜军威,等. 基于边界和距离的离群点检测[J]. 电子学报,2010,38(3):700-705

[11] Hu Y, Shi J, Wang C J, et al. Outlier Detection Algorithm based on Global Nearest Neighborhood[J]. Journal of Computer Applications, 2011, 31(10): 2778-2781(in Chinese)
胡云,施珺,王崇骏,等. 基于全局最近邻的离群点检测算法[J]. 计算机应用,2011,31(10):2778-2781

[12] Pan Y, Zhang J B. Parallel Programming on Cloud Computing Platforms: Challenges and Solution [J]. KITCS/FTRA Journal of Convergence, 2012, 3(4): 23-28

[13] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113

[14] Yi X W, Li T R, et al. Performance Testing and Analysis among Different MapReduce Runtime Systems[J]. Computer Science, 2015, 42(5): 24-27(in Chinese)
易修文,李天瑞,等. 不同 MapReduce 运行系统的性能测试与分析[J]. 计算机科学,2015,42(5):24-27

[15] Subramanyam R B V, Sonam G. Map-Reduce Algorithm for Mining Outliers in the Large Data Sets using Twister Programming Model[J]. International Journal of Computer Science and Electronics Engineering, 2015, 3(1): 81-86

[16] Liu X Y, Li J W, Yu H, et al. Adaptive Spectral Clustering Based on Shared Nearest Neighbors[J]. Journal of Chinese Computer Systems, 2011, 32(9): 1876-1880(in Chinese)
刘馨月,李静伟,于红,等. 基于共享近邻的自适应谱聚类[J]. 小型微型计算机系统,2011,32(9):1876-1880

[17] Lu W, Shen Y, Chen S, et al. Efficient processing of k nearest neighbor joins using mapreduce[J]. PVLDB, 2012, 5(10): 1016-1027

[18] Zhang C, Li F, Jestes J. Efficient parallel kNN joins for large data in MapReduce[C]// Proc. of the 15th Int'l Conf. on Extending Database Technology. ACM Press, 2012: 38-49