

一种网格化聚类算法的 MapReduce 并行化研究

张磊^{1,2}, 张公让^{1,2}, 张金广^{1,2}

(1. 合肥工业大学 管理学院, 安徽 合肥 230009;

2. 教育部过程优化与智能决策重点实验室, 安徽 合肥 230009)

摘要: 面对增量式增长的聚类数据, 受云计算并行化处理模式的启发, 文中对一种网格化聚类算法进行了 MapReduce 并行化研究。该算法首先利用网格处理技术对数据进行预处理, 用网格预处理后所得单元的重心点取代该单元中保存的所有点, 然后在 MapReduce 框架下将各个单元的重心点作为聚类分析的基本数据单元, 进行聚类分析。实验结果表明, 该算法 MapReduce 并行化后部署在 Hadoop 集群上运行, 具有与原来相同的聚类效果, 并能节省聚类分析的时间和降低计算的复杂度, 适用于高纬度、增量式海量数据的分析和挖掘。

关键词: 网格; 聚类; 数据挖掘; MapReduce 并行化

中图分类号: TP31

文献标识码: A

文章编号: 1673-629X(2013)02-0060-05

doi:10.3969/j.issn.1673-629X.2013.02.015

MapReduce Parallelization Research of a Clustering Algorithm Based on Grid

ZHANG Lei^{1,2}, ZHANG Gong-rang^{1,2}, ZHANG Jin-guang^{1,2}

(1. School of Management, Hefei University of Technology, Hefei 230009, China;

2. Key Laboratory of Process Optimization and Intelligent Decision-making,
Ministry of Education, Hefei 230009, China)

Abstract: As the incremental growth of clustering data and inspired by the parallel processing model of cloud computing, conducted the MapReduce parallelization research for a clustering algorithm based on grid. This algorithm, firstly, preprocessed the data using the grid processing method, then used the center of gravity of the grid unit as the basic data unit for the clustering analysis under the MapReduce framework, instead of using all the points stored in the unit. The result of experiments demonstrate that this clustering algorithm after its MapReduce parallelization had the same result as before running in the Hadoop cluster. This clustering algorithm can also save the time of analysis and reduce the computational complexity. So, it is suitable for the analysis and data mining of incremental massive data with high latitudes.

Key words: grid; clustering algorithm; data mining; MapReduce parallelization

0 引言

聚类分析是将客观对象的集合根据对象之间的某种相似性分组, 组成多个集合的过程, 使得不同集合中的数据尽可能相异, 而同一集合中的数据尽可能相似^[1]。目前研究人员已经提出大量的聚类分析算法, 大体上可以分为以下几种: 划分方法 (Partitioning

Method)、层次方法 (Hierarchical Method)、基于密度的方法 (Density-based Method)、基于网格的方法 (Grid-based Method) 以及基于模型的方法 (Model-based Method)。其中基于网格的方法是把对象空间量化为有限数目的单元, 形成了一个网格结构, 所有的聚类操作都在这个网格结构 (即量化的空间) 上进行。这种方法的主要优点是它的处理速度很快, 其处理时间独立于数据对象的数目, 仅依赖于量化空间中每一维的单元数目^[2]。这样就为有效地处理高维的、增量式的数据集提供了一种比较快速、有效的聚类分析方法。

现在的一些聚类分析算法集成了多种聚类分析方法的思想, 并不是某一种特定的、具体的聚类分析方法。文中所使用到的网格化聚类算法的均值近似方法就是这样一种算法。网格化聚类算法的均值近似方法

收稿日期: 2012-06-14; 修回日期: 2012-09-18

基金项目: 国家“863”云制造主题项目 (2011AA040501); 国家自然科学基金资助项目 (70871033); 安徽省教育自然科学基金重点项目 (KJ2011A006)

作者简介: 张磊 (1986-), 男, 河南南阳人, 硕士研究生, 研究领域为数据挖掘、机器学习; 张公让, 博士, 研究生导师, 主要研究方向为智能决策、神经网络。

是一种综合了基于密度和基于网格的聚类算法。该算法的基本思想为:对原始数据集利用网格处理技术进行预处理后所得到的网格单元,通过一个重心点来代表该网格单元内保存的所有点,然后对这些重心点进行聚类分析,得到聚类分析结果,也是原始数据集的聚类分析结果。

MapReduce 并行编程模式是由 Google 实验室提出来的,它利用大量廉价的硬件设备组成的集群来处理大规模的数据集,已经成为云计算平台主流的并行数据处理模式^[3]。Apache 开源组织利用 java 语言开发的云计算开源系统 Hadoop 模仿和实现了这一并行编程模式^[4]。现在,云计算开源系统 Hadoop 已经受到全世界越来越多人的关注。

网格化聚类算法的均值近似方法在处理高维的、增量式的数据集时,虽然具有良好的效果,但其串行的计算方法的时间复杂度比较高,处理效率也就大打折扣。文中受云计算开源系统 Hadoop 平台的启发,结合网格化聚类算法的均值近似方法的特点,研究了该方法在 MapReduce 并行编程模式下的实现方法,并进行了相关实验。

1 网格化聚类算法的均值近似方法的 MapReduce 并行化研究

1.1 相关概念

设 $A = \{A_1, A_2, \dots, A_n\}$ 是有界定义域,那么 $S = A_1 \times A_2 \times \dots \times A_n$ 是一个 n 维空间, $D = \{P_1, P_2, \dots, P_n\}$ 为 n 维空间上的一个点。其中 $P_i = \{V_{i1}, V_{i2}, \dots, V_{in}\}$ 。

定义1 给定一个网格单元宽度 d ,空间 s 可以分成 n 个长度等同连续而又不相交的单元,称这些单元为数据单元。对于任意单元 c ,定义其中包含所有点的数目为该单元的密度,用 $\text{density}(c)$ 来表示。

定义2 对任意一个数据单元 c ,称 $\text{mean}(c)$ 为单元 c 的几何重心点,则一个数据单元 c 的几何重心点为:

$$\text{mean}(c) = 1/2(\sum V_{i1}, \sum V_{i2}, \dots, \sum V_{in})$$

其中 i 的范围为 $1, 2, \dots, N$ 。 N 为数据单元 c 中的点的个数。

1.2 MapReduce 并行编程模式

MapReduce 是一种处理海量数据的并行编程模

式,用于大规模数据集的并行计算,由 Map(映射)操作和 Reduce(化简)操作两个阶段组成,Map 操作把任务分解成为多个任务,Reduce 操作把分解后任务处理的结果汇总起来,得到最终结果。适用于 MapReduce 处理的任务有一个根本的要求:待处理的数据集可以分解成许多个的数据集,而且每一个小的数据集都可以完全并行地进行处理^[5]。图1描述了 MapReduce 的运行机制。

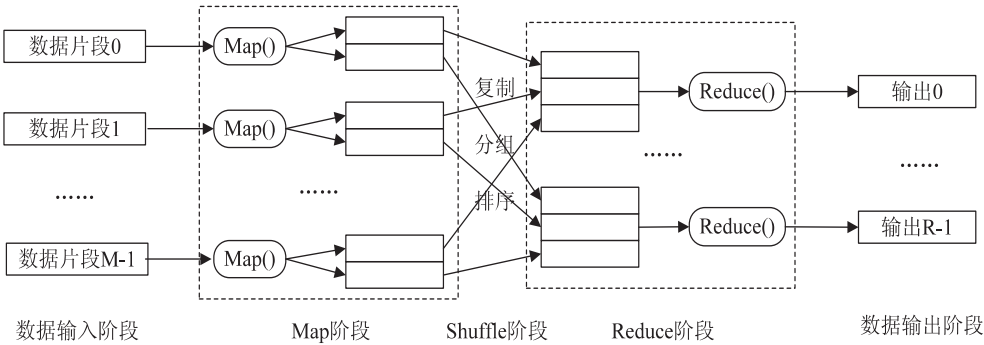


图1 MapReduce 的运行机制

在映射阶段,MapReduce 框架将用户输入的数据分割为 M 个片段,对应 M 个 Map 任务。每一个 Map 操作的输入都是数据片断中的键值对 $\langle \text{Key}_1, \text{Value}_1 \rangle$ 集合,Map 操作调用用户自定义的 Map 函数,输出一个中间态的键值对 $\langle \text{Key}_2, \text{Value}_2 \rangle$ 集合。接着,按照中间态的 Key_2 将输出的数据集进行排序,生成一个新的 $\langle \text{Key}_2, \text{list}(\text{Value}_2) \rangle$ 元组。然后按照 Key_2 的范围将这些元组分割为 R 个片断,对应 Reduce 任务的数目。

在化简阶段,每一个 Reduce 操作的输入是一个 $\langle \text{Key}_2, \text{list}(\text{Value}_2) \rangle$ 片断,Reduce 操作调用用户自定义的 Reduce 函数,生成用户需要的键值对 $\langle \text{Key}_3, \text{Value}_3 \rangle$ 进行输出。

1.3 网格化聚类算法的均值近似方法的 MapReduce 并行化算法设计

1.3.1 网格化聚类算法的均值近似方法的基本步骤

a) 给定网格单元宽度 d ,对原始数据集进行网格预处理,划分单元格,统计具有数据的单元格,并计算每个单元格的重心点,每个单元格的单元密度^[6]。

b) 给定初始聚类数目 k ,选择单元格密度最大的 k 个重心点作为初始聚类中心,将 a) 所得到的重心点分配给距离最近的聚类,并更新聚类的聚类中心,即计算每个聚类中所有重心点的均值,然后开始下一轮的迭代^[7,8]。

c) 删除最后聚类结果中包含点的数目少于给定参数的聚类。

假设原始数据集具有 n 个数据点,经过网格化预处理后,存在数据点的单元格数目为 m ,期望得到的聚类个数为 k ,算法的迭代次数为 t ,一次迭代中计算待分

配重心点到聚类中心点距离的时间复杂度为 p , 则串行实现算法的时间复杂度比较高, 为 $O(n) + m \times k \times t \times p$ 。其中一次迭代中计算待分配重心点到聚类中心点距离的操作是这个算法中最耗时间, 也是最容易实现并行处理的部分。在计算一个待分配点到聚类中心的距离的同时, 也可以计算其他的待分配点到聚类中心的距离。

1.3.2 网格化聚类算法的均值近似方法进行 MapReduce 的基本思路

在串行实现方法中利用 K-means 聚类算法, 对重心点进行聚类时, 每一次迭代可以看作是一次 MapReduce 计算过程, 并行地计算每一个待分配点到聚类中心的距离。

在 MapReduce 并行编程模式下, 待处理的数据记录需要先转化成 MapReduce 并行编程模式能够识别的形式, 然后再对待处理的数据记录进行分块, 并且块与块之间的数据记录是相互独立的^[9,10]。该过程由 MapReduce 运行的环境来实现, 无须人工完成。从而, 该并行化实现的主要工作就是设计迭代过程中的几个函数: Map 函数、Combine 函数和 Reduce 函数。

a. Map 函数的设计。

Map 函数的输入为 $\langle \text{Key}_1, \text{Value}_1 \rangle$ 键值对集合。 Key_1 为当前重心点所属的聚类的类别 ID, Value_1 为当前重心点各维坐标值的字符串。首先, 根据 Value 中各维坐标的值, 计算该重心点到 k 个聚类中心点的距离, 找到距离最近的聚类的类别 ID, 然后输出新的 $\langle \text{Key}_2, \text{Value}_2 \rangle$ 键值对集合, 其中 Key_2 为当前重心点所属的新聚类的类别 ID, Value_2 为当前重心点各维坐标值的字符串。Map 函数的伪代码为:

```
Map(  $\langle \text{Key}_1, \text{Value}_1 \rangle, \langle \text{Key}_2, \text{Value}_2 \rangle$  )
{
    从  $\text{Value}_1$  中解析出重心点的各维坐标值;
    变量 MinDis 为重心点到  $k$  个聚类中心点的距离的最小值,
    初始化为重心点到第一个聚类中心点的距离;
    ClassID 为该重心点所属聚类的类别, 初始化为 0;
    For  $i = 0$  to  $k - 1$ 
    {
        变量 Dis = 重心点到第  $i$  个聚类中心点的距离;
        If Dis 小于 MinDis {
            MinDis = Dis;
            ClassID =  $i$ ;
        } else {
            //重心点距离第 0 个聚类中心点距离最近, 所属聚类类别
            为 0
        }
        MinDis = 初始化值;
        ClassID = 初始化值;
    }
}
```

将 ClassID 作为新的 Key_2 ;
将重心点各维坐标值作为 Value_2 ;
输出 $\langle \text{Key}_2, \text{Value}_2 \rangle$;

b. Combine 函数的设计。

对 Map 函数输出的中间结果, 先做 Combine 操作, 即对中间结果中具有相同的 Key_2 的 $\langle \text{Key}_2, \text{Value}_2 \rangle$ 键值对进行合并, 产生一个新的 $\langle \text{Key}_2, \text{list}(\text{Value}_2) \rangle$ 元组。输出 $\langle \text{Key}_2, \text{Values} \rangle$ 中的 Values 包括两部分的信息: 具有相同 Key_2 值的重心点数目和所有这些重心点各维坐标值的累加和。Combine 操作可以减少中间结果 $\langle \text{Key}_2, \text{Value}_2 \rangle$ 键值对的数目, 从而减少数据传输中的网络流量。Combine 函数的伪代码为:

```
Combine(  $\langle \text{Key}_2, \text{list}(\text{Value}_2) \rangle, \langle \text{Key}_2, \text{Values} \rangle$  )
{
    list(  $\text{Value}_2$  ) 是一个列表, 记录着具有相同 ClassID 的所有重心点;
    Foreach list(  $\text{Value}_2$  )
    {
        从 list(  $\text{Value}_2$  ) 中解析出每一个重心点的各维坐标值, 将对应的各维坐标值累加;
    }
    输入的  $\text{Key}_2$  仍作为输出的  $\text{Key}_2$ ;
    将 list(  $\text{Value}_2$  ) 中点的个数和这些点各维坐标值的累加和一起作为 Values;
    输出  $\langle \text{Key}_2, \text{Values} \rangle$ ;
}
```

c. Reduce 函数的设计。

把 Combine 函数输出的结果按照 Key_2 的值的范围划分成 R 份, 对应 R 个 Reduce 任务。在 Reduce 函数中, 首先根据各个 Combine 函数传递过来的 $\langle \text{Key}_2, \text{Values} \rangle$, 解析出具有相同 Key_2 值的重心点的数目和所有这些重心点各维坐标值的累加和; 然后将对应的各维坐标值累加, 再除以总的具有相同 Key_2 值的重心点的数目, 即得新的聚类中心点的坐标; 最后用该新的聚类中心点坐标更新原有的聚类中心点坐标, 然后进行下一次迭代, 直到算法收敛。Reduce 函数的伪代码为:

```
Reduce(  $\langle \text{Key}_2, \text{list}(\text{Values}) \rangle, \langle \text{Key}, \text{Value} \rangle$  )
{
    list(  $\text{Values}$  ) 是一个列表, 记录着从 Combine 函数传递过来的具有相同 ClassID 的所有重心点;
    Foreach list(  $\text{Values}$  )
    {
        将各个 Values 中各维坐标值的累加和, 继续累加;
        累加各个 Values 中点的个数, 记作 Counts;
    }
    将各维坐标值最终的累加和, 除以最终的累加的点的个数 Counts, 得到新的聚类中心点坐标;
```

```
    输入的 Key2作为输出的 Key;  
    将新的聚类中心点坐标作为 Value;  
    输出<Key, Value>;  
}
```

1.3.3 算法的时间复杂度分析

在 MapReduce 并行编程模式下,待处理的数据记录被存储在 Hadoop 平台的分布式文件系统 (HDFS, Hadoop Distributed File System) 中,并且被分割成块,储存在不同的节点上。默认情况下,文件块的大小为 64MB^[11]。在 Map 阶段, M 个 Map 任务可以分布在 N 台计算机上并行运行,每台计算机可以运行多个任务。如此一来,算法串行实现的过程中由一台计算机处理的运算,现在就由 N 台计算机并行的处理,其时间复杂度也就大幅度降低。如果每台计算机平均完成 P 个 Map 任务 ($M = P \times N$),则其时间复杂度为 $O(n) + P/M \times m \times k \times t \times p$ 。

2 实验与结果分析

2.1 实验环境、数据集

实验选用 KEEL (Knowledge Extraction based on Evolutionary Learning)^[12]网站上的人工数据集对算法进行实验处理。该数据集总共有数据 5300 条。首先对数据集进行预处理,对所有数据扩大 100 倍,即 $x * 100$ (其中 x 表示数据集中任意一个数据);然后对所得数据集进行实验,其中实验中网格单元宽度 d 为 50。本实验利用虚拟机 VMware Workstation 在 Linux 操作系统下进行,虚拟机配置为 CPU Inter Core Duo E8400 3.00GHz,内存 1024MB。

2.2 实验结果

实验结果如图 2~6 所示,其中网格单元宽度 $d = 50$ 。

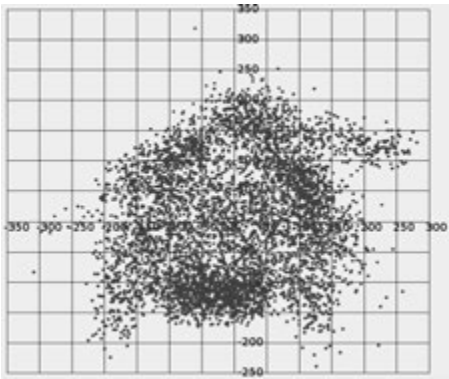


图2 数据集未经网格化处理显示出来的效果图

从实验结果可以看出来,同一个数据集,利用网格化聚类算法的均值近似方法进行聚类与利用其他的聚类算法进行聚类具有相同的结果。然而网格化聚类算法的均值近似方法进行聚类时需要处理的数据,相比其他的聚类算法却少得多(本实验中,经过网格化处

理后的实验数据仅剩 94 条),因此,聚类的速度有了明显的提升。

实验表明,网格化聚类算法的均值近似方法显示了其在处理高维的、增量式的数据集上的明显优势,再将其 MapReduce 并行化后部署在 Hadoop 集群上,其计算速度的优势将更加明显。

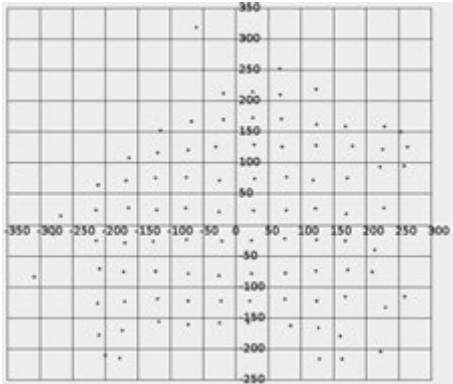


图3 数据集经过网格化处理后显示的效果图(有效网格数为 94 个)

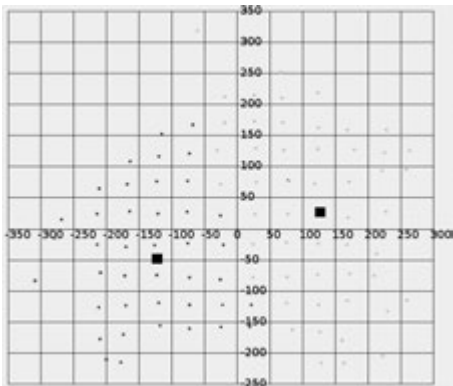


图4 利用网格化聚类算法的均值近似方法聚类后显示的效果图

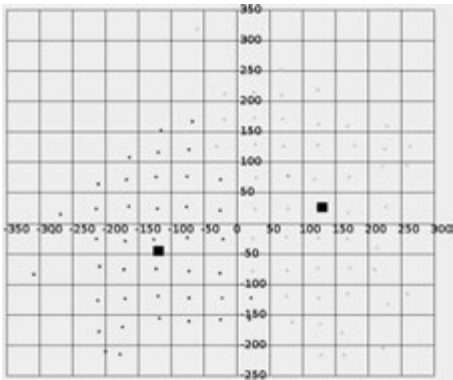


图5 利用均值近似方法的 MapReduce 并行化聚类后显示的效果图

3 结束语

本文利用云计算平台的 MapReduce 并行化编程模式,对网格化聚类算法的均值近似方法做了 MapReduce 并行化研究。该算法 MapReduce 并行化后在 Ha-

doop 集群上运行,不仅具有与原来串行环境下相同的聚类效果,而且能节省聚类分析的时间和降低计算的复杂度,适用于处理高维的、增量式的数据集。笔者将会进一步探索如何对该算法的 MapReduce 并行化过程进行改进,以获得更好的实验结果。

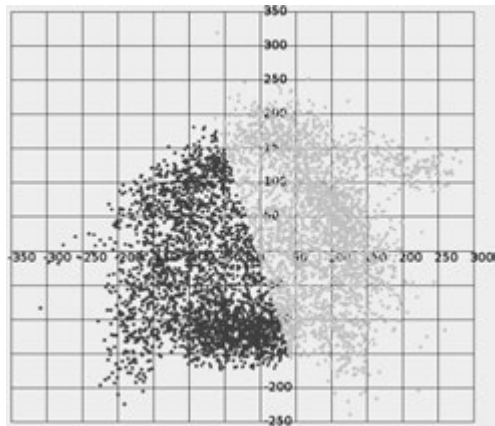


图 6 原数据集利用 K-Means 聚类算法聚类后显示的效果图

参考文献:

- [1] 杨善林,倪志伟. 机器学习与智能决策支持系统[M]. 北京:科学出版社,2004:331-332.
- [2] 张伟莉,倪志伟,赖建章. 一种新的基于网格的聚类算法[J]. 计算机应用研究,2008,25(5):1337-1338.

- [3] Wegener D,Mock M,Adranale D,et al. Toolkit-based High-performance Data Mining of Large Data on MapReduce Clusters [C]//IEEE International Conference on Data Mining Workshops (ICDMW). Washington:IEEE,2009:296-298.
- [4] Apache Hadoop. Hadoop[EB/OL]. 2012-06-05. <http://hadoop.apache.org>.
- [5] 刘 鹏. 云计算[M]. 北京:电子工业出版社,2010:163-165.
- [6] 焦 誉,赖建章,柯 佳. 一种基于密度的网格动态聚类算法的研究[J]. 安徽大学学报(自然科学版),2007,31(1):31-33.
- [7] 周爱武,于亚飞. K-Means 聚类算法的研究[J]. 计算机技术与发展,2011,21(2):62-63.
- [8] 黄 韬,刘胜辉,谭艳娜. 基于 k-means 聚类算法的研究[J]. 计算机技术与发展,2011,21(7):54-55.
- [9] 江小平,李成华,向 文,等. K-means 聚类算法的 MapReduce 并行化实现[J]. 华中科技大学学报(自然科学版),2011,39(Sup):121-123.
- [10] 赵卫中,马慧芳,傅燕翔,等. 基于云计算平台 Hadoop 的并行 k-means 聚类算法设计研究[J]. 计算机科学,2011,38(10):166-167.
- [11] 王 鹏. 云计算的关键技术与应用实例[M]. 北京:人民邮电出版社,2010:74-76.
- [12] KEEL. A software tool to assess evolutionary algorithms for data mining problems [EB/OL]. 2011. <http://sci2s.ugr.es/keel/datasets.php>.

(上接第 59 页)

由图 6 不难看出,量子遗传算法和遗传算法均在信噪比为 5dB 时,误码率均降为零,但在 2dB ~ 4dB 之间,量子遗传算法的误码率比遗传的低。在仿真过程中,量子遗传算法运行的时间少于遗传算法运行的时间。由此得出:量子遗传盲检测算法能够较为成功的恢复出发送序列,具有一定的研究价值。

4 结束语

介绍了通信中的盲均衡理论,并将量子遗传算法应用到盲均衡技术中,通过研究数学模型和大量的仿真实验证明量子遗传算法的可行性,在选取适当参数的情况下,利用量子遗传算法,能够得到较好的结果。

参考文献:

- [1] 张立毅,刘 婷. 遗传算法优化神经网络权值盲均衡算法的研究[J]. 计算机工程与应用,2009,45(11):162-163.
- [2] 杨俊安,庄镇泉,史 亮. 多宇宙并行量子遗传算法[J]. 电子学报,2004,32(6):923-928.
- [3] 李承祖. 量子通信和量子计算[M]. 长沙:国防科技大学出版社,2000:168-213.

- [4] Giannakis G B,Hua Y B,Stoica P,et al. Signal processing advance in wireless and mobile communications, volume: trends in channel estimation and equalization[M]. 北京:人民邮电出版社,2002:197-209.
- [5] Tong L,Xu G,Kailath T. Blind channel identification and equalization using second-order statistics: a time-domain approach[J]. IEEE Trans. on Inform. Theory,1994,40(3):340-349.
- [6] Ding Z,Li Y. Blind equalization and identification[M]. New York:Marcel Dekker,2000:175-202.
- [7] 张志涌,张 昀. 复数 Hopfield 盲恢复多用户 QPSK 信号[J]. 东南大学学报,2008,38(12):18-22.
- [8] 于舒娟,张志涌. 含公零点 SIMO 信道 QPSK 序列盲检测[J]. 东南大学学报,2005,36(6):867-871.
- [9] Han K H,Kim J H. Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization[J]. IEEE Trans. on Evolutionary Computation,2002,6(6):580-593.
- [10] Nguyen H L. Blind source separation for convolutive mixture[J]. Signal Processing,1995(45):209-229.
- [11] 杨淑媛,刘 芳,焦李成. 一种基于量子染色体的遗传算法[J]. 西安电子科技大学学报(自然科学版),2004,31(1):76-79.