文章编号: 1001-0920(2006)09-0974-05

一种基于密度单元的自扩展聚类算法

于勇前, 赵相国, 王国仁, 陈衡岳
(东北大学 信息科学与工程学院, 沈阳 110004)

摘 要: 提出一种高效的基于密度单元的自扩展聚类算法SECDU. 首先将数据空间等分为若干个密度单元, 再根据数据点的位置将其划分到所属的密度单元中, 然后针对密度单元进行聚类. 聚类首先产生在数据最密集的区域, 然后向周围低密度区域延伸. 聚类在延伸的过程中体积逐渐增大, 密度逐渐减小, 直到聚类的密度达到一个事先规定的限度时为止. 算法在保留原有数据分布特性的前提下利用密度单元对数据进行压缩, 并在保证具有较好效果的前提下大幅度地提高了聚类的速度.

关键词: 聚类分析; 密度单元; 聚类空间; 聚类算法
中图分类号: TP311.13 **文献标识码**: A

An Self-expanded Clustering Algorithm Based on Density Units

YU Yong-qian, ZHAO Xiang-guo, WANG Guo-ren, CHEN Heng-yue
(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China. Correspondent: ZHAO Xiang-guo, E-mail: zhaoxiangguo@mail.neu.edu.cn)

Abstract: An efficient self-expanded clustering algorithm based on density units (SECDU) is presented. The whole data space is divided into several density units equally. Each data point is put into a density unit according to the data point possition. The area with the highest data density is the starting point of clustering and it is expanded to the low-density area. The whole process will not stop until densities of all clusters reduce to the threshold set in advance. By compressing data into data units, SECDU can cluster large dataset at a high speed without destroying distribution feature.

Key words: Clustering analysis; Density unit; Cluster space; Cluster algorithm

1 引 言

聚类分析是数据挖掘技术中重要的组成部分, 其主要目的是将数据空间中的数据点划分到若干个类中, 其中将距离相近的数据点划分到相同的类中, 而将距离较远的数据点划分到不同类中. 虽然已经出现了较多聚类分析算法, 但针对数据规模日益增大这一现实情况, 高效率的聚类算法始终是聚类分析领域中主要研究方向之一.

本文提出一种基于密度单元的聚类算法SECDU, 该算法具有较好的聚类效果和较快的聚类速度, 对具有不同分布特性的数据集均有很好的适应性.

2 准备工作

现有的聚类算法可以分为分割型聚类[1]、层次聚类[2~4]、混合型聚类[5]和基于密度的聚类[6~8]等类型.

分割型聚类首先从全部数据中随机抽取若干个数据作为种子, 将距离种子较近的数据与种子划分到同一个类中, 直到将全部数据都划分到某个类中为止. 这种聚类算法的优点是算法复杂度小、聚类速度快, 缺点是聚类结果跟随机选种关系密切, 可能产生质量较差的聚类结果.

层次聚类又分为两种类型, 一种自顶向下, 另一种自底向上. 前者首先将全部数据看作一个类, 然后

收稿日期: 2005-09-08; **修回日期**: 2005-12-28.
基金项目: 国家自然科学基金项目(60273079, 60573089).
作者简介: 于勇前(1964—), 男, 沈阳人, 高级工程师, 从事数据挖掘及应用的研究; 王国仁(1968—), 男, 湖北崇阳人, 教授, 博士, 从事数据库理论和技术、分布与并行数据库等研究.

以某种规则或方法将大类划分成若干小类, 直到满足某个事先规定的条件时聚类结束; 后者相反, 首先将每个数据点看作一个类, 然后将若干个小类合并生成较大的类, 直到最后将全部数据合并成为一个类为止. 层次聚类的优点是聚类结果稳定且质量较高, 缺点是算法的复杂度较大.

混合型聚类结合了上述两种聚类算法的特点, 但仍没有彻底消除以上缺点.

基于密度的聚类是以数据分布的疏密为基础对数据集进行聚类. 本文所述的聚类算法即属于这种类型的聚类算法.

3　SECDU 算法介绍

3.1　相关定义

定义 1(聚类空间)　将包含了全部 d 维数据的 d 维空间称为聚类空间, 用 Ω^d 表示. 将每个数据看作 Ω^d 中的一个点, 点的坐标值就是数据在各个维上的值.

定义 2(密度单元)　对 d 维聚类空间 Ω^d 的各个维进行等宽划分, 将第 i 维等分为 n_i 份, $1 \le i \le d$. 这样, Ω^d 等分为 $\prod_{i=1}^{d} n_i$ 个超立方体区域, 每个超立方体区域就是聚类空间中的一个密度单元, 记为 u, $u \in \Omega^d$.

定义 3(密度单元位置向量)　在对每个维进行划分的同时, 沿各维的正方向对每个等分区域进行编号. 例如第一个区域编号为 1, 第二个区域编号为 2 等. 用 d 维向量 $v(v_1, v_2, \cdots, v_i, \cdots, v_d)$ 来表示密度单元 u 在 Ω^d 中的位置, $v_i \in N$ (N 为自然数集合), $1 \le i \le d$.

定义 4(密度单元间的绝对距离)　设两个密度单元 $u_m, u_n \in \Omega^d$, 位置向量分别为 $m(m_1, m_2, \cdots, m_d)$, $n(n_1, n_2, \cdots, n_d)$, 则它们的绝对距离 $\mathrm{dist}_{abs}(u_m, u_n) = \sum_{i=1}^{d} |m_i - n_i|$.

定义 5(邻居密度单元)　密度单元 u 的邻居密度单元 u' 是与 u 直接相邻的密度单元, 它具有如下性质: $\mathrm{dist}_{abs}(u, u') = 1$.

定义 6(密度单元的密度)　每个密度单元都是 Ω^d 中的一个区域, 区域内包含数据点的个数称为该密度单元的密度, 它是一个非负整数, 记为 ρ.

定义 7(基于密度单元的类)　聚类结果中的每个类是由位置相邻的若干密度单元组成的, 且每个类至少包含一个密度单元.

定义 8(邻居类)　如果两个不同的类分别包含互为邻居的密度单元, 则这两个类就是邻居类. 其形式化定义如下: 设 $c \ne c'$ 是两个不同的类, $(\exists u \in c)(\exists u' \in c')(\mathrm{dist}_{abs}(u, u') = 1) \Leftrightarrow c$ 和 c' 互为邻居类, 其中 u 和 u' 是密度单元.

定义 9(类的密度)　设 $u_1, u_2, \cdots, u_p \in \Omega^d$ 是类 c 包含的全部 p 个密度单元, 对应的密度分别为 $\rho_1, \rho_2, \cdots, \rho_p$, 则类 c 的密度为

$$\rho(c) = \frac{1}{p} \sum_{i=1}^{p} \rho_i \tag{1}$$

定义 10(聚类密度下限 ρ_d(为非负整数))　类总是由高密度区域向低密度区域延伸, 在延伸的过程中, 类的密度逐渐降低, 当类密度低于 ρ_d 时, 停止延伸. 用该参数来限定聚类结果中各个类的最低密度.

定义 11(拒绝聚类密度 ρ_0(为非负整数))　当高密度的类通过合并周围低密度类进行延伸时, 周围密度不大于 ρ_0 的类将不被高密度类合并.

定义 12(有效类)　在聚类结果中, 密度不小于 ρ_d 的类为有效类. 有效类中的全部数据构成该类.

定义 13(无效类)　在聚类结果中, 密度小于 ρ_d 的类为无效类. 无效类中的数据视为噪音数据.

3.2　SECDU 算法描述

聚类前将聚类空间划分为多个密度单元并统计各密度单元的密度, 此时将每个密度单元看作一个类, 密度单元的密度就是类的密度. 然后设置聚类参数 ρ_d 和 ρ_0, 建立两个集合 Aset(保存密度较大并有可能合并其他类的类) 和 Pset(保存不能主动合并其他类, 只能被 Aset 中的类合并的类).

算法首先将密度大于等于 ρ_d 的类保存在 Aset 中, 将其余的类保存在 Pset 中, 然后进行聚类操作. 聚类算法描述如下:

```
while(Aset ≠ NULL)
    c1 = get_max_density_cluster(Aset);
    c2 = get_highest_density_neighbor(c1);
    if (c2.density > ρ0 and can_merge(c1, c2))
        new_cluster = merge(c1, c2);
        insert(new_cluster, Aset); }
    else
        insert(c1, Pset);
loop while
return(Pset);
```

其中: get_max_density_cluster(Aset) 返回 Aset 中密度最大的类, 同时将这个类从 Aset 中删除; get_highest_density_neighbor(c_1) 函数返回 c_1 邻近类中密度最大的类; can_merge(c_1, c_2) 函数判断类 c_1, c_2 是否可以合并, 如果 c_1, c_2 合并后形成的新类密度不小于 ρ_d 则函数返回真, 否则返回假; merge(c_1, c_2) 函数先将 c_2 从其所属的集合中删除,

再合并类 c_1 和 c_2, 最后返回合并后形成的新类; insert(new_ cluster, A set) 函数将类 new_ cluster 插入到集合A set 中; insert(c_1, Pset) 将类 c_1 插入到集合 Pset 中.

聚类算法结束时集合 Pset 中包含了全部的聚类结果. 其中密度不小于 ρ_d 的类是有效类, 其余的是无效类.

图 1 举例说明了聚类的全部过程. 数据空间被划分为 4 × 5 个密度单元, 聚类参数 ρ_d = 4, ρ_0 = 2, 如图 1(a) 所示. 聚类开始时将每个密度单元作为一个类, A set 中保存密度不小于 4 的类, 此时有 3 个类符合条件, 分别用 A, B 和 C 进行标识, 如图 1(b) 所示. 在A set 中找到密度最大的类B, 其密度为 6. 再找到类B 的最大密度的邻近类A. 因为A 的密度大于 ρ_0 且A 与B 满足合并条件, 所以将A 和B 合并形成新的聚类D, 如图 1(c) 所示. 根据定义 9, 类D 的密度为(6 + 5)/2 = 5.5. 至此, 一次聚类延伸结束, A set 中包含了两个类C 和D. 再找出A set 中密度最大的聚类D, 同理它可以合并其密度最大的邻近类, 合并后形成新的聚类E, 如图 1(d) 所示. E 的密度为(6 + 5 + 3)/3 = 4.67. 聚类继续进行直到形成图 1(e) 的格局. 此时A set中有两个类E 和F. 从中选择密度最大的类 E, 它密度最大的邻近聚类的密度为 2. 因为这个密度不大于 ρ_0, 所以 E 与该类不能合并, 此时将 E 从A set 移动到 Pset 中, 同样将类 F 也从A set 移动到 Pset 中. 此时 A set 为空, 聚类结束, Pset 中包含了两个有效类 E 和 F, 密度分别为 4.67 和 4, 其他的类均是无效类.

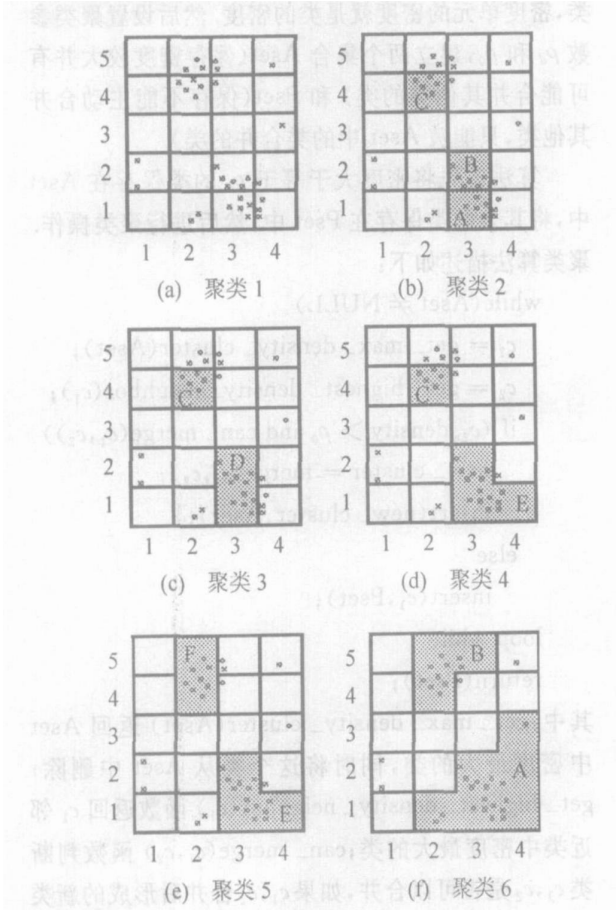

图 1 聚类算法举列

如果设置聚类参数 ρ_d = 3, ρ_0 = 0, 则最终的聚类结果如图 1(f) 所示. Pset 中两个有效类分别用A 和B 表示, 其密度分别为 3 和 3.33.

4 聚类参数分析及设置

4.1 聚类参数分析

利用较大的 ρ_d 值得出的聚类结果中各有效类的密度较大, 但包含密度单元的数量较少, 这样的参数具有较好的去噪能力, 缺点是可能将处在主体数据边缘的零星数据当作噪音数据而排除在类外. 此时通过适当调低 ρ_0 值可以在一定程度上弥补该缺点.

另一方面, 利用较小的 ρ_d 值可以聚出范围较大的类, 而且能够包含较多的边缘数据, 对噪音数据较少的数据集有较好的聚类效果. 但容易将噪音数据一同聚到类中. 这时就需要适当调高 ρ_0 值来弥补该缺点.

ρ_0 是拒绝聚类密度, 在类延伸的过程中, 依据 ρ_0 值来判断是否合并周围密度较低的类. 特别地, 当类的密度较高且范围较大时, 该类吸收低密度类的能力就很强. 如果没有利用 ρ_0 值对聚类延伸进行限制, 这个类就会扩展得非常庞大, 而且可以吸收周边很大范围内的空密度单元和较多的噪音数据, 使得聚类结果质量变差.

4.2 聚类参数的设置

ρ_d 和 ρ_0 对聚类的形成起着不同的作用, 合理地设置 ρ_d 和 ρ_0 值, 会产生质量较高的聚类结果. 为确定聚类参数的取值, 首先需要考察密度单元密度的分布情况. 因为影响密度单元密度的因素是多方面的, 既与数据分布情况有关, 又与密度单元划分有关, 而且这些影响因素没有相关性, 所以可以认为密度单元密度的分布近似服从正态分布[9], 记为 $X \sim N(\mu, \sigma^2)$, 其中: μ 是密度单元密度均值, σ 是密度单元密度的标准差. 因为在聚类前无法得知 μ 和 σ 的值, 因此要根据数据空间中全部的密度单元来估计出它们的值.

公式 1 设 $u_1, u_2, \cdots, u_n$ 是 Ω^d 中全部 n 个密度非零密度单元, 其密度分别为 $\rho_1, \rho_2, \cdots, \rho_n$, 则密度均值的矩估计值为

$$\hat{\mu} = \overline{X} = \frac{1}{n}\sum_{i=1}^{n}\rho_i, \tag{2}$$

密度标准差的矩估计值为

$$\hat{\sigma}=\left(\frac{1}{n}\sum_{i=1}^{n}(\rho_i-\overline{X})^2\right)^{\frac{1}{2}}. \tag{3}$$

因为有效类中的密度单元都是密度非零密度单元, 统计密度为零的密度单元对聚类不但没有意义, 反而会影响聚类效果, 所以公式 1 中没有统计密度为零的密度单元

正常情况下噪音数据只占全部数据的很少一部分, 其余数据均为有效数据, 因此可以作如下假设: 令密度大于 ρ_d 的密度单元占到全部单元的 90% 以上, 令密度小于 ρ_0 的密度单元占到全部密度单元的 0.5% 以下, 而另外 9.5% 的数据可能被划分到有效类中, 也可能被划分到无效类中, 这要根据数据分布的情况而定. 这样设置聚类参数可使聚类结果中有效类包含大部分的数据, 而无效类只包含少数的噪音数据. 于是建立如下方程组:

$$\begin{cases} P\{X>\rho_d\}=90\%, \\ P\{X<\rho_0\}=0.5\%, \end{cases} \tag{4}$$

其中 $X\sim N(\mu,\sigma^2)$. 根据 $\Phi(1.28)\approx 0.9$, $\Phi(2.57)\approx 0.995$[9], 得到聚类参数值

$$\rho_d=\hat{\mu}-1.28\hat{\sigma},$$
$$\rho_0=\hat{\mu}-2.57\hat{\sigma}.$$

5　实验与测试

本次实验环境如下: CPU 为 Pentium 4 2.6 GHz, 内存为 512 MB, 硬盘为 80 G 7200 转/min, 操作系统为 Microsoft Windows XP 操作系统

5.1　实验方案介绍

实验中针对 4 种具有不同分布特性的数据集进行聚类, 如图 2 所示. DS_1[2] 的噪音数据少, 数据分布简单; DS_2[2] 具有明显的层次特性; DS_3[7] 的数据分布具有特殊的形状; DS_4[4] 是一个规模较大且包含较多噪音数据的数据集

实验中密度单元在各维上边长取 $[\log_2^{0.5N}]$, 其中 N 是数据集中数据点的数量. 这种划分的粒度比较适中.

5.2　参数设置的合理性分析

表 1 列出了 SECDU 算法对 4 个数据集进行聚类的结果. 从表 1 中可以看出, 有效类中包含的数据点的数量占全部数据点数量的 92%～99%. 该实验结果符合 4.2 节的理论分析结果

表 1　SECDU 算法聚类结果的数据统计

数据集		DS_1	DS_2	DS_3	DS_4
数据点数		73 776	122 248	45 161	250 140
有效类	数据点数	71 909	114 607	44 628	230 454
	比例 %	97.47	93.75	98.81	92.13
无效类	数据点数	1 867	7 641	533	19 686
	比例 %	2.53	6.25	1.18	7.87

从图 2 中可以看出, 噪音数据较多的数据集为 DS_2 和 DS_4, 其聚类结果中无效类包含的噪音数据占全部数据的比例也较大. 可见, SECDU 算法对噪音数据具有较好的过滤效果

5.3　聚类效果测试

图 3 显示了利用 SECDU 算法对 4 个数据集进行聚类的结果. 从图 3 中可以看出, SECDU 算法对 4 种具有不同分布特性的数据都能聚出令人满意的结果. 算法不但能够识别出数据分布的形状, 对噪音数据也有很好的过滤能力. 可见, SECDU 算法是一种效果很好的聚类算法

SECDU 算法的聚类效果总结如下:

1) 该算法对于具有层次特征的数据集具有较

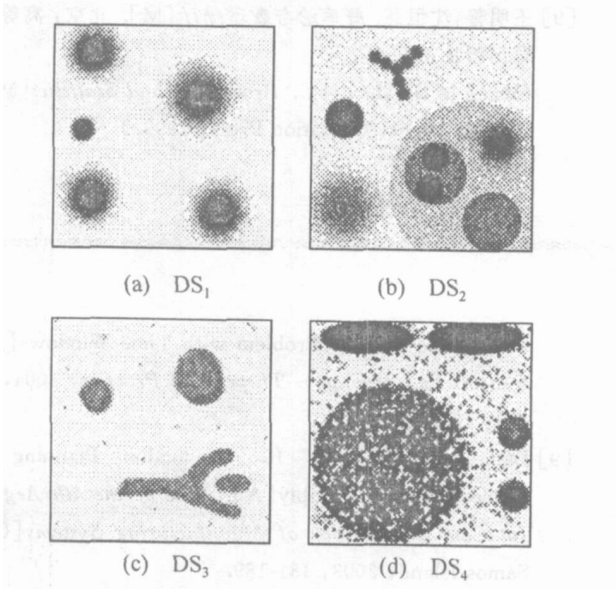

图 2　4 种具有不同分布特性的数据集

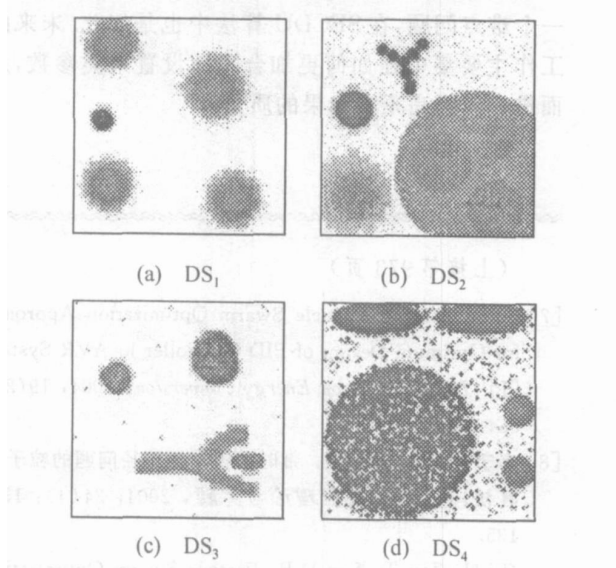

图 3　4 个数据集的聚类结果

好的聚类效果;

2) 该算法对于不同形状的数据集具有较好的聚类适应性;

3) 该算法对数据集中的噪音数据具有较好的抗干扰能力;

4) 该算法对于文本等非结构化的数据聚类并不适合[6, 7].

5.4 与DBSCAN和SUDBC算法的性能比较

DBSCAN 算法是一种高效率的聚类算法, 是典型的基于密度的聚类算法; SUDBC 算法也是一种基于密度单元的聚类算法. 图 4 为 3 种聚类算法对不同规模数据集进行聚类的性能比较. 从图 4 中可以看出, SECDU 算法具有最快的聚类速度, 与其他两种算法相比, 具有两个数量级的速度优势. 可见对于大规模的数据集, SECDU 算法的速度优势非常明显.

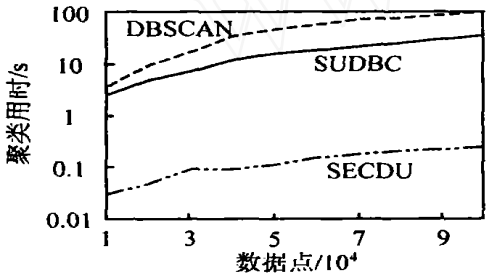

图 4 3 种聚类算法性能比较

6 结 语

本文介绍了一种基于密度单元的聚类算法 SECDU. 该算法聚类效果好、速度快、对具有不同分布特性的数据集有较好的适应性, 适合聚类大规模的数据集.

聚类参数的设置问题在聚类分析领域中始终是一个难点问题, 在SECDU 算法中也是如此. 未来的工作主要集中在如何更加合理地设置聚类参数, 从而进一步提高聚类结果的质量.

参考文献(References)

[1] Macqueen J. K-means: Some Methods for Classification and Analysis of Multivariate Observations [A]. *The 5th Berkeley Symp on Mathematical Statistics and Probability* [C]. Berkeley, 1976: 56-68.

[2] Markus M, Breunig, Hans-Peter Kriegel, et al. Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering[A]. *ACM SIGMOD* [C]. Santa Barbara, 2001: 99-112.

[3] Samer Nassar, Jorg Sander, Corrine Cheng. Incremental and Effective Data Summarization for Dynamic Hierarchical Clustering [A]. *ACM SIGMOD* [C]. Paris, 2004: 13-18.

[4] Guha S, Rastogi R, Shim K. CURE: An Efficient Clustering Algorithm for Large Databases [A]. *ACM Special Interest Group on Management of Data* [C]. Washington, 1998: 73-84.

[5] Zhang T, Ramakrishnan R, Livny M. BIRCH: An Efficient Data Clustering Method for Very Large Databases [A]. *ACM SIGMOD Int Conf on Management of Data* [C]. Montreal, 1996: 103-114.

[6] Ankerst M, Breunig M, Kriegel H, et al. OPTICS: Ordering Points to Identify the Clustering Structure [A]. *ACM Special Interest Group on Management of Data* [C]. Philadelphia, 1999: 49-60.

[7] Sander J. Density-based Clustering in Spatial Databases: The Algorithm GDBSCAN and It Applications [J]. *Data Mining and Konwledge Discovery*, 1998, 2(2): 169-194.

[8] Ester M, Kriegel H, Sander J. A Density-based Lgorithm for Discovering Clusters in Large Spatial Databases with Noise [A]. *Knowledge Discovery and Data Mining* [C]. Portland, 1996: 226-231.

[9] 王明善, 沈恒慈. *概率论与数理统计*[M]. 北京: 高等教育出版社, 1999.
(Wang M S, Shen H C. *Probability and Statistics*[M]. Beijing: Higher Education Press, 1999.)

(上接第973页)

[7] Gaing Z L. A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System [J]. *IEEE Trans on Energy Conversion*, 2004, 19(2): 384-391.

[8] 李宁, 邹彤, 孙德宝. 带时间窗车辆路径问题的粒子群算法[J]. *系统工程理论与实践*, 2004, 24(4): 130-135.
(Li N, Zou T, Sun D B. Particle Swarm Optimization for Vehicle Routing Problem with Time Windows [J]. *Systems Engineering — Theory and Practice*, 2004, 24(4): 130-135.)

[9] Daniel Q, Heinrich K. Production Planning in Semiconductor Assembly [A]. *Proc of the 4th Aegean Int Conf on Analysis of Manufacturing Systems* [C]. Samos Island, 2003: 181-189.