文章编号: 1001-0920 (2012) 07-0967-08

用于特征子集选择的异步并行微粒群优化方法

孔莉芳[1,2], 张 虹[1]

(1. 中国矿业大学 信息与电气工程学院，江苏 徐州 221116；2. 徐州空军学院 基础部，江苏 徐州 221002)

摘 要: 针对大量无关或冗余的特征通常会降低模式分类中分类器性能的问题, 提出一种基于异步并行微粒群优化的特征子集选择方法 (AP-PSO). 该方法采用二进制微粒群优化搜索特征子集, 利用异步并行方式提高算法的运算效率; 为有效协调种群的全局探索和局部开发能力, 充分利用混沌运动的遍历性和随机性, 提出一种一致混沌变异算子. 与已知 4 种特征子集选择方法进行比较, 所得结果验证了该算法的有效性.

关键词: 特征子集选择；微粒群优化；混沌变异；异步并行

中图分类号: TP183 **文献标识码:** A

Asynchronous parallel particle swarm optimizer for feature subset selection

KONG Li-fang[1,2], *ZHANG Hong*[1]

(1. School of Information and Electrical Engineering，China University of Mining and Technology，Xuzhou 221116, China；2. Department of Basic Teaching，Xuzhou Air Force College，Xuzhou 221002，China．Correspondent：KONG Li-fang，E-mail：klf030@163.com)

Abstract: In pattern classification system, many irrelevant and redundant features will lessen the performance of classifiers. Therefore, a feature subset selection method based on asynchronous parallel particle swarm optimization algorithm is proposed. This algorithm uses the binary particle swarm optimization to select feature subsets, and takes advantage of asynchronous parallel strategy to enhance time efficiency. In order to balance effectually the global exploration and the local exploitation of swarm, an uniform chaos mutation also is proposed by making the best use of the ergodicity, stochastic property and regularity of chaos. By compared with four known feature selection methods, the results show the effectiveness of the proposed algorithm.

Key words: feature subset selection；particle swarm optimization；chaos mutation；asynchronous parallel

1 引 言

特征子集选择是模式分类和数据挖掘领域的重要数据处理方法. 在某些模式分类系统中, 输入的数据对象往往含有大量的特征, 但是其中只有少部分特征与分类密切有关. 大量无关或冗余特征的存在, 一方面, 势必会大大增加系统学习及训练的时间; 另一方面, 可能导致分类或聚类精度的降低. 因此, 在对高维数据进行分类时, 需要运用特征选择算法找到具有较好可分性的特征子集, 实现降维, 从而达到降低机器学习的时间和空间复杂度的效果[1]. 简单而言, 特征子集选择是从一组 N 个特征中按一定的选择标准, 选择出一组由 $n(n < N)$ 个特征组成的特征子集. 该子集具有比特征全集更好或者和特征全集一样的分类功能[2].

处理特征子集选择问题时, 已有方法大体可以分为两类: 过滤算法和封装算法. 两类算法最大的区别是, 在应用分类器之前过滤算法已完成特征子集的选择. 利用不同特征的统计数据, 过滤算法从特征集合中不断删除不重要的特征. 除计算效率以外, 利用过滤算法时依然存在很多的争议. 例如, 某些过滤算法在利用通用的选择策略产生问题的特征子集时, 未考虑采用机器学习方法[3]. 封装算法是一类基于迭代搜索的特征选择方法, 在定义分类算法、性能测度和搜索策略之后, 该类算法采用迭代搜索策略在整个特征

收稿日期: 2011-01-12；**修回日期:** 2011-06-02.

基金项目: 国家自然科学基金项目(61005089)；高等学校博士学科点专项科研基金项目(20100095120016)；江苏省自然科学基金项目(BK2011215).

作者简介: 孔莉芳(1972−), 女, 讲师, 博士, 从事数据挖掘、故障诊断等研究；张虹(1942−), 女, 教授, 博士生导师, 从事图像处理、软件工程等研究.

空间中寻找优秀的特征子集. 每次迭代时, 首先由测试数据运行分类器计算当前特征子集的性能指标值; 基于所得性能指标值, 由指定的搜索策略产生新的特征子集. 由于特征子集的选择和机器学习同时执行, 封装算法所产生分类模型的预测效果及稳定性通常优于过滤算法[4]. 然而, 当采用封装算法时, 反复的机器学习势必需要花费大量的运算时间. 为了降低算法的计算复杂度, 很多学者尝试采用启发式搜索算法寻找问题的最佳特征子集, 代表方法如遗传算法、模拟退火和禁忌搜索等[5-7].

微粒群优化[8](PSO) 是一种启发式搜索算法, 其思想源于对鸟类觅食等复杂群体行为的模拟. 由于具有概念简明、方便实现和收敛速度快等优点, 该算法已在很多领域得到了应用. 将微粒群优化算法用于处理特征子集选择问题, 本文提出一种基于一致混沌变异的异步并行微粒群优化. 与已有相关研究相比, 该算法具有如下特点: 1) 采用异步并行方式来评价微粒的适应值和更新微粒的位置, 可以提高处理器的利用效率. 相比之下, 朱颢东等[9]提出的并行二进制免疫量子微粒群优化, 采用了同步并行方式更新微粒的适应值, 在问题较复杂的情况下, 同步并行方式很难有效均衡从处理器间的计算负载; 2) 提出一种一致混沌变异算子, 协调种群的全局搜索和局部开发能力. 高雷阜等[10]利用混沌序列初始化微粒的位置和速度, 用以改善种群的多样性. 为了防止种群陷入局部极值点, Leandro[11]给出了基于Zaslavskii混沌映射的混沌变异算子. 该算法中, 微粒变异的概率和微粒位置的变异范围皆是固定不变的. 相比之下, 本文所提出的一致混沌变异算子可以同时调节微粒变异的概率及其变异范围. 此外, 为充分利用准确率较低但特征数目较少的优秀微粒, 本文还提出了一种用于比较微粒优劣的ε支配关系.

2 相关知识

2.1 基本微粒群优化算法

在鸟类捕食的群体行为中, 每只鸟被命名为一个微粒, 每个微粒代表一个被优化问题的解. 这些微粒各自以一种特有的速度飞行, 穿过多维搜索空间, 最终找到最优解. 在n维目标空间中, 设微粒$\boldsymbol{x}_i$本身所找到的最佳位置为$\boldsymbol{p}_i = (p_{i1}, p_{i2}, \cdots, p_{in})$(一般称为微粒的个体最优点或局部引导者), 整个微粒群迄今为止搜索到的最佳位置为$\boldsymbol{p}_g = (p_{g1}, p_{g2}, \cdots, p_{gn})$(一般称为微粒群的全局最优点或全局引导者), 微粒的当前速度为$\boldsymbol{v}_i = (v_{i1}, v_{i2}, \cdots, v_{in})$, 则每次迭代时微粒根据如下公式[12]来调整自己位置:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1r_1(p_{ij}(t) - x_{ij}(t)) + c_2r_2(p_{gj}(t) - x_{ij}(t)), \tag{1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{2}$$

其中: t为迭代次数; c_1和c_2为学习因子; w为惯性权值, 用来协调微粒的全局搜索和局部搜索能力; r_1和r_2为[0,1]之间随机数.

2.2 二进制微粒群优化算法

最初的PSO算法是从处理连续优化问题发展起来的. 针对生活中常见的离散问题(如组合优化问题或者路径寻优问题), Kennedy等[13]首次将实数版本的PSO扩展为二进制PSO(BPSO). 在二进制PSO中, 微粒$\boldsymbol{x}_i$的每个分量x_{ij}被限制在$\{0,1\}$中取值. 每次更新微粒位置时, 由逻辑函数$s(v)$将式(1)得到的微粒速度$\boldsymbol{v}_i$转化为判断x_{ij}选择0还是1的概率, 具体转换公式如下:

$$x_{ij} = \begin{cases} 0, & s(v_{ij}) < U(0,1); \\ 1, & \text{otherwise}; \end{cases} \tag{3}$$

$$s(v_{ij}) = 1/(1 + \mathrm{e}^{-v_{ij}}).$$

除位置更新公式外, 其他操作与基本PSO相似. 为了防止$s(v)$函数饱和, Kennedy等建议将v_{ij}限制在$[-4,4]$之间.

2.3 用于特征子集选择的微粒群优化方法

近年来, 利用微粒群优化解决特征子集选择问题得到了学者们的重视. Wang等[14]提出了一种结合PSO和粗糙集的最优特征子集选择方法. 相比基于粗糙集的特征选择方法, 结果表明PSO算法是有效的. 乔立岩等[2]提出了一种基于离散微粒群优化和支持向量机封装模式的特征子集选择方法, 该算法利用离散微粒群优化算法对特征进行优化, 并采用支持向量机的10阶交叉验证结果指导算法的搜索. 同样, Garcia-Nieto等[15]研究基于支持向量机和微粒群优化的封装类特征选择方法, 给出了一种称为几何PSO的改进算法. 郭文忠等[16]提出了基于离散微粒群和相关性分析的特征子集选择算法, 该算法通过分析数据中所有特征之间的相关性, 利用离散微粒群算法在整个空间中进行搜索, 自动选择有效的特征子集. 针对两元聚类问题, Alper等[3]提出了一种改进的离散微粒群优化算法. 在选择特征子集包含特征时, 该算法充分考虑了待选特征与特征子集之间的相关性和独立性. 最近, 为了提高文本挖掘算法的运行速度, 降低占用的内存空间, 朱颢东等[9]提出了一种基于并行二进制免疫量子微粒群优化的特征子集选择方法. 该方法采用二进制免疫量子微粒群优化搜索特征子集, 利用同步并行算法提高时间效率.

2.4 并行微粒群优化算法

并行微粒群优化算法(PPSO)将计算机的高速并行计算能力和微粒群优化算法的天然并行性相结合, 极大提高了PSO解决复杂优化问题的能力. 与并行遗传算法类似, PPSO算法可分为3类[17]: 主从式PPSO(或全局式模型)、迁移式PPSO(或粗粒度模型)和扩散式PPSO(或细粒度模型). 限于篇幅, 本节只讨论主从式PPSO.

在主从式PPSO算法中, 主处理器负责执行微粒全局最优点和微粒位置的更新等全局操作, 执行方式为串行; 从处理器负责执行微粒适应值的评估等局部操作, 各从处理器之间并行运行. 从处理器在执行局部操作时, 现有技术大都采用了同步并行的方式: 所有从处理器同时计算所分配微粒的适应值; 当接收到所有从处理器的返回信息后, 主处理器才进行下一步操作. 由于未对串行微粒群优化算法的框架进行改动, 上述并行方式不可避免地存在主从处理器负荷不均衡的问题. 而且, 一次局部操作完成后, 从处理器都要向主处理器发送结果, 这易于造成大的通信延迟[18]. 为了克服上述缺陷, Byung-I1 等[19]给出了微粒群优化算法的异步并行执行方式, 其思想如下: 从处理器计算完微粒适应值之后立即将结果传输给主处理器; 主处理器接收到信息后, 随即更新该微粒的位置, 同时分配新的待评价微粒给空闲从处理器.

3 用于特征子集选择的异步并行微粒群优化

本节详细介绍用于特征子集选择的异步并行微粒群优化算法. 首先, 初始化每个微粒为特征选择问题的一个潜在特征子集; 然后基于包含若干特征子集的微粒群, 采用改进的异步并行微粒群算法进行寻优搜索, 并应用支持向量机[20](SVM)对所选特征子集的分类能力(即适应值)进行评估; 以此类推, 直到求得最优特征子集或者满足终止条件.

3.1 微粒编码

根据特征选择问题的特点, 把每一个特征看作微粒的一维二进制变量, 微粒变量的长度等于所有特征的数量. 如果第 i 位为 1, 则第 i 个特征被选中; 否则, 这个特征被屏蔽. 比如, 微粒 $\boldsymbol{x}_i = (1,0,0,0,1,0)$ 表明特征 1 和特征 5 被选中, 因此, 特征子集为 $\{1, 5\}$. 由此, 每一个微粒代表一个不同的特征子集, 也就是问题的一个候选解.

3.2 微粒适应值的比较

特征子集选择的目标是使用少量的特征达到相同或更好的分类效果, 因此微粒适应值的评价应该包含两部分内容: 1) 所确定分类器的准确率, 记为 η. 利用特征子集所确定的特征来训练SVM分类器, 本文采用10阶交叉验证的结果作为分类器的准确率, 并以此性能指标来指导微粒群的搜索. 2) 所包含的特征数量, 记为 λ. 在准确率和特征数量这两个因素中, 需要重点考虑的是准确率.

每个特征子集包含一定数量的特征, 如果两个特征子集获得的准确度相同, 则包含特征较少的子集便会胜出. 然而, 对于一些准确率较低(相对最优值)但特征数目少的特征子集, 一方面, 他们可能已经非常接近最优特征子集(两者之间仅有一个特征不同), 进一步利用它们将会增加算法找到优秀特征子集的可能性; 另一方面, 尽管该类特征子集的准确率略差, 但是, 相对准确率较高的特征子集, 其包含的特征数目明显减少. 鉴于特征数据提取的代价和分类器构建的复杂性, 在分类准确率允许的情况下, 选择上述特征子集作为问题的最终结果更具经济价值. 因此, 算法迭代过程中应该保存小部分准确率较低但特征数目较少的微粒.

基于此, 本文给出一种用于微粒优劣比较的 ε 支配关系. 对于给定的常数 $\varepsilon(\varepsilon > 0)$ 和任意两个微粒 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$, 如果满足如下条件之一: 1) $\lambda(\boldsymbol{x}_i) < \lambda(\boldsymbol{x}_j)$ 且 $\eta(\boldsymbol{x}_i) \geqslant \eta(\boldsymbol{x}_j) - \varepsilon$; 2) $\lambda(\boldsymbol{x}_i) = \lambda(\boldsymbol{x}_j)$ 且 $\eta(\boldsymbol{x}_i) > \eta(\boldsymbol{x}_j)$; 则称 $\boldsymbol{x}_i$ 支配 $\vec{\boldsymbol{x}}_j$. 特别地, 当 $\lambda(\boldsymbol{x}_i) = \lambda(\boldsymbol{x}_j)$ 且 $\eta(\boldsymbol{x}_i) = \eta(\boldsymbol{x}_j)$ 时, 称 $\boldsymbol{x}_i$ 等价于 $\boldsymbol{x}_j$. 在此, 常数 ε 反映了决策者对分类器准确度的要求. 决策者对分类器的准确度要求越高, ε 取值越小.

3.3 微粒引导者的更新

微粒引导者包括微粒全局最优点和个体最优点. 本文采用 ε 支配关系更新上述两个最优点. 微粒个体最优点是指微粒从初始到目前迭代次数所得到的最好位置, 它可以看作是微粒的记忆. 设微粒 $\boldsymbol{x}_i(t)$ 的个体最优点为 $\boldsymbol{p}_i(t)$, 第 $t+1$ 代新生微粒为 $\boldsymbol{x}_i(t+1)$, 则微粒个体最优点的更新方法如下: 当 $\boldsymbol{x}_i(t+1)\varepsilon$ 支配 $\boldsymbol{p}_i(t)$ 时, 个体最优点 $\boldsymbol{p}_i(t+1)$ 取为 $\vec{\boldsymbol{x}}_i(t+1)$; 当 $\boldsymbol{x}_i(t+1)$ 和 $\boldsymbol{p}_i(t)$ 等价时, 从 $\boldsymbol{p}_i(t)$ 和 $\boldsymbol{x}_i(t+1)$ 中随机选择一个作为微粒的个体最优点; 否则, $\boldsymbol{p}_i(t+1)$ 取 $\boldsymbol{p}_i(t)$.

全局最优点是整个微粒群迄今为止搜索到的最佳位置. 本文利用 ε 支配关系两两比较所有微粒的个体最优点, 最后胜出者为微粒群的全局最优点.

3.4 一致混沌变异

众所周知, 微粒群优化算法具有较快的收敛速度. 然而, 正是因为它的快速收敛性, 使得PSO算法易陷入局部收敛[21]. 混沌是一种非线性现象, 具有遍历性、随机性和对初始条件的敏感性等特性. 它可在一定范围内按自身规律不重复地遍历所有状态, 因此可

将其应用到优化算法, 以提高算法的全局搜索能力.

基于此, 本文提出一种一致混沌变异算子. 首先, 采用Zaslavskii映射函数[22]产生一混沌变量; 对所产生的混沌变量进行变换后(由混沌空间转化到所处理问题的决策空间), 接着对微粒的飞行速度进行变异. 所提变异算子的步骤如下:

MUTATE

Bound为速度的上下界, n为决策变量的维数, $T_{\max}$为算法终止代数*//

For i=1 to S //* S为微粒群规模*//

If $p_m = \mathrm{e}^{(-2t/T_{\max})} > r$

dim=rand(1,n)

range=(upper_Bound−low_Bound)$*p_m$

$y(t) = \mathrm{mod}[y(t-1) + v + az(t), 1]$

$z(t) = \cos(2\pi y(t-1) + e^{-r} z(t-1))$

$v_{i,\mathrm{dim}} = v_{i,\mathrm{dim}} + z(t)*$range

EndIf

EndFor

为使Zaslavskii映射展示出完全的混沌状态, 本文取$v = 400, r = 3, a = 12.669\,5$. 进一步, 图1展示了Zaslavskii映射产生的混沌状态. 可以看出, 在算法初期阶段, 所有微粒的飞行速度将受变异算子的影响. 在该速度的作用下, 算法将允许每个微粒在整个决策空间中取值. 因此, 在初始阶段算法具有好的全局探索能力. 随着迭代次数的增加, 变异算子的影响逐渐变弱, 这不仅反映在变异概率的减小, 也表现在逐渐收缩的变异范围. 在迭代后期, 算法将具有好的局部开发能力. 注意, 当变异后的微粒速度超出所允许范围时, 取其对应速度的边界值(即该速度变量的上界或下界)作为它的新速度.

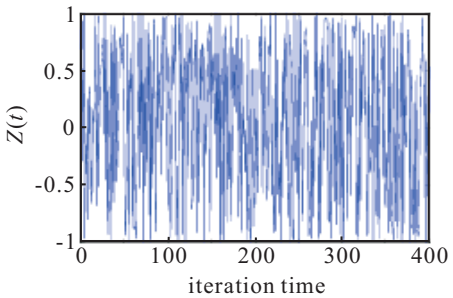

图1 $v = 400, r = 3, a = 12.669\,5$时, Zaslavskii产生的混沌状态

3.5 异步并行微粒群优化算法的步骤

在串行算法中, 循环计算每个微粒的适应值是最耗时的部分. 本文采用异步并行方式来降低微粒群优化算法的运行时间, 从而尽快获得问题的理想特征子集. 在该并行计算中, 对循环进行了分解, 把需要计算适应值的微粒通过消息发送给多个处理器, 并行计算它们的适应值; 主处理器接收到从处理器的反馈信息后, 继续进行算法的主操作. 具体思想如下: 首先, 在决策空间中随机初始化微粒群; 其次, 主处理器将待评价微粒$\boldsymbol{x}_i$及其个体最优点传递给空闲的从处理器; 接收到任务后, 从处理器计算该微粒所确定分类器的准确率, 由ε支配关系更新其个体最优点$\boldsymbol{p}_i$, 并将更新后的$\boldsymbol{p}_i$反馈给主处理器; 主处理器接收到从处理器反馈的信息后, 根据此反馈信息对决策进行评价, 确定出当前微粒群的全局最优点; 随后, 由式(1)和(3), 以及一致混沌变异算子更新微粒的位置. 主处理器按次序选择下一个待评价微粒, 并将其信息传递给空闲从处理器. 依次类推, 直到得到满意解.

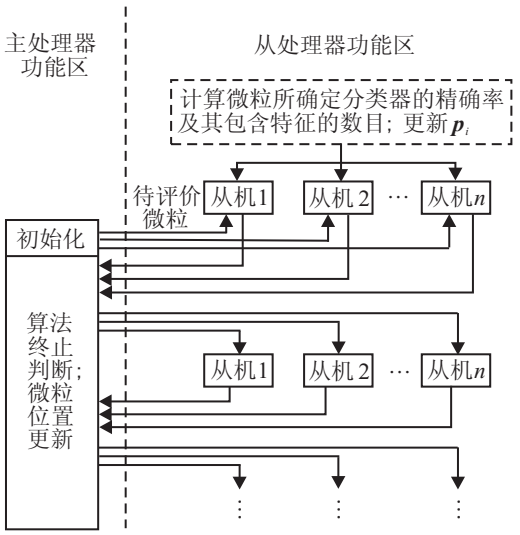

图2 一并行异步微粒群优化算法的结构图

图2给出了异步并行微粒群优化算法的结构图, 算法具体实现主要包括主处理器功能设定和从处理器功能设定两部分.

主处理器实现步骤:

1) 初始化系统优化所需参数、微粒位置和速度, 设定算法终止条件. 系统优化所需参数包括微粒群规模、惯性权值和学习因子等控制参数; 设置迭代次数$t = 1$.

2) 对待评价微粒的处理次序进行排序. 本文采用先进先出的排序方法决定微粒分配到从处理器上的次序, 即先提交给主处理器的从处理器计算结果先被处理.

3) 依照次序表中顺序, 依次发送待评价微粒及其个体最优点到当前空闲的从处理器, 并接收从处理器的反馈信息, 即更新后的个体最优点$\boldsymbol{p}_i$.

4) 基于当前个体最优点的集合, 由3.3节中方法确定当前微粒群的全局最优点.

5) 判断算法终止条件, 如果满足, 则终止算法, 并输出结果$\boldsymbol{p}_g$.

6) 由式 (1) 和 (3), 以及一致混沌变异算子更新微粒的位置.

7) 由 $t=\lceil En/|S|\rceil$ 更新算法迭代次数, 并转入 2), 其中 En 为评价微粒的总次数, $|S|$ 为微粒规模, $\lceil\bullet\rceil$ 为向上取整算子.

从处理器实现步骤:

1) 接收主处理器发送的待评价微粒.

2) 利用微粒确定的特征训练 SVM 分类器, 并采用 10 阶交叉验证的结果作为分类器的准确率.

3) 由 ε 支配关系更新其个体最优点 $\boldsymbol{p}_i$, 并向主处理器传送计算结果.

区别于同步并行微粒群优化算法, 异步并行微粒群优化算法的优点是: 已被评价的微粒无需等待其他微粒评价过程的完成, 即可进行下一次迭代. 该方式有效地保证了各从处理器之间的负载平衡.

4　实验及性能分析

测试本文特征子集选择方法 AP-PSO 的性能, 设计 2 组实验: 1) 选取包括遗传算法和微粒群优化在内的 4 种典型方法作为对比算法, 验证 AP-PSO 的优化性能; 2) 测试算法的并行效率, 验证 AP-PSO 算法并行设计的可行性.

4.1　实验设置

为了验证本文特征子集选择算法的有效性, 该实验采用 UCI 机器学习数据库[23]. 所选测试问题包括 Class 和 Sonar 等 10 个数据集, 表 1 为它们的相关详细信息. 其中 a/b 分别表示测试和训练数据的规模.

表 1　所用分类问题

测试数据集	类数目	总特征数目	样本数目
Class	7	10	214
Vowel	11	10	990
Wine	3	13	178
Letter	26	16	20 000
Vehicle	4	18	846
Segmentation	7	19	210/2 100
WDBC	2	30	569
Ionosphere	2	34	351
Satellite	6	36	4 435/2 000
Sonar	2	60	208

选用前向选择法 (SFS)[23], BPSO 算法, 混杂遗传算法 (HGA)[24]和文献 [3] 中改进离散微粒群优化算法 (MDPSO) 作为比较算法. 其中, HGA 算法取波动因子为 2 的情况. 为公平比较, 设置 BPSO, MD-PSO 和 AP-PSO 的种群规模为 20; SFS 算法的终止代数为 10 000, 其他 4 种算法的终止代数为 200. BPSO 中设置 $c_1=c_2=2$, 惯性权值 $w=0.48$; MD-PSO 和 AP-PSO 中设置 $c_1=2$, $c_2=1.5$, 惯性权值采用线性递减策略 ($w_{\max}=0.995$, $w_{\min}=0.5$). 此外, 所有实验中 AP-PSO 设置 $\varepsilon=0.005$.

本文采用 MPICH 并行计算环境, 在基于 Linux 操作系统的 Dell 高性能计算集群上并行运算 AP-PSO 算法. 其他比较算法选择在普通 PC 机上运行, 配置如下: 因特尔 CPU(2.60 GHz), 2 G 内存和 250 M 硬盘.

4.2　优化性能分析

在相同硬件条件下利用 5 种算法分别优化表 1 中问题 20 次, 表 2 给出了所得关于特征数目和分类器准确率的平均结果. 其中 d^* 为运行 SFS 和 HGA 时指定的特征数目; d^{**} 为 PSO 算法所得最优结果的平均特征数目. 采用文献 [24] 中建议, 取 d^* 为如下 4 个值: $N/5, 2N/5, 3N/5$ 和 $4N/5$, 其中 N 为测试问题的总特征数目. 每个测试问题的最优特征子集未知, 而 d^* 取值过细又会显著增加算法的运行时间, 因此 d^* 取上述 4 个值是可以接受的. 不可否认的是, 如果最优特征子集所包含的特征数目不在 $\{N/5, 2N/5, 3N/5, 4N/5\}$ 中, SFS 和 HGA 将没有机会得到最优特征子集.

由表 2 可以看出: 1) 对于所有测试问题, MDPSO 和 AP-PSO 获得了优于 BPSO 的平均准确率, 而 HGA 具有好于 SFS 的平均准确率; 2) 除总特征数目较少的 Class 和 Vowel 问题之外, 对于其他 8 个测试问题, MDPSO 和 AP-PSO 均获得了优于其他 3 种算法的分类准确率. 对于 Class 问题, 5 种算法皆获得了分类准确率为 100%, 特征数目为 3 的最优结果; 对于 Vowel 问题, SFS 和 HGA 得到了分类准确率为99.7%且特征数目为3的最优结果; 3) 对于测试问题 Wine,Vehicle, Segmentation 和 WDBC, BPSO 具有优于 HGA 的平均分类准确率; 对于测试问题 Vowel, Letter, Ionosphere, Satellite 和 Sonar, HGA 所得平均分类准确率又好于 BPSO 所得平均分类准确率. 4) 对于测试问题 Wine, Vehicle, WDBC, Ionosphere 和 Sonar, 尽管 MDPSO 得到略好于 AP-PSO 的平均分类准确率, 但其结果所包含特征的数目要明显多于 AP-PSO (除 Wine 外). 对于测试问题 Vowel, Letter, Segmentation 和 Satellite, AP-PSO 得到了分类准确率和特征数目皆优于 MDPSO 的结果.

综述所述, 对于总特征数目较少的测试问题 Class 和 Vowel, HGA 和 SFS 可取得较好的平均优化性能; 对于总特征数目相对较多的测试问题, MDPSO 和 AP-PSO 可以获得优于其他 3 种算法的平均优化性能. 而在考虑特征数目的情况下, AP-PSO 的总体性能要好于 MDPSO 的总体性能.

表 2 5种算法处理测试数据时所得结果

数据集	d^*	SFS	HGA	BPSO/%		MDPSO/%		AP-PSO/%	
				d^{**}	准确率	d^{**}	准确率	d^{**}	准确率
class	2	99.07	99.07	**3.0**	**100**	**3.0**	**100**	**3.0**	**100**
	4	**100**	**100**						
	6	**100**	**100**						
	8	**100**	**100**						
Vowel	2	62.02	62.02	8.94	99.49	8.87	99.59	8.25	99.64
	4	92.63	92.83						
	6	98.82	98.79						
	8	**99.70**	**99.70**						
Wine	3	93.82	93.82	8.06	98.78	7.85	**99.45**	7.70	99.42
	5	94.38	95.51						
	8	95.51	95.51						
	10	92.13	92.70						
Letter	3	47.09	47.09	15.10	95.38	15.05	96.46	13.80	**96.59**
	6	86.20	87.60						
	10	96.12	96.35						
	13	96.42	96.42						
Vehicle	4	62.77	69.74	11.56	74.70	12.24	**75.02**	11.33	74.86
	7	69.15	73.52						
	11	69.50	72.46						
	14	68.20	70.80						
Segmentation	4	92.81	92.81	11.20	97.58	10.44	97.90	10.12	**98.11**
	8	92.95	92.95						
	11	92.95	92.95						
	15	92.57	92.57						
WDBC	6	93.15	94.38	13.87	97.62	15.05	**97.75**	12.62	97.70
	12	92.26	94.06						
	18	94.02	93.99						
	24	92.44	93.58						
Ionosphere	7	93.45	95.50	12.20	93.73	14.25	**96.12**	12.30	95.84
	14	90.88	95.56						
	20	90.03	94.19						
	27	89.17	91.45						
Satellite	7	86.85	99.44	17.52	90.80	25.27	91.45	22.50	**91.68**
	14	89.45	90.87						
	22	90.45	91.44						
	29	90.40	91.24						
Sonar	12	87.02	94.71	30.80	92.90	27.70	**96.08**	24.58	95.98
	24	89.90	95.96						
	36	88.46	95.82						
	48	91.82	93.17						

4.3 并行性能分析

本小节采用Satellite数据集分析AP-PSO算法的并行性能. 为了评价算法的并行效果, 采用如下3个常用的性能测度[25]: 1) 加速比 $Sp = Ts/Tp$, Ts 是串行算法在最坏情况下的运行时间, Tp 是并行算法在最坏情况下的运行时间, 加速比可以反应算法的并行性对运行时间的改进程度. 2) 效率 $Ep = Sp/p$, p 为处理器的个数, 该测度反映了并行系统中从处理器的利用情况. 3) 串行份额

$$F_p = \frac{1/S_p - 1/p}{1 - 1/p}. \tag{4}$$

其中 F_P 是描述并行算法的扩展性的一个指示器. 随着处理器数目的增加, 如果 F_P 值也不断增加, 则意味处理器间通讯花费的增加, 此时并行算法的扩展性相对较差.

表3给出了同步和异步并行两种方式下本文方法所得性能测度值. 需要说明的是, 当处理器数为16时, 实验还考虑了 $15+1$ 的情况, 即存在一个计算性能较差的从处理器, 实验选择第15个处理器作为上述从处理器. 在不失实验本质的情况下, 作如下处理: 该处理器在完成所分配的计算任务后, 需要等待一段时间才将结果反馈给主处理器. 显然, 等待时间越长, 该处理器表现出的处理速度越慢.

由表3可以看出: 1) 随着从处理器数目的增加, 两种并行方式下算法的运行速度皆有明显提高. 但相

表 3　不同从处理器数下算法的运行时间和性能测度

处理器数	同步并行方式				异步并行方式			
	运行时间	加速比	效率	串行份额	运行时间	加速比	效率	串行份额
1	356	1.00	1.00	–	356	1.00	1.00	–
2	190	1.87	0.93	0.069 5	185	1.91	0.95	0.047 1
4	101	3.50	0.87	0.047 6	98	3.62	0.90	0.035 0
8	59	6.03	0.75	0.046 7	54	6.59	0.82	0.030 6
12	38	9.36	0.78	0.025 6	35	10.0	0.83	0.018 2
16	33	10.8	0.68	0.032 1	27.9	12.7	0.79	0.017 3
16(15+1)	46	7.7	0.48	0.071 9	28.7	12.4	0.78	0.019 4

比之下, 异步并行方式所得加速比更为优秀. 2) 对于并行效率而言, 不同从处理器数目下异步并行方式皆得到了优于同步并行方式的评价结果. 特别地, 当从处理器数目大于 8 时, 异步并行方式使得算法的并行效率得到了明显改善. 而当从处理器数目为 8 和 12 时, 同步并行方法的效率值下降到 75% 左右; 当从处理器数目为 16 时, 其效率值最低, 仅为 68%. 产生这种情况的原因在于, 当从处理器数目不是微粒群规模约数时, 从处理器的处理速度和工作量不同, 主处理器必须等待最差处理器结束工作后才能运作, 因此难以均衡从处理器之间的计算负荷. 相反, 对于异步并行方式而言, 由于能够充分利用空闲和处理速度快的处理器, 算法的并行效率得到了明显改善. 3) 对于同步并行方式, 当从处理器由 2 增加到 12 时, 其串行份额值逐渐变小; 而当从处理器数为 16 时, 其值略有反弹. 这说明, 当从处理器数大于 16 时, 数据传输的花费在整个时间花费中所占比值变大. 对于异步方式而言, 随着从处理器数目的增加, 其串行份额值始终在减小. 因此, 异步并行方式的扩展性好于同步并行方式. 4) 相对同步并行方式, 异步并行方式受单个处理器性能优劣的影响较小. 如表 3 所示, 当 16 个处理器中存在一个性能较差的处理器时, 同步并行方式的运行时间明显增大, 而异步并行方式的运行时间变化较小.

5　结　论

为解决特征子集选择问题, 本文提出了一种基于混沌变异的异步并行微粒群优化算法. 不同于已有混沌变异算子, 所提出的一致混沌变异算子可以同时调节微粒的变异概率和变异范围, 有效均衡了微粒的全局搜索和局部开发能力. 在保证分类准确率的情况下, 给出了一种用于比较微粒优劣的 ε 支配关系, 显著降低了优化结果所包含特征的数目. 为提高空闲和高速度处理器的利用效率, 设计了微粒群优化算法的异步并行方法, 有效提高了算法的并行效率. 通过与 4 种已知的特征子集选择方法进行比较, 结果表明本文算法可以提供高竞争力的最优特征子集. 利用半监督学习等方法来估计微粒的适应值, 提高算法的计算效率是作者将来的研究内容.

参考文献(References)

[1] George H J, Ron K, Karl P. Irrelevant features and the subset selection problem[C]. Proc of the Eleventh Int Conf on Machine learning. Piscataway: IEEE Press, 1994: 121-129.

[2] 乔立岩, 彭喜元, 彭宇. 基于微粒群算法和支持向量机的特征子集选择方法[J]. 电子学报, 2006, 34(3): 496-498.
(Qiao L Y, Peng X Y, Peng Y. BPSO-SVM wrapper for feature subset selection[J]. Acta Electronica Sinica, 2006, 34(2): 496-498.)

[3] Alper U, Alper M. A discrete particle swarm optimization method for feature selection in binary classification problems[J]. European J of Operational Research, 2010, 206(3): 528-539.

[4] Kohavi R, John G. Wrappers for feature subset selection[J]. Artificial Intelligence – Special Issue on Relevance, 1997, 97(1/2): 273-324.

[5] Li Y, Zhang S, Zeng X. Research of multi-population agent genetic algorithm for feature selection[J]. Expert Systems with Applications, 2009, 36(9): 11570-11581.

[6] 张昊, 陶然, 李志勇, 等. 基于自适应模拟退火遗传算法的特征选择方法[J]. 兵工学报，2009, 30(1): 81-85.
(Zhang H, Tao R, Li Z Y, et al. A feature selection method based on adaptive simulated annealing genetic algorithm[J]. Acta Armamentarii, 2009, 30(1): 81-85.)

[7] Pacheco J, Casado S, Nez L. A variable selection method based on tabu search for logistic regression models[J]. European J of Operational Research, 2009, 199(2): 506-511.

[8] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway: IEEE Press, 1995: 1942-1948.

[9] 朱颢东, 钟勇. 基于并行二进制免疫量子粒子群优化的特征选择方法[J]. 控制与决策, 2010, 25(1): 53-58.
(Zhu H D, Zhong Y. Feature selection method based on PBIQPSO[J]. Control and Decision, 2010, 25(1): 53-58.)

[10] 高雷阜, 刘旭旺. 基于混沌的弹性粒子群全局优化算法[J]. 控制与决策, 2009, 24(10): 1545-1548.
(Gao L F, Liu X W. Resilient particle swarm global optimization algorithm based on chaos[J]. Control and Decision, 2009, 24(10): 1545-1548.)

[11] Leandro dos S C. A quantum particle swarm optimizer with chaotic mutation operator[J]. Chaos, Solitons and Fractals, 2008, 37(5): 1409-1418.

[12] Shi Y, Eberhart R C. A modified particle swarm optimizer[C]. Proc of the IEEE Congress on Evolutionary Computation. Piscataway: IEEE Press, 1998: 303-308.

[13] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]. Proc of the 1997 Conf on Systems, Man, and Cybernetics. Piscataway: IEEE Press, 1997: 4104-4109.

[14] Wang X, Yang J, Teng X, et.al. Feature selection based on rough sets and particle swarm optimization[J]. Pattern Recognition Letters, 2007, 28(4): 459-471.

[15] García-Nieto J, Alba E, Jourdan L, et al. A comparison of PSO and GA approaches for gene selection and classification of microarray data[C]. Proc of Genetic and Evolutionary Computation Conf. London: ACM Press, 2007: 427.

[16] 郭文忠, 陈国龙, 陈庆良, 等. 基于粒子群优化算法和相关性分析的特征子集选择[J]. 计算机科学, 2008, 35(2): 144-146.
(Guo W Z, Chen G L, Chen Q L, et al. Feature subset selection based on particle swarm optimization algorithm and relevance analysis[J]. Computer Science, 2008, 35(2): 144-146.)

[17] Belal M, EI-Ghazawi T. Parallel models for particle swarm optimizers[J]. The Int J of Intelligent Computing and Information Sciences, 2004, 4(1): 100-110.

[18] 郭彤城, 慕春棣. 并行遗传算法的新进展[J]. 系统工程理论与实践, 2002, 22(2): 15-23.
(Guo T C, Mu C D. The parallel drifts of genetic algorithms[J]. Systems Engineering-theory and Practice, 2002, 22(2): 15-23.)

[19] Byung-I1 Koh, Alan D G, Raphael T H, et al. Parallel asynchronous particle swarm optimization[J]. Int J for Numerical Methods Engineering, 2006, 67(4): 578-595.

[20] Burges C J. A tutorial on support vector machines for pattern recognition[J]. Data Mining and Knowledge Discovery, 1998, 2(2): 121-167.

[21] Coello Coello A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 256-279.

[22] Zaslavskii G M. The simplest case of a strange attractor[J]. Physics Letters A, 1978, 69(3): 145-147.

[23] Whitney A. A direct method of nonparametric measurement selection[J]. IEEE Trans on Computers, 1971, 20(9): 1100-1103.

[24] Il-Seok O, Jin-Seon Lee, Byung-Ro Moon. Hybrid genetic algorithms for feature selection[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2004, 26(11): 1424-1437.

[25] Jaimes A L, Coello Coello C A. MRMOGA: A new parallel multi-objective evolutionary algorithm based on the use of multiple resolutions[J]. Concurrency and Computation: Practice and Experience, 2007, 19(4): 397-441.

(上接第966页)

[61] Wu H, Xu Y P, Shi F L,et al. Two-dimensional path planning based on improved estimation of distribution algorithm[J]. Computer Engineering, 2010, 36(16): 180-182.

[62] Zhou S D. Decision making for military distribution center location based on estimation of distribution algorithms[J]. J of China Academy of Electronics and Information Technology, 2010, 5(5): 532-536.

[63] 王凌, 王圣尧, 方晨. 一种求解多维背包问题的混合分布估计算法[J]. 控制与决策, 2011, 26(8): 1121-1125.
(Wang L, Wang S Y, Fang C. A hybrid distribution estimation algorithm for solving multidimensional knapsack problem[J]. Control and Decision, 2011, 26(8): 1121-1125.)

[64] Bengoetxea E, Larranaga P, Bielza C, et al. Optimal row and column ordering to mprove table interpretation using estimation of distribution algorithms[J]. J of Heuristics, 2011, 17(5): 567-588.

[65] Santana R, Bielza C, Larranaga P, et al. Mateda-2.0: Estimation of distribution algorithms in Matlab[J]. J of Statistical Software, 2010, 35(7): 1-30.