一种求解多处理机调度问题的α-平坦化调度算法

魏嘉银 秦永彬 许道云

(贵州大学计算机科学与信息学院 贵阳 550025)

摘 要 在分析多处理机调度问题的基础上,提出了α-平坦的概念,并将其引入到多处理机调度问题中;基于此,提出了一种新的基于α-平坦的求解多处理机调度问题的算法。算法首先对作业集合做平坦化处理,然后再对处理后所得的新问题进行求解,最终获得原调度问题的一个近似解。实验结果表明,通过该算法可以求得较好的结果,相对于其它启发式算法,该算法具有较好的稳定性。

关键词 多处理机调度,α-平坦,平坦化处理,调度算法

中图法分类号 TP393 **文献标识码** A

Scheduling Algorithm of α-Planarization for Solving the Problem of Multiprocessor Scheduling

WEI Jia-yin QIN Yong-bin XU Dao-yun

(College of Computer Science and Information, Guizhou University, Guiyang 550025, China)

Abstract In this paper, the concept of α-flatness was proposed based on analyzing the multiprocessors scheduling problem, and then introduced it into the multiprocessors scheduling problem. Finally, a new algorithm based on the concept of α- flatness was proposed to solved the multiprocessors scheduling problem. In this algorithm, the job set was flatting at first, and then solved the new problem obtained by the first step, finally an approximate solution for the original scheduling problem was obtained. The experimental results show that the solution obtained by this algorithm is good, and compared with the heuristic algorithm the result obtained by this algorithm is more stable.

Keywords Multiprocessors scheduling, α-flatness, Planarization processing, Scheduling algorithm

1 引言

随着工业化和现代科学技术的迅速发展,各领域的专家学者正日益重视和研究一类调度问题——多处理机调度问题(Multiprocessors Scheduling Problem, MSP)。这是一类重要的组合优化问题,其关注点是如何将一批既定任务分配给多个处理机处理,以使得从第一个任务开始执行到最后一个任务执行完毕所用的时间最少。多处理机调度问题已经被证明是NP完全的[1],难以用线性规划和动态规划等传统的优化方法来解决。因此,近年来国内外相关领域就此类问题所设计的求解算法大多属于近似、启发式算法,以达到在尽可能满足问题求解要求的同时降低问题求解的复杂度[2-8],其中较为典型的有蚁群算法[4,5]、启发式算法[6]和粒子群优化算法[7,8]等。但是随着研究的深入,发现这些算法均存在一定的缺陷,如蚁群算法存在算法运行时间较长,并且容易陷入局部最优等问题;而粒子群优化算法则存在对于离散的优化问题处理不佳,容易陷入局部最优等不足之处。那么是否能够找到一种既能较好地避免上述这些算法所存在的不足,又能较好地求解多处理机调度问题的算法呢?

本文在研究作业调度问题的基础上,提出了α-平坦的概念,并将它应用到多处理机调度问题求解算法的设计中。基于这一概念,提出了一种全新的多处理机调度算法——α-平坦化调度算法,该算法能够较好地避免蚁群算法等的不足之处并且其所求得的结果也是较优的。最后,本文通过仿真实验验证了α-平坦化调度算法的正确性和有效性。

2 问题描述

调度问题通常是指对生产过程的任务(作业)执行计划,例如学习计划的制定、产品生产流程的安排、工程制造的工序规划等。调度问题可以用$\alpha|\beta|\gamma$三元组来进行分类[9],其中用α描述机器加工环境,用β描述工件加工特性,用γ表示性能指标。例如$P_3|p_{ij}=1|C_{\max}$表示有3台机器、各工件仅有一道加工工序且以最大完成时间作为性能指标的调度问题。

多处理机调度问题[10,11]是调度问题中的一种特殊类型,本文所研究的多处理机调度问题可以表示为$P_m|p_i=1|C_{\max}$。在这类调度问题中,有一含有n个独立作业的作业集合$J=\{J_1,J_2,\cdots,J_n\}$和含有m台机器的机器集合$M=\{M_1,M_2,\cdots,M_m\}$,每个作业仅有一道加工工序,可以分配给M中

到稿日期:2011-05-12 返修日期:2011-07-12 本文受国家自然科学基金(60863005,61011130038),贵州省省长基金(200802),贵州大学自然科学青年基金(贵大自青基合字[2009]021号)和贵州大学研究生创新基金项目(校研理工2011036)资助。

魏嘉银(1986-),男,硕士生,主要研究方向为软件工程、算法设计与分析,E-mail:weijiayin05@sina.com;秦永彬(1980-),男,博士,讲师,主要研究方向为可计算性与计算复杂性、智能计算、可信计算;许道云(1959-),男,博士,教授,博士生导师,主要研究方向为SAT问题、非单调推理及其计算复杂性、可计算分析及其计算复杂性。

的任意一台机器处理，各作业所需的加工时间与其加工机器不相关，为一定值；作业在运行的过程中不允许中断，调度的目标为将 J 中的作业合理地分配到 M 中的机器上加工并使最大完成时间(makespan，亦称调度时间跨度)最短。

对于多处理机调度问题，其作业集合中，各作业所需的处理时间存在一定数量关系，基于此，给出关于刻画其某种特殊数量关系的 α-平坦概念。

定义 1(α-平坦) 设 $J=\{J_1,J_2,\cdots,J_n\}$ 为作业集合，$P=\{P_1,P_2,\cdots,P_n\}$ 为相应的处理时间集合，令 $P_{\min}=\min\limits_{1\leqslant i\leqslant n}\{P_i\}$，若 $\exists\alpha\in[0,1)$ 使得 $\forall i\neq j$，均有 $|P_i-P_j|\leqslant\alpha P_{\min}$，则称作业集 J 是 α-平坦的。例如当 $\alpha=1/2$ 时，与 $P=\{21,20,18,24,26\}$ 相对应的作业集合 J 是 1/2-平坦的，而与 $P=\{3,5,9,6\}$ 相对应的作业集合 J' 则不是 1/2-平坦的。

通过研究发现，对于作业集合为 α-平坦的多处理机调度问题，当 α 取某些特殊值时，可以很容易地得到问题的相关调度算法。例如当 $\alpha=0$ 时，仅需将作业集合中的各个作业均匀地调度至机器集合中的各台机器上加工处理即可。那么，当 α 取其他值时，是否也能够有一个较好的调度策略呢？若有，对于作业集合为非 α-平坦的多处理机调度问题是否又可以通过先将其作业集合转换成 α-平坦的，然后再对转换得到的新问题进行求解，从而避开直接对原问题进行求解所面临的困难呢？本文的研究目标就是通过在多处理机调度问题中引入 α-平坦的概念，并基于此提出一种相应的求解算法；然后，提出一种对非 α-平坦的作业集做平坦化处理的算法，以期对于任意类型的多处理机调度问题都能够借助 α-平坦化调度算法来进行求解，以得到一个较优的近似解。

3 数学模型

在 $P_m|p_i=1|C_{\max}$ 问题中，设 $J=\{J_1,J_2,\cdots,J_n\}$ 为作业集合，$O=\{O_1,O_2,\cdots,O_n\}$ 和 $P=\{P_1,P_2,\cdots,P_n\}$ 分别为相应的加工工序集合与处理时间集合，$M=\{M_1,M_2,\cdots,M_m\}$ 为机器集合，作业集合的平坦度参数为 α，则引入 α-平坦概念后的多处理机调度问题的整数规划模型可表示为：

目标函数：

$$\min z=\max_{1\leqslant i\leqslant m}\left(\sum_{j\in J}x_{ij}p_j\right) \tag{1}$$

约束条件：

$$\sum_{i\in M}x_{ij}=1,j\in\{1,2,\cdots,n\} \tag{2}$$

$$\sum_{j\in J}x_{ij}p_j\leqslant z,i\in\{1,2,\cdots,m\} \tag{3}$$

$$|P_i-P_j|\leqslant\alpha P_{\min},i,j\in\{1,2,\cdots,n\},i\neq j,\alpha\in[0,1) \tag{4}$$

$$|J|=n \tag{5}$$

$$|M|=m \tag{6}$$

$$x_{ij}\in\{0,1\},i\in\{1,\cdots,m\},j\in\{1,\cdots,n\} \tag{7}$$

其中，式(1)为目标函数，该式表示以 m 台机器各自处理时间总和的最大者作为调度的性能指标；

式(2)—式(7)为约束条件，各式的含义如下：

式(2)确保每个作业均只被调度至一台机器上处理；

式(3)表示 m 台机器各自处理时间总和均不大于目标函数值；

式(4)表示该调度问题的作业集合是 α-平坦的；

式(5)、式(6)分别表示作业集合与机器集合的规模分别为 n、m；

式(7)中的 x_{ij} 为指示变量，其取值涵义如下：

$$x_{ij}=\begin{cases}0, & \text{作业 } J_j \text{ 未被调度至机器 } M_i \text{ 上处理}\\1, & \text{作业 } J_j \text{ 被调度至机器 } M_i \text{ 上处理}\end{cases}$$

4 基于 α-平坦的多处理机调度算法

记 J 为待加工的作业集合，$|J|=n$ 为作业数且 J 是 α-平坦的；P 为与 J 对应的工序加工时间需求集合；M 为可用机器集合，$|M|=m$ 为机器数；令 $RN=\lfloor n/m\rfloor$，则 $RC=n-RN\cdot m$ 为调度过程中最后一轮少于机器数的作业个数；JMT 为作业-机器调度分配表，其存储的数据格式为(作业号，机器号)；T 为调度时间跨度。

α-平坦化调度算法的主要思想在于首先将 J 中的作业按照加工时间的非增序排列得新的作业集合 J'。然后，将 J' 中的前 $m\cdot RN$ 个作业按照“蛇行”方式依次分配给 M 中的各台机器处理。最后，若 RN 为偶数，则将剩下的 RC 个作业依次分配给编号从 $m-RC+1$ 至 m 的机器处理(即跳过编号为 1 至 $m-RC$ 的机器)；否则，直接将剩下的 RC 个作业依次分配给编号从 m 至 $m-RC+1$ 的机器处理。

从算法的主要思想可以看出，其调度过程分为两个部分：前半部分严格按“蛇行”方式将作业分配给机器处理；后半部分在 RN 为偶数时要进行一次跳跃，而 RN 为奇数时则不用跳跃。算法的后半部分主要是为了确保在最后一轮的分配过程中能够将剩下的 RC 个作业分配给已用处理时间最少的 RC 台机器处理，从而达到使总的时间跨度较小的目的。算法的具体描述如下。

算法 1 α-平坦化调度算法

Input：J、M、P；

Output：JMT、T。

Begin

Step1 将 J 中的作业按照 P 值的非增序排列得新的作业集合 J'。

Step2 计算 RN。

Step3 将 J' 中的前 $m\cdot RN$ 个作业按照“蛇行”方式依次分配给 M 中的各台机器处理，并在 JMT 中记录作业一机器的对应关系。

Step4 如果 RN 为偶数，则转 Step5，否则，转 Step6。

Step5 将 J' 中剩下的 RC 个作业依次分配给编号由 $m-RC+1$ 至 m 的机器处理，并在 JMT 中记录作业一机器的对应关系。

Step6 将 J' 中剩下的 RC 个作业依次分配给编号由 m 至 $m-RC+1$ 的机器处理，并在 JMT 中记录作业一机器的对应关系。

Step7 计算 $T=\max\limits_{1\leqslant i\leqslant m}\left(\sum\limits_{j=1}^{n}x_{ij}\cdot P_j\right)$

其中，$x_{ij}=\begin{cases}0, & (i,j)\notin JMT\\1, & (i,j)\in JMT\end{cases}$

Step8 输出 JMT 与 T。

End

例 1 设 $J=\{J_1,J_2,\cdots,J_7\}$，$P=\{23,18,22,22,21,29,27\}$，$M=\{M_1,M_2,M_3\}$，$\alpha=2/3$ 时，运用算法 1 进行调度的过程为：首先按 P 值的非增序排列 J 得 $J'=\{J_6,J_7,J_1,J_3,J_4,J_5,J_2\}$，计算 $RN=\lfloor 7/3\rfloor=2$，将 J' 中的前 $RN\cdot m=2\times3=6$ 个作业按照“伪蛇行”的方式分配给 M 中的机器处理的结果为 $JMT=\{(6,1),(7,2),(1,3),(3,3),(4,2),(5,1)\}$；又因为 $RN=2$ 为偶数且 J' 中尚未被调度的作业数为 $RC=n-RN\cdot m=7-2\times3=1$，所以根据算法将 J' 中剩下的 $RC=1$

个作业(即最后一个作业)分配给编号为 $m-RC+1=3-1+1=3$ 的机器处理。由此便求得该问题的一个调度策略:$JMT=\{(6,1),(7,2),(4,3),(1,3),(3,2),(5,1),(2,3)\}$,且该调度策略的时间跨度为 $T=\max\{50,49,63\}=63$。而本问题的一个最优调度策略为:$JMT'=\{(1,3),(2,1),(3,1),(4,2),(5,1),(6,2),(7,3)\}$,其时间跨度为 $T'=\max\{61,51,50\}=61$,而 $|T-T'|=|63-61|=2$。由此可知,该算法所求得的调度方案是有效的。

设 $J=\{J_1,J_2,\cdots,J_n\}$ 为作业集合,相应的处理时间需求集合为 $P=\{P_1,P_2,\cdots,P_n\}$,记 $p_{\min}=\min\limits_{1\leq i\leq n}\{P_i\}$,$P_{\max}=\max\limits_{1\leq i\leq n}\{P_i\}$,则判定作业集合 J 是否为 α-平坦的定理 1 可描述如下。

定理 1 对于作业集合 J,J 是 α-平坦的充要条件为 $P_{\max}\leq(1+\alpha)\cdot P_{\min}$。

证明:

(充分性)

由定义得 $|P_i-P_j|\leq P_{\max}-P_{\min},\forall i,j\in\{1,2,\cdots,n\}$

$\because P_{\max}\leq(1+\alpha)\cdot P_{\min}$

$\therefore P_{\max}-P_{\min}\leq\alpha\cdot P_{\min}$

$\therefore |P_i-P_j|\leq\alpha\cdot P_{\min},\forall i,j\in\{1,2,\cdots,n\}$

即若 $P_{\max}\leq(1+\alpha)\cdot P_{\min}$,则 $|P_i-P_j|\leq\alpha\cdot P_{\min},\forall i,j\in\{1,2,\cdots,n\}$。

故若 $P_{\max}\leq(1+\alpha)\cdot P_{\min}$,则 J 是 α-平坦的。

(必要性)

$\because J$ 是 α-平坦的

$\therefore |P_i-P_j|\leq\alpha\cdot P_{\min},\forall i,j\in\{1,2,\cdots,n\}$

$\because P_{\max}-P_{\min}=\max|P_i-P_j|,\forall i,j\in\{1,2,\cdots,n\}$

$\therefore P_{\max}-P_{\min}\leq\alpha\cdot P_{\min}$

即 $P_{\max}\leq(1+\alpha)\cdot P_{\min}$

综上所述,J 是 α-平坦的充要条件为 $P_{\max}\leq(1+\alpha)\cdot P_{\min}$

由定理 1 可知,作业集 J 的平坦程度可以用其处理时间最大与最小的两个作业来刻画,即 $\alpha\geq(P_{\max}-P_{\min})/P_{\min}$。因此,非 α-平坦(即 $\alpha<(P_{\max}-P_{\min})/P_{\min}$)的作业集可以通过某些方法来不断减小 $(P_{\max}-P_{\min})/P_{\min}$ 的值,从而使经过处理的新作业集 J' 为 α-平坦的。在即将介绍的作业集的平坦化处理算法中,为了使 $(P_{\max}-P_{\min})/P_{\min}$ 的值经处理而不断减小,所采用的方法为不断合并作业集中处理时间最小与次小的两个作业为一个新的作业。

作业集的平坦化处理算法的主要思想在于根据定理 1 判定作业链表 $JList$ 是否为 α-平坦的,若已经是 α-平坦的,则直接返回;否则,不断地对 $JList$ 做平坦化处理——合并处理时间最小与次小的两个作业成为新的作业(新作业的处理时间为被合并的两个作业的处理时间之和),直至 $JList$ 是 α-平坦的。

记 J 为待处理作业集合,$|J|=n$;P 为与 J 中作业对应的处理时间需求集合;$JList$ 为用于做 α-平坦化处理的作业序列;JF 为 $JList$ 中的第一个结点,JS 为 $JList$ 中的第二个结点,JL 为 $JList$ 中的最后一个结点。则作业集的平坦化处理算法的具体描述如下。

算法 2 作业集的平坦化处理算法;

Input:J,P,α;

Output:$JList$。

Begin

Step1 将 J 中的作业按照 P 值的非减序排列并存储至 $JList$ 中。

Step2 置 $P_{\min}$ 的值为 JF 的处理时间,置 $P_{\max}$ 的值为 JL 的处理时间。

Step3 若 $P_{\max}>(1+\alpha)\cdot P_{\min}$,则执行 Step4;否则,转 Step5。

Step4 将 JF 与 JS 合并生成处理时间为此二结点处理时间之和的新结点,将新结点插入到 $JList$ 的适当位置中并更新 P,同时从 $JList$ 中删除 JF 与 JS。转 Step2。

Step5 输出 $JList$ 并终止算法。

End

例 2 设 $J=\{J_1,J_2,\cdots,J_7\}$,$P=\{24,18,22,26,21,29,27\}$,$M=\{M_1,M_2,M_3\}$,$\alpha=1/2$ 时,运用算法 2 做平坦化处理的过程为:首先按 P 值的非减序排列 J 得 $JList=\{J_2,J_5,J_3,J_1,J_4,J_7,J_6\}$,置 $P_{\min}=18$,$P_{\max}=29$;因 $P_{\max}>(1+1/2)\cdot P_{\min}$,所以将 J_2 与 J_5 合并得 J_8,且 $P_8=P_2+P_5=18+21=39$,合并后得 $JList=\{J_3,J_1,J_4,J_7,J_6,J_8\}$,$P=\{22,24,26,27,29,39\}$;此时 $P_{\min}=P_3=22$,$P_{\max}=P_8=39$,所以 $P_{\max}>(1+1/2)\cdot P_{\min}$,继续如上的平坦化操作,直至所得的 $JList=\{J_9,J_{10},J_{11}\}$,$P=\{46,53,68\}$,此时 $P_{\min}=P_9=46$,$P_{\max}=P_{11}=68$,故 $P_{\max}\leq(1+1/2)\cdot P_{\min}$,$JList$ 已经是 α-平坦的,输出 $JList$ 并结束算法。

5 实验结果

为了说明本文所提算法的正确性与有效性,以 VC++ 作为开发环境,采用 C 语言为开发语言,对文中提出的两个算法加以实现并做如下两个实验来进行验证。

在实验一中运用本文所提出的算法分别对 4 种不同情形下的多处理机调度问题进行求解,实验中所使用的数据如表 1 所列。

表 1 实验数据

PID	JC	MC	处理时间	α 值
1	6	3	{6,4,5,5,3,4}	1/2
2	6	2	{1,3,3,3,3,3}	1/10
3	13	3	{6,4,17,15,3,4,15,60,7,6,3,18,5}	9/10
4	5	3	{7,7,9,9,9}	3/10

实验一的具体结果如表 2 所列,通过对本实验的结果进行分析可知,采用本文所提出的算法能够求解出问题的一个较优的近似解,且其与最优解之间的差值较小。

表 2 实验一的具体结果

PID	RC	⌊JC/MC⌋	本文算法解	OPT
1	零	偶数	11	9
2	零	奇数	10	9
3	非零	偶数	65	60
4	非零	奇数	16	16

为了更好地说明本文所提 α-平坦化调度算法的有效性,在实验二中,分别运用模拟退火算法、文献[12]中的算法和本文所提出的 α-平坦化调度算法对一个 benchmarks 问题重复求解 100 次,并统计出由各算法求解时所得到的最好解、最差解、解的平均值和最好解出现的次数。实验二中所使用的 benchmarks 问题中有 9 个作业和 3 台处理机,各作业所需的运行时间分别为 81、40、26、4、65、98、53、71、15,在此实验中设定 $\alpha=9/10$,实验所得的统计结果如表 3 所列。

表 3 实验二的统计结果

算法	最好解	最差解	平均值	最好解出现次数
模拟退火算法	151	160	152.19	35
文献[12]中算法	151	155	151.56	60
α-平坦化调度算法	152	152	152	100

通过分析实验二的统计结果可以得出，α-平坦化调度算法同模拟退火算法与文献[12]所提出的算法相比较而言，虽然不能求得该问题的最优解，但所求得的近似解与最优解之间仅相差 1 个单位，且在 100 次的重复求解过程中，所求得的解是一样的、确定的；而模拟退火算法和文献[12]中的算法虽然有时能够求得最优解，但最优解的出现具有随机性，且所求得的最差解同最优解之差较大。所以，综合而言，本文所提出的 α-平坦化调度算法的调度性能是较优的。

结束语 本文在研究多处理机调度问题的基础上，提出 α-平坦的概念，并基于此提出了一种基于 α-平坦的多处理机调度算法——α-平坦化调度算法。该算法运用将处理时间最小与次小的作业不断地相互合并的方法对非 α-平坦的作业集进行平坦化处理；在将作业集平坦化之后，采用"伪蛇行"的调度方式对其进行调度分配，从而求得原问题的一个近似解。通过这种方式可以有效地求得原问题的一个较优的近似解，很好地避开了直接对原调度问题进行求解时所遇到的问题，且不存在运用模拟退火算法等启发式算法进行求解时所带来的随机性问题。

实验结果表明，本文所提出的 α-平坦化调度算法在对多处理机调度问题求解中表现出较好的效果，但是如何确定参数 α 以使所求得的调度策略较优是较困难的，这将是我们下一步的研究重点。

参 考 文 献

[1] Garey M R, Johnson J S. Computers and Intractability: A Guide to the Theory of NP-Completeness [M]. San Francisco, CA: Freeman, 1979

[2] Ahmad I, Kwok Y-K. On Parallelizing the Multiprocessor Scheduling Problem [J]. IEEE Transactions on Parallel and Distributed System, 1999, 10(4): 414-432

[3] Yang Xiao-guang. A glass of generalized multiproces-sor scheduling problems[J]. Systems Science and Mathematical Sciences, 2000, 13(4): 385-390

[4] Huang K L, Liao C J. Ant colony optimization combined with taboo search for the job shop scheduling problem[J]. Computers & Operations Research, 2008, 35: 1030-1046

[5] 张志强，张璟，张翔，等. 解决作业车间调度问题的改进蚁群优化算法[J]. 应用科学学报，2010，28(2)：182-188

[6] 徐立芳，莫宏伟. 基于自适应克隆启发算法的作业车间调度[J]. 计算机工程，2009，35(4)：207-209

[7] 蔡斌，毛帆，傅鹂. 等. 解决作业车间调度的微粒群退火算法[J]. 计算机应用研究，2010，27(3)：856-859

[8] 唐海波，叶春明. 一种求解作业车间调度的混合粒子群算法[J]. 计算机应用研究，2011，28(3)：883-889

[9] Brucker P. Scheduling Algorithms [M]. Springer, 2006

[10] 冯斌，孙俊. 一种多处理机任务分配的启发式算法[J]. 计算机工程，2004，30(14)：63-65

[11] 王剑波，陈内萍. 多处理机独立任务调度问题的 DNA 计算机算法[J]. 湖南师范大学自然科学学报，2009，32(3)：36-41

[12] 高尚，杨静宇. 多处理机调度问题的粒子群优化算法[J]. 计算机工程与应用，2005，41(27)：72-73

(上接第 155 页)

结束语 针对数据流概念漂移问题，本文提出一种基于混合集成方法的数据流概念漂移检测方法 WE-DTB。首先用水平集成的方法建立集成分类器，然后利用朴素贝叶斯分类器去除噪音。实验表明，WE-DTB 方法能够有效检测概念漂移且具有较好的分类精度以及时空性能。然而，如何克服方法在误报率上的劣势以及探索其它分类器在混合模型中的作用，将是未来研究的重点。

参 考 文 献

[1] Widmer G, Kubat M. Learning in the Presence of Concept Drift and Hidden Contexts[J]. Machine Learning, 1996, 23(1): 69-101

[2] Schlimmer J C, Granger R H. Incremental Leaning from Noisy Data[J]. Machine Learning, 1986, 1(3): 317-354

[3] Wang H X, Fan W, Yu P S, et al. Mining Concept-Drifting Data Streams Using Eensemble Classifiers[C]// Proc of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington: ACM Press, 2003: 226-235

[4] Scholz M, Klinkenberg R. An Ensemble Classifier for Drifting Concepts[C]// Proc of 16th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Porto, 2005

[5] Fan W. Systematic Data Selection to Mine Concept-Drifting Data Streams[C]//Proc of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, 2004: 128-137

[6] Gao J, Fan W, Han J W. On Appropriate Assumptions to Mine Data Streams: Analysis and Practice[C]// Proc of the 7th IEEE International Conference on Data Mining. Omaha, 2007: 143-152

[7] Zhang P, Zhu X Q, Shi Y, et al. An Aggregate Ensemble for Mining Concept Drifting Data Streams with Noise[C] //Proc of the 13th Pacific-Asia Conference on Knowledge Discovery. Bangkok, 2009: 1021-1029

[8] Li P P, Hu X G, Wu X D. Concept Drifting Detection on Noisy Streaming Data in Random Ensemble Decision Trees[C] //Proc of the 6th International Conference on Machine Learning and Data Mining. 2009: 236-250

[9] Li Y, Zhang Y H, Hu X G. A Classification Algorithm for Noisy Data Streams [C] // Proc of 3rd International Conference of Fuzzy Systems and Knowledge Discovery (FSKD). Yantai, China: Springer, 2010: 2239-2244

[10] ACM Special Interest Group on Knowledge Discovery and Data Mining. KDDCUP99 data set [EB/OL]. http://kdd.ics.uci.edu/databases/ kddcup99, 1999

[11] Tan P-N, Steinbach M. 数据挖掘导论[M]. 范明，等，译. 北京：人民邮电出版社，2006

[12] Wang Y, Li Z-H, Zhang Y. Classifying Noisy Data Streams[C]// Proc of 3rd International Conference of Fuzzy Systems and Knowledge Discovery (FSKD). Xi'an: Springer, 2006: 549-558