IPSO 算法在黄酒发酵温控系统中的应用①

赵兢兢, 王志勇, 徐保国

(江南大学 物联网工程学院, 无锡 214122)

摘 要: 黄酒发酵温度控制系统是一个时变的、非线性的系统. 在黄酒发酵控制中温度是一个非常重要的参数, 针对黄酒发酵的温度特性, 设计出了一种基于位置加权和自适应惯性权重的改进粒子群算法(IPSO)的PID温度控制系统. 带位置加权的PSO算法减小了搜索过程的盲目性, 克服了基本PSO算法易陷入局部极值的缺陷. 自适应惯性权重兼顾了粒子的全局与局部搜索能力. 将 IPSO 优化算法用标准测试函数进行测试,结果表明该算法优化结果比标准PSO算法有所提高. 并采用LabVIEW与MATLAB混合编程的方法进行仿真比较, 结果表明, 该控制方法具有动态响应快、超调量小、鲁棒性强等优点, 有很好的实用性.

关键词: 黄酒温度控制; 粒子群优化; PID 控制; LabVIEW; MATLAB

Application of IPSO to Rice Wine Fermentation Temperature Control

ZHAO Jing-Jing, WANG Zhi-Yong, XU Bao-Guo

(School of IoT Engineering, Jiangnan University, Wuxi 214122, China)

Abstract: The control system for rice wine fermentation is an unstable system with time-varying and nonlinearity. Temperature is a very important parameter in rice wine fermentation control. For rice wine fermentation temperature characteristics, a PID temperature control system is designed based on particle swarm algorithm (IPSO) with Position Weighted and adaptive inertial weight. The improved PSO algorithm with Position Weighted, which can reduce the blindness in the search process to overcome the defect that basic PSO algorithm is easy to fall into local minima is proposed. And adaptive inertial weight can balance the global and local search ability of the particles. It uses the IPSO to optimize the standard test functions and analyzes the test results, and finds the test results are better than before. This paper uses LabVIEW and the MATLAB mix programming method to simulate and compare. The result shows the control strategy has a fast dynamic response, slight overshoot and strong robust, and with good usability.

Key words: rice wine temperature control; particle swarm optimization; PID control; LabVIEW; MATLAB

我国黄酒具有悠久的历史, 是我国的国粹. 目前黄酒生产技术总体比较落后, 实现黄酒发酵过程的全自动化, 已经成为工业控制领域的一个重要任务. 在各种环境条件中, 温度是影响发酵质量和口感的关键因素. 因此更好地控制调节黄酒发酵的温度是必要的.

传统的 PID 控制器不需要精确的数学模型, 结构简单, 具有较强的鲁棒性, 在许多领域得到了广泛应用. 然而, 当对象的参数改变时, 传统的PID控制应根据经验知识进行调整, 这可能会导致该系统的性能变差. 为了克服这个缺点, 目前常用的智能优化 PID 参数算法主要有神经网络算法和遗传算法等. 神经网络算法需要大样本训练, 且容易出现数值病态问题, 而遗传算法则需要进行复制、交叉与变异操作, 进化速度慢, 易性[1]. PSO 算法是基于群智能的并行全局搜索策略, 收敛速度快, 算法简单. 但标准的 PSO 算法容易陷入局部最优[2]. 本文采用 IPSO 算法来克服这一缺陷将其用于调整黄酒发酵温度控制系统中的 PID 参数值, 结合 LabVIEW 和 MATLAB 来进行混合编程使系

① 基金项目:教育部国防基础科研基金(A1420080177);江苏省博士后基金(1101021B)
收稿时间:2012-05-24;收到修改稿时间:2012-07-01

统运行能够实时的得到调整和整定, 以提高系统的鲁棒性和控制品质.

1　黄酒发酵温度控制系统

黄酒发酵温度控制系统由西门子工控机、西门子PLC、智能仪表以及温度传感器和执行阀门等组成. WEP-2312 铂电阻采集到的温度值能直接被智能仪表读取显示和进行 PID 智能算法控制冷水电磁阀, 并通过仪表的 RS-485 通信模块将数值传递给上位机处理; 上位机监控软件采用 LabVIEW 编写, 通过 DataSocket 技术实现对 OPC Server 的访问、PLC 开关量和模拟量的读取, 实现数据的分析、显示、报警、储存和打印以及实时、历史曲线绘制. 本系统还可以通过上位机程序向 PLC 发送指令, 以实现对阀门等执行机构的控制来调节温度. 黄酒发酵温度控制系统框图如图 1 所示.

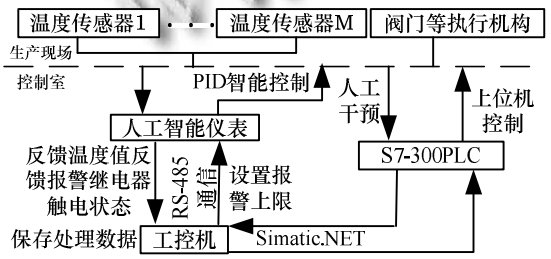

图 1　黄酒发酵温度控制系统框图

1.1 黄酒发酵温控系统中基于 IPSO 算法的 PID 控制器结构

黄酒发酵温度控制 PID 模型如图 2 所示. 它由常规 PID 控制器、IPSO 算法和被控对象三部分组成, 利用 IPSO 算法对 PID 的三个参数 K_p、K_i、K_d 进行整定[3].

PID 的控制方法采用如公式(1)所示的增量式 PID 的形式.

$$u(k)=u(k\text{-}1)+Kp[e(k)-e(k\text{-}1)]+Kie(k)+Kd[e(k)+e(k-2)-2e(k\text{-}1)] \tag{1}$$

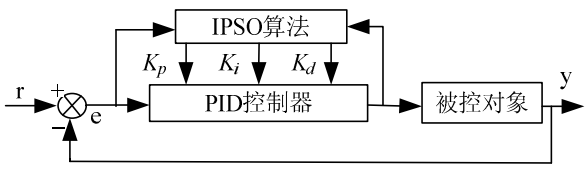

图 2　黄酒发酵温度控制 PID 模型

2　改进的粒子群优化算法(IPSO)

2.1 基本 PSO 算法

粒子群算法(PSO)是计算智能领域的一种群智能算法, 通过个体间的协作和竞争实现全局搜索. 系统初始化的随机解称为粒子, 不同于遗传算法, 此算法没有交叉和变异算子, 它是通过粒子群在解空间追随最优的粒子飞行, 完成数学公式的迭代寻优.

其数学描述为: 设一个 n 维的搜索空间, 由 m 个粒子组成的种群 $X=\{x_1,\cdots x_t,\cdots x_m\}$, 其中, 第 i 个粒子的位置为 $x_i=\{x_{i1},x_{i2},\cdots x_{im}\}^T$, 其速度为 $v_i=\{v_{i1},v_{i2},\cdots,v_{im}\}^T$. 其个体极值为 $pbest_i=\{pbest_{i1},pbest_{i2},\cdots,pbest_{im}\}^T$, 全局极值为 $gbset=\{gbset_1,gbest_2,\cdots,gbest_m\}^T$. 按照追随当前最优粒子的原理, 粒子按(2)、(3)式改变速度和位置.

$$v_i=w*v_i+c_1*r_1*(pbest_i-x_i)+c_2*r_2*(gbest_i-x_i) \tag{2}$$

$$x_i=x_i+v_i \tag{3}$$

其中 x 是粒子的位置, v 是粒子的速度, pbest 是粒子的个体最佳位置, gbest 是粒子的全局最佳位置, c 是学习因子, r 是一个随机数, w 是惯性权重.

2.2 IPSO 算法

2.2.1 位置加权

粒子前一次的位置更新对之后的位置有一定的影响, 将影响因子 α 引入到基本 PSO 算法公式的更新, 即在公式(3)后面加上公式(4).

$$x_i'(t+1)=\alpha x_i(t)+(1-\alpha)x_i(t+1) \tag{4}$$

从而对粒子新位置进行选择, 把构造的改进粒子群算法称为位置加权粒子群算法.

式(4)中, $x_i(t+1)$ 为粒子的当前位置, $x_i(t)$ 为粒子的前一次位置, $x_i'(t+1)$ 为粒子的新位置. $\alpha\in[0,1]$ 为加权因子, 代表粒子的前一次位置和当前位置在粒子新位置选择中所占的权重.

当 α 较小时, 粒子的前一次位置对新位置的影响较小, 算法的收敛速度和基本粒子群算法的收敛速度相当; 当 α 逐渐增大时, 粒子的前一次位置对新位置的选择所占的权重加大, 使得粒子在选择新位置的过程中有了参考点, 减少了粒子位置选择的盲目性, 因此收敛速度会逐渐提高; 而当 α 过大时, 粒子由于受到前一次位置的影响过大, 则往往容易陷入局部极值. 因此, 算法中, α 的取值是影响算法收敛速度的关键

因素[4]. 如果将公式(3)带入公式(4), 可以得到

$$x_i'(t+1)=x_i(t)+(1-\alpha)v_i(t+1) \tag{5}$$

这表明, 对位置的加权因子 α, 相当于对速度的约束因子 $(1-\alpha)$. 这相当于在搜索空间里面对搜索的方向未加限制, 而对搜索的步长加以约束. 这样能够有效地避免在搜索过程中由于步长过大而错过了全局极值, 减少了搜索过程中的盲目性, 因此能提高算法的搜索效率.

2.2.2 自适应变化惯性权重

惯性权重 w 的取值大小对粒子群的算法优化效果有着很大的影响. 惯性权重 w 的数值小则提高了算法的收敛性, 数值变大算法将改善算法的全局搜索能力, 避免陷入局部最优点. w 的值的选取要综合考虑两者的影响, 但是使用一般的线性减小策略时 w 会快速减小, 不能在算法初期长时间保持较大的值. 对此本文通过自适应变化 w 值来控制算法进程, 以兼顾粒子全局及局部搜索能力. w 的计算公式如下.

$$w=\frac{1}{1+e^{\max(\min\{l_{ij}\},\frac{1}{k})/\max\{l_{ij}\}}} \tag{6}$$

其中, l_{ij} 表示任意两粒子间的定义距离, k 为迭代次数. 由公式(6)可知, 为避免初期 $\min\{l_{ij}\}=0$ 对 w 的影响, 本文取 $\max(\min\{l_{ij}\},\frac{1}{k})/\max\{l_{ij}\}$, 以保证惯性权重 w 递减. 利用(6)式可使惯性权值 w 自适应非线性减小, 提高了全局与局部搜索能力, 改善了算法的收敛的性能.

粒子群算法优化参数的流程图如图 3 所示.

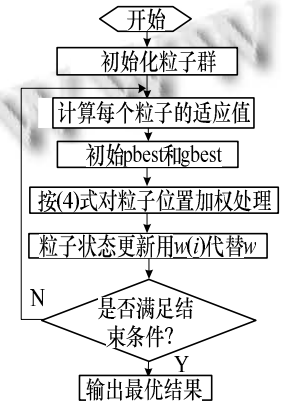

图 3 改进粒子群算法流程图

为了验证 IPSO 的优化效果, 下面通过 2 个经典的测试函数对本算法进行测试.

F1:Griewank 函数 $f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$

F2:Rosenbrock 函数 $f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$

测试的主要参数设置如下: 种群个数 N=50, 空间维数 D=10, c1=c2=1.5, 最大迭代次数为 500, w 的值按公式(6)计算. 考虑算法的随机因素, 测试中对每个函数测试 10 次. 取结果的平均值进行比较, 比较结果如表 1 所示. 从表中可以明显看出 IPSO 优于标准 PSO 算法(当 α =0), 并且 α 值的增加减少了迭代次数.

表 1 函数实验结果

位置加权	F1		F2	
α	迭代次数	最优值	迭代次数	最优值
0	201	4.2015	347	2.1
0.2	134	3.7019	258	0.81
0.4	95	1.9019	207	0.73
0.6	93	1.0019	199	0.75

3 基于LabVIEW的IPSO-PID在黄酒发酵温度控制中的应用

以古越龙山黄酒生产线为例, 选用的黄酒发酵温度模型的传递函数为:

$$G(s)=\frac{e^{-0.2\tau}}{s^2+25s+2}$$

实验中 PSO 算法参数设定为粒子的规模为 100, 最大迭代次数为 100 次, c1=c2=2, w 的值按公式(4)计算代入. 性能评价指标函数 J 为:

$$J=\int_0^{\infty}e^2(t)dt$$

3.1 编程及实现

本文把 LabVIEW 与 MATLAB 结合, 充分利用 MATLAB 提供的大量高效可靠的算法和 LabVIEW 的图形化编程能力, 发挥两者的优势, 大大提高了工作效率.

采用 LabVIEW 实现的 PID 控制器程序图如图 4 所示.

本文采用了改进了的 PSO-PID 控制算法, 先通过 IPSO 获得 PID 的三个控制参数 K_i、K_p、K_d, 再通过 PID 控制系统调节阀门. IPSO 算法在 LabVIEW 提

供的脚本和公式(script&Fomulas)功能中 MATLAB Script 脚本节点中实现，脚本程序先在 MATLAB 环境下编写调试完毕，在节点边框点击右键在弹出菜单中选择“Import”，直接导入；或者直接编写程序[5,6].

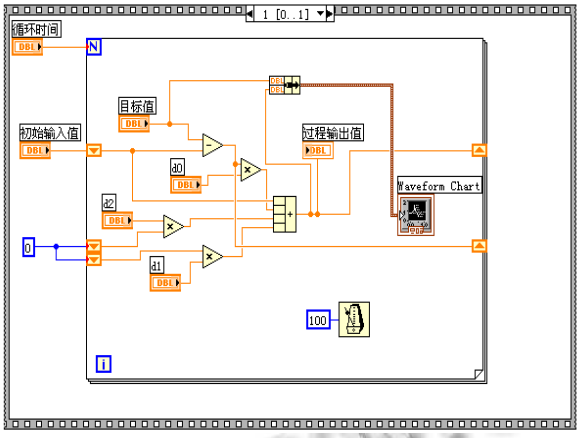

图 4　黄酒发酵温度 PID 程序图

MATLAB Script 节点的数据通信通过在节点边框添加输入、输出控件，还可以波形显示．MATLAB Script 节点部分程序图如图 5 所示.

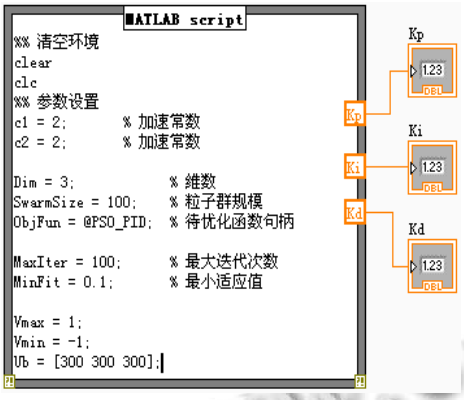

图 5　IPSO 算法 MATLAB Script 节点部分程序图

在阶跃信号输入下，对温度对象分别使用 IPSO-PID 控制和传统 PID 控制进行仿真比较．图 6 为 IPSO 整定的 PID 参数曲线图，经过 22 次的迭代找到最优值，具有较快的响应速度.

图 7 和图 8 分别为 IPSO 优化的 PID 的输出曲线和传统 PID 输出曲线．可以明显的看出，IPSO-PID 控制与传统 PID 控制相比具有明显的优越性，减小了系统的超调量，拥有良好的稳定性和鲁棒性.

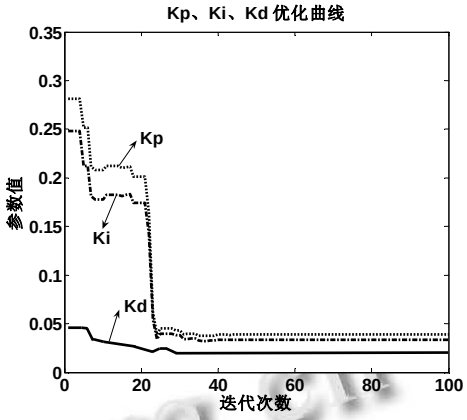

图 6　Kp、Ki、Kd 优化曲线图

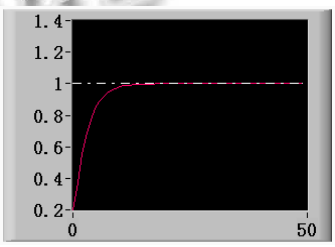

图 7　IPSO-PID 系统输出曲线

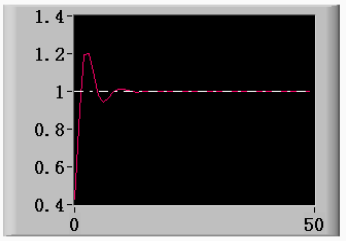

图 8　传统 PID 系统输出曲线

4　结论

PID 控制器的参数优化已成为一个重要的研究课题．以黄酒黄酒温度控制为对象，应用 IPSO 算法优化的 PID 控制器参数，提高了系统的智能决策水平和鲁棒性，并且能适应工作要求、算法结构简单．并利用 LabVIEW 中 MATLAB 脚本节点，将 MATLAB 语言实现的 IPSO 算法嵌入到 LabVIEW 流程图中．MATLAB 在后台提供大型算法供 LabVIEW 调用．实现了黄酒发酵温度的控制，满意的仿真结果表明，该算法在黄酒发酵温度控制问题上是有效的.

参考文献

1 陶永华.新型 PID 控制及其应用.第 2 版.北京:机械工业出版社,2005.

(下转第 90 页)

会生成一些动态文件描述进程状态和其他信息.

这些信息在内核中的格式为:

seq_printf(m, "%d (%s) %c %d……\n", pid_nr_ns(pid, ns),tcomm, state,ppid,……);

pid_nr_ns(pid, ns)是进程号; tcomm 是应用程序或命令的名字; state 是进程状态; ppid 是父进程 ID……. 嵌入式产品应用环境复杂, 而在其上运行的程序较受到外界影响导致程序的不可控, 有可能需要让设备自行重启.

在实际项目中的具体实现.

```
/*该方法检测某个子进程是否为僵死状态*/
int32_t IsProcessAlive(pid_t pid){
  FILE *fp;/*对应子进程号 pid 的文件指针*/
char_t CStr[64];/*子进程对应的路径*/
  /*从 fp 中读取数据到 CStr,若 fp 中没有数据则关闭 fp*/
    fgets(CStr, sizeof(CStr), fp)
    /*找到第一个右括号, 然后返回指针*/
    pCIdStr = strchr(CStr, ')');
    ……
    pCIdStr += 2;
    switch ((char_t)*pCIdStr) {
      case 'D':        /* 不可中断的睡眠*/
      case 'R':        /* 就绪  */
      case 'S':        /* 睡眠*/
      case 'T':        /* 跟踪或停止*/
          I32IsAlive = TRUE;
          break;
      case 'Z':         /*僵尸进程*/
      default:
          I32IsAlive = FALSE;
          break;
    }
    fclose(fp);
    return I32IsAlive;
}
```

在嵌入式程序开发中, 程序被分为若干子进程, 每个子进程完成自己的具体任务. 主进程创建完子进程后, 启动看门狗. 在这个循环中主进程监视每个子进程的状态, 监视过程是通过子进程进程号对应目录下的 stat 文件中的 state 字段进行判断. 若某个进程出现异常, 主进程应采取重新创建该子进程或系统重启, 从而保证程序的稳定性.

参考文献

1 Maybee S. File system framework. USA: Sun Microsystems Press, 2006.

2 Juan I, Florido S. Journal File Systems. www. Linuxg-azette.com.

3 Cheng GG, Liu XY, Bao KM. Startup process analysis of Linux kernel. Computer Engineering and Design. 2006, 27(9):1528−1529.

4 Huang YJ, Cheng YF. Linux Kernel Slab Memory Buffer Manage, Computer Engineering, 2006,32(24):31−33.

5 Bovet DP, Cesati M. Understanding The Linux Kernel. 3rd Edition.USA: O'Reilly Media. 2007. 328−371.

6 Corbet J, Rublini A, Hartman C. Linux Device Drivers. 3rd Edition. USA:O'Reilly Media. 2006. 21−234.

(上接第 94 页)

2 Yang W, Qang LQ. Summarization of PSO. Chin Eng Sci, 2004,6(5):87−94.

3 王东风,韩璜.基于粒子群优化的混沌系统比例-积分-微分控制.物理学报,2006,55(4):1644−1646.

4 朱童,李小凡,鲁明文.位置加权的改进粒子群算法.计算机工程与应用,2011,47(5):4−6.

5 赵旎,火长跃.基于G语言的液位控制系统研制.河南科技大学学报(自然科学版),2006,(8):46−48.

6 唐进元,黄云飞,磊文利.基于 LabVIEW 的数字 PID 控制器的设计研究.测控技术, 2007,26(3):35−37.