第25卷第1期
1996年2月

信息与控制
Information and Control

Vol. 25, No. 1
Feb., 1996

一个基于QSIM算法的定性仿真系统GQSS†

白方周　陈　源　鲍忠贵
（中国科技大学计算机系　合肥　230027）

摘　要　定性仿真是目前人工智能、仿真技术等领域一个新颖的研究方向，文中重点介绍了作者研制的一个基于QSIM算法的实验性通用型定性仿真系统GQSS(General Qualitative Simulation System)，并且在该系统中初步改进了QSIM算法.

关键词　定性仿真，人工智能，算法

1　引言

90年代，随着仿真技术与其他学科技术的交融，仿真技术又有了新的发展，出现了智能化仿真、分布式并行仿真、仿真支持系统等新兴仿真方法，仿真技术的应用前景越来越广阔. 然而无论是传统的计算机仿真，还是新兴的智能仿真与并行仿真都未离开数字仿真的范畴，它们都是建立在精确的系统数学模型基础上的仿真，实质上都是将系统内部结构表述为精确的数学方程，最终通过求解方程组来得出系统基于函数解上的行为描述，这些行为描述给出了每个状态变量在给定时间点上的精确数值. 在实际应用中，出现了许多上述仿真技术无法解决或难以解决的新问题.

(1) 实际系统由于过于复杂或知识不完备，人们无法构造系统精确的数学模型.

(2) 受计算机性能及现有算法的限制，对复杂系统进行传统的数字仿真将花费大量的计算时间和人力，且很难实现对快速动态系统的实时仿真，采用高性能的并行计算机，不仅价格昂贵，而且也未必能解决所有问题. 在实际应用中，我们往往并不需要完全精确的数值信息，而仅仅对有关系统参数的一些本质的定性知识感兴趣.

(3)在专家系统中，根据专家们的经验和判断所建立的知识是一种"浅层"知识，因此在外界条件发生变化，碰到新情况或系统知识不完备时，系统往往不能作出正确的判断. 然而在研究领域内，存在一些基本的规律和法则，即"深层"知识. 如何利用这些"深"知识来改善专家系统的性能呢?

解决以上3个问题的关键就是寻求一种能处理系统定性知识(描述定性知识和利用定性知识进行推理)的方法. 定性过程理论(定性建模、定性推理和定性仿真)提供了一种可行的方法.

定性模型理论研究如何建立起不完备知识系统的模型——定性模型，定性模型必须能够描述系统的定量和定性两部分知识. 定性推理则是研究如何用这种定性知识进行形式化推理，把定性推理应用于系统定性模型上，通过推理来产生和解释系统的行为便是定性仿真.

在定性仿真的研究中，美国学者起步较早，其中美国施乐公司的de Kleer和Brown关于定性模型和定性推理的研究给定性仿真理论打下了坚实的基础；美国麻省理工学院的Forbus

1994－12－10收稿
† 国家自然科学基金资助项目

则对定性仿真理论作了全面的论述，1986年美国德州大学的Kuipers提出的动态仿真算法QSIM使定性仿真接近于实用，近年来，Kuipers等人又在定性数学方法、定性模型的建立以及定性与定量结合等方面作出了新的成绩.

定性仿真不仅可以应用于系统的仿真研究，而且在系统的故障诊断和分析、各种仿真培训系统以及实时专家控制系统的构造等方面都有广泛的用途. 此外定性仿真所涉及到的定性知识处理和推理方法的研究对增进专家系统的灵活性也有重要的作用.

本文将重点介绍我们设计开发的GQSS系统. 在该系统中我们对原GQSS算法做了初步改进，提高了算法的效率，避免了部分奇异行为.

2 通用型定性仿真系统的设计与实现

2.1 GQSS简介

GQSS是一个集成的定性仿真软件，其使用的定性仿真算法以Kuipers的QSIM为基础. GQSS算法以系统的定性约束方程和系统的一个初始状态为输入，通过定性仿真，输出系统的定性行为. 为了便于使用者描述系统的定性结构，GQSS提供了一种系统定性模型描述语言QML. QML是一种简单的非过程性语言，它以接近于自然语言的形式向使用者提供了一种灵活方便的描述系统的参数集合、参数的初始路标值、系统的约束方程及系统的初始状态的方法. GQSS通过翻译用户用QML语言编写的系统描述文件建立起系统的定性模型，然后用QSIM算法进行定性仿真，产生出系统的定性行为，以直观的定性描绘图方式提供给用户.

GQSS提供了命令行和集成环境两种方式供用户选择使用. 命令行方式下，用户只要打入相应的命令，GQSS自动完成QML语言的翻译、系统的定性仿真和系统预测行为的显示输出，这种方式很适合于熟练用户使用. 集成环境方式提供了一个类似Turbo C的集成环境，用户在集成环境中，通过菜单选择进行模型描述文件的编辑、QML语言的翻译、系统的定性仿真、系统行为定性描绘图的浏览以及查看帮助信息等工作；编辑器是标准的全屏幕编辑，翻译程序能准确定位描述文件中的语法和语义错误，集成环境方式对新手和熟练用户都很方便. GQSS用Borland C++(2.0)实现，运行效率较高.

2.2 GQSS的总体设计思想和软件结构

2.2.1 QML语言

GQSS的核心是QSIM算法，QSIM需要系统的下述信息作为输入：(1)系统的参数集合；(2)参数的初始路标值；(3)参数间的约束方程；(4)系统的初始状态.

如何使用户方便地描述出QSIM需要的系统信息，是设计者遇到的首要问题. 解决这个问题的方法是提供用户一种描述系统定性结构的语言，它以一种类似自然语言的风格来描述相应的信息. 如系统有5个参数a,b,c,d,e，就描述为function:{a,b,c,d,e}；a,b之间有微分

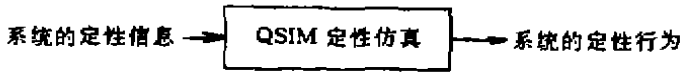


图1 GQSS流程图

关系就描述为DERIV(a,b). 用户只需在编辑器中,象编写普通程序一样,用描述语言提供的语句描述出系统的定性结构即可;这种方法需要设计一个描述语言的翻译程序,对描述语言进行语法分析、语义分析和语义动作的执行,为方便用户的使用,使系统具有更强的描述功能和灵活性.

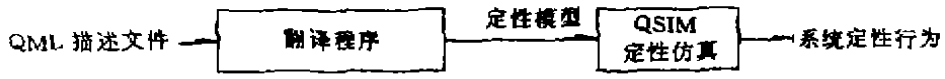


图2　GQSS仿真流程图

2.2.2　GQSS对QSIM算法的几点改进方案

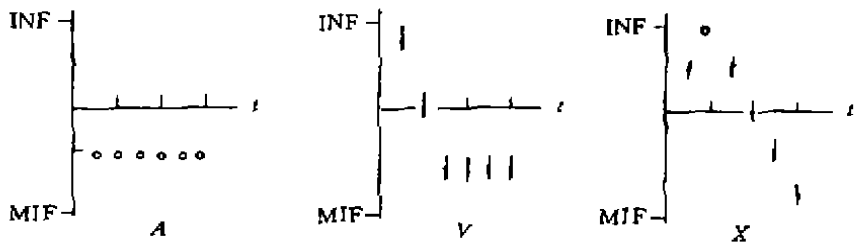
(1) 在全局过滤中有关状态一致的定义,Kuipers把它定义为两个状态的所有对应参数都等于相同路标值,而且变化方向相同. 在设计GQSS中,我们将判断状态一致的标准放宽了,只要求对应参数等于路标值,并不要求路标值相等,这是为了避免QSIM算法无终止地进行下去,把资源耗尽. 如弹簧振动系统中,它的定性行为描绘图见文[10]中的图2.

由该图可以看出,如果按Kuipers原来的严格定义,最终将导致无穷多个定性行为的产生,直到计算机资源耗尽.

(2) 在状态转换时,我们仅将形如($+\infty$, std)或($-\infty$, std)的定性状态过滤掉,而不象Kuipers所描述的那样,将($-\infty$, dec)和($+\infty$, inc)也过滤掉. 我们将($-\infty$, dec),($+\infty$, inc)保留下来是为了系统描述趋近于无穷的行为,因此,我们增加一个全局过滤规则:

如果所有参数的定性值都等于∞,则删除掉;如果其中有一个参数的状态为稳定,则把新状态加入结束状态表.

用该规则将参数状态为($-\infty$, dec)或($+\infty$, inc),且其中至少有一个参数状态为稳定的系统状态看作是系统的终止状态. 如果不作这样的改进,在重力场中,向上抛一个小球的系统的定性仿真将不会终止,因为小球下落后,并无描述小球碰地的约束方程,所以小球的速度和位移都将趋向无穷. 在全局过滤中,我们将定性状态 A:(g,std) , v:($-\infty$, dec),x:($-\infty$, dec)认为是系统的结束状态. 如图3GQSS所产生的定性行为.



↑,↓,。分别表示参数变化方向为增加、减少和不变

图3　GQSS产生的上抛运动定性行为描绘图

(3) 在 I-转换中,抛弃形如$(+\infty, std)$或$(-\infty, std)$的状态. 当产生新路标值时,不能简单地在原有的路标值集合里加上新路标值,这是因为同一个参数,当它朝着不同路径发展时,其路标值集合是不同的,在这个状态转换中发现了新的路标值,而在另一个转换中并未发现新的路标值. 所以需要给这个参数另外创建一个新路标集合,将其父节点的路标值集合复制过来,然后加入新发现的路标值. 但是,该参数会发生路标值和时间点的对应与其他参数不一致,从而产生一些奇异行为,这是 QSIM 算法本身造成的. 为了避免这种情形,在参数的新路标值集合中标记该新路标值和对应参数路标值,使得在局部过滤中滤除掉奇异行为.

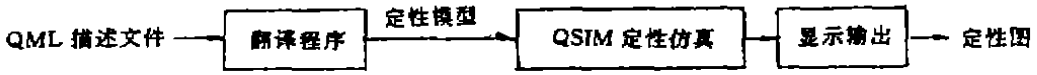


图 4 GQSS 仿真流程图

2.2.3 改进的 QSIM 算法的实现

QSIM 算法是通过不断地扩展当前状态,最终产生一棵系统状态树,从根节点(初始状态)到叶节点(终止状态)的路径上的所有状态就组成了系统的一个定性行为描述. 所以实现 QSIM 算法的关键就是维护一棵树,不断地对它进行扩展.

为了便于用户直观地了解系统行为的演化过程,GQSS 系统将预测出来的系统定性行为以定性描绘图的形式提供给用户. 为此,GQSS 需要一个系统定性行为的输出显示模块,来将 QSIM 算法产生的系统行为以定性描绘图的形式输出在屏幕上,见图 4. 程序流程见图 5.

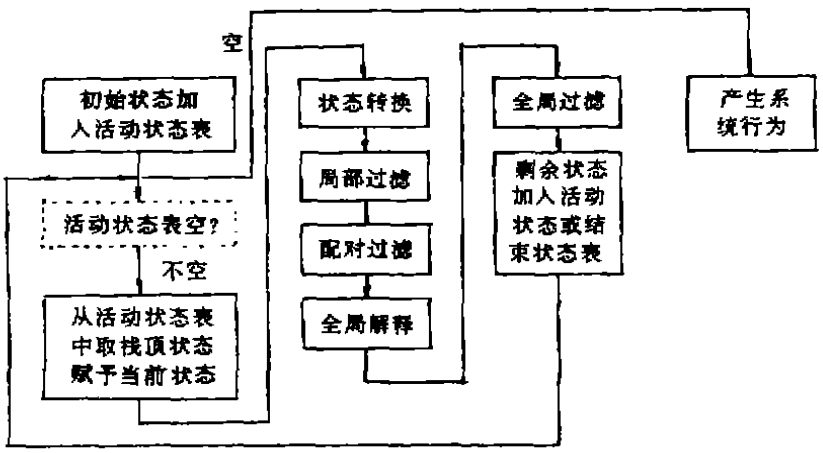


图 5 QSIM 算法流程图

2.2.4 系统环境

集成环境是通过菜单驱动的,GQSS 需要一个总控模块来管理有关操作,它还将管理菜单

和屏幕的显示和更新.对命令行方式,GQSS需要一个命令处理模块来处理用户输入的命令,以执行相应的功能.这样整个系统的软件结构如图6所示.

模块间的横向联系表示处理的流程,竖向连接表示模块调用关系,上层模块调用下层模块.

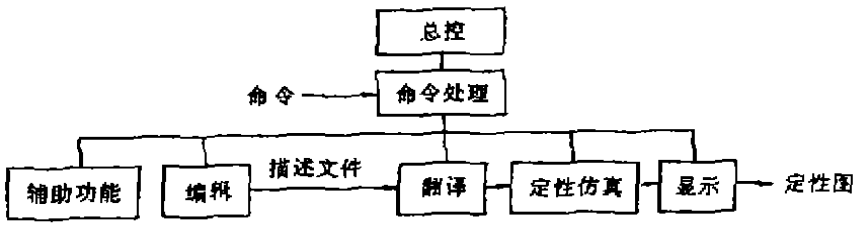


图6　GQSS总体软件结构

2.3　算法主要的数据结构

改进的QSIM算法是GQSS的关键,下面我们介绍GQSS中改进QSIM算法的设计与实现细节.GQSS的推理机制很适合Prolog语言实现,但考虑到Prolog的效率不高,而且存在和别的模块的接口问题,而C语言效率高,可移植性好,我们决定采用C语言来实现.

2.3.1　路标值集合

GQSS系统中,每个参数有一个路标值集合,路标值的基本特点是:(1)路标值间具有顺序关系;(2)在状态转换中,从一个路标值要能找到它的相邻路标值;(3)仿真过程中能够发现新的路标值,路标值集合要能扩展.根据路标值的上述特点,用一个双向链表来表示路标值是最合适的,用C代码描述的数据结构如下:

```
struct Mark_node
{
  Mark_node   *prev;   /*指向前一个路标值的指针*/
  Mark_node   *next;   /*指向前一个路标值的指针*/
  char        *name;   /*name是路标值的名字*/
};
```

2.3.2　单个参数的定性状态

每个参数的定性状态是用定性值和变化方向两部分来描述的,其中定性值通过路标值来描述,由此,我们得到描述性状态的数据结构:

```
struct Q_state
{
  Mark_node   *max, *min;   /*指向定性值上下界路标值的指针*/
  int         direction;    /*变化方向:STD(稳定),INC(增加),DEC(减少)*/
};
```

2.3.3　系统的定性状态

系统的定性状态是组成系统的各个参数定性状态的组合;另外每个系统状态都是生成的

一棵状态树上的节点，为了表示树形结构，我们采用了双亲表示法，即在每个节点中附设一个指向其父节点的指针. 系统的定性状态描述如下：

```
struct Sys_states
{
  Q_state   *state[MAX_STATES];  /*指向每个参数定性状态的指针数组*/
  Sys_states   *parent;  /*指向父节点的指针*/
};
```

2.3.4 状态转换

每个函数的一个状态转换统一描述为：

```
struct Transition
{
  Q_state   *from, *to;  /*指向转换前后状态的指针*/
  int       type;        /*转换的类型,Ii 或 Pi*/
  int       valid;       /*valid=1, 则转换有效*/
};
```

在相应的过滤算法中，将被过滤掉的状态的 valid 标志置为 0.

2.3.5 状态转换集合

每次转换中，每个参数可能不只有一个转换，最多可有 4 个转换，为表示参数的状态转换集合，定义如下的数据结构：

```
struct Trans_set
{
  Transition  trans[4];  /*状态转换集合*/
  unsigned int num;      /*实际的状态转换数目*/
};
```

2.3.6 参数间约束和元组

改进 QSIM 算法在每次状态转换结束后，根据每个约束方程形成相应的状态元组，它的数据结构设计如下：

```
struct Tuples
{
  unsigned int num;          /*元组的个数*/
  int          constraint;   /*约束的类型*/
  Trans_set    *transet1, *transet2, *transet3;  /*指向相应参数状态转换集合的指针*/
int            index[3];     /*约束的参数在参数表中的序数*/
struct Tuple
{
  Transition *trans[3];  /*元组*/
  int          valid;    /*元组的有效性标志*/
} *tuple;  /*元组集合*/
};
```

2.3.7 相邻约束

在配对一致性过滤中，需要检查每个约束的元组和它的相邻约束的元组对共享参数是否

赋予同样的转换.相邻约束用下列数据结构描述:

```
struct Adjacent
{
  Tuples   *tuples1, */tuples2;   /*指向相邻约束元组的指针*/
  int      index1, index2;        /*约束在约束表中的序号*/
};
```

2.3.8 全局变量

为了实现QSIM,我们还需要维护一些表.

(1) 系统参数表:char func_name[MAX_STATES];

(2) 输出参数表:char output_name[MAX_OUT];

(3) 参数状态转换表:Trans_set tra[MAX_STATES];

(4) 约束元组表:Tuples tup[MAX_TYPLES];

(5) 相邻约束表:adjaceng pair[MAX_PAIRS];

(6) 结束状态表:Sys_states end_table[MAX_NUM];

(7) 初始路标值表:Mark_node *landmark[MAX_STATES];

(8) 活动状态表:Stack active table.

另外,还需要4个变量分别用来存储当前参数状态和系统状态以及新扩展的参数状态和系统状态,他们分别定义如下:

```
Sys_states *cur_states, *new_states; /*当前和新生成的系统状态*/
Q_state *cur_qstate, *new_qstate; /*当前和新生成的参数状态*/
```

3 双液面系统GQSS仿真实例

图7是一个典型的双液面系统,对于输入流量Q,液箱1的高度H_1,液箱2的高度H_2.输出流量Q来说,系统处于稳态.假设变量对稳态值的变化很小,研究当输入增加到一个新值时,系统各参数的变化.

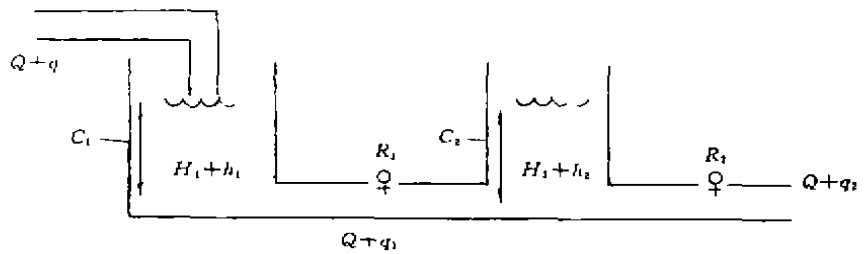


图7　双液面系统

对该系统进行定性建模,建立定性微分方程如下:

$(h_1-h_2)/R_1=q_1 \rightarrow \mathrm{ADD}(q_1,h_2,h_1)$

$c_1*dh_1/dt=q-q_1 \rightarrow \mathrm{DERIV}(h_1,dh_1);\ \mathrm{ADD}(dh_1,q_1,q)$

$h_2/R_2=q_2 \rightarrow M+(h_2,q_2)$

$c_2*dh_2/dt=q_1-q_2 \rightarrow \mathrm{DERIV}(h_2,dh_2);\ \mathrm{ADD}(q_2,dh_2,q_1)$

写成QML的语言形式：

% function：$\{h_1, \dot{h}_1, dh_1, d\dot{h}_1, q, q_1, q_2\}$

% output：$\{q, \dot{h}_1, h_1\}$

% landmark：

$h_1 = (minf, -a, 0, a, inf)$

$\dot{h}_1 = (minf, -a, 0, a, inf)$

$dh_1 = (minf, 0, minf)$

$d\dot{h}_2 = (minf, 0, inf)$

$q = (0, qin, inf)$

$q_1 = (minf, -a, 0, a, inf)$

$q_2 = (minf, -a, 0, a, inf)$

% initialization：

$q = (0, inc)$

$h_1 = (0, std)$

$\dot{h}_1 = (0, std)$

$dh_1 = (0, std)$

$d\dot{h}_2 = (0, std)$

$q_1 = (0, std)$

$q_2 = (0, std)$

% time：time_point

系统仿真结果如图8.

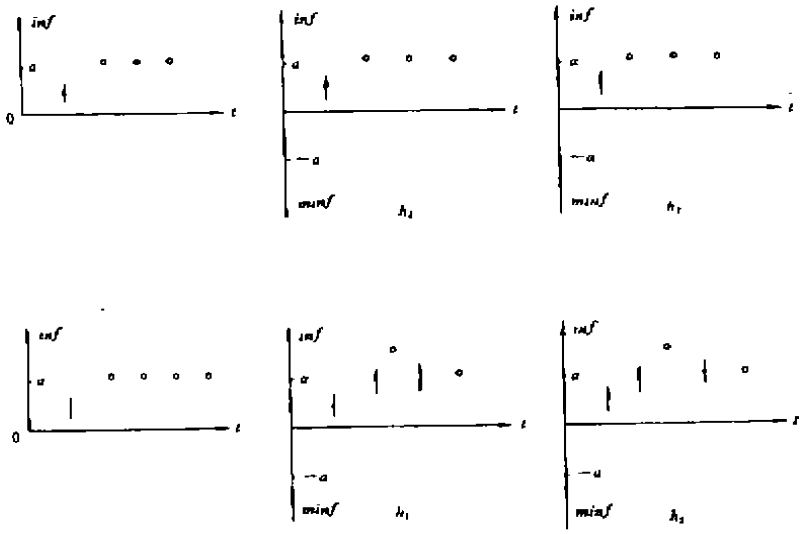


图8 定性行为

通过定性仿真，我们知道，系统将达到一个新的稳定状态. 多个结果的产生是由于仿真弱化了对系统各参数的要求，这种性质对知识获取不完全的系统特别有用. 当产生的结果太多，以至无法判断系统行为时，结合一定的定量知识减少冗余是一个有效的方法.

4 结束语

定性仿真能处理系统不完备知识，根据已知的系统不完全信息预测出系统的行为，这是传统的数字仿真无法做到的；定性仿真处理定性知识的方法是对专家系统知识处理的有益补充，能提高专家系统的处理能力和灵活性；Kuipers的动态定性仿真算法QSIM能使定性仿真快速、灵活和廉价地处理各种复杂的系统；近年来，定性仿真在Kuipers的理论的基础上又有了新的发展，尤其是定量与定性的结合方法以及复杂系统的分解理论使定性仿真理论越来越成熟，更加富有生命力. 尽管目前由于理论的局限性，定性仿真还未完全走向实用阶段，但随着定

性仿真理论的不断发展与成熟以及系统建模和专家系统构造过程中遇到的越来越多的传统方法无法解决的问题,定性仿真已经引起人们越来越多的重视.我们完全有理由相信,定性仿真在不远的将来在系统设计、分析与研究以及和专家系统构造中发挥巨大的作用.

我们应用GQSS系统对无磨擦弹簧质量系统、上抛小球、双液面系统等实例进行了仿真实验,改进的算法在消除部分冗余的前提下,较好地描述了对象行为上的差异.实验表明,GQSS不仅适用于单输入单输出系统,而且适用于多输入多输出系统.

我们设计实现的GQSS的意义在于它提供了一个灵活的定性仿真环境,全面实现并初步改进了Kuipers的仿真理论,为下一步的研究打下了坚实的基础.

今后定性仿真领域研究的主要方向将是寻求更好的定性数学工具来描述系统的定性与定量知识,使这种描述尽可能的包含足够多的系统已知信息,在定性推理方面,寻找一种更准确和效率更高的推理方法.

参 考 文 献

1 Kuipers B J. Qualitative Simulation. Artif Intell, 1986, (29)

2 Dalle D T, Kuipers B J, Edgar T F. Qualitative Modeling and Simulation of Dynamic System. Comput Chem Engng, 1988, 12(9)

3 Forbus K D. Qualitative Process Theory. Artif Intell, 1984, (24)

4 Kuipers B J. Qualitative Simulation: Then and Now. Artif Intell, 1991, (59)

5 Kuipers B J. Qualitative Simulation: Reasoning with Qualitative Models. Artif Intell, 1991, (59)

6 De Kleer J, Brown J S. A Qualitative Physics Based on Confluence. Artif Intell, 1984, (24)

7 Liver Dordan O. Mathematical Problem Arising in Qualitative Simulation of a Differencial Equation. Artif Intell, 1992, (55)

8 Catino C A, Grantham S D, Ungar L H. Automatic Generation of Chemical Process Units. Computers Chem Engng, 1991, 15(8)

9 Andrea Bonarini, Vittorio Maniezzo. Integrating Qualitative and Quantitative Modeling. International Journal of Expert Systems, 1991, 4(1)

10 白方周,蒋 鑫,鲍忠贵.动态系统的定性推理:定性模型的建立与定性仿真方法.信息与控制,1995,24(4):222~229

A QSIM-BASED GENERAL QUALITATIVE SIMULATION SYSTEM (GQSS)

BAI Fangzhou CHEN Yuan BAO Zhonggui

(Computer Dept, University of Science and Technology of China)

Abstract Qualitative simulation is a new direction in the area of current AI and simulation. This paper mainly introduces a experimental general qualitative simulation system (GQSS) based on QSIM algorithm, and the QSIM algorithm in the system has been improved.

Key words qualitative simulation, artificial intelligence, algorithm

作者简介

白方周,男,61岁,教授.研究领域为复杂工业过程的计算机控制,智能控制,定性建模和定性仿真.

陈 源,25岁,硕士生.研究领域为人工智能.

鲍忠贵,25岁,硕士生.研究领域为建模与仿真.