上海地铁 ATS 仿真系统进路自动排列的设计与实现

吴 江,郭秀清
(同济大学 控制科学与工程系,上海 201804)

摘 要:进路自动排列(ARS)能够大大减轻操作员的工作量,提高列车运行效率。以上海地铁 5 号线列车自动监控仿真系统(ATS)为例,通过对其自动进路排列设计进行分析,结合标准模板库 STL 给出了一种进路搜索算法,并在这种算法的基础上提出了一种自动排列进路的方法,较好地实现了进路的自动排列功能。

关键词:进路自动排列;仿真;列车自动监控系统;进路搜索算法

中图分类号:TP319 **文献标识码**:A **文章编号**:1674-7720(2012)17-0007-04

Design and implementation of automatic route setting of ATS simulation system of Shanghai metro

Wu Jiang, Guo Xinqing
(Department of Control Science and Engineering, Tongji University, Shanghai 201804, China)

Abstract: Automatic route setting(ARS) can greatly reduce the operator's workload, improve the efficiency of train moving. The article take the ATS simulation system of line 5 as an example, after analyzing its route arrangement design, propose a route searching algorithm combined with STL and give an automatic route setting method based on this algorithm, through this method, ARS works well.

Key words: ARS; simulation; ATS; route searching algorithm

近年来,轨道交通快速进入高速期,成为带动经济增长的重要因素。列车自动监控系统(ATS)是一种智能化自动监控系统,对 ATS 系统能否进行正确的操作,将影响到列车能否安全运行。这对轨道交通运营管理人员的后勤培训提出了很高要求,因此 ATS 仿真系统应运而生。

若 ATS 仿真系统采用人工排列进路,对操作员的业务素质将会提出很高要求,而且操作量大、效率低。而自动进路排列功能的实现将大大降低操作员的工作量,减小失误率,提高 ATS 系统的效率。

1 自动进路排列的设计

自动进路排列的工作原理为:当一列车步进到一个特别配置的轨道区段时,即触发排列下一条进路的指令。这些特别配置的轨道区段被称为“运营触发点”,运营触发点接近于即将被排列的进路。列车的位置可由列车追踪功能获取,因为事先已经把进路的信息保存在文件中,下一条进路即可以从文件中获取;然后将进行进路一致性检查。如果检查表明,没有理由不排列该进路,就向系统联锁模块发出一个指令,锁定进路中元素;最后进行进路排列检验,若没有问题,则开放始端信号灯。进路自动排列 ARS(Automatic Route Setting)请求处理步骤如图 1 所示。

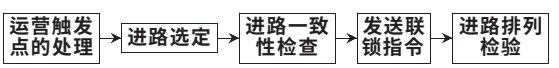

图 1 ARS 请求处理步骤

为了能详细分析自动进路排列理论,本文以如图 2 所示的上海地铁 5 号线的部分线路图为例进行说明。

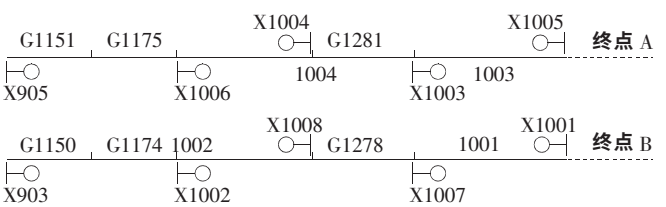

图 2 上海地铁 5 号线的部分线路图

1.1 运营触发点处理

在运营触发点的处理上，选择一条进路的最后一条区段为下一条进路的运营触发点，如图 2 中进路 X905~X1006，下一条进路的运营触发点就是这条进路的最后一条区段 G1175。当判断列车到达 G1175 后，则发送要求排列下一条进路的指令。

运营触发点需要拥有一定的信息（如触发点触发的有效方向），本文把这部分信息采用 XML 纯文本存储。XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，层次结构清晰，易于读写与共享。

下面是运营触发点的数据结构，采用 XML 纯文本保存。

```
<TriggerPt>
    <ID>1</ID>
    <Name>TgPt1</Name>
    <Owner>G1175</Owner>
    <TriggerDirection>R</TriggerDirection>
</TriggerPt>
```

其中，标记<Owner>存储了分配给运营触发点的轨道区段，标记<TriggerDirection>存储了运营触发点有效时列车的运行方向。

1.2 进路的选定

在运营触发之后，ARS 功能将为这一列车选定拟排列的进路。从图 2 中可以看出，若从 X905 排列一条上行进路，这条进路是存在并且是唯一的，依次经过 G1151 区段，G1175 区段到 X1006。但若从 X1002 出发排列一条进路，进路虽然存在但却不是唯一的，分别为经过 1002 道岔反位，然后接 1004 道岔反位，经 G1281 区段到达信号灯 X1003 和经过 1002 道岔定位，再通过 G1278 区段到达信号灯 X1007，因此仅一个始端信号灯还不足以构成选定一条进路的条件。为此，需要从列车追踪功能传输出来的车次号中获取列车的目的地代码，从而获取列车运行方向。

在确定了始端信号灯（触发后可获取，见图 6 中各表关系）和列车运行方向后，为了能够让进路搜索程序搜索到符合条件的进路，可以建立一个适合搜索的并且能够真实形象地反映现实路线结构的数据结构。可以构建一棵二叉树来表示信号机与它的直接邻居之间的连通关系。每个信号机均构建一棵二叉树，然后把整个站场的所有信号机构建的二叉树组织起来。若把上例中信号灯 X905 和 X1002 的二叉树建立起来，其结果如图 3 所示。

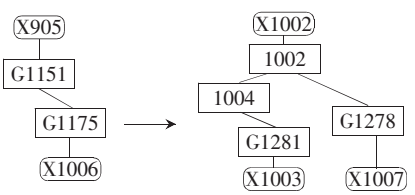

图 3 X905 和 X1002 所建立的二叉树

图中椭圆表示信号灯，矩形表示轨道或道岔，图 3（a）表示道岔，图 3（b）表示轨道。在图 2 中，假设进路从 X1002 出发，终点站为 B，则进路的选定存在以下几种情况：

（1）若全部轨道正常，那么从 X1002 结合方向搜索，会建立到 X1007 的进路。

（2）若存在以下的特殊情况，从 X1002→X1007 的进路不能正常建立，则 ARS 将改变进路的选定。

①若进路中存在长期障碍，阻止了正常进路的自动排列。如图 2 中 1002 道岔被锁定在反位状态，正常的进路 X1002→X1007 将不能建立，此时进路自动排列功能将会去变更进路，并将访问图 3（a）子树，选定从 X1002 经 1002 道岔反位，接 1004 道岔反位，通过 G1281 区段到达信号灯 X1003 这条进路，然后经 1003 道岔反位到终点 B。

②若进路中存在短期障碍，比如此时 G1278 上正好被占用（如停着一辆车），正常的进路排列被阻止，那么自动排列功能将试图排列越行进路，同样会去访问图 3（a）子树，选定从 X1002 经 1002 道岔反位，接 1004 道岔反位，通过 G1281 区段到达 X1003 这条进路。

（3）若道岔 1002 出现了故障，进路将不能排列。

若把图 3 中进路的路径抽取出来，则很容易就得到优化二叉树，如图 4 所示。

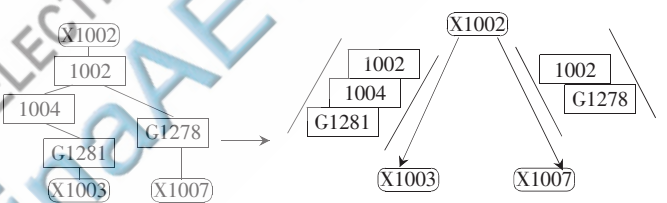

图 4 X1002 优化二叉树

由图 4 提取信息，可以建立每条进路的数据存储结构。本设计采用 XML 来存储每条进路的结构信息，下面是 X1002→X1007 的进路用 XML 保存的数据结构。

```
<Route>
    <ID>36</ID>
    <Name>X1002-X1007</Name>
    <StartSignal>X1002</StartSignal>
    <EndSignal>X1007</EndSignal>
    <RouteBlocks>
        <Axle>G1278</Axle>
    </RouteBlocks>
    <SwitchList>
        <Switch>
            <Name>1002</Name>
            <State>DW</State>
        </Switch>
    </SwitchList>
</Route>
```

其中，<StartSignal>表示进路的始端信号灯，<EndSignal>

表示进路的终端信号灯，<Axle>表示进路中的区段，<Switch>表示进路中的道岔。把线路图中的所有进路都用这种数据结构表示出来，放在一个 XML 文件中，以供程序查询。这样通过以始端信号灯结合方向，用方向来确定道岔的定/反位，就能选定下一条进路。

1.3 进路一致性检查

在进路选定后，接下来即进行进路的一致性检查。进路一致性检查的目的是要防止不能被执行的指令被传送至联锁。进路一致性检查包括如下步骤：

(1)检查请求是否已被执行

如果拟排进路的始端信号机已处于开放状态，说明操作员已经为列车人工排列了进路，ARS 功能会中止此 ARS 请求，并记录该操作。如图 2 中，若已经在步骤 2 中选定了进路 X1002→X1007，那么此时就应该检查一下此进路有没有已经被排列。若之前已经被操作员手动排列了进路，则此时这条进路就不需再自动排列了。

(2)检查指令输出是否存在短期障碍

为此，需调查拟排进路的始端与终端要素之间的所有轨道要素以判断是否其中某个元素存在障碍。

1.4 发送联锁指令

在进路可用性检查成功后，即可输出联锁指令。发送联锁指令将锁定进路中的道岔、区段和交叉，以防再被其他进路征用。

首先，系统检查该列车是否仍在拟排定进路的接近区段。如果列车已不在拟排定进路的接近区段，ARS 将中止此 ARS 请求；如果列车仍在拟排进路的接近区段，则排列该进路的指令将送至相应的联锁。ARS 功能只把下一个进路排列指令传送到该联锁，只要它已经接收到对此进路排列请求的肯定确认。

1.5 排列检查

指令输出之后，自排进路功能等待来自计算机联锁控制系统的肯定确认。作为肯定确认，对每条进路来说就是开放始端信号机。只要信号机一开放，该 ARS 请求立即终止。

经过以上 5 步后，进路的自动排列已经基本完成。图 5 为进路自动排列流程图。

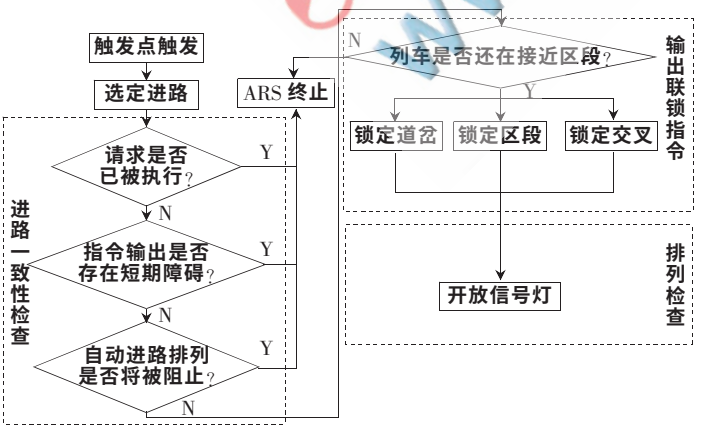

图 5 自动进路排列完整流程图

2 自动进路排列的软件实现

将信号灯、区段、道岔的信息用类似 XML 数据结构存储，每类轨道元素都分别存放在各自的 XML 表中，这样就存在 5 份 XML 表，在本设计中，本质上 XML 就充当了一个小型数据库的角色。表 1 为各个轨道元素在 XML 中的存储信息。

表 1 各元素数据结构的 XML 存储表

进路元素	运营触发点 (TriggerPt)	进路 (Route)	信号灯 (Signal)	区段 (Axle)	道岔 (Switch)
元素存储结构信息	名字 Name	始端信号灯 StartSignal	名字 SignalName	名字 AxleName	名字 SwitchName
	拥有区段 Owner	终端信号灯 EndSignal	接近区段 JJAxle	左端元素 LeftEle	定位端元素 DWEle
	有效方向 TriggerDirection	进路中的区段 Axle	离去区段 LQAxle	右端元素 RightEle	前向端元素 PreEle
		进路中的道岔 Switch			反位端元素 FWEle

注：表中忽略了各元素在 XML 中不重要的结构信息(如 ID)

在开发过程中，需要读取保存在 XML 中的轨道元素的信息，因此设计中对应 XML 中轨道元素的信息为每个轨道元素都建立了一个封装类，如 Switch 封装类结构如下：

```
class Route
{
 public:
  Route (CString ID, CString Name, CString StartSignal, CString ZDXH, RouteQDArray RouteBlocks, RouteDCInfoArray DCInfo);
  ~Route();
  CString          m_ID;
  CString          m_Name;              //进路名
  CString          m_StartSignal;       //始端信号灯名
  CString          m_ EndSignal;        //终端信号灯名
  RouteQDArray     m_RouteBlocks;       //因为一个进路中
                                        //可能有很多区段，所以保存在数组
  RouteDCInfoArray m_DCInfo;
                                        //道岔，同样保存在数组中
  Bool             m_faultflag;         //故障标志
  ……
};
```

把保存在 XML 表中的轨道元素信息用 XML 解析类 CMarkUp 解析后，用解析出来的各轨道元素存储信息去构造一个对应的类。因为线路图中存在很多信号灯、区段等轨道元素，而每一个都对应着自己的一组信息，也就是每一个元素都可以构造一个类，很好地实现了 XML 数据与对应类的绑定。为了方便查询和使用，把相同轨道元素的类保存在 STL 的 Vector 数组中，这样就分别有运营触发点、进路、道岔、区段、信号灯 5 个 Vector

数组。每一类轨道元素都是相互联系的，因此，在程序中需要通过一类元素获取到另一类的信息（如需要查询始端信号灯 StartSignal 获取到一条进路 Route）。图 6 所示为 5 个轨道元素的 XML 表联系图。

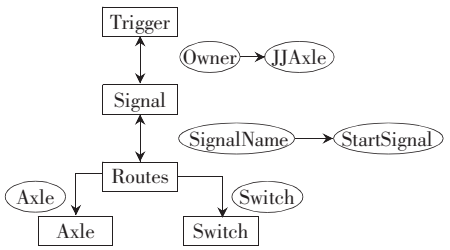

图 6 各 XML 表联系图

图中，矩形表示各个 XML 表，椭圆形表示 XML 表的某个轨道元素的其中某个存储信息。两个 XML 的联系就是通过寻找某个有相同的某个存储信息来实现的。如 Owner 和 JJAxle，因为每个 Trigger 都对应着一个拥有区段，因此可以用此区段去对比 Signal 表中的 JJAxle 信息。若找到此信息相同，即可找到下一条进路的始端信号灯。根据以上的轨道元素数据结构和 XML 表联系图，给出选定进路的伪码算法如下：

```
Function SearchNextRoute(……)
{
 Then OwnerAxle=GetOwnerQD()        //当符合触发条件
          //后，从列车跟踪模块获取列车所在的区段，
                    //即触发点拥有区段 OwnerAxle
    XHIterator=FindSignal(OwnerAxle)      //利用获取的
    //OwnerAxle 作为 JJAxle 去查找 Signal 数组 Vector 中
          //查找到相应的关联类，返回这个类的迭代器
If(XHIterator=SignalVector.end) then  return;
                                  //如果未找到，则返回
RouteIterator=FindRoute(*(XHIterator)->SignalName)
    //利用上面查找到的信号灯类获取此信号灯的名字
     //然后以此为关键字查找进路，返回进路的迭代器
If(RouteIterator =RouteVector.end) then return;
                        //如果未找到进路，则返回
AxleIterator=FindAxle(*(RouteIterator)->Axle)
    SwitchIterator=FindSwitch(*(RouteIterator)-> Switch)
    //利用查找到的进路类获取此进路中的道岔、区段、
    //获取到它们相应的类。这两个类的获取主要用于
          //后面的进路一致性检查和区段，道岔的锁定
 }
```

在以上伪码中，最重要的就是查找算法。本文很好地利用了 STL 的非变异算法 find_if 来查找进路元素。

因为要每隔一定时间就去判断列车运行距离来判断列出是否到运营触发点，所以在定时

器响应函数来判断是否去开放下一条进路，这样通过定时器的方法也就达到了进路自动开放的效果。自动排列进路源码如下：

```
Function OnTime(……)
 {
   If(Direct&&Location)
                  //判断方向和列车位置有没有到触发点
     linRet=SearchNextRoute()
                  //如果到达触发点，则查找下一条进路
   if(linRet)
     linRet1=CheckValid()
                  //若查找到进路，则进行一致性检查
   if(linRet1)
     linRet2=LockGDElement()  //一致性检查没问题，
                  //则发送联锁指令，锁定轨道元素
   if(linRet2)
     linRet=OpenSignal()//都没问题后，则开放信号灯
}
```

通过以上方法，顺利地实现了列车进路自动排列，在所截取上海地铁 5 号线的部分线路图上实现结果如图 7 所示。

图 7 自动进路排列效果图

由图中可以看出，当列车开进区段 G1278，并未人工开放进路，下一条进路就被正确地开放了，证明了本文所提出的方法的有效性。

本文提出了一个 ATS 仿真系统的自动进路排列的方案，阐述了自动进路排列的大体过程。在这个过程中，进路的选定尤为重要，应用自动进路排列，可减轻操作员的劳动量和减少出错率，提高系统的运行效率，从而可以有效地提高培训效率。

参考文献

[1] 潘国梅，梅登华.广州地铁微机联锁仿真培训系统进路排列、联锁和进路解锁算法及实现[J].装备制造技术，2007(8)：82-84.

[2] 王野，郭秀清.基于组件技术的列车自动监控仿真系统开发平台[J].计算机应用，2007，27(S2)：286-288.

[3] 赵根苗，陈永生.基于三层分布式架构的列车自动监控仿真培训系统的分析与设计[J].微型电脑应用，2003，19(12)：35-37.

（收稿日期：2012-05- ）

作者简介：

吴江，男，1988 年生，硕士研究生，主要研究方向：城市轨道列车运行仿真。

郭秀清，女，1965 年生，硕导，副教授，主要研究方向：过程控制与计算机控制。