基于区域分割的交通仿真死锁处理算法研究

赵 春,方 敏
(四川大学锦城学院 计算机科学与软件工程系,四川 成都 611731)

摘 要:多智能体交通仿真系统出于减少通信量的目的,在采用虚拟交警对交通区域实施空间分割后,局部区域中的智能体不可能也无法感知处于不断动态变化的整体交通环境,也就无法适时地疏导交通,以致会引起系统中的某些控制节点出现死锁,进而会引发全局死锁效应。为此,采用基于触发器消息的智能体通信机制,通过在发生死锁时系统发送专用的死锁消息给处于死锁节点处负责调度指挥的智能体,使其在提供的死锁消息处理函数中采用基于队列结构的运动物体调度策略来解除死锁,进而实现死锁节点处运动物体之间的避让效果,以解决控制节点处的死锁现象。仿真实验结果表明,所提出的死锁处理算法可有效处理多智能体交通仿真系统中的控制节点死锁现象,能在最大程度上提高实时交通自动控制的效率和效益。

关键词:仿真;智能体;死锁;调度
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2017)05-0025-05
doi:10.3969/j.issn.1673-629X.2017.05.006

Investigation on Deadlock Resolution Algorithm for Traffic Simulation with Region Segmentation

ZHAO Chun, FANG Min
(Department of Computer Science and Software, Jincheng College of Sichuan University, Chengdu 611731, China)

Abstract: After the whole traffic region is partitioned by the virtual traffic police for the purpose of reducing traffic in the multi-agent traffic simulation system, and the traffic cannot be directed in time because the dynamic overall traffic environment is unobserved for those agents in local region. For this reason, those regional deadlocks will be triggered in some control nodes, which result in the global deadlock effect. To solve this problem, using the communication mechanism based on trigger message, the system sends a specialized deadlock message to the agent responsible for the traffic command of the control node when deadlock occurred. By using the method of dispatching moving objects based on queue structure in the message processing function for deadlock resolution, avoidance between the moving objects can be achieved to solve the problem of the deadlock. The simulation results show that the proposed algorithm can effectively settle those deadlocks at the control nodes in the multi-agent traffic simulation system, and farthest promote the efficiency and benefit of dynamic traffic auto-control.

Key words: simulation; agent; deadlock; dispatching

1 概 述

基于智能体的计算机仿真技术采用“自底向上”的研究策略,用智能体模拟现实系统中主体的行为和它们彼此间的相互作用,从而模拟出系统的整体性质和演化过程[1]。这样的研究方法与交通系统微观仿真研究所遵循的“由部分到整体”的思路基本一致。地面交通仿真需要每个运动物体拥有主动发起从起点到终点的行驶行为,并在行驶过程中能够感知外部交通环境的变化,及时做出适当反应[2]。智能体代表了现实世界中具有智能性的自治实体[3]。基于智能体的交通仿真方法是采用智能体对交通系统底层中的各组成元素(如运动物体、行人、交通灯、交叉口和路段等)实施仿真建模,并通过这些智能体固有的行为方式和彼此之间的相互影响和作用来模拟整个交通环境[4-6]。多智能体系统主要研究智能体之间的相互协作与竞争[7-8]。智能体通过通信行为彼此联系和影响。智能

收稿日期:2016-05-26 **修回日期**:2016-09-08 **网络出版时间**:2017-03-07
基金项目:四川省应用基础项目(2010JY0023)
作者简介:赵 春(1978-),男,硕士,副教授,研究方向为数据库技术、互联网应用。

体之间的通信方式在很大程度上影响着系统整体的信息复杂度[9]。

采用触发器消息的智能体通信是一种有效的通信方式。根据通用触发器系统模型,智能体在完成某一动作后触发对应的触发器,发出特定消息。其他智能体都能通过触发器消息感知到外部环境的变化,并做出反应。集中化的触发器系统能够根据触发器消息的优先级和影响范围对其进行过滤,从而保证智能体对高优先级消息的优先处理[10]。然而系统在通信过程中的信息量依然很大。选择给部分智能体赋予类似交警的行为特征,让其负责控制和调度局部交通,从而实现对整个交通系统空间的区域分割,是一种能够切实降低整体通信量的有效策略[11]。

但在采用区域分割降低整体通信量的同时,却会给交通系统带来附加的影响。即将会导致单个智能体对于整体交通环境的不可感知,以致无法根据道路的实时使用状况做出适当的反应来调整自己的行驶路线,从而出现死锁现象,即多个运动物体在同一道路中出现阻塞的情况。

区域分割策略导致了这种局部死锁现象的形成。结合典型的控制节点死锁情况,提出采用队列结构来分别表示驶入车辆与驶出车辆集合的方法,从而重新定义了负责指挥交通的智能体结构,设计出了死锁的判定算法;并通过对常用避让算法的对比分析,提出了一种新的死锁处理算法。该算法基于触发器的智能体通信机制,创建了专用的死锁消息和消息处理函数;发生局部节点死锁时,系统发送消息给节点处负责控制调度的智能体,使其通过消息处理函数对经过该节点的运动物体实施有效调度,以解除死锁,从而达到了避免因局部死锁而引发全局死锁的效果。

2 交通仿真中的死锁

对于地面交通的计算机仿真,交通环境可以看作若干个离散时间点上的环境状态的集合:$E=\{e_{t0},e_{t1},\cdots,e_{tm}\}$。运动物体需要在每个时间点上感知当前的环境状态,并做出适当反应。如果将运动物体所能做出的反应用集合 C 表示,则运动物体(MO)就可以表示成函数:MO:$E\rightarrow C$。要为函数 MO 提供输入,就需要运动物体在每个离散时刻都要感知到外部环境,获得在各个时刻的环境状态。定义 $S(\mathrm{MO}_i,t)$ 表示第 i 个运动物体在时刻 t 的状态信息,则在一个包含 n 个运动物体的交通环境中,第 k 个运动物体在 t 时刻所需要感知的外部环境为:$e_t=<S(\mathrm{MO}_0,t),S(\mathrm{MO}_1,t),\cdots,S(\mathrm{MO}_n,t)>$。如果认为每个运动物体在每个离散时刻的状态信息为一个单位的信息量,则在每个离散时刻由 n 个智能体构成的多智能体系统中所需传输的信息总量为 $n\times(n-1)$,即信息量为 $O(n^2)$。为降低每个离散时刻需要传递的信息量,可以通过给部分智能体赋予类似交警的行为特征,让其负责指挥和调度局部交通,从而对整个交通区域进行分割,以分化需要传输的信息量[12-13]。

在整体的交通空间被分割成为若干小的区域以后,运动智能体不再需要感知其他运动物体,而是仅仅只需要感知负责指挥其归属区域的智能体所发出的指令[14]。因此,运动物体可以重新表示为 MO:Order→AC,Order 表示指挥智能体所发出的指令集合。如果把每个运动物体在每个离散时刻的状态信息和每个指挥智能体所发出的指令都视为一个单位的信息量,则在一个包含 n 个运动物体的交通系统中,每个离散时刻所需传输的信息量可降低为 $O(n)$。

区域分割策略虽然能够有效降低每个离散时刻需要传递的信息量,但这种空间分割策略却造成了每个参与交通的智能体只能对自己所处的局部环境进行感知,而不能获得整个交通环境的全部信息。因此,整个环境对于单个智能体是不可观察的。而车辆的运动、避让和速度变化等动作在时刻改变着交通环境的状况。由于环境的不可观察性和动态性,造成交通系统在运行过程中会出现死锁现象。这种死锁现象的显著特征表现为道路上的某个运动物体会一直被要求重新寻路,或者某个路口的各条道路上的运动物体因为互为障碍而出现阻塞。

交通仿真系统中的死锁问题虽然是一个全系统的死锁问题,但是往往是由于系统中某个控制节点的死锁引起的全局效应。对于一个控制节点,最常见的死锁情况如图 1 所示。

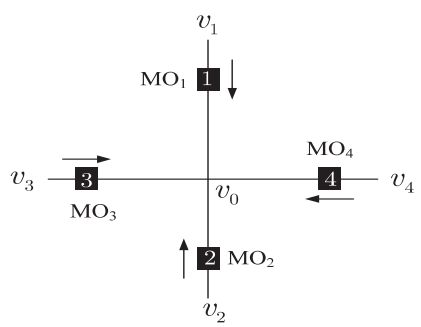

图 1 一个控制 VP 处形成的死锁现象

MO_1、MO_2、MO_3、MO_4 的预定路线分别为 $r_1(v_1,v_0,v_2)$,$r_2(v_2,v_0,v_1)$,$r_3(v_3,v_0,v_4)$ 和 $r_4(v_4,v_0,v_3)$。位于 v_0 处的负责调度指挥的智能体(VP)所管辖的四条道路上都有运动物体驶入。此时 VP 将会检查每一个将要进入的运动物体是否能按照预定路线通行;如果不能,则使用空闲道路来实施避让。由于每条道路最多只允许一个运动物体通行,且 VP 管辖下的四条道路都被驶入的运动物体占用,因此 VP 无法使用空闲

道路对任何一个运动物体实施避让,从而造成死锁。

在交通模型中,每一个负责指挥交通的智能体 VP 都拥有结构体变量 roadRecord,它保存了该 VP 管辖下所有道路的相关信息:

```
struct roadRecord
{
int nodeId; //道路远端节点的 ID
point nodePos; //远端节点的坐标
double roadWidth; //道路宽度
double roadLength; //道路长度
int roadType; //道路类型
bool singleWay; //该道路是否分时单行
moList inQueue; //驶入的车辆队列
moList outQueue; //驶离的车辆队列
//关于该条道路的地图信息
vector<ctrl_node_record>vpath;
}
```

moList 采用队列结构,记录了道路的驶入与驶离车辆信息,定义如下:

```
struct moList
{
int moNum; //队列中车辆的数量
//队列中车辆的最大宽度
doublemaxWidth;
//队列中的车辆记录
std::vector<registeredMo * > mos;
}
```

可以注意到,这种死锁现象的一个重要特点就是对于 VP 而言,它管辖下的所有道路的道路记录 roadRecord 中的结构体 InQueue(驶入的车辆队列)都不为空,没有一条道路上的 OutQueue(驶离的车辆队列)是可用的。需要说明的是,考虑到每条道路最多只允许有一个运动物体通行,因此每条道路的 InQueue 队列与 OutQueue 队列不能同时可用。为处理死锁现象,可首先根据这种死锁现象的特征来设计算法,判定死锁是否产生。

算法 1:判定死锁

```
begin
lock←true //初始化死锁标志
//获取 VP 控制的道路数量
length←vp. roads. length
//遍历 VP 控制的所有道路
while length>0
//判断道路的驶入队列是否为空
ifvp. roads[length]. inQueue=null
//道路驶入队列为空,解除死锁标志
lock←false
break
else
length←length-1
end
```

3　死锁解除算法

在基于触发器消息的多智能体交通仿真系统中,智能体对外部环境的获得,由主动感知变为被动感知。只有当外部环境发生变化时,智能体才会被触发器消息通知。而集中化消息触发器所具有的分发机制,使得环境信息的传播由整个区域内的广播变为点到点的传播。

如果出现道路阻塞,运动智能体一般可以根据实际情况采取以下两种避让算法中的一种来处理死锁问题。

(1)假定运动物体 MO_1 从节点 n_i 到节点 n_j,途经控制节点 n_k,则行驶路径可表示为 $r_1(n_i,n_k,n_j)$;运动物体 MO_2 由节点 n_j 到节点 n_h,同样途经控制节点 n_k,则行驶路径可表示为 $r_2(n_j,n_k,n_h)$。MO_1 在 t_1 时刻到达 n_k 节点,MO_2 在 t_2 时刻到达 n_k 节点。设 $W(x)$ 表示 x 的宽度,如果 $t_1<t_2$,且 $W(n_j,n_k)<W(MO_1)+W(MO_2)$,则 MO_1 和 MO_2 必然相撞。因此为避免死锁,在 MO_1 到达 n_k 后,需要等待 t_2-t_1,才能继续运行。

(2)运动智能体 MO_1 和 MO_2 的行驶路径分别为 r_1 和 r_2。如果 r_1 和 r_2 的方向相逆、宽度相同,且道路宽度小于 MO_1 和 MO_2 的宽度之和,即 $(r_1=-r_2)\Lambda(W(r_1)=W(r_2)<(W(MO_1)+W(MO_2)))$,则 MO_1 和 MO_2 必然相撞。因此 MO_1 或 MO_2 必须重新选路,以避免相撞。

由于每条道路最多只允许有一个运动物体通行,因此在一条道路上不可能出现多个运动物体同时逆向行驶的现象。在这种情况下,上述两种在多智能体交通仿真系统中常用的避让算法都是无效的。为此,可以创建一个专用的死锁消息和消息处理函数。在产生死锁时,系统对位于产生死锁节点位置处的 VP(位于 v_0 处)发送一个死锁消息,然后由 VP 来负责调度运动物体,从而解除死锁。对应的消息处理函数的逻辑主要包括:

(1)VP 给管辖内所有道路 InQueue 队列中的 MO(运动智能体)发送停止消息,MO 全部停止运动。

(2)VP 获取并判定所有 MO 的优先级(假定 MO_1、MO_2、MO_3、MO_4 的优先级依次降低),发送消息让优先级最高的 MO_1 驶入,在距离 v_0 较近处停止等待。

(3)VP 在 MO_3 与 MO_4 之间选择优先级较高的 MO_3,将其暂时地从道路 (v_0,v_3) 的 InQueue 队列中删除。

(4)MO_2 驶向 v_0,在离 v_0 较近处转向 v_3;并将 MO_2

从道路(v_0,v_3) 的 OutQueue 队列中删除,进入道路(v_0,v_3) 的 InQueue 队列。

(5)恢复之前暂时删除的 MO_3 到道路(v_0,v_3) 的 InQueue 队列中。

(6)MO_1 继续驶入,顺利通过 v_0。

此时死锁解除,剩下的其他运动物体则可以在 VP 的调度下利用 MO_1 通过 v_0 以后空闲出来的道路实施避让,从而均顺利通过 v_0。

算法 2:解除死锁

```
begin
//停止所有运动物体的移动
call stopMOS( )
//判定 VP 控制的所有道路的驶入队列是否为空
while vp. roads. inQueue<>null
//选择优先级高的运动物体驶向 VP
call hpMODriveToVP( )
//清除优先级高的运动物体通过 VP 的阻碍
call clearObstruction( )
//让优先级高的运动物体通过 VP
call hpMOPassVP( )
end
```

4 算法效果

图 2 显示处于 v_0 处的 VP 发现了自己处于特殊的死锁状态。这时它向所有行驶在自己管辖内的四条道路上的车辆发送停止驶入消息,四辆车停止了行驶。图 3 显示 v_0 处的 VP 让优先级最高的车辆 MO_1 驶入,并在到达后停止。

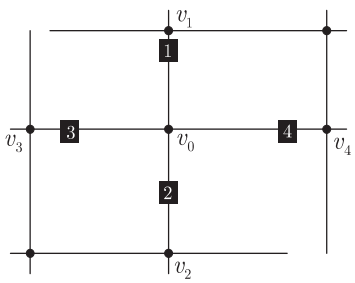

图 2 四辆车同时驶向 v_0 形成死锁

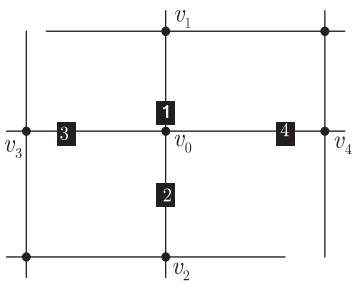

图 3 优先级最高的 MO_1 最先驶向 v_0

图 4 表示 v_0 处的 VP 将车辆 MO_3 从道路(v_0, v_3) 的 InQueue 队列中暂时删除(这种删除不影响车辆在地图上的显示,只是车辆暂时不在其道路的 InQueue 队列中),并且同时给车辆 MO_2 一条特别的消息,让其驶向(v_0, v_3) 并且在距离 v_0 较近处掉头;然后将 MO_2 加入到(v_0, v_3) 道路的 InQueue 队列中;之后恢复 MO_3 到(v_0, v_3) 道路的 InQueue 队列中。至此死锁解除,v_0 处 VP 的消息处理函数结束。处于 v_0 控制下的四辆车可以利用简单的避让算法顺利通过 v_0。

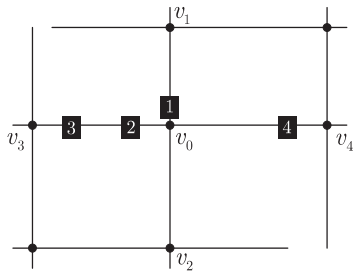

图 4 MO_2 驶入道路 (v_0,v_3) 并掉头

图 5 表示在阻碍被清除的情况下,v_0 处的 VP 恢复车辆 MO_1 的运动状态,车辆通过 v_0 进入了(v_0, v_2) 驶向 v_2。在此之后,车辆 MO_2 驶过 v_0 进入(v_0, v_1);MO_4 进入道路(v_0, v_1) 实施避让,MO_3 通过 v_0;最后 MO_4 通过 v_0。至此,四辆车全部通过了控制点 v_0。

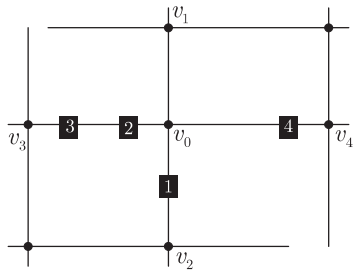

图 5 MO_1 通过 v_0

在死锁处理过程中,还可能会出现一些更为复杂的情况。比如在某个 VP 管辖的四条道路中,其中有一条道路之前是空闲的,其他三路的车辆在接近控制点 v_0 时,突然在先前空闲的道路上出现一辆车向 v_0 点驶来。这种情况说明位于 v_0 处的 VP 在消息处理函数中,首先应该计算需要临时驶入车辆并掉头的道路是否满足驶入车辆的要求;如果发现不满足,VP 可以命令在这些道路上的车辆先都向远离 VP 的方向倒退最远距离,以保证如上例中的车辆 MO_2 能够临时进入道路(v_0, v_3)。

如果发现用来暂时驶入的道路比较短,不能满足两辆车同时停下(其中一辆还要掉头,而调头比停止需要占用更大的道路长度),或者需要让路的道路,如上例中的(v_0, v_3) 线上驶入的车辆太多,需要做的处理太多时,处理函数可以采用一种简单的处理办法。比如发消息让(v_0, v_2) 道路上所有 InQueue 队列中的车辆调头,让它们在自动寻路系统的支持下重新寻路。而其他道路的车辆则在系统的调度下依次通过 v_0 处的 VP。

5　结束语

针对交通仿真系统中可能出现的死锁现象,通过进行多方位分析,提出了建设性的解决思路,并优化设计算法,在基于系统结构可嵌入的原则下,实现了一种新的死锁处理算法,使得交通仿真系统能够避免大多数的死锁情况发生。

交通的实际情况总是千变万化。对于交通仿真系统来说,所提出的算法可以根据出现的新情况新问题研究相应的对策,最大程度上保证交通管理在不需要用户干预的情况下,自动、高效地完成交通的自动控制任务。

参考文献:

[1] Roozemond D A. Using intelligent agents for pro-active, real-time urban intersection control[J]. European Journal of Operational Research, 2001, 131(2): 293-301.

[2] Chaib-Draa B, Moulin B, Mandiau R, et al. Trends in distributed artificial intelligence[J]. Artificial Intelligence Review, 1992, 6(1): 35-66.

[3] 张飞舟,曹学军,孙　敏. 基于多智能体的城市交通集成控制系统设计[J]. 北京大学学报:自然科学版,2008,44(2):289-292.

[4] 郭建钢,伍雄斌. 多智能体技术在交通系统协调控制中的应用[J]. 华东交通大学学报,2005,22(6):38-41.

[5] 吴继伟,杨定鹏,萧蕴诗. 多智能体协作方法及其应用研究[J]. 控制与决策,2004,19(2):216-218.

[6] Almejalli K, Dahal K, Hossain A. An intelligent multi-agent approach for road traffic management systems[C]//Control applications, intelligent control. [s. l.]: IEEE, 2009: 825-830.

[7] 何　涛,白振兴. 多智能体系统设计的关键技术研究[J]. 现代电子技术,2006,29(14):31-34.

[8] Hirankitti V, Krohkae J, Hogger C. A multi-agent approach for intelligent traffic-light control[J]. World Congress on Engineering, 2007, 79(3): 116-121.

[9] 王龙飞,陈　红,李　扬,等. 多智能体在城市交通系统中应用现状综述[J]. 计算机系统应用,2010,19(1):198-203.

[10] 史　乐,李　辉,原江波. 基于消息通信的多智能体系统的应用[J]. 计算机应用,2008,28(2):531-534.

[11] 贺　雷,刘正熙,毋攀良. 基于通用触发器系统的地面交通仿真[J]. 计算机应用,2007,27(11):2623-2625.

[12] 吴　越,周学农. 智能协作技术在交通管理中的应用[J]. 系统工程,2001,19(1):52-55.

[13] 欧海涛,张卫东,张文渊,等. 基于多智能体技术的城市智能交通控制系统[J]. 电子学报,2000,28(12):52-55.

[14] 李振龙,赵晓华. 基于 Agent 的区域交通信号协调控制[J]. 武汉理工大学学报:交通科学与工程版,2008,32(1):130-133.

(上接第 24 页)

果表明,气象私有云平台已建存储架构设计合理,不同类型存储经过优化配置后优势互补,能够满足各类气象业务科研系统的存储需求,实际运行监控指标良好。未来还将跟踪技术发展趋势,结合业务系统发展需要,在多元化存储和统一存储管理平台等方面做进一步研究,并应用到实际工作中。

参考文献:

[1] Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-degree compared[C]//Grid computing environments workshop. [s. l.]: IEEE, 2008: 1-10.

[2] 陈　康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报,2009,20(5):1337-1348.

[3] 刘正伟,文中领,张海涛. 云计算和云数据管理技术[J]. 计算机研究与发展,2012,49(S1):26-31.

[4] 陈　全,邓倩妮. 云计算及其关键技术[J]. 计算机应用,2009,29(9):2562-2567.

[5] 王意洁,孙伟东,周　松,等. 云计算环境下的分布存储关键技术[J]. 软件学报,2012,23(4):962-986.

[6] 沈文海. 气象业务信息系统未来基础架构探讨-"云计算"和"大数据"在气象信息化中的作用[J]. 气象科技进展,2015(2):64-66.

[7] 沈文海. 从云计算看气象部门未来的信息化趋势[J]. 气象科技进展,2012(2):49-56.

[8] "气象私有云":我们身边的云计算[EB/OL]. 2014-08-13. http://www.cma.gov.cn/kppd/kppdkjzg/201408/t20140813_257125.html.

[9] Wikibon Survey[EB/OL]. 2012-08-23. http://wikibon.org/wiki/v/VMware_vSphere_5_Users_Move_Beyond_the_Storage_Protocol_Debate.

[10] Getting the hang of IOPS v1.3[EB/OL]. 2012-01-28. http://www.symantec.com/connect/articles/getting-hang-iops-v13.

[11] VMwarev Sphere Storage APIs-Array Integration (VAAI)[EB/OL]. 2013-03-17. http://www.vmware.com/resources/techresources/10337.

[12] What'san acceptable I/O latency? [EB/OL]. 2010-09-19. http://kaminario.com/company/blog/whats-an-acceptable-io-latency/.

[13] 李月安,曹　莉,高　嵩,等. MICAPS 预报业务平台现状与发展[J]. 气象,2010,36(7):50-55.

[14] 吴焕萍,张永强,孙家民,等. 气候信息交互显示与分析平台(CIPAS)设计与实现[J]. 应用气象学报,2013,24(5):631-640.

[15] 王　彬,周　斌,魏　敏. 气象计算网格模式预报系统的建立与优化[J]. 计算机应用研究,2010,27(11):4182-4184.