一种基于多智能体的二层路径规划模型研究

熊慕舟 黎 勇
(中国地质大学(武汉)计算机学院 武汉 430074)

摘 要 随着人群运动仿真技术的日趋成熟,其应用也得到了很好的推广,人群运动特征开始成为研究热点。路径规划系统作为人群仿真中的重要组成部分,为行人决定自身在环境中的行走路线提供了决策依据。为了仿真行人路径决策的过程,提出了一种二层路径规划模型,其中第一层模型产生一条粗略的路径,第二层模型根据第一层的粗略路径做精细的导航。实验结果表明,所提出的二层路径规划模型能够综合考虑环境中的静态和动态因素,为仿真模型提供了较好的路径规划,并且具有较高的仿真执行效率。

关键词 人群仿真,路径规划,全局规划,局部规划,基于智能体的仿真
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.012

Research on Two-layered Path Planning System Based on Multi-agent Simulation

XIONG Mu-zhou LI Yong
(School of Computer Science, China University of Geosciences, Wuhan 430074, China)

Abstract With the development of modeling and simulation in crowd movement, it has been widely applied to various applications for crowd movement estimation and safety evaluation. Recently, Crowd simulation has been an efficient tool for research on crowd movement feature and pattern. As one of the most important compositions of crowd simulation model, path planning system illustrates the process of pedestrian's decision for her/his route in environments. In order to simulate the process of pedestrian's route decision, this paper proposed a two-layered path planning system. The including a high-lever model produces a rough path, and the low-level model provides precise navigation according the route previously computed. The related experiment results indicate that the proposed model is able to consider both static and dynamic environment issues, which leads to a good path planning for simulation model. In addition, the proposed model is also effective in simulation execution.

Keywords Crowd simulation, Path planning, Global planning, Local planning, Agent-based simulation

1 引言

近年来,一些重大灾害事故(例如大型火灾、地震、泥石流等)以及公共场合的人群安全问题(例如人群踩踏事件、矿洞坍塌事件、商场火灾疏散)屡次出现。如何对现实世界中人群行为特征进行预测和分析,已引起了越来越多学者的关注。人群仿真作为一种近年来的新兴计算机技术,已成为研究人群疏散的主要手段之一。目前已有的仿真模型很多,例如,Eric Bouvier提出的粒子模型[1],其优点在于建模简单实用,但是尚不能仿真复杂的行人行走行为;Dirk Helbing等提出的恐慌状态下人群撤离的仿真模型[2],是经典的社会力模型,能够准确地描述恐慌状态下人群的行为,但是计算所需的时间开销大;元胞自动机模型[3-5],适合用于大规模仿真,但是其基于离散空间的特性制约了对人群复杂行为的描述能力;Yohei Murakami等提出的基于场景情节设置和规则描述的人群运动仿真模型[6],对于场景、规则和人群的限制不再那么严格,是一种比较通用的仿真模型,其缺陷在于人群规则的建立以及相互之间的学习能力较差;Hughes提出的流体力学模型[7],用于仿真高密度人群流动现象,但其中的力学方程计算量大且不能产生细粒度的仿真结果。基于智能体(Agent)的模型[8-11]是一种微观仿真模型,能够为仿真中的每个行人在每个仿真步长中产生一个细粒度的仿真结果。每一个智能体代表现实场景中的一个行人,它能够感知环境信息,例如障碍物和人群分布,并分析出候选路径节点的位置等,这样它就能建立环境信息的知识库。

在上述模型中,路径规划的工作分为两种:静态规划与动态规划。静态规划是指在智能体进入仿真环境时便已规划好行走路径;而动态规划则是根据仿真环境动态地对智能体的路径进行规划。在静态规划方面,仿真中的行人为了到达其行走目标,在每个仿真步长中需要选择一个速度向目标靠近并且避免碰撞,这就需要路径规划来实现。现有的路径规划方法大多采用所谓的"最省力"路径即全局最短路径作为行走路径。大多数系统都是使用A*算法原理产生当前位置到目标的最短路径,但为了避免碰撞等问题发生通常需要引入一些优化函数。这些方法的共性是很适用于静态环境,而在动态环境中就会有多局限性。而动态规划虽能作出动态响应,但是其缺点在于仿真成本较高,每当需要对环境作出动态响应时便需要执行模型的路径规划部分。

到稿日期:2015-07-08 返修日期:2015-08-22 本文受国家自然科学基金:基于大规模人群仿真的紧急事件预警与应对策略研究(61103145)资助。

熊慕舟(1980—),男,博士,副教授,主要研究方向为复杂系统建模与仿真、移动计算,E-mail:mzxiong@gmail.com;黎 勇(1989—),男,硕士生,主要研究方向为人群建模与仿真,E-mail:353933336@qq.com。

本文基于智能体仿真模型，为了更好地模拟人群路径规划的过程，提出了一种二层路径规划系统。上层路径规划模型产生粗略的全局路径，下层路径规划模型基于上层计算的全局路径进一步做精细的局部规划。此系统的思想反映了人类思考问题的基本过程，即一个人在行走过程中一般会先选择到目的地的子目标节点，然后在子目标节点之间根据动态环境与人群信息具体地规划运动方式。实际上，第一层的规划是一种静态规划，而第二层则为动态规划。该模型既可以利用静态规划的高效性，同时在动态规划方面也不需要大范围地进行重新调整，在一定程度上能够减少仿真开销，提高仿真结果的实时性。

2 二层路径规划模型设计

路径规划是人群仿真过程中智能体自主导航的关键技术之一，人群仿真中的路径规划是指寻找一条从智能体起始位置到其目标位置的适当路径，使得该智能体能够在移动过程中绕过所有的障碍物以及其他智能体。根据对环境信息的掌握程度，路径规划分为两种：环境信息完全知道的全局路径规划和环境信息完全未知或局部未知的局部路径规划。一般的路径规划分为 3 个步骤：环境建模、路径搜索、路径平滑。本文的人群路径规划分为基于宏观环境信息的全局路径规划来解决路径搜索和路径平滑的问题，以及基于智能体周围邻居信息的局部路径规划，在考虑动态环境信息的前提下，解决智能体的行走的决策问题。

2.1 二层路径规划系统的总体设计

本文所提出的路径规划系统分为两层。第一层执行全局路径规划，首先，基于静态的环境信息，产生一系列从智能体当前位置到目标的路径节点。与已有的全局路径规划方法不同，这一阶段产生的路径节点不会为智能体提供一条精确的路径轨迹，只是在不考虑动态环境信息的情况下，粗略地给出智能体必须经过的重要节点。其次，假设智能体熟悉环境，并且依据“路径最短，平滑度其次”的原则选择一个路径节点系列作为全局最优路径。

基于第一层所生成的全局路径，第二层的路径规划将提供精确的路径规划，也称为二次规划。在这一层，系统会考虑诸如人群密度和移动方向等环境动态信息。所谓的二次规划不是再次规划全局路径，而是规划智能体从当前位置到第一层规化结果中离当前位置最近点之间的路径。实际上，第一层的路径规划是保证所选取的全局最优路径节点之间没有障碍物并且节点间的距离不会太长。这意味着第二层的路径规划可以尽可能多地考虑动态的环境信息来规划从当前位置到下一个目标点的路径，也即局部路径，而不需要重复规划全局路径。本文二层路径规划系统框架如图 1 所示。

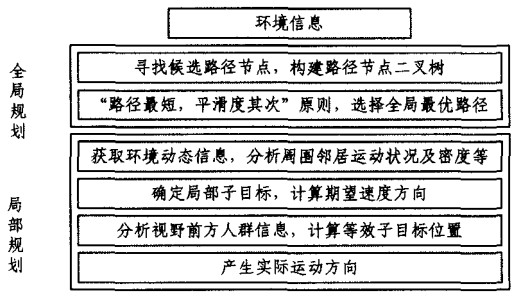

图 1 二层路径规划框架

2.2 全局路径规划

第一次路径规划即全局路径规划，其任务是根据环境信息寻找从起始点到目标点的、符合一定性能要求的可行路径，其具体的执行步骤描述如下。

首先根据宏观场景的环境信息计算出智能体当前位置与其目标点之间的障碍物，本文按如下的方式进行判断。一般来说，只有在如图 2 所示的矩形范围内的障碍物才有可能会挡在智能体移动到目标点的路上，我们认为矩形范围之外的障碍物不会阻挡智能体的移动，因此不将它算在智能体的有效障碍物范围内。

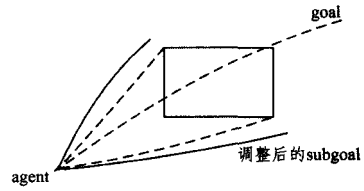

图 2 智能体绕行障碍物的节点

其次，建立一个包含所有可选路径的二叉树，二叉树从根节点到叶节点的一次深度遍历就是智能体初始位置到目标节点的一条路径。建立此二叉树的方法是：将当前智能体的有效障碍物按到初始位置的距离从小到大排列；取其中第一个结点开始遍历，计算距离最近的障碍物的边角与智能体之间的夹角，并求出边角中与智能体夹角最大和最小的角。我们认为这两个边角是智能体到目标点的过程中绕过该障碍物的最佳转折点。当然为了避免直接碰撞，将这两个节点的位置分别向障碍物外围做出一个调整，使得智能体在这些节点处绕行时不至于直接和障碍物碰撞，如图 3 所示。每个障碍物附近可以得到两个绕行节点，然后以这两个边角位置为智能体的新起点进行以上两步规划，这样，在遍历完所有有效障碍物后，得到的节点便可组成一个二叉树，当智能体绕过最远的障碍物后，便可直接看到目标点。因此这个由绕行节点组成的二叉树就是智能体到达目标点的所有理想路径的集合，每一个节点都是智能体到达目标节点目标点的候选子目标点。

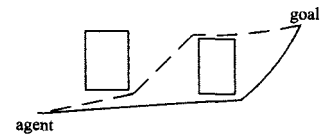

图 3 路径长度与转折点的对比图

全局路径规划上，首先对环境进行建模，使用网格划分；然后利用全局信息建立智能体当前位置到目标节点目标点的理想路径的二叉树；之后依照“长度最短优先，平滑度其次”的原则对理想路径二叉树进行路径评估，得到一条路径长度最短、平滑程度较好、能避免障碍物碰撞的全局最优路径。

宏观上选择目标和路径规划的方法采用基于子目标的最短路径规划算法。这个做法的基本假设是：当智能体与出口之间是直接可达(在可见区域内，没有被障碍物阻挡)时，智能体沿直线向出口移动；当智能体与出口之间不是直接可达时，智能体从子目标库中选择一个直接可达的最佳子目标，先向子目标移动，到达该子目标后，再继续向出口移动；当然，如果仍不直接可达，则再选择一个中间子目标，如此反复。选择最佳子目标的方法是：搜索所有的子目标，选出可见区域内即直接可达的子目标，从中挑选智能体到子目标的距离与子目标到出口的距离之和最小的。该过程的算法为 FINDPATH

(S,p,*detination*,*pathtree*),输出结果是路径节点二叉树,其算法描述如图 4 所示。

```
procedure FINDPATH(S, p, detination,pathtree)
  /*S是环境信息对象,包含障碍物、智能体信息等;p是位置信息对象;destination是目标的位置;pathtree是存储路径节点二叉树结构体对象*/
  obstacle *ob //障碍物类 class obstacle
  agpionts *awl, *aw2//位置信息类 class agpionts
  ob=Count_And_Sort_Ob(S,p,detination)
  GetEdgePiont_OnOb(ob,p,destination,awl,aw2)
  if pathtree.left!=NULL&&pathtree.right!=NULL
      pathbyob *f//二叉树节点结构体
      f=pathtree.search(pathtree,p)
      f.AddLeftNode(awl)
      f.AddRightNode(aw2)
  else
      pathtree.AddLeftNode(awl)
      pathtree.AddRightNode(aw2)
  end if
//输出从起始位置到目标位置的路径节点二叉树
end FINDPATH
```

图 4　可选路径二叉树生成算法

其中,函数 *Count_And_Sort_Ob*(*S*,*p*,*detination*)将点 *p* 和 *destination* 连线间的障碍物排序并返回距离 *p* 最近的障碍物。*GetEdgePiont_OnOb*(*ob*,*p*,*destination*,*awl*,*aw2*)函数求得障碍物 *ob* 上与 *p*-*destination* 连线夹角最大的两个边角并将其存储在 *awl* 和 *aw2* 中。*search*(*pathtree*,*p*)查找二叉树 *pathtree* 上节点值为 *p* 的节点并返回;*AddLeftNode*(*awl*)将 *awl* 作为左叶节点插入,同样,*AddRightNode*(*awl*)是插入右叶节点。

对所得到的二叉树进行深度遍历,并比较所有候选路径的长度,选取长度最短的路径作为第一层路径规化的节点序列。但是当人群具备一定规模、场景设置较为复杂时,一个智能体到目标节点的最短路径可能不止一条,单纯地以路径长短来区分优劣已不能排除多余的路径。路径的优劣还与路径的安全性和平滑程度等有关,因此除了路径长度以外,还可以用路径的平滑程度来区分路径的优劣。如图 3 所示,虚线路径 1 与实线路径 2 长度相同,但是路径 1 经过了 2 次转折,而路径 2 只有 1 次转折,本文利用"长度最短优先,转折点个数其次"的原则对全局路径进行选择。显然路径 2 的转折点较少,即平滑程度较好,因此应该选择路径 2 作为最优路径。其执行过程如图 5 所示。

```
procedure SELECTWAY(S,pathtree)
  vector(pathbyob*) fpoint//二叉树节点容器
  pahbyob *temp,*opt
  //搜索 pathtree,将叶节点存入 fpoint
  opt=fpoint[0]
  for i=0 to fpoint.size()
  temp=fpoint[i]
  opt=Judgeway(S,temp,opt)
  repeat
  //在 pathtree 中查找以 opt 为叶节点的路径节点系列集合并输出
end SELECTWAY
```

图 5　最佳全局路径选择算法

其中,容器 *fpoint* 存储 *pathtree* 中的叶节点,函数 *Judgeway*(*S*,*temp*,*opt*)根据"路径最短,平滑度其次"的原则判断以 *tenp* 为叶节点和以 *opt* 为叶节点的路径哪一条更优并返回。

2.3　局部路径规划

第一层全局路径规划后,智能体便可以依据全局最优路径向目标行进,但由于环境信息的更新和变化,智能体可能会偏离全局路径规划所得到的最优路径,其原因在于智能体可能会与周围的智能体邻居碰撞,或者智能体周围的人群密度会对其向最近的目标节点的移动造成影响。因此需要二次的路径规划来避免智能体之间的碰撞以及反映人群密度的改变对其移动的影响问题。

上一阶段从当前位置开始到目标位置规划路径,而在这一阶段,智能体根据人群密度和运动方向调节其运动行为,并且只规划当前位置到上一阶段产生的子目标位置的路径。由于当前位置与子目标之间没有障碍物,智能体可以考虑环境中所有的动态信息。这一阶段的结果是产生一个期望速度,包括速度大小和方向。

产生的速度大小假设只与智能体周围密度大小有关,式(1)所示的速度-密度关系式[12]用于计算智能体运动速度的大小。

$$f(\rho)=\begin{cases}A, & \rho\leqslant\rho_t\\ A\sqrt{\dfrac{\rho_t}{\rho}}, & \rho_t<\rho\leqslant\rho_c\\ A\sqrt{\dfrac{\rho_t\rho_c}{\rho_m-\rho_c}}\sqrt{\dfrac{\rho_m-\rho_c}{\rho}}, & \rho_c<\rho\leqslant\rho_m\end{cases}\tag{1}$$

其中,ρ 代表智能体周围人群密度;ρ_t,ρ_c,ρ_m 和 A 分别代表 0.8m^{-2},2.8m^{-2},5.0m^{-2} 和 1.4m/s。实际上,这里的变量 A 表示智能体期望速度的大小。

式(1)反映了智能体在自身周围密度不同的情况下运动速度的大小,不同密度下速度方向的选择过程如图 6 所示。

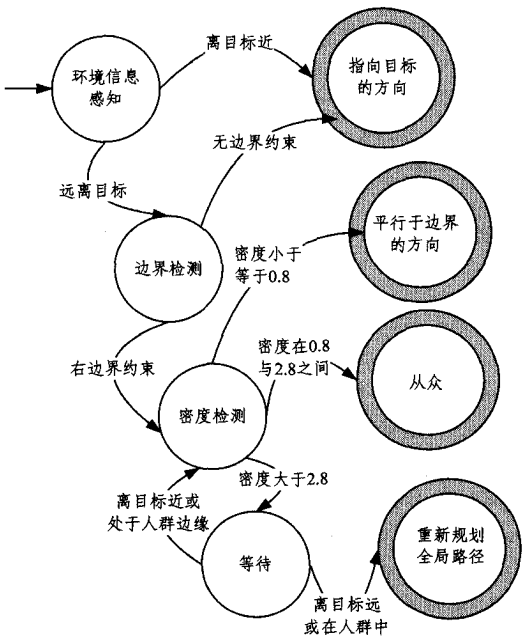

图 6　局部路径规划下速度方向的选择

其中 3 个密度值分别对应式(1)中的 ρ_t,ρ_c 和 ρ_m,不同密度下人群路径规划过程设计如下。

2.3.1　低密度场景局部路径规划

当周围人群密度小于 ρ_c 时,智能体能够相当从容地在环境中移动。然而,这不代表它将选择从当前位置指向子目标

的方向运动。以狭小的通道中行人运动特征为例，其中的行人总是趋向于平行通道墙壁的方向行走[13]。在智能体行走过程中，需要避免与周围其它邻居智能体以及障碍物的碰撞，那么一个平行于边界的速度将有助于智能体与周围邻居及障碍物保持安全距离且降低智能体调整速度方向的频率。考虑到以上运动特征，在局部路径规划中，智能体先检测周围的环境信息中是否存在边界约束其运动。当智能体向下一个子目标运动时，如果局部环境中有边界约束，它将选择一个平行于边界的速度向子目标靠近，速度方向的计算如式(2)所示。

$$\vec{d}=\begin{cases}\dfrac{\overrightarrow{P_iP_s}}{\|\overrightarrow{P_iP_s}\|}, & \text{没有约束条件}\\ \dfrac{\overrightarrow{P_bP_c}}{\|\overrightarrow{P_bP_c}\|}, & \text{有约束条件}\end{cases} \tag{2}$$

$$\overrightarrow{P_iP_s}\cdot\overrightarrow{P_bP_c}\geqslant 0 \tag{3}$$

其中，向量$\vec{d}$是产生的期望速度的方向，P_i 是智能体的当前位置，P_s 是第一阶段产生的子目标的位置，$\overrightarrow{P_iP_s}$向量是 P_i 指向 P_s 的向量。当智能体被边界约束时，P_b 是边界上离其最近的点，P_c 是边界切线上不等于 P_b 的任意点，即 $P_b\neq P_c$。这个配置可以保证当智能体以式(2)所产生的速度方向$\vec{d}$运动时，只要$\vec{d}$满足式(3)，那么智能体就不会偏离子目标太远。

低密度且有边界约束时，智能体实际上做了一次从当前位置到下一个子目标的局部的重新规划并输出一个期望的速度方向，整个过程的算法描述如图 7 所示。

```
procedure REFINDING(S,wpiont,p)
  /* wpiont 为第一次规划输出的全局路径节点
  p 为智能体当前位置 */
  agpionts subg=wpiont. front()//位置信息类对象
  agpionts p_sub=subg-p//向量减法
  bool b=IsConstraint(S,p,subg)
  if (b)
    agpionts d=MinDistToVertex(p)
    agpionts dir= GetConstraintVector(p,d)
    if p_sub. dotproduct(dir)>=0
      输出 dir
    end if
  end if
end REFINDING
```

图 7　低密度局部路径规划算法描述

其中，函数 *IsConstraint*(*S*, *p*, *subg*)判断位置 *p* 和 *subg* 之间有没有边界或者障碍物边界约束智能体，距离边界较近或者中间有障碍物则被约束。函数 *MinDistToVertex*(*p*)获取边界上距离 *p* 最近的位置信息并返回。函数 *GetConstraintVector*(*p*,*d*)则根据位置 *p* 和边界上的位置 *d* 计算并返回边界切线的向量。*p_sub. dotproduct*(*dir*)返回两个向量的点乘。

2.3.2　中密度场景局部路径规划

由式(1)可知，当智能体周围密度超过 ρ_c 时，其移动速度会减慢。密度增加的另一个影响因素是智能体会遇到更多潜在的碰撞，这将使得智能体不得不更频繁地调整自身运动，从而增加了行走时间。这种情况下，智能体通常不会重新规划全局路径，而是做一个长远的碰撞避免预测。通常，在人群密度较高时，行人会自主地跟随前方与自身同向的人行走[14]。这种现象可以解释为智能体跟随同向人群，从而避免前方潜在的碰撞。因此在密度较高时，通过加入同向人群和跟随人群，智能体可以避免前方潜在的碰撞。为了加入同向人群，智能体需要与视野前方的邻居智能体保持一个相对较大的碰撞时间。基于智能体当前位置和子目标的连线做一条垂线。在垂线上选取样本点，使用式(1)和式(2)分别求得每个样本点处所具有的速度大小和方向，即速度。假设样本点以此速度运动，计算它与视野前方所有智能体的碰撞时间。那么碰撞时间大于阈值 TH_{tc} 且距离智能体最近的碰撞点被选中。智能体将以被选中的点为子目标，通过式(1)、式(2)计算运动速度，以选取点为起点，跟随和加入同向人群。

为了跟随和加入同向人群，智能体的期望速度方向是其当前位置指向选取点。而行走过程中为了避免局部碰撞，其实际速度可能会与期望速度方向有所偏差。因此，每个仿真步长内智能体都将重新选取加入同向人群的起点，直到加入人群。在加入人群后智能体就跟随人群向子目标运动。整个过程描述如图 8 所示。其中 *GetSampledot*(*p*, *preferdir*)依据位置 *P* 做向量 *prefer* 的垂线，返回在垂线上间隔所取的点的集合。函数 *GetNeighborInSight*(*p*, *preferdir*, *rang*, *sight*)以位置 *P* 为基点(*preferdir* 为视野方向，*rang* 为视野范围，*sight* 为视野距离)，计算视野前方可见范围内的人群状况并返回。函数 *CaculateLanePiont*(*p*, *neightInsight*, *sample*)假设智能体从 *sample* 内的样本点出发与 *neightIns-ight* 内的人群碰撞，返回碰撞时间大于阈值 TH_{tc} 且距离智能体最近的碰撞点。

```
procedure GETLANEPOINT(S,p,subg,rang,sight)
    /* subg 为智能体的下一个子目标
    rang 为智能体视野距离
    sight 为智能体视野范围 */
    agpionts preferdir=subg-p //位置信息类对象
    vector<agpionts*> sample//位置信息容器
    vector<agent*>neighborInsight//智能体容器
    sample=GetSampledot(p,preferdir)
    neighborInsight=GetNeighborInSight(p,preferdir,rang,sight)
    lpiont=CaculateLanePiont(p,neightInsight,sample)
//输出 lpiont 为加入视野前方人群的切入点
end GETLANEPOINT
```

图 8　中密度情形局部路径规划算法描述

2.3.3　高密度场景局部路径规划

随着智能体周围人群密度的进一步增加，其运动速度明显下降，行走时间更长，甚至陷入拥塞。因此，行人一般会避免向这类区域行走，但是如果偶然陷入高密度区域，智能体可能会重新规划全局路径[15]。然而其他某些因素也许会改变智能体的策略，例如，视野前方的密度值、当前位置与目的地的距离等。一般来说，当智能体距离目的地较近时，虽然周围密度高，但仍会选择等待；距离目的地较远，但是当前位置处在高密度人群之外，同样会选择等待；反之，若距离目的地较远且周围密度较大，则需要重新规划路径。

3　实验设计及结果分析

3.1　实验设计

仿真平台根据文中提出的模型利用 Visual Studio2008 构建，配合使用 Matlab 2011b 将仿真数据导入并绘制实验效果图及性能分析图。本文模型构建的平台在仿真中可以观察到人群在路径选择上的基本思维方式，根据静态环境信息选择

全局路径节点系列，并根据行走过程中所感知到的环境变化信息来调整策略，显示出不同的行为特性。例如，靠近障碍物被边界约束时，沿着边界行走；前方行人较多时，显示出跟随行为；距离障碍物较远、不受其边界约束且离目标较近时，直接向目标行走。

实验的仿真场景及效果如图 9 所示，场景是长宽均为 400 的二维方图。在二维坐标系中，横纵坐标的范围分别为 $x\in(-200,200)$ 和 $y\in(-200,200)$。其中右上方的障碍物处在第一象限，其左下坐标是(10,10)，右上坐标是(50,50)，另外 3 个障碍物分别是它关于 Y 坐标轴对称、原点对称以及 X 坐标轴对称的。仿真的人群数目为 200，分别初始化在场景中 4 个角的其中 1 个区域内。4 个区域中，每个区域的人群的目的地都是对角区域的某一点，即所有人群在仿真中过程中都向对角区域移动。当仿真中的智能体到达目的地后，退出仿真，不再停留在场景中。

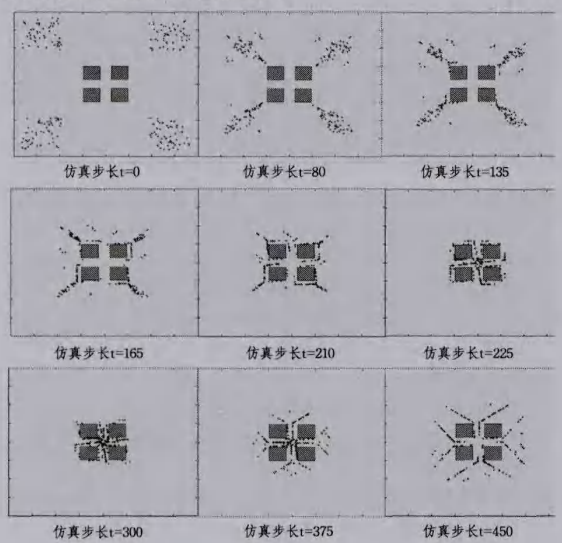

图 9　200 个 agent 在不同仿真时刻的人群分布图

3.2　实验结果及分析

仿真实验在 2.67GHz Intel i5 4 核处理器、4GB 内存的计算机上执行。仿真场景分别将智能体数目设置为 100，200 和 300，其中场景中的静态信息保持不变。以场景中智能体数目设置为 200 的仿真截图为例。

图 9 中，$t=0$ 时刻是仿真的初始时刻，人群载入场景准备按照第一次规划路径的节点开始行走；$t=80$ 时刻，人群向子目标靠近，感知到边界约束，于是趋于边界切线方向行走；$t=135$ 时刻，人群前面的行人靠近障碍物，开始沿着障碍物边界切线方向行走；$t=165$ 时刻，人群前方的一部分人都开始沿边界行走，后面的行人则开始表现出跟随行为；$t=210$ 时刻，后方人群跟随前方行人行走，并且都受到障碍物边界的约束，表现出沿边界行走的行为；$t=255$ 时刻，4 个区域的人群沿着边界向前行走，开始在障碍物之间的狭窄区域聚集；$t=300$ 时刻，由于行人不断在狭窄区域聚集，部分行人周围密度增大，为了避免不断改变行走方向，行人不在沿边界行走，而是紧紧跟随视野前方的人群一起行走；$t=375$ 时刻，行人在周围密度较大时跟随前方人群向下一个密度较小的子目标移动。由于密度减小，人群又开始表现出沿边界行走的行为；$t=450$ 时刻，人群走出了障碍物所在区域，前方行人不在受到边界约束，于是直接向着目标行走。

可以看出，构建的仿真平台很好地反映了人群在受到边界约束或周围密度变化时会表现出不同的行为。这些观察到的行为有效地验证了本文所提出的二层路径规划模型的有效性。

图 10 中纵坐标为实际仿真时间，横坐标为仿真步长，反映的是对应的一个仿真步长所消耗的实际仿真时间。可以看出，人群数量越多，每一个仿真步长所需的实际仿真时间增加更快，所需仿真步长也增加更快。仿真开始时由于人群数量是初始状态以及行人都开始规划到第一个子目标的局部路径，因此每个步长所需实际仿真时间较长；随后人群靠近障碍物，受到边界约束，开始沿边界行走，不再需要持续改变速度方向，因此随着越来越多行人沿边界行走，每个步长的实际仿真计算量大幅减小，仿真时间显著下降；而当行人都涌入狭窄的障碍物之间时，随着人群密度的不断增加，行人开始不再沿边界行走，而是寻找前方同向行人并跟随行走，因此实际仿真时间要开始增加；随后人群离开障碍物区间，人群密度减小，行人又开始沿边界行走，甚至不再受边界约束且距离目标较近时，直接向目标行走，因此仿真时间开始下降。

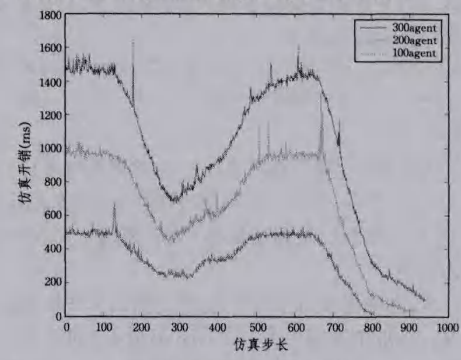

图 10　模型执行时间对比

结束语　本文提出一种二层路径规划方法来模仿人们的思维习惯，将行人的路径选择过程分为两部分，即先考虑全局路径，然后根据环境信息变化(主要是边界约束和密度变化)，做不同的择路决策。实验结果表明该模型能够准确地反映行人在不同周围环境下选择较好的行走路线以及表现出沿边界行走和跟随等行为特性。

虽然该模型在行人路径规划策略上的仿真已经有不错的结果，但是人群运动毕竟是一个复杂的过程，还有一些细节有待改善和提高，例如，密度较高时，行人加入前方人群的细节以及行人对周围信息的评估。因此未来的工作中还要对人群运动的具体特性进行深入学习，以期将模型进一步完善。

参 考 文 献

[1] Bouvier E, Cohen E, Najman L. From crowd simulation to airbag deployment: Particle systems, a new paradigm of simulation[J]. Journal of Electronic Imaging, 1997, 6(1): 94-107

[2] Helbing D, Farkas I, Vicsek T. Simulating dynamical features of escape panic[J]. NatureATURE, 2000, 407(6803): 487-490

[3] Beigy H, Meybodi M R. A mathematical framework for cellular learning automata[J]. Advances in Omplex Systems, 2004, 7(3/4): 295-319

[4] Liu Zhen-yu, Rri Xiao-ping, Dong Cheng-wei, et al. Simulation of urgent evacuation in subway station based on Cellular Automation[J]. Computer Engineering and Applications, 2009, 45(27): 203-205(in Chinese)
刘真余，芮小平，董承玮，等. 元胞自动机地铁人员疏散模型仿真

[J]. 计算机工程与应用,2009(27):203-205

[5] Tao Ping, Zhang Xiao-ying, Ma Heng-Liang. Simulation of personnel evacuation based on cellular automaton model[J]. Computer Simulation, 2009, 26(10): 319-322(in Chinese)
陶平,张小英,马恒亮. 基于元胞自动机模型的人员疏散仿真研究[J]. 计算机仿真,2009(10):319-322

[6] Murakami Y, Minami K, Kawasoe T, et al. Multi-agent simulation for crisis management[C]// Proceedings IEEE Workshop on Knowledge Media Networking. Kyoto, Japan: IEEE Computer Society, 2002: 135-139

[7] Hughes R L. The flow of human crowds[J]. Annual Review of Fluid Mechanics, 2003, 35: 169-182

[8] Kountouriotis V, Thomopoulos S C A, Papelis Y. An agent-based crowd behaviour model for real time crowd behaviour simulation[J]. Pattern Recognition Letters, 2014, 44(1): 30-38

[9] Xu Gao. Simulation model for crowd evacuation based on agent technology[J]. Journal of Southwest Jiaotong University, 2003, 38(3): 301-303(in Chinese)
徐高. 基于智能体技术的人员疏散仿真模型[J]. 西南交通大学学报,2003(3):301-303

[10] Cui Xi-hong, Li Qiang, Chen Jin, et al. Study on MA-based Model of Occupant Evacuation in Public Facility[J]. Journal of System Simulation, 2008, 20(4): 1006-1010(in Chinese)
崔喜红,李强,陈晋,等. 基于多智能体技术的公共场所人员疏散模型研究[J]. 系统仿真学报. 2008,20(4):1006-1010

[11] Huang Xi-fa, Wang Ke-jun, Guo Lian-ying, et al. Microscopic simulation model study on pedestrian evacuation based on agent technology[J]. Journal of System Simulation, 2009, 21(15): 4568-4571(in Chinese)
黄希发,王科俊,郭莲英,等. 基于 Agent 技术的人员疏散微观仿真模型研究[J]. 系统仿真学报,2009(15):4568-4571

[12] Hughes R L. A continuum theory for the flow of pedestrians [J]. Transportation Research Part B-Methodological, 2002, 36 (6): 507-535

[13] Santra S B, Schwarzer S, Herrmann H. Fluid-induced particle-size segregation in sheared granular assemblies[J]. Physical Review E, 1996, 54(5): 5066-5072

[14] Makse H A, Havlin S, King P R, et al. Spontaneous stratification in granular mixtures[J]. Nature, 1997, 386(6623): 379-382

[15] Santra S B, Schwarzer S, Herrmann H. Fluid-induced particle-size segregation in sheared granular assemblies[J]. Physical Review E, 1996, 54(5): 5066-5072

(上接第 38 页)

1000～2000 之间时,算法用时随着 n_r 的增大而增多;当 n_r 的取值在 2000～4000 之间时,算法用时随着 n_r 的增大而减少;当 n_r=4000 时算法用时到达最小,此后算法用时又随着 n_r 的增大而增多。

进一步,在确保 n_r 值固定的情形下测试算法性能随 i 变化的情况,i 的取值范围为 1～5。图 8 显示了 i 对算法性能的影响,即随着 i 的增加,算法用时相应减少,这是因为随着过滤点个数的增加,能够预先过滤的数据对象也相应增加。

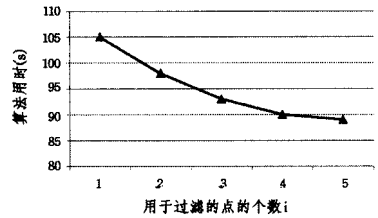

图 8 i 对过滤效果的影响

结束语 Skyline 查询在多目标优化、数据挖掘等领域有着重要的应用,传统的基于单机的串行 Skyline 查询算法在大数据量的情形下很难满足用户的需求。本文基于流行的分布式并行编程框架 MapReduce,提出了适用于在大数据集上计算 Skyline 查询的并行算法。该并行算法对现有的基于角度的划分策略进行了改进,提出了 Balanced Angular 划分策略和 Map 端过滤数据的策略。实验结果显示,本文提出的并行 Skyline 查询算法能显著提升系统性能。

参 考 文 献

[1] Balke W T, Güntzer U. Multi-objective query processing for database systems[C]// Int'l Conference on Very Large Data Bases (VLDB). 2004: 936-947

[2] Wei Xiao-juan, Yang Jing, Li Cui-ping, et al. Skyline query[J]. Journal of Software, 2008, 19(6): 1386-1400(in Chinese)
魏小娟,杨婧,李翠平,等. Skyline 查询处理[J]. 软件学报,2008,19(6):1386-1400

[3] Kung H T, Luccio F, Preparata F P. On finding the maxima of a set of vectors[J]. Journal of ACM, 1975, 22(4): 469-476

[4] Preparata F P, Shamos M I. Computational Geometry: An introduction[M]. New York: Springer-Verlag, 1985

[5] Börzsönyi S, Kossmann D, Stocker K. The skyline operator[C]// IEEE International Conference on Data Engineering (ICDE). 2001: 421-430

[6] Chomicki J, Godfrey P, Gryz J, et al. Skyline with presorting[C]// IEEE International Conference on Data Engineering (ICDE). 2003: 717-816

[7] Godfrey P, Shipley R, Gryz J, et al. Maximal vector computation in large data sets[C]// Int'l Conference on Very Large Data Bases (VLDB). 2005: 229-240

[8] Bartolini I, Ciaccia P, Patella M, et al. Efficient sort-based skyline evaluation [J]. ACM Transactions on Database Systems (TODS), 2008, 33(4): 1-45

[9] Tan K L, Eng P K. Efficient progressive skyline computation[C]// IEEE International Conference on Data Engineering (ICDE). 2001: 301-310

[10] Papadias D, Tao Y. An optimal and progressive algorithm for skyline queries[C]// ACM SIGMOD Int'l. Conference on Management of Data (SIGMOD). 2003: 467-478

[11] Papadias D, Tao Y. Progressive skyline computation in database systems[J]. ACM Transactions on Database Systems (TODS), 2005, 30(1): 41-82

[12] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[13] Zhang Bo-liang, Zhou Shui-geng, Guan Ji-hong. Skyline under MapReduce framework [J]. Computer Science and Exploration, 2011, 5(5): 385-397(in Chinese)
张波良,周水庚,关佶红. MapReduce 框架下的 Skyline 计算[J]. 计算机科学与探索,2011,5(5):385-397

[14] Chen L, Hwang K, Wu J. MapReduce Skyline Query Processing with A New Angular Partitioning Approach[C]// Proc. of International Parallel and Distributed Processing Symposium. 2012: 2262-2270