元胞自动机简介及其应用

周吕文

2017 年 5 月

引言

元胞自动机 (Cellular Automata, CA)[1], 又称细胞自动机, 最早由 John von Neumann 在 1950 年代为模拟生物细胞的自我复制而提出的. 当时 John von Neumann 假想有一台机器漂浮在池塘上面, 这个池塘里有很多机器的零部件, 这个机器是一台通用的建造器: 只要给出它自身的描述, 这台机器就能在池塘里寻找合适的零件再创造出自己. 但这个模型仍然是以具体物质原材料的吸收为前提, 后来在 Stanislaw M.Ulam 的建议下, 他从细胞的视角思考这个问题, 于是元胞自动机就延生了. 作为描述自然界复杂现象的简化数学模型, 元胞自动机可以表现出极其复杂的形态, 因此常用于复杂系统的建模. 上个世纪五十年代, John von Neumann 和 Stanislaw M.Ulam 提出元胞自动机后, 并未受到学术界重视. 直到 1970 年, 剑桥大学的 John Horton Conway 设计了一个电脑游戏 “生命游戏 (Game of Life)”[2] 后才吸引了科学家们的注意. 此后，Stephen Wolfram 对初等元胞机 256 种规则 [3] 所产生的模型进行了深入研究，并用熵来描述其演化行为, 将细胞自动机分为平稳型, 周期型, 混沌型和复杂型 [1, 4].

元胞自动机是复杂系统研究的一个典型方法, 特别适合用于空间复杂系统的时空动态模拟研究. 不同于一般的动力学模型, 元胞自动机不是由严格定义的物理方程或函数确定, 而是用一系列模型构造的规则构成. 凡是满足这些规则的模型都可以算作是元胞自动机模型. 因此, 元胞自动机是一类模型的总称, 或者说是一个方法框架. Stephen Wolfram 给出了元胞自动机需要满足的五个最基本要素:

- 包含元胞 (Cell) 的规则的网格 (Lattice Grid). 空间被离散成网格, 元胞分布在网格中的各个格子中.
- 离散的时间步长. 元胞状态的改变是定义在两个时间步之间的, 同一个时间步内, 元胞的状态是固定的.
- 每个元胞的状态数是有限的. 也就是说任何元胞只能在固定的几种状态间切换.
- 所有元胞状态的改变更新都遵循同样的规则. 也就是说系统中的每一个元胞都是一样的, 没有享受特殊待遇的元胞.
- 元胞自动机的规则定义在局部. 也就是说元胞的状态改变只取决于它周围的元胞的状态.

在元胞自动机模型中, 散布在规则格网中的每一元胞取有限的离散状态, 遵循同样的作用规则, 依据确定的局部规则作同步更新. 大量元胞通过简单的相互作用而构成动态系统的演化. 其特点是时间, 空间, 状态都离散, 每个变量只取有限多个状态, 且其状态改变的规则在时间和空间上都是局部的.

随着计算机的迅速发展, 元胞自动机这种非常适合在计算机上实现的模型越来越被受到重视. 目前, 元胞自动机已经被广泛应用于社会学, 经济学, 生态学, 图形学, 数学, 物理学, 化学, 地理, 环境和军事学等各个领域. 这里仅列举出元胞自动机在以下三个领域的应用:

- **社会学**: 无胞自动机经常用于研究个人行为的社会性, 流行现象. 例如人口的迁移, 公共场所内人员的疏散, 流行病的传播.

- **图形学**: 元胞自动机以其特有的结构的简单性, 内在的并行性以及复杂计算的能力成为密码学中研究的热点方向之一. 例如被用于图像的加密.
- **物理学**: 在物理学中, 元胞自动机已成功的应用于流体, 磁场, 电场, 热传导等的模拟. 例如格子气自动机.
- **生物学**: 元胞自动机的设计思想本身就来源于生物学自繁殖的思想，因而它在生物学上的应用更为自然而广泛。肿瘤细胞的增长机理和过程模拟, 人类大脑的机理探索, 艾滋病病毒 HIV 的感染过程, 自组织, 自繁殖等生命现象的研究.

方法介绍

在本章的第1节中已经给出了元胞自动机所需要满足的要素. 应用元胞自动机模型来模拟动力学系统, 首先需要给出这些要素的定义. 在抽像出元胞自动机的各要素前, 本节先介绍两个经典的元胞自动机, 以使读者对元胞自动机先有个感性认识.

第一个要介绍的是在第1节中提到的 “生命游戏”[2]. “生命游戏” 是一个典型的**二维元胞自动机**, 它是 Conway 在 2 世纪 6 年代末设计的, 它包括一个二维正方型网格世界, 这个世界中的每个方格居分布着一个活着的或死了的细胞. 一个细胞在下一个时刻生死取决于相邻八个方格中活着的或死了的细胞的数量. 如果相邻方格活着的细胞数量过多, 这个细胞会因为资源匮乏而在下一个时刻死去; 相反, 如果周围活细胞过少, 这个细胞会因太孤单而死去. 生命游戏中元胞只具有 “生 (☺)” 和 “死 (☠)” 两种状态 (在计算机中一般用 1 和 0 来分别表示生和死两种状态); 所有元胞在离散的时间步长上更新状态. 一个元胞的生死由该时刻本身的状态和周围八个邻居的状态决定, 具体规则如下:

- 当前元胞状态为 “死” 时, 当周围有 3 个状态为 “生” 的元胞时, 该元胞状态变成 “活”. 这一规则用来模拟生命的繁殖.
- 当前元胞状态为 “活” 时，当周围低于 2 个 (不包含 2 个) 状态为 “活” 的元胞时, 该元胞状态变成 “死”. 这一规则用来模拟生命数量稀少时, 个体也难以存活.
- 当前元胞状态为 “活” 时, 当周围有 2 个或 3 个状态为 “生” 的元胞时, 该元胞状态继续保持为 “生”.
- 当前元胞状态为 “活” 时，当周围有 3 个 (不包含 3 个) 以上的状态为 “活” 的元胞, 该元胞状态变成 “死”. 这一规则用来模拟生命数量过多, 竞争导致个体消亡.

以上四条规则可以简化为以下两条规则

- **死亡规则**: 当前时刻, 如果某元胞状态为 “生”, 且八个邻居元胞中有两个或三个状态为 “生”, 则下一时刻该元胞继续保持为 “生”, 否则 “死”, 图1是这一规则的示意图;

图 1: 死亡规则

- **磐涅规则**: 当前时刻, 如果某元胞状态为 “死”, 且八个邻居元胞中正好有三个状态为 “生”, 则下一时刻该元胞复活为 “生”, 否则保持为 “死”, 图2是这一规则的示意图.

图 2: 磐涅规则

实际中, 玩家可以设定周围活细胞的数目怎样时才适宜该细胞的生存. 如果这个数目设定过低, 世界中的大部分细胞会因为找不到太多的活的邻居而死去, 直到整个世界都没有生命; 如果这个数目设定过高，世界中又会被生命充满而没有什么变化. 通常这个数目一般选取 2 或者 3, 这样整个生命世界才不至于太过荒凉或拥挤，而是一种动态的平衡. 由于设置的初始状态和迭代次数不同，游戏过程中会得到各种令人叹服的优美图案: 杂乱无序的细胞会逐渐演化出各种精致, 有形的结构; 这些结构往往有很好的对称性, 而且每一代都在变化形状; 一些形状已经锁定, 不会逐代变化; 有时, 一些已经成形的结构会因为一些无序细胞的 “入侵” 而被破坏. 但是形状和秩序经常能从杂乱中产生出来. 图3为游戏过程中得到的几种模式.

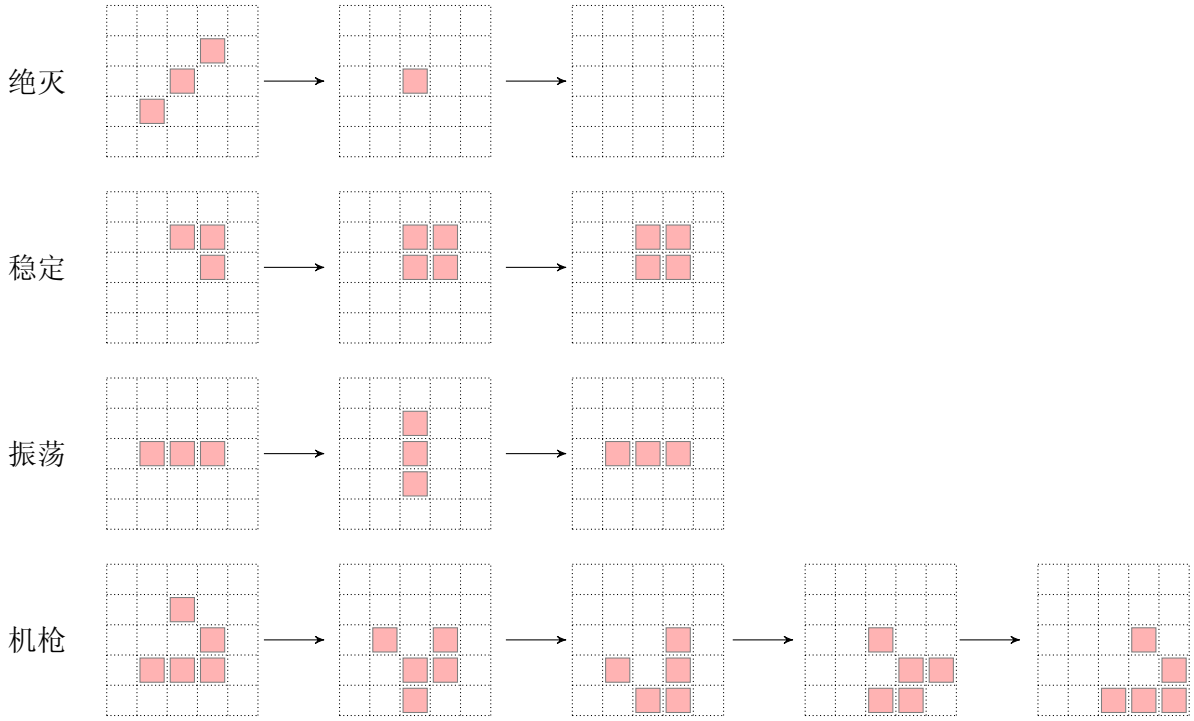

图 3: 生命游戏中的几种模式

Wolfram 认为生命游戏的缺点是只研究了一种规则, 他要尽可能系统地研究各种规则元胞自动机的行为. 先从一维状态链做起, 例如演化规则的半径为 $r = 1$ 的情况, 包括自身共 3 个邻居. 若每个位置可有 2 个状态, 则这 3 个邻居可以取 $2^3 = 8$ 种排列, 对每一种排列, 下一个时刻可以取 2 个不同的值, 因此一共有 $2^{2^3} = 256$ 个不同的元胞自动机. 本节要介绍的第二个例子就是这 256 种**一维元胞自动机**中的第 184 种 (通常称为第 184 号规则). 184 号元胞自动机中也是最基本的交通流元胞自动机模型. 如图4所示, 184 规则给出了相邻三个元胞的所有 8 种可能状态在下一时刻中间位置元胞的状态. 在 184 号规则中, 道路被划分为等间距的格子, 每一个格子表

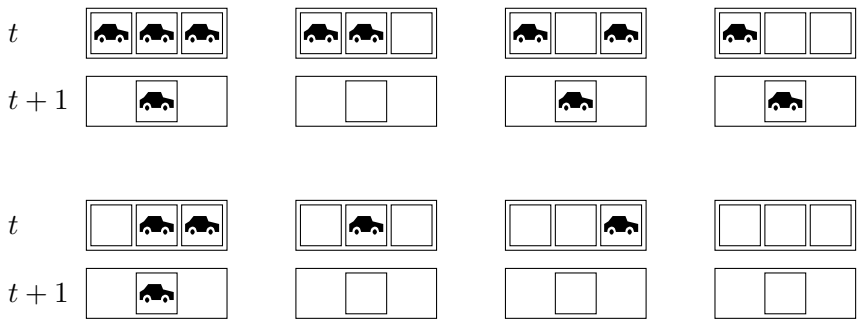

图 4: 第 184 号规则

示一个元胞. 任一元胞只有两种状态, 或者被一辆占据, 或者为空. 所有车辆都向同一方向行进 (向右), 在某一时间步内, 如果第 n 辆车前方元胞为空, 则该车辆向前行驶一格, 否则保持原地不动. 在 184 号规则下, 只要给定道路初始时刻的状态, 可以得到之后任何不同时刻道路的状态. 如图5所示是一条长度为 20 的道路, 道路中有 5 辆车, 根据 184 号规则得到的前 10 个时间步的元胞自动机状态. 每一行代表一维元胞自动机在某时刻的状态, 从上到下时间逐行增加一步. 需要注意的是, 由于第一个元胞和最后一个元胞分别缺少左邻居和右邻居, 一种简单的做法是以最后一个元胞作为第一个元胞的左邻居, 同时也以第一个元胞作为最后一个元胞的右邻居. 这样操作后, 相当于这一条路段是首尾相接的圆环, 从最右端驶出路段的车辆又从最左端驶入. 这种边界的处理方式称为**周期性边界条件**. 类似于图5的图, 一般称为**时空图** (横向表示空间, 纵向表示时间).

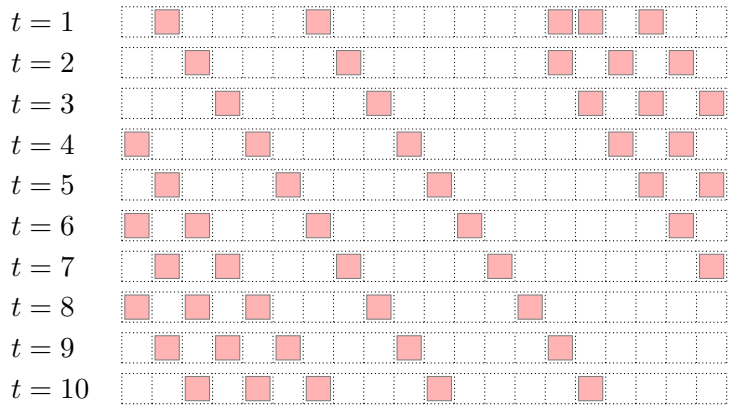

图 5: 第 184 号规则得到的时空图

通过以上生命游戏元胞自动机和 184 号规则交通元胞自动机的介绍, 相信读者对元胞自动机和元胞自动机的构成已经有了一定的感兴认识. 一个标准的元胞自动机 (A) 由元胞, 元胞状态, 邻域和状态更新规则构成, 用数学表示为

$$A = (L, d, S, N, f) \tag{1}$$

其中 L 为元胞空间; d 为元胞自动机内元胞空间的维数; S 是元胞有限的, 离散的状态集合; N 为某个邻域内所有元胞的集合; f 为局部映射或局部规则. 一个元胞通常在一个时刻只有取自一

个有限集合的一种状态, 例如 $\{0,1\}$. 元胞状态可以代表个体的性质, 特征, 行为等. 在空间上与元胞相邻的细胞称为邻居，所有邻居组成邻域. 在计算机中, 元胞所在的网格一般具有边界, 处于边界上的元胞, 它周围的邻居元胞数量小于普通的元胞. 因此对于边界上的元胞需要给出边界条件, 使所有元胞都具有相同数量的邻居. 常见的边界条件有周期型, 反射型和定值型. 接下来本节将具体介绍元胞自动机的各个构成要素:

- **元胞**: 元胞是元胞自动机模型中最基本的单元, 就如同每一个细胞是生命机体组织中的一个最基本的单元. 因此元胞自动机也叫细胞自动机. 作为规则网格中的一个格子, 元胞可以有很多形状. 元胞根据它周围的邻居元胞的状态, 按照规则来改变状态. 状态可以是 $\{0,1\}$ 的二进制形式. 或是 $\{s_0, s_1, \cdots, s_n\}$ 整数形式的离散集, 严格意义上, 元胞自动机的元胞只能有一个状态变量. 但在实际应用中, 往往将其进行了扩展. 例如每个元胞可以拥有多个状态变量.

- **网格**: 所有元胞都是放在网格中的, 就像细胞是放在生物体上一样. 从维度上分, 网格主要有一维, 二维, 三维及更高维. 最简单的情况就是一维网格, 一维网格就是一系列的元胞排成一条线. 在模拟单条车道的交通流时就是用的一维网格. 最常用的是二维网格, 所有元胞排成在一个面内. 图6, 7, 8是三种常见的二维网格. 这三种网格各有优缺点: 三角网格的

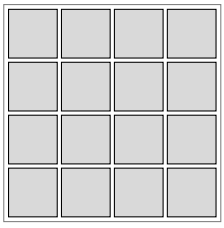

图 6: 正方形网格

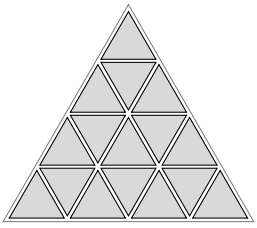

图 7: 三角形网格

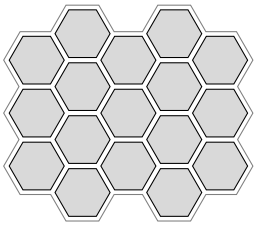

图 8: 六边形网格

  优点是拥有相对较少的邻居数目, 这在某些时候很有用; 其缺点是在计算机的表达与显示不方便, 需要转换为正方形网格. 正方形网格的优点是直观而简单, 而且特别适合于在现有计算机环境下进行表达显示; 其缺点是不能较好地模拟各向同性的现象, 例如格子气模型中的 HPP 模型. 六边形网格的优点是能较好地模拟各向同性的现象, 因此, 模型能更加自然而真实, 如格气模型中的 FHP 模型, 其缺点同三角网格一样, 在表达显示上较为困难和复杂. 最常用的二维网格是正方形网格, 一是正方形网格本身简单, 并且一般情况的二维问题都适用, 二是正方形网格可以用矩阵或数组直接表示, 方便程序的实现. 比如模拟多条车道是, 车可能会换道, 这时一维网格不能满足要求, 需要用二维正方形网格. 三维及三维以上的网格较复杂.

- **邻居**: 以上的元胞及元胞的网格空间只表示了系统的静态成分, 为将 “动态” 引入系统, 必须加入演化规则. 在元胞自动机中，这些规则是定义在空间局部范围内的, 即一个元胞下一时刻的状态决定于本身状态和它的邻居元胞的状态. 因而, 在指定规则之前, 必须定义一定的邻居规则，明确哪些元胞属于该元胞的邻居. 在一维元胞自动机中, 通常以半径, 来确定邻居, 距离一个半径内的所有元胞均被认为是该元胞的邻居. 二维元胞自动机的邻居定义较为复杂, 在二维的方型网格中, 常用的邻居三种, 如图9, 10, 11所示, 分别是以周围四个, 八个和二十四个元胞为邻居的类型, 当然你可以根据你的需要自行设计邻居的形式.

- **边界条件**: 在理论上, 元胞空间通常是在各维向上是无限延展的, 这有利于在理论上的推理和研究. 但是在实际应用过程中，我们无法在计算机上实现这一理想条件, 因此，实际模拟中就存在边界. 在定义元胞的邻居时, 会出现一个问题: 边界上元胞的邻居不足. 元胞自动机对每个元胞施加同样的规则, 而边界上的元胞与非边界上的元胞具有不同的邻居数. 因此需要设定不同的边界条件, 使边界上的元胞也与正常的元胞具有相同的邻居数量. 边界条件主要有四种类型: 定值型, 周期型, 反射型和吸收型, 如图12所示. 其中较为常用的是周期型边界, 周期型是指相对边界连接起来的元胞空间. 对于一维空间，元胞空间表现

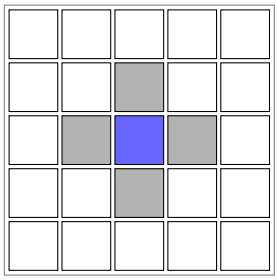

图 9: vonNeumann 邻居

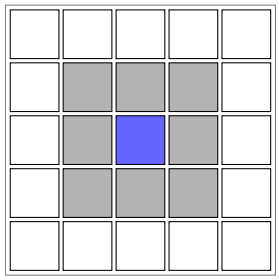

图 10: Moore 邻居

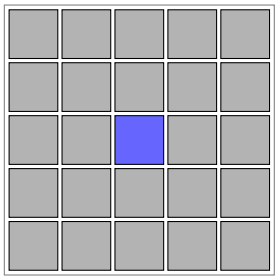

图 11: 扩展 Moore 邻居

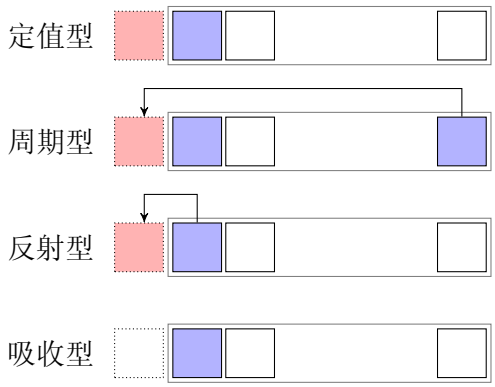

图 12: 边界条件

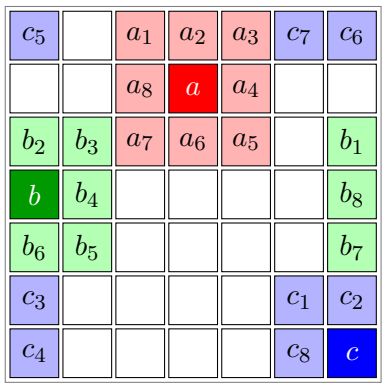

图 13: 二维方网格的周期边界条件

为一个首尾相接的 “圈”. 对于二维空间, 上下相接, 左右相接. 而形成一个拓扑圆环面, 形似车胎或甜点圈. 周期型空间与无限空间最为接近, 因而在理论探讨时, 常以此类空间型作为试验. 图13所示为二维方型网格周期边界条件时, 三种元胞 (a, b 和 c) 的 Moore 邻居. 元胞 c 为非边界元胞, 它有八个正常的邻居, 而 b 和 c 是边界元胞, 它们没有足够的邻居, 通过周期性边界条件, 可以规定图13所示的邻居.

- **规则**: 根据元胞当前状态及其邻居状况确定下一时刻该元胞状态的动力学函数, 简单讲, 就是一个状态转移函数. 我们称该函数为元胞自动机的局部映射或局部规则. 规则是支配整个元胞自动机的动力学行为的. 一般规则都是定义在局部. 通过局部间元胞的相互作用而引起全局变化. 比如说流行病的传播, 只有相互靠的近的人才能进行直接的传播, 这就是局部作用, 但当流行病传播一定时间时就会引起大范围的变化, 这就是全局变化. 而这个全局变化是由一个个局部作用引起的. 元胞自动机的规则一般为总和型. 就是说某个元胞下时刻的状态只决定于它所有邻居当前状态及自身的当前状态. 一个简单的例子是足球场观众台上的人浪, 当你最靠近的人即邻居站起来时, 你才能站, 当他们坐下后你才能坐下, 这就是规则.

应用举例

本节将主要介绍两个元胞自动机模型及其 MatLab 程序实现, 使读者近一步深入了解元胞自动机模型, 并掌握简单的元胞自动机模型编程的实现方法.

森林火灾元胞自动机规则介绍

根据模拟的需要, 元胞自动机规则中还可以引入概率, 一般把规则中含有概率的元胞自动机称为概率元胞自动机 (概率机). 一个最简单的二维概率元胞自动机是森林火灾模型, 该模型的规则是由 Drossel 和 Schwabl[5] 在 1992 年给出的, 因此该模型又称 Drossel-Schwabl 森林火灾模型. 森林火灾元胞自动机模型定义在正方网格上, 元胞有三种状态: 树 (未燃烧的树), 火 (正

在燃烧的树) 和空状态. 某元胞下时刻状态由该时刻本身的状态和周围四个邻居 (VonNeumann 邻居) 的状态决定. 具体规则

- **着火**: 若某元胞的状态为 “树”, 且其最邻近的 4 个邻居元胞中有状态为 “火” 的元胞, 则该元胞下时刻的状态由 “树” 变为 “火”, 用以模拟火势的漫延. 对于邻居中没有 “火” 的情况, 状态为 “树” 的元胞也会以一个低概率 f 变为 “火”, 用以模拟闪电引起的火灾.

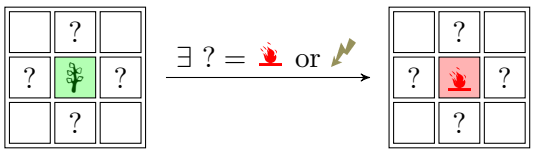

图 14: 着火规则

- **烧尽**: 状态为 “火” 的元胞, 其状态在下一时刻将变成 “空”, 用以模拟着火的树经过一定时间后被烧完.

图 15: 烧尽规则

- **新生**: 状态为 “空” 的元胞, 在下一时刻以一个低概率 p 变为 “树”, 用以模拟新树的长出.

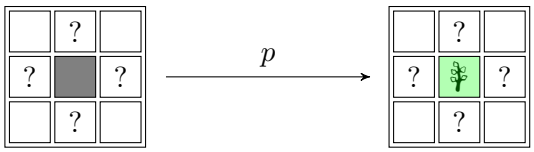

图 16: 新生规则

以上的三条规则中, 着火规则中的闪电以及新生规则都涉及概率, 并体现了该模型在演化过程中出现的随机性和不确定性. 图17是森林火灾元胞自动机模拟效果图.

假设元胞空间足够大以至于没有临界作用产生, 在周期性边界条件下, 开始于一个任意的原始条件, 系统经过一段时间后达到稳定状态, 此时系统可由燃烧着树的密度 ρ_f, 空地密度 ρ_e, 树的密度 ρ_t 三个参数值描述. 这三个参数之和为一, 即:

$$\rho_f + \rho_e + \rho_t = 1 \tag{2}$$

系统达到稳定状态时, 正在生长的树的数量与正在燃烧中的树的数量相等, 即

$$\rho_f = p\rho_e \tag{3}$$

树木大量生长的前提条件是燃烧着的树的密度 ρ_f 很小, 若 ρ_f 很大时, 树是不可能成长为森林. 式 (3) 表明只有当树生长的速率 p 接近于 0 时发生火灾的概率才会很小. 为了能保证大片森林的形成, 闪电的概率 f 还必须满足 $f \ll p$. 当闪电击中一小丛树, 而且没有树在它们周围时它们会烧的很快. 但是, 如果闪电击中了一大丛树时, 将需要一段甚至很长的时间去燃烧, 而且在燃烧的同时周边地区还有一些树木生长, 这样的火灾很难扩大. 为了观察其中关键的自我复制的行为, 大树丛与小树丛必须以同样的方式去燃烧. 当取树木的生长速率 p 足够小, 使得即使是大规模的树林在周围没有新树生长的条件下也可以迅速的燃烧. 在这种情况下, 系统的变化取决于

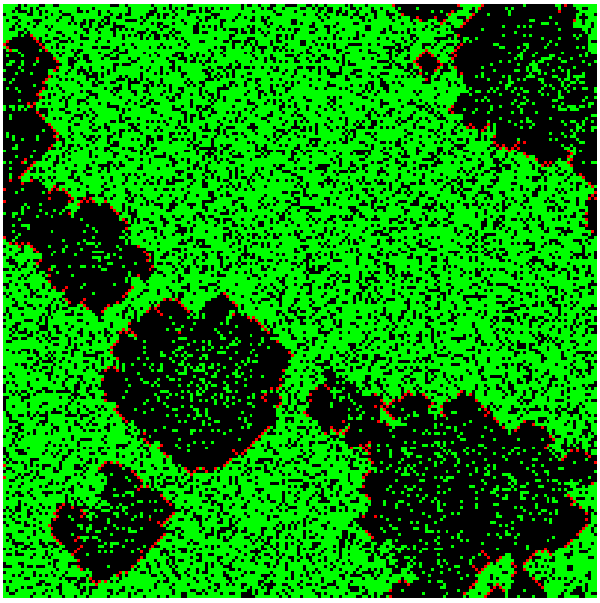

图 17: 森林火灾元胞自动机模拟

f/p, 而不是 f 和 p 独立值. f/p 实际上给出了平均两次闪电间新生出的树的数量 [6, 7]. 当 f 和 p 以同样的比例减小时, 系统整体会以这种比例进行变化, 而不依赖两次闪电发生的时间间隔中生长的树的数量, 这种情形下森林迅速燃烧的方式可以用 $p \ll T_{s\max}$ 表示, 其中 $T_{s\max}$ 大规模森林燃烧需要的时间. 综上所述, 概率元胞自动机在森林火灾模型中的运用条件为

$$f \ll p \ll T_{s\max} \tag{4}$$

其中 f 为闪电引起树变为火的概率, p 为长出新树的概率, $T_{s\max}$ 大规模森林燃烧需要的时间.

森林火灾元胞自动机程序实现

利用 MATLAB 强大的逻辑运算功能, 森林火灾元胞自动机很容易实现, 图17所示的森林火灾元胞自动机模拟效果图是由以下程序代码执行后得到的, 代码如下:

```
n = 200;              % 表示森林矩阵的尺寸: n x n
Pltg = 5e-6;          % 闪电的概率
Pgrw = 1e-2;          % 生长的概率
NW = [n 1:n-1];       % 用于构造北邻居veg(NW,:)和西邻居veg(:,NW)
SE = [2:n   1];       % 用于构造南邻居veg(SE,:)和东邻居veg(SE,:)
veg = zeros(n);       % veg = {0表示空, 1表示火, 2表示树}
imh = image( cat(3,(veg==1),(veg==2),zeros(n)) );

for i=1:3000
    % 周围四个邻居中状态为火的数量
    num =             (veg(NW,:)==1) + ...
        (veg(:,NW)==1)      +       (veg(:,SE)==1) + ...
                          (veg(SE,:)==1);

    veg = 2*( (veg==2) | (veg==0 & rand(n)<Pgrw) ) - ...
            ( (veg==2) & (num >0 | rand(n)<Pltg) );

    set(imh, 'cdata', cat(3,(veg==1),(veg==2),zeros(n)) );
    drawnow
end
```

以上程序构造了一个 $n \times n$ 的数组 veg 来表示一片森林, 数组中分别用 0, 1, 2 表示空, 火, 树三种元胞状态. 在周期边界条件下, 向量 NW 用于构造所有元胞的北邻居 veg(NW,:) 和西邻居 veg(:,NW), SE 用于构造南邻居 veg(SE,:) 和东邻居 veg(SE,:). 程序的11至13行求得所有元胞东南西北四个邻居中为火的数量 num. 比如 num(i,j) 就表示在 veg 中第 i 行 j 列的元胞周围四个元胞中状态为火的数量. 程序的15, 16行是森林元胞自动机规则的体现:

- 第15行程序可以分解成 2*(veg==2) 和 2*(veg==0 & rand(n)<Pltg) 两部分, 这两部分分别表示了 veg 中某元胞下一时刻的状态为树的两种前提条件: 要么该元胞本身的状态就是树; 要么该元胞的状态为空, 但满足生长概率.
- 第16行程序可以分解为 veg==2 & num >0 和 veg==2 & rand(n)<Pgrw 两部分, 这两部分分别表示了 veg 中某元胞下一时刻状态为火的两种条件: 当前时刻该元胞状态为树, 且周围四个邻居中有状态为火的元胞; 当前时刻该元胞状态为树, 且该元胞满足闪电的概率.

这两行程序很巧妙地用减号连接了起来, 当第16行程序的逻辑表达式为真时, 则15行的逻辑表达式的结果会被减去 1, 恰好实现了元胞状态从树变为火的转变. 需要注意的是15, 16行的程序中隐含了一种情况: 对于 veg 中为 1(火) 的元胞下一时刻变为 0(空). 第7行程序是将矩阵可视化, 用黑色, 红色和绿色的格子分别表示状态为空, 火和树的元胞. 第18行程序则是更新图形的颜色数据.

交通流 NS 元胞自动机规则介绍

在第2节中已经给出了 184 号交通流元胞自动机模型. 不过大多情况下, 用 184 号规则来模拟交通问题还是过于简单, 很难捕捉复杂的交通现象. 1992 年, 德国物理学家 Kai Nagel 和 Michael Schreckenberg 184 号规则基础上提出了一维交通流 Nagel Schreckenberg(NS) 模型 (又称 NS 规则)[8]. NS 规则是最常用的交通流模型之一, 该模型即简单又能模拟出交通堵塞的特征, 比如能够展现出当交通拥挤时, 车辆的平均速度变小的行为. 单行道 NS 模型也是一个概率元胞自动机, 每辆车的状态都由它的速度和位置所表示. 图18是 NS 模型的四条规则的示意图, 初始时状态四辆车的位置分别为 $x_1 = 1$, $x_2 = 3$, $x_3 = 6$,$x_4 = 7$, 速度分别为 $v_1 = 2$,$v_2 = 1$,$v_3 = 1$,$v_4 = 0$. NS 模型的四条规则分别为:

1. **加速规则**: 司机总是期望以最大的速度行驶. 如果第 n 辆车的速度未达到最大速度, 则其速度增加 1: $v_n \longrightarrow \min(v_n + 1, v_{\max})$. 系统允许的最大速度为 $v_{\max} = 2$, 经过加速后, 图18中四辆车的速度分别变为 $v_1 = 2$, $v_2 = 2$, $v_3 = 2$, $v_4 = 1$.
2. **防止碰撞**: 为避免与前车碰撞, 若第 n 辆车的速度 v_n 超过了前方空元胞数 d_n, 则其速度强制降为 $d_n - 1$: $v_n \longrightarrow \min(v_n, d_n - 1)$. 经过防止碰撞规则后, 图18中四辆车的速度分别变为 $v_1 = 1$, $v_2 = 2$, $v_3 = 0$, $v_4 = 1$.
3. **随机减速**: 由于诸多不确定因素, 以及司机的过度反应, 都会造成的车辆的减速. 若第 n 辆车的速度 $v_n > 1$, 则 v_n 以一个较低的概率降低 1: $v_n \xrightarrow{p} \max(v_n - 1, 0)$. 经过随机减速后, 图18中四辆车的速度分别变为 $v_1 = 0$, $v_2 = 2$, $v_3 = 0$, $v_4 = 1$.
4. **位置更新**: 模拟车的前进, 若第 n 辆车的速度为 v_n, 位置为 x_n, 则下一时该的位置为: $x_n = x_n + v_n$. 经过位置更新后, 图18中四辆车的速度分别变为 $x_1 = 1$, $x_2 = 5$, $x_3 = 6$, $x_4 = 8$.

用计算机无法直接模拟一条无限长的车道, 通常用周期性边界构造一条环道来模拟车流密度固定的无限长车道的交通情况. 也有应用开放边界条件, 在车道的一端不断地有随机生成的车辆流入, 当车辆通过车道的另一端则从系统中移除. 如果没有特殊考虑, 周期性边界就能很好的满足模拟的要求. 相对开放边界条件, 周期性边界条件 (封闭环行车道) 更为简单, 更容易控制和固定车流密度, 平均速度和流量也不会有太大波动, 便于测试不同密度下的各种参量. 日本及 MIT 的学者对封闭环形车道内的交通情况都做过实验 [9] 和研究, 非常有意思, 有相关的报道 [10, 11]

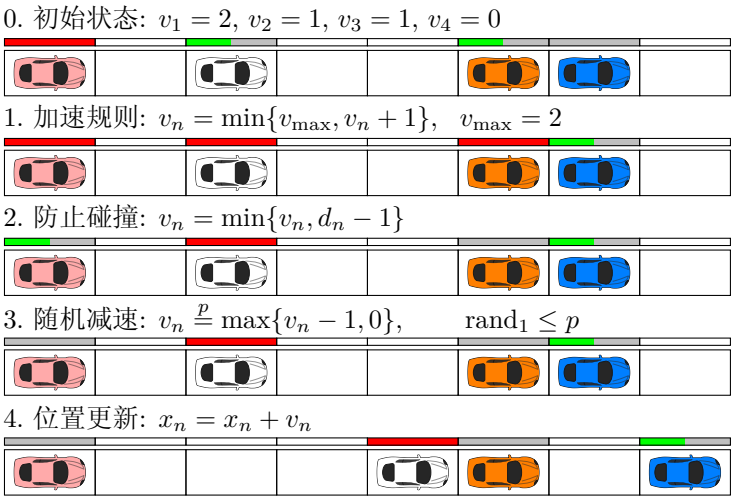

图 18: 交通流的元胞自动机 NS 模型

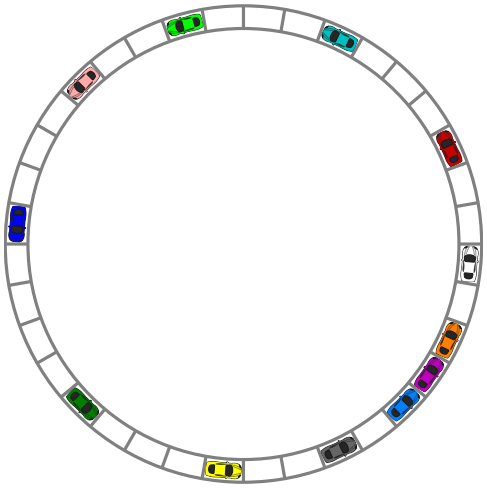

图 19: 单行道 NS 规则的模拟 ($L = 36$, $\rho = 1/3$, $v_{\max} = 5$)

和实验视频 [12] 及动画 [13]. 因此有一些现象和结论可以供参考和对比. 图19在周期性边界条件下 (环形车道), 交通流的元胞自动机 NS 模型的模拟效果. 图19是单行道 NS 规则模拟的效果图 (为了出图效果, 效果图中的道路长度被降为 36). 图19中形成阻塞的现象与实验视频 [12] 及动画 [13] 非常相似.

如果将随机减速的概率设置为 0, 去掉 NS 规则中的随机因素, 则通过简单的分析就可以得到单行道元胞自动机的 NS 模型的一些性质. 比如系统平衡时, 系统的平均速度和平均流量 [14]

$$v = \min(v_{\max}, d), \quad J = \min(\rho v_{\max}, 1 - \rho) \tag{5}$$

具体分析过程可见参考文献 [14]. 通过模拟, 也可以发现随机减速的概率 p_{brak} 设为 0 时, 系统最终会达到一种平均, 即系统以一定的周期做循环运动, 系统的平均速度也不再改变. 图20是密度与流量的观测值及模拟值的对比, 左图为观测值 [15], 右图中的点为本文的模拟得到的值, 红线是由式 (5) 给出的理论值. 图20中右图的模拟结果是由一条长为 $L = 100$ 的封闭环形车道, 车辆的最大速度 $v_{\max} = 5$, 改变车流密度 $\rho \in [0, 1]$, 每种密度情况下模拟多次, 测试不同密度情况下的平均流量和平均速度得到的. 此外, 本文还对比了航拍得到的时空轨迹图 [9] 和模拟得到的时空曲线图 (随机减速的概率 $p_{\text{brak}} = 0.15$), 具体如图21所示. 通过图20和图21, 可以说明 NS 规则能合理的反映交通流的特点.

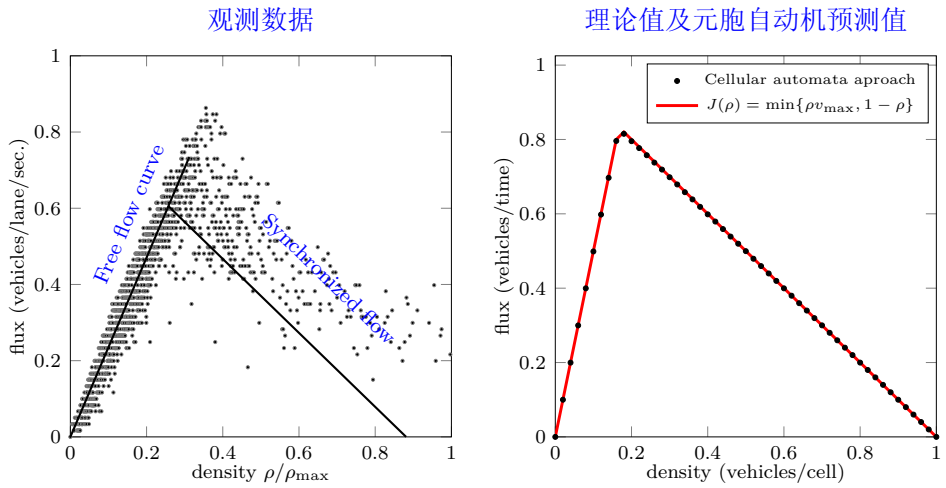

图 20: 单车道交流 NS 规则: 密度与流量

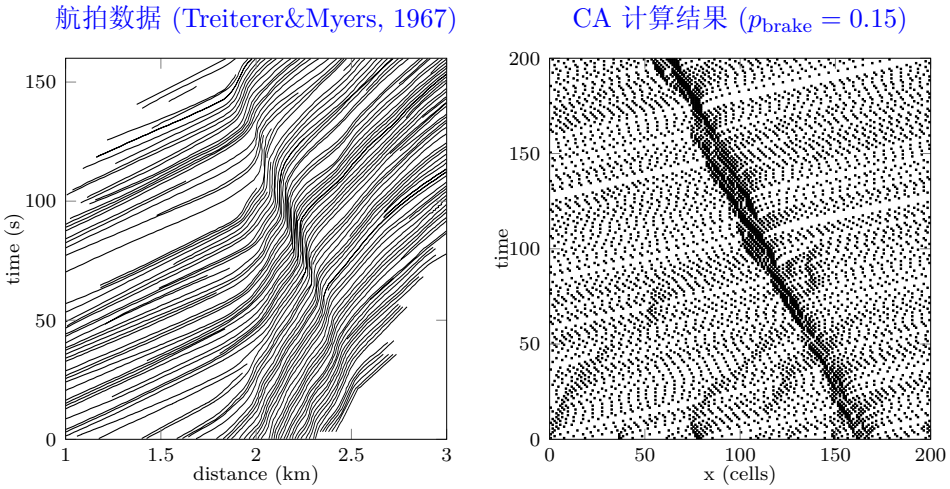

图 21: 单车道交流 NS 规则: 时空轨迹

交通流 NS 元胞自动机程序实现

利用 MATLAB 强大的数组运算, 单环形车道交通流的元胞自动机 NS 模型很容易实现, 可以定义如下函数来实现 ns 模型的模拟并计算出平均流量等参数.

```
function [flux, vmean] = ns(rho, p, L, tmax)
% 输入: rho-密度; p-随机减速概率; L-环形车道长; tmax-模拟总时间步数
% 输出: flux-平均流量; vmean-平均速度
%
ncar = ceil(L*rho);                 % 根据密度 ρ 计算系统中车辆数 n_car = ⌈L·ρ⌉
vmax = 5;                           % 车辆最大行驶速度
x = sort(randperm(L, ncar));        % 位置向量: 随机生所有车辆的位置, 并排序
v = vmax * ones(1,ncar);            % 速度向量: 初始化所有车辆的速度为 v_max
flux = 0; vmean = 0;                % 流量和平均速度: 初始化为零

h = plotcirc(L, x, 0.1);            % 绘制环形车道和车辆, 并返回车辆句柄

for t = 1:tmax
    v = min(v+1, vmax);             % 加速规则
```

```
15
16      gaps = gaplength(x, L);
17      v = min(v, gaps-1);              % 防止碰撞
18
19      vdrops = ( rand(1,ncar)<p );
20      v = max(v - vdrops, 0);          % 随机减速
21
22      x = x + v;                       % 位置更新
23
24      passed = x>L;                    % 找出超出边界的车辆
25      x(passed) = x(passed) - L;       % 周期边界
26
27      if t>tmax/2
28         flux  = flux  + sum(v)/L; % 计算流量: f = ρ·v̄ = ∑v/n_car · n_car/L = ∑v/L
29         vmean = vmean + mean(v);  % 计算平均速度
30      end
31      plotcirc(L, x, 0.1, h);          % 更新图像, 以形成动画
32  end
33
34  flux  = flux/(tmax/2);               % 流量对时间平均
35  vmean = vmean/(tmax/2);              % 平均速度对时间平均
```

以上 MatLab 程序定义了一个 ns 函数, 只需给出环形车道上的车辆密度, 随机减速概率, 环形车道长度和模拟时间步, 该函数就能计算出环道上的平均流量和流速. 以上程序比较简单, 仅以下几行需要额外解释一下:

- 第7行中的randperm(L,ncar)函数是从 $[1, 2, \cdots, L]$ 中随机抽取ncar个数, 以表示ncar辆车初始随机分布于环形车道上, 再由sort函数排序, 向量x中的车辆按位置从小到大排列.
- 第24-25行是实施周期性边界条件, 在经过第22行的位置更新后, 可能会有车辆的位置超出了L, 对于位置超出L的车辆 (passed为真的车辆) 需要减掉一个周期L, 使车辆的位置仍然处于 1 到L之间.
- 第27-30行用来计算平均流量和平均速度, 前tmax/2的时间用来让系统达到平衡, 平均流量和平均速度的计算仅在t>tmax/2的时间里进行. 这里是通过计算 $\rho\bar{v}$ 来计算流量的. 流量的计算也可通过选取一点, 计算单位时间通过的车辆数来计算, 比如计算passed为真的平均数量.

以上程序中还调用了两个自定义的函数 gaplength 和 plotcirc. 函数 gaplength 用来计算相邻两边间的距离, 函数 gaplength 的主要代码如下:

```
1  function gaps = gaplength(x, L)
2  % 输入: x-各车辆位置; L-环形车道长        输出: gaps-相邻两车间的距离
3  %
4
5  ncar = length(x);                    % 获得系统中的车辆数
6  gaps = inf*ones(1, ncar);            % 初始化相邻两车间的距离
7  if ncar>1
8      gaps = x([2:end 1]) - x;         % x([2:end 1]) - x(1:end-1 end)
9      gaps(gaps<0) = gaps(gaps<0) + L;      % 周期性条件
10 end
```

以上程序中的第8行用于计算两邻两车的距离. 对于第 1 辆车, gaps(1) = x(2)−x(1); 对于第 i 辆车, gaps(i) = x(i+1)−x(i); 周期性边界构成首尾相接的环形车道, 对于最后一辆车 gaps(end) = x(1)−x(end), 这时可能出现gaps中出现负值, 第9行通过对gaps中小于 0 的值加一个周期 L 来保证所有车辆都有正确的与前车的距离.

函数 plotcirc 用来将数据图形化, 动态地显示每辆车的位置, 其主要代码如下:

```
1  function h = plotcirc(L, x, dt, h)
2  % 输入: L-环形车道长; x-各车辆位置; dt-动画相邻两帧间隔时间; h-车辆句柄
3  % 输出: h-车辆句柄
4  %
5
6  W = 0.05;  R = 1;                    % 车道的宽度; 环形车道的半径
7  ncar = length(x);                    % 获得系统中的车辆数
8
9  theta = 0:2*pi/L:2*pi;               % 环形车道上径向网格线的极角
10 xc  = cos(theta); yc  = sin(theta);          % 径向网格线的中心
11 xin = (R-W/2)*xc; yin = (R-W/2)*yc;          % 径向网格线内端点
12 xot = (R+W/2)*xc; yot = (R+W/2)*yc;          % 径向网格线外端点
13
14 xi = [xin(x);  xin(x+1); xot(x+1); xot(x)];% 车辆所在网格的四个端点
15 yi = [yin(x);  yin(x+1); yot(x+1); yot(x)];%
16
17 if nargin == 3                       % 如果函数输入量等于3, 用于首次绘图
18     color = randperm(ncar);          % 为各车辆随机生成一个独立的颜色
19     h = fill(xi,yi, color); hold on          % 画出所有车辆
20     plot([xin; xot] ,[yin; yot] ,'k','linewidth',1.5); % 画径向网格线
21     plot([xin; xot]',[yin; yot]','k','linewidth',1.5); % 画环形网格线
22     axis image;
23 else                                  % 如果函数输入量等于4, 用于图像更新
24     for i= 1:ncar                     % 更新图像中车辆的位置
25         set(h(i),'xdata',xi(:,i),'ydata',yi(:,i));
26     end
27 end
28 pause(dt)                            % 暂停dt秒
```

在以上程序中, 第17行中的nargin为调用该函数时实际输入参数的个数. 输入参数的个数可以为 3, 也可以为 4. 输入参数个数为 3 时, 用于首次调用函数 plotcirc 来绘制车道和车辆的初始位置, 并返回表示所有车辆的矩形的句柄 h. 输入参数个数为 4 时, 额外的一个参数为表示每个车辆的四边形的句柄 h, 这种情况下无需重新绘制车道和车辆, 只需通过第25行的set来更新车辆的位置.

参考文献

[1] Cellular automaton - wikipedia, the free encyclopedia. Available at http://en.wikipedia.org/wiki/Cellular_automaton[Accessed March 1, 2014].

[2] Conway's game of life, November 2014. Available at http://en.wikipedia.org/wiki/Conway's_Game_of_Life[Accessed October 10, 2014].

[3] Wolfram atlas: Elementary cellular automata:index of rules. Available at http://atlas.wolfram.com/01/01/rulelist.html[Accessed October 10, 2014].

[4] Elementary cellular automaton, October 2014. Available at http://en.wikipedia.org/wiki/Elementary_cellular_automaton[Accessed October 10, 2014].

[5] Forest-fire model, May 2014. Available at http://en.wikipedia.org/wiki/Forest-fire_model[Accessed August 10, 2014].

[6] S. Clar, K. Schenk, and F. Schwabl. Phase transitions in a forest-fire model. *Physical Review E*, 55(3):2174–2183, March 1997.

[7] Peter Grassberger. On a self-organized critical forest-fire model. *Journal of Physics A: Mathematical and General*, 26(9):2081, 1993.

[8] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. 2(12):2221 - 2229. Available at http://fix.bf.jcu.cz/~berec/NagelSchreckenberg1992.pdf[Accessed March 1, 2014].

[9] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. 10(3):033001. Available at http://iopscience.iop.org/1367-2630/10/3/033001/pdf/1367-2630_10_3_033001.pdf[Accessed March 1, 2014].

[10] Experiment creates test track jam. Available at http://www.traffictechnologytoday.com/news.php?NewsID=3962[Accessed March 1, 2014].

[11] 堵车有哪些原因? Available at http://www.zhihu.com/question/20781120[Accessed March 1, 2014].

[12] Shockwave traffic jam test - video. Available at http://www.maniacworld.com/shockwave-traffic-jam-test.html[Accessed March 1, 2014].

[13] Formation of a 'phantom traffic jam' | MIT video. Available at http://video.mit.edu/watch/formation-of-a-phantom-traffic-jam-8152/[Accessed March 1, 2014].

[14] Kai Nagel and Hans J. Herrmann. Deterministic models for traffic jams. *Physica A: Statistical Mechanics and its Applications*, 199(2):254–269, October 1993. Available at http://www.comphys.ethz.ch/hans/p/156.pdf[Accessed March 1, 2014].

[15] Benjamin Seibold, Morris R. Flynn, Aslan R. Kasimov, and Rodolfo Ruben Rosales. Constructing set-valued fundamental diagrams from jamiton solutions in second order traffic models. Available at http://arxiv.org/pdf/1204.5510.pdf[Accessed March 1, 2014].