南通大學

本科毕业设计

题目	基于元胞自动机模拟和遗传算法改进的动态网络分配模型分析

学生姓名: 叶 冬 冬

专　　业: 交通运输

指导教师: 曹　　阳

完成日期: 2019.5.12

诚 信 承 诺 书

本人承诺：所呈交的论文是本人在导师指导下进行的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签　名：_____日 期：_____

本论文使用授权说明

本人完全了解南通大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（**保密的论文在解密后应遵守此规定**）

学生签名：_____指导教师签名：_____日期：_____

摘　要

智能交通系统的研究主要是为了建立现代化交通运输系统和交通信息管理系统，而其中的核心领域之一就是动态网络分配模型和算法。

基于经典的 NaSch 模型，本文对模拟交通路网的元胞自动机模型进行了深入研究，提出了两类新模型，即车道元胞自动机模型和交叉口元胞自动机模型。本文详细阐明了路径搜索原则，当问题从一维变为二维时，还加入了交叉口模型以及转向模型，得到了一套自适应的交通网模拟系统，从而根据其中的数据以及当前所研究的各种动态网络分配模型定义了该系统的双目标函数，并采用 NSGA—II 算法对该双目标规划模型进行了求解。此外，本文还对一个二维路网进行了数值模拟，并对模拟结果进行了详细分析。

研究结果表明本文提出的模型可以较好的用于动态网络分配。数值实验结果表明，在对交通网络基于完善规则的模拟的情况下，应用恰当的算法可以对交通网进行一定程度的优化。

关键词：动态网络分配；元胞自动机；NaSch 模型；遗传算法；双目标规划

ABSTRACT

The research of intelligent transportation system is mainly to establish a modern transportation system and traffic information management system, and one of the core areas is dynamic network allocation model and algorithm.

Based on the classical NaSch model, this paper makes an in-depth study on the cellular automaton model of the traffic network. Two new models are proposed, namely the lane cellular automaton model and the intersection cellular automaton model. In this paper, the path search principle is clarified in detail. When the problem changes from one-dimensional to two-dimensional, the intersection model and the steering model are added, and an adaptive traffic network simulation system is obtained, based on the data and the current research. The various dynamic network allocation models define the dual objective function of the system, and the double objective programming model is solved by NSGA-II algorithm. In addition, the paper also numerically simulates a two-dimensional road network and analyzes the simulation results in detail.

The research results show that the proposed model can be used for dynamic network allocation. The numerical experiments show that the traffic network can be optimized to a certain extent by applying appropriate algorithms in the case of the traffic network based on the simulation of perfect rules.

Keywords: Dynamic Network Allocation; Cellular Automaton; NaSch; Genetic Algorithm; Bi-objective Planning

目 录

第一章 绪 论

1.1 研究背景

面对经济的发展和社会进步，交通规划者和交通工程师在解决交通问题或优化交通系统时不再仅仅关注拥堵。近年来，道路运输对环境的福祉产生负面影响并使其承载能力恶化这一事实引起了极大的关注。据美国环境保护署（USEPA）称，公路运输（汽车，卡车和公共汽车）被认为是一氧化碳（CO），二氧化氮（NO x）和二氧化硫（SO x）的最大来源。燃料的燃烧，以及通过大气中的化学反应是产生颗粒物质（PM）的反应物的主要贡献者。车辆排放明显导致各种健康问题，包括癌症，心血管和呼吸系统疾病以及围产期死亡率。运输也是温室气体排放的重要来源，这导致不可逆转的人为全球变暖。科学家们认为，从长远来看，气候变化和损害成本对于生态退化可能是灾难性的。这些警告强调需要加大努力，提高交通运输部门的环保意识，并且必须监测交通与环境之间的关系，以及发展确保经济增长的交通系统，同时又要具备可持续性，并保护生态系统。

交通事件（车辆事故，道路封闭，需求高峰等）每天都在道路网络上发生。大多数典型影响与当地交通容量减少或全球交通流量增加有关，这可能导致道路过饱和，拥堵和延误。到目前为止，人们已经提出了许多方法来减少静态和动态情况下的交通拥堵，包括（1）经济政策，例如拥堵定价，可交易信用，奖励，过境补贴，停车定价；（2）工程控制方案，例如速度限制，匝道计量，车牌配给，车道控制等。尽管在一些城市已经实施了流行的拥堵定价方案，但要消除公众不愿意接受通行费的方法仍有很长的路要走。对于可交易的信用额度，据我们所知，尚未在任何城市实施。对于工程控制方案，大多数研究基于仿真方法研究这些方案对网络性能的影响。

上世纪 90 年代智能交通系统（Intelligent Transportation System，简称 ITS）的研究应运而生，这项研究主要是为了建立现代化交通运输系统和交通信息管理系统。其中最核心的领域就是动态网络分配（Dynamic Traffic Assignment，简称 DTA）模型和算法。由于此类问题具有强大的时变性和巨大规模的特征，没有确切的目标函数形态，故目前并没有行之有效的求解模型的算法。为实现向道路网上的车辆提供实时的道路信息，计算最佳的行驶路径，实时诱导交通流的目标，尚且需要更深入的理论与方法的研究。

1.2 研究目的与意义

随着城市道路的不断发展，交通需求量也在不断增长，目前仅仅依靠交通设施的扩建已经无法满足城市道路交通的出行需求。城市交通动态网络分配在城市运行中将占据重要地位。本文主要聚焦于现有的动态网络分配模型，分析微观车辆在交叉路口的转向以及信号控制的影响，并做模拟交通流的实时运行情况，利用遗传算法进行信号配时的优化，建立有效的动态网络分配的模型与方案。

1.3 国内外研究现状

1.3.1 国外研究现状

一般有两种描述交通流状况的模型，包括道路交通流模型和动态道路网络交通流模型。其中道路交通流模型的发展始于 20 世纪三四十年代的概率论模型，经过了运动学模型、跟驰模型、流体动力学模型的不断推进，发展到 90 年代的元胞自动机模型。道路网络交通流模型的研究始于 20 世纪的出行路径选择的用户均衡原理与等价数学规划模型，随后出现了随机效用理论、组合网络模型、网络设计问题的模型与求解、随机均衡分配模型，到 90 年代的动态分配模型。2018 年，Kucharski[1]等人提出了信息合规模型（ICM），该模型扩展了 Gentile 在 2016 年提出的宏观单日动态交通网络分配模型，当交通事件及其对拥塞的影响信息在驾驶员之间和交通网络上传播时，ICM 能够模拟预测出事件在时间与空间中路径重塑的演变，从而创造性地将 DTA 中动态用户均衡的概念进一步细化。2018 年，Zhao[2]等人用图解法分析了基于单日激励的路径选择策略的三交替网络动态交通分配问题，基于变分法绘制了动态系统最优（DSO）分配路线，比较完整地提出了实现 DSO 的一种策略。2018 年，Friesz[3]等人将变分不等式理论应用于动态用户均衡问题，旨在为研究人员提供一种新的参考。2018 年，Wang[4]等人综述了 DTA 模型在环境可持续公路运输应用中的最新方法学进展，包括有关环境排放和排放定价的交通信号控制。这些模型极大地推动了动态交通网络分配的研究方法，从不同角度为解决动态交通网络分配问题作了巨大贡献。

1.3.2 国内研究现状

我国对动态交通网络分配的研究起步较晚，但进步迅速。2016 年，孙瑜[5]仔细研究了交通路段与交叉口的通行能力在离散化的交通行为中的约束条件，提出并且构建了基于路网超点模型的交通动态网络分配模型。此外最短路径方面提出了改进版本的深度优先搜索算法用来对动态网络分配中的路径搜索作出决策，并且将非支配排序的遗传算法应用于交

通流分配中双目标函数的求解。2016 年，俞灏[6]在交通传输模型中加入了信号控制，从而提出了网络动态交通流传输模型，除此之外还将动态用户均衡条件融入该模型，构建了动态网络分配以及信号控制的协作模型，广泛地剖析了两者之间的协同工作状态。2017 年，费文鹏[7]在研究动态网络分配时发现，动态环境下的最短路径算法还有提升的空间。在迪杰斯特拉最短路径算法的基础上，考虑了路段阻抗的瞬时变化对最短路径的影响，从而提出了改进的动态环境下的迪杰斯特拉最短路径算法。2017 年，张艳[8]构筑了一个小型的随机变化的路网，认为出行者出行的路径选择标准以理想的出行费用最小为依据，构建了多种出行模式下的动态网络分配模型以及相应的变分不等式模型，在此基础上仔细研究了相关的求解模型的算法流程。2017 年，张传琪[9]等人深入研究了城市路网拥挤的因素，即路网的拥挤程度与城市的工业聚落以及居民区分布有关，故可以根据各种人群的出行时间，确定最小的出行时间成本，决策最佳的出行时刻，提出了动态出行拥挤条件下的多模式车辆路径模型，并给出了相应的求解算法。2017 年，李东岳[10]另辟蹊径，基于 METANET 以及系统动力学的思想，模拟出了交通系统自然演化的过程，构建了相应的发展动态方程。根据相关的实际的路段出行数据以及路段的结构信息，提出了改进的动态反馈的 OD 逆推模型。2017 年，余婷[11]则从阻抗计算细化的角度出发，将阻抗计算为路段阻抗的计算以及节点阻抗的计算，在此基础上演化出了动态网络分配最短路径模型。2017 年，李潇逸[12]在元胞传输模型的基础上做出创新，车流的在路径选择中会实时变化，故在无流量约束的离散化条件下，构建了动态网络最优分配模型。2018 年，王钰[13]提出了新的预测算法，预测的进程随着时间的滚动而变化。2018 年，项俊平[14]应用了多种智能规划算法，设计出优化车辆出行路径的信号控制算法。2018 年，邹娟[15]在元胞传输模型的基础上，结合蜜蜂算法进行优化信号配时模型的研究。2018 年，李杰[16]等人在 NaSch 模型的基础上，重新定义了交通行为的相关规则，构建了考虑周期边界条件的双向四车道交通流元胞自动机模型。2018 年，夏运达[17]在仔细研究慢化概率的基础上，改进了 VE 模型。

1.4 主要研究内容

本课题利用元胞自动机模拟交通网，选用恰当的动态网络分配模型，应用 NSGA—II 算法进行求解进行相关分析。

第一章：绪论。主要解释研究的背景、目的、意义等，描述了当前随着城市经济与基础设施的发展，城市交通情况日益恶化，不采取适当的管理规划方案，对自然与社会都造成

危害。然后列举了当前国内外对动态网络分配领域的研究进展。

第二章：基于元胞自动机的城市动态交通网络分配模型。介绍了经典的道路交通流元胞自动机模型。提出了模拟交通路网的车道元胞自动机以及交叉口元胞自动机模型，并且阐明了路径搜索算法以及相应的双目标函数。

第三章：基于遗传算法的动态网络分配的求解。介绍了遗传算法的基本原理，进而引入 NSGA—II 算法来对上文所述的目标函数进行求解，简要阐明了求解的原理，以及详细描述了改进后的求解算法步骤。

第四章：基于元胞自动机的城市动态网络分配的数值仿真实验。提出了一个二维交通路网，对第三四章所论述的模型，利用 MATLAB 编程进行仿真数值实验。对运行结果得到的图像以及数据表格进行分析模型的可行性以及实用性。

第五章：结论。通过对当前该领域的发展现状，结合本文提出的模型进行理论分析。当前计算设备的发展，算法的研究以及开源平台的发展是为该领域进行交叉研究提供了良好的环境。

1.5 文章结构

见图 1.1 文章结构图

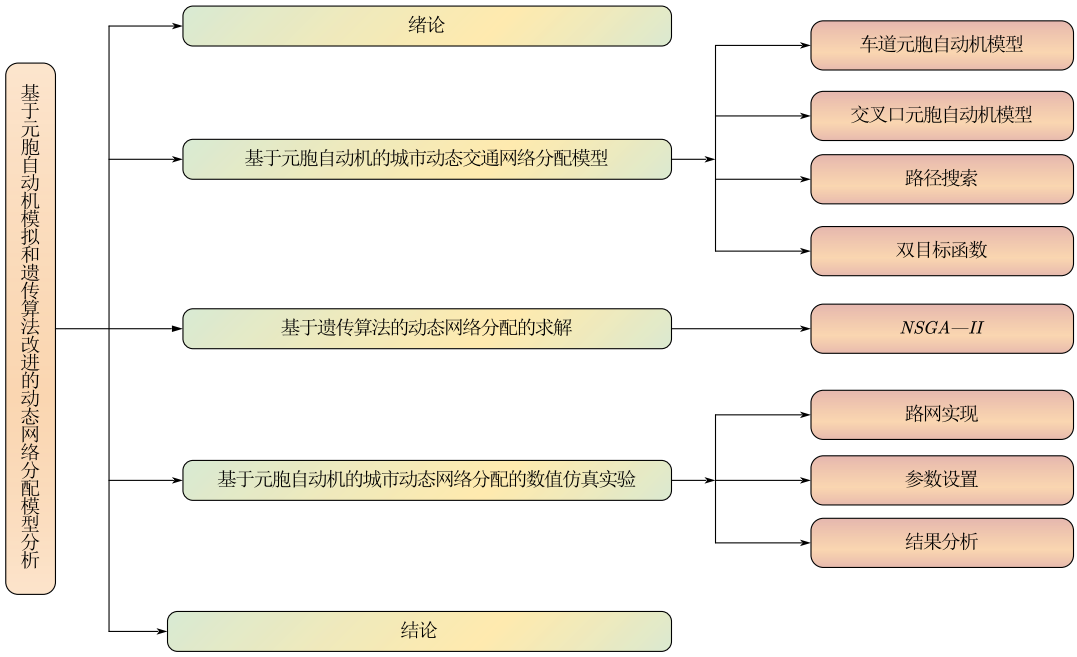


图 1.1 文章结构图

第二章 基于元胞自动机的城市动态交通网络分配模型

2.1 经典的道路交通流元胞自动机简介

2.1.1 NaSch 模型

NaSch 模型是由最初出版于 1992 年的 Kai Nagel 和 Michael Schreckenberg[30]的先前工作中提出的，在其论文中汽车被放置成一维阵列，其中每个元胞可以被占用或未被占用，并且汽车在每个占用的元胞中的速度被设定为零和指定的系统最大速度之间随机速度。多辆汽车不可能占用相同的元胞，并且该模型的每次迭代由同时发生的四个基本操作控制。这四个步骤是：

1) 加速

对于所有未达到最大速度$v_{\max}$（道路的速度限制）的车辆，以及前方有超过v_n+1个空单元的车辆，加速一个单位，即$v_n \to \min\{v_n+1, v_{\max}\}$。

2) 安全性减速

如果汽车前面有空元胞，并且在第一步之后它的速度大于d_n，则它将速度降低到v_n，$v_n \to \min\{d_n, v_n\}$，其中$d_n = x_{n+1} - x_n - l_n$，$x_n$代表车辆$n$的位置，$l_n$代表车辆$n$的长度。

3) 随机减速

对于速度大于 0 的汽车，速度以概率p减少一个单位，即$v_n \to \max\{v_n-1, 0\}$。

4) 车辆更新

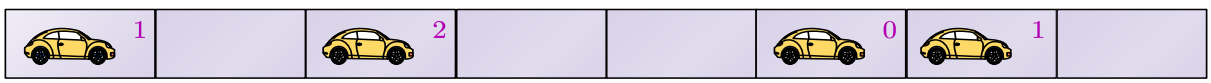
在步骤 1~3 之后，基于其当前速度v_n为每辆汽车分配新的位置$x_n \to x_n + v_n$。

对于该模型在时刻$t \to t+1$的演化过程如图 2.1 所示。

时刻t时车辆速度与位置的分布（数字显示车辆速度）

步骤 1：加速

步骤 2：减速



步骤 3：随机慢化

步骤 4：车辆更新

图 2.1 NaSch 模型的演化过程

根据前面的介绍对 NaSch 模型编程进行数值模拟。模型参数取值$L_{load}=1000$，$p=0.3$，$v_{max}=5$，边界条件为周期性边界。统计前 10000 个时间步可以得到交通流量与密度的关系，如图 2.2 所示。当密度为 0 ~ 0.17 时，流量随密度的增加而增加；当密度超过 0.17 时，流量开始随密度的增加而下降。

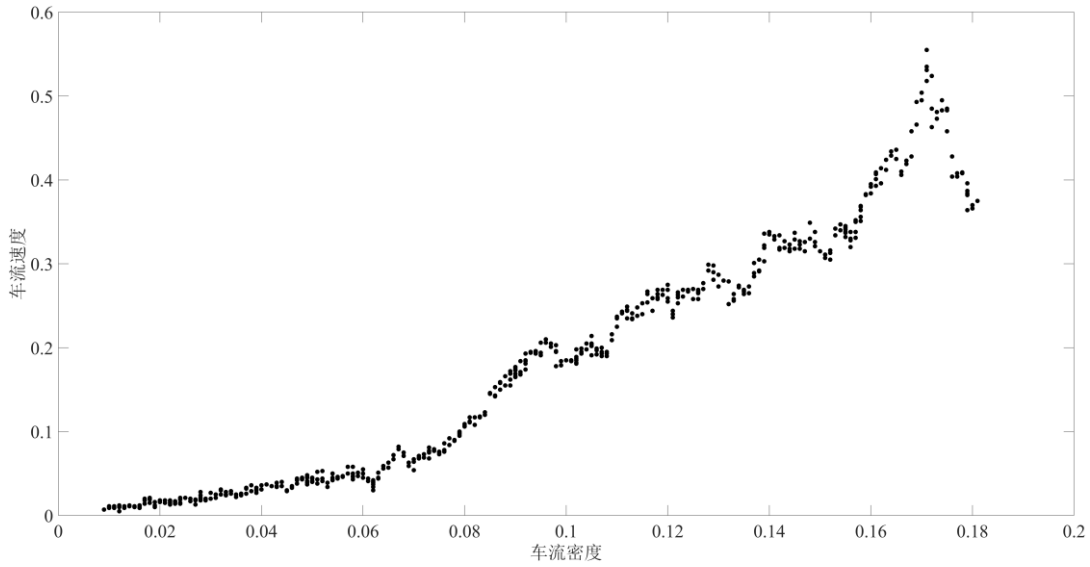


图 2.2 NaSch 模型模拟密度流量图

随机慢化概率以及最大速度的引入，使得 NaSch 模型可以描述一些实际且复杂的交通现象。例如可以模拟自发的堵塞现象以及交通拥堵。NaSch 模型的时空分布图如图 2.3 所示：

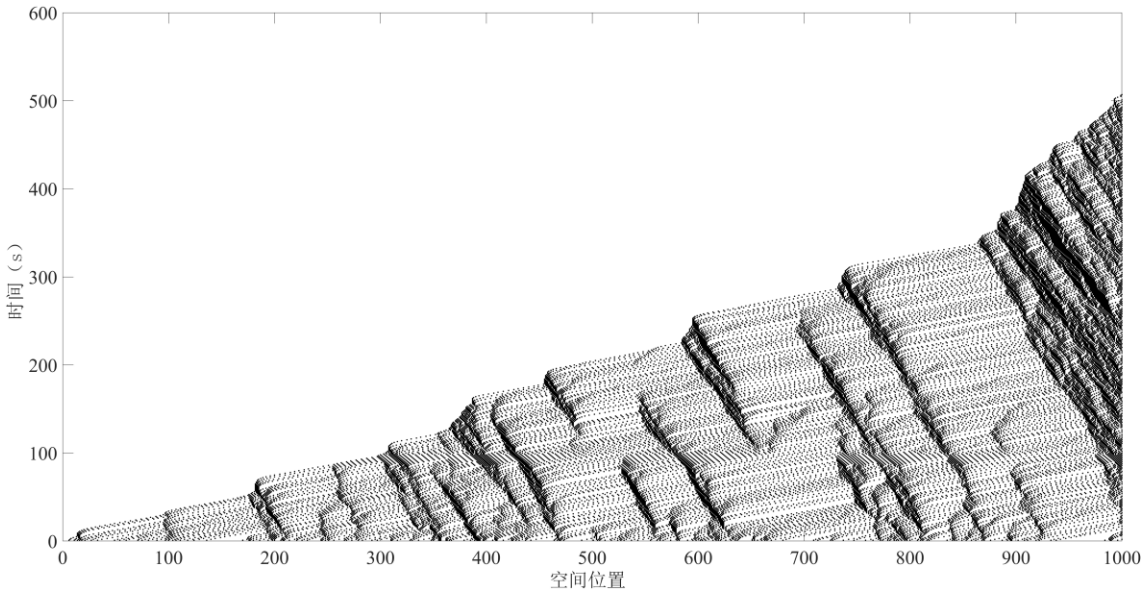


图 2.3 NaSch 模型时空分布图

从图中可以看出在一些特殊情况下会出现“幽灵拥堵”现象，随着时间的推移以及位置移动的逐步变化，拥挤现象的发生会越来越频繁。在基于元胞自动机的交通流模型中，周期性边界条件是核心问题。这里简要介绍一下周期性条件。

周期性条件：结束每次移动更新后，确定路段上各车辆的车头位置x_{lead}，如果$x_{\mathrm{lead}} > L_{link}$，车辆就会从路段的末端重新进入路段，从路段的末端转换到路段的始端，进而位置变为$x_{last} = x_{lead} - L_{link}$，$v_{last} = v_{lead}$。其中$x_{last}, x_{lead}, v_{last}, v_{lead}$分别代表首车，末车的位置与速度，$L_{link}$代表既定路段的长度。

2.1.2 BML 模型

前面介绍的 NaSch 模型以及由其衍生的多车道模型都是模拟的一维道路模型，而城市路网是二维的，故要实现纵横交错的交通路网需要新的模型来描述。1992 年，Biham O、Levine D A 与 Middelet A 三人提出了二维元胞自动机模型（BML）。该模型是在一个正方形网格上进行研究的。BML 模型是具有正方晶格的简单二维 CA 模型。格子的每个单元代表东向街道和北向街道的交叉点。交叉点之间的街道为了方便起见一般会进行简化处理。每个元胞可以是空的，当然也可以是在每个奇数步上向东的车辆，而在每个偶数时间步骤更新向北的车辆。根据以下规则实现车辆的速度更新。如果前面的单位是既定的规则下可以前进，则车辆就会按照既定的规则向前移动；除此之外，车辆会保持在原地不动。这样的一个过程可以对应实际交通行为中的一次信号灯循环。在这个模型中，所有东向车辆的初始数量等于所有北方车辆的初始数量，并且由于车道变化是不可能的，因此保留了每条街道上的车辆数量。初始时将车辆随机的分布，网格点有三种状态：空、由南北向车辆占

据、由东西向车辆占据。除此之外该模型还可实现信号灯功能。当时间步为奇数时，仅东西向车辆通行；当时间步为偶数时，仅南北向车辆通行。当其它车辆占着这个位置（元胞）的右侧毗邻元胞时，即使此刻该阻挡车辆向前驶出该位置，该车依然保持不动。虽然车辆在交通网络中的分布是无法预料的，但是整体不断迭代的过程上可以逐步显示出车辆位置一定的规律性。BML 模型的自组织自适应呈现出一种循序渐进的状态，且演化进程极为复杂，与实际交通流相似度极高，故具有极高的研究价值。

在 BML 模型的更新过程中，通过车辆的初始分布引入其不确定性。不同的车辆密度和初始分布导致两种不同的最终状态，即自由流动状态或在它们之间具有尖锐的一阶相变的干扰状态。确定最终状态是自由流动状态还是堵塞状态的车辆密度称为临界点ρ_c。如果车辆密度低于ρ_c，则所有车辆自由移动；否则，所有车辆都会缓慢移动并最终停止。

由于其简单性和高效性，BML 模型是城市交通建模的优秀模型。然而，大多数研究关注于从 BML 模型的自由流状态到堵塞状态的相变的理论分析。由于其局限性，很少有学者将其运用在交通问题的建模中。因此，BML 模型不适用于交通拥堵问题中的短时交通流预测问题。为了解决这样的问题，可以选择来自真实城市交通网络的路线，并将它们中的每一个映射到东向和北向的模型中。同一路线的双向映射表示同一道路上的双向交通。一条路线由几个路段组成。当两条路线交叉时，它们的交叉不仅可以是交叉路口，还可以是立交桥，隧道或角落。例如 M-BML 模型来预测车辆从不同方向引起的堵塞，以便在交通拥堵分析中忽略除当前交叉口以外的其他交叉口。此外，如果两条路线越过一条交叉路口，则需要进一步分析以确认实际堵塞的路线。最后，两条路线可能不会交叉，或者同一路线上的双向交通流可能是平行的。在这两种情况下，相应的单元格表示则空。因此，根据实际城市交通网络中每条东向路线与每条北向路线之间的关系，在运行 M-BML 模型后，可以通过设置不同的堵塞阈值来确认堵塞区域并缩小堵塞范围，直到最终获得模型中的堵塞单元。然后，通过使用从 M-BML 模型到真实城市交通网络的映射策略，可以确定真实城市交通网络中的堵塞交叉点。

2.2 交通路网元胞自动机模型改进

在动态网络分配的过程就是将同一节点或者路段具有相同起讫点的车辆分配到相应路段中去，主要考虑的是交通路网上的几何限制以及交叉口的转向限制。研究动态交通网络分配的重要环节就是研究交叉口的信号控制对交通流产生的决策影响，在车流运动中找寻

最佳决策方案是该研究需要达到的目标。为了有效模拟微观车辆在交通路网上的通行情况以及在交叉口的转向情况，需要提出合适的元胞自动机模型，此外还需要保证流量计算的约束以及满足流量守恒的条件。元胞自动机的运行中产生的实际数值方便理论分析，故接下来介绍本文用于研究动态网络分配的元胞自动机模型。

2.2.1 车道的元胞自动机模型

（1）安全行车距离的确定

跟驰理论是聚焦于单车道车辆行驶状态的一种理论，单车道的限制是无法超车，需要根据前车的状态调整自身状态。跟驰模型在交通安全，交通管理与道路通行能力等研究方向发挥着重要作用。了解单车道的交通运行特性需要对跟驰模型有着深入的研究，从而提高道路通行能力以及降低追尾事故。

当车流密度较小时，车辆之间间距较大，车辆以其最理想的速度前进，在整个车流来看即以自由流速度行进；当车流密度较大时，车辆之间间距较小，规律性地时走时停，偶尔会发生交通拥堵现象，故驾驶员驾驶过程中会高度集中注意力，保持与前车的安全距离。这种交通微波会在整个交通路网上不断传播。当然除此之外车辆的行驶状态还受其他因素的影响，例如路面的平整程度、道路的几何限制以及车辆自身的性能等，但其中的关键因素就是前后两车的距离。跟驰关系见下图 2.4。

为了防止车辆碰撞的发生，车辆之间会保持的一定的距离，这个距离就是安全距离。对于车辆的既定速度有相对应的安全距离，保持避免交通事故的最小距离是临界安全距离，也是最应该关注的安全距离。

现在给出临界安全距离的计算方法，令n车为前车，则$n+1$车为后车。在后车由发现前方车辆减速到自己开始减速到最终停车经历了三个阶段：反应时间t_1、启动制动时间t_2、减速到停止时间t_3，设t_1,t_2时间内车辆保持匀速行驶，且$T=t_1+t_2$，在车辆减速过程中，车辆做匀减速运动。分别设这个三个时间段的行驶距离为S_1、S_2和S_3。由此可以设后车初速度为v，减速过程的加速度为a_2，可得到公式(2.1)：

$$S_F=S_1+S_2+S_3=v(t_1+t_2)+\frac{v^2}{2a_2}=vT+\frac{v^2}{2a_2} \tag{2.1}$$

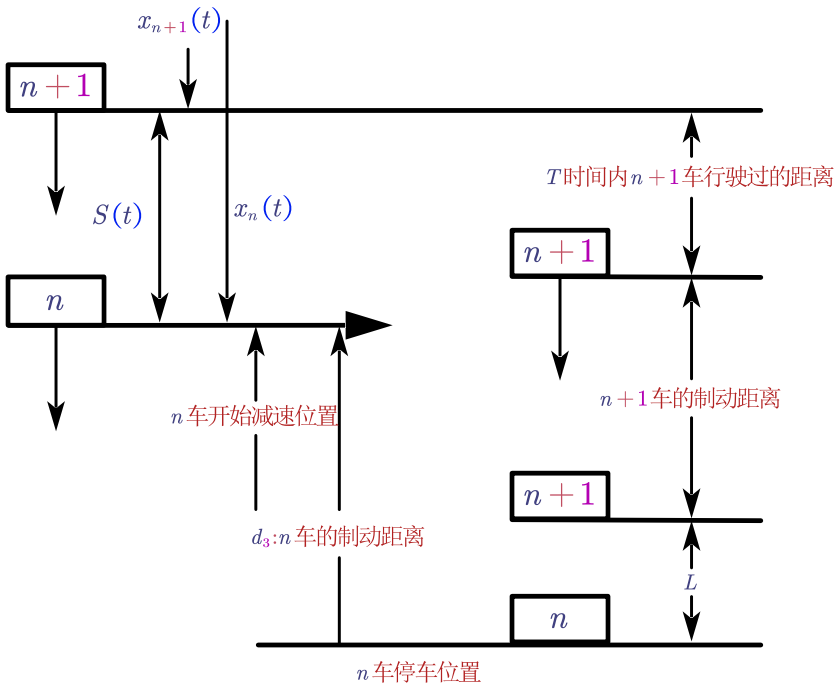


图 2.4 跟驰关系图

在这里可以设前后两车具有相同的初始速度，前车从开始减速到停止为止做匀减速运动，设其加速度为a_1，这段时间行驶的距离为d_3，见公式(2.2)：

$$d_3 = \frac{v^2}{2a_1} \tag{2.2}$$

设L为两车完全停止后两车车头之间的距离，故两车的临界安全距离见公式(2.3)：

$$S_r = S_F + L - d_3 = L + vT + \frac{v^2}{2}\left(\frac{1}{a_2} - \frac{1}{a_1}\right) \tag{2.3}$$

若前车因为某些故障或者撞上障碍物会跳过制动的情况，由速度v瞬间变 0，在这种情况下刹车距离则为 0，那么对应这种情况的临界安全距离的公式(2.4)：

$$S_r = S_F + L = L + vT + \frac{v^2}{2a_2} \tag{2.4}$$

（2）车辆前进模型

Takayasu 等人在传统的 CA 模型的基础上加入了慢启动原则，发展出了 TT 模型，该模型是 NaSch 和跟驰模型的联结，由此来更加真实地模拟交通流的运行情况，因为在这当中加入了车辆慢启动规则。具体的更新步骤如下：

1）加速

驾驶员总是以期望的最大速度行驶，即$v_n \to \min(v_n + 1, v_{max})$。

2）慢启动

车辆停车后有一定概率不会立即启动，即以概率$1 - p_s (0 \le p_s \le 1)$有$0 \longrightarrow v_1$。

3）减速

与 NaSch 模型类似，车辆的速度以$v_n \to \min(v_n, d_n - 1)$。

4）随机慢化

车辆会以一定的概率p减速$v_n \to \max(v_n - 1, 0)$，即模拟交通运行中各种不确定的意外情况。

5）位置更新

经过上述步骤后，更新车辆的显性的运行状态，以更新数据的方式进行，即$x_n \to x_n + v_n$。

（3）车辆换道模型

为了描述并模拟车辆在实际道路网中的换道情况，有必要对换道的规则进行清晰的描述。Chowdhury[31]曾经提出过支持对称换道原则的双车道元胞自动机模型，该模型引入了两个变量，即安全条件和换道动机，也就是说在换道情况发生前，通常要判断这两个条件是否符合。对于换道动机，指的是当前行驶车道的速度不能满足驾驶员所期望的速度或者另一车道的车流情况优于当前行驶车道。

对于路网交通，需要对 Chowdhury 的模型做一些改进。具体就是关于转弯情况的判断，如果下一个路口需要转弯则换道到可转弯的车道；若车辆已经在可转弯的车道了，但是若司机想在下一个路口保持直行，那么司机也会进行换道。因此基于交通路网的换道模型具体如下：

1）换道动机：$d_n < \min(v_n + 1, v_{\max})$和$d_{n,\text{other}} > d_n$，$lane_n \neq lane_{obj}$。

2）$d_{n,back} > d_{safe}$。

v_n表示第n辆车的速度；

$d_{safe} = v_{max} + 1$表示换道过程中的不会发生碰撞的临界安全距离；

d_n表示第n辆车和该车前车之间空的单元数；

$d_{n,other}$表示第n辆车与它想换车道的前车之间的空单元数；

$d_{n,back}$表示第n辆车与它想换车道的后车之间的空单元数；

$lane_n$表示第n辆车行驶的车道；

$lane_{obj}$表示第n辆车的目标换道车道。

除此之外在满足安全距离以及安全条件之下，也会以概率$p_{\text{change}}(0 \le p_{\text{change}} \le 1)$进行换道，

这样可以更加真实地模拟实际的交通路网情况。

2.2.2 交叉口的元胞自动机模型

在城市路网的交通元胞自动机模型中，必不可少的内容就是交叉口的的研究。交通路网由车辆以及交通基础设施组成。交通基础设施囊括了道路模块以及信号控制模块，需要定义特定的信号灯控制规则以及车辆在交叉路的转规则。

对于十字形道路交叉口的元胞自动机模型如图 2.5 所示。

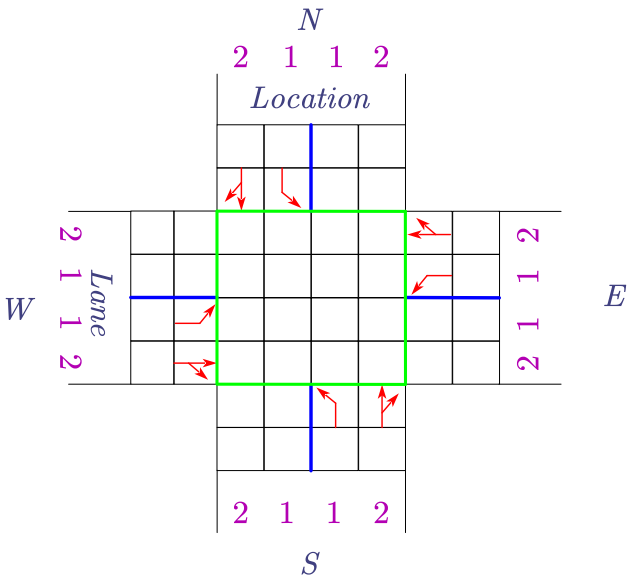


图 2.5 双车道的十字道路交叉口模型图

（1）信号灯控制规则

使用定时控制，也就是交通信号灯按照一定时间进行周期循环对交通流进行周而复始的控制。其中具体的四相位信号控制情况如图 2.6 所示。

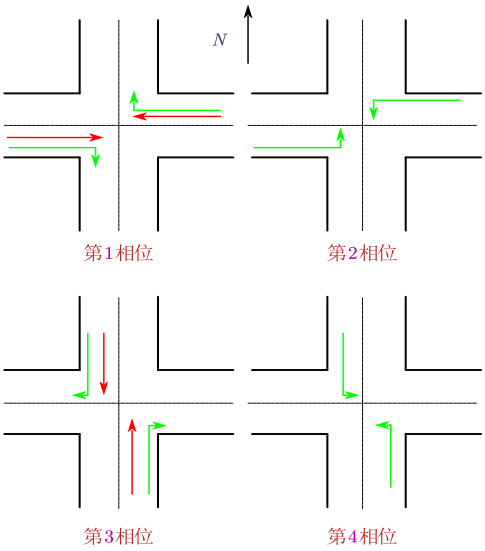


图 2.6 四相位信号控制交通流图

（2）车辆左转模型

在交叉口的转向需要清晰的规则，因为在实际的交通路网中往往会发生很复杂的运行情况，所以有必要对左转的情况作出特别的处理。

1）减速，$v_n \to \min(v_n, d_n)$，在车辆行至交叉路口时，信号灯一般状态是不允许车辆继续通行，车辆必须以较低速度通行，在既定的规则下，需要左转的车辆需要转换到相应的左转车道，进而等待红绿灯的变换，在停车线处等待直至可通行。

2）左转，$x_n \to x_n \pm 1, y_n \to y_n \pm 1$，当许可车辆通行时，车辆开始左转，车辆位置的更新规则为位置变换到左前方的邻近元胞，并设置左转速度为一个单元胞时步。

3）位置更新，经过上述步骤实际更新所有路网数值。

v_n 表示第 n 辆准备左转的车的速度；

x_n 以及 y_n 表示车辆在模拟路网中的横坐标与纵坐标；

d_n 代表车辆与当前前车之间的距离。

（3）车辆右转模型

在已经分流交叉口的转向需要清晰的规则，由于在一般的实际路网中右转一般不会特别设计信号灯，在满足安全距离的情形下，车辆即可进行右转。

1）减速，$v_n \to \min(v_n, d_n)$，在车辆行至交叉路口时，信号灯一般状态是不允许车辆继续通行，车辆必须以较低速度通行，在既定的规则下，需要右转的车辆需要转换到相应的右转车道，进而等待红绿灯的变换，在停车线处等待直至可通行。

2）左转，$x_n \to x_n \pm 1, y_n \to y_n \pm 1$，当许可车辆通行时，车辆开始右转，车辆位置的更新规则为位置变换到右前方的邻近元胞，并设置右转速度为一个单元胞时步。

3）位置更新，经过上述步骤实际更新所有路网数值。

v_n 表示第 n 辆准备右转的车的速度；

x_n 以及 y_n 表示车辆在模拟路网中的横坐标与纵坐标；

d_n 代表车辆与当前前车之间的距离。

2.3 出行路径搜索

若要进行最优的动态交通分配，则需要基于瞬时交通阻抗，列举各 OD 对交通流量的所有出行路径，从而利用相关算法得到最优路径。

（1）路径及其费用

在基于元胞自动机的动态网络分配模型中，一条路径上的所有费用等同于所有路段上元胞所组成的费用之和。路段费用的组成多种多样，如路段的长度、舒适度、通行费、行程时间等指标。在这里可选用出行时间作为主要的衡量因素来对路段的费用进行计算。其数学表达式为式(2.5)，(2.6)：

$$P_a = t_a \tag{2.5}$$

$$P_r = \sum_{a \in r} P_a \tag{2.6}$$

其中，P_a——路段a的出行费用；

t_a——路段a的行驶时间；

P_r——路径r的出行费用；

a——属于路径r的一条路段。

（2）路径选择

从驾驶员角度出发，一次出行的路径选择是多种多样的，并且随着时间的推移又会不断地发生变化。一般情况下，驾驶员虽然主观意愿上想选择最优出行路径，但往往却并不是系统最优出行路径，相对最优路径的选择概率缺是最高的，故为了评价所选择的路径的满意程度，则引入了路径的效用函数，定义为式(2.7)：

$$U_r = \frac{1}{P_r} \tag{2.7}$$

式中，U_r——路径r的效用值。

该路径选择模型可以用 Logic 方法进行建模处理，可以在数学上求出最优解。Logic 方法的函数表达式为：

$$L(R_j) = \frac{e^{\mu U_j}}{\sum_i e^{\mu U_i}} \tag{2.8}$$

式中，U_j——路径j的效用值；

$L(R_j)$——路径j被选择的概率；

μ——Logic 的敏感系数。

驾驶员选择不同效用路径的概率分布由敏感系数所决定。敏感系数反映的是驾驶员选

择选择路径与最优路径之间的距离，直观地说就是驾驶员是否能感知到自身所选择的路径就是最优路径。若驾驶员选择不同路径的概率相差无几，则敏感系数低；若几乎所有的驾驶员都选择最优路径，则敏感系数较高。

（3）路径搜索

多重路径的动态网络分配需要搜索并存储起讫点之间所有的出行路径，当路网规模庞大时，这种算法所带来的时空消耗往往是无法接受的。为降低算法的时间复杂度，依然需要使用空间换时间的方法，但并非是在工作站中存储所有的出行 OD 对之间路径信息，而是仅仅存储一次迭代运行过程中产生且被使用的最短路径。由于动态网络分配方法的时变性，不同时间点同一个 OD 对之间的最短路径可能是不同的，故将这些路径一并存储以供后来使用。

在每个评价时刻的开端启动路径搜索模块，上一次迭代过程同一个评价时间间隔测算出来的行驶信息为搜索依据，最后将上文所阐述的路径费用作为路径选择的标准。在每次搜索的过程中避免较差路段始终无法被选到的做法是将其行驶时间赋值较小的数，这样使得最终产生的结果具有很好的鲁棒性。

2.4 双目标网络流分配方法

基于驾驶员的主观判断所期望走的最短路径往往并不是系统所决定的最短路径，在交通出行的最初可能是比较顺畅的，但后续的道路状况却无法预估。故本模型需要基于驾驶员行车时间最短以及交通总延误最小的要求来进行交通分配。这样可以在系统最优与用户最优之间达到一个平衡，同时也是一个双目标规划问题。

2.4.1 路段阻抗

路段阻抗指的是车辆经过该路段所需要的时间，涉及的因素包括路段的长度，路段的车流量。一般理想条件下，出行时间与出行距离会成正比例关系，但在实际的交通网络上，由于车流的停滞以及信号控制的不合理，车辆常常无法以自由状态行驶。本文使用的路段阻抗函数如式(2.9)：

$$t_a(w) = t_a(0)\left[1+\alpha\left(\frac{q_a}{e_a}\right)^{\beta}\right] \tag{2.9}$$

式中，t_a——表示路段a上的阻抗，即出行时间；

$t_a(0)$——表示路段a上车辆自由行驶的时间；

q_a——表示路段a上的交通流量；

e_a——表示路段a的通行能力；

α,β——待标定的参数，一般分别取 0.15 和 0.4。

2.4.2 交叉口阻抗

交叉口的阻抗一般来自交叉口设施以及车流的阻力作用，如信号配时，车道转弯限制，车流密集等。一般可用车辆的出行延误来对交叉口阻抗进行衡量，应用广泛的 Webster 公式如式(2.10)：

$$\bar{y}_i = \frac{T_c\left(1-\frac{g_i}{T_c}\right)^2}{2\left(1-\frac{g_i}{T_c}x_{ij}\right)} + \frac{x_{ij}^2}{2q_{ij}(1-x_{ij})} - 0.65\left(\frac{T_c}{q_{ij}^2}\right)^{\frac{1}{3}} x_{ij}^{(2+5x_{ij})} \tag{2.10}$$

式中，$\bar{y}_i$——表示i相位车辆的平均延误，即时间阻抗；

g_i——第i相位有效绿灯时间；

T_c——表示信号周期的时长；

q_{ij}——表示第i个相位第j个进口道实际到达的交通流量；

x_{ij}——表示第i个相位第j个进口道的路段饱和度。

Webster 公式一共分为三部分，第一部分描述了信号控制的基本延误，第二部分引入了随机延误来进行泛化，第三部分进行了修正处理，以使得结果更加合理，对于交通信号控制的延误做了精确的数学描述。

2.4.3 目标函数

第一个目标函数为每个道路路段的阻抗最小，第二个目标函数为固定相位信号周期内，每个交叉口的交通延误最低，其中阻抗的计算由式(2.9)以及式(2.10)决定。

目标函数Z_1：

$$\begin{aligned}\min Z_1 &= \sum_{i=1}^{n} \min z[x_a(i)] \\ &= \sum_{i=1}^{n}\left[\sum_a \int_{e_a(i-1)}^{e_a(i)} t_a(\omega)d\omega + \sum_a \int_{e_b(i-1)}^{e_b(i)} t_a(\omega)d\omega\right]\end{aligned} \tag{2.11}$$

约束条件：

$$s.t.\begin{cases}\sum_{p}\int_{p}^{rs}(i)=q_{rs}(i)+\sum_{a\in B(r)}o_a^s(i)\\ x_a(i)\le q_{a,\max}\\ f_p^{rs}(i)\ge 0\\ x_a(i)=\sum_{r}\sum_{s}\sum_{p}f_p^{rs}(i)\delta_{a,k}^{vs}\\ e_a(i)=e_a(i-1)+x_a(i)-o_a(i)\\ o_a(i)=\sum_{r}o_a^s(i)\\ O_a(i)=x_a(i)\\ i=i+t_a[e_a(i)]\end{cases}$$

式中，Z_1——时段$[0,T]$的网络总阻抗；

$z[x_a(i)]$——为时段i时网络总阻抗；

$x_a(i)$——时段i时在路段a上的交通流量；

t_a——路段a的阻抗；

$f_p^{rs}(i)$——时段i时 OD 对r到s之间路径p的交通流量；

$\delta_{a,k}^{vs}$——路段a在r到s之间路径p上，则$\delta_{a,k}^{vs}=1$，否则得 0；

$q_{a,max}$——路段a交通容量上限；

$e_a(i-1),o_a(i)$——分别为各个时段路段a上交通进入与离开量。

目标函数Z_2：

$$\min Z_2=\min\left(\frac{\sum_{i}\sum_{j}\left[\alpha q_{ij}\bar{y}_{ij}\right]}{\sum_{i}\sum_{j}q_{ij}}\right) \tag{2.12}$$

约束条件：

$$s.t.\begin{cases}\sum_{i}g_i+t_{\mathrm{TL}}=T_c\\ g_{i,\min}\le g_i\le g_{i,\max}\\ T_{c\min}\le T_c\le T_{c\max}\end{cases}$$

式中，t_{TL}——交叉口信号周期内的总损失时间；

$g_{i,\min},g_{i,\max}$——分别为相位i时的最小绿灯时间和最大绿灯时间；

$T_{c\min}, T_{c\max}$——分别为最小信号周期长度以及最大信号周期长度。

2.5 本章小结

介绍了经典的道路交通流元胞自动机模型。提出了模拟交通路网的车道元胞自动机以及交叉口元胞自动机模型，并且阐明了路径搜索算法以及相应的双目标函数。从最基础的NaSch 模型出发，进行思考，问题从一维变为二维，加入了交叉口模型以及转向模型，形成了一套自适应的交通网模拟系统，从而根据其中的数据以及当前所研究的各种动态网络分配模型定义了该系统的双目标函数。

第三章 基于遗传算法的动态网络分配的求解

3.1 遗传算法概述

研究生物进化论受到启发，美国密歇根大学 J. Holland[32]教授在 1975 年创造性地提出遗传算法（Genetic Algorithm，GA）。遗传算法演化于自然界的“物竞天择，适者生存”的发展规律，在种群的不断迭代过程中具有相应的智能。对于适应度函数能产生优值的子代会被保留加以繁殖，而对于适应度函数产生劣值的子代会被渐渐地抛弃。在解空间的不断搜索的过程就是自然界的各个种群不断繁殖发展的过程，自然界具有自适应的特点，无论多大的灾害，自然界会不断地去修正这种巨大的影响，正如遗传算法拥有的这种特质，该算法具有的智能可以在每次迭代中不断适应，不断进化。但有时初始设定的变异概率，交叉概率以及初始种群过低时会导致解空间的搜索结果会是局部最优解，也就是该解在整体来看不是我们所期望得到的最优解。

遗传算法的基本步骤为：

① 设置初始种群，每个染色体由N个基因组成，对各个染色体的适应度进行评价；

② 判断 GA 收敛准则是否满足，若满足则搜索结束，输出答案；否则，继续执行；

③ 对得到的适应函数值，进行择优选取，对选定的染色体进行复制操作；

④ 染色体以概率P_c进行交叉操作；

⑤ 染色体以概率P_m进行变异操作；

⑥ 返回步骤②并进行判断。

在上述的计算操作中，初始种群的设置要合理，一般可以根据目标函数的复杂性设置一个比较高的初始种群值，这样可以在求解时扩大解空间的搜索范围。除此之外可以防止得到的解过快收敛，并且一般不是实际问题可接受的解。收敛标准的确定也是需要相关的研究来确定，收敛标准设置得适当可以将计算的时间显著降低，节省计算设备的时间与空间损耗，但是若是不恰当的收敛标准会大大地提高计算的时间，占据大量的计算设备的时间与空间损耗。此外，变异概率以及交叉概率也需要设置得合理一些，因为在实际的自然界演化过程中，变异是小概率事件，不会经常发生，一旦这种事情变得常态化会导致不可预估的可怕后果。交叉概率也是固定的时间段产生的，动物种群不可能永远都在交配，只会在某个固定的时间段进行相应的交配工作。为了真正地保留每次繁殖的最优种群，就必须真实地模仿大自然的演化规律。因为无数次实验已经证明，大自然具有某种超越人类的

智慧，可以在无数种可能中给出复杂问题的自适应的最优解。

采用遗传算法求解相关问题的优势如下：

1）遗传算法对染色体的处理方式是通过编码来处理的，可以比较细致的模拟染色体各个环节的变化情况，并且可以直观地展现染色体的实时变化情况；

2）遗传算法的计算方式是对多个初始种群进行并行计算，所以在解空间的计算时间上以及在空间复杂度上都优于其他类似的智能搜索算法；

3）遗传算法完美地模拟了自然界的随机性，在随机选择种群进行演化发展的过程中，也同时保证了每次迭代的种群多样性以及种群的优越性，这样在求解的过程中也可以防止快速收敛的弊端；

4）遗传算法对于非线性规划问题具有天然的优势，原因在于非线性规划是在数学上是没有通用的解法的，意味着它的随机性很强，而对于这样随机性的问题，带有智能的算法则具有更加好的优势。

遗传算法虽然具有很多优点，但是对于最优解的求解很大程度上取决于初始种群的设置。因为在每次种群的迭代过程中，总是会有不可预料的情况发生，所以会导致每次运行的结果都不一样。这种性能差的表现就在于遗传算法的固有缺陷。如何得出最优的初始种群在遗传算法的应用中至关重要，所有在遗传算法提出来的这么多年来，已经产生了各种类型的改进版遗传算法，例如 NSGA—II。

3.2 NSGA—II 算法求解流程

对于多目标函数求最优解问题，传统的优化方法（Multi-criterion decision-making methods）提出将多目标优化问题转换为单目标优化问题，强调一次仿真运行只获得一个最优解，然后通过多次仿真希望获得多个不同的最优解。NSGA-II 是最受欢迎的多目标优化算法之一，具有三个特征，快速非支配排序方法，快速拥挤距离估计程序和简单拥挤的比较算子。

NSGA—II（Non-dominated sorting and sharing Genetic Algorithm）改进主要是针对如上所述的三个方面：

① 提出了非支配排序的概念，传统的遗传算法只是简单的采用轮盘赌的形式来进行种群的选取，在 NSGA—II 算法中对种群进行排序后，在数学意义上可以严格证明该方法可以显著提高最优解的求解效率；

② 解决了在多目标求解问题中设置权重的问题，传统的多目标函数的最优目标函数值求解过程中需要人为地设定权重，那么求解的结果就具有非常主观的特点，而 NSGA—II 则是自主的自适应地解决相应的问题；

③ 采用拥挤度的策略不仅可以防止解的快速收敛，还可以增加种群的多样性，若某处群落过于聚集，则需要对其进行干预处理，打乱其排布方式，对于 Pareto 域的求解以及 Pareto 域的全面性平滑性具有非常好的适应度。

在 NSGA 中，通过克服问题中现有目标的其他响应来评估和组织响应。每个响应的适用性是根据其对其他响应的成功数量来考虑的。为了在搜索空间中分发响应，使用用于紧密响应的适应度共享方法。然而，鉴于该算法对适应度共享以及其他参数的高灵敏度，后来以 NSGA—II 的形式提出了该算法的新版本。在这种新算法中，使用密度距离而不是适应度共享，在该算法中，锦标赛选择运算符也用于生成新世代。

多目标的函数求解问题，与单目标函数求最优解的差别很大。虽然看似问题从一元变成二元或者多元，问题的复杂度的上升却是指数级别的，在以往这种问题可能是 NP—Hard 问题，不过在合理的情况下还是可以得到比较符合实际的最优解。一般若是对实际问题有了足够的研究就可以在完全理解问题的基础上设置除相应的人为设定好的权重，当然缺点就是主观性太强，不适用于大部分需要自适应的多元目标函数的求解。大部分情况下，不可以为了某一维的函数而放弃其他维度的目标函数，所以就需要找到一个曲线甚至是曲面来平衡这种多元目标函数的解空间。这种解称作非支配解(Nondominated Solutions)或 Pareto 最优解(Pareto Optimal Solutions)。

NSGA—II 算法的基本思想为：初始时，随机产生规模为 N 的初始种群，根据问题范围和约束初始化总体，非支配排序后通过遗传算法的选择、交叉、变异三个基本操作得到第一代子代种群；从第二代开始，将已初始化的群体按照非支配标准的进行分类，进行快速非支配排序，排序完成后，拥挤距离值正面分配，根据等级和拥挤距离选择种群中的个体，同时计算每个非支配层中的个体的拥挤度，选取合适的个体组成新的父代种群，计算方式以非支配关系以及个体的拥挤度为根据；使用具有拥挤比较运算符的二元锦标赛选择来执行个体的选择；使用模拟二进制交叉和多项式变异的实数编码 GA；将后代种群和当代世代种群组合，并通过选择来设定下一代的个体，随后每个阵营填补新一代，直到种群规模超过当前种群规模。相应的程序流程图如图 3.1 所示。

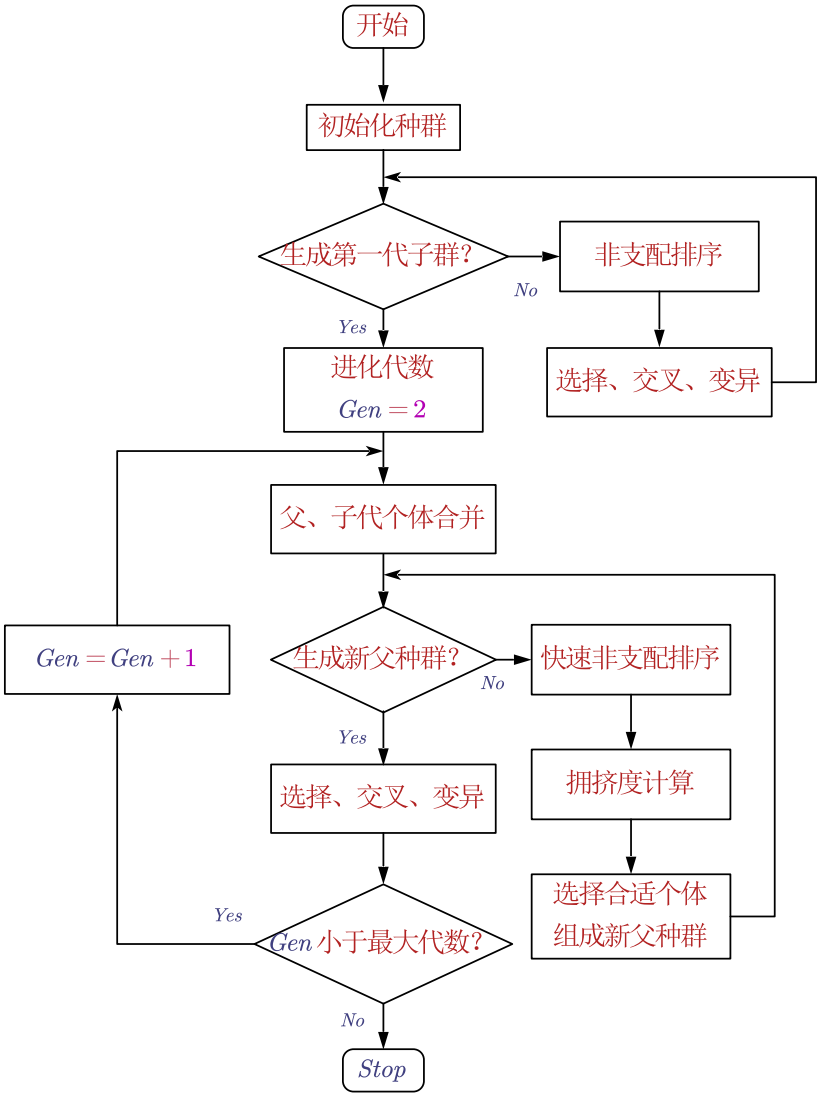


图 3.1 NSGA——II 算法流程图

假设双目标函数的解集种群为Q，目标是需要计算Q中的代表每个解的个体q中的参数n_q和S_q。在这里，n_q代表的是解集种群中支配个体数q的个数，S_q为解集种群中被个体q支配的个体数集合。

对于求解基于元胞自动机的动态网络分配模型的双目标函数的 NSGA—II 算法关键步骤为：

① 首先搜索种群中所有$n_q=0$的个体，并存储在集合F_i中；

② S_i为所支配个体的集合，对于其中的个体l，做$n_l=n_l-1$的操作，若$n_l=0$，将该个体存储在集合H中；

③ 令存储在F_i中的个体为初始层中非支配的个体，同时将H标记为初始集合，循环上述步骤直至所有种群得到明确的分级。

则该非支配排序遗传算法的伪代码可见图 3.2。

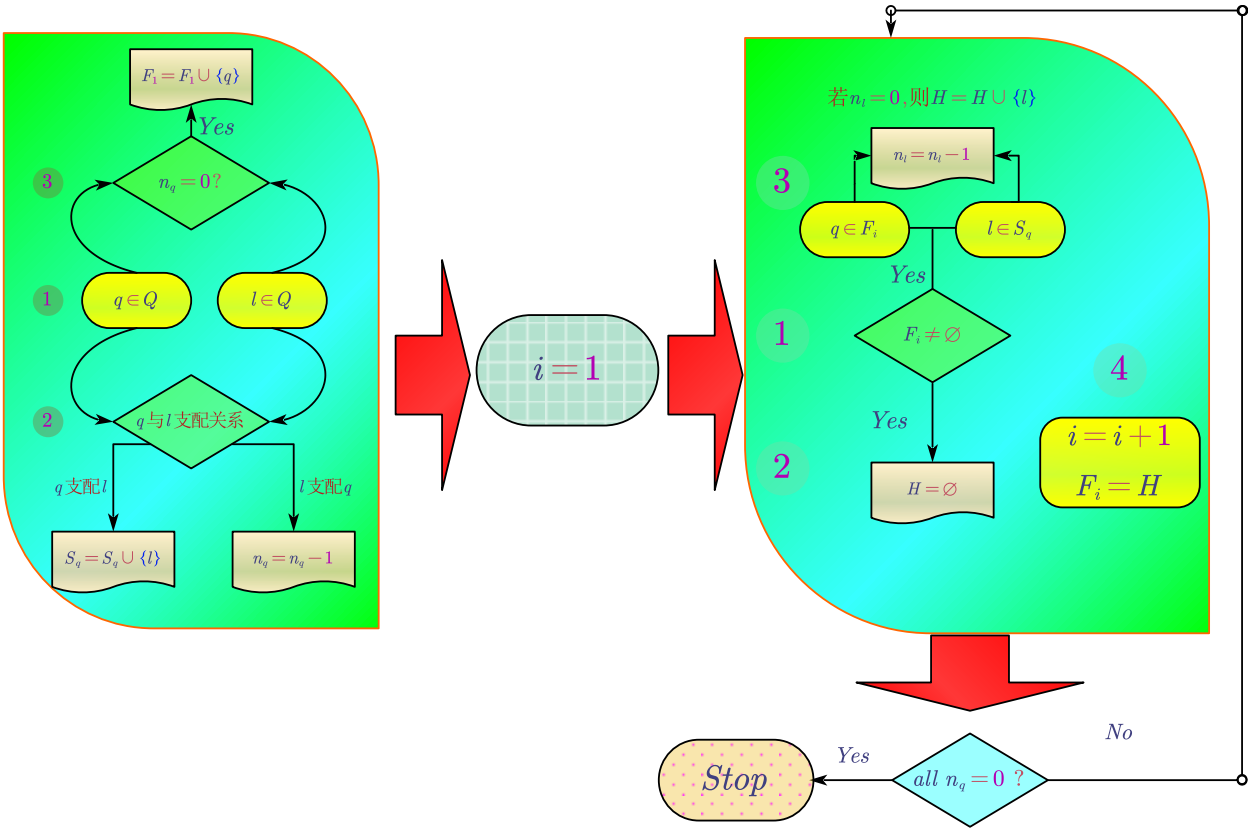


图 3.2 非支配排序遗传算法的伪代码

3.3 本章小结

介绍了遗传算法的基本原理，进而引入 NSGA—II 算法来对上文所述的目标函数进行求解，简要阐明了求解的原理，以及详细描述了改进后的求解算法步骤。

第四章 基于元胞自动机的城市动态网络分配的数值仿真实验

4.1 仿真数值实验体系结构

对于前文建立的模型进行仿真实验需要设计一套完备的仿真体系，城市交通路网系统的仿真模拟是用来描述复杂的实际交通流运行情况。根据不同的要求，需要实现不同的模块，图 4.1 则是展现了将要实现的仿真系统。

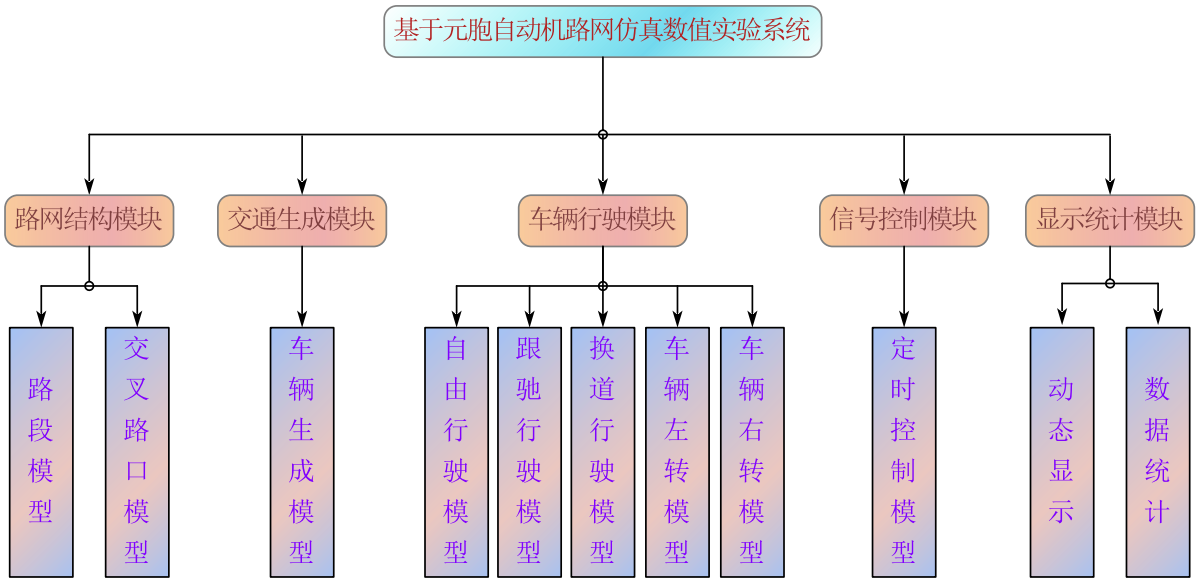


图 4.1 基于元胞自动机路网仿真数值实验系统结构图

基于元胞自动机的动态网络分配仿真模型主要由五个模块组成，包括车辆行驶模块、交通生成模块、信号控制模块、路网结构模块、显示统计模块。

城市路网主要由直行路段与十字交叉路口组成，采用周期性边界条件来生成车辆，并设置车辆生成的概率。

① 自由行驶模块主要设置的是随机慢化概率；

② 跟驰模型的设置是为了更加真实地反映交通流组织方式，具体设置根据式(2.1-4)。

③ 车辆换道是为了多车道模型进行的匹配设置，其换道概率随着系统并行变化，具体的设置方式由第三章的换道模型决定。

④ 交叉口模块的设置除了对交通流进行约束外，也为式(2.5-8)提供了输入集。

⑤ 数据统计部分会产生车辆位置，速度，交通流密度等信息，为数据分析，得出结论做准备。

在整个元胞自动机仿真路网的基础上，结合遗传算法才是完整的动态网络分配模型，

具体的算法流程如图 4.2 所示：

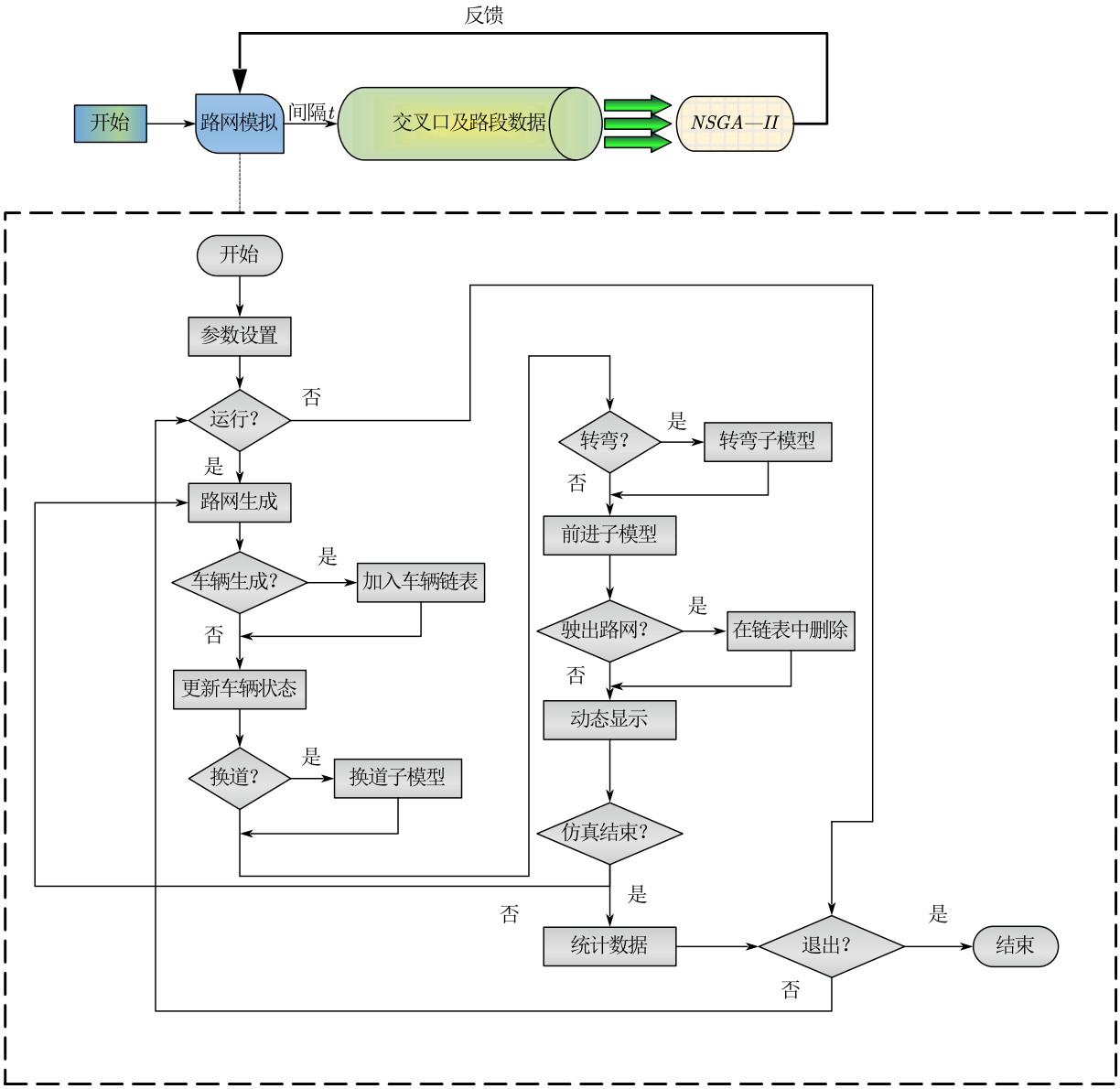


图 4.2 基于元胞自动机模拟的动态网络分配算法流程图

4.2 数值实验

4.2.1 路网实现与设定

基于上文所描述的模型，设计了如图 4.3 的路网图来进行数值实验，进而对模型进行数值分析。其中共有 32 个路段，14 个交叉路口，设置了 12 个入口。路网长度为 120 个元胞格子，宽度为 90 个元胞格子，中间设置了隔离带，模型的时间没有量纲，元胞车辆的最大速度为 5。可见路网的精细程度与复杂程度已经达到了可以验证模型的有效性以及可操作性的标准。

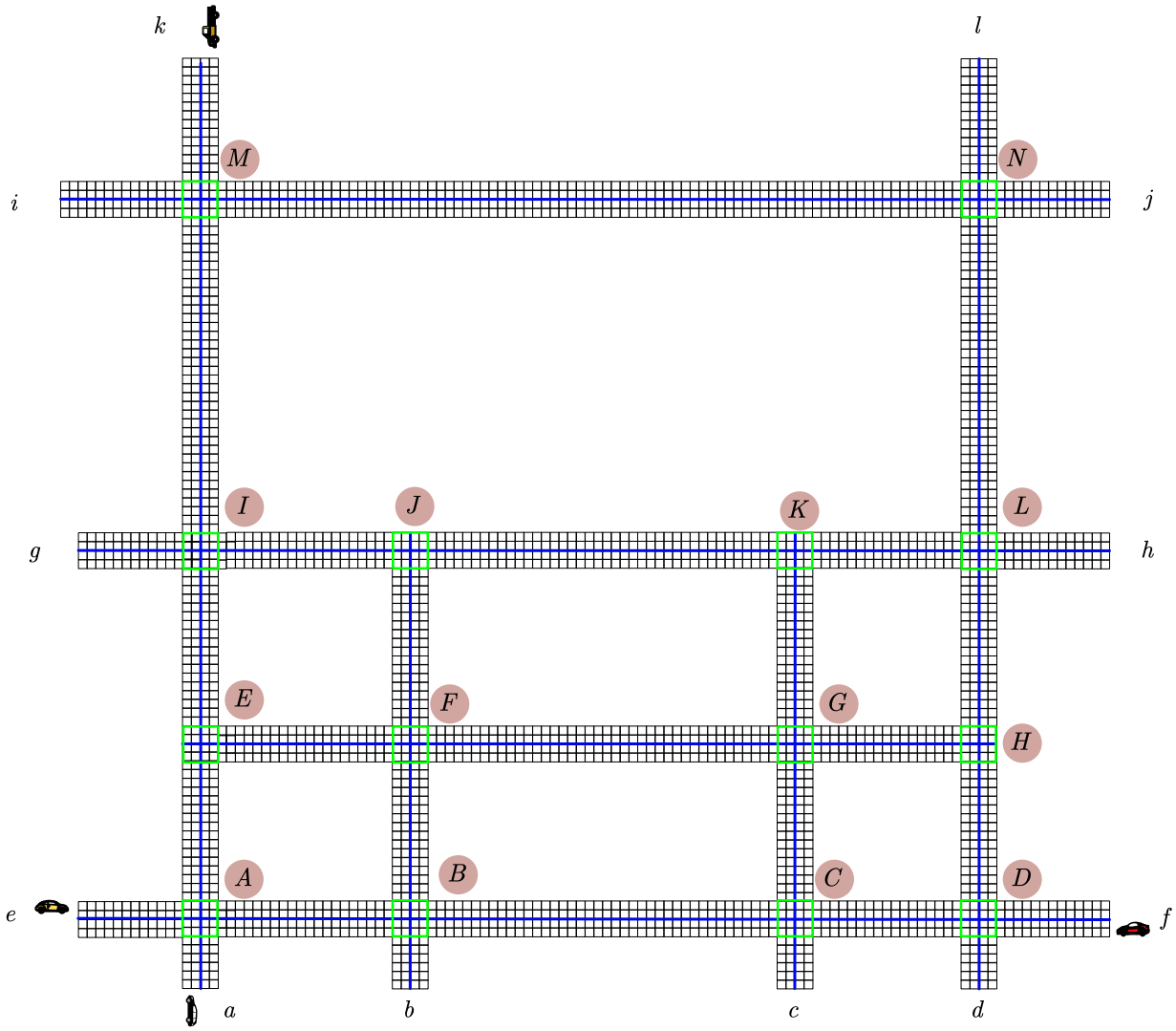


图 4.3 数值实验交通网络模型图

4.2.2 NSGA—II 的设定

对于 NSGA—II 算法进行了一些初始设定：

① 基于实数编码，将道路路段的车辆数确定为优化变量；

② 设置种群规模为$M=200$个，算法执行的最大迭代次数为$G=200$次，交叉率为$P_c=0.85$，变异率为$P_m=0.015$。

此外目标函数以及约束条件都囊括在式(2.11)与(2.12)中，算法会在帕累托域中初始化道路网上各个路段的车辆数种群，进而判别每种模式对应的目标函数值，选择较优的一代进行接下来的竞标赛竞争，然后选择、交叉、变异，最终得到最优解，在最优解中选择符合实际的解作为动态分配的实际操作方案。

4.2.3 交叉口信号配时的设定

设置周期时长为 20，其中第 1 相位为 1~5，第 2 相位为 5~10，第 3 相位为 10~15，第

4 相位为 15~20。

4.2.4 车流初始化设置

路网初始状态无车行驶，设置初始有车率为 0。假设车辆从路段gI以进入概率$d_p=0.8$进入路网，设从g进口道进入路网的车辆的目的地为h；车辆从路段lN以进入概率$d_p=0.8$进入路网，从l进口道进入路网的车辆的目的地为a。车辆在一开始进入路网时便开始进行出行路径的选择。

4.3 实验结果分析

应用 MATLAB 编程将上述模型以及输入整合，生成运算结果(见附录一二)。首先可以得到基于元胞自动机模拟路网运行的一些结果。图 4.4 为数值仿真实验路网的三维时空图，图 4.5 为所模拟路网的车流密度与速度关于时间变化的三维插值曲面图。图 4.4 的绘制过程是将每一次迭代的路网横纵坐标随着时间 Z 轴不断叠加起来，每一层的绘制都使用了随机上色用于区分，可以看出在交叉口部分的元胞点数量是非常密集的，在整个仿真过程的分布量是很规律的，虽然随着时间的推移也会发生一些自适应的现象，如某一部分层会较其他的时间层的点群更加稀薄。对于图 4.6 大体上呈现的规律是：在仿真初始阶段，密度低时车流速度会很大，随着时间的推移，车流密度越来越大，车流速度也逐渐增长，当超过一定的阈值区间时，开始呈现比较明显的反向关系。可以得出该路网的模拟情况基本符合客观规律。此外图 4.7 呈现了在 MATLAB 的 Figure 中运行的路网动态展示情况。

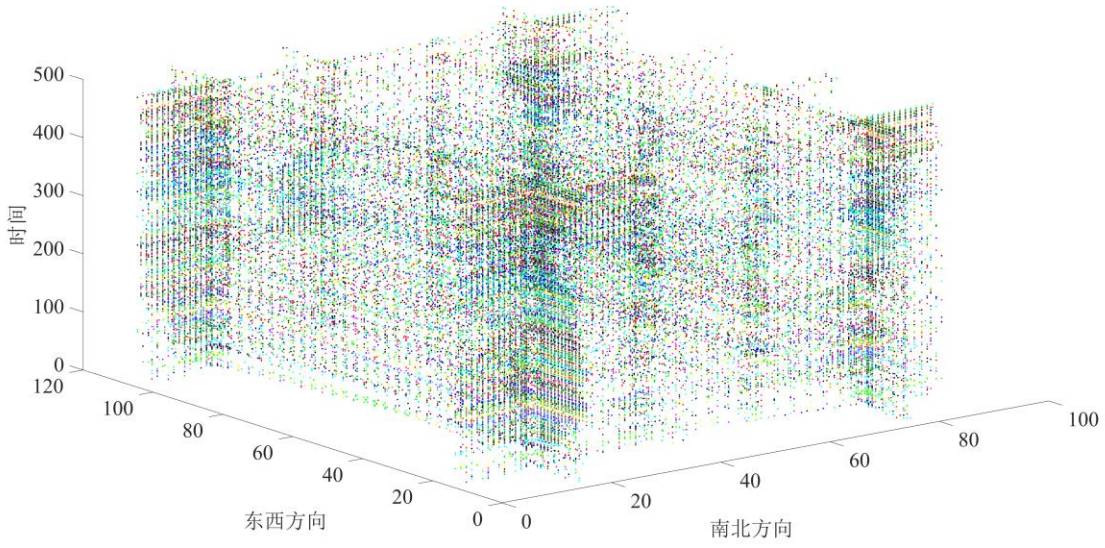


图 4.5 数值仿真实验路网的三维时空图

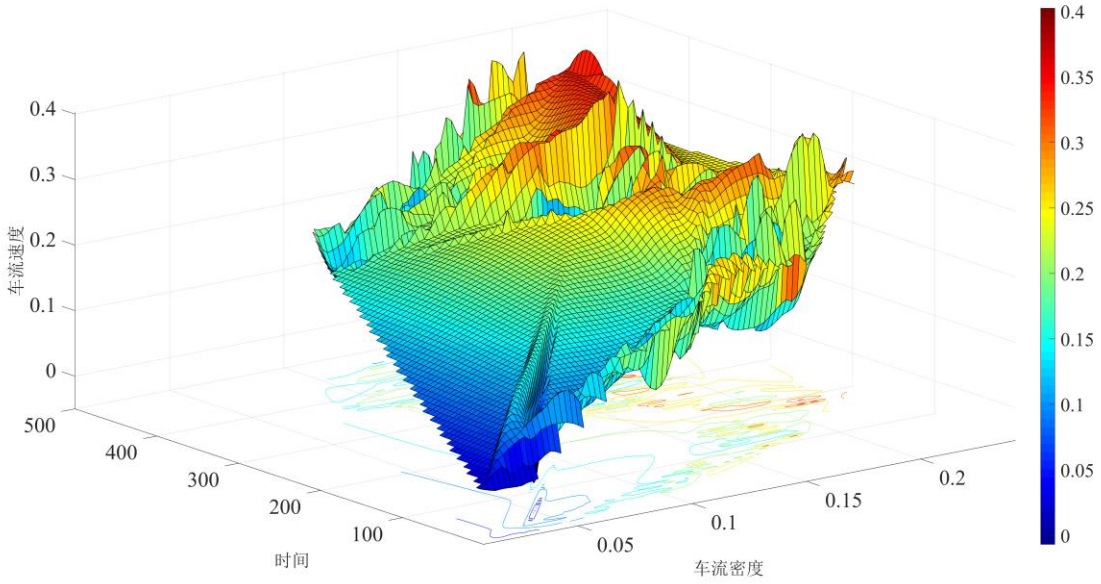


图 4.6 车流密度与速度关于时间变化的三维插值曲面图

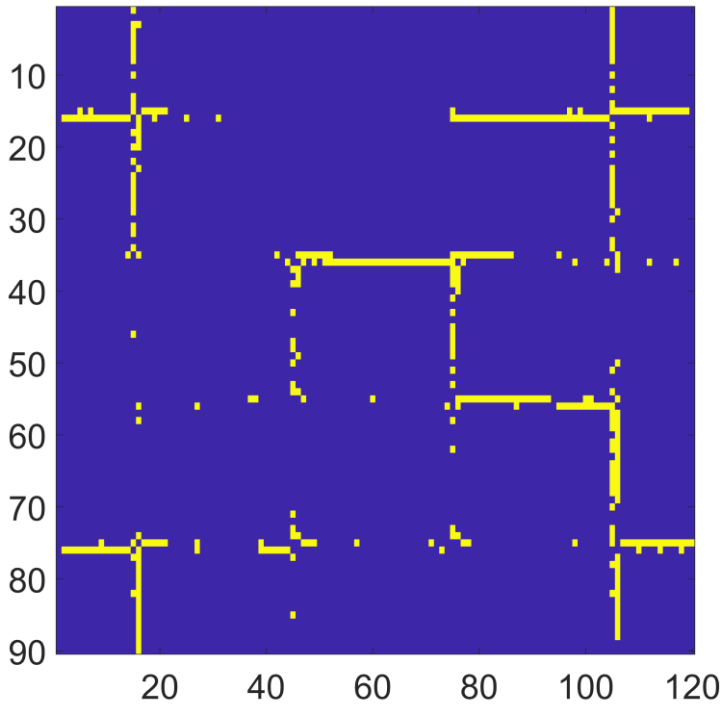


图 4.7 路网动态显示图

程序仿真了 500 个时间步，每隔 10 个时间步调用一次 NSGA—II 算法进行动态分配的计算，随机取其中 6 次 NSGA—II 求得的帕累托域作成曲线图，可得图 4.8。从图中可以看出曲线的边界平滑且清晰，即边界是最优的。所以每次得出的最优解是可以接受的。

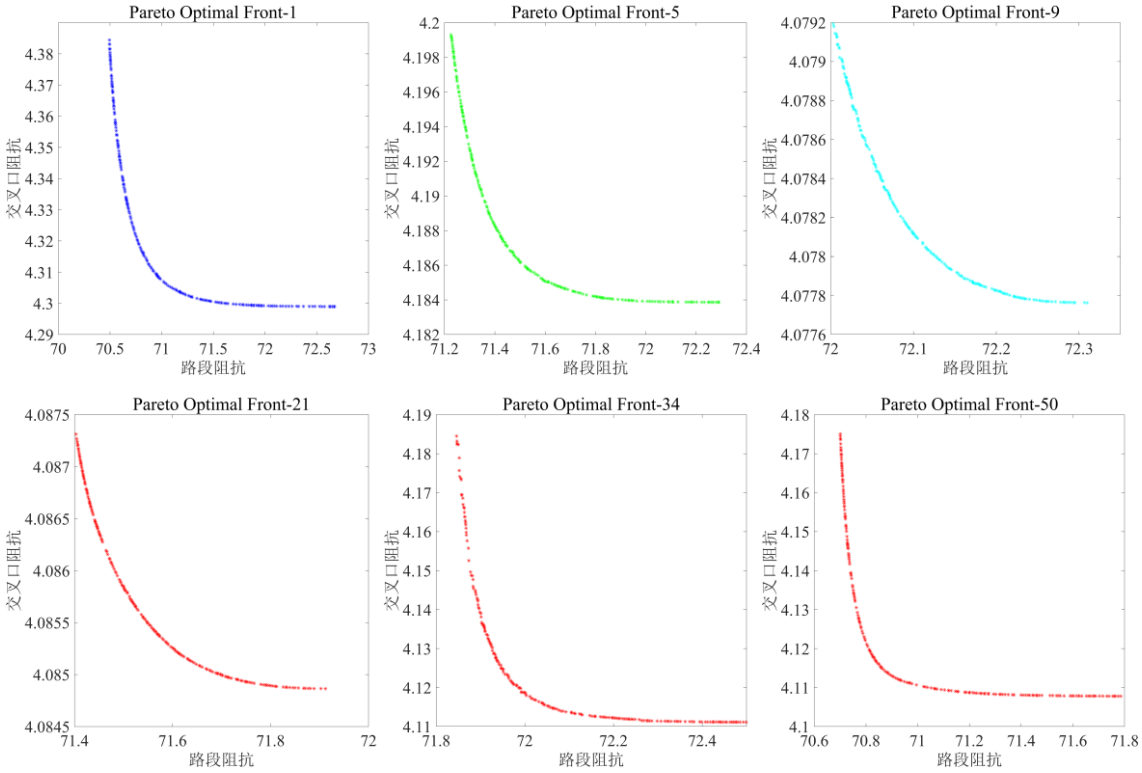


图 4.8 双目标模型 NSGA—II 所求得解的 Pareto 域

由此可得最终的分配结果，现取路段gI的实验数据制成表格，如表 4.1：

表 4.1 路段gI数据

时间	车辆分配 h			瞬时路径阻抗 h			延误 h		
	左	直	右	左	直	右	左	直	右
1	1.509	1.000	1.082	21.708	31.611	21.564	1.282	1.282	21.564
2	72.261	31.616	30.097	22.239	32.493	22.348	1.282	1.282	22.348
3	67.673	15.903	85.205	22.220	32.210	22.898	1.282	1.282	22.898
4	87.659	19.755	25.085	22.298	32.291	22.273	1.282	1.282	22.273
5	80.709	1.027	1.000	22.272	31.615	21.555	1.282	1.282	21.555
6	100.000	2.063	2.054	22.340	31.710	21.646	1.282	1.282	21.646
7	99.618	10.925	17.474	22.339	32.085	22.139	1.282	1.282	22.139
8	99.592	1.767	78.300	22.339	31.686	22.844	1.282	1.282	22.844
9	99.871	4.872	30.101	22.340	31.872	22.348	1.282	1.282	22.348
10	99.742	2.715	54.446	22.339	31.756	22.633	1.282	1.282	22.633

续表 4.1

时间	车辆分配 h			瞬时路径阻抗 h			延误 h		
	左	直	右	左	直	右	左	直	右
11	99.691	1.112	51.219	22.339	31.624	22.600	1.282	1.282	22.600
12	47.548	5.853	16.812	22.127	31.914	22.126	1.282	1.282	22.126
13	1.177	4.209	1.000	21.707	31.840	21.555	1.282	1.282	21.555
14	79.322	1.014	56.930	22.267	31.613	22.657	1.282	1.282	22.657
15	95.049	27.267	3.149	22.323	32.425	21.714	1.282	1.282	21.714
16	99.818	1.326	64.961	22.340	31.646	22.732	1.282	1.282	22.732
17	99.925	1.667	72.640	22.340	31.678	22.798	1.282	1.282	22.798
18	99.961	2.522	51.151	22.340	31.743	22.600	1.282	1.282	22.600
19	98.447	2.014	87.010	22.335	31.706	22.912	1.282	1.282	22.912
20	98.948	3.682	3.570	22.337	31.813	21.736	1.282	1.282	21.736
21	99.506	1.000	1.371	22.338	31.611	21.591	1.282	1.282	21.591
22	11.059	32.101	1.496	21.859	32.500	21.602	1.282	1.282	21.602
23	73.900	2.313	5.365	22.246	31.728	21.816	1.282	1.282	21.816
24	99.997	4.237	2.698	22.340	31.842	21.688	1.282	1.282	21.688
25	99.933	4.786	1.099	22.340	31.868	21.565	1.282	1.282	21.565
26	98.451	1.000	94.609	22.335	31.611	22.967	1.282	1.282	22.967
27	99.364	1.000	88.230	22.338	31.611	22.921	1.282	1.282	22.921
28	99.806	1.060	33.143	22.339	31.618	22.390	1.282	1.282	22.390
29	99.677	1.031	59.744	22.339	31.615	22.684	1.282	1.282	22.684
30	99.825	3.676	1.389	22.340	31.812	21.593	1.282	1.282	21.593
31	96.420	2.478	32.159	22.328	31.740	22.376	1.282	1.282	22.376
32	99.981	1.731	70.574	22.340	31.683	22.781	1.282	1.282	22.781
33	61.576	4.641	2.021	22.194	31.861	21.643	1.282	1.282	21.643
34	1.000	1.000	90.974	21.709	31.611	22.941	1.282	1.282	22.941
35	99.531	1.721	26.785	22.339	31.682	22.299	1.282	1.282	22.299
36	99.628	1.193	70.507	22.339	31.633	22.780	1.282	1.282	22.780

实验数据可以说明每一路段在不同时间段车辆的分配情况，出行路径的瞬时交通阻抗，以及各个车辆在不同交叉口的交通延误时间。除此之外还可以考虑道路几何尺度的限制进一步进行分析。由此从微观车辆的运行情况可以近似地反映宏观车流运行的动态网络分配。

4.4 本章小结

提出了一个二维交通路网，对第三四章所论述的模型，利用 MATLAB 编程进行仿真数值实验。对运行结果得到的图像以及数据表格进行分析模型的可行性以及实用性。

第五章 结　论

动态网络分配理论是研究的智能交通系统的核心理论。目前研究动态分配理论的方法主要是数学推导与计算机模拟，而对于计算机模拟是当前研究的热点。结合已有的数学理论，利用强大的计算设备可以得出研究领域的有益结论。当前时代计算设备的发展日新月异，从多核 CPU 到众核 GPU，再到 Google 的 AI 芯片 TPU，计算设备的发展纵深已经触及了当前人类科技的极限。所以对于动态网络分配的问题，目前在理论层方面需要依托这些领域的发展，研究需要聚焦于横向发展。

基于元胞自动机的交通网络模拟是行之有效的方法，发展的方向就是要不断地将模型复杂化，加入更多的限制，加入更多的变量才能更加近乎真实地再现实际城市交通路网的复杂情况。在动态网络分配理论刚开始出现时，受碍于计算设备的发展，只有从纯理论角度研究该问题。当前时代，各种算法的层出不穷，各种开源平台，各种服务器的集群发展，使得该领域的研究迎来了最好的时代。元胞自动机，遗传算法是很久之前就提出的模型，经过实践的证明，应用于本领域也是非常有用的工具。本文所研究的模型与实现的简单的实验证明，多学科交叉是目前本领域的发展方向。针对于交通网络模拟的问题，本文从最基础的 NaSch 模型出发，进行思考，问题从一维变为二维，加入了交叉口模型以及转向模型，形成了一套自适应的交通网模拟系统，从而根据其中的数据以及当前所研究的各种动态网络分配模型定义了该系统的双目标函数。进而应用 NSGA—II 算法并对其进行对应本文提出系统的改进，求解双目标规划模型。实验结果证明，在对交通网络基于完善规则的模拟的情况下，应用恰当的算法可以对交通网进行一定程度的优化。

对于该领域的研究虽然目前已经有了一些研究成果，但离真正的应用还有很长的路要走。对于交叉口的通行能力的限制，行人非机动车的混合影响，路段与交叉口的瞬时阻抗函数的研究，自适应的信号控制研究还需要后人不断地探索与研究。对于动态网络分破中的路径搜索同样是一个复杂的问题，在一个时变性很强的系统内，路段阻抗总是在动态变化，如何应用算法（如 A*）搜索最短路径也是需要时间去考虑的问题。

交通规划长期以来都是一个重要问题，涉及交通设施（通常是街道，高速公路，自行车道和公共交通线路）的设计和选址以及公路扩建或建设。DTA 模型可以解释交通动态和时变需求。由 DTA 建模的交通流量可以与运动波理论一致，考虑相邻链路之间的交通互动。因此，DTA 模型可以提供更真实的交通流量数据作为更准确的排放估算的输入。此外，时

变需求考虑对于实际中提出不同的减排措施至关重要。进行综合的道路网络设计，将 DTA 模型与环境问题相结合，将是交通研究领域的自然演变。多式联运网络设计的扩展，包括绿色交通和公共交通模式，也是未来的另一个研究方向。除了动态交通信号控制外，还有各种动态交通管理策略，如可变信息标志（VMS）或动态路线信息面板（DRIP），变速限制，动态道路标记和匝道计量，这些都是未来的研究方向。

对于 DTA 建模，可以确定三个未来方向。首先，最近的研究已经证明低离散宏观基本图（MFD）在模拟城市拥堵的复杂动态方面显示了非常有前景的结果，且 MFD 也被纳入 DUO 研究。然而，尚未发现其在系统行程时间或系统行程时间与排放相结合方面与 DSO 的相关度。这些扩展为开发更环保的实时交通控制和需求管理策略提供了有用的评价标准，可以在这个方向上进行富有成效的未来分析研究。其次，现有的面向优化环境目标的 DTA 模型是使用经典确定性交通流模型的确定性设置而开发的。然而，实际上，即使注意到了需求和供应随机性，但在 DTA 中尚未考虑环境目标。因此，考虑的方向是封装现有的交通流模型/网络加载程序或改进现有的交通流模型/网络加载程序以处理供应不确定性并使用现有的随机规划或机会约束技术处理需求随机性，另一个方向是开发建模方法来处理与 MFD 相关的不确定性。第三，虽然交通模拟模型和排放模型之间的整合已成为一个快速发展的研究领域。目前已经确定了微观交通仿真模型的不足，包括集成了用于排放估算的瞬时排放模型。因此，需要通过进一步研究仿真模型的内部机制（如跟车，减速和加速模型）来提高表征交通动态的准确性，以便通过整合研究更准确地预测车辆动态分布。

参考文献

[1] Kucharski R, Gentile G. Simulation of rerouting phenomena in Dynamic Traffic Assignment with the Information Comply Model[J]. Transportation Research Part B: Methodological, 2018, https://doi.org/10.1016/j.trb.2018.12.001.

[2] Zhao C L, Leclercq L. Graphical solution for system optimum dynamic traffic assignment with day-based incentive routing strategies[J]. Transportation Research Part B: Methodological, 2018, 117(A): 87–100.

[3] Friesz T L, Han K. The mathematical foundations of dynamic user equilibrium[J]. Transportation Research Part B: Methodological, 2018, https://doi.org/10.1016/j.trb.2018.08.015.

[4] Wang Y, Szeto W Y, Han K, et al. Dynamic traffic assignment: A review of the methodological advances for environmentally sustainable road transportation applications[J]. Transportation Research Part B: Methodological, 2018, 111: 370-394.

[5] 孙瑜. 城市动态交通流分配模型与算法[D]. 湖南大学，2016.

[6] 俞灏. 动态交通条件下交通诱导与信号控制协同研究[D]. 东南大学，2016.

[7] 费文鹏. 面向突发事件下动态路径诱导的拥堵传播与消散预测模型[D]. 北京交通大学，2017.

[8] 张艳. 多模式下城市交通动态网络分配研究[J]. 科技资讯，2017，15(19)：251-252.

[9] 张传琪，张杨. 动态路网下多车型车辆路径问题研究[J]. 交通运输工程与信息学报，2017，15(02)：112-118.

[10] 李东岳. 基于动态交通需求估计和预测的交通分流模型[D]. 北京建筑大学，2017.

[11] 余婷. 基于实时路况的交通网络耗时最优路径研究[D]. 南京林业大学，2017.

[12] 李潇逸. 基于 CTM 的城市路网动态系统最优分配问题研究[D]. 东南大学，2017.

[13] 王钰. 基于海量轨迹数据的动态交通诱导技术研究[D]. 华南理工大学，2018.

[14] 项俊平. 城市道路交通信号区域均衡控制方法及应用研究[D]. 中国科学技术大，2018.

[15] 邹娟. 基于改进元胞传输模型的交通信号优化控制[D]. 武汉科技大学，2018.

[16] 李杰，杨晓芳，付强. 分析驾驶行为的快速路交通流元胞自动机模型[J]. 物流科技，2018，41(12)：63-67.

[17] 夏运达. 基于元胞自动机模型的道路交通流复杂特性的研究[D]. 湖南大学，2018.

[18] 朱森来. 基于贝叶斯统计的城市路网 O-D 矩阵估计[D]. 东南大学，2017.

[19] 吕磊. 短时交通流预测与线路推荐研究[D]. 山东大学，2016.

[20] 赵小方. 基于博弈论的交通控制和动态网络均衡组合优化研究[D]. 北方工业大学，2016.

[21] 周扬栋. 城市道路短时交通流预测方法研究[D]. 江西理工大学，2018.

[22] 常桃宁. 城市道路交通拥堵扩散机理研究[D]. 西南交通大学，2018.

[23] 李昀轩，贾兴无. 基于驾驶人行为选择的交叉口混合交通流建模与分析[J]. 现代交通技术，2018，15(05)：50-55.

[24] 杨达，周小霞，文成，等. 信号交叉口机动车-电动车交通流建模及仿真[J]. 哈尔滨工业大学学报，2018，50(08)：181-187.

[25] 陈芳，张卫华，丁恒，等. 基于出行者路径选择行为的 VMS 诱导策略研究[J]. 系统工程理论与实践，2018，38(05)：1263-1276.

[26] 刘媛. 基于关键路段流量限制的动态交通分配研究[J]. 交通运输工程与信息学报，2017，15(02)：125-130.

[27] 张春生，刘树东，谭覃. 基于改进遗传算法优化 BP 神经网络的短时交通流量预测方法研究[J]. 天津城建大学学报，2017，23(02)：143-148.

[28] 鲁岳，符锌砂，汪烨，等. 基于元胞自动机的车道缩减区交通仿真软件设计与开发[J]. 公路工程，2017，42(02)：150-153+164.

[29] Cantelmo G, Viti F. Incorporating activity duration and scheduling utility into equilibrium-based Dynamic Traffic Assignment[J]. Transportation Research Part B: Methodological, 2018，https://doi.org/10.1016/j.trb.2018.08.006.

[30] Nagel K, Schreckenberg M. A cellular automaton model for freeway traffic[J]. Journal de Physique I, 1992, 2(12): 2221-2229.

[31] Chowdhury D, Schadschneider A. Self-organization of traffic jams in cities: Effects of stochastic dynamics and signal periods[J]. Physical Review E, 1999, 59(2): 1311-1314.

[32] Holland J H. Adaptation in Natural and Artificial System[M]. Adaptation in natural and artificial systems. MIT Press, 1992.

致　谢

这些年我一直保持清醒的认知，明白我的存在是特别也是普通的。我知道成功不容易但是成就自我更加难能可贵，感谢曹阳老师，从最开始到最终的论文定稿。在四年的学习生涯中，我取得的每一点进步都离不开学院以及各个任课老师对我的指导与帮助，我深深地感到惭愧与内疚，我的努力还不够，我还要继续努力。

同时，感谢我的家人对我的无私奉献和关爱，感谢一直陪伴我的同学们，更重要的是一直围绕在我周围的舍友们，使他们让我的生活不再单调，感谢学院老师们的辛勤教导，感谢南通大学交通与土木工程学院给予我丰富精彩的大学生活。“祈通中西，力求精进”，我会将校训牢记在心，再接再厉，不辜负你们对我的期望。

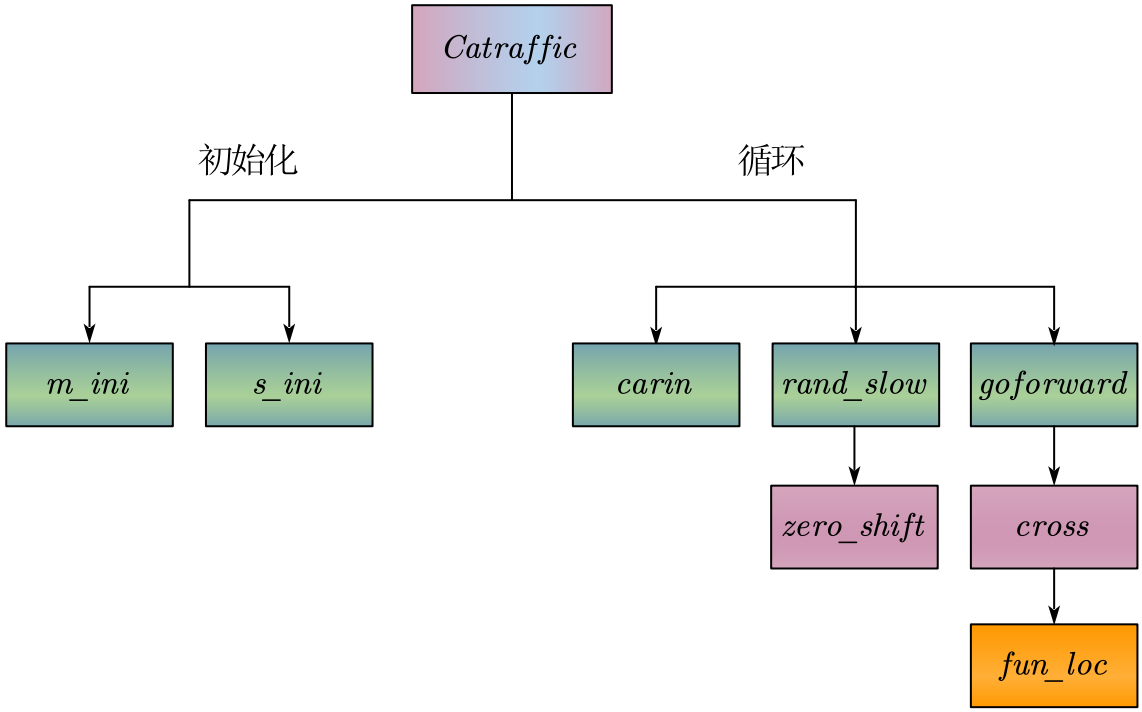
感谢我最挂念的故乡，那里有我最深潜的心流，那里有我最深爱的人，那里有我最初的回忆。史铁生说生命的意义在你是否拷问了，是否描画了，是否沉潜了。我拷问自己，描画未来，沉潜经年，不断追寻。

感谢我的女朋友，她不黏人，不拜金，很文静，善解人意，不苟言笑，最难能可贵的是她根本不存在。感谢她 25 年来没有出现在我的生活里，我才能如此地负重前行。感谢陈晨，感谢她一开始的莞尔一笑，感谢她陪我在漫长的人生中度过了幸福的半个月，感谢她的悄然离去。94 年的我，已然快到了而立之年，有人 25 岁成为了正教授，有人 25 岁已经有了举世瞩目的学术成就，有人 25 岁红遍大江南北，有人 25 岁坐拥百万家产，而我却一事无成。

真正的光明决不是永没有黑暗的时间，只是永不被黑暗所掩蔽罢了。真正的英雄决不是永没有卑下的情操，只是永不被卑下的情操所屈服罢了。

附录一：元胞自动机 MATLAB 程序

- 函数逻辑图：



main.m

```
%% 运行所有程序运行及画图
%% 主脚本程序
clear;clc;

o = 0;
pa = [0.8,0.5,0.3];
T = [5,5];
N = 500;
vmax = 5;
k = [0.05,0.15];
draw = 1;

tic
[JL,JL_in,r,alloc_info]= CAtraffic(o,pa,T,N,vmax,k,draw);
toc
xlswrite('final_data.xlsx',alloc_info,1);
%输出最终的分配以及阻抗延误数据
%% 二维时空分布图
str_data = ['k.';'b.';'c.';'g.';'m.';'r.';'w.';'y.'];
tic
figure(2);
cnt = 1;
set(gcf,'outerposition',get(0,'screensize'));
set(gca,'looseInset',[0 0 0 0]);
for i = 1 : N
    str = str_data(mod(ceil(10 * rand())+1,8)+1 );
for j = 1:90
    for k = 1 : 120
```

```
        if JL{i,1}{1,1}(j,k) > 0
            plot3(j,k,i,[str,'.']);
            hold on;
        end
    if ~mod(cnt,10000)
        clc;
        fprintf('二维时空分布图绘制 %2.2f%%
completed\n',(cnt/(N*90*120))*100);
    end
    cnt = cnt +1;
   end
end
end
set(gca,'FontSize',24,'Fontname', 'Times New Roman');
xlabel('南北方向');
ylabel('东西方向');
zlabel('时间');
set(get(gca,'XLabel'),'Fontsize',24,'Fontname', '宋体');
set(get(gca,'YLabel'),'Fontsize',24,'Fontname', '宋体');
set(get(gca,'ZLabel'),'Fontsize',24,'Fontname', '宋体');
fprintf('开始保存\n');
set(gca,'looseInset',[0 0 0 0]);
print(1,'-dpng','-r300','二维时空分布图');
toc

%% 流量密度时间图
tic
disp('流量密度时间图绘制')
flow = zeros(1,N);
density = zeros(1,N);
for i = 1:N
    flow(i) = sum(JL{i,1}{1,2}(16,:))/120;
    density(i) = sum(JL{i,1}{1,1}(16,:))/120;
end
time = 1:N;
PlotGriddata(density',time',flow');
toc
```

m_ini.m

```
function [m,m_in,r] = m_ini(flag)
m=zeros(90,120);
%水平框架
m(15,:)=2;m(16,:)=1;
m(35,:)=2;m(36,:)=1;
m(55,17:104)=2;m(56,17:104)=1;
m(75,:)=2;m(76,:)=1;
%垂直框架
m(:,15)=3;m(:,16)=4;
m(37:end,45)=3;m(37:end,46)=4;
m(37:end,75)=3;m(37:end,76)=4;
m(:,105)=3;m(:,106)=4;
%主要路口（有红绿灯）
m(15:16,[15,16,105,106])=6;
m(35:36,[15,16,105,106])=6;
m(55:56,[45,46,75,76])=6;
m(75:76,[15,16,45,46,75,76,105,106])=6;
```

```
%主要路口（无红绿灯）
m(35:36,[45,46,75,76])=5;
m(55:56,[15,16,105,106])=5;
%开放
if flag==1
    %水平
    m(26,17:74)=1;m(25,17:74)=2;
    m(42,77:104)=1;m(41,77:104)=2;
    m(48,77:104)=1;m(47,77:104)=2;
    %竖直
    m(17:34,45)=3;m(17:34,46)=4;
    m(17:34,75)=3;m(17:34,76)=4;
    m(37:54,90)=3;m(37:54,91)=4;
    %路口
    m(15:16,[45,46,75,76])=5;
    m(25:26,[15,16,45,46,75,76])=5;
    m(35:36,[90,91])=5;
    m(41:42,[75,76,90,91,105,106])=5;
    m(47:48,[75,76,90,91,105,106])=5;
    m(55:56,[90,91])=5;
end
m_in=zeros(90,120);
m_in([16,76],1)=1;
m_in([15,75],120)=1;
m_in(1,[15,105])=1;
m_in(90,[16,106])=1;

r=sum(sum(m>=5))./sum(sum(m~=0));
end
```

s_ini.m

```
function s=s_ini(m,p,vmax)
%p 为初始有车率
%vmax 为最大速度
s=cell(1,3);
[a,b]=size(m);
%有无车
s{1}=(rand(a,b)<=p).*(m~=0);
s{1}(m==5)=0;
%vx
s{2}=randi([0,vmax],a,b).*(m==1).*s{1};
s{2}=s{2}+randi([-vmax,0],a,b).*(m==2).*s{1};
%vy
s{3}=randi([0,vmax],a,b).*(m==3).*s{1};
s{3}=s{3}+randi([-vmax,0],a,b).*(m==4).*s{1};
end
```

carin.m

```
function [s,JL_in] = carin(s,m,m_in,p,vmax)
[a,b]=size(s{1});
s_in=(rand(a,b)<p).*m_in;
s_new=(s{1}==0).*s_in;
JL_in(1)=sum(sum(s_new));%实际进入的
```

```
JL_in(2)=sum(sum(s_in));%本该进入的
s{1}=s{1}+s_new;
s{2}=s{2}+(m==1).*s_new.*randi(vmax,a,b);
s{2}=s{2}-(m==2).*s_new.*randi(vmax,a,b);
s{3}=s{3}+(m==3).*s_new.*randi(vmax,a,b);
s{3}=s{3}-(m==4).*s_new.*randi(vmax,a,b);
end
```

rand_slow.m

```
function s = rand_slow(s,k)
[a,b]=size(s{1});
sum_car_1=zeroshift(s{1},[-2,0])+zeroshift(s{1},[-1,0])+...
    zeroshift(s{1},[1,0])+zeroshift(s{1},[2,0]);%竖直
sum_car_2=zeroshift(s{1},[0,-2])+zeroshift(s{1},[0,-1])+...
    zeroshift(s{1},[0,1])+zeroshift(s{1},[0,2]);%水平
p_1=rand(a,b)<=(k(1).*abs(s{3})+k(2).*sum_car_1);
p_2=rand(a,b)<=(k(1).*abs(s{2})+k(2).*sum_car_2);
%右行
right=s{1}.*(s{2}>0);
s{2}=s{2}-p_2.*right;
%左行
left=s{1}.*(s{2}<0);
s{2}=s{2}+p_2.*left;
%下行
down=s{1}.*(s{3}>0);
s{3}=s{3}-p_1.*down;
%上行
up=s{1}.*(s{3}<0);
s{3}=s{3}+p 1.*up;
```

goforward.m

```
function s = goforward(s,m,i,T,p,vmax)
%此处的 p 是转弯概率
if mod(i,sum(T))<=T(1)-1
    flag=1;
else
    flag=2;
end
[a,b]=size(s{1});
%右行
right=s{1}.*(m==1);
[right_1,right_2]=find(right);
[right_2,i_r]=sort(right_2,'descend');
right_1=right_1(i_r);
for i=1:length(right_1)
    v=0;
    leave=0;
    incross=0;
    for v_i=1:vmax
        if right_2(i)+v_i>b
            s{1}(right_1(i),right_2(i))=0;
            s{2}(right_1(i),right_2(i))=0;
            %s{3}(right_1(i),right_2(i))=0;
```

```
            leave=1;
            break;
        end
        light=~((flag==2)&(m(right_1(i),right_2(i)+v_i)==6));
        if s{1}(right_1(i),right_2(i)+v_i)==0 && light
            v=v_i;
            if m(right_1(i),right_2(i)+v_i)==5 ...
                ||m(right_1(i),right_2(i)+v_i)==6
                incross=1;
                break;
            end
        else
            break;
        end
    end
    if leave==0 %没有离开才有这些
        v=min(v,s{2}(right_1(i),right_2(i))+1);
        s{1}(right_1(i),right_2(i))=0;
        s{1}(right_1(i),right_2(i)+v)=1;
        if incross~=1
            s{2}(right_1(i),right_2(i)+v)=v;
        else

s{2}(right_1(i),right_2(i)+v)=max(s{2}(right_1(i),right_2(i)),v);
        end
        s{2}(right_1(i),right_2(i))=0;
    end
end
%左行
left=s{1}.*(m==2);
[left_1,left_2]=find(left);
[left_2,i_r]=sort(left_2);
left_1=left_1(i_r);
for i=1:length(left_1)
    v=0;
    leave=0;
    incross=0;
    for v_i=-1:-1:-vmax
        if left_2(i)+v_i<1
            s{1}(left_1(i),left_2(i))=0;
            s{2}(left_1(i),left_2(i))=0;
            %s{3}(left_1(i),left_2(i))=0;
            leave=1;
            break;
        end
        light=~((flag==2)&(m(left_1(i),left_2(i)+v_i)==6));
        if s{1}(left_1(i),left_2(i)+v_i)==0 && light
            v=v_i;
            if m(left_1(i),left_2(i)+v_i)==5 ...
                ||m(left_1(i),left_2(i)+v_i)==6
                incross=1;
                break;
            end
        else
            break;
        end
    end
    if leave==0 %没有离开才有这些
        v=max(v,s{2}(left_1(i),left_2(i))-1);
```

```
        s{1}(left_1(i),left_2(i))=0;
        s{1}(left_1(i),left_2(i)+v)=1;
        if incross~=1
            s{2}(left_1(i),left_2(i)+v)=v;
        else
            s{2}(left_1(i),left_2(i)+v)=min(s{2}(left_1(i),left_2(i)),v);
        end
        s{2}(left_1(i),left_2(i))=0;
    end
end
%下行
down=s{1}.*(m==3);
[down_1,down_2]=find(down);
[down_2,i_r]=sort(down_2,'descend');
down_1=down_1(i_r);
for i=1:length(down_1)
    v=0;
    leave=0;
    incross=0;
    for v_i=1:vmax
        if down_1(i)+v_i>a
            s{1}(down_1(i),down_2(i))=0;
            %s{2}(down_1(i),down_2(i))=0;
            s{3}(down_1(i),down_2(i))=0;
            leave=1;
            break;
        end
        light=~((flag==1)&(m(down_1(i)+v_i,down_2(i))==6));
        if s{1}(down_1(i)+v_i,down_2(i))==0 && light
            v=v_i;
            if m(down_1(i)+v_i,down_2(i))==5 ...
                ||m(down_1(i)+v_i,down_2(i))==6
                incross=1;
                break;
            end
        else
            break;
        end
    end
    if leave==0 %没有离开才有这些
        v=min(v,s{3}(down_1(i),down_2(i))+1);
        s{1}(down_1(i),down_2(i))=0;
        s{1}(down_1(i)+v,down_2(i))=1;
        if incross~=1
            s{3}(down_1(i)+v,down_2(i))=v;
        else
            s{3}(down_1(i)+v,down_2(i))=max(s{3}(down_1(i),down_2(i)),v);
        end
        s{3}(down_1(i),down_2(i))=0;
    end
end
%上行
up=s{1}.*(m==4);
[up_1,up_2]=find(up);
[up_2,i_r]=sort(up_2);
up_1=up_1(i_r);
for i=1:length(up_1)
    v=0;
    leave=0;
```

```
    incross=0;
    for v_i=-1:-1:-vmax
        if up_1(i)+v_i<1
            s{1}(up_1(i),up_2(i))=0;
            %s{2}(up_1(i),up_2(i))=0;
            s{3}(up_1(i),up_2(i))=0;
            leave=1;
            break;
        end
        light=~((flag==1)&(m(up_1(i)+v_i,up_2(i))==6));
        if s{1}(up_1(i)+v_i,up_2(i))==0 && light
            v=v_i;
            if m(up_1(i)+v_i,up_2(i))==5 ...
                ||m(up_1(i)+v_i,up_2(i))==6
                incross=1;
                break;
            end
        else
            break;
        end
    end
    if leave==0 %没有离开才有这些
        v=max(v,s{3}(up_1(i),up_2(i))-1);
        s{1}(up_1(i),up_2(i))=0;
        s{1}(up_1(i)+v,up_2(i))=1;
        if incross~=1
            s{3}(up_1(i)+v,up_2(i))=v;
        else
            s{3}(up_1(i)+v,up_2(i))=min(s{3}(up_1(i),up_2(i)),v);
        end
        s{3}(up_1(i),up_2(i))=0;
    end
end
%路口
s=cross(s,m,p);
end
```

zero_shift.m

```
function X = zeroshift(X,k)
[m,n]=size(X);
X = circshift(X,k);
if k(1)>0
    X(1:k(1),:)=0;
elseif k(1)<0
    X(m+k(1)+1:m,:)=0;
end
if k(2)>0
    X(:,1:k(2))=0;
elseif k(2)<0
    X(:,n+k(2)+1:n)=0;
end
end
```

cross.m

```
function s = cross(s,m,p)
%此处的 p 是转弯概率
turn=s{1}.*(m==5|m==6);
[turn_1,turn_2]=find(turn);
rdp=randperm(length(turn_1));
turn_1=turn_1(rdp);
turn_2=turn_2(rdp);
for i=1:length(turn_1)
    vx=abs(s{2}(turn_1(i),turn_2(i)));
    vy=abs(s{3}(turn_1(i),turn_2(i)));
    loc=fun_loc(m,turn_1(i),turn_2(i));
    if (abs(vx)+abs(vy))==0
        if loc==1
            if m(turn_1(i),turn_2(i)-1)~=0
                if rand<=0.5
                    s{2}(turn_1(i),turn_2(i))=-1;
                else
                    s{3}(turn_1(i),turn_2(i))=+1;
                end
            else
                s{3}(turn_1(i),turn_2(i))=+1;
            end
        elseif loc==2
            if m(turn_1(i)-1,turn_2(i))~=0
                if rand<=0.5
                    s{2}(turn_1(i),turn_2(i))=-1;
                else
                    s{3}(turn_1(i),turn_2(i))=-1;
                end
            else
                s{2}(turn_1(i),turn_2(i))=-1;
            end
        elseif loc==3
            if m(turn_1(i)+1,turn_2(i))~=0
                if rand<=0.5
                    s{2}(turn_1(i),turn_2(i))=+1;
                else
                    s{3}(turn_1(i),turn_2(i))=+1;
                end
            else
                s{2}(turn_1(i),turn_2(i))=+1;
            end
        elseif loc==4
            if m(turn_1(i),turn_2(i)+1)~=0
                if rand<=0.5
                    s{2}(turn_1(i),turn_2(i))=+1;
                else
                    s{3}(turn_1(i),turn_2(i))=-1;
                end
            else
                s{3}(turn_1(i),turn_2(i))=-1;
            end
        end
    else
        if loc==1
            if m(turn_1(i),turn_2(i)-1)~=0
                if rand<=p
                    s{2}(turn_1(i),turn_2(i))=-abs(vy);
                    s{3}(turn_1(i),turn_2(i))=abs(vx);
```

```
60                  end
61              else
62                  s{2}(turn_1(i),turn_2(i))=0;
63                  s{3}(turn_1(i),turn_2(i))=abs(vx)+abs(vy);
64              end
65          elseif loc==2
66              if m(turn_1(i)-1,turn_2(i))~=0
67                  if rand<=p
68                      s{2}(turn_1(i),turn_2(i))=-abs(vy);
69                      s{3}(turn_1(i),turn_2(i))=-abs(vx);
70                  end
71              else
72                  s{2}(turn_1(i),turn_2(i))=-(abs(vx)+abs(vy));
73                  s{3}(turn_1(i),turn_2(i))=0;
74              end
75          elseif loc==3
76              if m(turn_1(i)+1,turn_2(i))~=0
77                  if rand<=p
78                      s{2}(turn_1(i),turn_2(i))=abs(vy);
79                      s{3}(turn_1(i),turn_2(i))=abs(vx);
80                  end
81              else
82                  s{2}(turn_1(i),turn_2(i))=(abs(vx)+abs(vy));
83                  s{3}(turn_1(i),turn_2(i))=0;
84              end
85          elseif loc==4
86              if m(turn_1(i),turn_2(i)+1)~=0
87                  if rand<=p
88                      s{2}(turn_1(i),turn_2(i))=abs(vy);
89                      s{3}(turn_1(i),turn_2(i))=-abs(vx);
90                  end
91              else
92                  s{2}(turn_1(i),turn_2(i))=0;
93                  s{3}(turn_1(i),turn_2(i))=-(abs(vx)+abs(vy));
94              end
95          end
96          vx=s{2}(turn_1(i),turn_2(i));
97          vy=s{3}(turn_1(i),turn_2(i));
98          if vx>0
99              vx=1;
100             %vy=0;
101         elseif vx<0
102             vx=-1;
103             %vy=0;
104         elseif vy>0
105             %vx=0;
106             vy=1;
107         else
108             %vx=0;
109             vy=-1;
110         end
111         if s{1}(turn_1(i)+vy,turn_2(i)+vx)==0 &&
112 m(turn_1(i)+vy,turn_2(i)+vx)~=0
113             s{1}(turn_1(i),turn_2(i))=0;
114             s{1}(turn_1(i)+vy,turn_2(i)+vx)=1;
115
116 s{2}(turn_1(i)+vy,turn_2(i)+vx)=s{2}(turn_1(i),turn_2(i));
117             s{2}(turn_1(i),turn_2(i))=0;
118
```

```
119
120  s{3}(turn_1(i)+vy,turn_2(i)+vx)=s{3}(turn_1(i),turn_2(i));
121              s{3}(turn_1(i),turn_2(i))=0;
122          end
123      end
124  end
     end
```

fun_loc.m

```
1   function loc = fun_loc(m,i,j)
2   %m 是地图
3   %i,j 是要判断点的坐标
4   if m(i+1,j)==5||m(i+1,j)==6
5       if m(i,j+1)==5||m(i,j+1)==6
6           loc=1;
7       else
8           loc=2;
9       end
10  else
11      if m(i,j+1)==5||m(i,j+1)==6
12          loc=3;
13      else
14          loc=4;
15      end
16  end
17  end
```

left_road.m

```
1   function zukang_left = left_road(n,flag)
2   global e_1;
3   x_i1 = e_1 /100 / 72;
4   if flag == 1
5       zukang_left = 20 * (1 + 0.15 * (n / e_1) ^ 0.4)...
6           + (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 * x_i1))...
7           + (x_i1 ^ 2) / (2 * n * (1 - x_i1))...
8           -  0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i1 ^ (2 + 5 *
9   x_i1));
10  else
11      zukang_left = (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
12  x_i1))...
13          + (x_i1 ^ 2) / (2 * n * (1 - x_i1))...
14          -  0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i1 ^ (2 + 5 *
15  x_i1));
16  end
17  end
```

stright_road.m

```
1   function zukang_stright = stright_road(n,flag)
2   global e_2;
3   x_i2 = e_2 /100 / 92;
4   if flag == 1
```

```
    zukang_stright = 30 * (1 + 0.15 * (n / e_2) ^ 0.4)...
          + (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
x_i2)) ...
          + (x_i2 ^ 2) / (2 * n * (1 - x_i2))...
          - 0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i2 ^ (2 + 5
* x_i2));
else
    zukang_stright = (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
x_i2)) ...
          + (x_i2 ^ 2) / (2 * n * (1 - x_i2))...
          - 0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i2 ^ (2 + 5
* x_i2));
end
end
```

right_road.m

```
function zukang_right = right_road(n,flag)
global e_3;
x_i3 = e_3 /100 / 80;
if flag == 1
    zukang_right = 20 * (1 + 0.15 * (n / e_3) ^ 0.4)...
      + (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 * x_i3)) ...
      + (x_i3 ^ 2) / (2 * n * (1 - x_i3))...
      - 0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i3 ^ (2 + 5 *
x_i3));
else
    zukang_right = (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
x_i3)) ...
      + (x_i3 ^ 2) / (2 * n * (1 - x_i3))...
      - 0.65 * ((10 / (n ^ 2)) ^ (1/3)) * (x_i3 ^ (2 + 5 *
x_i3));
end
end
```

PlotGriddata.m

```
function  PlotGriddata(x,y,z)
mx=min(x);%求 x 的最小值
Mx=max(x);%求 x 的最大值
my=min(y);
My=max(y);
Nx=100; %定义 x 轴插值数据点数，根据实际情况确定
Ny=100;%定义 y 轴插值数据点数，根据实际情况确定
cx=linspace(mx,Mx,Nx);%在原始 x 数据的最大值最小值之间等间隔生成 Nx
个插值点
cy=linspace(my,My,Ny);%在原始数据 y 的最大值最小值之间等间隔生成 Ny
个插值点
cz=griddata(x,y,z,cx,cy','cubic');%调用 matlab 函数进行立方插值
figure;
set(gcf,'outerposition',get(0,'screensize'));
surfc(cx,cy,cz) %绘制曲面
set(gca,'FontSize',24,'Fontname', 'Times New Roman');
xlabel('车流密度');
ylabel('时间');
```

```
zlabel('车流速度');
set(gca,'looseInset',[0 0 0 0]);
set(get(gca,'XLabel'),'Fontsize',20,'Fontname', '宋体');
set(get(gca,'YLabel'),'Fontsize',20,'Fontname', '宋体');
set(get(gca,'ZLabel'),'Fontsize',20,'Fontname', '宋体');
colorbar;
colormap(jet);
print(1,'-dpng','-r300','时间流量密度三维图');
```

附录二：NSGA—II 算法 MATLAB 代码(目标函数部分)

evaluate_objective.m

```
function f = evaluate_objective(x, M, V)%%计算每个个体的 M 个目
标函数值
global e_1 e_2 e_3;
x_i1 = e_1 /100 / 72;
x_i2 = e_2 /100 / 92;
x_i3 = e_3 /100 / 80;
f(1) =   20 * (1 + 0.15 * (x(1) / e_1) ^ 0.4)...
         +   30 * (1 + 0.15 * (x(2) / e_2) ^ 0.4)...
         +   20 * (1 + 0.15 * (x(3) / e_3) ^ 0.4);
f(2) = (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 * x_i1)) ...
          + (x_i1 ^ 2) / (2 * x(1) * (1 - x_i1))...
          -  0.65 * ((10 / (x(1) ^ 2)) ^ (1/3)) * (x_i1 ^ (2
+ 5 * x_i1))...
          + (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
x_i2)) ...
          + (x_i2 ^ 2) / (2 * x(2) * (1 - x_i2))...
          -  0.65 * ((10 / (x(2) ^ 2)) ^ (1/3)) * (x_i2 ^ (2
+ 5 * x_i2))...
          + (10 * (1 - 0.5 ) ^ 2) / (2 * (1 - 0.5 *
x_i3)) ...
          + (x_i3 ^ 2) / (2 * x(3) * (1 - x_i3))...
          -  0.65 * ((10 / (x(3) ^ 2)) ^ (1/3)) * (x_i3 ^ (2
+ 5 * x_i3));
end
```