浙江工业大学

本科毕业设计论文

题目：基于 AGV 小车的
搬运系统优化控制研究

作者姓名 褚青君

指导教师 赵云波教授

专业班级 自动化 1302 班

学　　院 信息工程学院

提交日期 2017 年 6 月 4 日

浙江工业大学本科毕业设计论文

基于 AGV 小车的搬运系统优化控制研究

作者姓名：褚青君

指导教师：赵云波教授

浙江工业大学信息工程学院

2017 年 6 月

Dissertation Submitted to Zhejiang University of Technology
for the Degree of Bachelor

Research on Majorization Control of Handling System Based on AGV

Student: Xuan Qingjun

Advisor: Professor Zhao Yunbo

College of Information Engineering
Zhejiang University of Technology

June 2017

浙江工业大学

本科生毕业设计(论文、创作)诚信承诺书

本人慎重承诺和声明：

1. 本人在毕业设计（论文、创作）撰写过程中，严格遵守学校有关规定，恪守学术规范，所呈交的毕业设计（论文、创作）是在指导教师指导下独立完成的；

2. 毕业设计（论文、创作）中无抄袭、剽窃或不正当引用他人学术观点、思想和学术成果，无虚构、篡改试验结果、统计资料、伪造数据和运算程序等情况；

3. 若有违反学术纪律的行为，本人愿意承担一切责任，并接受学校按有关规定给予的处理。

学生（签名）：褚青君

2017 年 06 月 04 日

浙 江 工 业 大 学

本科生毕业设计（论文、创作）任务书

专 业 自动化 班 级 1302 学生姓名/学号 祺青君/201303080324

一、设计（论文、创作）题目： 基于 AGV 小车的搬运系统优化控制研究

二、主要任务与目标：

1.熟悉使用 AGV 小车实验仿真平台；2.总结现有的 AGV 小车搬运控制算法；3.将现有典型控制算法在前述平台成功实现，并针对某些特定需求研究新算法及实现。

三、主要内容与基本要求：

1.学习了解 AGV 相关开发材料；2.阅读相关文献，了解现有相关算法；3. 研究算法设计和算法在给定平台的实际应用问题；4. 撰写毕业论文。

四、计划进度：

2016.12.19~2017.2.20 收集相关资料文献，学习相关 AGV 知识；完成外文翻译、文献综述；熟悉课题，做好开题准备；

2017.2.21~2017.3.12 完成开题报告，参加开题交流；

2017.3.13~2017.4.23 完成平台搭建和算法验证等工作，接受中期检查；

2017.4.24~2017.5.28 在平台基础上进行算法设计和实现等工作。撰写毕业论文初稿；

2017.5.29~2017.6.12 论文修改，毕业答辩，提交相关文档资料。

五、主要参考文献：

[1] 王喜富. 物联网与智能物流[M]. 清华大学出版社, 2014. [2] 孙奇. AGV 系统路径规划技术研究[D]. 浙江大学, 2012. [3] 孟文俊，刘忠强. 视觉导引 AGV 的路径跟踪控制研究[J]. 控制工程，2014,21（3）：321-325.

任务书下发日期 2016 年 12 月 19 日

设计（论文、创作）工作自 2016 年 12 月 19 日 至 2017 年 6 月 12 日

设计（论文、创作）指导教师 赵云波

学科（方向）负责人 _____

主管院长 _____

基于 AGV 小车的搬运系统优化控制研究

摘　要

随着物联网与智能物流系统，信息化管理系统，柔性制造系统(FMS)等的发展，自动引导车系统（AGVs）的应用日益广泛，AGV 系统在工业生产中的应用可以解放劳动生产力，提高工作效率，降低生产运输成本。未来在各行各业 AGV 必将保持较快的发展趋势。如何控制优化 AGV 小车的搬运系统已成为研究 AGV 的一大热点，其中路径规划技术作为 AGV 系统的核心技术之一，更是当下研究 AGV 系统技术的潮流。

本文的研究内容是基于 AGV 小车搬运系统优化控制研究，主要针对 AGV 系统路径规划方面，对于单个 AGV 小车的路径规划，利用基于栅格法的深度优先搜索算法和基于拓扑图论法的 Dijkstra 算法来实现对 AGV 运行地图环境建模和路径规划，并在 Visual C++ 6.0 环境平台进行仿真上进行实验的仿真验证。对于多 AGV 系统的路径规划，利用 Dijkstra 算法和时间窗原理相结合，实现多 AGV 无碰撞路径规划。论文的主要工作如下：

1.综述课题研究背景与意义，以及 AGV 小车研究的国内外现状，介绍了 AGV 系统的关键技术。

2.对于单个 AGV 小车的路径规划，利用基于栅格法的深度优先搜索算法和基于拓扑图论法的 Dijkstra 算法来实现对 AGV 运行地图环境建模和路径规划，并在 Visual C++ 6.0 环境平台进行仿真上进行实验的仿真验证。

3.对于多 AGV 系统的路径规划方面，研究的是怎样为每台 AGV 规划一条无碰撞、无冲突的路径。在单 AGV 路径规划的基础上，提出此问题解决的有效方法。

关键词：AGV、路径规划、深度优先搜索算法、Dijkstra 算法、时间窗

Research on Majorization Control of Handling System Based on AGV

ABSTRACT

With the development of ASV and intelligent logistic system, information management system and flexible manufacturing system (FMS), the application of AGVs is becoming more and more widely. The application of AGV system in industrial production can liberate labor productivity and improve work. Efficiency, reduce production and transportation costs. The future development of AGV will maintain a rapid development trend. How to control and optimize the handling system of AGV has become a hot topic in the study of AGV. Among them, path planning technology is one of the core technologies of AGV system, and it is the trend of AGV system technology.

The research content of this paper is based on the optimization control of AGV trolley handling system. Based on the path planning of AGV system, the depth-first search algorithm based on grid method and the Dijkstra algorithm based on topological graph theory are used for the path planning of single AGV The modeling and path planning of AGV operation map environment is realized, and the simulation is carried out on the Visual C ++ 6.0 environment platform. For the multi-AGV system path planning, the multi-AGV collision-free path planning is realized by combining the Dijkstra algorithm and the time window principle. The main work of the paper is as follows:

1. Review the background and significance of the paper, as well as AGV car research at home and abroad, introduced the AGV system key technology.

2.For the path planning of a single AGV car, the Dijkstra algorithm based on the raster method and the Dijkstra algorithm based on the topological graph theory are used to realize the modeling and path planning of the AGV runtime map environment and the Visual C ++ 6.0 environment platform Simulation of the simulation on the experiment.

3. For multi-AGV system path planning, the main study is how to plan for each AGV a collision-free, conflict-free path. On the basis of single AGV path planning, this paper proposes an effective method to solve this problem.

Keywords: AGV, Path planning, Depth-first search, Dijkstra algorithm，Time window

目　　录

第 1 章　绪　论

1.1　课题研究背景及意义

自动导引车（automated guided vehicle，AGV）是一种智能的搬运小车，属于移动机器人范畴，它与传统的货物搬运车相比，具有自载导航设备，可以根据指定的路径自主行走，可以智能调整运行模式，可以自主充电，续航能力持久，具有保护设备等优势。AGV 智能物流系统中的搬运工具，能够满足柔性制造系统和立体仓库的要求，是整个自动化物流系统中最核心的部件之一[1]。

AGV 是在工业生产中安装的设备数量增长最快的材料输送设备之一，并且成为生产运输的常用工具。该设备可以快速响应经常变化的运输模式，并可集成到全自动控制系统中。AGV 已经越来越多地用于在配送物流领域中区输送各种应用材料，能够自主行走、自动控制、智能判断，并可由系统调度安排完成相应的工作任务，如取货、卸货、拖运、和升降等。

AGV 起源于美国，1954 年第一台 AGV 发明并在仓储投入使用并实现货物自动入库，此后，AGV 迅速成为了物流运输系统中的新星，在许多行业的得到相当广泛的应用，如存储仓库、集成箱码头、生产车间等执行搬运工作。AGV 作为智能移动机器人的一种，它不只是应用于普通生产线货物搬运场所，而且军事、核电、化工等具有放射、有害气体的特殊工作场所中 AGV 也有相应的应用，AGV 可以代替操作人员在上述环境恶劣的场所中进行一些危险性较大的工作。AGVs 是指 AGV 系统，由多个 AGV 小车构成，协同完成安排的相应任务工作，有序控制有规则地控制多 AGV 能高效的完成物料的运输工作，多 AGV 系统对空间环境的适应性较好，可以充分利用场地空间资源，调整输送线路进行适当的调度。AGV 由于其自动化程度高，灵活性强，已经成为智能物流系统中不可或缺的运载工具，是提高物流系统自动化、集成化、信息化水平重要手段之一[2][3][4]。

1.2　AGV 国内外研究现状

福特汽车公司于 1913 年将最早的自动搬运车应用到汽车底盘装配上，体现了无人操作搬运车的优越性[3]。但当时是有轨导引的(现在称为 RGV)，到了上世纪 50 年代中

期，英国人首次用电磁感应作为导航方式导引 AGVS，采用地下埋线的方式替代了地面有轨道引方式，1959 年 AGV 开始应用于车间仓库和物流运输系统中。

20 世纪 60 年代，AGV 系统中应用了计算机信息技术，AGV 系统从自动化仓储进入到柔性制造系统（FMS）中，从而使 AGV 得到了迅速的发展。虽然 AGV 是在美国发明的，却在英国等欧洲国家得到快速发展和广泛使用，并成为生产制造和物流系统中一种新兴的货物搬运工具。

我国 AGV 的研究较国外有很大的差距，起步较晚，因受历史条件、技术水平等各方面的影响，我国的 AGV 技术在总体发展水平上与美国等强国相比还是有很大差距的，但通过不断地努力，这种差距正在缩小。目前国内的绝大部分工厂自动化水平不高，搬运任务仍然是停留在人工操作机械来完成，20 世纪 80 年代后期，北京起重运输机械研究所研制出第一台 ADB 型 AGV。20 世纪 90 年代初，邮电部北京邮政科学研究所成功研制了 WZC 及 WZC-1 两种类型的 AGV，并应用于上海新火车站邮政枢纽、济南军区仓库[3]。在国家“863”计划支持下，中科院沈阳自动化所完成了多项移动机器人应用基础研究和应用技术开发项目，并开发出且在实践应用中较为成熟的 AGV（电磁导引）及其系统技术[11]。

图 1-1　AGV 在各行各业的应用实例

1.3 AGV 系统的关键技术

AGV 作为智能物流的物料搬运工具，是自主移动机器人的一个重要领域的应用，其主要技术源于自主移动机器人技术。自主移动机器人技术是现在智能信息时代的一个研究热点。AGV 从出现至今，随着移动机器人技术的研究发展，AGV 的技术也已经有了许多突破性的进展。AGV 的关键技术与自主移动机器人的关键技术相似，其难点和关键点主要包括导航技术，路径规划技术，任务调度技术，多 AGV 协调控制技术，网络通信技术[2]。

1.3.1 导航定位技术

导航定位技术是 AGV 系统研究的最重要技术之一，主要是解决自动引导小车“我在哪”的问题。AGV 的导航系统要求能够判断自身位置以及目标的位置，实现指引机器人向目标方向前进，AGV 的导航定位主要是通过相应的传感设备获取周围环境的信息来确定自身的位置情况。AGV 发展应用至今，导航定位多种技术途径，主要包括以下几种：电磁导航技术、直接坐标定位技术、激光导航技术、声学导航技术、光学导航技术、视觉导航技术、惯性导航技术、GPS 导航技术[17]。

1.3.2 路径规划技术

车辆路径规划是指确定将由车队执行的一个或多个路径或路径序列的过程。目的是访问地理上分散在必须被关注的预定位置的一组节点。根据 Vivaldini 等人，在工业环境中的 AGV 路径的任务是为每个 AGV 从其当前位置到期望目的地找到路径，确保沿着所选路径的无冲突行进。近年来，已经提出了几种算法来解决路径问题。它们分为两类：静态路径算法和动态路径算法。

路径规划技术是解决 AGV 从起点到目标终点的行走路线问题，是解决“到哪去”、“怎样去”的问题[15]。针对已知环境、未知环境和动态环境等不同的环境，需要应用不同的路径规划算法。路径规划一直是机器人研究领域的热点和难点问题，现已经有大量的算法用于机器人路径规划，常用的动态路径规划算法有 A*算法，人工势场法，模糊路径规划等算法，常用的静态路径规划算法有图论算法、蚁群算法，神经网络，遗传算法等[15]。根据系统要求和性能指标的不同，为 AGV 系统选取恰当的路径规划算法，并对算法改进，使其符合系统的性能要求，是一个理论应用于实践的问题。

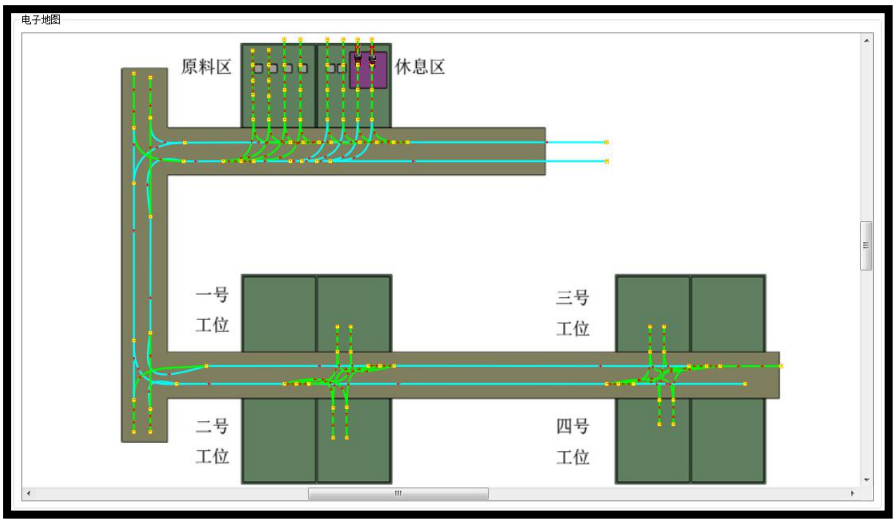

图 1-2　路径规划示意图

1.3.3　任务调度

任务调度问题是多 AGV 系统的基本问题之一。任务调度旨在最小化成本，确保无冲突的操作和任务履行。调度包括考虑到一些限制（例如期限，优先级等）而为运输系统分配任务列表。其最终目标与资源的优化相关联，遵守生产时间，最小化 AGV 数量，但同时需要保持或最大化生产率。大多数调度问题的主要目标是尽可能快地运输负载以满足时间窗口约束。其他标准可以是最小化最大负载等待时间和关键队列中的最大项数。

然而，调度可以分为两个关键因素：是确定劳动操作的计划开始和完成时间的预测机制，以及监视调度进度并处理意外事故如故障，故障，取消，日期变化等。调度系统决定车辆何时何地必须执行其任务。 如果在规划一段工作之前所有任务都已知，则可以离线解决问题。实际上，在路线规划之后任务信息的改变使离线调度变得复杂。合理的性能指标应该综合考虑各 AGV 的执行任务次数，充电时间，空闲时间，空载行走时间，负载行走时间等多个因素，以对系统进行性能优化。

1.3.4　多 AGV 协调控制

对于规模大、产量高的生产制造系统来说，单 AGV 工作远远无法满足日常工作量的需求，生产流程有许多工作需要多个 AGV 一起完成。多 AGV 系统的应用可以提高

生产效率，最大限度地利用场地资源，比单 AGV 系统更加具有实用性，然而，多 AGV 系统并不仅仅是数个 AGV 的简单组合。多 AGV 系统会存在多个 AGV 同时工作时的路径冲突，相互碰撞，路径阻塞等一系列问题，如果无法合理地解决多 AGV 系统中存在的问题，就会有损系统性能，甚至会引起系统崩溃。多 AGV 系统的技术难点是如何为每个 AGV 规划出一条无冲突的最优路径。

现在在多 AGV 协调控制的技术方法上，许多学者也做了很多的研究，如两阶段控制协调法，基于交通规则的优先级法，速度调节策略，几何路径调节策略[16]。至今，在多 AGV 协调控制的解决方法应用上仍然不成熟，有很多技术难点仍需更深一层优化研究。

1.3.5 网络通信技术

监控终端与 AGV 车载端间采用无线通讯方式，构成无线局域网。监控终端通过无线局域网向 AGV 车载端发出控制指令、任务调度指令、避碰调整指令。监控终端同时接收来自 AGV 车载端的通讯信号。AGV 通过无线局域网向控制台报告各类任务指令的执行情况、AGV 车辆状态与任务状态，提供物流统计信息用于物流优化。

1.4 主要研究内容

本论文依据某生产车间的场地环境提出AGV的路径规划问题。从而解决AGV系统路径规划问题中可能出现的问题，进行路径规控制划算法和优化方法的研究。主要章节内容如下：

第一章是总体上综述课题研究背景与意义，以及 AGV 小车研究的国内外现状，介绍了 AGV 系统的导航定位、路径规划等关键技术，最后介绍本文的组织结构和章节安排。

第二章是描述 AGV 路径规划问题，对地图环境的建模做出合理的假设，对于单个 AGV 小车的路径规划，利用基于栅格法的深度优先搜索算法和基于拓扑图论法的 Dijkstra 算法来实现对 AGV 运行地图环境建模和路径规划，并在 Visual C++ 6.0 环境平台进行仿真上进行实验的仿真验证。

第三章是对多 AGV 系统路径规划中存在的相关问题进行概述，并提出相应的解决方法。详细介绍了利用 Dijkstra 算法和时间窗原理相结合，实现多 AGV 无碰撞路径规

划。包括时间窗模型、时间计算、时间窗计算，无碰撞路径规划算法过程。

第四章是对所做毕业设计工作的做出总结，并对研究内容做出进一步的展望。

1.5 本章小结

本章内容在查阅国内外AGV相关文献的基础上，介绍了课题的研究背景及意义，以及AGV国内外研究现状和发展历程，之后详细价绍了AGV系统的路径规划、多AGV协调控制等关键技术问题，为后面的AGV路径规划研究工作做好准备。最后介绍本文的主要研究内容。

第 2 章　单 AGV 系统路径规划

路径规划是在在一个给定的行驶区域环境中，在起始点和目标终点已知的情况下，AGV 根据某一性能指标规划出从起点到目标终点的一条无碰撞优化路径。由于 AGV 需要在已知行驶区域环境信息中进行路径规划，并在不断地前进过程中，通过传感器不断感知行驶区域四周环境状况信息，及时做出相应调整，引导 AGV 到达目标终点。路径规划涉及环境地图描述和路径搜索方法这两个方面问题。路径搜索是在环境地图的基础上，对环境信息执行相应的搜索，能够快速地搜索到目标终点。首先依据 AGV 小车行驶区域的实际情况提出基本假设。

2.1　基本假设

1）全部的 AGV 小车都具有相同的物理参数、性能指标，即 AGV 的结构形状、行驶速度、最小转弯半径、爬坡能力、最大运载容量等都相同。本文研究中 AGV 的行驶速度为匀速恒定不变；

2） 不考虑 AGV 行驶区域的高度信息，可将行驶路径规划规划简化为二维平面内的路径规划问题；

3）行驶区域地面的摩擦因数、平整度相同；

4）行驶区域的全局环境信息是已知的，并且路径具有连通性，即任意两点间可到达；

5)AGV 的总体数量应不超过地图的最大容量，即环境地图中工作站点的数量总和。

2.2　环境地图建模

环境地图是一种根据小车行驶区域的实际环境信息建立，便于计算机储存和计算的电子地图，是对实际区域环境信息的描述，其作用是让 AGV 确定所在位置，同时也是让 AGV 识别环境中的物体具体是什么，是路径规划的基础。AGV 首先要认识到自身在行驶区域环境中的确切位置，才能够完成路径规划和调度安排。环境地图是 AGV 路径规划的基础，环境地图建立的质量对路径搜索的效率有着非常大的影响。本文使用栅格法和拓扑图伦法分别创建两种不同的环境地图模型。

栅格法将 AGV 的工作环境分解成一系列具有二元信息的网格单元，多采用四叉和八叉树表示。该法以单位栅格划分，有障碍物的地方标记为障碍物栅格且拥有较高的灰度值，反之就标记为自由栅格且拥有较低的灰度值，这样现实地图就转化为图 2-1 形式，其中 0 表示自由栅格，即小车允许通过的路径。1 表示障碍栅格，即小车不允许通过的障碍物位置。

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0

图 2-1 栅格地图

拓扑地图法是一种紧凑的环境地图建模方法。这种方法以图的形式来表现现实环境的连通性，通过特定的点来对环境进行描述同时用两点之间的连线来表达特定点之间的路径连通，拓扑地图是一种有效的表示方法。由于两点之间的连线并没有包含长度信息，为了将环境模型进行更好的表达，出现了新的混合拓扑和尺度地图相结合的方法，通过加入节点之间的长度信息，这样的地图表示方法同时具有拓扑地图的高效性和尺度地图的精确性，如图 2-2 所示，在 AGV 路径搜索的过程中同时使用拓扑信息和长度信息会得到更加符合实际情况的最优路径。

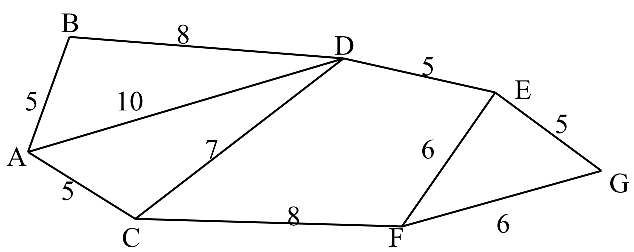

图 2-2 拓扑地图

2.3 基于深度优先搜索的路径规划

2.3.1 深度优先搜索法基本原理

深度优先搜索法基本原理：假设初始状态图是是图中所有节点都没有被访问，则深度优先搜索可以从图中的某个节点 v 开始，访问此节点，然后依次从 v 的没有被访问的相邻点出发深度遍历图，直到图中所有和 v 有有路径连通的节点都被访问到；如果此时图中还有节点没有被访问，则选择图中某个没有被访问的节点作为起始点，重复上述步骤，直到图中所有节点都被访问为止。

2.3.2 深度优先搜索路径实现

深度优先搜索的算法步骤：

环境地图中的所有点都没有被访问，从任意节点 S 开始；

1）访问节点 S；

2）按顺序从 S 相邻的没有被访问的点开始，对环境地图使用深度优先搜索；直到环境地图里和 S 存在连通性的全部的点都被访问；

3）遍历全部的节点，直到全部节点都作为搜寻的起始点访问过才结束。

深度优先搜索算法（DFS）的伪代码如下所示：

```
DFS（Node）
If(Node=目标终点)
then  //找到目标重点，结束
For each next∈d[Node]
Do DFS(next)
End
```

最短路径搜索流程根据图 2-3 进行。首先，在离散的栅格地图中运用深度优先搜索法搜索得到可行的点的有序集合；其次，按照点和点之间的关系，依据之前一有点的连通路径，搜寻两节点之间的可连通路径，依据 AGV 刚开始的行驶方向确定可行驶的路段，直到到达目标终点；然后，在全部有效可行的路径中，最终的路径选择对象是到目标终点的最短的一条路径；最后，输出最终的选择路径结果。

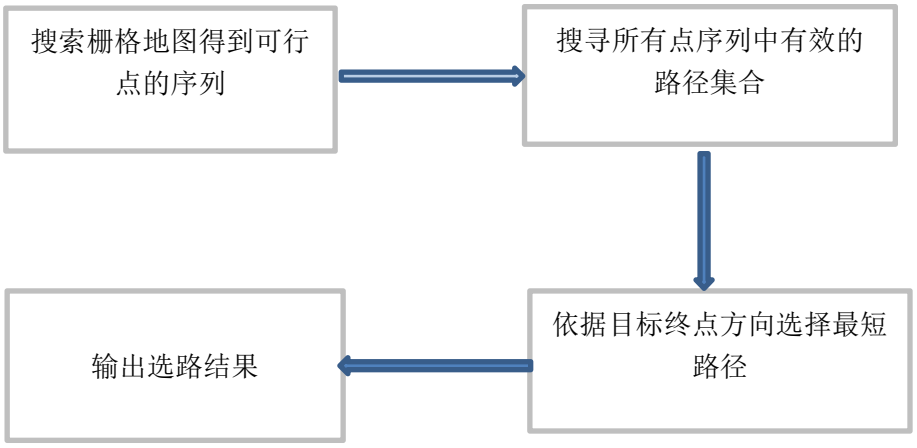

图 2-3　路径搜索流程

2.3.3　仿真结果

本文使用 Visual C++ 6.0 环境平台进行仿真，利用栅格法设计 AGV 路径规划的电子地图模型，地图模型界面如图 2-4 所示，模型界面包括路径规划区域、设置起点、设置终点、设置货架点、开始寻路、地图重置功能按钮，模型界面人机交互灵活方便，可以依据实际情况进行相关操作。根据实际的地图环境信息，在路径规划区域设置货架位置，货架用黄色的栅格表示，如图 2-5 所示。在实际规划的情况中，把货架作为障碍物处理，路径规划区域中黑色的栅格表示可行路径区域。

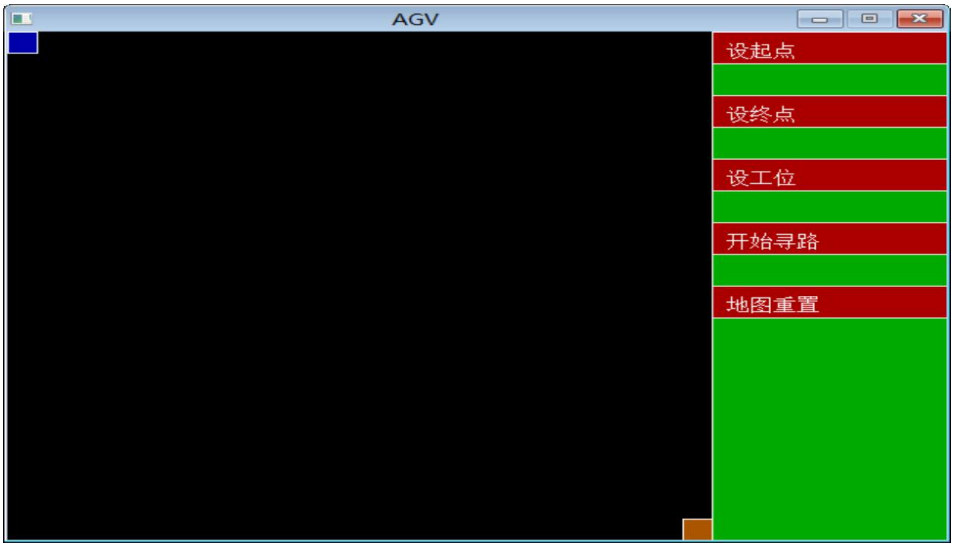

图 2-4　地图模型设置界面

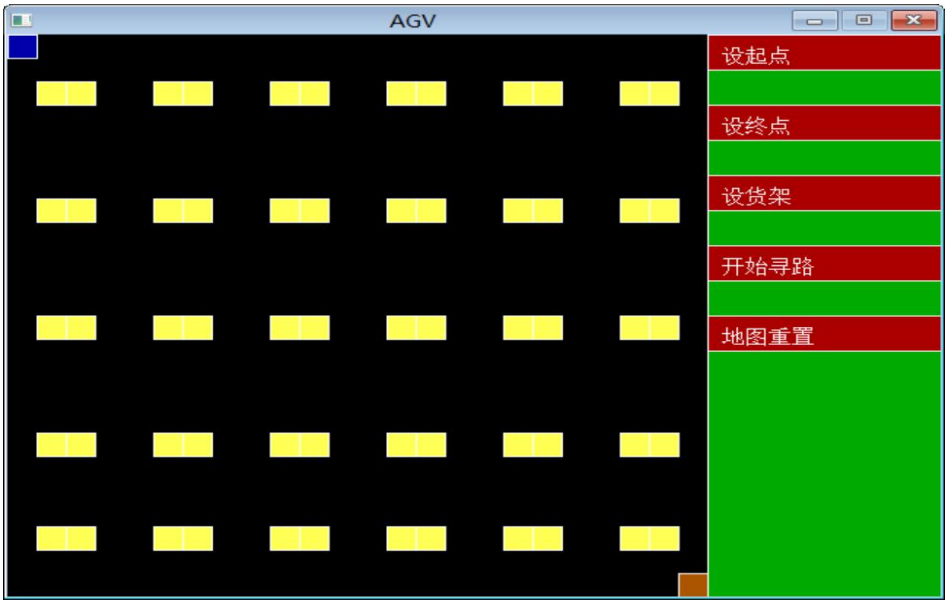

图 2-5　路径规划界面

在图 2-5 中所示的路径规划界面认可根据需要在任意位置设置起点和终点，例如设置货架 1 为起点，货架 29 为终点，规划这两点间的可行路径，路径规划显示结果如图 2-6 所示，所需步长如图 2-7 所示。

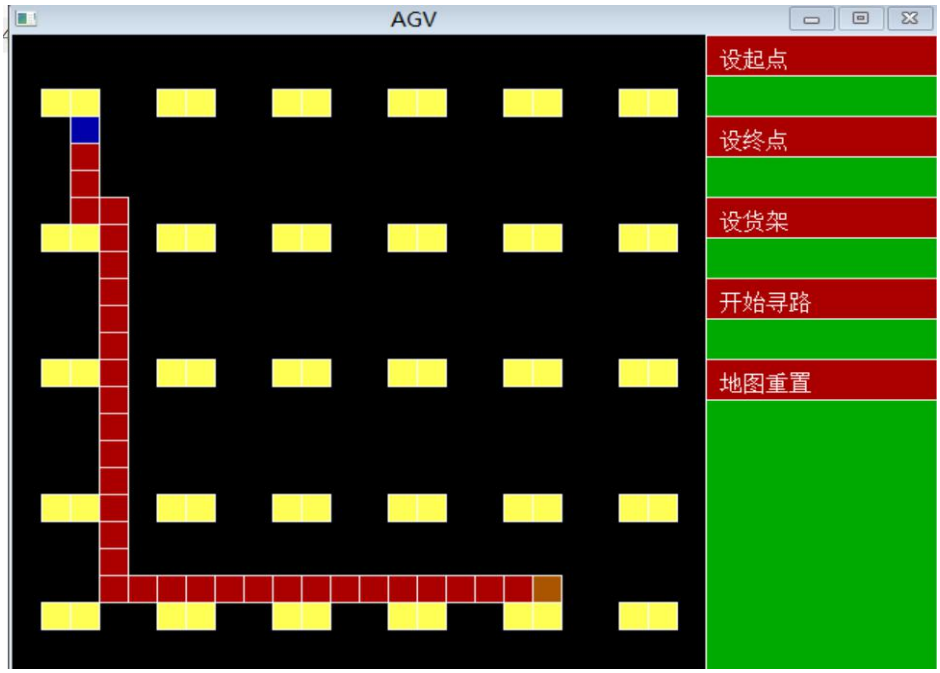

图 2-6　路径规划结果

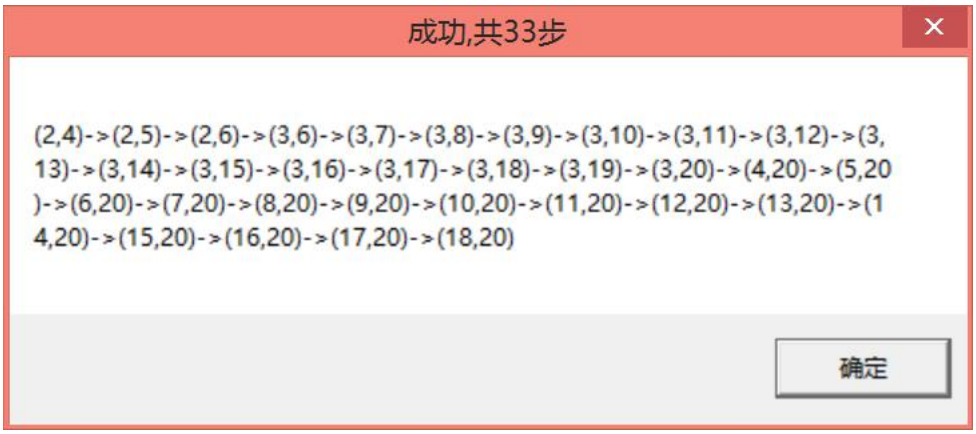

图 2-7 路径步长

2.4 基于 Dijkstra 算法的路径规划

2.4.1 Dijkstra 算法基本原理

在有向加权图 G(V, E)中，V 是指有向加权图中全部点的集合，E 是指图中全部弧段的集合，是两个节点连线组成的有序边[19]。每条边都有对应的权重，用 w(e)表示，w(u,v)是指从节点 u 到节点 v 的花费值。花费可以表示两个节点间的距离长度。Dijkstra 算法可以找出点 s 到点 t 的最小花费，也能够解决两点间最短路径问题。

Dijkstra 算法的基本原理是，如果路径 v_1 -v_2 -v_3 -v_4 -v_5 -v_6…… v_n（v 是指图中节点）是从 v_1 到 v_n 的最短路径，则路径 v_1 -v_2 -v_3 -v_4 -v_5 -v_6…… v_{n-1} 也肯定是 v_1 到 v_{n-1} 的最短路径。按照这各递归原理，算法以起点作为中心向外层层扩展，直到扩展到目标终点。具体实现过程为：在 G(V, E)图中，把所有节点划分成两组，分别是 S 集合和 U 集合，S 集合代表已经搜索到最短路径的点集，U 集合表示没有求出最短路径的节点集合，按递增的规则排序最短路径的长度，依次把 U 集合的节点添加到 S 里，按照以下准则：从起点至 S 集合中的每个节点的最短路径长度小于或等于从起点到 U 集合中任何一个节点的最短路径距离，每个节点与起始点间都有一个相应的距离，用 l(v_i)表示，S 集合中节点的距离就表示起点到节点的最短路径距离。U 集合中的节点距离，表示从起点到该节点并且只经过 S 集合中节点的此刻最短路径。

标记量：

S 具有距离标记的节点集；

l(v)节点 v 到起点的距离；

f(v)表示节点 v 的父节点，用于最短路径所有节点集的表示。

2.4.2 Dijkstra 算法路径规划实现步骤

Dijkstra 算法具体步骤：

（1）初始状态时，集合 S 中仅仅有起始点 v_0，u= v_0，l(u)=l(v_0)=0，集合 U 中包括除去起点之外的其它全部节点；

（2）判断集合 U 是否是空集，如果是空集，则跳出循环；

（3）对 v∈U，如果 l(v)>l(u)+e(v,u) ，则 l(v)=l(u)+e(v,u)，f(u)=u；

（4）令从 U 集合中选出 l(v)最小值 l(v*)的 v*，使得 S=S∪{v*}，u=v*；

（5）重复（2）；

算法流程如图 2-8 所示。

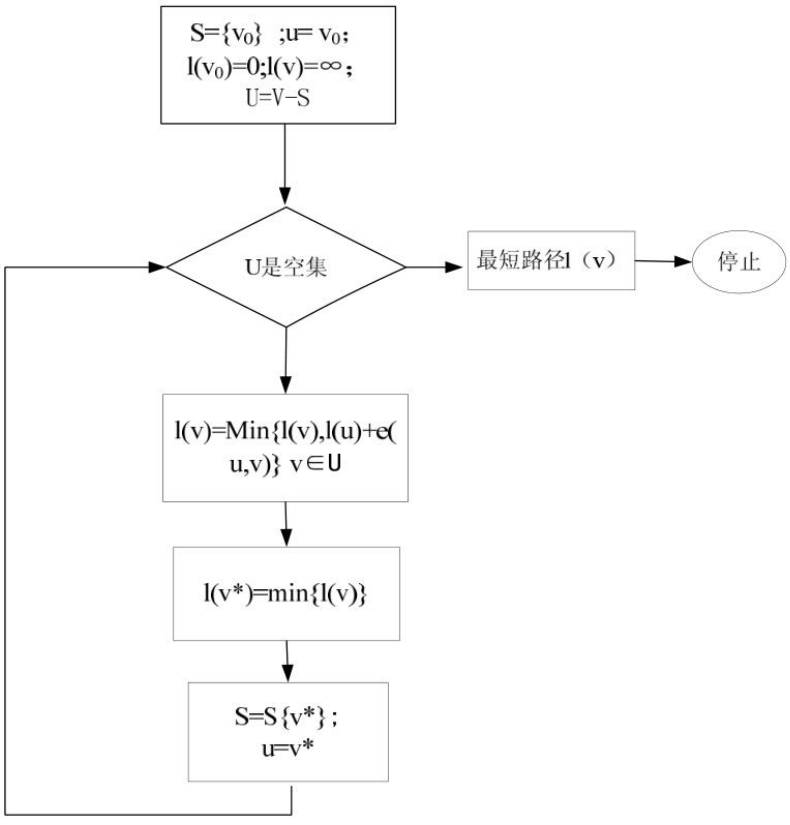

图 2-8 Dijkstra 算法流程图

2.4.3 仿真结果

本文使用 Visual C++ 6.0 环境平台进行仿真，利用拓扑图论法设计 AGV 路径规划的拓扑关系图。根据实际情况创建一个 G(12, 17)的拓扑关系图，各节点间的拓扑关系如图 2-9 所示，各节点的入度信息如图 2-10 所示。利用 Dijkstra 算法实现最短最短路径规划，本文以 v1 为起点搜寻到个点节点的最短距离，并输出最短距离以及行驶路径。仿真结果如图 2-11 所示。

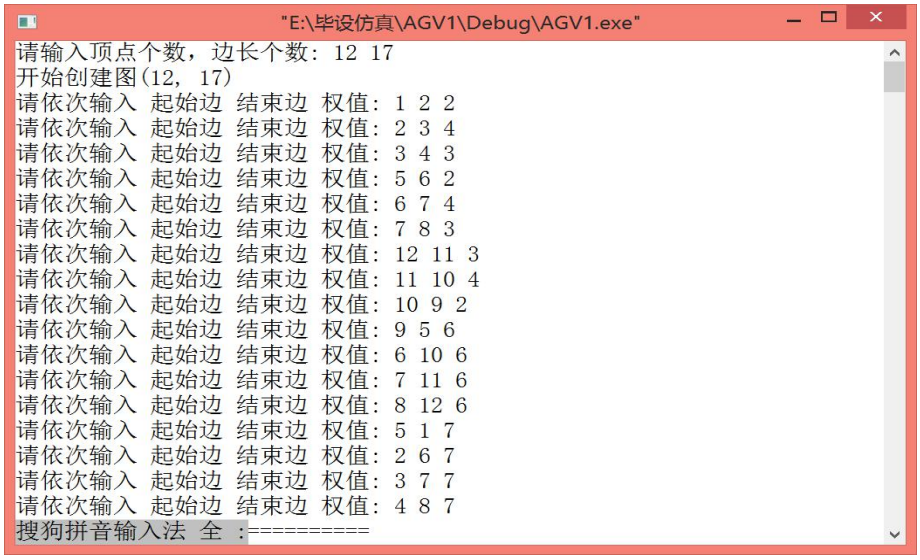

图 2-9 各节点间拓扑关系

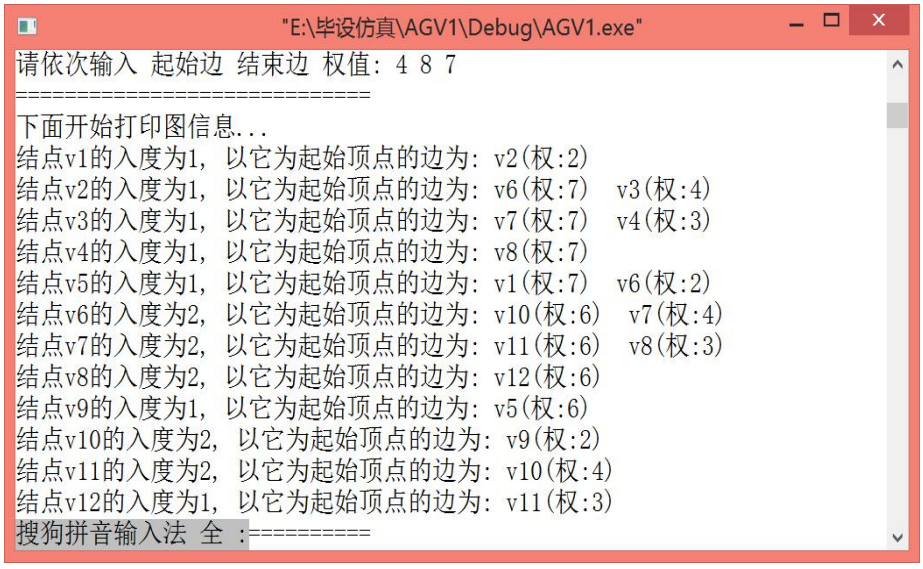

图 2-10 各节点间入度信息

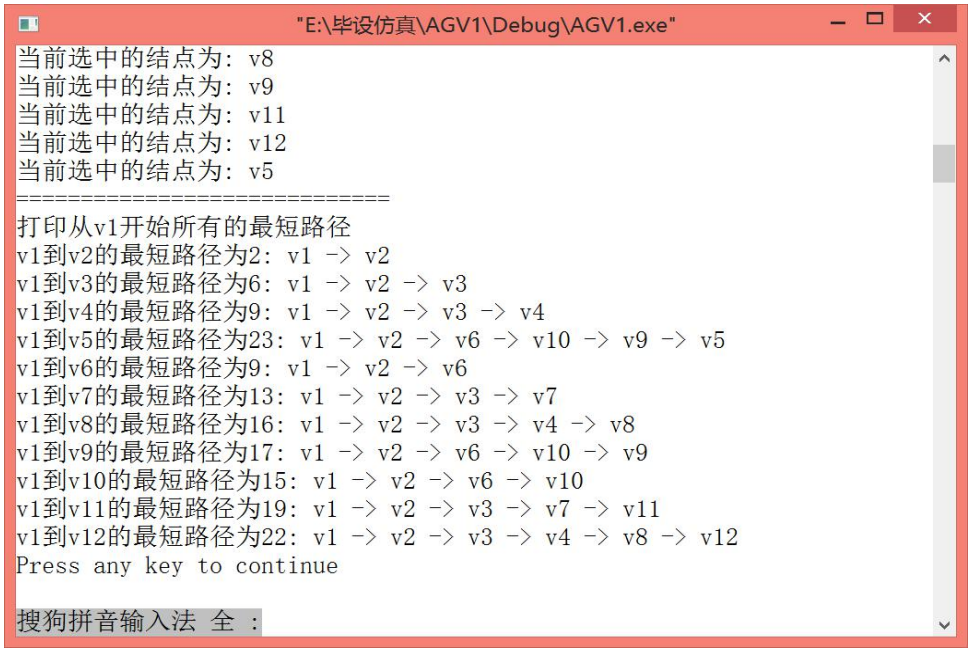

图 2-11　v1 到个点的最短路径

2.5 本章小结

本章主要研究 AGV 路径规划问题，对地图环境的建模做出合理的假设，对于单个 AGV 小车的路径规划，利用基于栅格法的深度优先搜索算法和基于拓扑图论法的 Dijkstra 算法来实现对 AGV 运行地图环境建模和路径规划，并在 Visual C++ 6.0 环境平台进行仿真上进行实验的仿真验证。

第 3 章 多 AGV 系统路径规划

3.1 多 AGV 路径问题描述

3.1.1 问题的由来

随着自动引导小车在物流运输系统中的广泛运用，单个AGV远远无法满足物流运输系统的需求，由于生产运输过程的任务的多样性和差异性，需要有多个AGV组成的AGV系统共同完成调度运输需求。然而在多个AGV组成的复杂的运输系统中，AGV在执行任务行驶的过程中就会遇到很多问题，如碰撞冲突、交通阻塞等。这时，为了能够让AGV运输系统安全可靠、高效率地运行。这就需为执行任务的AGV规划一条无碰撞、无冲突的路径。

多AGV系统是由多台AGV组成，需要在保证无碰撞运行的条件下共同完成分配的指定任务，这就使得各个AGV之间的协调控制方法显得非常重要，因此如果仅仅只是对多个单AGV进行路径规划的话，就一定会发生多台AGV之间的冲突从而引起系统死锁情况。在一个多AGV系统中，主要需要解决的问题是各个AGV之间碰撞问题和时间冲突问题。这就需要了解AGV之间的冲突类型和解决方法。

3.1.2 冲突的类型

在多AGV系统中，冲突类型主要有三种，即路口冲突、超越冲突和相向冲突。下面，分别对这三种冲突发生的实际情形做分析。

（1）路口冲突

路口冲突是多个AGV小车从不同方向同时经过同一个路口引起的冲突，路口冲突示意图如图3-1所示。

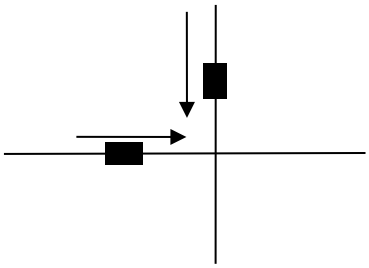

图3-1　路口冲突示意图

（2）超越冲突

超越冲突是两辆AGV小车在同一路段同向行驶，由于速度不同，后面一辆速度大于前面一辆，而追赶上引起的冲突，超越冲突示意图如图3-2所示。

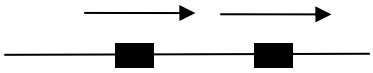

图3-2　超越冲突示意图

（3）相向冲突

相向冲突是指两辆AGV小车在同一路段相向行驶，产生相向碰撞而引起的冲突，超越冲突示意图如图3-3所示。

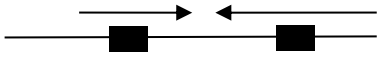

图 3-3　相向冲突示意图

在多AGV的系统中，针对工作任务调度安排的迥异和行驶区域环境的不同，AGV行驶过程中遇到冲突的情况就会多样化，但基本上是由以上这三种最基本的冲突构成。AGV小车间冲突问题发生的实质都是时间和空空间重和，即在同一时间两辆车在统一位置内相遇并发生碰撞。要解决多AGV小车行驶过程中的冲突碰撞问题，首先需要解决好以上三种最基本的冲突，从而确保AGV系统能够安全有效的运行。

3.1.3　AGV 冲突解决方法

AGV冲突解决方法主要有路径铺设法、区域控制法、预测式避碰方法和反应式避碰方法[2]。

（1）路径铺设法主要通过设置主副车道，再加以如果干规则来进行，算法简单，但不具备通用性。

（2）区域控制法通过将工作环境分成互不重叠覆盖的几个区域，且每个区域只能允许一台 AGV 运行的方法来实现避碰，这样的系统明显增加了运输过程的中间环节，

效率较低。

（3）预测式避碰方法首先对各个 AGV 进行路径规划，然后根据每台 AGV 的路径预测 AGV 之间是否会发生冲突，如果会发生，则调整 AGV 的路径来避免碰撞。预测式避碰通过合适的冲突检测和冲突解决方法，可以对系统做很好的优化，一直是该问题的主要研究方向之一。

（4）反应式避碰方法与预测式避碰方法大不相同，AGV 的运行路径不是事先规划好的，而是在运行过程中根据系统当前的状态一段一段的确定 AGV 的运行路径，反应式避碰方法能迅速响应系统环境变化，具有很好的发展前景。

本文研究采用的基于时间窗的多 AGV 路径规划方法属于预测式防冲突控制方法。多 AGV 路径规划是一个要在保证系统无冲突的情况下，为每个 AGV 规划出一条无碰撞且最优的路径，因此在路径规划问题中的优化也属于多目标优化类型。

3.2 多 AGV 系统路径规划

在 AGV 系统离线路径规划时就考虑尽量避免碰撞和产生资源上的冲突，采用基于先验决策的 AGV 无碰撞路径规划方法。把改进的 Dijkstra 算法和时间窗原理相结合，顺序规划各个 AGV 路径[20]。在已规划 AGV 路径的基础上运用基于 Dijkstra 的算法继续顺序规划各个 AGV 路径，实现无碰撞路径规划。

3.2.1 多 AGV 系统路径规划环境

环境是基于图论表示，如图 3-4，采用有向图 G=(N,A), 其中 $N=\{\mathrm{n}_1,\mathrm{n}_2,\mathrm{n}_3,\cdots,\mathrm{n}_\mathrm{n}\}$ 表示一系列节点, $A=\{a_1,a_2,a_3,\cdots,a_m\}$ 表示一系列有权重的单元弧段，节点表示上下工位点，边权重代表相邻两点间有效路径的长度，各节点的拓扑关系如表 3-1 所示。我们做如下规定：

1） 每个边代表双向的单行道，AGV 可以前后双向行驶，即 AGV 不必旋转 180° 朝相反方向行驶；

2） 小车行驶的时候速度保持恒定, 这里设为 v=0.5m/s；

3） 小车的负载为容量为 1（任务），即接到任务指令后，当且完成此任务后才能接受下一指令，执行下一个任务；

4） 小车在同一时间只能占用一个单元路段；

5） 为了保障 AGV 的安全，这里我们定义一个最小安全车距 Ls ，由 AGV 的车身长度和速度确定，即在（车身长度+2 倍安全车距）形成的方形区域内，只允许有一辆 AGV。

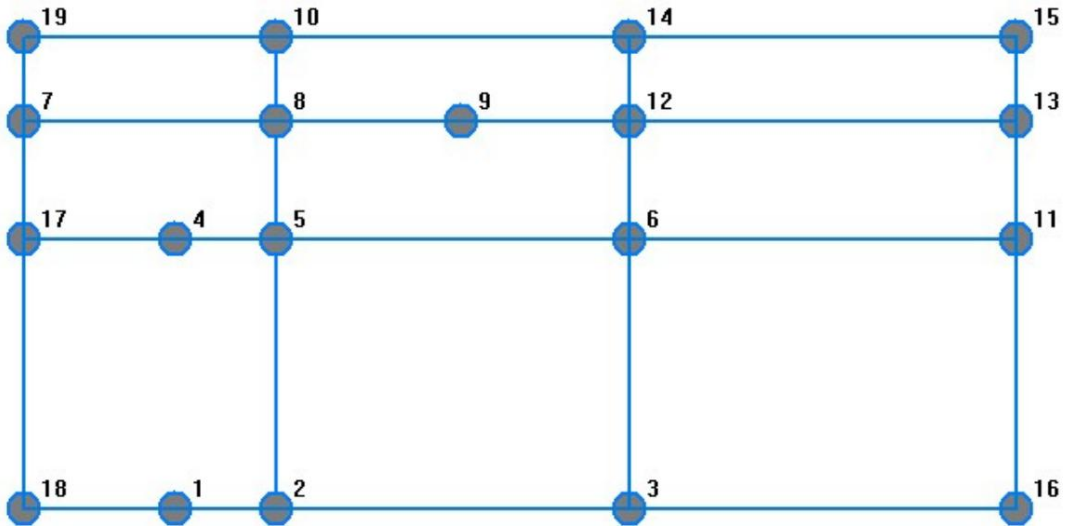

图 3-4 多 AGV 调度环境模型

表 3-1 各节点拓扑关系

节点 ni	节点 nj	权重		节点 ni	节点 nj	权重
1	2	6		7	8	15
1	18	7		7	19	7
2	3	21		8	10	5
2	5	16		8	12	21
3	6	16		8	9	10
3	16	20		9	12	11
4	5	6		10	14	21
4	17	7		11	13	7
5	6	21		11	16	16
5	8	7		12	13	20
6	12	7		12	14	5
6	11	20		13	15	5

3.2.2 AGV 系统路径规划算法

假设有 k 辆 AGV 小车需要规划路径，每个小车已经分配好任务，即给出起始点和终点。我们先离线根据 Dijkstra 算法对每个 AGV 的规划出最短路径。根据每个任务完成时间的长短，对 k 个小车的任务从大到小排序，m 1 >m 2 >m 3 >m 4 >m 5 >m 6 …m k 。首先对 AGV1 采用 Dijkstra 算法进行无碰撞的路径规划，根据 AGV1 进入和离开各个节点的时间，得出 AGV1 的时间窗，在保证和 AGV1 无碰撞的前提下，规划花费时间仅次于 AGV1 的 AGV2，即在 AGV1 的空闲时间窗上规划 AGV2。即得 AGV2 的无碰撞规划路径。同理，AGV3 的规划是在保证与 AGV1、AGV2 无碰撞的前提下规划最短路径，在规划第 k 条路径时，在保证不与前 k-1 辆小车碰撞前提下，规划最短路径，重复执行上述过程，直到规划完所有 AGV。

在所建环境模型的基础上，建立与其相对应的时间窗模型，如图 3-5 所示，引入变量：

r^{j}_{i} 是指第 i 个节点上的第 j 个被占用时间窗；

f^{n}_{m} 是指第 m 个节点的第 n 个空闲时间窗。

AGV1 和 AGV2 已经规划出最短路径，分别对应的路径为：AGV1 的起始点为节点 17,通过节点 4,节点 5，到目标终点点 6，相应节点的占用时间窗为 $r^{1}_{17}, r^{1}_{4}, r^{1}_{5}, r^{1}_{6}$ 。AGV2 从起始节点 1，节点 2 节点 5，节点 8，到达目标节点 10；相应的节点占用时间窗 $r^{1}_{1}, r^{1}_{2}, r^{2}_{5}, r^{1}_{8}$ 。对 AGV3，起始点在节点 9，目标点为节点 1，需要对 AGV3 规划一条不与 AGV1，AGV2 产生冲突的从起始节点 9 到目标节点 1 的无碰撞路径。相邻节点间的距离在表 3-1 中给出，单位为米。

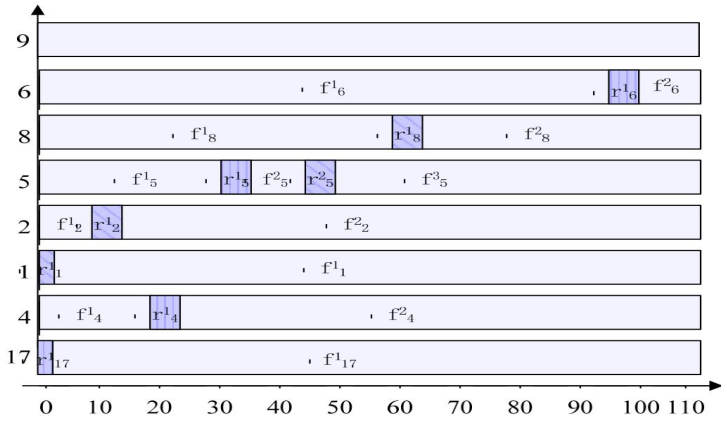

图 3-5 相对应的时间窗模型

3.3 时间窗模型

时间窗指 AGV 进入到离开该节点时间所组成的时间段，此时间段只能给该小车使用，其它小车在该车保留时间窗内不允许通过该节点[15]。所以当多 AGV 规划路径后，每个节点在任务开始到任务结束的这段时间内，被多个不同时间点经过该节点的 AGV 小车划分成各自相应的保留时间段[20]。每个节点保留时间段的中间间隔的空闲时间段则可用来规划其它的 AGV，其它 AGV 可以在空闲时间段内经过该节点。基于时间窗的多 AGV 路径规划是规划空闲时间窗，取代传统拓扑路径上的节点。所以算法的关键是考察各个节点的空闲时间窗是否具有可连接性。本文采用时间窗图论法，每个节点的空闲时间窗代表一个节点。节点的占用时间窗标记如下：

$$R = \{r_n^k = [c_n^k, d_n^k]\} \tag{3-1}$$

其中，r_n^k 是指在节点 n 的时间窗。

c_n^k 是指 AGV 进入节点的时间点，即占用时间窗的开始时间点。

d_n^k 是指 AGV 离开节点的时间点，即占用时间窗的结束时间点。

同理，定义节点空闲时间窗如下:

$$F = \{f_n^k = [a_n^k, b_n^k]\} \tag{3-2}$$

其中，f_n^k 表示在节点 n 的时间窗。

a_n^k 表示 AGV 进入节点的时间点，即空闲时间窗的开始时间点。

b_n^k 表示 AGV 离开节点的时间点，即空闲时间窗的结束时间点。

$f_n^k - a_n^k \geq t_n$ ， t_n 为 AGV 小车经过某节点的时间。

R 集合 F 集合组成节点的所有集合。

3.3.1 时间计算

时间窗的前提条件是保证的时间点的准确性，从而时间的计算是非常重要的。这里假设 AGV 直线路径匀速且速度是 V_s，转弯速度是 V_c，AGV 本身长度是 L_v，节点长度是 L_n。通过节点的时间可表示为 AGV 经过图 3-6 所示长度的时间，节点按照所处地点的差异可分为直线路径上的节点和转弯路口处的节点，分别如图 3-7 和图

3-8 所示。

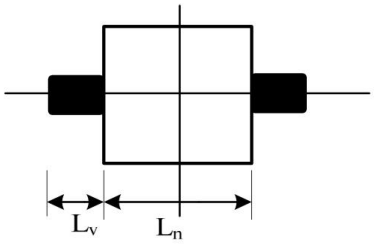

图 3-6　通过直线路径节点

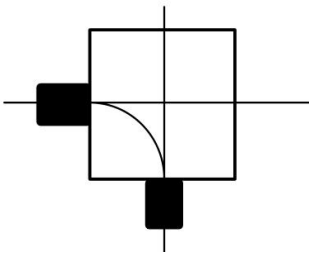

图 3-7　通过拐弯路径节点

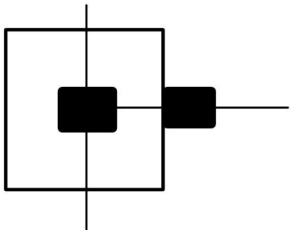

图 3-8　通过起始点路径节点

图 3-6 中通过直线路径上的节点所用时间为t_n 通过如下公式计算：

$$t_n = \frac{L_v + L_n}{V_S} \tag{3-3}$$

图 3-7 中为通过交叉路口 AGV 采取拐弯路线示意图，经过此节点的时间记为t_n

$$t_n = \frac{L_V + \frac{\pi L_n}{2}}{V_c} \tag{3-4}$$

图 3-8 为 AGV 通过起始点的示意图，经过起始点的时间为t_n

$$t_n = \frac{\frac{L_v}{2} + \frac{L_n}{2}}{V_s} \tag{3-5}$$

上面给出了 AGV 通过不同位置节点所用的时间，接下来给出 AGV 通过直线路段所花费的时间，如图 3-9 所示，设 AGV 通过 ij 两节点之间的路段，路段长度为 L(i,j)，通过路段[i,j] 所用时间为 t_{ij}, t_{ij} 的计算公式如下：

$$t_{ij} = \frac{L_{(i,j)} - L_v - L_n}{V_s} \tag{3-6}$$

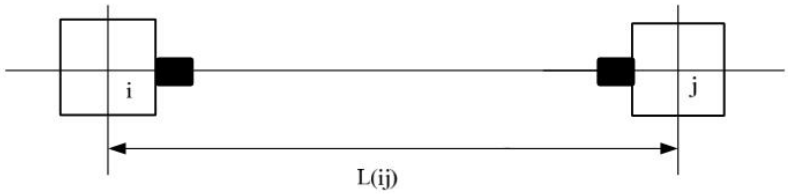

图 3-9　节点 ij 间直线路段示意图

3.3.2　时间窗计算

考察前后相邻节点的时间窗的连续性 ，包括空间连续性和时间连续性，这里假设考察节点 i 和节点 j。

首先考虑空间上是否连续，其次考察时间上是否连续，即 AGV 必须保证在该节点的某一空闲时间段内进入和离开。这也是本算法的重点，需要定义几个参数。

AGV 在 f_j^q 空闲时间段内到达 j 点的最早时间定义为 $T_a(f_j^q, f_i^p)$，具体示意如图 3-10 所示。

$$T_a(f_j^q, f_i^p) = L(f_i^p) + t_{ij} + t_i \tag{3-7}$$

$T_a(f_j^q, f_i^p)$ 表示从 i 点到达 j 点的时间

$L(f_i^p)$ 表示能在 f_i^p 空闲时间窗内到达 i 点的时间

t_{ij} 表示 i,j 点之间的弧段的行走时间；

AGV 在时间窗 f_j^q 内能够进入点 j 定义为 T_e，

$$T_e = \max\{T_a, a_j^q\} \tag{3-8}$$

其中：T_e 表示基于 f_j^q 从 i 点出发能进入到 j 点的时间。

a_j^q 表示为 $f_j^q = [a_j^q, b_j^q]$]中的 a_j^q。

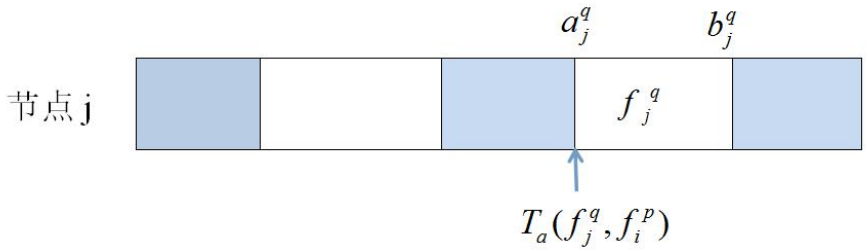

图 3-10　$T_a(f_j^q, f_i^p)$ 示意图

3.4　多 AGV 系统无碰撞路径规划算法过程

这里定义 3 个集合，由前 i-1 辆已规划 AGV 得到的时间窗集合，设为 F 集；计算过的符合无碰撞条件的空闲时间窗为 T 集合；U 集合表示在剩下 F-T 集合中，距离 T 集合中元素最近的时间窗。P 集合中存放的是相应节点的前继节点，例如 $P(f_2^1)=f_1^0$ ，表示节点 2 的前置节点为节点 1。

1)　初始化，小车起始点所占的时间窗设为 f_s^0，$L(f_s^0)$ 取值为 t_n ，初始时 T 集合为 f_s^0 ，P 集合为空；

2)　对每一个 $f_j^q \in F-T$；

(a) 对 $f_j^q \in T$，分别计算从 i 点 到 j 点的 T_a 和 T_e ；并考察 ij 两点空间、时间及有无潜在碰撞的可能性；

(b) 对集合 T 中的时间窗 f_k^h ，令 $T_{\min}=\min\{T_e\}$；

(c) 对于 f_j^q ，$L(f_j^q)>T\min$, 则 $L(f_j^k)=T_{\min}$ ，$P(f_j^q)=f_i^p$ ；

(d) f_j^q 加入集合 U；

3)　在集合 U 中，取 U 集合中的最小值，加入到 T 集合；

4)　检验目标节点是否在 T 集合中，如果目标节点在 T 集合中，继续往下执行；否则返回执行第二步；

5) 整理路径，由最后目标节点开始，依据 $P(f_j^q)=f_i^p$ 依次推出前置节点，直到起始节点。

在本文的环境模型中，在已规划两辆 AGV 无碰撞路径的前提下，运用多 AGV 系

统无碰撞路径规划算法规划第 3 辆 AGV。下面例举节点 8 的算法过程。

（1）初始化

AGV3 起始节点为节点 9，$r_9^1 = f_9^0, L(f_9^0) = 0, T = \{f_9^0\}, U = \varnothing;$

（2）对 F-T 集合中的元素，考察空间连续性和时间连续性

对于 f_8^1：

1）$T_a(f_8^1, f_9^0) = L(f_9^0) + t_{98} + t_9 = 24; T_e = \max\{T_a, a_j^q\} = \max\{24, a_8^1\}$

2）$T_{\min} = \min\{T_e\} = 24$

3）$L(f_8^1) = T_{\min} = 24, P(f_8^1) = f_9^0$

4）$U = \{f_8^1\}$

其余节点算法过程主要计算数据以表格方式列于表 3-2

表 3-2 无碰撞规划算法实例数据

循环	L	T	P
1	0	f^0_9	f^0_9
2	$L(f^1_8)= L(f^0_9)+t_{98}=24$ $L(f^2_8)= 64$	$f^1_8 f^0_9$	$P(f^1_8)= f^0_9$
3	$L(f^1_5) = \infty$ $L(f^2_5)=38$ $L(f^3_5)=48$	$f^2_5 f^1_8 f^0_9$	$P(f^2_5)=f^1_8$
4	$L(f^2_4)=54$ $L(f^1_6)=84$ $L(f^3_5)=48$ $L(f^2_2)=70$	$f^3_5 f^2_5 f^1_8 f^0_{93}$	$P(f^3_5)=f^2_5$
5	$L(f^2_4)=54$ $L(f^1_6)= 84$ $L(f^2_2)=70$	$f^2_4 f^3_5 f^2_5 f^1_8 f^0_9$	$P(f^4_5)=f^3_5$
6	$L(f^2_2)=48+32=80$	$f^2_2 f^2_4 f^3_5 f^2_5 f^1_8 f^0_9$	$P(f^2_2)=f^3_5$
7	$L(f^1_1)= 80+12=92$	$f^1_1 f^2_2 f^2_4 f^3_5 f^2_5 f^1_8 f^0_9$	$P(f^1_1)= f^2_2$

根据 P 集合中的前后关系，从目标终结点依次往前递推得出 AGV3 的行使顺序：$f^0_9\ f^1_8\ f^2_5 f^3_5\ f^2_2\ f^1_1$，图 3-11 为对应的时间窗。

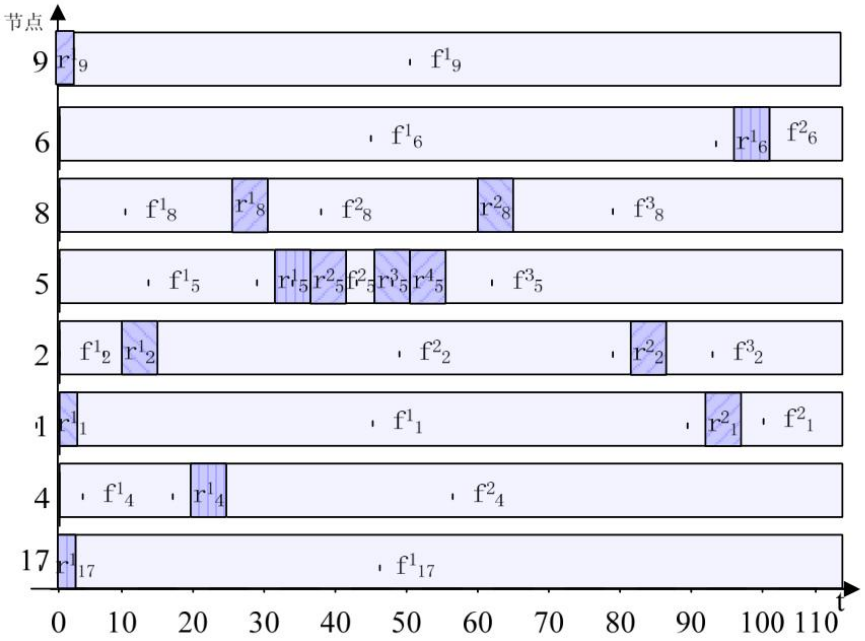

图 3-11 AGV3 对应的时间窗图

3.5 本章小结

本章针对多 AGV 系统运行过程中出现的碰撞冲突问题，提出一种基于时间窗原理的车辆路径规划方法，解决多 AGV 小车行驶过程中出现的冲突问题。文中介绍了 AGV 系统中出现的冲突类型以及时间窗原理、时间窗模型和时间计算。最后通过实例计算验证时间窗算法的可行性，利用时间窗原理算法，在离线情况下为 3 台 AGV 小车规划处一条无碰撞、无冲突的可行路径。本章中的路径规划是在离线情况下完成的，必须确保每台小车在规定的时间完成搬运任务。

第 4 章　总结与展望

4.1　毕设工作总结

随着物联网与智能物流系统，信息化管理系统，柔性制造系统(FMS)等的发展，自动引导车系统（AGVs）的应用日益广泛，AGV 系统在工业生产中的应用可以解放劳动生产力，提高工作效率，降低生产运输成本。未来在各行各业 AGV 必将保持较快的发展趋势。如何控制优化 AGV 小车的搬运系统已成为研究 AGV 的一大热点，其中路径规划技术作为 AGV 系统的核心技术之一，更是当下研究 AGV 系统技术的潮流。

本文的研究内容是基于 AGV 小车搬运系统优化控制研究，主要针对 AGV 系统路径规划方面，如何规划一条有效可行的最优路径。现将本文的毕设工作总结如下：

1）在查阅有关 AGV 系统研究文献的基础上，介绍课题研究背景与意义，以及 AGV 小车研究的国内外现状，介绍了 AGV 系统的导航定位、路径规划等关键技术。

2）描述 AGV 路径规划问题，对地图环境的建模做出合理的假设，对于单个 AGV 小车的路径规划，利用基于栅格法的深度优先搜索算法和基于拓扑图论法的 Dijkstra 算法来实现对 AGV 运行地图环境建模和路径规划，并在 Visual C++ 6.0 环境平台进行仿真上进行实验的仿真验证。

3）介绍了多 AGV 小车系统运行存在的问题，对于多 AGV 系统的路径规划方面，主要研究的是怎样为每台 AGV 规划一条无碰撞、无冲突的路径。介绍了在单 AGV 路径规划的基础上，提出利用 Dijkstra 算法和时间窗原理相结合的方法，实现多 AGV 无碰撞路径规划，并通过实例计算验证该方法的可行性。

4.2　未来展望

随着毕业设计课题的深入研究，越来越深刻地意识到本文研究内容的不足。由于本人的时间及能力有限，目前的课题内容研究还处于初级阶段，还有许多需要改进的地方：

（1）本文运用的 Dijkstra 算法和深度优先搜索法，虽然能保证找到最优解，但其搜索效率不高，只能适用于较简单的 AGV 小车路径规划。

（2）本文自能对于单个的 AGV 路径寻优进行仿真验证，没能实现多 AGV 系统中仿真模拟。

参 考 文 献

[1] 王喜富. 物联网与智能物流[M]. 清华大学出版社, 2014.

[2] 孙奇. AGV 系统路径规划技术研究[D]. 浙江大学, 2012.

[3] 张辰贝西，黄志球. 自动导航车(AGV)发展综述[J]. 机械设计与制造工程，2010, 39(1):53-59.

[4] 曹智荀，王宝华. 激光导引叉车在国内的应用[J]. 物流技术与应用，2006. 07：98-99.

[5] 孟文俊，刘忠强. 视觉导引 AGV 的路径跟踪控制研究[J]. 控制工程，2014,21（3）：321-325.

[6] 樊征，曹其新，杨扬,等. 面向移动机器人的拓扑地图自动生成[J]. 华中科技大学学报(自然科学版), 2008(s1):172-175.

[7] Kalinovcic, L, Petrovic, T, Bogdan. Modified Banker's algorithm for scheduling in multi-AGV systems[C]// Automation Science and Engineering. IEEE, 2011:351-356.

[8] Vivaldini, K.C.T., et al.: Automatic Routing System for Intelligent Warehouses[C]. In: IEEE Int. Conference on Robotics and Automation, pp. 93–98 (2010).

[9] Vivaldini K C T, Tamashiro G, Junior J M, et al. Communication Infrastructure in the Centralized Management System for Intelligent Warehouses[J]. Communications in Computer & Information Science, 2013, 371(1):127-136.

[10] 齐东流. 基于智能控制的 AGV 路径规划技术研究[D]. 合肥工业大学硕士论文，2006.

[11] 李乐军,施业琼,韦宝秀.关于 AGV 及其在中国的应用发展探析[J].科技自 寻.2007，NO，34.

[12] 李进，陈无畏等. 自动导引车视觉导航的路径识别与跟踪控制[J]. 农业机械学报. 2008，2，v01 39：20—24.

[13] Smolic-Rocak N, Bogdan S, Kovacic Z, et al. Time Windows Based Dynamic Routing in Multi-AGV Systems[J]. IEEE Transactions on Automation Science & Engineering, 2010, 7(1):151-155.

[14] Ullrich C A. Integrated machine scheduling and vehicle routing with time windows[J]. European Journal of Operational Research, 2011, 227(1):152-165.

[15]贺丽娜. AGV 系统运行路径优化技术研究[D]. 南京航天航空大学硕士论文，2011.

[16]冯海双. AGV 自动运输系统调度及路径规划的研究[D]. 哈尔滨工业大学硕士论文，2013.

[17]曹智荀，王宝华. 激光导引叉车在国内的应用[J]. 物流技术与应用，2006. 07：98-99.

[18]Ling Qiu, WenJing Hsu, ShellYing Huang, et al. Scheduling and routing algorithms for AGVs: A survey[J]. International Journal of Production Research, 2002, 40(3):745-760.

[19]王光武. Dijkstra 最短路径算法分析与改进[J]. 工业控制计算机, 2011, 24(10):63-63.

[20]张峥炜，陈波，陈卫东. 时间窗约束下的 AGV 动态路径规划[J]. 微型电脑应用，2016, 32(11):46-49.

附录 1：基于深度优先搜索路径规划代码

```
#include <graphics.h>
#include <stdio.h>
#include <memory.h>
#include <queue>
using namespace std;

const int w= 24, h = 24;

struct Block
{
	BYTE x,y,depth,data[200];
	 bool searched,isblock;

	//获取地图数据信息函数
	void getdata( Block father,int dx,int dy)
	{
		depth=father.depth+1;
		memcpy( data,father.data,( depth-1)=sizeof( BYTE)=2);
		data[2=depth-2]=father.x+dx;
		data[2=depth-1]=father.y+dy;
	 }

	 //清除地图信息函数
	void clean()
	{
		memset(data,0,sizeof(char)=200);
		searched=false;
		depth=0;
	}

}map[w][h];

MOUSEMSG msg;
bool success=false;
int sx=0,sy=0,ex=w-1,ey=h-1,mousemode=1;
queue<Block> q;

 //地图路径搜索函数
void map_search(int x,int y)
{
```

```
    if(tx>=0&&ty>=0&&tx<w&&ty<h&&!map[tx][ty].searched)
    {
        map[tx][ty].searched=true;
        if(map[tx][ty].isblock)
            return;
        if(tx==ex&&ty==ey)
        {
            success=true;
            int i;
            char jieguo[5000]="",text[100];
            for(i=0;i<map[ex][ey].depth;i++)
            {
                char temp[20];
                if(i!=map[ex][ey].depth-1)
                    sprintf(temp,"(%d,%d)->",map[ex][ey].data[i=2],map[ex][ey].data[i=2+1]);
                else
                    sprintf(temp,"(%d,%d)",map[ex][ey].data[i=2],map[ex][ey].data[i=2+1]);

                strcat(jieguo,temp);

                if(i!=map[ex][ey].depth-1)
                {
                    setfillcolor(RED);

fillrectangle(map[ex][ey].data[i=2]=20,map[ex][ey].data[i=2+1]=20,map[ex][ey].data[i=2]=20+20,
                    map[ex][ey].data[i=2+1]=20+20);
                }
            }
            sprintf(text,"成功,共%d 步",map[ex][ey].depth);
            MessageBox(GetHWnd(),jieguo,text,0);
        }
        if(tx+1<w)
        {
            map[tx+1][ty].getdata(map[tx][ty],1,0);
            q.push(map[tx+1][ty]);
        }
        if(ty+1<h)
        {
            map[tx][ty+1].getdata(map[tx][ty],0,1);
            q.push(map[tx][ty+1]);
        }
        if(tx>0)
        {
            map[tx-1][ty].getdata(map[tx][ty],-1,0);
```

```
                q.push(map[tx-1][ty]);
            }
            if(ty>0)
            {
                map[tx][ty-1].getdata(map[tx][ty],0,-1);
                q.push(map[tx][ty-1]);
            }
        }
}

void mapclean()
{
    int i,j;
    for(i=0;i<w;i++)
        for(j=0;j<h;j++)
        {
            map[i][j].clean();
            map[i][j].isblock=false;
            map[i][j].x=i;
            map[i][j].y=j;
        }
    sx=0;
    sy=0;
    ex=w-1;
    ey=h-1;
    success=false;
}
 //执行寻路函数
void runxunlu()
{
    int i,j;
    for(i=0;i<w;i++)
        for(j=0;j<h;j++)
        {
            map[i][j].clean();
            map[i][j].x=i;
            map[i][j].y=j;
        }
    success=false;

    q.push(map[sx][sy]);

    while(!q.empty()&&!success)
    {
```

```
            map_search(q.front().x,q.front().y);
            q.pop();
        }

        if(!success)
            MessageBox(GetHWnd(),"很遗憾,寻路失败","提示",0);

        while(!q.empty())
            q.pop();
}

IMAGE buffer(w=20+160,h=20);
//界面按钮设计函数
void render()
{
        SetWorkingmage(&buffer);

        cleardevice();

        setbkmode(TRANSPARENT);

        setfillcolor(GREEN);
        fillrectangle(w=20,0,w=20+160,h=20);

        setfillcolor(RED);

        fillrectangle(w=20,0,w=20+160,30);
        outtextxy(w=20+10,10,"设起点");

        fillrectangle(w=20,60,w=20+160,90);
        outtextxy(w=20+10,70,"设终点");

        fillrectangle(w=20,120,w=20+160,150);
        outtextxy(w=20+10,130,"设工位");

        fillrectangle(w=20,180,w=20+160,210);
        outtextxy(w=20+10,190,"开始寻路");

        fillrectangle(w=20,240,w=20+160,270);
        outtextxy(w=20+10,250,"地图重置");

        setfillcolor(BLUE);
        fillrectangle(sx=20,sy=20,sx=20+20,sy=20+20);
```

```
    setfillcolor(BROWN);
    fillrectangle(ex=20,ey=20,ex=20+20,ey=20+20);

    setfillcolor(YELLOW);
    int i,j;
    for(i=0;i<w;i++)
        for(j=0;j<h;j++)
        {
            if(map[i][J].isblock)
                fillrectangle(i=20,j=20,i=20+20,j=20+20);
        }
    SetWorkingImage();
    putimage(0,0,&buffer);

}
int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)
{
    initgraph(w=20+160,h=20);

    while(true)
    {
        while(MouseHit())
        {
            msg=GetMouseMsg();
            if(msg.uMsg==WM_LBUTTONDOWN)
            {
                if(msg.x>=w=20)
                {
                    if(msg.y>=0&&msg.y<=30)
                        mousemode=1;

                    else if(msg.y>=60&&msg.y<=90)
                        mousemode=2;

                    else if(msg.y>=120&&msg.y<=150)
                        mousemode=3;

                    else if(msg.y>=180&&msg.y<=210)
                        runxunlu();

                    else if(msg.y>=240&&msg.y<=270)
                        mapclean();
                }
                else
```

```
                {
                    switch(mousemode)
                    {
                        case 1:
                            sx=msg.x/20;
                            sy=msg.y/20;
                            break;
                        case 2:
                            ex=msg.x/20;
                            ey=msg.y/20;
                            break;
                        case 3:
                            map[msg.x/20][msg.y/20].isblock=!map[msg.x/20][msg.y/20].isblock;
                            break;
                     }
                }
            }
        }
        render();
    }
    return 0;
}
```

附录 2：基于 Dijkstra 算法的路径规划代码

```
#include <iostream>
#include <limits>
using namespace std;

struct Node {     //定义表结点
   int adjvex;     //该边所指向的节点的位置
   int weight;     //  边的权值
   Node =next;    //下一条边的指针
};

struct HeadNode{     //  定义头结点
      int nodeName;    //  节点信息
      int inDegree;     //  入度
      int d;                 //表示当前情况下起始节点至该节点的最短路径,初始化为无穷大
      bool isKnown;    //表示起始节点至该节点的最短路径是否已知,true 表示已知，false 表示未知
      int parent;         //表示最短路径的上一个节点
      Node =link;       //指向第一条依附该节点的边的指针
};

//G 表示指向头结点数组的第一个结点的指针
//nodeNum 表示结点个数
//arcNum 表示边的个数
void createGraph(HeadNode =G, int nodeNum, int arcNum) {
   cout << "开始创建图(" << nodeNum << ", " << arcNum    << ")" << endl;
   //初始化头结点
   for (int i = 0; i < nodeNum; i++) {
      G[i].nodeName = i+1;     //位置 0 上面存储的是结点 v1,依次类推
      G[i].inDegree = 0;          //入度为 0
      G[i].link = NULL;
   }
   for (int j = 0; j < arcNum; j++) {
      int begin, end, weight;
      cout << "请依次输入  起始边  结束边  权值: ";
      cin >> begin >> end >> weight;
      //  创建新的结点插入链接表
      Node =node = new Node;
      node->adjvex = end - 1;
      node->weight = weight;
      ++G[end-1].inDegree;       //入度加 1
      //插入链接表的第一个位置
```

```
        node->next = G[begin-1].link;
        G[begin-1].link = node;
    }
}

void printGraph(HeadNode =G, int nodeNum) {
    for (int i = 0; i < nodeNum; i++) {
        cout << "结点 v" << G[i].nodeName << "的入度为";
        cout << G[i].inDegree << ",  以它为起始节点的边为: ";
        Node =node = G[i].link;
        while (node != NULL) {
            cout << "v" << G[node->adjvex].nodeName << "(权:" << node->weight << ")" << "    ";
            node = node->next;
        }
        cout << endl;
    }
}

//得到 begin->end 权重
int getWeight(HeadNode =G, int begin, int end) {
    Node =node = G[begin-1].link;
    while (node) {
        if (node->adjvex == end - 1) {
            return node->weight;
        }
        node = node->next;
    }
}

//从 start 开始，计算改节点到每一个节点的最短路径
void Dijkstra(HeadNode =G, int nodeNum, int start) {
    //初始化所有结点
    for (int i = 0; i < nodeNum; i++) {
        G[i].d = INT_MAX;       //到每一个节点的距离初始化为无穷大
        G[i].isKnown = false;     //  到每一个节点的距离为未知数
    }
    G[start-1].d = 0;               //到其本身的距离为 0
    G[start-1].parent = -1;       //表示该结点是起始结点
    while(true) {
     //====  如果所有的结点的最短距离都已知, 那么就跳出循环
     int k;
     bool ok = true;                          //表示是否全部 ok
     for (k = 0; k < nodeNum; k++) {     //只要有一个节点的最短路径未知,ok 就设置为 false
         if (!G[k].isKnown) {
```

```
            ok = false;
            break;
        }
    }
    if (ok) return;
    //=========================================

    //==== 搜索未知结点中 d 最小的,将其变为 known
    //==== 这里其实可以用最小堆来实现
    int i;
    int minIndex = -1;
    for (i = 0; i < nodeNum; i++) {
        if (!G[i].isKnown) {
            if (minIndex == -1)
                minIndex = i;
            else if (G[minIndex].d > G[i].d)
                minIndex = i;
        }
    }
    //=========================================

    cout << "当前选中的结点为: v" << (minIndex+1) << endl;
        G[minIndex].isKnown = true;              //将其加入最短路径已知的节点集
        // 将以 minIndex 为起始节点的所有的 d 更新
        Node =node = G[minIndex].link;
        while (node != NULL) {
            int begin = minIndex + 1;
            int end = node->adjvex + 1;
            int weight = getWeight(G, begin, end);
            if (G[minIndex].d + weight < G[end-1].d) {
                G[end-1].d = G[minIndex].d + weight;
                G[end-1].parent = minIndex;              //记录最短路径的上一个结点
            }
            node = node->next;
        }
    }
}

//打印到 end-1 的最短路径
void printPath(HeadNode =G, int end) {
    if (G[end-1].parent == -1) {
        cout << "v" << end;
    } else if (end != 0) {
        printPath(G, G[end-1].parent + 1);   // 因为这里的 parent 表示的是下标，从 0 开始，所以要加 1
```

```
        cout << " -> v" << end;
    }
}
int main() {
    HeadNode =G;
    int nodeNum, arcNum;
    cout << "请输入节点个数，边长个数: ";
    cin >> nodeNum >> arcNum;
    G = new HeadNode[nodeNum];
    createGraph(G, nodeNum, arcNum);

    cout << "================================" << endl;
    cout << "下面开始打印图信息..." << endl;
    printGraph(G, nodeNum);

    cout << "================================" << endl;
    cout << "下面开始运行 dijkstra 算法..." << endl;
    Dijkstra(G, nodeNum, 1);

    cout << "================================" << endl;
    cout << "打印从 v1 开始所有的最短路径" << endl;
    for (int k = 2; k <= nodeNum; k++) {
        cout << "v1 到 v" << k << "的最短路径为" << G[k-1].d << ": ";
        printPath(G, k);
        cout << endl;
    }
}
```

致 谢

在这个栀子花开的季节，四年的大学本科学习生活即将结束，在这期间我从一个懵懂无知的高中生成长为一个能勇敢走入社会，直面各种困难的应届毕业生，这都多亏了有老师和同学对我的帮助，让我能在遇到困难和挫折的时候，能一一克服这些障碍。在这里正好借这个机会对所有帮助过我的老师和同学表示深深的感谢。

首先，我要特别感谢我的母校浙江工业大学，是她提供了优越的学习环境，让我们能有更多的资源进行学习，去追求我们的梦想。

其次，我要感谢我的导师赵云波教授。赵老师治学态度严谨，工作认真负责，待人和蔼可亲。在整个毕业设计的过程中，赵老师给予了我很多建设性的意见，悉心地指导我完成了论文的研究方向选择，研究方法的确定以及各项研究工作。借此机会，我要再一次向赵老师表达我的感谢之意，没有赵老师的指导，我无法顺利完成毕业设计工作及毕业论文的撰写。

然后，我要感谢在毕设期间帮助过我的所有学长学姐。在做毕业设计的过程中，他们给予了我很多宝贵的建议，让我在这条研究的道路上少走了很多弯路。当我有什么不懂的地方去请教学长学姐的时候，他总能很耐心的向我讲解。在此，向学长学姐们表示深深的谢意。

最后，我要感谢我的父母，是你们给了我学习的机会，是你们在本就不怎么宽裕的情况下给了我无忧无虑的生活，让我能够专心完成自己的学业，也是你们给了我精神上的支持，让我能够有足够的勇气来面对生活中的困难。你们是我坚强的后盾，你们的鼓励是我前进的动力。

大学的这四年，我有过欢声和笑语，也有过迷茫和沮丧，但重要的是我学会了如何调整自己的心态，积极自信地去面对一切。未来将有更多的困难在等着我，希望自己能够始终保持这种心态，去迎接人生中的挑战！