

北京交通大学

硕士专业学位论文

考虑节点失效风险的物流网络选址模型及算法研究

Models and Algorithms for Logistics Network Location Problem
Considering Node Disruption Risk

作者：余润峰

导师：员丽芬

北京交通大学

2023 年 5 月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

(保密的学位论文在解密后适用本授权说明)

学位论文作者签名:

导师签名:

签字日期: 年 月 日 签字日期: 年 月 日

北京交通大学

硕士专业学位论文

考虑节点失效风险的物流网络选址模型及算法研究

Models and Algorithms for Logistics Network Location Problem
Considering Node Disruption Risk

作者姓名: 余润峰

学 号: 21125806

导师姓名: 员丽芬

职 称: 副教授

专业学位类别 (领域) : 交通运输

学位级别: 硕士

北京交通大学

2023 年 5 月

致谢

突然想起两年前研究生刚入学的日子，感慨时光飞逝，不知不觉间在北京交通大学已经六年。人生能有多少六年？小学六年，中学六年，高等教育六年……

这篇文章作为刚刚过去六年的收尾，向过去的自己告别，迎接未知的世界。在此，我衷心感谢我的导师，北京交通大学交通运输学院物流系员丽芬副教授。感谢员老师两年来对我学业的指导和生活的关心，一直督促我前进。同时，感谢北京邮电大学现代邮政学院范宏强老师在算法方面对我的指导。两年来，与范老师的交流总能让我获得更多灵感，使我对算法有了充分的了解。

此外，感谢父母、女友、女友家人、以及亲人、挚友对我学业的无限支持和理解，感谢中国人民解放军国防大学联合勤务学院黄剑炜教授对我的学业和未来发展的关心，感谢北京交通大学交通运输学院物流系各位老师六年来的传道授业。

感谢下列对本文提供帮助的组织和个人：员老师和范老师提供了创新的想法，北京交通大学计算中心提供 Matlab 正版计算平台，Gurobi 中国代理刃之砺公司提供 Gurobi 求解器学术许可。

最后，感谢同一个师门的各位同学，她们是：孙逸宸、张钰儒、刘妍希、金珉宇以及各位师弟师妹。感谢大家在这个师门营造了其乐融融的氛围，即将分别之时，衷心祝福各位前途似锦，一路繁花。

真正的生活并不是活过的日子，而是那些被我们记住的片段。感谢上述各位不求回报的支持与帮助，我会将这些美好珍藏心中，山水有相逢，望君多珍重。

摘要

物畅其流则百业兴，我国正在加速构建以国内大循环为主体，国内国际双循环相互促进的新发展格局。打造安全可靠、高效便捷的现代物流网络系统迫在眉睫。物流节点作为物流网络的重要支撑，其科学布局、稳定运行是制约网络可靠性的重要关键因素。因此，在物流网络设计之初将物流节点失效的可能性考虑在内具有现实意义：不仅可以提升物流网络的可靠性以应对愈加频繁的自然灾害和人为事故，亦可以降低运营成本并增加网络运营效率以促进物流行业高质量发展。

本文的研究目的是强化物流网络的可靠性，降低节点失效时因临时决策产生的高昂成本。综合考虑节点失效风险对物流网络的影响，针对可能存在的节点失效状况以及应对措施，为物流网络选址问题构建数学优化模型，目标是寻找期望总成本最小的物流节点建设位置和客户分配方案。本文提出线性化方法将非线性模型转换为线性模型，并对模型进行一系列的性质分析。

由于考虑节点失效风险的物流网络选址问题是 NP-hard 问题，为求解该问题的大规模实例，本文开发了基于迭代局部搜索改进的拉格朗日松弛启发式算法。其中，松弛问题的最优值提供原问题的下界，启发式方法获得原问题的上界，局部迭代搜索可进一步强化上界。

数值实验证明了上述算法求解大规模问题的性能。实验结果表明，拉格朗日松弛算法是求解该选址问题的有效方法，其收敛性强、下界质量高，受算例参数的影响较小。本文应用上述模型和算法得到可靠物流网络的拓扑结构。参数灵敏度结果揭示了相关经济规律，表明节点的失效概率是影响网络可靠性相对重要的因素，而仅依靠增加每个客户备用节点数量对于提升网络可靠性的效果有限。

关键词：物流节点选址；选址问题；拉格朗日松弛；迭代局部搜索

ABSTRACT

Logistics nodes are important support for logistics networks, and their scientific layout and stable operation are key factors that restrict network reliability. Therefore, it is practical to consider the possibility of logistics node disruption at the beginning of logistics network design. This can not only improve the reliability of logistics networks to cope with increasingly frequent natural disasters and human accidents, but also reduce operating costs and increase network operational efficiency to promote high-quality development of the logistics industry.

The purpose of this study is to strengthen the reliability of logistics networks and reduce the high costs of temporary decision-making when nodes fail. Taking into account the impact of node disruption risk on logistics networks, this article constructs a mathematical model for this problem, focusing on node disruption conditions and corresponding countermeasures. The objective of the model is to find the location of logistics nodes with the minimum expected total cost and customer distribution plan. This paper proposes a linearization method and conducts a series of property analyses on the model.

Since the problem is NP-hard, in order to solve large-scale instances, this paper develops a Lagrangian relaxation heuristic with iterative local search operator. The optimal value of the relaxed problem provides a lower bound for the original problem, the heuristic method obtains an upper bound, and local iterative search can further enhance the upper bound.

This paper designs a series of numerical experiments to demonstrate the performance of the above algorithm in solving large-scale problems. The experimental results show that the Lagrangian relaxation is an effective method for solving such location problems, with strong convergence and high quality lower bounds, and is less affected by instance parameters. The topological structure of a reliable logistics network in that region is obtained. The sensitivity analysis results reveal relevant economic laws, which indicate that the probability of node disruption is a relatively important factor affecting network reliability, and that relying solely on increasing the number of backup nodes for each customer has limited effect on improving network reliability.

KEYWORDS: Logistics Node Location; Facility Location Problem; Lagrange Relaxation; Iterative Local Search

目录

摘要	v
ABSTRACT	vii
1 引言	1
1.1 研究背景和意义	1
1.1.1 研究背景	1
1.1.2 研究意义	3
1.1.3 研究创新点	4
1.2 研究路线	4
1.3 本章小结	6
2 文献综述	7
2.1 可靠选址理论综述	7
2.2 物流节点可靠选址理论综述	10
2.3 本章小结	12
3 考虑设施失效风险的物流网络选址模型构建	13
3.1 模型假设	13
3.2 问题描述和符号说明	14
3.3 模型构建	17
3.3.1 目标函数推导	17
3.3.2 数学模型	18
3.4 模型线性化	19
3.5 模型性质	20
3.5.1 NP-hard	20
3.5.2 子问题：试错序列问题	21
3.6 本章小结	23
4 基于迭代局部搜索改进的拉格朗日松弛算法设计	25
4.1 LR 算法原理	25
4.2 算法设计	28
4.2.1 约束松弛	30
4.2.2 下界获取	30

4.2.3 上界获取	35
4.2.4 乘子更新	36
4.2.5 停止条件	37
4.2.6 上界改进	37
4.3 本章小结	39
5 数值实验设计及算法性能分析	41
5.1 实验设计	41
5.2 综合求解性能分析	42
5.2.1 求解性能对比	42
5.2.2 算例参数 ρ 的影响	43
5.2.3 算例参数 R 的影响	45
5.2.4 优化曲线	46
5.3 下界求解性能分析	47
5.4 上界求解性能分析	49
5.5 乘子更新效果讨论	51
5.6 线性化效果讨论	52
5.7 算例结果及灵敏度分析	55
5.7.1 结果展示	55
5.7.2 参数 ρ 灵敏度分析	57
5.7.3 参数 R 灵敏度分析	58
5.8 本章小结	59
6 总结与展望	61
6.1 工作总结	61
6.2 研究展望	62
参考文献	63
附录 A	69
附录 B	89
作者简历及攻读硕士学位期间取得的研究成果	101
独创性声明	103
学位论文数据集	105

1 引言

物畅其流则百业兴，我国正在加速构建以国内大循环为主体，国内国际双循环相互促进的新发展格局，打造安全可靠、高效便捷的现代物流网络系统迫在眉睫。其中，物流网络基础设施节点的科学布局、稳定运行是制约物流网络可靠性的关键要素。尤其近年来，物流网络面临愈加频繁的自然灾害、公共卫生事件、复杂多变的国际经济政治形势等突发风险。因此，如何在物流网络规划阶段就考虑节点失效风险，高效计算并设计出科学合理、安全可靠的物流网络，是我国物流业降本增效、抢占“双循环”机遇制高点的一个重要抓手。

1.1 研究背景和意义

1.1.1 研究背景

现代物流是经济发展的“经脉”，连接着生产与消费的全部流程^[1]，将运输、仓储、包装、配送、装卸搬运、流通加工、信息处理等服务功能高度集成，是延伸产业链、提升价值链、打造供应链的重要支撑，是促进国民经济循环畅通的重要抓手^[2]。在国务院《“十四五”现代物流发展规划》宏伟蓝图的指引下，我国社会物流效率逐渐稳固提升，市场环境不断向好发展。近五年来，我国物流经受住了多重压力，实现了行业稳步增长。2018年至2022年，我国物流社会总额从283.1万亿元增长至347.6万亿元，复合年均增长率4.2%，物流业总收入从10.1万亿元增长至12.7万亿元，复合年均增长率4.7%¹。

随着行业的快速发展，不断增加的物质资源、信息资源和社会生产活动将物流节点连接成一个复杂且精密的网络。然而，网络中的节点受到诸多自然因素和人为因素的影响从而面临失效风险，例如2012年北京特大暴雨和2016年1号台风“尼伯特”等自然因素；2015年天津滨海新区爆炸事件和上海浦东申通三林分公司的罢工事件等人为因素，直接导致当地部分基础设施节点和线路失效；2020年新型冠状病毒肺炎疫情爆发，运力资源短缺、运输网络阻隔、运营安全欠缺等挑战蜂拥而至，“九省通衢”的武汉节点及关联网络失效，物流体系受到巨大影响，且由于针对节点失效风险的物流网络规划较少，导致物流网络不同程度受损，区域物流网络几近瘫痪，同时也对国民经济中诸多行业造成雪崩式影响。因此，在节点选址布局设计时就考虑可能存在的失效风险，或针对既有网络布局寻求优化方案，

¹数据来源：中国物流与采购联合会。

从空间布局上适度安排合理的备用节点。一旦当前常用节点失效，则由其备用节点提供物流服务，以减少成本、时间上的损失，提高整个物流网络的可靠性。

此外，即便在信息化高度发达的当今社会，物流节点失效也常常伴随电力设施损毁、基础通信设施破坏，导致信息传递不及时甚至中断；亦或者客户已知节点损坏，但不确定因素导致客户不清楚节点何时可以恢复服务。此时，客户不能准确判断节点能否或何时提供服务，处于一种“不完全信息”的状态。在不完全信息的状态下，客户通常会采取一种试错策略，以寻求可能存在的服务。在现实生活的各个行业中，这种试错策略很普遍：对于零售企业（商场、零售店等）或是基础服务设施（电动汽车充电桩、停车场、ATM机等）来说，在客户到达这类设施之前，并不能准确知道自己是否可以被该设施服务，因此决定先前往试一试运气，如果不能被服务则前往下一个设施。在物流行业中，除了末端零售节点以外，客户采用试错策略访问中转节点的场景也有很多：例如，早年的“双十一购物节”，快递爆仓时有发生，若大量快递滞留在某个中转中心来不及处理，则可能被转运至其他中转中心以分散压力。另一个近期的案例是 2021 年 5 月下旬，作为世界上第三大港口的深圳盐田港因港区工人确诊新冠肺炎导致严重的货物积压，不得不暂停了五天的集装箱入闸业务，恢复后也只能在短时间内接受船舶预计到港日期前三天内的出口重柜^[3]。在暂停业务的期间，一部分集装箱选择绕过该港口尝试从其他港口出港，但成本有会有不同程度的提升。

在不完全信息场景下，虽然只有一个节点为客户的单次需求提供服务，但每个客户可能有多个节点为其长期提供备用服务。客户的试错策略可能使得客户在寻求服务时逐个访问多个节点，使得节点之间产生了额外联系。这些节点距离客户的远近不同。相应地，客户寻求服务时也以总期望成本最小的顺序逐个尝试。然而，一般的经典选址模型（例如 P-中心、P-中值、集合覆盖模型等）假设设施建设后能持续为客户提供服务，并未预留一个使客户在设施失效后能够及时获得服务的备用方案或容错方案。在本世纪初的选址理论发展过程中，可靠固定成本选址 (Reliable Fixed-charge Location, RFL) 理论被提出^[4]，旨在解决上述经典理论中未考虑到的失效因素。RFL 模型考虑了设施失效的情况，即假定每个设施在建成后都存在损坏的概率。为了在设施损坏时尽可能降低损失，在规划初期，RFL 模型为每个客户指派了一个常用设施之外，还指派了多个备用设施。通过“一个常用设施，多个备用设施”的策略，RFL 模型降低了常用设施损坏时，因临时决策产生的高昂成本。本文的第2章针对 RFL 模型的发展现状和理论背景展开了详细的叙述。

针对现实存在的问题和需求，基于 RFL 选址理论，本文研究了考虑节点失效风险的物流网络选址问题，目的是以一种预先规划的主动强化策略提升物流网络的韧性。其中，可靠性是指在系统内部某些部分失灵的情况下，整个系统仍能良

好运行的能力^[5]；不完全信息是指客户对节点的状态没有完全地了解^[6]；试错策略是指客户在常用节点失效后，逐个访问备用节点的过程。由于在节点阻塞的状态（例如，拥堵、爆仓等）被视为失效状态，本文在模型的考虑因素中主动忽略了设施的容量限制。因此，本文研究的模型是可靠无容量固定费用选址问题 (Reliable Uncapacitated Fixed-charge Location, RUFL) 的相关拓展和应用，其原始理论为无容量固定费用选址问题 (Uncapacitated Fixed-charge Location, UFL)^[4]，同时也是网络和离散选址问题的一个分支^[7]。另外，从上述介绍可知本文研究的不完全信息场景适用于设施端提供服务的节点，例如末端零售、中转节点、港口、仓库等。区别于狭义的“消费者”的概念，本文考虑的客户是一个广义的角色，是指接受节点服务的个人或组织。例如，对于快递的中转中心来说，狭义的客户是指快递的收件人，但本文考虑的广义客户是物流公司自身，因为其接受了中转中心的服务。最后，值得一提的是本文讨论的问题并不针对于某种特殊的物流节点类型，而是提出了一个普适的、与物流节点相关的可靠选址理论，第1.1.2节阐明了研究该理论的现实意义和理论意义。

1.1.2 研究意义

上述内容中提到，物流节点是物流网络中的重要组成，承担了物流基础活动中除运输之外的大部分功能，如分拣、仓储、包装、搬运、流通加工等^[8]。研究物流节点的可靠选址问题，有利于构建一个具有韧性的、可靠的现代物流网络。本文的研究意义可分为现实意义和理论意义：

从现实意义出发，由于设施的建设成本较高，且建设后将长期提供服务，因而选址问题成为企业长期的战略问题。错误的选址决策会导致运营成本增加和企业竞争力下降^[7]。在物流网络拓扑结构设计之初，考虑物流节点失效和不完全信息场景，将降低运营过程中由于节点失效导致的高昂损失，提升节点失效时物流网络的运营效率，为客户提供更加优质的物流服务。

从理论意义出发，本文研究的物流节点选址问题，丰富了相关选址理论，是RUFL理论的进一步深化和实际应用。本文采用了数学优化方法对现实问题进行抽象并建模，虽然所求解的模型的最优性是数学意义上的，但是建模的过程（分析目标函数和约束，收集并整理数据）也为改进选址决策提供了参考和依据。此外，求解模型的过程，丰富了相关算法理论研究，是拉格朗日松弛算法求解大规模整数规划问题的具体实践。

1.1.3 研究创新点

本文研究的问题是 RFL 理论的进一步拓展和应用，从物流节点的角度出发，考虑了节点可能失效的情况与应对失效的措施；从广义客户的角度出发，考虑了信息的不完全性以及客户接受服务的试错过程；刻画了客户的行为，并计算了期望成本；为了求解该问题的大规模实例，定制了优化算法。综上，本文的创新点集中体现在模型创新与算法创新两个方面，具体如下：

(1) 考虑了物流节点的失效风险。在传统物流节点选址问题中，通常认为节点在建成后可以持续为客户提供服务，然而在现实中，节点所处环境中的风险可能导致节点失效。本文将节点的失效概率纳入模型中，这使得模型更加贴近现实，优化的结果也更具现实意义。

(2) 描述了客户在节点失效时尝试获取服务的试错策略，引入了客户在不能接受服务时产生的试错过程，这种试错过程是应对设施失效、强化网络可靠性的方法之一，但很少出现在选址理论的研究中。将客户试错过程与设施失效相结合，并体现在模型构建中，是考虑节点失效风险的物流网络选址模型的创新之处。

(3) 设计了基于迭代局部搜索改进的拉格朗日松弛算法。由于研究的问题是非确定性多项式难 (Non-deterministic Polynomial Hard, NP-hard) 类型，问题的规模随节点数量的增加指数级增长。本文为该问题定制的混合优化算法可以有效求解大规模的算例。该算法能够得到一个高质量下界，并通过多种精确和启发式方法保证了上界的质量。算法精心设计的拉格朗日乘子初始化及更新规则可使得上下界快速收敛并证明解的优劣。

1.2 研究路线

针对物流节点可能失效的情况，考虑到节点失效同时信息的不完全特性以及选址决策的特点，本文对考虑节点失效风险的物流网络选址问题进行研究。本文的技术路线如图1-1所示，全文共分为 6 个章节，每章的内容如下：

第1章介绍了本文的研究背景，从繁荣发展的物流行业背后仍存在节点失效的现象出发，引出本文的研究主题。结合研究背景，从理论和现实两个方面明确了本文的研究意义；分别从模型与算法两个角度阐明本文的创新性；确定了本文的研究路线和文章结构安排。

第2章首先回顾并总结了可靠选址理论的发展过程，着重总结了一系列建模考虑的因素和求解模型的算法，指出了研究的薄弱之处和未来可能的研究方向。然后简要概述了可靠选址理论在物流选址问题中的应用，特别是相关考虑因素和解决方案。

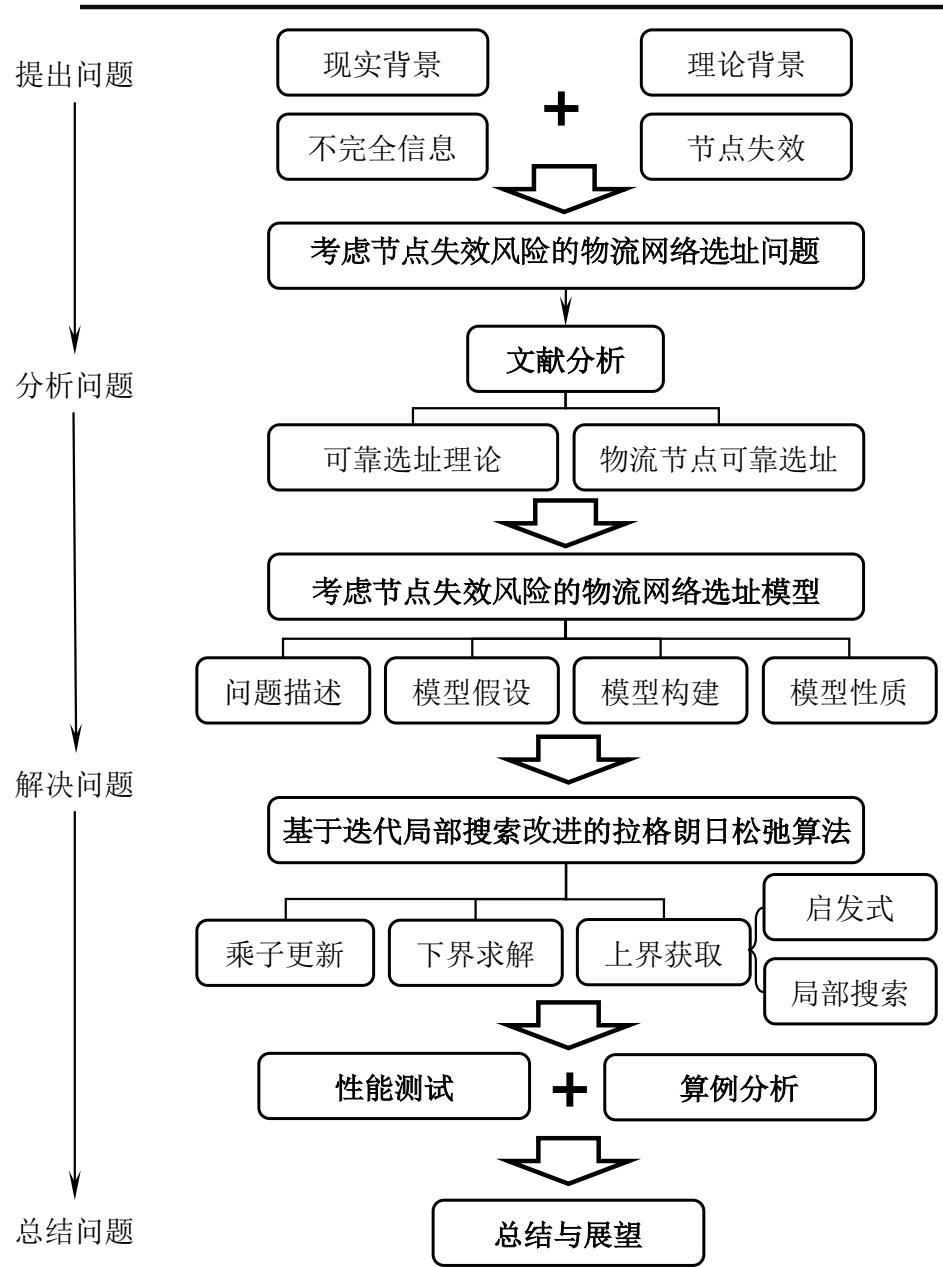


图 1-1 技术路线
Fig 1-1 Technical Route

第3章为考虑节点失效风险的物流网络选址问题构建了数学模型，并证明了该问题的 NP-hard 属性。由于该模型属于混合整数规划 (Mixed Integer Programming, MIP) 类型，且带有二次约束和二次目标项，第3章中应用了线性化技术消除模型的二次成分。此外，模型性质分析中还讨论了一个子问题，本文称为试错序列问题，该问题是基本最短路问题更为广泛的形式之一。

第4章开发了基于迭代局部搜索改进的拉格朗日松弛算法，从一个一般问题出发推导算法的基本原理，并开发了一系列求解上界、下界的精确方法和启发式方法，详细阐述了算法的控制流程。

第5章采用经典测试数据集验证了算法的性能和有效性，对比了算法与商业求解器之间的性能差异，分析了算法中重要算子的性能。并对算例数据中的重要参数进行了参数灵敏度分析，从分析结果中给出管理学启示。

第6章总结了全文，指出本文研究工作的不足和待完善之处，并展望了该问题未来的研究方向和研究内容。

1.3 本章小结

本章确定了考虑节点失效风险的物流网络选址问题的研究背景，通过现实案例介绍了物流节点所面临的失效风险以及客户的试错策略。此外，本章介绍了研究的理论意义和现实意义以及模型和算法的创新性。最后，确定了本文的研究思路和文章架构。在讨论的过程中，明确了本文研究的问题基于可靠选址问题。因此，第2章回顾了可靠选址问题相关的研究，特别是与物流节点选址相关的文献。

2 文献综述

选址问题是一个经典的优化问题。该问题可追溯至 Alfred Weber^[9] 于 1909 年首次提出的一个完善且成熟的工业选址理论，该理论试图以最低成本找到工厂的最佳位置。经过一个世纪的发展，已形成了一系列的经典选址模型。基于需求和设施空间分布，这些选址模型可分为解析选址模型、连续选址模型、网络选址模型和离散选址模型。本文讨论的物流节点可靠选址问题属于离散选址问题，而有关于上述四种模型的区别，请参考 Daskin 的著作^[7]。本章的第 2.1 节回顾了可靠选址的发展历程，总结了相关研究工作。第 2.2 节描述了物流节点可靠选址理论研究现状，指出当前研究的进展、考虑因素和解决方案。

2.1 可靠选址理论综述

离散选址问题是给定一个网络拓扑结构，在设施候选建设位置集合中寻找满足条件的最佳子集。UFL 问题和 P-中值问题是离散选址问题的重要分支，二者的共同点是在网络中以最小的成本为设施选择建设的最佳位置并服务所有客户^[7]，区别在于 P-中值问题指定了设施选址的数量为 P，而 UFL 选址问题对设施数量没有限制。Mak^[10] 及 Daskin^[7] 对 P-中值问题和 UFL 选址问题的基础理论进行了详细阐述。本世纪初，学者们逐渐意识到不确定因素对设施网络的冲击和影响，开始着重研究节点选址中的不确定因素。考虑节点失效风险的选址问题起源于对供应链的中断现象的研究^[11]，基于上述的 UFL 问题和 P-中值问题，Snyder 等人^[4]提出了这两种选址理论的可靠版本：可靠 P-中值问题 (Reliable P-Median Problem, RPMP) 和可靠无容量固定费用选址问题 (RUFL)。可靠选址理论在经典选址模型的基础上，考虑每个设施损坏的概率，并为每个客户安排一个常用设施和多级备用设施，极大地提高了网络的可靠性。

Snyder^[4] 在其可靠选址基础理论中，将设施分为完全可靠类型（即永久不会失效）和不完全可靠类型，在后续研究中该理念仍被采纳^[12-14]，但大部分学者将所有设施都认定为不完全可靠状态，即每个设施都有可能失效。在 Snyder^[4] 的研究基础上，Cui^[15]、Shen^[16]、王艳敏^[17] 松弛了设施失效概率相等的假设，即设施的失效概率不同。Aboolian^[18] 不仅松弛了失效概率相等的假设，还松弛了每个客户拥有的备用设施的数量上限，但大部分学者仍接受客户拥有有限个备用设施的约束。

在可靠选址基本理论的基础上，更多现实因素加入模型构建中。首先，导致设

施损坏的灾害往往也使得通讯中断，或者使得客户对设施的状态没有准确把握，导致造成不完全信息状态的产生。Berman^[6] 最先研究了这种状态下的设施选址问题，Yun^[19] 基于 RUFL 理论提出了不完全信息下的可靠选址理论。在该研究成果的基础上，松弛了节点失效概率相等的假设^[20]，并考虑了返程的成本^[21]，对所研究的可靠选址理论进行了完善。考虑到设施可在失效之前被强化的可能，Li 等人^[22-24] 提出使用有限的成本强化设施，以降低系统的总成本。在 Li^[22,23] 的研究中，假设设施被强化后转变为完全可靠设施，因此要求每个客户只能拥有一个常用设施和一个备用设施，客户拥有完全可靠的设施后并不需要其余备用设施。除了增加预算抵抗设施的失效，风险规避也是提升可靠性的方法之一。Yu^[25,26] 提出了风险规避型可靠选址理论。在以往的理论中，通常计算整个网络的最小期望成本，Yu^[25,26] 则控制每个客户承担的风险。设施的失效可以是独立的，也可以是关联的，该研究可以得到比传统模型更可靠的网络结构。

经典的 RUFL 理论考虑了设施的损坏失效情况，但假设设施失效是独立的。Lu 等人^[27] 假定设施的失效之间存在一定的关联性，采用了鲁棒优化方法研究了给定边际失效概率条件下的最小期望成本选址方案。鲁棒优化是研究设施失效关联性的一种方法之一，与鲁棒优化相关的可靠选址研究还可以参见 Du 等人^[28] 设计的双层可靠 P-中心网络结构，Peng 等人^[29] 研究了考虑节点失效的物流网络结构，Li 等人^[30]、An 等人^[31] 提出了两阶段鲁棒优化可靠选址理论。另一种研究失效相关的方法是连续近似模型，Li^[32] 构建了节点失效的关联性公式，优化的目标是得到期望总成本最小的选址方案。有关可靠选址的连续近似模型相关的讨论，可见参考文献^[15,33,34]。除了设施失效可能存在关联，设施之间的相关性也可以体现在共同应对风险^[35]，使用更复杂的网络支撑结构提高整个网络的强度^[36]。

上述的选址模型通过考虑设施失效的概率或边际失效概率以量化失效风险。Berman^[37] 将设施的可靠性与客户之间的距离关联，研究了相关可靠选址问题。此外，在现实中，不确定性还体现在多个方面，例如需求不确定性的可靠设施选址^[38]，估计设施失效概率的误差^[39]。Snyder^[40] 概括了设施选址中的不确定因素，Snyder^[41] 近期分析了供应链中的不确定因素，Govindan^[42] 调查了供应链网络设计中的不确定因素。这些研究表明，考虑不确定因素的影响是至关重要的，可从供应链设计中借鉴经验并有效开展可靠设施选址研究。周渝峰^[11] 针对可靠选址理论的发展展开了详细描述，并指出了未来的研究方向。

已知可靠选址问题是 NP-hard 问题^[4]，这意味着当前不存在求解该问题的多项式时间算法。因此，开发有效的求解算法是研究可靠选址的另一重点内容。当前主流的可靠选址求解算法包含精确方法和启发式方法。精确方法包括 Branch-and-Bound, Benders 分解, Branch-and-Price 等方法，启发式方法包括拉格朗日松弛以

及包括禁忌搜索、模拟退火、遗传算法等在内的元启发式方法。通常，元启发式方法虽然不能保证求解的质量，但可在可接受的时间内得到一个经过优化的可行解，因此对于大规模问题，解决方法主要以启发式方法为主。

精确求解可靠选址问题的相关研究数量上相对较少，求解方法主要以 Benders 分解算法为主。有关 Benders 分解求解基本设施选址问题的研究可见参考文献^[43-45]。Mohammad^[46] 使用加速 Benders 分解算法求解了多层的可靠网络结构，Azad^[47] 开发了 Benders 分解算法解决了一个可应对节点失效的弹性供应链网络结构设计问题。Pirniya^[48] 使用了列生成 (Column Generation) 算法求解了可靠选址、客户分配及路径问题，并取得了良好的效果。

而采用启发式算法求解可靠选址问题的研究相对较多。拉格朗日松弛是一种求解整数规划问题以及混合整数规划问题的高效算法^[49,50]。对于求解最小化问题，拉格朗日松弛不仅可以快速获得优化上界，还提供了优于线性松弛的下界，因此可以证明上界的质量。拉格朗日松弛在求解选址问题方面取得了显著成就，Geoffrion^[51] 首次使用拉格朗日松弛求解了 UFL 问题，Cornuejols 等人^[52] 首次使用拉格朗日松弛求解了 P-中值问题。拉格朗日松弛算法同样也是求解可靠设施选址问题的主流算法，其设计过程具有灵活性，可根据问题定制。因此即便都采用了拉格朗日松弛算法，算法的技术细节都不尽相同。Snyder^[4] 首次使用了拉格朗日松弛算法求解了 RUFL 问题，后续的研究结果表明拉格朗日松弛求解 RUFL 问题效果显著：Cui^[15] 使用该算法求解了不等失效概率的可靠选址问题，Yun^[19] 开发了拉格朗日松弛算法框架求解了不完美信息下的可靠选址问题，Li^[22] 为考虑强化策略的可靠选址问题定制了拉格朗日松弛算法。Yu^[26] 基于拉格朗日松弛框架设计了对偶分解算法求解了风险规避型可靠选址问题。

前文中提到拉格朗日松弛算法具有灵活性和实用性，因此可将拉格朗日松弛作为算法框架，将其他算法嵌入拉格朗日松弛算法中，也可以将拉格朗日松弛嵌入其他算法中。Xie^[53] 将列生成和局部搜索嵌入拉格朗日松弛，研究了可靠选址路径问题。Yu 等人^[25] 将 Branch-and-Cut 算法和拉格朗日松弛算法相结合，求解了风险规避型可靠选址问题。此外，应用拉格朗日松弛算法求解可靠选址问题，还可见参考文献^[21,23,54]，有关拉格朗日松弛算法的介绍可见参考文献^[49]，有关算法原理的推导可见参考文献^[50,55]。

除了拉格朗日松弛之外，还有学者采用其他启发式方法求解可靠选址问题，包括 Shen^[16] 提出的近似算法和贪心算法，Aboolian^[18] 提出的近似方法，以及 Albareda^[12] 提出的近似流方法。而元启发式方法也在求解大规模选址问题时展现了显著优势，例如遗传算法^[56]、邻域搜索^[57]、禁忌搜索^[58]、变邻域搜索^[59] 等。求解可靠设施选址问题，Peng 等人^[29] 基于遗传算法和邻域搜索算法开发了混合元启

发式方法, Li^[24] 开发了禁忌搜索算法, Afify^[60] 开发了进化学习算法, 王艳敏^[17]开发了模拟退火-粒子群算法。

综上, 可靠选址问题的研究主要集中在两个方面: 其一是在构建模型时考虑各种现实因素, 以强化模型的现实意义。从当前的研究成果来看, 大多可靠选址理论假设节点失效独立, 而有关节点失效相关联的研究成果较少。在大多数可靠选址理论中, 节点的失效是以先验概率的形式表达的, 这对估计失效概率的精准程度有较高的要求, 有关失效概率预估误差对选址结果造成影响的研究较少。其二是设计有效的算法求解该 NP-hard 问题。以提升求解复杂模型的能力。当前可靠选址求解算法研究的热门仍是启发式方法, 采用精确方法求解该问题的研究较为薄弱。

2.2 物流节点可靠选址理论综述

物流节点是物流网络的构成要素, 节点的选址决策将显著影响物流系统的运行效率, 关系物流企业的经济利益。窦志武等人^[61] 回顾了物流节点选址的方法, 包括重心法、整数规划模型法、层次分析法、数据包络分析法、模糊综合评价法等。本文研究的物流节点可靠选址问题采用了整数规划模型法, 其他选址方法可见参考文献^[62-64]。

物流节点可靠选址的大部分研究关注于节点的状态, 特别是节点因某些原因不能提供服务的场景。通常, 物流节点失效状态由失效概率表示, 即该节点以某个概率值不能提供服务。以供应链为研究对象, 王艳敏^[17] 假设每个设施的失效概率不等, 建立了一个容量有限的供应链设施可靠选址的多目标模型; 张莹^[65] 研究了供应中断风险的选址-路径问题, 考虑所有的设施均以先验概率发生失效, 多个设施也可以同时发生失效的场景; 汤罗浩^[66] 设计了一种可强化节点的供应链网络, 通过对节点进行保护并为需求点分配备份节点来提高供应链网络的可靠性; 王继光^[67] 从供应链系统的可靠性和鲁棒性两方面研究了中断情境下的可靠选址问题, 针对随机中断和非随机中断分别提出了混合整数非线性规划模型和两阶段动态博弈模型。以铁路建设为研究对象, 考虑了建筑材料的时效性需求, 周浩^[68] 研究了铁路沿线的线状需求的物流节点连续型可靠选址问题, 获得了靠近需求线路中点的物流节点选址位置的解析解。以区域物流为研究对象, 朱江华等人^[69] 以贵州省为案例开展物流园区可靠选址规划分析, 构建了考虑突发事件的物流园区选址与服务分配模型; 董鹏^[8] 以京津冀地区为例, 构建了信息失效情景下考虑双向行程的可靠节点连续选址模型。

而另一部分研究关注于线路的状态, 即节点和客户之间的线路发生中断从而不能向客户服务的场景。颜高民^[70] 研究了突发事件下应急物流可靠路径搜索问题,

考虑了突发事件下路径畅通的不确定性，构建了最短路和畅通路径综合模型，并设计了一定规则修正路径。李锐等人^[71]研究可靠绿色物流配送选址-路径问题的同时，考虑了运输油耗和碳排放及配送中心和运输线路的中断，建立物流配送网络选址-路径优化模型，以最小总成本满足车辆路径约束。考虑到供应网络中每条路径都存在运输中断风险，任慧^[72]研究了多源协同供应的三级供应网络可靠选址路径集成优化问题，在有限的设施（工厂、配送中心）能力下，提出了运作成本和运输可靠性的双目标选址路径模型。石褚巍等人^[73]研究了网络中存在节点和线路损坏不确定性的可靠物流网络，提出了一种基于两阶段鲁棒优化的可靠物流网络设计方法。

除了节点和线路的不确定性，需求和供应的不确定性也是研究的重点。考虑到二次灾难可能造成应急配送中心失效以及救援过程中道路中断情况的发生，张鹏阁^[74]研究了需求不确定的应急物流选址-路径优化问题，建立了一个包含常规型和可靠型的应急物流网络。孙晓飞等人^[75]研究了不确定环境下配送中心选址的多目标优化问题，探讨了配送系统的可靠性。姚琦^[76]研究了生鲜农产品配送中心选址的问题，考虑时间可靠度和品质可靠度对配送中心选址的影响，运用层次分析法与模糊综合评价法建立了可靠物流配送中心选址模型。

此外，设施的失效常常与自然灾害相关联。周渝峰^[77]等人以应急物资保障的及时性和可靠性为目标，考虑在不同地区建立储备库的不同失灵概率，构建了一个应急物资储备库的可靠 P-中值选址模型，并设计了一种拉格朗日松弛求解算法。李政祥^[78]考虑设施的中断，以成本和时间最小化为目标，建立了应对灾害的应急物流多目标可靠选址模型。李政祥^[79]构建了考虑设施设防及设施中断的多目标大规模地震灾害应急物资储备库的选址-分配模型，设计了非支配解排序遗传算法 (NSGA-II) 以求解多目标问题。朱建明^[80]同时考虑设施可能的损毁情景以及设施两两之间的调度时间，建立了可靠连通应急设施选址模型，设计了基于遗传算法的求解方法。于冬梅等人^[81]建立了中断情境下服务能力有限的可靠应急设施选址-分配多目标优化模型，采用带精英策略的快速 NSGA-II 对模型予以求解。王子墨^[82]建立了考虑失效风险的救灾物资储备库可靠选址模型，对储备库的位置、容量和需求点与救灾物资储备库的分配关系进行决策，并通过遗传算法对模型进行求解。

综上，物流节点的可靠选址研究将可靠选址理论与实际结合，根据现实场景和需求进一步拓展和应用可靠选址理论。物流节点可靠选址问题的研究主要考虑了节点、线路、供应和需求的不确定性，研究对象涵盖了供应链网络优化、物流园区设计、配送中心选址、应急物流规划等内容。

2.3 本章小结

选址问题是运筹优化领域经久不衰的一项重要研究内容，而可靠选址研究是近二十年来选址理论的重要发展成果。本章详细展开了可靠选址的发展过程，介绍了可靠选址的起源以及可靠选址问题重要分支。这些研究成果可分为理论创新和算法创新。理论创新考虑了更多的现实因素，而随着越来越多的现实因素加入模型中，求解模型的困难程度提升，也促进了相关算法的发展。

此外，可靠选址是物流节点选址的热门研究内容。物流系统可能会面临各种不确定因素，这些不确定因素可发生于供给端、需求端以及二者之间连接过程，对物流网络造成严重的负面影响，因此有必要在选址时将不确定因素考虑其中。由本章文献综述可知，考虑节点失效风险和不完全信息场景下客户试错策略的研究相对较少。研究该问题有助于强化物流网络结构，合理规划节点选址和客户分配，为物流网络结构设计提供理论支撑。为采用科学方法研究该问题，本文的第3章构建了该问题的数学优化模型。

3 考虑设施失效风险的物流网络选址模型构建

如第1章所述，物流节点的运营常常伴随风险。这些风险导致物流节点失效的同时还伴随不完全信息状态的产生。一般情况下，节点失效的概率是先验的。可通过包括气象、新闻在内的历史数据，统计失效时长占总统计时长的比例获得节点的失效概率，或统计失效次数占总次数的比例获得，还可以通过调研对未来失效概率进行预测。由于不同地区、不同位置的候选点的外部环境不同，每个节点的失效概率也不相同。根据客户的试错策略，访问某一级备用节点的概率将与其之前访问的所有节点的失效概率相关。为了表示这种概率关系，本章采用了非线性方法表达模型的目标函数和约束。同时本章提供了线性化技术处理模型中非线性成分，降低了使用精确方法（例如商业求解器）求解模型的难度。

本章的主要内容和结构如下：首先，第3.1节声明了模型中存在的理想化假设。其次，第3.2节使用了数学语言表述了问题。然后，第3.3节建立了数学模型。此外，第3.4节提供了模型的线性化方案，第3.5节分析了该模型的一系列性质，证明该问题的 NP-hard 性质，并讨论了模型的子问题与基本最短路问题的关系。最后，第3.6节总结了本章的内容。

3.1 模型假设

本文考虑节点失效风险的物流网络选址模型同其他数学模型一样存在一系列理想化假设。由于本文的模型根源于 UFL 选址理论，因此继承了 UFL 问题的一般假设：

- (1) 客户的需求已知；
- (2) 客户的单次需求不可拆分；
- (3) 所有节点均不考虑容量限制。

此外，针对本文讨论的问题的特殊性，提出了以下的理想化假设：

- (4) 假设每个节点的失效概率是先验的；
- (5) 节点失效发生彼此独立。

最后，本文考虑的问题基于一般图，因此基于图理论假设：

- (6) 任意两点之间的成本是非负且不可变动的；
- (7) 两点之间的道路是永远联通的（即，图中弧和弧的权重不变）。

3.2 问题描述和符号说明

本节将使用数学语言更为严谨地描述前文所述的问题，相关符号表示如下（为方便索引，在本节末尾的表3-1给出了符号及其对照解释）：设 I 表示客户点的集合，设 J 表示候选点的集合。给定一个有向图 $G = (V, A)$ ，其中 $V = I \cup J$ 是点集， $A = \{(i, j) | i \in V, j \in J\}$ 是弧集。每个客户 $i \in I$ 的需求为 λ_i ，每个候选点 $j \in J$ 的建设成本为 f_j ，失效的概率为 q_j 。每条弧 $(i, j) \in A$ 的权重为 c_{ij} ，表示从点 $i \in V$ 到点 $j \in J$ 单位需求的运输价格（本章简称为价格）。

任一客户 $i \in I$ 可以同时拥有一个常用节点和多个备用节点，但每个客户的备用节点数可能不同。额外引入虚拟节点 j_0 ，并规定客户试错过程的最终节点一定为虚拟节点 j_0 。访问虚拟节点表示当客户的所有实体节点（常用节点和备用节点）都失效时接受的惩罚。从点集 V 中的点 $i \in V$ 到虚拟节点 j_0 的价格 $c_{ij_0}, \forall i \in V$ 等于惩罚价格 π ，表示客户用尽所有备用节点都没有获得服务后产生的损失。虚拟节点的建设成本 $f_{j_0} = 0$ ，失效概率 $q_{j_0} = 0$ ，定义集合 J 被 j_0 扩充后的集合记为 $\bar{J} = J \cup \{j_0\}$ 。此外，限制每个客户拥有的实体节点数之和小于等于 R ，表示客户可接受的最大尝试次数。

为了可视化地展示客户试错过程，以及推导期望运输成本，图3-1展示了客户的试错策略。图中客户用正方形表示，实体节点用实体圆形表示，虚拟节点用空心圆形表示，弧旁符号表示该段弧的价格。图中集合 $J_i = \{j_i^r | r = 0, 1, \dots, R - 1\}$ 为服务客户 i 的实体节点，上标 r 表示访问的先后顺序，即客户第 r 等级的备用节点 ($r = 0$ 表示常用节点)，因此集合 J_i 中元素的个数 $|J_i|$ 不超过 R 。

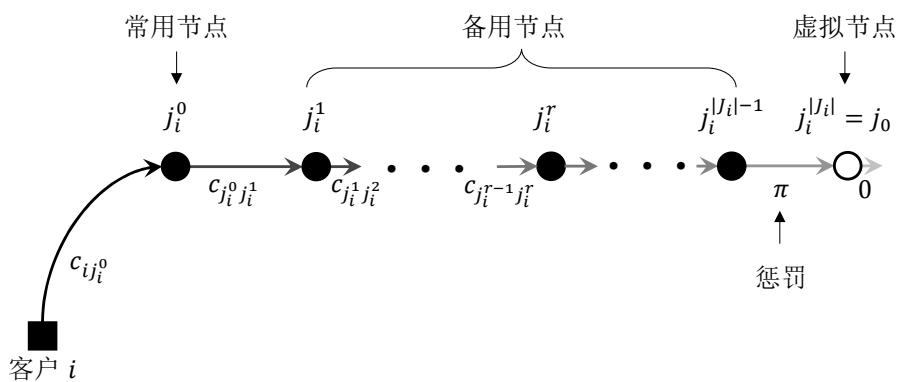


图 3-1 客户试错策略示意图

Fig 3-1 A Trail-and-error Strategy of a Customer

当客户 i 访问节点 j_i^0 时， j_i^0 以 $q_{j_i^0}$ 的概率失效，以 $1 - q_{j_i^0}$ 的概率提供服务。如果节点 j_i^0 正常服务，那么客户 i 从节点 j_i^0 获得服务的期望成本等于 $\lambda_i c_{ij_i^0} (1 - q_{j_i^0})$ 。

如果节点 j_i^0 失效，那么客户 i 将从 j_i^0 继续出发访问节点 j_i^1 。同样，若节点 j_i^1 正常服务，那么客户 i 从节点 j_i^1 获得服务的期望成本等于 $\lambda_i(c_{ij_i^0} + c_{j_i^0 j_i^1})q_i^0(1 - q_i^1)$ ，即在上一个点 j_i^0 失效的情况下点 j_i^1 没有失效。若节点 j_i^1 失效，则客户 i 将从 j_i^1 继续出发访问节点 j_i^2 ，以此类推，直至客户接受服务或访问完所有节点。

归纳上述推导过程，客户 i 访问第 $r(r \geq 1)$ 等级的节点的期望成本 C_e^{ir} ，如公式(3-1)所示：

$$C_e^{ir} = \lambda_i \left(c_{ij_i^0} + \sum_{m=1}^r c_{j_i^{m-1} j_i^m} \right) \left(\prod_{n=0}^{r-1} q_{j_i^n} - \prod_{n=0}^r q_{j_i^n} \right) \quad (3-1)$$

于是，对所有等级 r 求和，客户 i 的总期望成本 C_t^i 如公式(3-2)所示：

$$C_t^i = \lambda_i \sum_{r=1}^R \left((c_{ij_i^0} + \sum_{m=1}^r c_{j_i^{m-1} j_i^m}) \left(\prod_{n=0}^{r-1} q_{j_i^n} - \prod_{n=0}^r q_{j_i^n} \right) \right) + \lambda_i c_{ij_i^0} (1 - q_{j_i^0}) \quad (3-2)$$

公式(3-2)等号右侧的第一项表示从 $r = 1$ 至 $r = R$ 的期望运输成本之和，右侧第二项表示虚拟节点的期望惩罚成本。提取前面的一个求和项 $r = 1$ 与后面的项合并得到，

$$\begin{aligned} C_t^i &= \lambda_i \sum_{r=2}^R \left((c_{ij_i^0} + \sum_{m=1}^r c_{j_i^{m-1} j_i^m}) \left(\prod_{n=0}^{r-1} q_{j_i^n} - \prod_{n=0}^r q_{j_i^n} \right) \right) \\ &\quad - \lambda_i \left(\prod_{n=0}^1 q_{j_i^n} \right) \left(c_{ij_i^0} + \sum_{m=1}^1 c_{j_i^{m-1} j_i^m} \right) \\ &\quad + \lambda_i \left(c_{ij_i^0} + c_{j_i^0 j_i^1} \prod_{n=0}^0 q_{j_i^n} \right) \end{aligned} \quad (3-3)$$

再提取一项 $r = 2$ 与后面的项合并，得到，

$$\begin{aligned} C_t^i &= \lambda_i \sum_{r=3}^R \left((c_{ij_i^0} + \sum_{m=1}^r c_{j_i^{m-1} j_i^m}) \left(\prod_{n=0}^{r-1} q_{j_i^n} - \prod_{n=0}^r q_{j_i^n} \right) \right) \\ &\quad - \lambda_i \left(\prod_{n=0}^2 q_{j_i^n} \right) \left(c_{ij_i^0} + \sum_{m=1}^2 c_{j_i^{m-1} j_i^m} \right) \\ &\quad + \lambda_i \left(c_{ij_i^0} + c_{j_i^0 j_i^1} \prod_{n=0}^0 q_{j_i^n} + c_{j_i^1 j_i^2} \prod_{n=0}^1 q_{j_i^n} \right) \end{aligned} \quad (3-4)$$

如此进行，直至全部展开，最终得到，

$$\begin{aligned} C_t^i &= -\lambda_i \left(\prod_{n=0}^R q_{j_i^n} \right) \left(c_{ij_i^0} + \sum_{m=1}^R c_{j_i^{m-1} j_i^m} \right) \\ &\quad + \lambda_i \left(c_{ij_i^0} + c_{j_i^0 j_i^1} \prod_{n=0}^0 q_{j_i^n} + c_{j_i^1 j_i^2} \prod_{n=0}^1 q_{j_i^n} + \dots + c_{j_i^{R-1} j_i^R} \prod_{n=0}^{R-1} q_{j_i^n} \right) \end{aligned} \quad (3-5)$$

由于设定 $r = R$ 时，备用节点一定为虚拟节点，于是有 $q_{j_i^R} = q_{j_0} = 0$ ，所以公式(3-5)的负数项可消除 ($\prod_{n=0}^R q_{j_i^n} = 0$)，后面的正数项经过整理，可表示为，

$$C_t^i = \lambda_i \left(c_{ij_i^0} + \sum_{r=1}^R c_{j_i^{r-1} j_i^r} \left(\prod_{n=0}^{r-1} q_{j_i^n} \right) \right) \quad (3-6)$$

公式(3-6)得到了一个客户 $i \in I$ 的期望运输成本, 所有客户的期望运输成本等于:

$$C_t = \sum_{i \in I} \lambda_i \left(c_{ij_i^0} + \sum_{r=1}^R c_{j_i^{r-1} j_i^r} \left(\prod_{n=0}^{r-1} q_{j_i^n} \right) \right) \quad (3-7)$$

至此, 关于客户的期望运输成本的推导已完成。考虑节点失效风险的物流网络选址问题可正式定义为: 给定一个客户集 I 和节点候选拓展集 \bar{J} , 以及弧的权重 $\{c_{ij} | \forall i \in I \cup \bar{J}, \forall j \in \bar{J}\}$, 节点的损坏概率 $q_j, \forall j \in \bar{J}$ 、建设成本 $f_j, \forall j \in \bar{J}$, 客户的需求 $\lambda_i, \forall i \in I$, 求一个子集 $J^* \subseteq \bar{J}$ 以及每个客户 i 访问子集 $J_i^* \subseteq J^*$ 的序列 $s_i = (j_i^0, \dots, j_i^r, \dots, j_i^R), \forall j_i^r \in J_i^*, \forall i \in I, r = 0, \dots, R$, 使得建设节点的固定成本 $C_f = \sum_{j \in J^*} f_j$ 与客户的期望运输成本 $C_t = \sum_{i \in I} \lambda_i \left(c_{ij_i^0} + \sum_{r=1}^R c_{j_i^{r-1} j_i^r} \left(\prod_{n=0}^{r-1} q_{j_i^n} \right) \right)$ 之和最小。

最后, 建模过程用到的所有符号定义如表3-1所示:

表 3-1 符号释义表

Table 3-1 Notation List

符号	解释
G	有向图
V	图 G 的点集
A	图 G 的弧集
I	客户点集
J	候选点集
λ_i	客户 $i \in I$ 的需求
f_j	节点 $j \in J$ 的建设成本
q_j	节点 $j \in J$ 的失效概率
c_{ij}	弧 (i, j) 的权重
π	惩罚价格
j_0	虚拟节点
R	每个客户备用节点的最大个数
\bar{J}	集合 J 与 $\{j_0\}$ 的并集
r	备用节点的等级
j_i^r	客户 i 的第 r 等级的节点
C_e^r	客户 i 访问第 r 等级备用节点的期望成本
C_t^i	客户 i 的总期望运输成本
C_t	总期望运输成本
C_f	总建设成本
J_j^+	可在 j 之前访问的节点集合
J_j^-	可在 j 之后访问的节点集合

3.3 模型构建

本节将构建考虑节点失效风险的物流网络选址问题的非线性混合整数规划模型，该模型的目标函数为最小化总成本，包括建设节点的固定成本（投资成本）和客户的期望运输成本，其中期望运输成本的计算过程已在3.2节推导。第3.3.1节明确了模型的目标函数，第3.3.2节描述了模型的约束。

3.3.1 目标函数推导

本小节将构建模型的目标函数，关于问题描述中符号的定义参见表3-1。在定义决策变量之前，为了表达客户访问节点的先后顺序，本章将采用两点之间是否存在弧的方法表示客户的试错方案。使用这种“弧”的概念借鉴了车辆路径问题(Vehicle Routing Problem, VRP)的决策变量处理思路^[83] 以及 Albareda 等人^[12] 研究可靠 P-中值问题对决策变量的处理。

由于每个客户从自身位置出发，因而起点不同。定义集合 J_{ij}^+ 和集合 J_{ij}^- 分别表示客户 i 在点 j 之前和之后可访问的点集，公式如下：

$$J_{ij}^+ = \begin{cases} \{i\} \cup J \setminus \{j\} & \text{if } j \neq j_0 \\ \{i\} \cup J & \text{if } j = j_0 \end{cases}, \forall i \in I, j \in J \quad (3-8)$$

$$J_{ij}^- = \begin{cases} \bar{J} \setminus \{j\} & \text{if } j \neq i \\ \bar{J} & \text{if } j = i \end{cases}, \forall i \in I, j \in J \quad (3-9)$$

上述公式 (3-8) 和公式 (3-9) 可解释为：对于客户 i ，只能从客户点 i 以及不包括节点 j 自身的任意实体节点访问节点 j ，但可从客户点 i 以及任意实体节点访问虚拟节点 j_0 。同理，对于客户 i ，客户从 i 出发后可以访问任意节点，但是访问节点 j 之后访问的下一个点不能再是节点 j 。

该模型共需要两种决策变量和一种中间变量，具体定义如下：

(1) $y_j, \forall j \in J$: 选址 0-1 决策变量，等于 1 表示在候选节点 $j \in J$ 处建设节点，等于 0 表示不在该处建设。

(2) $x_{ijk}, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^-$: 试错序列 0-1 决策变量，等于 1 表示弧 (j, k) 属于客户 i ，等于 0 表示弧 (j, k) 不属于客户 i 。

(3) $p_{ijk}, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^-$: 概率连续中间变量，上界等于 1 下界等于 0，表示客户 i 从节点 j 访问节点 k 的概率。

模型的目标函数是最小化总成本，包含建设节点的固定成本以及期望运营成

本，具体如公式 (3-10) 所示。

$$\min C = \sum_{j \in J} f_j y_j + \sum_{i \in I} \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in J} c_{kj} p_{ikj} x_{ikj} \quad (3-10)$$

公式 (3-10) 的第一部分计算了建设节点的成本，第二部分计算了期望运输成本。目标函数中期望运输成本是在公式 (3-6) 的基础上引入决策变量构成的。公式 (3-6) 的累乘项 $\prod_{n=0}^{r-1} q_{j_n^n}$ 被中间变量 p_{ikj} 替换。关于 p_{ikj} 的递推关系，详见约束 (3-15) 和约束 (3-16)。值得注意的是决策变量 x_{ikj} 和中间变量 p_{ikj} 相乘，因此该目标函数是非线性（二次）的。

3.3.2 数学模型

考虑节点失效风险的物流网络选址模型如下：

$$\min C = \sum_{j \in J} f_j y_j + \sum_{i \in I} \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in J} c_{kj} p_{ikj} x_{ikj} \quad (3-11)$$

s.t.

$$\sum_{k \in J_{ij}^+} x_{ikj} \leq y_j, \forall i \in I, j \in J \quad (3-12)$$

$$\sum_{j \in J_{ii}^-} x_{iij} = \sum_{j \in J_{ij_0}^+} x_{ij_0} = 1, \forall i \in I \quad (3-13)$$

$$\sum_{k \in J_{ij}^-} x_{ijk} = \sum_{k \in J_{ij}^+} x_{ikj}, \forall i \in I, j \in J \quad (3-14)$$

$$p_{iij} = x_{iij}, \forall i \in I, j \in J_{ii}^- \quad (3-15)$$

$$q_j \sum_{k \in J_{ij}^+} x_{ikj} p_{ikj} = p_{ijj'}, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (3-16)$$

$$\sum_{j \in J \cup \{i\}} \sum_{k \in J_{ij}^-} x_{ijk} \leq R, \forall i \in I \quad (3-17)$$

$$y_j \in \{0, 1\}, \forall j \in J \quad (3-18)$$

$$x_{ijk} \in \{0, 1\}, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^- \quad (3-19)$$

$$0 \leq p_{ijk} \leq 1, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^- \quad (3-20)$$

目标函数 (3-11) 表示最小化总成本。约束 (3-12) 规定只有已建设的备用节点才有可能成为某个客户的常用或备用节点；约束 (3-13) 表示客户出发点和终点的流平衡约束，客户必须出发前往某个节点，也必须从某个点到达虚拟节点（终点）；约束 (3-14) 是中间点的流平衡约束，规定进入节点的流量等于流出节点的流量；约

束 (3-15) 初始化每个客户出发对应弧的概率；约束 (3-16) 表达了两段相邻弧的概率递推关系；约束 (3-17) 限制了每个客户尝试的次数，即每个客户拥有的弧的个数；最后，约束 (3-18) 和 (3-19) 是决策变量 0-1 整数约束，约束 (3-20) 表示中间变量的上下界。

3.4 模型线性化

由于第3.3节给出的数学模型是非线性混合整数规划模型，本小节针对该模型提出了降低模型复杂程度的线性化方法。模型中的非线性的部分，分别出现在目标函数 (3-10) 和约束 (3-16) 中。采用等价替换方法消除模型中的二次项，令 $w_{ijk} = p_{ijk}x_{ijk}, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^-$ ，则目标函数 (3-11) 可替换为，

$$\min C = \sum_{j \in J} f_j y_j + \sum_{i \in I} \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in J} c_{kj} w_{ikj} \quad (3-21)$$

使用相同的方式，替换约束 (3-16) 中的非线性部分，则约束 (3-16) 变为，

$$q_j \sum_{k \in J_{ij}^+} w_{ikj} = p_{ijj'}, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (3-22)$$

需注意，当前的替换并不是等价的。 $w_{ijk}, \forall i \in I, j \in J, j' \in J_{ij}^-$ 作为中间变量，在替换时需补充一些等价约束，这些等价约束限制了中间变量的取值范围，保证替换的等价性。注意到变量 $x_{ijk}, \forall i \in I, j \in J, k \in J_{ij}^-$ 为 0-1 约束，变量 $p_{ijk}, \forall i \in I, j \in J, k \in J_{ij}^-$ 为 [0,1] 之间的连续变量。由于 $w_{ijk}, \forall i \in I, j \in J, k \in J_{ij}^-$ 等于对应变量相乘，那么 w_{ijk} 的上界将取二者的较小值，下界将大于二者之和减 1，即等价约束如下：

$$w_{ijk} \leq p_{ijk}, \forall i \in I, j \in J \cup \{i\}, j' \in J_{ij}^- \quad (3-23)$$

$$w_{ijk} \leq x_{ijk}, \forall i \in I, j \in J \cup \{i\}, j' \in J_{ij}^- \quad (3-24)$$

$$w_{ijk} \geq p_{ijk} + x_{ijk} - 1, \forall i \in I, j \in J \cup \{i\}, j' \in J_{ij}^- \quad (3-25)$$

$$0 \leq w_{ijk} \leq 1, \forall i \in I, j \in J \cup \{i\}, j' \in J_{ij}^- \quad (3-26)$$

考虑节点失效风险的物流网络选址模型的线性化版本如下：

$$\min C = \sum_{j \in J} f_j y_j + \sum_{i \in I} \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in J} c_{kj} w_{ikj} \quad (3-27)$$

s.t.

约束 (3-12) - 约束 (3-15)

约束 (3-17) - 约束 (3-20)

约束 (3-22) - 约束 (3-26)

3.5 模型性质

本节将对考虑节点失效风险的物流网络选址数学模型进行一系列性质分析。首先，第3.5.1小节简要说明了复杂度理论，并证明了该模型的 **NP-hard** 性质，其次，第3.5.2小节提出了该问题的一个子问题，在本文中称为试错序列问题，该问题是基本最短路问题更一般的形式。

3.5.1 NP-hard

根据复杂度理论，如果一个问题如果能在多项式时间 (Polynomial time) 内求解，那么这个问题归类为 **P** 问题 (Polynomial Problem)。如果一个问题可以在多项式时间内验证解的正确性，那么这个问题是非确定性多项式时间问题 (Nondeterministic Polynomial-time Problem) 即 **NP** 问题。其中，“非确定性”是指不知道这个解是怎样获得的，且很有可能是猜测的。

显然，**P** 问题是 **NP** 问题的子集，因为在多项式时间内能解决的问题，一定能在多项式时间内验证解的正确性。但是 **P** 问题是否等于 **NP** 问题至今没有准确的结论。

在 **NP** 问题中，有一类问题称为 **NP** 完全问题或 **NPC** 问题 (Nondeterministic Polynomial-time Complete Problem)。当一个问题时 **NP** 问题，且所有的 **NP** 问题都能多项式归约至这个问题时，那么这个问题是 **NPC** 问题。这说明 **NPC** 问题可以相互多项式归约转化。如果 **NPC** 问题存在多项式时间解法，则说明所有的 **NP** 问题都能在多项式时间内解决，那么就证明了 **P=NP** 问题。当前研究已为一些 **NPC** 问题找到了伪多项式时间 (Pseudo Polynomial-time algorithm) 算法（时间复杂度是问题规模的多项式，空间复杂度是问题规模的非多项式），这类问题称为弱 **NPC** 问题，另一类不存在伪多项式时间算法的 **NPC** 问题称为强 **NPC** 问题。

当一个问题满足“所有的 **NP** 问题都能多项式归约至这个问题”的条件时，那么这个问题称为 **NP-hard** 问题 (Nondeterministic Polynomial-time Hard Problem)。这说明 **NP-hard** 问题目前没有多项式时间算法，也说明 **NPC** 问题属于 **NP-hard** 问题。

在简单介绍完复杂度理论后，接下来将证明考虑节点失效风险的物流网络选址问题是 **NP-hard** 问题。本节将从集合覆盖问题逐步推导至 UFL 问题，再推导至本文研究的问题。集合覆盖问题是指一类经典的组合优化问题，该问题的决定性版本（即，回答为“是”或“否”的问题，例如有一个问题的解，判断其是否为最优解）是 Karp^[84] 定义的经典 21 个 **NPC** 问题之一，其优化版本（即，找到该问题最优解的优化过程）是 **NP-hard** 问题。集合覆盖问题是指给定一个全集 U 包含 m 个元素，集合 S 中包含了 n 个集合且这 n 个集合的并集等于 U ， $S = \{S_1, S_2, \dots, S_n\}, \bigcup_{i=1, \dots, n} S_i = U$ 。

在 S 中寻找一个最小的子集，使该子集的并集等于 U 。关于引理的证明可参考 Garey 等人^[85] 对于 NPC 问题与复杂度理论的详细论述。证明集合覆盖问题是 NPC 问题超过了本文的范畴，因此本文不再复述其具体的证明过程，而直接将其作为引理使用。

引理 1 集合覆盖问题的决定性版本是 NPC 问题，优化版本是一个 NP-hard 问题。

推论 1 无容量限制固定费用选址问题 (UFL) 是 NP-hard 问题。

证明 1 关于 UFL 问题论述所用的符号直接沿用了第3.2节的定义。基于集合覆盖问题的定义（符号同样沿用），做一个辅助图 H ， H 分为左右两个部分。对于左部分，若存在一个元素 $u \in U$ ，则左侧增加一个点 h_u 。对于右部分，若存在一个子集 $S_k \subseteq \{h_1, h_2, \dots, h_m\}$ ，则在右侧增加一个点 S_k ，子集数量上限为 n 。如果 $h_u \in S_k$ ，则存在一条边 $e_{h_u S_k}$ 。图 H 构建完成后，将在该图上进一步考虑一个 UFL 问题。令点集 $I := \{h_1, h_2, \dots, h_m\}$ 表示客户点，点集 $J := \{S_1, S_2, \dots, S_n\}$ 表示候选节点。令 UFL 问题中，固定成本为 1，即 $f_j = 1$ ，若边 $e_{h_u S_k}$ 存在，则 $c_{h_u S_k} = 0$ 。这样，一个 UFL 问题变为以最少的节点数（ H 中右侧）使得每个客户 (h_u) 可以被与之相连的节点 (S_k ，且 S_k 包含 h_u) 服务。该 UFL 问题就转换成了一个集合覆盖问题。显然，当且仅当集合覆盖问题取最优时，这个特殊的 UFL 问题取最优。

上述的转换过程，称为多项式归约。它使得一个 UFL 问题转换成一个已知的 NP-hard 问题，进而证明了 UFL 问题是一个 NP-hard 问题。从 UFL 问题出发，可推导考虑节点失效风险的物流网络选址问题是 NP-hard 问题。

推论 2 考虑节点失效风险的物流网络选址问题是 NP-hard 问题。

证明 2 在第3.3节构建的模型中，令 $R=1$ ，则每个客户只有一个常用节点可以使用，无可选的实体备用节点。由于常用节点的失效概率已知，则相关价格的期望值也已知。那么该问题退化成一个 UFL 问题，并且由推论 1 可知 UFL 问题是 NP-hard 问题，那么未退化的原问题也是 NP-hard 问题。换句话说，该问题拓展了 UFL 问题，进而也是 NP-hard 问题。该证明详情可参考 Li^[22] 等人的研究。

3.5.2 子问题：试错序列问题

根据客户的试错策略，本文提出了考虑节点失效风险的物流网络选址问题的一个子问题，本文称作试错序列问题，是指在给定已建成的节点集合 J^* 中，为每个客户指派一条期望成本最低的试错序列。研究试错序列问题的解法可以带来两方面的好处。首先，当求解主问题的启发式算法给出了节点选址方案时，快速求解试错序列问题可以提升评估方案质量的效率。其次，假设所有的节点均可用并

增加访问节点的一项固定成本时，该问题则是主问题的拉格朗日松弛形式，这将在第4章中介绍。

该问题的正式定义如下：给定已建设的节点集合 J^* ，客户集合 I ，虚拟节点 j_0 以及弧的权重 $\{c_{ij} | \forall i \in I \cup J^*, \forall j \in J^*\}$ ，节点的损坏概率 $q_j, j \in J^*$ ，客户的需求 $\lambda_i, \forall i \in I$ ，每个客户最大尝试次数 R ，对于每个客户 $i \in I$ 来说，求从 i 出发访问 J^* 中的节点，并前往 j_0 的总期望运输成本最小的序列，并且该序列的中虚拟节点和已建设节点的总数不超过 R 。图3-1展示了这种试错序列的起点与终点定义，此外，期望运输成本的计算方法参见公式(3-6)。同样，本节给出试错序列问题一般形式的数学模型，为了方便表述，该模型使用的符号继承表3-1的定义。不失一般性地假设 $J^* = J$ ，即所有的设施均被建设。给定一个客户点 $i \in I$ ，定义 0-1 决策变量 $z_{kj}, \forall j \in \bar{J}, \forall k \in J_{ij}^+$ 等于 1 时表示客户 i 经过该弧 (k, j) ，等于 0 时表示不经过；连续 $[0, 1]$ 决策变量 $p_{kj}, \forall j \in \bar{J}, \forall k \in J_{ij}^+$ 表示客户 i 经过弧 (k, j) 的概率，试错序列模型如下：

$$\min C_t^i = \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in \bar{J}} c_{kj} p_{kj} z_{kj} \quad (3-28)$$

s.t.

$$\sum_{j \in J_{ii}^-} z_{ij} = \sum_{j \in J_{j_0}^+} z_{j_0} = 1 \quad (3-29)$$

$$\sum_{k \in J_{ij}^-} z_{jk} = \sum_{k \in J_{ij}^+} z_{kj}, \forall j \in J \quad (3-30)$$

$$p_{ij} = z_{ij}, \forall j \in J_{ii}^- \quad (3-31)$$

$$q_j \sum_{k \in J_{ij}^+} z_{kj} p_{kj} = p_{jj'}, \forall j \in J, j' \in J_{ij}^- \quad (3-32)$$

$$\sum_{j \in J \cup \{i\}} \sum_{k \in J_{ij}^-} z_{jk} \leq R \quad (3-33)$$

$$z_{jk} \in \{0, 1\}, \forall j \in \bar{J}, \forall k \in J_{ij}^+ \quad (3-34)$$

$$0 \leq p_{ijk} \leq 1, \forall j \in \bar{J}, \forall k \in J_{ij}^+ \quad (3-35)$$

模型的目标(3-28)是最小化客户 i 的总期望运输成本；约束(3-29)和约束(3-30)是流出发、结束和守恒约束；约束(3-31)和约束(3-32)是概率初始化和递推关系；约束(3-33)规定客户的试错次数；约束(3-34)和约束(3-35)是整数约束和连续变量的上下界约束。注意第3.4节的线性化技术同样适用于求解该问题，在此不再赘述试错序列模型的线性化版本。

试错序列问题本质上属于最短路问题的变种，是资源受限的基本最短路问题(Elementary Shortest Path Problem with Resource Constraints, ESPPRC)的特殊形式。

该特殊形式的 ESPPRC 的定义如下：给定一个图 $H = (V_H, E_H)$ ，其中 $s \in E_H$ 是源点， $t \in E_H$ 是汇点，每个点 $v \in E_H$ 的需求为 λ_v ，每条边 $e_H \in E_H$ 的距离为 c_{e_H} ，求一条从 s 到 t 的路径，使得该路径的总距离之和最小的同时，路径上所有经过的点的需求和不超过 Q 。

注意上述定义是 ESPPRC 问题的特殊形式，仅考虑了容量限制。事实上，ESP-PRC 问题中每个点有时间窗限制，但在本节讨论的问题中并无该限制，因此做出了简化（可将每个点的时间窗无限放宽以消除时间窗）。有关 ESPPRC 详尽的阐述与求解方法，可见参考文献^[86]。给定一个客户 $i \in I$ ，令 $s = i$, $t = j_0$, $\lambda_v = 1$, $c_{e_H} = c_{ij}$, $Q = R$, $q_j = 1, \forall j \in J^*$ 则试错序列问题等价于 ESPPRC 问题。由于 ESPPRC 问题已被证明是 NP-Hard 问题^[87]，进而试错序列问题也是 NP-Hard 问题。这说明当前不存在求解该问题的多项式时间算法。

综上，试错序列问题是基本最短路问题的一个变种，每个节点附加的失效概率和期望目标成本增加了求解的难度。本文的第4章设计的拉格朗日松弛算法在求解上界与下界时，都不可避免地多次求解这个问题，求解该问题的算法将在第4.2.2节详细阐述。

3.6 本章小结

本章的核心内容是构建考虑节点失效风险的物流网络选址模型。在建立模型之前，使用数学语言的严谨描述了该问题，并对客户的试错行为进行刻画，推导了由此产生的期望运输成本。引入决策变量构建了二次约束二次目标的混合整数规划模型，使用线性化的方法消除了模型中非线性的部分，使得精确求解模型的难度降低。为了方便后续算法开发，本节研究了客户试错序列子问题，开发求解该子问题的有效算法可加速求解主问题的过程。最后，由于该问题被证明为 NP-hard 问题，为了求解大规模算例，第4章开发了基于迭代局部搜索改进的拉格朗日松弛算法。

(本页留空)

4 基于迭代局部搜索改进的拉格朗日松弛算法设计

拉格朗日松弛算法 (Lagrangian Relaxation, 以下简称“LR 算法”) 是一种基于数学优化过程的启发式方法，是求解整数规划问题的有效手段之一。相较于精确解法 (分支定界、分支切割等)，LR 算法可以在短时间内获得质量较优的近似最优解；相较于元启发式算法 (遗传算法、禁忌搜索等)，LR 算法提供了优化边界 (上界和下界) 进而可以证明解的质量。在求解上界时，迭代局部搜索 (Iterated Local Search, ILS) 算子进一步强化上界质量。ILS 算子是一种提升上界质量的方法，其特长并不是全局搜索的能力，而是作为其他算法的嵌入过程，提升算法整体性能。ILS 是一种易于理解的元启发式方法，但却有强大的局部搜索能力^[88]。

本章为第3章的模型开发了基于迭代局部搜索改进的拉格朗日松弛 (Lagrangian Relaxation and Iterated Local Search, LR-ILS) 混合算法，该算法基于标准 LR 算法框架并将 ILS 算子嵌入其中以改进上界质量。第4.1节先推导了拉格朗日松弛的基本原理，第4.2节介绍了该算法的框架，并详细阐述了上下界获取、乘子更新等细节，第4.3节总结了本章的内容。本章中涉及的代码均在作者的 Github 开源仓库中 (地址见附录 A)，此外本文附录 A 提供了全部代码，供有需要的读者下载。

4.1 LR 算法原理

LR 算法并非拉格朗日提出，而是后人在求解优化问题时，借鉴了拉格朗日乘数法的思路，经过进一步概括总结，命名为拉格朗日松弛。有关 LR 算法简要的发展历程如下：LR 算法可追溯至 1963 年 Everett^[89] 使用拉格朗日乘数法求解离散问题，1973 年 Fisher^[90] 使用了该算法求解了调度问题，Geoffrion^[51] 正式命名该算法为“拉格朗日松弛”。LR 算法已经在求解调度问题^[91]、选址问题^[19] 发挥了巨大作用。本节将从一个一般的整数规划问题出发，概述 LR 算法的基础原理。更多有关 LR 算法的介绍及应用，可参考 Fisher^[55] 关于拉格朗日松弛的入门指导，孙小玲^[49] 有关整数规划著作中关于拉格朗日对偶理论的推导，以及优化百科全书^[50] 的中关于拉格朗日松弛及拉格朗日对偶的概述。

给定一个待求解的整数规划问题 P，形式如下：

$$\begin{aligned}
 Z &= \min cx \\
 \text{s.t. } Ax &= b \\
 Dx &\leq e \\
 x &\geq 0 \text{ and integral}
 \end{aligned} \tag{4-1}$$

松弛 P 的整数约束，得到的线性松弛问题定义为 LP。其中， A 是 m 行 n 列的矩阵， x 是 n 行的向量，其他所有向量的维度与之对应。在 P 问题中，第一个约束是强约束，而第二个约束是弱约束。为了处理强约束，使得问题容易求解，LR 算法松弛了第一个约束，并将其与拉格朗日乘子 μ 相乘后添加至目标函数中。这种处理方法借鉴了拉格朗日乘子法求多元函数极值的思路。

定义得到的松弛后的问题为 LR_u ，其中， $u = \{u_1, u_2, \dots, u_m\}$ 是拉格朗日乘子：

$$\begin{aligned} Z_D(u) &= \min c x + u(Ax - b) \\ \text{s.t. } &Dx \leq e \\ &x \geq 0 \text{ and integral} \end{aligned} \tag{4-2}$$

那么，一定有 $Z_D(u) \leq Z$ 。证明如下：已知 P 问题的可行解一定是 LR_u 问题的可行解。由于 LR_u 相较于 P 问题松弛了约束，因此可行域扩大。令 x^* 是 P 问题的最优解，那么 x^* 未必是 LR_u 的最优解（在 LR_u 的可行域内，但不在 P 的可行域内，可能存在比 x^* 更优的、使得 LR_u 目标函数值更小的解）。因此，有 $Z_D(u) \leq cx^* + u(Ax^* - b)$ 。当 $x = x^*$ 时，有 $Ax^* = b$ ，即 $Ax^* - b = 0$ 。那么 $Z_D(u) \leq cx^* = Z$ ，证毕。

如果 $Ax = b$ 为不等形式，则通过控制 u 的正负仍可以满足上述不等性质。当 $Ax \leq b$ ，则令 $u \geq 0$ ，有 $Z_D(u) \leq cx^* + u(Ax^* - b) = Z$ 。反之同理，当 $Ax \geq b$ ，则令 $u \leq 0$ 。

在特定的 u 的取值下， $Z_D(u)$ 等于 Z 。如何给 u 取值，使得 $Z_D(u)$ 的值尽可能大、尽可能接近 Z 的问题就变成了一个和 u 相关的问题。该问题成为拉格朗日对偶问题 D，即：

$$Z_D = \max_u Z_D(u) \tag{4-3}$$

该问题的决策变量是 u ，目标函数是尽可能最大化 $Z_D(u)$ 的值，由于 $Z_D(u)$ 不可能超过 Z 的值，因此求 Z_D 的过程就是在不断调整 u 的值，使 $Z_D(u)$ 的值逼近 Z 过程。当前，D 问题是一个抽象的形式，采用如下方法重构该问题。

假设拉格朗日松弛 LR_u 的可行解集 $X = \{x | Dx \leq e, x \geq 0 \text{ and integer}\}$ 是有限集，则我们可以令 $X = \{x^t, t = 1, \dots, T\}$ ，令 $w = Z_D(u)$ 。 $Z_D(u)$ 是最小化 $cx + u(Ax - b)$ ，那么 $Z_D(u)$ 一定小于等于 LR_u 的任意一个可行解 x^t 带入该表达式的值，即 $Z_D(u) = w \leq cx^t + u(Ax^t - b)$ ，当 x^t 是 LR_u 问题的最优解时等号成立。因此，D 问题重构为 \bar{D} ：

$$\begin{aligned} Z_D &= \max w \\ \text{s.t. } &w \leq cx^t + u(Ax^t - b), t = 1, \dots, T \end{aligned} \tag{4-4}$$

问题 \bar{D} 的线性对偶 \bar{P} 的形式如下：

$$\begin{aligned}
 Z_P = & \min \sum_{t=1}^T \lambda_t c x^t \\
 \text{s.t. } & \sum_{t=1}^T \lambda_t A x^t = b \\
 & \sum_{t=1}^T \lambda_t = 1 \\
 & \lambda_t \geq 0, t = 1, \dots, T
 \end{aligned} \tag{4-5}$$

显然 \bar{P} 不是 P 的线性松弛，因此 \bar{P} 和 LP 并不是等价问题。由于 x^t 是 LR_u 的可行解，则 x^t 是一定满足约束 $Dx \leq e$ 。在所有满足 $Dx \leq e$ 的解中，一定存在某个解满足 $Ax = b$ 。因此，当 λ_t 是整数时， \bar{P} 等价于 P 。

问题 \bar{D} 表明， $Z_D(u)$ 是一系列线性函数的值的下限。为可视化理解该过程，假设 x 的维度为 1， T 的值取 4，即 LR_u 有 4 个可行解。 $Z_D(u)$ 随 u 变化如图4-1所示，该图源自文献^[55]。图中加粗的黑线的表示 $Z_D(u)$ 的取值，表示在每个 u 固定的情况下所有线性表达的最小值。注意图中 x^t 是已知的常数。在这个简单例子中， u 的线性组合是一条直线，在一般情况下是多维空间的超平面，上述 $Z_D(u)$ 取值操作相当于求下包络面（线）。由此可知， $Z_D(u)$ 是连续且凹的（参考国际定义）但不可微的。求 $Z_D(u)$ 的最大值等价于求不可微的优化问题，本文采用了次梯度法进行乘子更新，这是一种经典的乘子更新的方法。

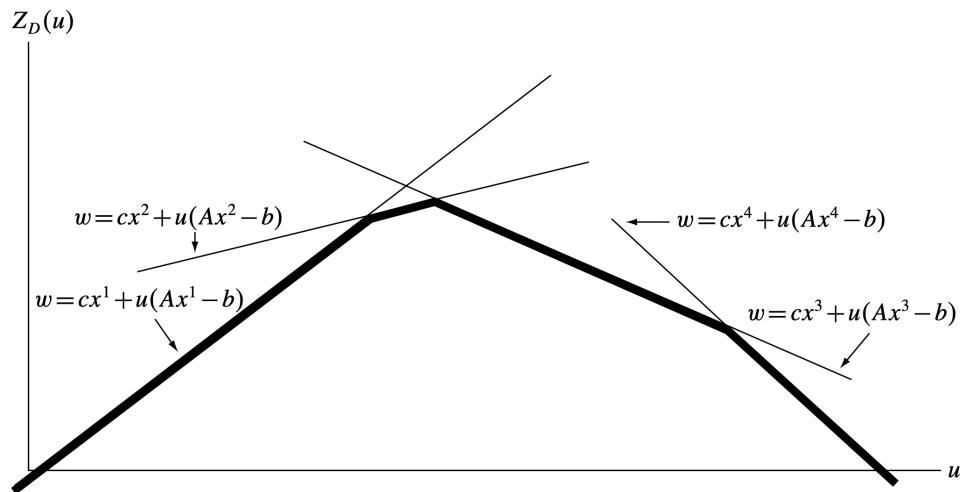


图 4-1 $Z_D(u)$ 随 u 的变化 (图中加粗黑线)^[55]

Fig 4-1 The value of $Z_D(u)$ as a function of u ^[55]

给定 u 的初始值 u_0 ，后续乘子第 k 次迭代的值为 u^k ，迭代公式如下：

$$u^{k+1} = u^k + t_k(Ax^k - b) \tag{4-6}$$

其中，其中 x^k 是 LR_{u^k} 的最优解， t_k 是一个正数，称为迭代步长。括号内的项 $Ax^k - b$ 称作迭代方向。Geoffrion^[51] 证明了当 $t_k \rightarrow 0$ 且 $\sum_{i=0}^k t_i \rightarrow \infty$ 时， $Z_D(u^k) \rightarrow Z_D$ 。常用的迭代步长公式如下：

$$t_k = \frac{\alpha_k(Z^* - Z_D(u^k))}{\|Ax^k - b\|^2} \quad (4-7)$$

其中， α_k 是一个 $(0,2]$ 之间的标量， Z^* 是 Z_D 的上界，通常由启发式获得。 α_k 的初始值一般设置为 2，当一定迭代次数后， $Z_D(u)$ 的值仍不能得到提升，则 α_k 的值减半。

采用上述的次梯度法，不断迭代 u^k 的值，然后求解 $Z_D(u^k)$ ，使得松弛问题的目标函数值不断接近原问题的最优值。需注意，由于松弛了 P 问题的约束，因此 $Z_D(u^k)$ 的解对于 P 问题来说往往是不可行的。通常采用启发式算法对 $Z_D(u^k)$ 的解的进行修正，以获得 P 问题的可行解。

至此，本节推导了 LR 算法的原理。第4.2节将根据该原理，为考虑节点失效风险的物流网络选址模型设计 LR-ILS 算法框架。

4.2 算法设计

本节将介绍采用 LR-ILS 算法求解考虑节点失效风险的物流网络选址模型的框架，根据4.1节所述，LR 算法主要包括获取下界（求解松弛模型）、获取上界、更新乘子等步骤。注意，第3.4节中针对原模型进行了线性化处理，所有推导内容均基于模型的线性化版本。本节的 LR-ILS 算法框架参考了 Snyder^[4]，Yun 等人^[19,21]的文章，以及 Daskin^[7] 使用 LR 关于求解 UFL 的方法。

为了便于理解，图4-2直观展示了 LR-ILS 的迭代流程，图4-3提供了实现该算法的伪代码。如图4-3所示，在设计-ILS 算法之前，需要先构造原问题 M 的松弛问题 RM，第4.2.1节介绍了如何松弛问题 M 得到 RM 以及选取约束进行松弛的方法。初始化已知最佳上界 ub^* 为正无穷，最佳选址方案 Y^* 为空，乘子初始化方法见第4.2.4节。LR-ILS 算法迭代优化，每次迭代使用精确方法获得下界（第4.2.2节），根据下界解的选址方案构造启发式以获取上界可行解 ub （第4.2.3节），当上界解 ub 优于 ub^* 时，则更新 ub^* 。每次迭代中，LR-ILS 算法使用次梯度法更新乘子，第4.2.4节介绍了乘子更新的方法。当 ILS 条件被触发时，ILS 算法对当前解进行改进（第4.2.6节）。若算法满足第4.2.5节的任意条件之一时，则退出迭代。

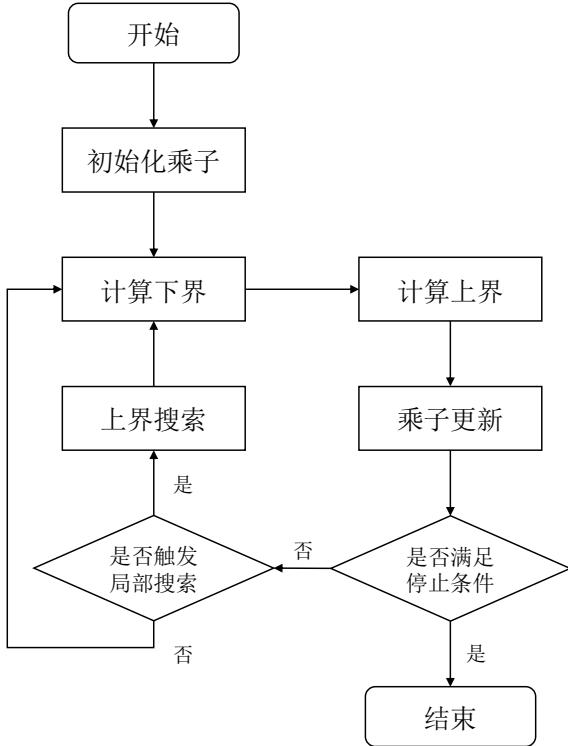


图 4-2 LR-ILS 流程图
Fig 4-2 Flow chart for LR-ILS

Input: 原模型 M, 松弛模型 RM	▷ 参见第4.2.1节
1: $\mu :=$ 初始化乘子 ()	▷ 参见第4.2.4节
2: $ub^* := +\infty$	
3: $Y^* := \emptyset$	
4: repeat	
5: 下界 lb , 选址方案 $Y :=$ 下界算法 ($RM(\mu)$)	▷ 参见第4.2.2节
6: 上界 $ub :=$ 上界算法 ($M(Y)$)	▷ 参见第4.2.3节
7: if $ub < ub^*$ then	
8: $ub^* := ub$	
9: $Y^* := Y$	
10: end if	
11: $\mu :=$ 更新乘子 (μ, lb, ub^*)	▷ 参见第4.2.4节
12: if ILS 条件触发 then	
13: $ub^* := ILS(ub^*)$	▷ 参见第4.2.6节
14: end if	
15: until 满足终止条件	▷ 参见第4.2.5节
16: return ub^*, Y^*	

图 4-3 LR-ILS 算法伪代码
Fig 4-3 Pseudocode for LR-ILS

4.2.1 约束松弛

一般情况下，松弛不同的约束产生的效果不同，选取特定的约束进行松弛可加速求解过程。Fisher^[55]提出了两个选取松弛约束的准则：下（上）界的质量以及获得下（上）界所需计算量。这两个准则相互矛盾的，获得更高质量的界需要更多的计算量，较少的计算量生成的解的质量不佳。因此，不同学者使用拉格朗日松弛求解选址问题时，构造松弛问题的方法也不尽相同，以权衡求解时长与求解质量。Daskin^[7] 使用拉格朗日求解 UFL 问题中松弛了关于所有客户都必须由某一个设施服务的约束，对应本文的约束 (3-12)。Snyder 等人^[4] 求解 RUFL 问题松弛了每个客户的每一级都需要指派一个备用设施的约束，对应本文的约束 (3-17)。Yun 等人^[21] 松弛了已建成的设施才能成为客户的某一级的实体设施的约束，对应本文的约束 (3-12)。

本文采用了 Yun 等人^[21] 的松弛方法，松弛约束 (3-12)。使用该方法获得的松弛问题具有可继续拆分的特殊结构，利用此特性可以加快获取下界的过程。第4.2.2节介绍了松弛问题的可拆分的特性以及获取下界的方法。

4.2.2 下界获取

令原始模型为 M ，设拉格朗日乘子 $\mu = \{\mu_{ij} \geq 0 | \forall i \in I, j \in J\}$ ，并将松弛的约束 (3-12) 与乘子相乘后与目标函数 (3-27) 相加，得到拉格朗日对偶问题的目标函数：

$$\begin{aligned} \max_{\mu} \min_{x,y,w} C_D &= \sum_{j \in J} f_j y_j \\ &+ \sum_{i \in I} \lambda_i \sum_{k \in J_{ij}^+} \sum_{j \in J} c_{kj} w_{ikj} \\ &+ \sum_{i \in I} \sum_{j \in J} \mu_{ij} \left(\sum_{k \in J_{ij}^+} x_{ikj} - y_j \right) \end{aligned} \quad (4-8)$$

整理可得，

$$\begin{aligned} \max_{\mu} \min_{x,y,w} C_D &= \sum_{j \in J} (f_j - \sum_{i \in I} \mu_{ij}) y_j \\ &+ \sum_{i \in I} \sum_{j \in J} \sum_{k \in J_{ij}^+} (\lambda_i c_{kj} w_{ikj} + \mu_{ij} x_{ikj}) \\ &+ \sum_{i \in I} \sum_{k \in J_{ij_0}^+} \lambda_i c_{kj_0} w_{ikj_0} \end{aligned} \quad (4-9)$$

固定一组 μ 的值, 得到的松弛模型 $\text{RM}(\mu)$ 如下, 为了方便阅读, 本节给出了 $\text{RM}(\mu)$ 的约束, 这些约束与第3.4节中的线性约束是一致的。

$$\begin{aligned} \min_{x,y,w} C_D(\mu) = & \sum_{j \in J} (f_j - \sum_{i \in I} \mu_{ij}) y_j \\ & + \sum_{i \in I} \sum_{j \in J} \sum_{k \in J_{ij}^+} (\lambda_i c_{kj} w_{ikj} + \mu_{ij} x_{ikj}) + \sum_{i \in I} \sum_{k \in J_{ij_0}^+} \lambda_i c_{kj_0} w_{ikj_0} \end{aligned} \quad (4-10)$$

s.t.

$$\sum_{j \in J_{ii}^-} x_{iij} = \sum_{j \in J_{ij_0}^+} x_{ijj_0} = 1, \forall i \in I \quad (4-11)$$

$$\sum_{k \in J_{ij}^-} x_{ijk} = \sum_{k \in J_{ij}^+} x_{ikj}, \forall i \in I, j \in J \quad (4-12)$$

$$p_{iij} = x_{iij}, \forall i \in I, j \in J_{ii}^- \quad (4-13)$$

$$q_j \sum_{k \in J_{ij}^+} w_{ikj} = p_{ijj'}, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (4-14)$$

$$\sum_{j \in J \cup \{i\}} \sum_{k \in J_{ij}^-} x_{ijk} \leq R, \forall i \in I \quad (4-15)$$

$$w_{ijk} \leq p_{ijk}, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (4-16)$$

$$w_{ijk} \leq x_{ijk}, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (4-17)$$

$$w_{ijk} \geq p_{ijk} + x_{ijk} - 1, \forall i \in I, j \in J \cup \{i\}, j' \in J_{ij}^- \quad (4-18)$$

$$0 \leq w_{ijj'r} \leq 1, \forall i \in I, j \in J, j' \in J_{ij}^- \quad (4-19)$$

$$x_{ijk} \in \{0, 1\}, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^- \quad (4-20)$$

$$0 \leq p_{ijk} \leq 1, \forall i \in I, j \in J \cup \{i\}, k \in J_{ij}^- \quad (4-21)$$

$$y_j \in \{0, 1\}, \forall j \in J \quad (4-22)$$

对于给定的任意 μ 的值, $\text{RM}(\mu)$ 的最优解一定是 \mathbf{M} 的下界, 相关证明见4.1节。根据目标函数 (4-10) 的结构, 松弛模型 $\text{RM}(\mu)$ 可拆分成两组子模型 $\text{RM}(\mu)_{sub1}$ 和 $\text{RM}(\mu)_{sub2}$ 。其中, $\text{RM}(\mu)_{sub1}$ 如下,

$$\min_y \sum_{j \in J} (f_j - \sum_{i \in I} \mu_{ij}) y_j \quad (4-23)$$

s.t.

约束 (4-22)

$\text{RM}(\mu)_{\text{sub2}}$ 如下,

$$\max_{\mu} \min_{x,w} \sum_{i \in I} \sum_{j \in J} \sum_{k \in J_{ij}^+} (\lambda_i c_{kj} w_{ikj} + \mu_{ij} x_{ikj}) + \sum_{i \in I} \sum_{k \in J_{i0}^+} \lambda_i c_{k0} w_{ikj_0} \quad (4-24)$$

s.t.

约束 (4-11) - 约束 (4-21)

在给定任意一组 μ 的值的情况下, $\text{RM}(\mu)_{\text{sub1}}$ 将变得十分容易求解, 即给定一个最小化问题 $\min_y \sum_{j \in J} (f_j - \sum_{i \in I} \mu_{ij}) y_j$, 其中 $y_j, \forall j \in J$ 是 0-1 变量。显然, 当 $f_j - \sum_{i \in I} \mu_{ij} \geq 0$ 时, $y_j = 0$; 当 $f_j - \sum_{i \in I} \mu_{ij} < 0$ 时, $y_j = 1$ 。

在给定任意一组 μ 的值的情况下, $\text{RM}(\mu)_{\text{sub2}}$ 还可进一步拆分成 $|I|$ 个独立模型, 每个子模型 $\text{RM}(\mu)_{\text{sub2}}^i, \forall i \in I$ 是试错序列问题更一般的形式, 该问题的具体表述见第3.5.2节。 $\text{RM}(\mu)_{\text{sub2}}^i$ 与第3.5.2节中介绍的问题的不同之处在于 $\text{RM}(\mu)_{\text{sub2}}^i$ 的目标函数不仅计算了期望成本, 还计算了关于 μ 的项。不妨将 $\mu_{ij}, \forall i \in I, j \in J$ 看作是客户 i 访问节点 j 的固定访问成本, 那么, $\text{RM}(\mu)_{\text{sub2}}^i$ 可理解为从 i 点出发, 寻找一条到 j_0 的期望成本与固定访问成本最小的路径, 并且该路径经过的虚拟节点和实体节点总数不得大于 R 个。

本节已经给出了 $\text{RM}(\mu)_{\text{sub2}}$ 的线性混合整数规划模型, 使用上述的拆分方法固定 $i \in I$ 的值即可获得 $\text{RM}(\mu)_{\text{sub2}}^i$, 然后可使用求解器求解 $\text{RM}(\mu)_{\text{sub2}}^i$ 。当 $\mu = 0$ 时, $\text{RM}(\mu)_{\text{sub2}}^i$ 等价于第3.5.2节中介绍的试错序列问题。 $\text{RM}(\mu)_{\text{sub2}}^i$ 实质上是一个广义的最短路问题, 由于该问题是 NP-hard 问题, 使用精确求解方法的计算时间较长, 而 LR 算法会多次迭代并反复调用求解该问题的方法, 因此本节给出两种快速求解 $\text{RM}(\mu)_{\text{sub2}}^i$ 的启发式算法以及一个精确方法。启发式方法生成一个近似解, 精确方法在这个近似解的基础上进行改进, 可加快求解速度并获取下界的精确值。

(1) 插入启发式

已知客户 i 一定会从点 i 出发最终前往虚拟节点 j_0 , 因此可初始化一个从客户 i 到虚拟设施 j_0 的初始序列 s_i 。在该方法中, s_i 的期望成本与固定访问成本之和, 即目标函数 (4-24), 记作 $C_{lb}(s_i)$ 。定义 s_i 中连续两点之间的位置集合为 $P(s_i)$ 。插入启发式算法的步骤如下:

Step 0 初始化序列 $s_i := [i, j_0]$, 计算 $C_{lb}(s_i)$ 。

Step 1 向 s_i 中每个位置 $\rho \in P(s_i)$ 添加节点 $j \in J$, 并计算成本 $C_{lb}^{\rho j}(s_i)$ 。

Step 2 计算 $\Delta_{\rho j} = C_{lb}(s_i) - C_{lb}^{\rho j}(s_i)$ 。

Step 3 若所有 $\Delta_{\rho j} < 0$, 则结束; 否则将 $\Delta_{\rho j}$ 最大值对应的节点 j^* 插入 s_i 的 p^* 位置, 更新 s_i , $P(s_i)$, $C_{lb}(s_i)$ 以及令 $J := J \setminus \{j^*\}$ 。

Step 4 若 $J = \emptyset$ 或 $|s_i| = R + 1$, 则结束, 否则重复 Step 1-4。

插入启发式从一个仅包含客户和虚拟节点的完整的初始序列出发，因此初始序列的总成本等于惩罚成本。每次插入实体节点之前，评估每个节点插入每个位置之后得到的结果，然后将实体节点插入序列中当前阶段的最佳位置，使得总成本快速下降，直至不能插入任何节点。

(2) 构造启发式

已知客户 i 一定会从点 i 出发前往虚拟节点 j_0 。构造启发式仅考虑每一阶段的最优，向序列末尾逐个增加节点，具体步骤如下：

- Step 0 初始化序列 $s_i := [i]$ ，令 $C_{lb}(s_i) = 0$ 。
- Step 1 计算所有点 $j \in \bar{J}$ 添加至 s_i 末尾的增量成本 $\Delta_j = C_{lb}(s_i + \{j\}) - C_{lb}(s_i)$ 。
- Step 2 令 $j^* = \arg \min \Delta_j$ ，更新 $s_i := [s_i, j^*]$ 和 $C_{lb}(s_i)$ ， $J := J \setminus \{j^*\}$ 。
- Step 3 若 $j^* = j_0$ ，则结束；若 $|s_i| = R$ ，则 $s_i := [s_i, j_0]$ ，计算 $C_{lb}(s_i)$ 并结束，否则重复 Step 1-3。

构造启发式从一个仅含客户的不完整路径出发，每次向路径的末尾增加一个节点。每次增加的过程都将增量成本最小的客户放在末尾，直至添加了虚拟节点（构成了完整路径）或不再能添加节点（此时在末尾补充一个虚拟节点构成一个完整路径）。

上述两种算法均基于贪心算法，插入启发式算法先接受高昂成本，再插入节点以降低总成本。插入启发式类似节约算法，初始成本很高，成本随节点插入逐渐降低。构造启发式算法从无到有，逐渐拓展路径的长度，每次拓展将增量成本最低的节点加入路径末尾。插入启发式算法时间复杂度等于 $O(m^2n)$ ，构造启发式算法时间复杂度等于 $O(mn)$ ，其中 $m = R$ 是客户拥有实体节点个数的上限， $n = |\bar{J}|$ 是备选点的数量。

需要注意，在给定 μ 的值的前提下，上述的两种方法仅可以在多项式时间内获得近似最优解，而不一定可以获得最优解。这将导致 $RM(\mu)_{sub2}$ 不能获得最优解。已知只有在子问题获得最优的前提下，才能保证下界小于上界（证明见第4.1节）。因此，采用上述方法获得的下界很有可能在最终收敛时超过上界，使算法无法证明上界的质量。

求解最短路问题经典的精确算法包括标签算法^[92]、脉冲算法^[86]等。但这些算法不能直接应用于试错序列问题，因为这些算法解决的问题中每个点并不存在失效概率，且总成本等于路径长度简单和。由于上述的启发式算法可以获得一个近似最优解，本文将上述启发式算法获得的近似最优解带入深度优先搜索算法 (Deep First Search, DFS)，并设计剪枝策略以求解试错序列问题的最优解。

(3)DFS 算法

DFS 算法的伪代码如图4-4所示。DFS 类似于构造启发式，从客户点出发，每

一次向下分支就将一个节点添加至路径末尾，直至遍历完这个分支的所有可能，再选择该分支的某个分支继续遍历，如此递归直至遍历完问题的所有可能。对于该问题来说，递归的最大深度是有限的，判定某个路径 s_i 是否到达最大深度的准则有三个：(1) $\bar{J} = \emptyset$ ；(2) $|s_i| = R$ ，此时 s_i 只能以 j_0 结尾；(3) s_i 的已经以 j_0 结尾。当搜索满足上述三种情况之一时，不再继续分支（到达叶子节点）。然后计算路径成本 $C_{lb}(s_i)$ 。当 $C_{lb}(s_i)$ 小于已知最优成本 $C_{lb}(s_i^*)$ ，则更新已知最优路径和已知最优成本。DFS 分支的过程采用了递归调用，当搜索未达到最大搜索深度时，继续向下分支的过程中，保存当前的父节点并在遍历完子节点及所有分支后恢复。分支过程中，若当前路径的成本 $C_{lb}(s_i + \{j\})$ 大于已知最优成本 $C_{lb}(s_i^*)$ 时，则不再继续分支（剪枝）。

```

Input: 序列  $s_i$ , 成本  $C_{lb}(s_i)$ , 最优路线  $s_i^*$ , 最优成本  $C_{lb}(s_i^*)$ , 当前概率  $p$ , 节点集合  $\bar{J}$ 
1: if  $s_i$  已达到最大深度 &  $C_{lb}(s_i) < C_{lb}(s_i^*)$  then
2:    $s_i^*, C_{lb}(s_i^*) :=$  更新  $(s_i, C_{lb}(s_i), s_i^*, C_{lb}(s_i^*))$ 
3:   return  $s_i^*, C_{lb}(s_i^*)$ 
4: else
5:   for  $j \in \bar{J}$  do
6:     stack := 记录  $(s_i, C_{lb}(s_i), p)$ 
7:     if  $C_{lb}(s_i + \{j\}) < C_{lb}(s_i^*)$  then % 剪枝
8:        $s_i := s_i + \{j\}$ 
9:        $\bar{J} := \bar{J} \setminus \{j\}$ 
10:       $p := p * q_j$ 
11:       $s_i^*, C_{lb}(s_i^*) :=$  DFS( $s_i, C_{lb}(s_i), s_i^*, C_{lb}(s_i^*), p, \bar{J}$ ) % 递归
12:       $s_i, C_{lb}(s_i), p :=$  恢复 (stack)
13:    end if
14:   end for
15: end if
16: return  $s_i^*, C_{lb}(s_i^*)$ 

```

图 4-4 DFS 算法伪代码

Fig 4-4 Pseudocode for DFS

理论上，DFS 可以求解任何最短路问题，但是该算法并不是多项式时间算法，因此并不能保证计算时长在合理范围内。本文可以采用 DFS 算法是因为搜索深度和客户的最大试错次数 R 正相关，计算时间随搜索深度指数级增长，但通常 R 的值并不会很大，这使得 DFS 方案可行。其次，上述的两个启发式方法可以获得近似解，利用这个近似解可以在搜索过程中剪枝，避免遍历所有情况。最后，所有的成本均为非负值，即所有弧的权重为非负值，这使得一条不完整路径 s_i （不一定以 j_0 结尾）的产生的成本 $C_{lb}(s_i)$ 一定小于等于 s_i 被节点 $\{j\}$ 拓展的路径的成本 $C_{lb}(s_i + \{j\})$ ，即到达子节点产生的费用一定大于等于父节点的费用。

综上，求解下界问题相当于求解一个最短路问题，由于该问题是 NP-hard 问题，本小节给出了两种多项式时间的启发式算法以获得近似最优解，以近似最优解为已知最优解，使用了 DFS 算法进行剪枝搜索以获得下界最优解。此外，本小节给出了求解下界的数学规划模型，亦可使用求解器对下界进行求解。

4.2.3 上界获取

子问题的 $\text{RM}(\mu)_{sub1}$ 的解给出了选址方案 J^* ，令选址方案对应的选址决策变量 $y_j = 1, \forall j \in J^*$ 以及选址方案以外的选址决策变量 $y_j = 0, \forall j \in J \wedge j \notin J^*$ ，作为约束条件加入原模型 \mathbf{M} 中，使用精确算法求解该模型，即可获得原模型的上界。上界值等于建设节点的固定成本再加上客户的期望运输成本之和。在已知选址方案 J^* 的情况下，建设节点的固定成本等于被选中的候选节点的固定成本 f_j 的简单和。已知选址方案 J^* ，求解客户的期望运输成本之和，完全等价于第3.5.2节介绍的试错序列问题，即令拉格朗日乘子等于 0。

然而，客户试错序列问题是一个 NP-hard 问题（第3.5.2节给出了证明），求解该问题的启发式算法以及精确方法已在第4.2.2节中介绍。相较于下界，求解上界的过程较为简单：首先，下界问题中所有的候选节点都可以被使用，而在求解上界的过程只能使用已经建成的节点，问题规模缩小。其次，下界问题中所有节点的都存在固定成本（拉格朗日乘子），而上界问题中不存在固定成本。

由于本文考虑的是无容量限制的问题，因此在给定选址方案后，模型可拆分成 $|I|$ 个子问题，这些子问题的最优解之和等于上界最优解，每个子问题等价于试错序列问题也等价于 $\text{RM}(\mu)_{sub2}^i$ ，因此上界完全可以使用第4.2.2节中介绍的算法，但为了加速获得每个客户的试错序列、获得高质量上界，本节介绍了一种 Dijkstra 的变种算法，可以在多项式时间内求解上界。注意该算法仅能快获得一个近似最优解，并仅在不考虑访问节点固定成本（求解上界无拉格朗日乘子）的情况下表现良好。类似 Dijkstra 原始版本，采用永久标号法的步骤如下：

Step 0 初始化每个节点的权重为 ∞ ，到达每个节点的概率 p_j 为 1，每个点的状态为未标记，设置客户为起点，每个节点的追踪标记等于客户。

Step 1 从起点出发，计算到达所有未标记节点所需的成本，如果到达当前未标记节点的所需费用小于节点的权重，则更新该节点的权重。

Step 2 在所有未标记节点中，找到权重最小的节点，将其设为起点，更新其到达概率等于上一个节点的到达概率乘以上一个节点的失效概率，更新其追踪标记为上一个起点。

Step 3 重复 Step 1-2。若当前起点为 j_0 ，则结束，返回当前点的权重。若逆向追踪得到的路径中节点和客户总数等于 R ，则在路径最后补充一个 j_0 ，计算路径成本，

结束并返回路径成本。

该算法对经典 Dijkstra 算法进行了修改，在原有算法的基础上引入了到达该节点的概率，并添加了终止规则。算法的时间复杂度等于 $O(mn)$ ，其中 m 是客户最大尝试次数， n 是节点的个数。该算法可以求解上界近似最优解，也可以求解下界近似最优解。但是，求解下界解时加入了节点的固定访问成本，增加问题的复杂性，使得这种方法求解下界的效果不佳。在求解上界的过程中，由于不存在固定访问成本（拉格朗日乘子），算法迭代更新虚拟设施 j_0 的权重的次数提升，而每次更新 j_0 后，经过逆向追踪都可以获得一条更优的完整路径。因此，求解上界的 Dijkstra 的变种算法并不适用于下界求解。此外，Yun^[21] 等人设计了一种辅助图方法，基于辅助图，开发了另一种求解客户试错序列问题 Dijkstra 的变种算法。

回到整个拉格朗日算法框架，获得上界的前提是子问题 $\text{RM}(\mu)_{sub1}$ 的解给出了选址方案 J^* ，基于已知的选址方案可以获得质量较好的上界^[7,19]。本小节介绍了获取上界的方法，根据已知的选址方案，使用 Dijkstra 变种算法生成每个客户的试错序列。然而该方法对于求解下界效果不佳，但可以用来求解上界的近似最优解。此外，求解下界的 DFS 方法同样适用于求解上界客户试错序列问题中，将 Dijkstra 变种算法得到的解传递给 DFS 算法，可在此近似解的基础上得到最优解，该方法的详细操作见本章第4.2.2节。

4.2.4 乘子更新

第4.1节介绍了 LR 算法常采用次梯度法进行乘子更新。本文的算法和大多数 LR 算法一样^[4,7,19]，其算法的效率很大程度依赖乘子 μ 的初始值。合理的乘子初始值可以减少算法的迭代步骤，进而提升效率。本文乘子初始值设定参考了 Lim^[13] 的方法，令 $\mu^k = \{\mu_{ij}^k \geq 0\}$ 表示算法第 k 迭代时的乘子 μ 的取值， μ^0 表示乘子的初始值，其计算公式如下：

$$\mu_{ij}^0 = \frac{1}{|J|} (\lambda_i c_{ij} + f_j), \forall i \in I, j \in J \quad (4-25)$$

在第4.1节拉格朗日松弛概述中，公式(4-6)展示了一般问题中拉格朗日乘子更新的迭代方法。针对原问题 \mathbf{M} ，乘子更新方法如公式(4-26)所示，

$$\mu_{ij}^{k+1} = \mu_{ij}^k + t_k \left(\sum_{k \in J_{ij}^+} x_{ikj} - y_j \right), \forall i \in I, j \in J \quad (4-26)$$

其中， t_k 表示迭代步长，计算方式如下，

$$t_k = \frac{\alpha_k (ub_k^* - lb_k)}{\sum_{i \in I} \sum_{j \in J} \sum_{k \in J_{ij}^+} |x_{ikj} - y_j|} \quad (4-27)$$

公式(4-27)中 ub_k^* 和 lb_K 分别表示第 k 次迭代的已知最佳的上界和下界,迭代步长系数 α_k 是一个常量,当算法连续 κ_{lb} 次迭代都未能使下界提升时(一般在情况下,下界会在此时产生振荡), α_k 将除以比例因子 θ_{lr} ,以缓解振荡。关于公式(4-27)右侧分母项,本文参考了Yun^[21]关于迭代步长的设计,修改了经典拉格朗日算法迭代步长分母取平方操作,改用绝对值,以获得更优的收敛性。

4.2.5 停止条件

为了在有限的时间内获得高质量的上界,证明上界的质量,避免程序无意义的迭代,当LR-ILS算法满足下列条件其中之一时,算法停止迭代:

- (1) 迭代次数 k 大于等于最大迭代次数 η_{lr} ;
- (2) 算法优化时长到达时间上限 τ_{lim} ;
- (3) 上下界间隙(gap) $(ub_k - lb_K) / ub_k \leq \xi$;
- (4) 迭代步长系数 $\alpha_k <= \alpha_{min}$ 。

条件(1)限制了拉格朗日算法的迭代次数,条件(2)限制程序的运行时长,条件(3)表明期望接受的解的优劣程度,条件(4)避免算法后期不能提升下界的无用迭代。

4.2.6 上界改进

上述小节4.2.1至小节4.2.5已详细阐述了LR-ILS算法的细节,使用上述方法可以获得可接受的结果^[7,19,55]。但上述算法仍有一些缺陷:上界解的选址方案是由下界解提供的,下界解在算法后期收敛过程中并不能提供足够优质的选址方案,或者下界在后期产生振荡。为了加速算法上下界的收敛,本节介绍的迭代ILS算法对当前最佳选址方案 J^* 进行改进。

ILS算法可以作为独立的一种元启发式算法求解考虑节点失效风险的物流网络选址模型,即在ILS每次迭代中,ILS提供选址方案,再根据第4.2.3节获取上界的方法评估方案质量,不断迭代直至算法连续多次迭代都不能获得更优解时结束。但在本文中,将ILS嵌入LR迭代框架中(见图4-3),当ILS算法被触发,会对 J^* 进行邻域搜索,并将搜索到的更优解返回给LR,以减少上下界之间的间隙。

本文嵌入的ILS算法的框架设计参考了Lourenço^[88]等人书中内容,该算法的伪代码如图4-5所示。ILS算法是一种基于单个解的局部搜索算法,给定一个初始选址方案 J^* (已知最优解),ILS使用邻域算子搜索其邻域,根据一定准则接受邻域中某个邻域解,然后从这个邻域解出发继续搜索,搜索其邻域,如此往复,迭代进行。在搜索过程中,如果发现某个解优于已知最优解,则用这个解替换已知最

优解，并提前结束返回该解。

```

Input: 选址方案  $J^*$ 
1: 已知最佳选址方案  $J_{best}^* := J^*$ 
2: repeat
3:   邻域  $\mathcal{N}(J^*) :=$  邻域算子  $(J^*)$ 
4:    $obj_{\mathcal{N}(J^*)} :=$  目标函数  $(\mathcal{N}(J^*))$ 
5:   if  $\min(obj_{\mathcal{N}(J^*)}) < obj_{J_{best}^*}$  then
6:      $J_{best}^* := j$ 
7:     return  $J_{best}^*$ 
8:   else
9:      $J^* :=$  接受准则  $(\mathcal{N}(J^*))$ 
10:  end if
11: until 终止条件满足
12: return  $J_{best}^*$ 
```

图 4-5 ILS 算法伪代码

Fig 4-5 Psuedocode for ILS

算法的邻域搜索过程包括三种算子^[93,94]：关闭、开启、交换算子。关闭算子是指关闭 J^* 中的某个已建设节点，使得固定成本下降，运输成本上升，以在效益背反的博弈中使得总成本降低；开启算子是指开启 J^* 中的某个未建设节点，使得固定成本上升，运输成本下降；交换算子是指交换 J^* 中关闭某个已建设节点并开启某个未建设的节点。

每次迭代 ILS 评估所有邻域解的质量，发现更优的解将返回这个解，若没有发现更优解，则接受一个相对较差的解。接受准则借鉴了模拟退火算法的设计思路^[95]，当邻域解 $j^* \in \mathcal{N}(J^*)$ 未改进 J_{best}^* 时， j^* 仍有 $e^{[obj(j^*) - obj(J_{best}^*)]/T}$ 的概率被接受，其中 T 为模拟退火温度，其初始值设定为 T_0 。 T_0 的取值应使得 ILS 在 $obj(j^*)/obj(J_{best}^*) = \theta_{sa}$ 时的接受概率等于 50%，其中 θ_{sa} 是常数。即在 ILS 算法初次迭代时，如果某个邻域解的目标函数是已知最优目标函数值 θ_{sa} 倍时，ILS 有 50% 的概率接受这个解。温度 T 随 ILS 迭代次数线性减少，但不低于 T_{min} 。

当 LR 迭代过程中，连续 κ_{ub} 次未降低上界时，ILS 算法被触发。ILS 算法至多迭代 η_{ils} 次，该值应设置成一个较小的数，因为过多的搜索次数将降低算法整体的迭代效率。此外，ILS 也可以将当前迭代的邻域传递给下一次 ILS 触发，相当于搜索算法的重启操作^[96]，重启 ILS 很可能产生不同的搜索过程，进而提升发现更优解的概率。

4.3 本章小结

本章阐述了求解考虑节点失效风险的物流网络选址模型的算法，第4.2.1节松弛了模型中所有客户必须分配给已建设设施的约束，使得松弛问题可以拆分成多个子问题。其中求解每个子问题的过程等价于一个最短路问题，第4.2.2节采用了启发式加深度优先搜索算法求解这个子问题。松弛模型的节提供了选址方案，将该选址方案提供给上界问题（原模型），求解上界的过程仍然是求解最短路问题，第4.2.3节开发了 Dijkstra 变种算法，在多项式时间内求解客户在给定选址方案的情况下试错序列。第4.2.4节和第4.2.5节阐述了拉格朗日乘子更新和停止条件。特别地，本章的第4.2.6节给出了上界改进的方法，介绍了一种可以嵌入拉格朗日松弛迭代中可以改进上界的 ILS 算法。最后，值得一提的是，本章中所有的求解上界或下界的过程，都可以使用求解器进行求解，虽然求解时间随问题规模指数增长，但提供了一种检验自设计算法有效性的方法或标准。第5章对算法的性能进行了详细的分析，包括求解时间、求解质量等指标。

(本页留空)

5 数值实验设计及算法性能分析

本章将进行一系列实验以测试 LR-ILS 算法的性能，以此验证算法的有效性，特别关注于 LR-ILS 算法在解决大规模问题时的求解质量和求解时长。第5.1节阐述了计算实验的平台与环境，算法的参数取值以及数据来源等。第5.2节对算法的整体性能进行了分析；第5.3节和第5.4节分别分析了算法求解上下界的能力；第5.5节讨论了不同拉格朗日乘子初始设置和更新策略的效果；第5.6节讨论了模型线性化带来的求解增益效果；第5.7节展示了算例的选址结果及灵敏度分析结果；最后，第5.8节总结了本章的主要工作。

5.1 实验设计

实验采用的计算平台的配置如下：主频 3.1Ghz 的 6 核心 12 线程 CPU(AMD 1600) 搭配 8Gb 内存，操作系统为 Windows 10。LR-ILS 算法主程序由 Matlab 2022b 构建，使用 Python 3.9 调用 Gurobi 9.5.2 进行建模。为了加快程序运行速度，额外使用了 Matlab Coder 将算法的关键步骤编译成二进制 mex 文件，并在其中使用了多线程计算。

本文的第4章 LR-ILS 算法设计考虑了一系列控制算法效果的参数，经过多次的预实验，这些参数的取值如表5-1所示。在预实验中，给定算法的参数初始值（根据经验预估），每次调整固定算法的其他参数值，只调整一个参数，直至在该参数下算法效果最好，再更换其他参数进行调整。

表 5-1 LR-ILS 算法参数取值
Table 5-1 Parameter Setting for LR-ILS

LR		ILS		Gurobi	
参数	取值	参数	取值	参数	取值
α_0	2	θ_{sa}	1.2	MIPGap	0.00001
α_{min}	0.0001	T_{min}	0.0001	TimeLimit	1000
θ_{lr}	1.05	κ_{ub}	200		
κ_{lb}	10	η_{ils}	10		
η_{lr}	3000				
τ_{lim}	1000(s)				
ξ	0.01				

本节数值实验采用的数据来源于 Snyder^[4]，是选址研究的经典数据集。该数

据集包含了 49、88、150 个点的三套数据，可在 Snyder^[4] 的个人主页¹下载。原始数据为 1990 年美国房价普查数据，包含了美国各个州的人口和房价数据。学者们常使用该数据模拟节点选址中的固定成本和需求^[4,7,15,19]。本文参考了 Yun^[20] 的处理方法，令客户的需求等于州人口数除以 10^5 ，节点的固定成本 $f_j, \forall j \in J$ 等于房价的中位数，令节点失效的概率等于 $q_j = \rho e^{-f_j/200000}, \forall j \in J$ ，其中 ρ 是控制失效概率大小的参数，惩罚成本 $\pi = 10^4$ ，客户最大尝试次数 $R = 5$ 。此外，令客户点等于需求点，即 $I = J$ ，这样可以使图 G 中点的数量翻倍，也不必再区分数据集中哪些点是客户点或是节点候选点。令任意两点之间的距离等于大圆距离，该距离的具体计算方法可见参考文献^[4,19,30]。

5.2 综合求解性能分析

本节分析了算法的综合性能，分为四个内容，首先对比测试了 LR-ILS 算法求解不同规模、不同参数取值的数据的性能；其次，对比 Gurobi 分析了求解时间和求解质量；然后，分析了模型中两个重要参数 ρ 和 R 对 LR-ILS 求解性能的影响；最后，通过上下界优化曲线直观展示了算法的迭代过程。附录 B 展示了使用 LR-ILS 求解包含 49、88、150 个点的数据集的所有计算结果，其中参数 ρ 分别取 0.01、0.05、0.1、0.2、0.3、0.4、0.5， R 分别取 2 至 10。

5.2.1 求解性能对比

表5-2分别展示了 LR-ILS 算法和 Gurobi 求解器求解 49 个点、88 个点、150 个点以及参数 ρ 不同取值的计算结果。表5-2的所有实验均默认参数 $R = 5$ ，其中 gap 值等于上下界之差除以上界，相对差距等于 LR-ILS 得到的上界值减去 Gurobi 得到的上界值再除以二者之间的较小值，因此相对差距为负时，LR-ILS 获得了更优的上界，该值越小表明 LR-ILS 得到的结果越好。相对速度等于 Gurobi 的求解时间除以 LR-ILS 的求解时间，该值越大表明 LR-ILS 求解速度越快。

数值结果表明 LR-ILS 算法的显著优势在于能够在相对较短的时间内获得质量较高的近似最优解，具体分析如下：首先，从求解的时间和速度进行分析，除了第一组数据外，Gurobi 求解时长均到达了设置的最大运行时长，并且 150 个点的问题全部没有在规定时间内得到结果。相比之下，LR-ILS 算法的求解时间显著优于 Gurobi，注意到该问题为 NP-hard 问题，问题规模随节点数量增加而指数级增长。因此在求解 49 个点的问题时，LR-ILS 算法的求解效率是 Gurobi 的数千（百）

¹<https://coral.ise.lehigh.edu/larry/research/data-sets-for-reliability-models-for-facility-location-the-expected-failure-cost-case>

表 5-2 LR-ILS 与 Gurobi 结果对比
Table 5-2 Results for LR-ILS and Gurobi

数据集	ρ	LR-ILS			Gurobi			相对 差距 (%)	相对 速度 (倍)
		上界	gap(%)	时间 (s)	上界	gap(%)	时间 (s)		
49	0.01	965061.90	0.99	0.21	965227.40	0.19	39.37	-0.02	187.48
49	0.05	1018144.58	0.89	0.40	1021312.55	2.54	1000.00	-0.31	2500.00
49	0.10	1076487.05	0.99	0.53	1077872.84	8.92	1000.00	-0.13	1886.79
49	0.20	1198200.17	1.15	13.27	1224021.70	19.67	1000.00	-2.16	75.36
88	0.01	1361557.62	1.00	0.76	1360245.50	1.16	1000.00	0.10	1315.79
88	0.05	1440083.62	0.99	1.18	1438341.99	6.09	1000.00	0.12	847.46
88	0.10	1527361.61	0.92	6.67	1544990.68	12.90	1000.00	-1.15	149.93
88	0.20	1721135.80	2.70	98.52	1762412.99	23.85	1000.00	-2.40	10.15
150	0.01	2176492.30	1.00	13.63	-	-	1000.00	-	-
150	0.05	2279427.84	1.39	19.26	-	-	1000.00	-	-
150	0.10	2391069.73	1.93	46.93	-	-	1000.00	-	-
150	0.20	2611120.81	3.31	184.14	-	-	1000.00	-	-

倍，求解 88 个点的问题时，求解效率退化至数百倍，求解 150 个点的问题时，求解效率退化至数十倍，表 5-2 中“相对速度（倍）”列展示了这种趋势。相比 Gurobi，LR-ILS 算法的求解速度具有显著优势。在上界质量接近的情况下，LR-ILS 的求解时间更短。

其次，从求解的质量进行分析。对比 Gurobi，对于 49 个点的全部算例，LR-ILS 获得了更加优质的上界；对于 88 个点的部分算例，LR-ILS 的上界值略差于 Gurobi，但二者之间的相对差距 ($\leq 0.12\%$) 几乎可以忽略不计；对于 150 个点的全部算例，LR-ILS 得到了上界的计算结果，而 Gurobi 未能得到可行解。Gurobi 的上界质量和 LR-ILS 上界质量相差无几，观察表中的 gap 值项，除了第一组较为简单的数据集之外，Gurobi 的 gap 值均大于 LR-ILS 的 gap 值。这表明证明上界的最优化方面，Gurobi 明显不如 LR-ILS，即拉格朗日松弛得到的下界优于 Gurobi 内置的线性松弛得到的下界。

5.2.2 算例参数 ρ 的影响

尽管 LR-ILS 可以求解更复杂的问题并提供了优质下界，但观察到无论是 LR-ILS 还是 Gurobi，求解问题所需的时长以及 gap 值都随参数取值的变化而变化。参数 ρ 的取值将影响 LR-ILS 算法的求解性能。该参数控制节点的失效概率大小，使得节点的失效概率接近 ρ 自身，但由于节点的建设成本不一，各个节点的失效概率也不相等。随着参数 ρ 的增加，模型的复杂程度提高，LR-ILS 算法的表现也越来越退化。本节使用了 49 个点的算例进行测试， ρ 取 0.1 至 0.9 间隔为 0.1 的 9 个

值, R 取 2 至 10 的整数, 求解时间及 gap 值随 ρ 变化如图5-1和图5-2所示。

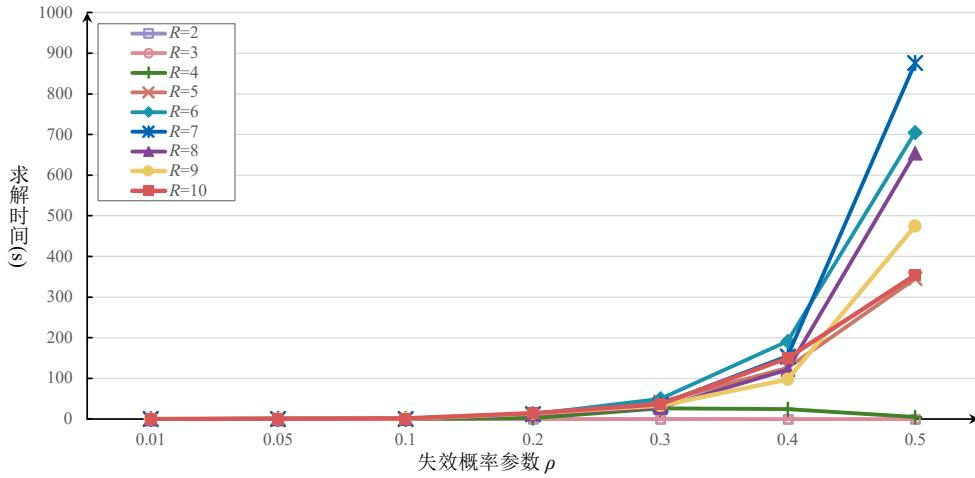


图 5-1 LR-ILS 算法求解时间随 ρ 的变化曲线
Fig 5-1 Curves of LR computating time with respect to ρ

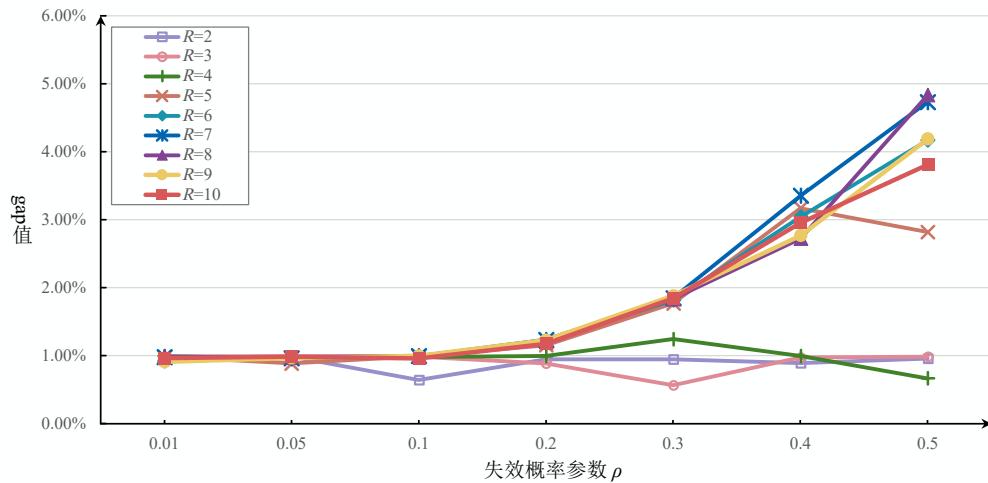


图 5-2 LR-ILS 算法求解 gap 随 ρ 的变化曲线
Fig 5-2 Curves of LR gap values with respect to ρ

图5-1和图5-2的横坐标轴表示参数 ρ 的变化范围, 纵坐标轴分别表示在此 ρ 取值下求解所需的计算时间以及结果的 gap 值, 不同颜色的线条表示不同 R 取值的结果。首先分析求解时间与 ρ 的关系, 图5-1中可见, 参数 ρ 显著影响了求解的时间, 在大多数测试中, 求解时间随参数 ρ 的增加而指数级增加, 特别是在 $\rho = 0.2$ 至 0.5 的区间内, 求解时间的增加变化十分明显。这表明随着 ρ 取值变大, 模型的复杂度提升, 求解的难度随之增加。求解难度提升的本质原因是随着节点失效概率增加, 求解客户试错序列的难度增加, 即安排客户不同等级的备用节点以充分降低惩罚成本, 以及在设施建设成本、期望运输成本以及惩罚成本三者之间权衡的难度增加。这导致了算法的精确过程, 例如 DFS 搜索过程的效率显著降低, 由于节点的失效概率增加, 惩罚成本所占比重增加, 启发式不能提供一个优质的近

似解导致 DFS 剪枝过程受影响，搜索次数、深度明显增加。注意到在一些测试中，求解时长并不受 ρ 的显著影响，这是因为这些测试的 R 的取值较小 (≤ 4)，每个客户的常用节点和备用节点的指派任务较为简单（最优客户试错序列），因此求解时长变化并不明显。在 $R \leq 4$ 的情况下，对于 49 个点的算例，LR-ILS 算法可以在很短的时间内求解得到近似最优解。对于 R 取值较大的情况，LR-ILS 也能在可接受的时间内求解得到一个结果。

其次，分析求解结果 gap 值随 ρ 的变化的关系。与求解时间随 ρ 变化的情况类似，在大部分测试中，gap 值同样随 ρ 增加而增加，特别是在 $\rho = 0.2$ 至 0.5 的区间内的变化较为明显。gap 值证明了上界的最优性，图5-2中所有的结果表明，LR-ILS 求解结果的 gap 值不超过 5%，即 LR-ILS 算法求解 49 个客户点与 49 个备选点（或 49 个点以下）的问题时，有能力得到一个在 5% 误差内的近似最优解。同样， $R \leq 4$ 的测试结果中，gap 值受 ρ 变化的影响较小。本文中 LR-ILS 算法设置停止准则的 gap 值为 0.01，因此一些测试的 gap 值只停留在 1% 附近，若改变算法参数，gap 值有望进一步降低。

5.2.3 算例参数 R 的影响

注意到在图5-1和图5-2中，并非 R 的值越大，求解的时间越久或 gap 值越高。例如，在 $\rho = 0.5$ 时 $R = 7$ 的求解时间最长，并且求解结果的 gap 值较高。图5-3和图5-4 从另一个角度分析了求解时间和 gap 值受参数 R 的影响程度。图5-3和图5-4 的横坐标表示 R 的不同取值，纵坐标分别表示求解时间和 gap 值，不同颜色的线条表示不同 ρ 的取值。

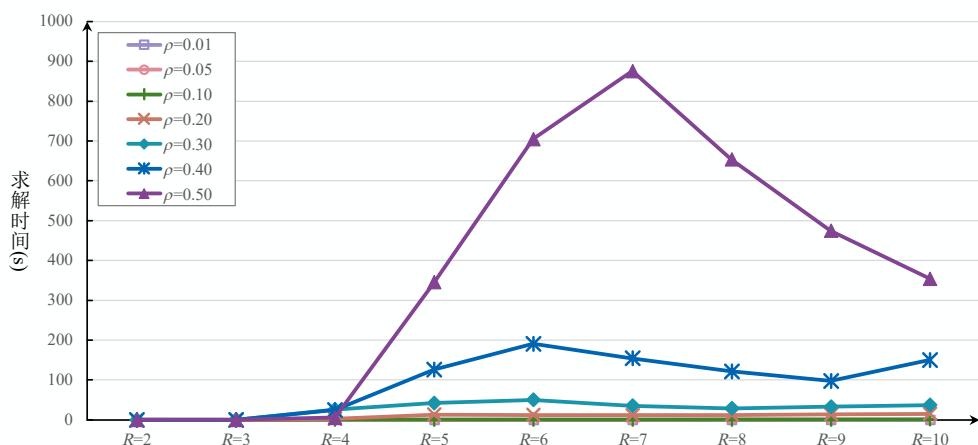


图 5-3 LR-ILS 算法求解时间随 R 的变化曲线

Fig 5-3 Curves of LR computating time with respect to R

在图5-3和图5-4中，LR-ILS 求解的时长以及 gap 值随参数 R 的增加先上升后下

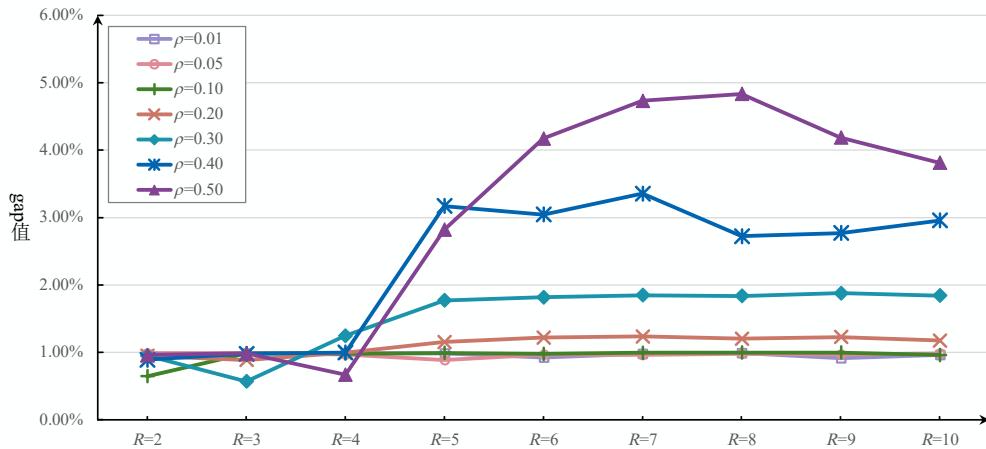


图 5-4 LR-ILS 算法求解 gap 随 R 的变化曲线

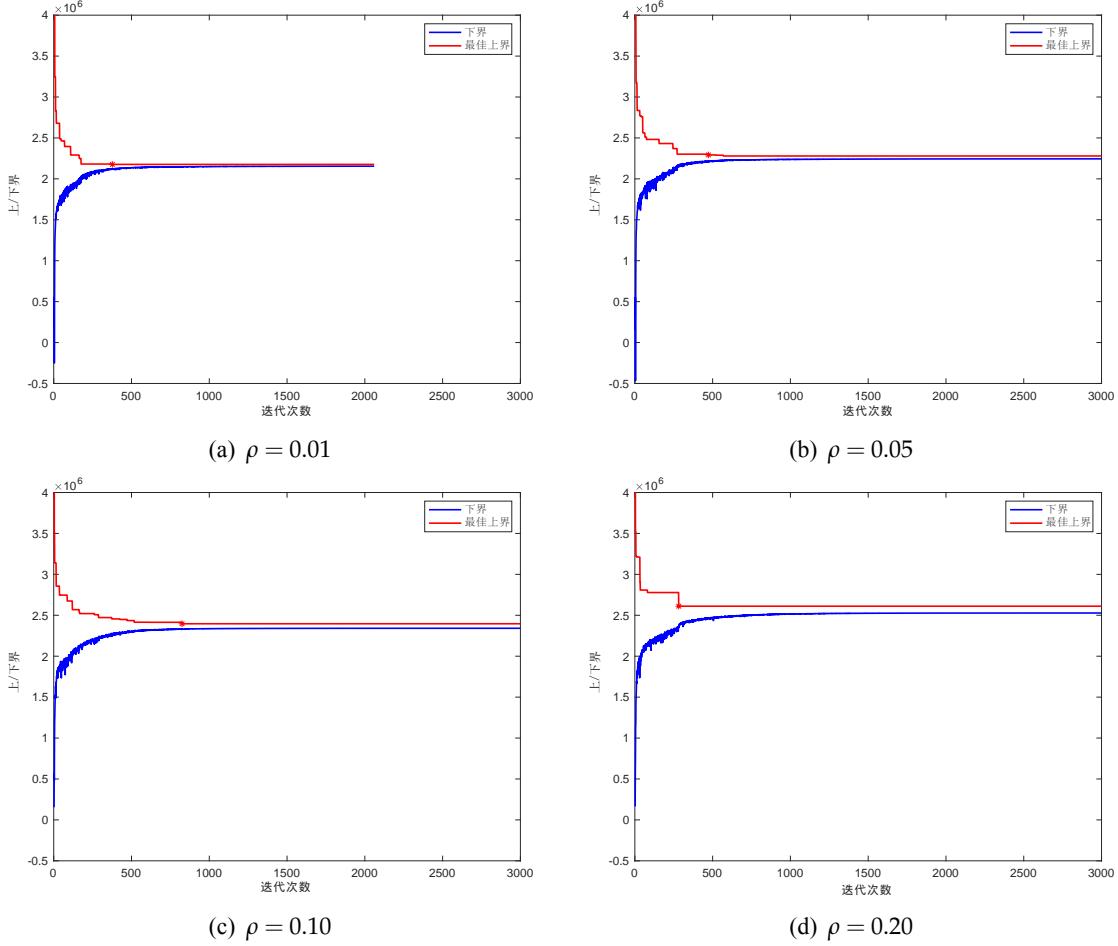
Fig 5-4 Curves of LR gap values with respect to R

降，这种趋势在 ρ 取值较大时更为明显。产生该现象的原因是参数 R 和参数 ρ 对问题求解复杂程度的复合影响。在 ρ 取值较大时，惩罚成本占比较高，随着 R 不断增加，为了降低惩罚成本，优化过程会为每个客户尽可能多地指派备用节点以降低惩罚成本，因此求解模型的时间增加且 gap 值增加。当 R 增加至一定值时，求解时间或 gap 值增加到顶点。对于 49 个点的算例来说，这个值是 7 或 8。随后， R 值提升导致客户前往接受惩罚的概率降低，再增加 R 的值，惩罚成本在总成本中的占比显著降低，这反而降低了求解子问题的难度，求解子问题时 DFS 剪枝策略又开始生效，求解效率提升。但是，随着 R 增加，问题更加复杂，因此求解时间和 gap 值的表现也很难恢复至之前 R 取值较小的水平。

注意，上述分析单纯从数值角度出发，在实际中，很少有节点伴随如此高的失效概率 ($\rho = 0.5$ 时节点平均失效概率为 35.14%)，因此该算法仍可以求解现实问题。本文同样给出了提高求解质量的途径：参数调整方面，可适当增加 LR 的迭代次数，降低迭代步长比例因子，算法设计方面，可改进上界获取的启发式，或开发求解试错序列问题的更有效算法。即便 LR-ILS 算法求解失效概率较大的问题时，获得的近似最优解 gap 不佳，但该近似最优解仍优于 Gurobi 获得的解。

5.2.4 优化曲线

最后，图5-5直观展示 LR-ILS 算法优化过程，使用了 150 个点的数据集，默认参数 $R = 5$ ，分别展示了 $\rho = 0.01, 0.05, 0.1, 0.2$ ，不同取值的 LR-ILS 优化曲线。图5-5中，横坐标表示迭代次数，纵坐标表示上下界的取值；红色的曲线表示最佳上界的变化，呈现阶梯状；蓝色的曲线表示下界的变化，呈现锯齿状；红色曲线上标记表示 ILS 算子找到了更优的选址方案。阶梯状上界的产生是因为 LR-ILS 每

图 5-5 不同 ρ 取值的 LR 优化曲线Fig 5-5 LR optimization curves for different ρ

次迭代中仅记录最佳上界，锯齿状下界的产生是因为迭代步长根据最佳上界值和当前下界值反复修正。从图5-5中可发现，算法的收敛过程相对较快，可在 500 至 1000 次迭代内找到 150 个点的近似最优解，然后下界不断提升证明其质量。从图中可发现，ILS 可以改进上界，借助其搜索能力，很容易根据当前上界解找到更优上界解，进一步改进上界。LR-ILS 的大部分迭代过程在证明上界的质量，优质上界主要由子模型 $RM(\mu)_{sub1}$ 提供，ILS 算子的改进过程起到辅助作用。

5.3 下界求解性能分析

本节将分析算法中启发式与 DFS 算法求解下界解的性能，由于构造启发式的时间复杂度优于插入启发式算法（第4.2.2节），本节仅测试了插入启发式配合 DFS 算法求解下界的能力。实验采用 49 个点的数据集， ρ 的值分别取 0.1, 0.2, 0.3, R 值分别取 2 至 10 的整数，拉格朗日乘子分别取全 0、初始值与随机值，其中随机值为 0 至 30000 之间的随机数。仍采用 Gurobi 作为对比项，Gurobi 参数设置同表5-1—

致。

测试结果如表5-3所示，表中 HEU 表示仅使用启发式，HEU-DFS 表示使用启发式与 DFS 算法，GRB 表示使用 Gurobi 求解。在测试得到的结果中，HEU-DFS 与 GRB 得到的结果为最优解，HEU 得到的结果为近似最优，因此计算了 HEU 与最优值之间的差距，并命名为 gap*，求解时间的单位为秒，HEU 及 HEU-DFS 均为 Matlab 编码并转译生成的二进制文件，采用了多线程计算。表5-3中所有的结果均为 R 值分别取 2 至 10 的平均值，该表的完整内容见附录 B 中的附表 2。

表 5-3 下界求解性能

Table 5-3 Performance on solving lower bound

乘子 取值	ρ	HEU-DFS		HEU			GRB	
		上界值	时间	上界值	时间 (s)	gap*	上界值	时间 (s)
全零	0.1	224533.85	0.009	228752.63	0.001	0.42%	224532.63	85.260
	0.2	475214.35	0.012	493006.24	0.001	1.24%	475211.87	204.624
	0.3	756334.13	0.019	801825.67	0.000	2.68%	756331.82	388.021
初始	0.1	406887.38	0.002	411208.16	0.000	0.54%	406887.38	24.521
	0.2	702966.10	0.006	723855.37	0.000	1.92%	702966.10	120.713
	0.3	1023301.45	0.023	1075474.89	0.000	3.23%	1023301.45	349.158
随机	0.1	2077863.06	0.001	2082803.64	0.000	0.20%	2077863.06	15.280
	0.2	2570295.08	0.001	2600906.18	0.000	1.07%	2570295.08	15.393
	0.3	3066433.78	0.002	3140069.58	0.000	2.20%	3066433.78	22.739

表5-3展示了采用三种方法求解得到的结果。首先，分析三种方法的求解时间的差异。尽管 DFS 算法并不是多项式时间算法，但采用启发式搭配 DFS 算法得到的最优解的计算时间并不长，并且随参数变动而变动的幅度较小。而 Gurobi 的求解时间随参数变动而变动，特别是随着 ρ 值升高，Gurobi 求解时间显著增长。此外，拉格朗日乘子的取值对 Gurobi 的影响也十分大，当乘子全为 0 时，Gurobi 计算所有案例的平均时长为 225.969s，当乘子为初始值时，平均时长为 164.797s，当乘子取随机值时，平均时长为 17.804s。产生这种现象的原因是精确求解方法受参数影响较大。当 R 值升高时，约束 (3-17) 变得松弛，因此精确方法（Gurobi 内置的各种 Branch and Bound, Cutting Planes 等方法）效果变差。同理，随着 ρ 参数增加，节点失效概率参差不齐，模型同样变得复杂。拉格朗日乘子被视作访问节点的固定成本，因此当乘子等于 0 时，求解器求解该模型相对困难，原因同上。当乘子不等于 0 时，精确方法可快速缩小可行域范围，进而求解时间较短。回到本文的启发式算法，由于该算法是多项式时间算法，因此求解时长随参数变动较小。DFS 算法虽然不是多项式时间算法，但是剪枝过程大幅度缩短了计算所需的步骤，与 Gurobi 相比是一种更有效地获取最优解的策略。

其次，分析三种方法求解结果的差异，已知启发式搭配 DFS 算法和 Gurobi 可

以获得最优解，启发式方法可以获得近似解。该启发式方法在 R 取值较大时、 ρ 取较小时，表现良好。并且受乘子取值的影响，乘子不为零时表现良好。分析该启发式的贪心策略可知，算法每次将成本最小的节点增加至末尾。当 R 值增加时，惩罚成本的比重逐渐被稀释，使得贪心策略效果提升。而 R 值较小时，贪心策略仅能考虑当前最优，未能考虑最终惩罚成本的影响。同理，当乘子不为零时，乘子影响的贪心过程的决策，进而提升了启发式的质量。启发式方法获得的近似解总体质量较好（最大误差 8.18%，最小 0.02%，平均 1.50%），但为了得到下界的最优解以证明上界的质量，在本文算法仍使用了启发式搭配 DFS 的方式。最后，值得注意的是，尽管 DFS 和 Gurobi 都获得了最优解，但在一些算例中，Gurobi 得到的值小于 DFS 方法得到的值。经过排查，为决策变量 w_{ijk} 的值小于 10^{-6} ，在 Gurobi 中认定为 0。有关 Gurobi 数值精度问题，请参考 Gurobi 在线手册²。

5.4 上界求解性能分析

本文的第4.2.3节介绍了获取上界值的方法，在给定选址方案 J^* 的前提下，可很容易计算得到节点建设的固定成本，计算客户的期望运输成本等于求解第3.5.2节的客户试错序列问题。第4.2.3节介绍的 Dijkstra 变种算法可以获得客户试错序列的近似解；第4.2.3节提出可用 DFS 算法获得上界的最优值。此外，还可使用 Gurobi 求解器求解模型的上界值。

本节对求解上界的性能进行了测试，分别使用了 Dijkstra 变种算法（本节简称为 DIJK），Dijkstra 变种算法搭配 DFS 算法（本节简称为 DIJK-DFS），以及 Gurobi 求解器（本节简称为 GRB）求解了 49 个点的不同算例。已知选址方案 J^* 中设施的个数显著影响算法求解时长，因此，本节分别测试了 10, 20, 30, 40 个点的选址方案，参数 ρ 取值 0.1, 0.2, 0.3，参数 R 取值为大于等于 2 小于等于 10 的整数。测试结果如表5-4所示，表中近似解与最优之间的差距命名为 gap*。求解时间单位为秒，DIJK 算法与 DIJK-DFS 算法均为 Matlab 编码并转译成 C++ 二进制文件，采用了多线程计算。表5-4中所有的结果均为 R 值分别取 2 至 10 的平均值，该表的完整内容见附录 B 中的附表 3。

表5-4展示了使用三种方法求解上界的结果，从求解时间的角度分析，对比 Gurobi 求解器，DIJK 算法以及 DIJK-DFS 算法具有显著优势。对于所有算例，上述两种算法可以在 0.1 秒内给出上界值，而 Gurobi 虽然可以获得上界值，但求解时间明显与这两种算法不在同一数量级（最小差距 1131.29 倍，最大差距 63136.36 倍，详见附表 3）。并且 Gurobi 的求解时间随参数变动而变动， $|J^*|$ 越大、参数 ρ

²https://www.gurobi.com/documentation/9.5/refman/guidelines_for_numerical_i.html

表 5-4 上界求解性能

Table 5-4 Performance on solving upper bound

$ J^* $	ρ	DIJK-DFS		DIJK			GRB	
		上界值	时间	上界值	时间 (s)	gap*	上界值	时间 (s)
10	0.1	1611587.91	0.001	1640555.97	0.006	1.90%	1611590.82	5.111
	0.2	1925003.60	0.001	2011904.94	0.001	4.68%	1925001.32	5.803
	0.3	2287526.81	0.001	2472292.62	0.001	8.35%	2287525.39	6.340
20	0.1	2067339.22	0.001	2072951.15	0.001	0.22%	2067342.67	10.999
	0.2	2335167.39	0.002	2371565.20	0.001	1.39%	2335168.60	11.597
	0.3	2639304.64	0.002	2750515.28	0.001	4.01%	2639302.63	14.668
30	0.1	2752865.35	0.003	2761269.07	0.001	0.22%	2752853.58	19.238
	0.2	3011919.92	0.006	3097686.57	0.001	2.61%	3011907.14	25.567
	0.3	3305273.39	0.006	3490345.05	0.001	5.16%	3305262.35	45.998
40	0.1	3323545.36	0.004	3331937.55	0.001	0.20%	3323545.56	48.869
	0.2	3575894.34	0.006	3640638.02	0.001	1.66%	3575893.29	84.519
	0.3	3858144.43	0.017	4031583.03	0.001	4.25%	3858140.97	156.080

越大、参数 R 越大，则模型越复杂，导致 Gurobi 求解时间越长。最短可在几秒内完成简单算例的求解，最长需要几分钟。DIJK 算法以及 DIJK-DFS 算法求解时间则非常稳定，最长不超过 0.1 秒。Gurobi 求解时长随参数变化的原因已在第5.3节分析。

已知在本测试中，Gurobi 和 DIJK-DFS 获得上界的最优值，但 DIJK 算法只能获得近似值。表5-4的 gap* 列展示了该算法获得的近似上界值和最优值之间的相对差距，在 R 值较大时，DIJK 算法得到的解质量较好。但在 R 值较小时，DIJK 算法表现劣化。DIJK 算法是基于 Dijkstra 算法进行了问题的适配，但保留了 Dijkstra 算法永久标号的主要想法，所以不可修改的标号将在 R 值较小时生成质量欠佳的决策，进而影响解的质量。当 R 值较大时，即便初始步骤生成的路径质量欠佳，可通过后续决策降低惩罚成本的比重。即便 DIJK 算法求解 R 值较小的问题的效果较差，但 DFS 算法在 R 值较小时因搜索深度较小而展现了良好的性能。

综上，求解上界的 Dijkstra 变种算法在求解 R 值较小的问题时，得到的上界质量较差，但 DFS 算法的搜索深度较浅，可快速基于 Dijkstra 变种算法的解获得最优解。在 R 值较大时，Dijkstra 变种算法得到的上界质量较好，为 DFS 算法提供了一个质量较好的已知解，加速了 DFS 中剪枝的过程。两种算法优势互补，使求解上界的过程十分有效。

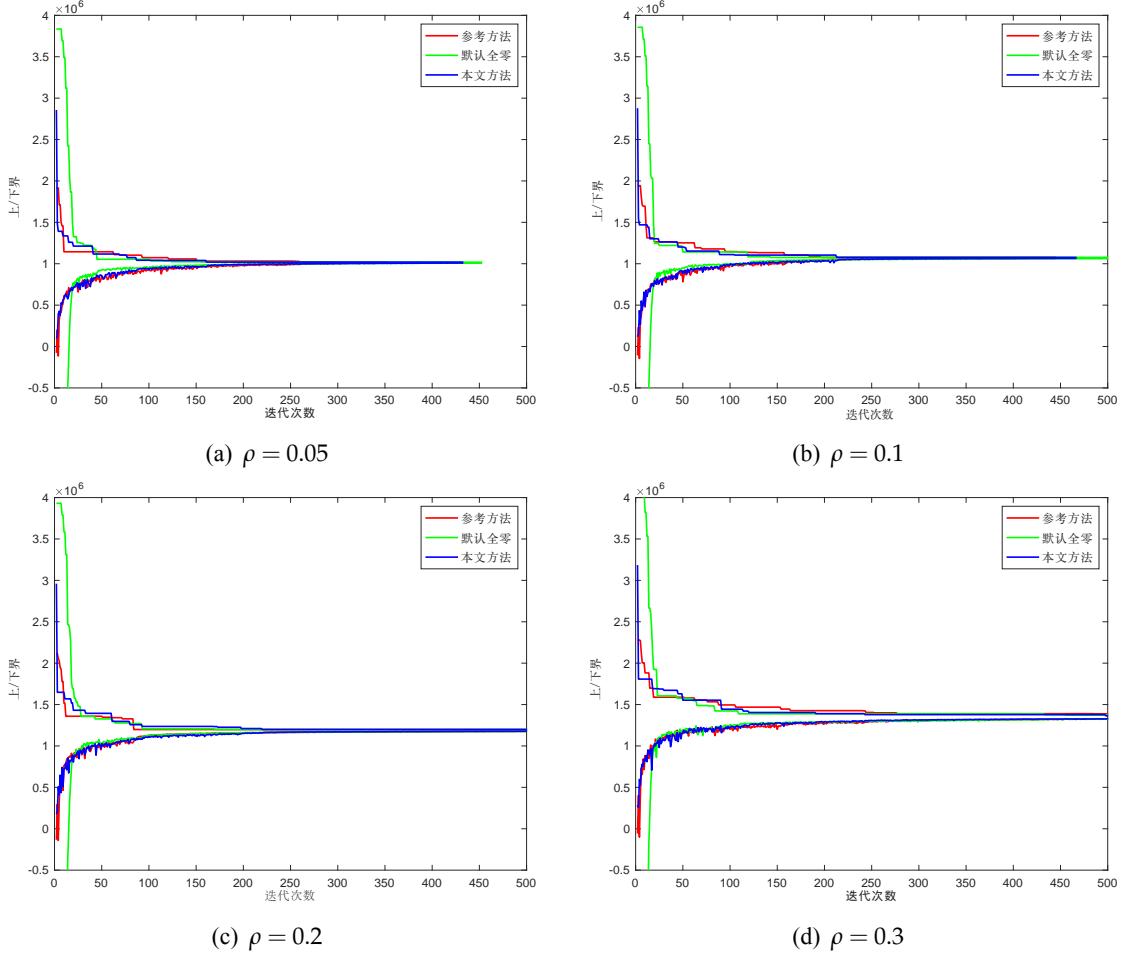


图 5-6 乘子初始值的不同策略对优化过程的影响

Fig 5-6 Effect of initialization strategies of the multiplier on the optimization process

5.5 乘子更新效果讨论

本文的第4.2.4小节阐述了拉格朗日乘子初始设置以及乘子更新的方法，本节将对比多种乘子初始化方法以及两种乘子更新方法的不同效果。为了显著体现不同拉格朗日乘子设置产生的效果，本节使用了相对较小的49个点的数据集进行实验，对比分析拉格朗日优化曲线的收敛性。在本节中，默认设置参数 $R = 5$ ，参数 ρ 分别取值0.05、0.1、0.2、0.3。为了消除其他因素的干扰，本节的所有测试均关闭了ILS算子。

图5-6中展示了在不同 ρ 取值的情况下，算法前500次迭代的结果。图中绿色曲线表示将全部乘子初始值设置成0的结果，红色表示文献^[19]使用的乘子初始化方法，蓝色曲线表示本文使用的乘子初始化方法。从整体优化的趋势来看，三种方法的优化曲线都呈现良好的收敛趋势，最终上下界都收敛至最优解附近。然而，效果最差的初始乘子方法是默认全零方法，该方法的初始gap值非常大，并且在 $\rho = 0.05$ 的情况下迭代次数最多。虽然该方法的前期gap值非常差，但可以快速收

敛，并且在所有测试中下界收敛效果最好。对比文献^[19]的方法，本文的方法在 LR 迭代的第一次迭代时得到的上界更大，但可以快速下降。本文方法获得的上界结果不如文献^[19]的方法，但获取下界的效果更好，体现在下界初始波动更小，获得的下界值更高。为了方便展示优化曲线的主体部分，图5-6忽略了默认全零方法前几次迭代的下界结果，该下界结果的量级在 -10^7 左右。相比之下，其他两种方法的初始下界均在 0 附近。

得益于良好的迭代策略，三种方法最后都收敛于最优解附近。公式 (4-27) 展示了拉格朗日乘子步长设置的准则，对基础乘子更新策略中迭代步长进行了改进，取消了分母的二次项并设置为一次项。

图5-7展示了算法两种不同乘子更新策略的结果，同样以 49 个点的为基准进行测试，其中默认 $R = 5$ ，参数 ρ 分别取值 0.05、0.1、0.2、0.3。图中红色的曲线表示若公式 (4-27) 分母项取二次（即原始步长公式）获得的上下界迭代效果，蓝色曲线表示公式 (4-27)（即本文使用的改进公式）的效果。显然，无论是收敛性还是上下界质量，使用改进公式的效果都十分明显。产生这样的差异的原因是，当公式 (4-27) 的分母取平方项时，生成的迭代步长过小，每次迭代对拉格朗日乘子的修正较小，因此获取上下界的效果较差。改进的公式取消了平方操作，每次迭代的步长较大，乘子更新更为激进，进而得到了良好的效果。

5.6 线性化效果讨论

本文的第3.4节提出了模型的线性化方法，该方法消除了模型中的二次项，在理想情况下可降低采用精确方法求解问题的难度。本节将对比使用 Gurobi 求解线性模型和非线性模型的性能差距。求解器参数设置同表5-1一致，测试了 15 个点的小规模数据集，其来源于 49 个点的数据集的前 15 个点。设置参数 ρ 分别等于 0.05, 0.1, 0.2, 0.3，参数 R 为大于等于 2 小于等于 10 的整数，求解结果如表5-5所示。表中上界值为模型的目标函数值，即目标函数 (3-11) 和目标函数 (3-27)，gap 值为 Gurobi 返回的 gap 结果，求解时间为 Gurobi 求解时长（秒）。

从表5-5的结果综合来看，使用 Gurobi 求解器求解线性模型和非线性模型的困难程度相差不大。在上述测试中，求解线性模型的平均时间为 494.35 秒，求解非线性模型的平均时间为 469.79 秒；求解线性模型平均 gap 值为 1.82%，非线性模型平均 gap 值为 1.87%。对于小规模的 15 个点的问题，当参数 ρ 较小时，求解线性模型相对容易，当参数 ρ 较大时，线性化的作用并不明显，在某些数据测试中，求解线性模型较为简单，在其他数据测试中，求解非线性模型较为简单。

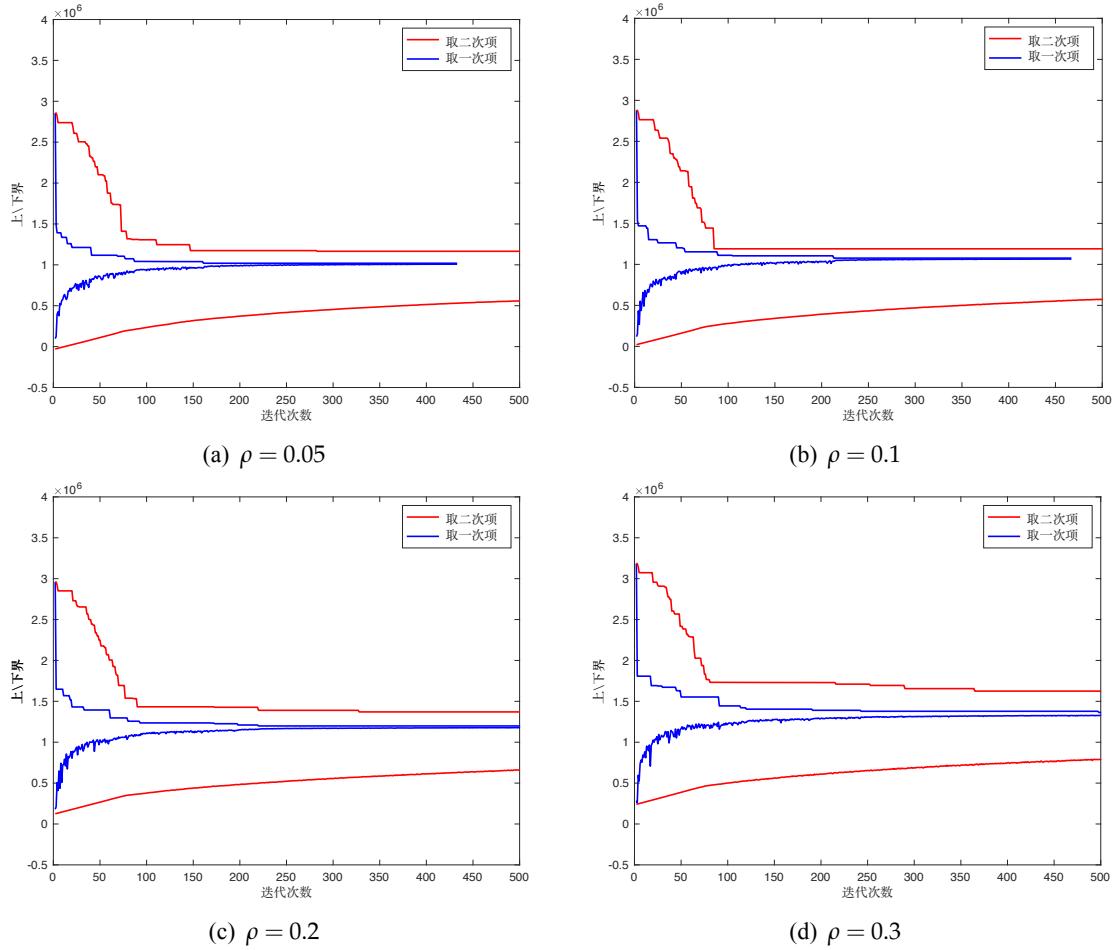


图 5-7 迭代步长对优化过程的影响

Fig 5-7 Effect of step length strategies of the multiplier on the optimization process

表 5-5 非线性模型和线性模型求解结果对比

Table 5-5 Comparing the results of solving the linear and non-linear models

ρ	R	非线性模型			线性模型		
		上界值	gap	时长 (s)	上界值	gap	时长 (s)
	2	1184928.86	0.00%	0.07	1184928.86	0.00%	0.08
	3	664900.02	0.00%	3.48	664900.02	0.00%	2.83
	4	644206.22	0.00%	6.02	644206.22	0.00%	5.47
	5	643430.51	0.00%	5.55	643430.51	0.00%	4.52
0.05	6	643401.30	0.00%	6.55	643401.87	0.00%	4.22
	7	643401.56	0.00%	6.03	643401.17	0.00%	3.70
	8	643401.34	0.00%	4.57	643402.93	0.00%	3.57
	9	643409.86	0.00%	6.67	643401.18	0.00%	3.97
	10	643401.32	0.00%	5.30	643401.11	0.00%	5.13
	2	1736825.35	0.00%	0.04	1736825.35	0.00%	0.05
	3	777800.34	0.00%	11.69	777800.34	0.00%	13.23

(接续表 5-5)

(续表 5-5)

ρ	R	非线性模型			线性模型		
		上界值	gap	时长(s)	上界值	gap	时长(s)
0.1	4	698526.23	0.00%	509.63	698526.23	0.10%	1000.08
	5	692637.63	0.06%	1000.07	692637.63	0.00%	164.69
	6	692205.27	0.00%	117.82	692205.25	0.01%	138.43
	7	692175.16	0.05%	1000.07	692174.56	0.00%	210.79
	8	692176.10	0.00%	45.59	692174.51	0.04%	1000.05
	9	692176.11	0.00%	44.68	692174.55	0.00%	187.83
	10	692175.82	0.00%	50.29	692174.53	0.04%	1000.09
	2	2719339.96	0.00%	0.05	2719339.96	0.00%	0.06
	3	1114008.80	0.00%	14.36	1114008.80	0.00%	9.11
	4	845062.91	5.40%	1000.24	845062.91	5.92%	1000.04
0.2	5	804766.33	2.55%	1000.25	804766.33	2.01%	1000.15
	6	798898.18	2.19%	1000.20	798898.18	2.03%	1000.31
	7	798124.48	1.96%	1000.16	798124.48	1.63%	1000.06
	8	798124.42	1.88%	1000.07	798124.48	2.22%	1000.13
	9	798124.48	1.66%	1000.09	798124.48	1.64%	1000.07
	10	798124.48	1.27%	1000.49	798124.47	1.79%	1000.13
	2	3625538.91	0.00%	0.06	3625538.91	0.00%	0.07
	3	1568969.60	0.00%	70.28	1568969.60	0.00%	36.28
	4	1068970.09	18.97%	1000.29	1068970.07	18.19%	1000.22
	5	946883.17	8.00%	1000.27	949174.41	8.29%	1000.27
0.3	6	921146.27	5.79%	1000.63	920491.77	4.79%	1000.21
	7	914794.24	4.68%	1000.26	914460.06	4.34%	1000.30
	8	913246.31	4.41%	1000.12	913246.31	4.47%	1000.27
	9	913511.86	4.43%	1000.25	913246.31	4.20%	1000.13
	10	913246.31	4.08%	1000.10	913246.31	3.88%	1000.23

线性化提升求解效率的局限性主要体现在三个方面：首先，线性化将模型中的二次项消除，但相应增加了决策变量和等价约束的个数，其中变量增加数为 $O(|I||J|^2)$ ，约束增加数为 $O(|I||J|^2)$ 。这些额外增加的约束使得线性模型求解难度并没有降低太多。此外，线性化没有改变模型是混合整数规划模型的本质，因此，增加的约束和变量增加了模型的维度，使分支定界的过程同样变得复杂。最后，线性化没有改变问题是 NP-hard 的本质，采用精确求解的方法并不适用于大规模问题或者小规模但复杂的问题。

但线性化仍有一定意义，它提供了一种可能的降低模型求解难度的方法，在表5-5中，对于某些数据集，线性模型显著优于非线性模型。并且在所有测试结果中，两种模型的上界值相对误差几乎为 0，区别在于证明上界的最优化以及求解所

需的时长。这表明对于线性模型 Gurobi 可能已经得到了一个近似最优或最优的上界，Gurobi 的大部分计算时间在证明其最优性。注意到在本问题中，线性化技术解决的问题是客户的试错序列中的概率递推关系。而该问题本质上是一个基本最短路问题，有学者^[97]开发了 Branch and Cut 算法求解最短路问题，向模型中增加有效割平面的 Branch and Cut 算法可能会更加适用于线性模型。

5.7 算例结果及灵敏度分析

本章的前几个小节验证了算法的有效性，本节将展示算例的选址结果并对模型中重要参数的灵敏度展开分析。由于本文可以求解 150 个客户及 150 个选址点的大规模算例，因此围绕 150 个点的数据集展开结果分析。

5.7.1 结果展示

选取 150 个点的数据集，设置参数 $R = 5$ ，即每个客户最多拥有 1 个常用节点、3 个备用节点和 1 个虚拟节点，设置参数 $\rho = 0.1$ 。得到选址方案及网络拓扑结构如图 5-8 所示，图中蓝色点表示客户点，红色点表示选址点，图中节点和客户之间的直线表示分配关系，节点与节点之间的直线表示客户试错策略中两节点的

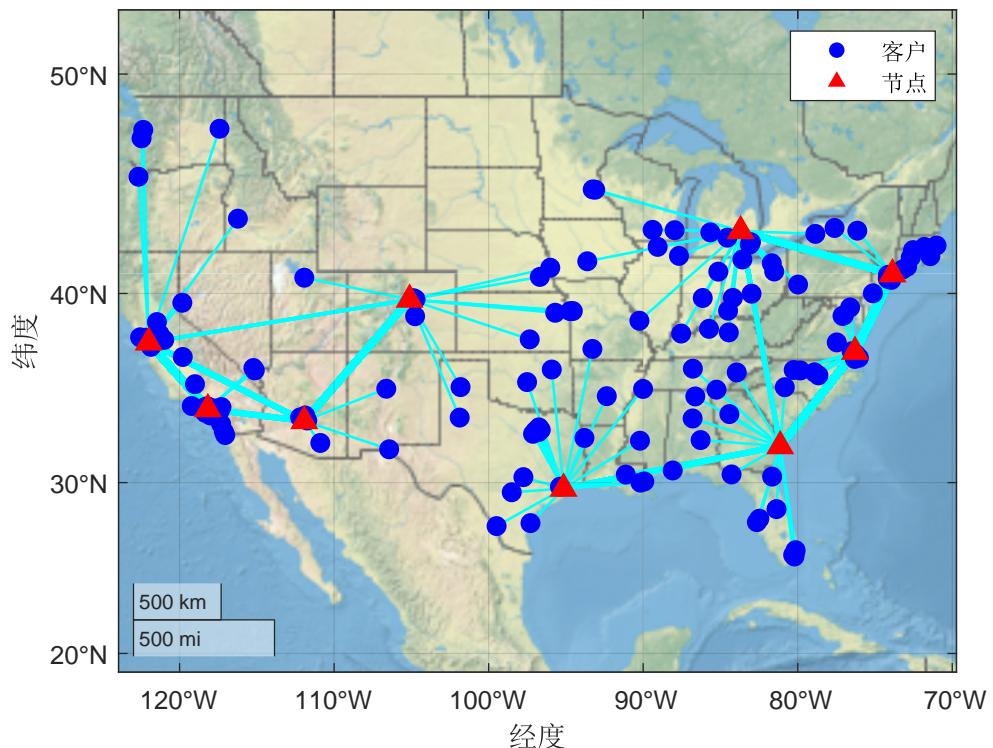


图 5-8 算例结果
Fig 5-8 Results for numerical study

关联性，两个若节点同时作为客户的相邻等级的常用节点或备用节点，则两个节点的直线更粗。

在该网络中，优化结果共选取了 9 个点作为建设节点，为全网络中 150 个客户提供服务。图5-8中所有点的分布呈现西疏东密的趋势，但在选址结果上，东西两个部分的选址数量接近。空间上，节点建设更偏向于沿海地区，中部地区建设节点数量较少。这与经典选址理论将节点安排在拓扑结构相对中心的位置的经验结果是相悖的，产生这样现象的原因是节点的损坏概率和建设费用对选址结果的共同作用。

图5-9展示了每个客户的常用节点以及一级至三级备用节点。空间上，该图中每个客户与其常用节点的距离相对较近，当常用节点失效时，客户会前往其余备用节点。从网络的拓扑结构可发现，该网络可分成两个独立的网络，第一个子网络由经度上从左向右 4 个节点以及相关客户构成，第二个子网络由其余 5 个节点和相关客户构成。产生这样现象的原因是客户和节点的空间位置分布呈现明显的左右分割特征。由于客户的试错策略以及节点的失效概率，即使某些备用节点空间上更加靠近客户，但其并不能成为低等级的备用节点。

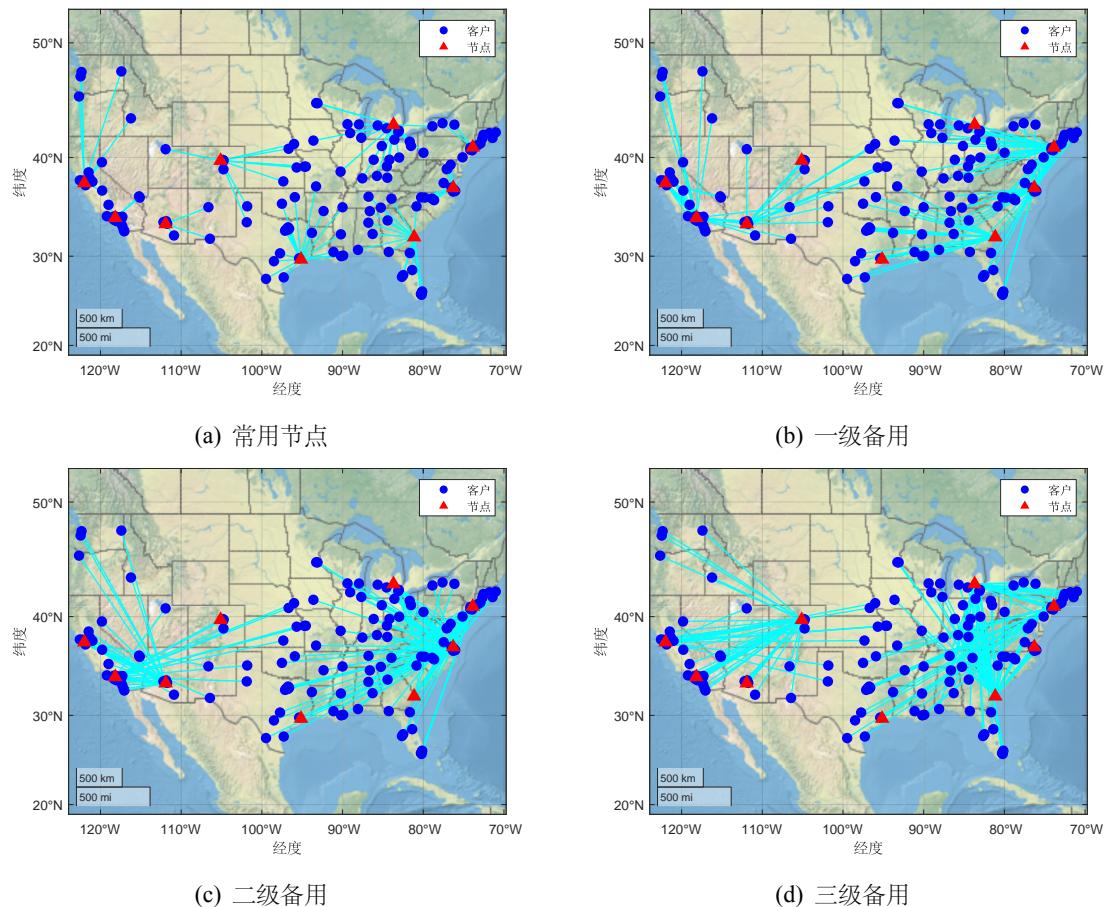


图 5-9 客户的常用节点和备用节点

Fig 5-9 Primary and backup nodes for customers

5.7.2 参数 ρ 灵敏度分析

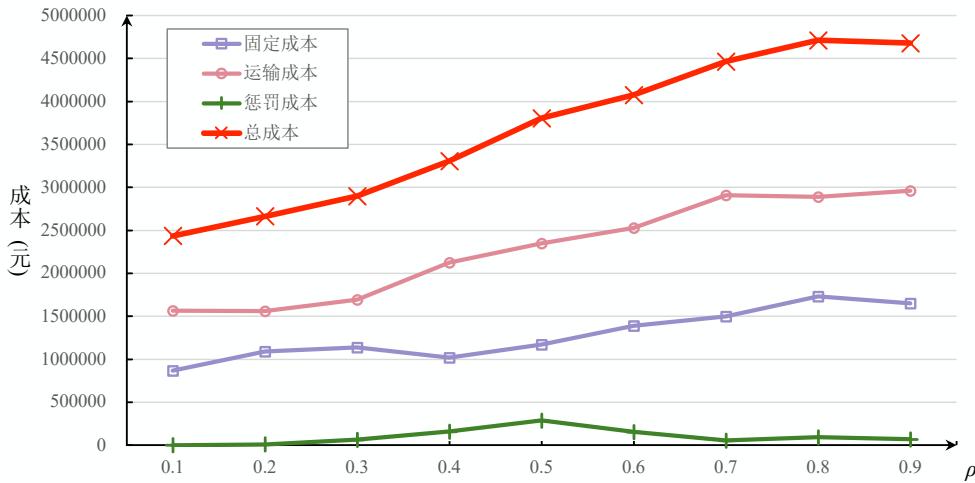
参数 ρ 控制节点的失效概率，本小节分析了在参数 $R = 5$ 的情况下，参数 ρ 的取值对选址结果和成本的影响。其中，参数 ρ 取 0.1 至 0.9 之间，间隔为 0.1 的值。相关选址结果及成本如表5-6所示。在本文中，惩罚成本计算在期望运输成本中（参考第3章模型目标函数），为了分析成本的构成及变化情况，本小节将期望运输成本拆分成运输成本和惩罚成本。

表 5-6 不同参数 ρ 取值的选址结果。Table 5-6 Results for different ρ .

ρ	选址方案	总成本	固定成本	期望运输成本	
				运输成本	惩罚成本
0.1	81,94,120,123,127,131,141,142,150	2435470.40	870000.00	1564493.34	977.06
0.2	81,94,112,120,123,126,127,131,142,150	2663160.76	1090000.00	1562646.61	10514.15
0.3	81,86,94,120,123,127,131,132,138,142,150	2898709.21	1140000.00	1694377.11	64332.10
0.4	81,94,95,120,122,132,142,143,150	3306483.46	1020000.00	2126422.07	160061.38
0.5	86,88,94,112,120,122,126,135,142	3804266.94	1170000.00	2347124.47	287142.46
0.6	18,65,81,120,126,135,137,140,142	4075532.71	1390000.00	2531507.27	154025.44
0.7	8,31,54,57,65,110	4464655.64	1500000.00	2910757.78	53897.86
0.8	27,54,61,65,66,124,126	4713189.05	1730000.00	2889043.65	94145.40
0.9	8,9,43,54,102,122	4679935.27	1650000.00	2960134.85	69800.42

随着参数 ρ 增加，选址节点的数量在不断变化，且选址方案变化较大。在相对较小的 ρ 取值下，结果表明候选点 81、94、112、120、122、127、131、142、150 更易被选为建设节点，当 ρ 值较大时，节点建设方案不固定且变化巨大。这表明，节点的失效概率影响选址结果，特别是影响选址建设方案。在参数 ρ 变化一定程度内，选址方案变动不大。当超过一定值时，选址方案随参数的变化发生巨大变动。

综合分析成本，随着节点的失效概率增加，系统的总成本呈现增加趋势，为可视化各项成本的变动，图5-10展示了各项成本随参数 ρ 变化的曲线。首先，总成本曲线表明系统的总成本随 ρ 的增加而增加。其次，三项成本之间的权衡曲线表明，随着节点失效概率的增长，最开始，系统不得不建设更多的设施、产生更多的固定成本以缓和总成本的增加，接着运输成本大幅上升，系统不得不缩减固定成本以降低总成本增长趋势。运输成本随节点失效概率的增加而增加，相应结果下固定成本下降，产生“效益背反”现象。最后，系统的惩罚成本随节点的失效概率的增加先上升后下降，同样为系统的成本权衡结果。为了降低总成本，系统不得不牺牲一部分的可靠性，换取相对较低的总成本。

图 5-10 成本随 ρ 变动的权衡曲线Fig 5-10 Trade-off curves for the variation of each cost with ρ

5.7.3 参数 R 灵敏度分析

本小节将分析参数 R 对选址结果的影响，设置参数 $\rho = 0.1$ ，参数 R 等于 2 至 10 的不同整数值，即每个客户拥有 0 个至 8 个备用节点。参数 R 控制每个客户拥有的常用节点、备用节点和虚拟节点的总数量。已知每个客户必须拥有一个虚拟节点和一个常用节点，则 R 的最小值为 2。表 5-7 展示了不同 R 取值下的选址方案和成本构成。

表 5-7 不同参数 R 取值的选址结果。Table 5-7 Results for different R .

R	选址方案	总成本	固定成本	期望运输成本	
				运输成本	惩罚成本
2	8,13,54,66	4670884.11	1620000	1984693.82	1066190.29
3	81,94,95,120,127,142,149,150	2591989.01	890000	1510101.65	191887.36
4	81,94,120,123,127,131,141,142,149,150	2410503.11	1000000	1395715.25	14787.86
5	81,94,120,123,127,131,141,142,149,150	2397743.05	1000000	1396787.65	955.40
6	79,81,94,120,123,127,131,138,141,142,150	2398531.47	1130000	1268481.33	50.14
7	81,94,120,123,127,131,138,141,142,150	2402250.12	1000000	1402247.14	2.98
8	81,94,120,123,127,131,141,142,149,150	2396859.14	1000000	1396858.93	0.21
9	81,94,120,123,127,131,141,142,149,150	2396858.96	1000000	1396858.95	0.01
10	81,94,120,123,127,131,141,142,149,150	2396858.95	1000000	1396858.95	0.00

首先分析选址方案的变化，当 $R = 2$ 时，此时问题退化成 UFL 问题，此时选址的数量较少，但固定成本较高。由于未向任何客户指派备用节点，因此 $R = 2$ 时方案的惩罚成本非常高。当 $R = 3$ 时，即向每个客户指派 1 个备用节点，惩罚成本大幅度下降。这说明向客户即便向每个客户仅指派 1 个备用节点，也能显著提高网络的可靠性。随着每个客户备用节点数量的增加，惩罚成本下降。最终选址

方案趋于稳定，总成本不再明显变化。选址方案受参数 R 的影响较小，结果鲁棒性较强，表明在 81、94、120、123、127、131、142、149、150 等选址点处建设设施具有可靠性。

其次，相关成本随 R 的变化如图5-11所示，随着参数 R 的增加，网络的总成本逐渐下降并趋于稳定。产生该现象的原因是为每个客户安排常用节点，降低了惩罚成本，当惩罚成本几乎不变时，选址结果同样不会变动。图5-11中展示的优化曲线表明为每个客户安排一个常用节点和三个备用节点时，该网络已经变得足够可靠，再向其添加备用节点对选址总成本不会产生较大的影响。

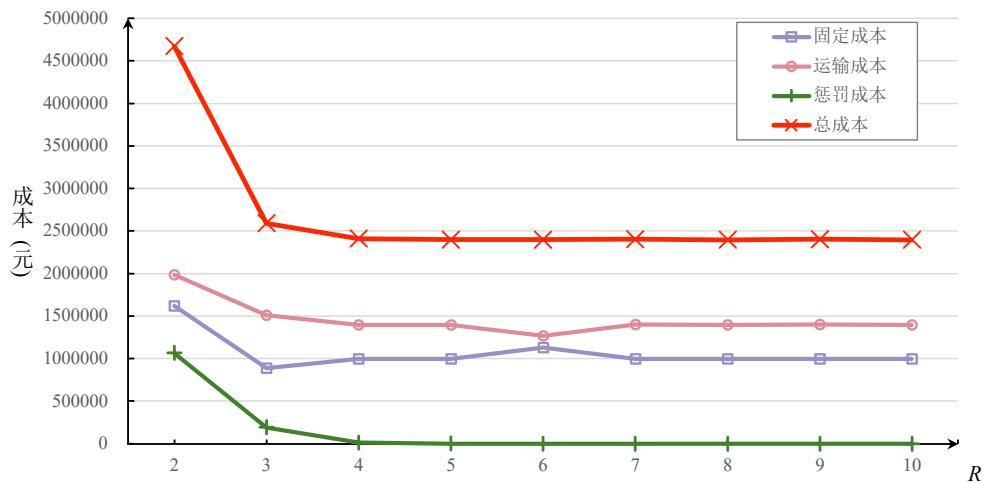


图 5-11 成本随 R 变动的权衡曲线

Fig 5-11 Trade-off curves for the variation of each cost with R

在本算例的灵敏度分析中，系统的可靠性随备用节点个数的增加而增加。然而，仅依靠增加备用节点个数以提高网络可靠性是有边界的。超过此边界后，再增加节点也很难影响选址方案和总成本。在本算例中， $R = 5$ 是每个客户拥有的最佳节点数量。若 R 低于此值，网络变得不可靠，若 R 高于此值，增加了运算量的同时对网络选址结果不产生较大影响。

5.8 本章小结

本章的主要内容为数值实验和算例分析。首先采用经典选址数据集构造了数值实验的算例，对比分析了 LR-ILS 算法与商业求解器的求解结果。结果表明，根据问题定制的 LR-ILS 算法可以在短时间内获得近似最优解，并且可以求解 150 个客户与 150 个候选位置的大规模数据。在绝大多数算例中，LR-ILS 求解结果和求解时间显著优于求解器。此外，数值实验还针对 LR-ILS 算法的各种算子性能和算法设计展开了分析，特别是求解模型上界、下界的性能，以及多种乘子初始化和更新方法的效果。数值实验还讨论了模型线性化的效果，对比分析使用求解器求

解非线性模型和线性模型的差异。最后，给出了算例选址结果，以及重要参数的灵敏度分析结果。值得注意的是，由于该算法使用了元启发式算子，算法本质上带有随机性，因此会造成运行最终结果中上界不稳定的情况。但是，算法的下界提供了上界质量的证明。

6 总结与展望

本章的6.1节总结本文工作内容，并指出了本文可完善的部分，为后续工作指明改进方向，此外，第6.2节还展望了有关可靠选址问题的未来研究方向，提出了当前研究的热点和未来有意义的研究内容。

6.1 工作总结

本文研究了物流节点可靠选址及客户分配模型与算法，研究的主要内容为模型和算法。考虑到物流节点在现实运营中将面临一系列可能导致其失效的风险因素，以及节点失效时客户的行为，本文构建了考虑节点失效风险的物流网络选址模型，并设计了基于迭代局部搜索改进的拉格朗日松弛算法。数值实验结果展示了算法可求解大规模算例的优越性能。最后，本文进行物流节点可靠选址算例计算，为物流节点选址提供了理论支撑。论文的主要研究内容总结和结论如下：

(1) 基于可靠选址理论，考虑到客户的不完全信息状态，本文描述了客户的试错过程，构建了节点端提供服务的物流节点可靠选址问题，该问题被证明为 NP-hard 问题。本文的问题可构建为非线性整数规划模型，使用线性化技术消除模型中的非线性成分。此外，本文提出了一个与该问题相关的最短路问题，并给出了该问题的数学模型。

(2) 本文定制了基于迭代局部搜索改进的拉格朗日松弛算法，启发式算子快速获得模型的上下界的近似值，深度优先搜索算法再对上下界结果进行改进，得到上下界的精确值。迭代局部搜索算子对拉格朗日松弛上界选址方案进行局部搜索以进一步提升上界质量。

(3) 本文对比了基于迭代局部搜索改进的拉格朗日松弛算法和求解器的性能，实验结果表明基于迭代局部搜索改进的拉格朗日松弛算法具有求解大规模问题和现实案例的能力。本文详细讨论了算法每个算子的效果，通过数值实验对比了算子的效率，虽然深度优先搜索算子是非多项式时间算法，但在求解该问题时仍展现了良好的效果。

(4) 本文进行了灵敏度分析，结果表明：每个客户拥有一个常用节点和两个备用节点即可提升网络的可靠性。在一定范围内，随着运价和失效概率的提升，系统的总成本提升但能保持选址方案不变。当运价和失效概率的提升超过一定幅度，增加节点建设数量是降低系统成本、维持系统可靠性的方法之一。

限于作者的自身理论水平以及代码编写水平，本文的研究内容也有一系列的局限性。模型方面，本文描述了客户常用节点失效后的试错以寻找设施的行为，绝

对的不完全信息场景和绝对的完全信息场景都是假设前提，但现实中的情况远远比实际情况复杂，很可能是不完全信息场景和完全信息场景的混合。此外，模型假设每个节点的失效是彼此独立的，该假设主动忽视了节点之间的联系。算法方面，虽然拉格朗日松弛算法能提供一个相对较紧的下界，但对于大规模的问题，该算法的效果同样有限。并且，拉格朗日松弛算法在大多数情况下只能得到一个近似最优解。此外，算法的算子为非多项式时间算法，对于一些特殊且复杂的问题，在最差的情况下无法保证求解时间。最后，受限于作者编写代码水平和 Matlab 解释型语言的性能，当前算法仍有进一步改进的空间。

6.2 研究展望

针对当前研究的局限性，未来研究的可改进之处以及研究方向如下：

(1) 现有模型考虑因素的改进。

未来研究可对当前的模型考虑的不完全信息场景进一步改进，可混合完全信息和不完全信息场景，使得模型更加贴近现实。此外，现实中风险可导致多个节点同时失效，模型的节点失效独立性假设也可进一步改进。如何构建节点之间的失效关联方程，特别是离散选址模型中该方程的定义，是未来研究的重点。对于当前研究的可靠选址问题，未来的研究中还可将其应用至更多场景中，例如可靠基础设施选址（电动车充电站、零售企业选址）等，还可与其他经典选址问题融合，构建可靠选址-路径问题、可靠选址-库存问题、多周期可靠选址问题等。

(2) 现有算法的改进。

在本文原模型的上界和下界的求解过程中，都不可避免地反复求解一个最短路问题。该最短路问题考虑了每个节点的失效可能，具有一定的特殊性，使得经典求解最短路问题的算法失效。但是，在该最短路问题中，所有弧的权重为正值，利用该特性可以降低求解该问题的复杂性。由于该问题同样为 NP-hard 问题，因此求解该问题的不存在多项式时间算法，但可能存在伪多项式时间算法。可考虑将脉冲算法、标签算法移植到本问题中，推动求解原问题的进程，使得算法更具有效率。

(3) 求解模型的其他算法。

使用拉格朗日松弛算法得到结果不能保证其最优性，但可获得解的下界。利用本文中构建迭代局部搜索算子的思路，可以开发出元启发式算法。但元启发式算法只能给出一个可行解，并不能证明解的质量。所以，未来研究的另一个方向是开发求解模型的精确算法。Benders 分解是一种求解混合整数规划问题、选址问题常用的精确算法。开发求解本问题的 Benders 分解算法，是未来研究中有深度的算法创新。

参考文献

- [1] 唐仁敏. 推动“十四五”时期现代物流高质量发展 [J]. 中国经贸导刊, 2023, 1039 : 20–28.
- [2] 王微. 在现代化新征程中更好地发挥现代物流的循环畅通作用 [J]. 中国远洋海运, 2023 : 24–26+8.
- [3] 中国航务周刊编辑部. 深圳港口拥堵严重, 部分船公司宣布跳港 [J]. 中国航务周刊, 2021 : 20.
- [4] SNYDER L V, DASKIN M S. Reliability Models for Facility Location: The Expected Failure Cost Case[J]. Transportation Science, 2005, 39(3) : 400–416.
- [5] BERMAN O, KRASS D, MENEZES M B C. Facility Reliability Issues in Networkp-Median Problems: Strategic Centralization and Co-Location Effects[J]. Operations Research, 2007, 55(2) : 332–350.
- [6] BERMAN O, KRASS D, MENEZES M B C. Locating Facilities in the Presence of Disruptions and Incomplete Information[J]. Decision Sciences, 2009, 40(4) : 845–868.
- [7] DASKIN M S. Network and Discrete Location : Models, Algorithms, and Applications[M]. Somerset, UNITED STATES : John Wiley & Sons, Incorporated, 2013.
- [8] 董鹏. 基于信息失效情景的区域物流网络可靠性选址问题研究 [D]. 北京: 北京交通大学, 2018.
- [9] WEBER A. Ueber den Standort der Industrien[M]. Tübingen, Germany : Mohr, 1922.
- [10] MAK H-Y, SHEN Z-J M. Integrated Modeling for Location Analysis[J]. Foundations and Trends® in Technology, Information and Operations Management, 2016, 9(1–2) : 1–152.
- [11] 周渝峰. 考虑失灵风险的可靠性设施选址问题综述 [J]. 重庆工商大学学报 (自然科学版), 2016, 33(02) : 58–67.
- [12] ALBAREDA-SAMBOLA M, HINOJOSA Y, PUERTO J. The reliable p-median problem with at-facility service[J]. European Journal of Operational Research, 2015, 245(3) : 656–666.
- [13] LIM M, DASKIN M S, BASSAMBOO A, et al. A facility reliability problem: Formulation, properties, and algorithm[J]. Naval Research Logistics (NRL), 2010, 57(1) : 58–70.
- [14] 李汉卿. 中断风险下的供应链选址策略改进 [D]. 北京: 北京交通大学, 2014.
- [15] CUI T, OUYANG Y, SHEN Z-J M. Reliable Facility Location Design Under the Risk of Disruptions[J]. Operations Research, 2010, 58(4-part-1) : 998–1011.
- [16] SHEN Z-J M, ZHAN R L, ZHANG J. The Reliable Facility Location Problem: Formulations, Heuristics, and Approximation Algorithms[J]. INFORMS Journal on Computing, 2011, 23(3) : 470–482.
- [17] 王艳敏. 基于可靠性的供应链设施选址问题的优化模型 [J]. 科学技术与工程, 2012, 12(11) : 2517–2520.
- [18] ABOOLIAN R, CUI T, SHEN Z-J M. An Efficient Approach for Solving Reliable Facility Location Models[J]. INFORMS Journal on Computing, 2013, 25(4) : 720–729.

- [19] YUN L, QIN Y, FAN H, et al. A reliability model for facility location design under imperfect information[J]. *Transportation Research Part B: Methodological*, 2015, 81 : 596–615.
- [20] YUN L, WANG X, FAN H, et al. A reliable facility location design model with site-dependent disruption in the imperfect information context[J]. *PLoS One*, 2017, 12(5) : e0177104.
- [21] YUN L, WANG X, FAN H, et al. Reliable facility location design with round-trip transportation under imperfect information Part I: A discrete model[J]. *Transportation Research Part E: Logistics and Transportation Review*, 2020, 133 : 101825.
- [22] LI Q, ZENG B, SAVACKIN A. Reliable facility location design under disruptions[J]. *Computers & Operations Research*, 2013, 40(4) : 901–909.
- [23] LI Q, SAVACKIN A. A heuristic approach to the design of fortified distribution networks[J]. *Transportation Research Part E: Logistics and Transportation Review*, 2013, 50 : 138–148.
- [24] LI Q, SAVACKIN A. A Fast Tabu Search Algorithm for the Reliable P-Median Problem[C] // GAO D, RUAN N, XING W. Springer Proceedings in Mathematics & Statistics, Vol 95 : ADVANCES IN GLOBAL OPTIMIZATION. 233 SPRING STREET, NEW YORK, NY 10013, UNITED STATES : SPRINGER, 2015 : 417–424.
- [25] YU G, HASSELL W B, LIU Y. Resilient facility location against the risk of disruptions[J]. *Transportation Research Part B: Methodological*, 2017, 104 : 82–105.
- [26] YU G, ZHANG J. Multi-dual decomposition solution for risk-averse facility location problem[J]. *Transportation Research Part E: Logistics and Transportation Review*, 2018, 116 : 70–89.
- [27] LU M S, RAN L, SHEN Z J M. Reliable Facility Location Design Under Uncertain Correlated Disruptions[J]. *M&Som-Manufacturing & Service Operations Management*, 2015, 17(4) : 445–455.
- [28] DU B, ZHOU H, LEUS R. A two-stage robust model for a reliable p-center facility location problem[J]. *Applied Mathematical Modelling*, 2020, 77 : 99–114.
- [29] PENG P, SNYDER L V, LIM A, et al. Reliable logistics networks design with facility disruptions[J]. *Transportation Research Part B: Methodological*, 2011, 45(8) : 1190–1211.
- [30] LI Y, LI X, SHU J, et al. A General Model and Efficient Algorithms for Reliable Facility Location Problem Under Uncertain Disruptions[J]. *INFORMS Journal on Computing*, 2022, 34(1) : 407–426.
- [31] AN Y, ZENG B, ZHANG Y, et al. Reliable p-median facility location problem: two-stage robust models and algorithms[J]. *Transportation Research Part B: Methodological*, 2014, 64 : 54–72.
- [32] LI X, OUYANG Y. A continuum approximation approach to reliable facility location design under correlated probabilistic disruptions[J]. *Transportation Research Part B: Methodological*, 2010, 44(4) : 535–548.
- [33] YUN L, FAN H, LI X. Reliable facility location design with round-trip transportation under imperfect information part II: A continuous model[J]. *Transportation Research Part B: Methodological*, 2019, 124 : 44–59.
- [34] FAN H Q, YUN L F, LI X P. A linear-time crystal-growth algorithm for discretization of continuum approximation[J]. *Transportation Research Part E-Logistics and Transportation Review*, 2022, 161 : 26.

- [35] JIANG Z, OUYANG Y. Reliable location of first responder stations for cooperative response to disasters[J]. *Transportation Research Part B: Methodological*, 2021, 149 : 20–32.
- [36] LI X P, OUYANG Y F, PENG F. A supporting station model for reliable infrastructure location design under interdependent disruptions[J]. *Transportation Research Part E-Logistics and Transportation Review*, 2013, 60 : 80–93.
- [37] BERMAN O, DREZNER Z, WESOLOWSKY G O. Locating service facilities whose reliability is distance dependent[J]. *Computers & Operations Research*, 2003, 30(11) : 1683–1695.
- [38] CHENG C, ADULYASAK Y, ROUSSEAU L-M. Robust facility location under demand uncertainty and facility disruptions[J]. *Omega*, 2021, 103 : 102429.
- [39] LIM M K, BASSAMBOO A, CHOPRA S, et al. Facility location decisions with random disruptions and imperfect estimation[J]. *Manufacturing & Service Operations Management*, 2013, 15(2) : 239–249.
- [40] SNYDER L V. Facility location under uncertainty: a review[J]. *IIE Transactions*, 2006, 38(7) : 547–564.
- [41] SNYDER L V, ATAN Z, PENG P, et al. OR/MS models for supply chain disruptions: a review[J]. *IIE Transactions*, 2015, 48(2) : 89–109.
- [42] GOVINDAN K, FATTABI M, KEYVANSHOKOOH E. Supply chain network design under uncertainty: A comprehensive review and future research directions[J]. *European Journal of Operational Research*, 2017, 263(1) : 108–141.
- [43] FISCHETTI M, LJUBIĆ I, SINNL M. Redesigning Benders decomposition for large-scale facility location[J]. *Management Science*, 2017, 63(7) : 2146–2162.
- [44] ORTIZ-ASTORQUIZA C, CONTRERAS I, LAPORTE G. An exact algorithm for multilevel uncapacitated facility location[J]. *Transportation Science*, 2019, 53(4) : 1085–1106.
- [45] WENTGES P. Accelerating Benders' decomposition for the capacitated facility location problem[J]. *Mathematical Methods of Operations Research*, 1996, 44 : 267–290.
- [46] ROHANINEJAD M, SAHRAEIAN R, TAVAKKOLI-MOGHADDAM R. An accelerated Benders decomposition algorithm for reliable facility location problems in multi-echelon networks[J]. *Computers & Industrial Engineering*, 2018, 124 : 523–534.
- [47] AZAD N, HASSINI E. A Benders Decomposition Method for Designing Reliable Supply Chain Networks Accounting for Multimitigation Strategies and Demand Losses[J]. *Transportation Science*, 2019, 53(5) : 1287–1312.
- [48] PIRNIYA G. Reliable Location Allocation Routing Design under Disruption: An Improved Column Generation Decomposition Approach[D]. Montreal, Quebec, Canada : Concordia University, 2017.
- [49] 孙小玲. 整数规划 [M]. 北京 : 科学出版社, 2010.
- [50] HEARN D W, LOWE T J. Lagrangian duality: BASICS[M] // FLOUDAS C A, PARDALOS P M. Encyclopedia of Optimization. Boston, MA : Springer US, 2009 : 1805–1813.
- [51] GEOFFRION A M. Lagrangean relaxation for integer programming[M] // BALINSKI M L. Approaches to Integer Programming. Berlin, Heidelberg : Springer Berlin Heidelberg, 1974 : 82–114.

- [52] CORNUJOLS G, FISHER M L, NEMHAUSER G L. Exceptional Paper—Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms[J]. Management Science, 1977, 23(8) : 789–810.
- [53] XIE W, OUYANG Y, WONG S C. Reliable Location-Routing Design Under Probabilistic Facility Disruptions[J]. Transportation Science, 2016, 50(3) : 1128–1138.
- [54] WANG J, SU K, WU Y. The Reliable Facility Location Problem Under Random Disruptions[J]. Wireless Personal Communications, 2018, 102(4) : 2483–2497.
- [55] FISHER M L. The Lagrangian Relaxation Method for Solving Integer Programming Problems[J]. Management Science, 2004, 50(12) : 1861–1871.
- [56] RAHMANI A, MIRHASSANI S A. A hybrid Firefly-Genetic Algorithm for the capacitated facility location problem[J]. Information Sciences, 2014, 283 : 70–78.
- [57] RAINWATER C, GEUNES J, ROMEIJN H E. A facility neighborhood search heuristic for capacitated facility location with single-source constraints and flexible demand[J]. Journal of Heuristics, 2011, 18(2) : 297–315.
- [58] SILVA M R, CUNHA C B. A tabu search heuristic for the uncapacitated single allocation p-hub maximal covering problem[J]. European Journal of Operational Research, 2017, 262(3) : 954–965.
- [59] RAHDAR S, GHANBARI R, GHORBANI-MOGHADAM K. Tabu search and variable neighborhood search algorithms for solving interval bus terminal location problem[J]. Applied Soft Computing, 2022 : 116.
- [60] AFIFY B, RAY S, SOEANU A, et al. Evolutionary learning algorithm for reliable facility location under disruption[J]. Expert Systems with Applications, 2019, 115 : 223–244.
- [61] 窦志武, 邵亚楠, 原智慧, 等. 物流节点选址研究综述 [J]. 物流工程与管理, 2020, 42(07) : 1–4.
- [62] 王柯. 基于层次分析法的亚龙商贸物流园区选址研究 [D]. 昆明: 昆明理工大学, 2017.
- [63] 李楠. 地下物流网络的节点选址和线路优化 [D]. 武汉: 江汉大学, 2020.
- [64] 王项, 刘刚. 基于数据包络分析的区域物流绩效分析——以全国性物流节点城市所在的省市为例 [J]. 物流工程与管理, 2015, 37 : 78–81.
- [65] 张莹. 考虑中断风险的供应链优化模型和算法研究 [D]. 北京: 清华大学, 2016.
- [66] 汤罗浩. 考虑设施失效的选址问题、模型与算法 [D]. 长沙: 国防科学技术大学, 2016.
- [67] 王继光. 中断情境下供应链设施选址优化研究 [D]. 太原: 山西大学, 2015.
- [68] 周浩. 考虑线状需求的可靠性物流节点选址研究 [D]. 北京: 北京交通大学, 2020.
- [69] 朱江华, 崔娜, 朱庆, 等. 服务失效情况下可靠的物流园区选址模型 [J]. 济南大学学报(自然科学版), 2019, 33(02) : 101–106+114.
- [70] 颜高民. 突发事件下应急物流可靠路径的搜索 [D]. 武汉: 华中科技大学, 2010.
- [71] 李锐, 李晓会, 陈鑫. 可靠性绿色物流配送选址-路径问题研究 [J]. 计算机工程与应用, 2020, 56(23) : 237–244.
- [72] 任慧, 王东宇. 考虑多源协同供应链可靠选址路径集成优化问题 [J]. 运筹与管理, 2023, 32(02) : 132–138.

- [73] 石褚巍, 马昌喜, 麻存瑞. 基于两阶段鲁棒优化的可靠性物流网络设计 [J]. 交通运输系统工程与信息, 2023: 1–20.
- [74] 张鹏阁. 考虑配送中心中断的应急物流选址-路径优化研究 [D]. 郑州: 郑州大学, 2021.
- [75] 孙晓飞, 张强. 物流配送中心选址的多目标优化模型 [C] // 中国运筹学会企业运筹学分会. 中国企业运筹学 [2010(1)]. 吉林: 电子科技大学出版社, 2010: 176–182.
- [76] 姚琦. 基于可靠性的生鲜农产品配送中心选址研究 [D]. 大连: 大连交通大学, 2012.
- [77] 周渝峰, 马祖军, 王恪铭. 应急物资储备库的可靠性 P-中位选址模型 [J]. 管理评论, 2015, 27(05): 198–208.
- [78] 李政祥. 基于可靠性的应急物流多目标选址问题模型研究 [J]. 商, 2016(30): 262.
- [79] 李政祥. 考虑设施中断情景的可靠性应急物流网络优化设计研究 [D]. 重庆: 重庆工商大学, 2017.
- [80] 朱建明. 基于损毁情景的可靠连通应急设施选址问题 [J]. 电子科技大学学报 (社科版), 2012, 14(03): 44–48.
- [81] 于冬梅, 高雷阜, 赵世杰. 中断情境下可靠性应急设施选址-分配多目标优化模型 [J]. 控制与决策, 2020, 35(06): 1415–1420.
- [82] 王子墨. 基于可靠性的设施选址问题研究 [D]. 青岛: 山东科技大学, 2018.
- [83] TOTH P, VIGO D. Models, relaxations and exact approaches for the capacitated vehicle routing problem[J]. Discrete Applied Mathematics, 2002, 123(1-3): 487–512.
- [84] KARP R M. Reducibility among Combinatorial Problems[M] // MILLER R E, THATCHER J W, BOHLINGER J D. Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations. Boston, MA: Springer US, 1972: 85–103.
- [85] GAREY M, JOHNSON D. Mathematical Sciences Series: Computers and Intractability: A Guide to the Theory of NP-completeness[M]. New York, USA: Freeman, 1979.
- [86] LOZANO L, DUQUE D, MEDAGLIA A L. An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints[J]. Transportation Science, 2016, 50(1): 348–357.
- [87] DROR M. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW[J]. Operations Research, 1994, 42(5): 977–978.
- [88] LOURENÇO H R, MARTIN O C, STÜTZLE T. Iterated Local Search: Framework and Applications[M] // GENDREAU M, POTVIN J-Y. Handbook of Metaheuristics. Boston, MA: Springer US, 2010: 363–397.
- [89] EVERETT H. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources[J]. Operations Research, 1963, 11(3): 399–417.
- [90] FISHER M L. Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part I[J]. Operations Research, 1973, 21(5): 1114–1127.
- [91] 温旭红. 铁路车流分配优化模型与拉格朗日松弛算法求解研究 [D]. 北京: 北京交通大学, 2016.
- [92] FEILLET D, DEJAX P, GENDREAU M, et al. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems[J]. Networks: An International Journal, 2004, 44(3): 216–229.

- [93] SANTOSA B, KRESNA I G N A. Simulated Annealing to Solve Single Stage Capacitated Warehouse Location Problem[J]. Procedia Manufacturing, 2015, 4 : 62 – 70.
- [94] MICHEL L, VAN HENTENRYCK P. A simple tabu search for warehouse location[J]. European Journal of Operational Research, 2004, 157(3) : 576 – 591.
- [95] BO L, BISSAN G, JATIN N. Electric vehicle routing with charging/discharging under time-variant electricity prices[J]. Transportation Research Part C, 2021, 130 : 103285.
- [96] SCHNEIDER M. The vehicle-routing problem with time windows and driver-specific times[J]. European Journal of Operational Research, 2016, 250(1) : 101 – 119.
- [97] JEPSEN M, PETERSEN B, SPOORENDONK S. A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with Resource Constraints[J]. Cut and Column Generation, 2008 : 129.

附录 A

ILS-LR 算法源码

该代码包含算法的主程序及所有函数，算法所需的数据请见正文中的参考文献。此外，作者的 Github 开源仓库¹ 提供了同样的代码、数据以及环境配置方法，以方便感兴趣的读者复现。本附录的第一部分是 LR-ILS 算法的 Matlab 算法，第二部分是 Gurobi 建模 Python 代码。

LR-ILS 算法源码 (Matlab)

```

1 % 拉格朗日松弛求解可靠设施选址问题(RUFL)
2 % 编码方式: UTF-8
3 % 操作系统: Windows 10/11 & MacOs 13
4 % 运行平台:
5 % Matlab R2022a/b
6 % 工具箱 Matlab Coder & Parallel Computing
7 % Gurobi 9.5.2
8 % Python 3.10
9
10 %% 清除
11 clc
12 clear all
13 clear classes
14 close all
15 diary off
16 diary 'LR.log'
17 disp(['程序开始:', datestr (now)])
18
19 %% 参数控制
20 lr_para = struct(); % LR 算法的参数
21 lr_para.alpha = 2; % 步长参数初始值
22 lr_para.alpha_min = 0.0001; % 步长参数最小值
23 lr_para.theta_lr = 1.05; % 步长比例系数
24 lr_para.kappa_lb = 10; % 下界连续不变次数
25 lr_para.eta_lr = 3000; % 迭代次数
26 lr_para.tau_lim = 1000; % 优化时长
27 lr_para.xi = 0.01; % 接受gap
28 lr_para.theta_sa = 1.2; % 模拟退火初始温度参数
29 lr_para.T_lim = 0.0001; % 模拟退火最低温度
30 lr_para.kappa_ub = 100; % 触发ILS上界连续不变次数 (200)
31 lr_para.kappa_ubdfs = 500; % 触发上界DFS搜索(delete)
32 lr_para.eta ils = 10; % ILS迭代次数
33 lr_para.grb_model = 1; % 使用Gurobi建模 取值 0/1
34 lr_para.grb_ub = 1; % 使用Gurobi获取上界 取值0/1

```

¹https://github.com/yrf990409/LR_for_RUFL

```

35 lr_para.dfs_gap = 0.2;      % gap小于此值才启动DFS
36 lr_para.print = true;      % 解池的大小
37
38 %% 案例参数
39 % lr_case = struct ();
40 % path = './data/SnyderData/49nodes/';
41 lr_case.data = data_reader(path);
42 lr_case.rho = prob_rho;          % 损坏概率控制参数
43 lr_case.q = lr_case.rho * exp(-lr_case.data.fix/200000); % 损坏概率
44 lr_case.q(1) = 1;               % 虚拟设施的损坏概率
45 lr_case.max_try = out_index;    % 最大尝试次数(R)
46 lr_case.cus_num = length(lr_case.data.dmd); % 客户数量
47 lr_case.node_num = size(lr_case.data.price,1); % (虚拟实体客户)的总数
48 lr_case.fac_num = lr_case.node_num - lr_case.cus_num - 1; % 设施总数
49 lr_case.bar_J = (0: lr_case.fac_num)+1; % 设施集合拓展集
50 lr_case.I = (lr_case.fac_num+1 : lr_case.node_num-1)+1; % 客户集合
51 lr_case.mu = zeros(lr_case.cus_num, length(lr_case.bar_J)); % 生成初始乘子
52 for i = 1: lr_case.cus_num
53     cus_ind = lr_case.I(i)-lr_case.I(1)+1;
54     for j = lr_case.bar_J(2:end)
55         lr_case.mu(:,j) = (lr_case.data.dmd(cus_ind) * lr_case.data.price(cus_ind,j))
56             + ...
57             lr_case.data.fix(j)) / length(lr_case.data.fix);
58     end
59 end
60
61 %% 优化
62 lr_result = lr_ils(lr_para, lr_case);
63
64 %% 结果处理
65 % draw_fig(lr_result)
66
67 %% 保存
68 file_name = ['结果', ...
69     num2str(lr_case.node_num), '—', ... % 点数
70     num2str(lr_case.rho), '—', ... % rho取值
71     num2str(lr_case.max_try), '—', ... % 最大试错次数
72     '.mat'];
73 save(file_name, 'lr_result')
74
75 % fprintf('rho \t 最大试错次数 \t 上界 \t 下界 \t 求解时间\n')
76 fprintf('%d\t%2d\t%2d\t%.2f\t%.2f\t%.2f\t%.2f\n', lr_case.node_num, lr_case.
77     rho, lr_case.max_try, lr_result.bst_ub, lr_result.bst_lb, lr_result.time)
78 diary off
79
80 function data = data_reader(f_path)
81 %DATA_READER 读取数据
82 % data包含三个数据
83 %% 数据读取
84 % disp(['导入数据, 当前路径: ', f_path])
85 data.price = load([f_path, 'cost.csv']); % 价格矩阵

```

```

84 data.dmd = load([f_path, 'dmd.csv']); % 客户需求
85 data.fix = load([f_path, 'fc.csv']); % 固定成本
86 % disp('导入完成!')
87 % disp('-----')
88
89 % 验证数据
90 validateattributes (data.price, {'double'}, {'size', [length(data.dmd)+length(data.fix)
91     ), length(data.fix)]})
92
93
94 function [ub,best_r] = lb_dfs(r, best_r, ub, cur_cost, cur_prob, fac, cus_dmd, ...
95     cost_mat, R, pi, probDisr, cus_mu)
96 % global x
97 % x = x+1
98 if length(r) <= R+1 % 没有到达设施数量的上界
99     if r(end) == 1 % 如果结尾是虚拟, 那么获得了一个完整路径
100         tempCost = cur_cost;
101         if tempCost < ub % 小于上界 则更新
102             ub = tempCost;
103             best_r = r;
104             return
105         end
106     return
107 else % 否则不是一个完整路径
108     if isempty(fac) % 没有设施可以填充了, route最后一个不以虚拟结尾
109         r = [r,1]; % route必须是虚拟收尾了
110
111         tempCost = cur_cost + cus_dmd*pi*cur_prob;
112         if tempCost < ub % 小于上界 则更新
113             ub = tempCost;
114             best_r = r;
115             return
116         end
117     return
118 else % 还有设施可以填充
119     if length(r) == R+1 % 已经填满, 放不下了, 只差一个虚拟设施
120         r = [r,1]; % route必须是0收尾了
121         tempCost = cur_cost + cus_dmd*pi*cur_prob;
122         if tempCost < ub % 小于上界 则更新
123             ub = tempCost;
124             best_r = r;
125             return
126         end
127     return
128 else % 向下继续分支
129     for i = 1:length(fac) % i : index of facilities
130         stackR = r;
131         stackC = cur_cost;
132         stackP = cur_prob;
133

```

```

134     r = [r, fac(i)];
135     tempCost = cur_cost + cost_mat(r(end-1),r(end))*cur_prob*cus_dmd
136     + cus_mu(1,fac(i));
137
138     if tempCost>ub % 剪枝
139         r = stackR; % 直接恢复不再细分
140     else
141         tempFacy = fac; % 递归 继续分支
142         tempFacy(tempFacy==fac(i)) = [];
143         cur_prob = cur_prob*probDisr(r(end));
144         [ub, best_r] = lb_dfs(r, best_r, ub, tempCost, cur_prob, tempFacy,
145                               cus_dmd, cost_mat, R, pi, probDisr, cus_mu);
146         r = stackR;
147         cur_cost = stackC;
148         cur_prob = stackP;
149     end
150 end
151 end
152 else
153     return
154 end
155 end
156
157 function [ trans_cost , plan ] = lb_x( lr_case , flag_fast )
158 %lb_dfs_x 深度搜索获取每个客户的尝试路径
159
160 % 输入 lr_case 字段
161 % I 客户索引 向量
162 % bar_J 设施索引 向量
163 % data 数据 结构体 字段 price 价格矩阵 dmd 客户需求 fix 固定成本
164 % q 每个设施失效概率 向量
165 % mu 拉格朗日乘子 矩阵
166 % max_try 客户最大尝试次数 整数
167 % dfs_falg 是否使用DFS 逻辑值
168
169 % 输出
170 % trans_cost 期望运输成本 浮点数矩阵
171 % plan 计划 整数矩阵
172
173 %% 初始化
174 % 提取
175 I = lr_case.I;
176 bar_J = lr_case.bar_J;
177 data = lr_case.data;
178 q = lr_case.q;
179 mu = lr_case.mu;
180 max_try = lr_case.max_try;
181 trans_cost = zeros(length(I),1); % 记录每个客户的期望运输成本
182

```

```

183 %% 贪心路径构建
184 [plan, trans_cost] = greedy_build(I, max_try, bar_J, data, mu, q, trans_cost);
185
186 %% DFS改进当前解
187 if ~flag_fast % 快速模式不启动dfs
188     dmd = data.dmd;
189     price = data.price;
190     pi = price(2,1); % 惩罚成本
191     parfor j = 1:length(I)
192         cus = I(j);
193         best_r = plan(j,:); % 当前最优路径
194         ub = trans_cost(j); % 当前上界
195         cus_dmd = dmd(j); % 客户的需求
196         cus_mu = mu(j,:); % 拉格朗日乘子
197
198         R = coder.ignoreConst(max_try-1);
199         coder.varsize('cus', [1 100], [0 1]);
200         coder.varsize('best_r');
201         coder.varsize('bar_J');
202         coder.varsize('price');
203         coder.varsize('q');
204         coder.varsize('cus_mu');
205
206         [ub, best_r] = lb_dfs(cus, best_r, ub, 0, 1, bar_J, cus_dmd, price, R, pi, q,
207             cus_mu); % 深度优先搜索
208         trans_cost(j) = ub; % 更新上界
209
210         if length(best_r) < max_try+1
211             plan(j,:) = [best_r, zeros(1, max_try+1-length(best_r))];
212         else
213             plan(j,:) = best_r; % 记录方案
214         end
215     end
216 end
217 end
218
219
220 function [plan, trans_cost] = greedy_build(I, max_try, bar_J, data, mu, q, trans_cost
221 )
222 % 路线构建的贪心算法
223 % 在每个路径的最后增加一个算子
224 plan = zeros(length(I), max_try+1); % 记录每个客户的方案
225 dmd = data.dmd; % 客户的需求
226 price = data.price; % 价格矩阵
227 for k = 1:length(I)
228     cus = I(k);
229     % 直接采用构造法生成一个初始路径
230     route = zeros(1, max_try+1); % 路径
231     route(1) = cus; % 路径的起点是客户
232     prob = 1; % 初始概率是1

```

```

232 fee = 0;    % 初始费用是0
233 fac = bar_J;    % 设施的复制
234 cus_dmd = dmd(k); % 客户的需求
235 jdg_brk = false ;
236
237 mu_k = mu(k,:);
238 % 每次讲增量成本最小的设施放在路线最后（贪心构建）
239 for i = 2:max_try
240     add_fee = zeros(1,length(fac));
241     for j = 1:length(fac)
242         add_fee(j) = prob * cus_dmd * price(route(i-1),fac(j)) ...
243             + mu_k(fac(j));
244     end
245     [min_add_fee,min_ind] = min(add_fee); % 指向最小的费用
246     route(i) = fac(min_ind); % 路线更新
247     prob = prob * q(route(i)); % 概率更新
248     fee = fee + min_add_fee; % 费用更新
249     if route(i) == bar_J(1)
250         jdg_brk = true;
251         break % 如果虚拟设施被添加到路径中则退出
252     end
253     fac(min_ind) = []; % 否则去除已经添加的设施
254 end
255
256 if jdg_brk % 提前退出
257     plan(k,:) = route; % 记录路径
258     trans_cost(k) = fee; % 记录成本
259 else
260     route(route==0) = []; % 去除路径中的多余0
261     trans_cost(k) = fee + cus_dmd * prob * price(route(end),1); % 直接计算成本
262
263     formated_route = [route, 1, zeros(1, max_try-length(route))]; % 记录路径
264     plan(k,:) = formated_route; % 记录路径
265 end
266 end
267 end
268
269
270 function [ cost_for_fac , location ] = lb_y( lr_case )
271 %LB_Y获取y变量的下界
272 % 给定一个乘子，返回y的最佳选址location以及设施建设成本cost_for_fac
273
274 % 初始化
275 bar_J = lr_case .bar_J;
276 data = lr_case .data;
277 mu = lr_case .mu;
278
279 location = false (1,length(bar_J));      % 选址方案
280 cost_for_fac = zeros(1,length(bar_J)); % 下界Sub_I问题的目标函数
281
282 for j = 2:length(bar_J)

```

```

283     fac = bar_J(j);
284     sum_mu = sum(mu(:,fac));
285     jdg = data.fix(fac) - sum_mu; % 判别
286
287     if jdg > 0 % 判别数大于0 不选这个位置
288         location(fac) = 0;
289     else
290         location(fac) = 1; % 小于等于0 选
291         cost_for_fac(fac) = data.fix(fac) - sum_mu;
292     end
293
294 end
295
296 end
297
298 function lr_result = lr_ils (lr_para, lr_case)
299 % LR_ILS 拉格朗日松弛-局部迭代搜索算法
300 % 传入 案例参数lr_case 以及算法参数lr_para
301 % 传出 lr_result 结果
302
303 %% 初始化
304 tic
305 % 记录 recorder
306 rec_lb = zeros(lr_para.eta_lr, 1); % 下界记录 record
307 rec_ub = zeros(lr_para.eta_lr, 1); % 上界记录
308 rec_ils = false (lr_para.eta_lr, 1); % ILS发挥作用记录
309
310 % 计数器 count
311 cnt_step = 0; % 下界连续不上升计数
312 cnt_ils = 0; % 上界连续不下降计数
313 cnt_ub = 0; % 最佳上界保持的迭代次数
314 cnt_iter = lr_para.eta_lr; % 算法真正的迭代次数
315
316 % 最佳解 best_solution
317 bst_loc = false (1, length(lr_case.bar_J)); % 最佳选址方案 location
318 bst_sqc = zeros(length(lr_case.I), lr_case.max_try+1); % 最佳客户序列 sequence
319 bst_ub = inf; % 最佳上界
320 bst_lb = -inf; % 最佳下界
321 gap = 1;
322
323 % 条件判断 flag
324 flag_fast = true; % 算法快速模式
325 flag_modify = false; % 快速&慢速切换时强制更正上下界
326 flag_ils_continue = false; % ILS是否继承上一次的结果
327
328 % 邻域
329 cur_ub = inf;
330 cur_loc = false (1, length(lr_case.bar_J)); % 选址方案
331
332 %% LR-ILS优化
333 for iter = 1:lr_para.eta_lr

```

```

334 % 获取下界
335 [lb_val_x, lb_sqc] = lb_x( lr_case, flag_fast ); % 获取Sub_2的值
336 [lb_val_y, lb_loc] = lb_y( lr_case ); % 获取Sub_1的值
337 lb = sum(lb_val_x) + sum(lb_val_y);
338
339 % 获取上界
340 [ub, sqc, ~] = ub_xy(lr_case, lb_loc, flag_fast );
341
342 % ILS搜索上界
343 if cnt_ub >= lr_para.kappa_ub && gap > 2*lr_para.xi
344     if flag_ils_continue
345         [nb_ub, nb_loc, cur_ub, cur_loc] = ub_ils( lr_case, cur_ub, cur_loc,
346             lr_para );
347     else
348         [nb_ub, nb_loc, cur_ub, cur_loc] = ub_ils( lr_case, bst_ub, bst_loc,
349             lr_para );
350     end
351
352     if nb_ub < bst_ub
353         % 得到更好的上界
354         bst_loc = nb_loc; % 选址方案
355         bst_ub = nb_ub; % 上界
356         rec_ils( iter ) = true ;
357     else
358         flag_ils_continue = true; % ILS继承上一次的结果
359     end
360
361     cnt_ub = 0;
362
363 % 记录最佳上界
364 if ub <= bst_ub
365     % 得到了更好的上界
366     bst_loc = lb_loc; % 选址方案
367     bst_sqc = sqc; % 客户序列
368     bst_ub = ub; % 上界
369     % 计数器更新
370     cnt_ils = 0; % 上界连续不下降
371     cnt_ub = 0; % 最佳上界保持次数
372     % 局部迭代搜索更新
373
374 else
375     % 计数器更新
376     cnt_ils = cnt_ils + 1; % 上界连续不下降
377     cnt_ub = cnt_ub + 1; % 最佳上界保持次数
378
379 % 记录最佳下界
380 if lb >= bst_lb
381     bst_lb = lb;
382     cnt_step = 0;

```

```

383     else
384         cnt_step = cnt_step + 1; % 否则下界未更新计数+1
385     end
386
387     % 更新乘子
388     [ lr_case.mu, cnt_step, lr_para.alpha] = update_mu(lr_case, lr_para, lb, bst_ub,
389     lb_sqc, lb_loc, cnt_step);
390
391     % 迭代记录
392     rec_lb( iter ) = lb;           % 下界记录 record
393     rec_ub( iter ) = bst_ub;       % 上界记录
394
395     % 打印
396     gap = (bst_ub-bst_lb) / bst_ub; % 计算gap
397     t = toc;
398     if lr_para.print
399         formatSpec = " iter :%.0f, best-ub:%.2f, best-lb:%.2f, gap:%.4f%%, time:%.2f,
400         ub:%.2f, lb:%.2f\n";
401         fprintf(formatSpec, iter, bst_ub, bst_lb, gap*100, t, ub, lb);
402     end
403
404     % 快速慢速切换
405     if gap <= lr_para.dfs_gap && ~flag_modify % gap小于一定的值 关闭快速模式
406         flag_fast = false; % 关闭快速模式
407         flag_modify = true; % 强制修正
408         % 快速模式获得的下界不是最优解，因此下界是虚标的，在此处进行修正
409         [lb_val_x, ~] = lb_x(lr_case, flag_fast); % 获取Sub_2的值
410         [lb_val_y, ~] = lb_y(lr_case);             % 获取Sub_1的值
411         bst_lb = sum(lb_val_x) + sum(lb_val_y);    % 强制修正下界
412         gap = (bst_ub-bst_lb) / bst_ub;            % 修正之后的gap
413     end
414
415     % 终止条件
416     if t > lr_para.tau_lim
417         disp('stop,□time') % 时间限制
418         cnt_iter = iter;
419         break
420     else
421         if lr_para.alpha < lr_para.alpha_min
422             disp('stop,□alpha') % 迭代值限制
423             cnt_iter = iter;
424             break
425         elseif gap < lr_para.xi
426             disp('stop,□gap') % gap值达标
427             cnt_iter = iter;
428             break
429         end
430     end
431 end

```

```

432 if cnt_iter == lr_para. eta_lr
433     disp('stop,□ iteration') % 迭代次数限制
434 end
435
436 %% 封装 & 返回
437 % 重新计算
438 t = toc;
439 lr_result = struct();
440 lr_result.rec_lb = rec_lb;
441 lr_result.rec_ub = rec_ub;
442 lr_result.rec_ils = rec_ils;
443 lr_result.bst_loc = bst_loc;
444 lr_result.bst_sqc = bst_sqc;
445 lr_result.bst_ub = bst_ub;
446 lr_result.bst_lb = bst_lb;
447 lr_result.iter = cnt_iter;
448 lr_result.time = t;
449
450
451 end
452
453 function [best_nb_ub, best_nb_loc, current_ub, current_loc] = ub_ils( lr_case, ub,
454     location, lr_para )
455 %UB_ILS 局部迭代搜索
456 %% 初始化
457 best_nb_ub = ub; % 记录全局最佳上界
458 best_nb_loc = location; % 记录全局最佳选址方案
459 current_ub = ub; % 当前上界
460 current_loc = location; % 当前方案
461
462 temperature = -(lr_para.theta_sa-1)*ub/log(0.5); % 初始温度
463 t_gap = temperature / (lr_para.eta_ils *0.4); % 每次降低的温度
464
465 for iter = 1: lr_para.eta_ils
466     % 求邻域
467     nb = get_nb( current_loc ); % 获取邻域
468     nb_sz = size(nb,1);
469
470     % 求所有邻域中成本最小的几个
471     nb_cost_rec = zeros(nb_sz,1);
472     for i = 1: size(nb,1)
473         [nb_cost_rec(i), ~] = ub_xy(lr_case, nb(i,:), true); % 求快速解
474     end
475     [~, sort_ind] = sort(nb_cost_rec); % 最小成本索引
476
477     % 计算前 5 个 最佳索引的精准上界
478     temp_best_loc = nb(sort_ind(1:5),:);
479     temp_cost_rec = zeros(5,1);
480     for i = 1:5

```

```

481 [temp_cost_rec(i), ~] = ub_xy(lr_case, temp_best_loc(i,:), false); %求快速
482     解
483
484 [best_nb_cost, min_ind] = min(temp_cost_rec);
485
486 %接受上界
487 if best_nb_cost < current_ub    %得到更优质的解
488     current_ub = best_nb_cost;      %记录临时上界
489     current_loc = temp_best_loc(min_ind,:); %临时选址方案
490 else %概率接受不那么好的解
491     if rand < exp((best_nb_cost - current_ub )/temperature)
492         current_ub = best_nb_cost;      %记录临时上界
493         current_loc = nb(i,:);        %临时选址方案
494     end
495 end
496
497 %更新最优解
498 if current_ub < best_nb_ub %获得全局最优解直接返回
499     best_nb_ub = current_ub;          %记录全局最佳上界
500     best_nb_loc = current_loc;        %记录全局最佳选址方案
501 end
502
503 %更新温度
504 temperature = temperature - t_gap;
505 if temperature <= 0
506     temperature = 0.0001;
507 end
508 end
509
510 end
511
512 function ns = get_nb(location)
513 location(1) = []; %先删除虚拟设施
514 opened = find(location == 1);
515 closed = find(location == 0);
516
517 %计算需要邻域的大小
518 ns_sz = length(opened) + length(closed) + length(opened) * length(closed);
519 ns = false(ns_sz,length(location));
520
521 count = 1;
522 %关闭一个设施
523 for i = 1:length(opened)
524     temp = location ;
525     temp(opened(i)) = 0;
526     ns(count,:) = temp;
527     count = count + 1;
528 end
529
530 %开启一个设施

```

```

531 for i = 1:length(closed)
532     temp = location ;
533     temp(closed(i)) = 1;
534     ns(count ,:) = temp;
535     count = count + 1;
536 end
537
538 % 交换设施状态
539 for i = 1:length(opened)
540     for j = 1:length(closed)
541         temp = location ;
542         temp(opened(i)) = 0;
543         temp(closed(j)) = 1;
544         ns(count ,:) = temp;
545         count = count + 1;
546     end
547 end
548
549 % 补充虚拟设施
550 ns = [ true( size(ns,1) ,1) ns ];
551 end
552
553
554 function [mu, cnt_step , alpha] = update_mu(lr_case, lr_para , lb, ub, plan_lb,
555                                              location , cnt_step )
556 %乘子更新
557
558 % 提取
559 I = lr_case .I;
560 bar_J = lr_case .bar_J;
561 mu = lr_case .mu;
562 alpha = lr_para .alpha;
563
564 % 更新系数
565 if cnt_step > lr_para .kappa_lb %长时间未更新下界则步长因子打折
566     alpha = alpha / lr_para .theta_lr ; %步长系数打折
567     cnt_step = 0; %打折后计数器归零
568 end
569
570 temp = zeros(length(I), length(bar_J)); %记录客户i是否使用了设施j
571 % 计算分母
572 for i = 1: size(plan_lb ,1)
573     fac_use = plan_lb(i, 2:end); % 对于i来说使用了的设施
574     fac_use(fac_use==0) = [];
575     temp(i, fac_use) = 1;
576 end
577 violate = temp - location ; % 违背约束
578
579 % 计算迭代步长
580 k_step = lr_para .alpha * (ub-lb) / sum(abs(violate)," all ");
581

```

```

581 % 更新乘子
582 for m = 2:length(bar_J)
583     j = bar_J(m); %j 是设施
584     if location(j)
585         y_j = 1;
586     else
587         y_j = 0;
588     end
589
590     for n = 1:length(I)
591         i = I(n) - bar_J(end); %i 是客户
592         if any(plan_lb(i,2:end) == j)
593             x_ikj = 1;
594         else
595             x_ikj = 0;
596         end
597         mu(i, j) = mu(i, j) + k_step * (x_ikj - y_j);
598         if mu(i,j) < 0
599             mu(i,j) = 0;
600         end
601     end
602 end
603
604
605 function [obj, plan, trans_cost] = ub_xy(lr_case, location, flag_fast)
606 %UB_YX求解模型的上界
607 %OBJ 返回原问题的最优目标函数
608 %plan 返回最优路径方案
609 %trans_cost 返回各路径的运输成本
610
611 %% 初始化
612 % 提取
613 I = lr_case.I;
614 bar_J = lr_case.bar_J;
615 data = lr_case.data;
616 q = lr_case.q;
617 max_try = lr_case.max_try;
618 dmd = data.dmd;
619 price = data.price;
620
621 trans_cost = zeros(length(I), 1); % 记录每个客户的期望运输成本
622 fix_cost = data.fix(location); % 计算固定成本
623 plan = zeros(length(I), max_try+1); % 记录每个客户的方案
624 location(1) = 1; % 虚拟设施指定是已经建设的
625 q_loc = q(location); % 已建设施的损坏概率
626 located = find(location==1); % 已经建设的设施的索引
627
628 price_located = price(location,:); % 建设的节点之间的价格
629 price_cus_fac = price(length(bar_J)+1:end,:); % 顾客和建设的节点之间的距离
630 parfor i = 1:length(I)
631     cus = I(i);

```

```

632 % 初始化客户相关变量
633
634 dmd_cus = dmd(i); % 节点需求
635 price_cus = [ price_located ;
636             price_cus_fac(i,:); % 价格矩阵去掉多余行
637 price_cus = price_cus(:, location); % 价格矩阵去掉多余列
638
639 % dijkstra
640 [pind_without_cus, trans_cost_cus] = mod_dijkstra(price_cus, q_loc, dmd_cus,
641 max_try);
642
643 % 记录
644 trans_cost(i) = trans_cost_cus;
645 temp = [cus, pind_without_cus];
646
647 if length(temp) < max_try+1
648     plan(i,:) = [temp, zeros(1, max_try+1-length(temp))];
649 else
650     plan(i,:) = temp; % 记录方案
651 end
652
653 for i = 1: size(plan,1)
654     pind_with_cus = plan(i,:);
655     pind_with_cus(pind_with_cus==0) = [];
656     temp = [pind_with_cus(1), located(pind_with_cus(2:end))];
657     if length(temp) < max_try+1
658         plan(i,:) = [temp, zeros(1, max_try+1-length(temp))];
659     else
660         plan(i,:) = temp; % 记录方案
661     end
662 end
663
664 obj = sum(trans_cost) + sum(fix_cost); % 目标函数值
665
666 if ~flag_fast
667     % 快速模式不启动dfs
668     [obj, plan, trans_cost] = ub_dfs(I, bar_J, location, plan, trans_cost, data, q,
669     max_try);
670 end
671
672
673
674 function [pind_without_cus, trans_cost] = mod_dijkstra(price, q_loc, dmd_cus, max_try
675 )
676 %MOD_DIJKSTRA 修正的Dijkstra算法
677 % 输入一个i行j列的距离矩阵(从i到j), 返回从最后一个点到第一个点最短成本的路径
678 % 传入的距离矩阵最后一行表示客户, 第一行是虚拟设施
679 start = size(price,1); % 初始起点

```

```

680 fee = 0; % 当前产生的费用
681 preceding_ind = zeros(1, length(q_loc)); % 前向记录
682 prob = ones(1, length(q_loc)+1); % 概率记录
683 unmark = 1:length(q_loc); % 未标记的点
684 weight = inf * ones(1, length(q_loc)); % 点的权重
685
686 count = 0;
687 while count < max_try
688     % 更新未扫描的节点
689     for j = 1:length(unmark)
690         % 上一个点产生的费用 加上 到这个点需要的费用
691         temp = fee + prob( start ) * dmd_cus * price( start , unmark(j));
692
693         if temp < weight(unmark(j)) % 得到新的权重
694             weight(unmark(j)) = temp;
695             preceding_ind(unmark(j)) = start ;
696         end
697     end
698
699 [ fee , mind_unmark] = min(weight(unmark)); % 在所有未标记的权重中选取最小
700 start = unmark(mind_unmark); % 更新起点
701 unmark(unmark==start) = []; % 更新标记
702 prob( start ) = prob(preceding_ind( start )) * q_loc( start ); % 更新概率
703
704 if start == 1
705     break
706 end
707 if isempty(unmark)
708     break
709 end
710 count = count + 1;
711 end
712
713 trans_cost = weight(1);
714 pind_without_cus = get_plan(preceding_ind, max_try, 1, size(price,1));
715
716 end
717
718 function plan_without_cus = get_plan(preceding, times, p_start, p_end)
719 % 返回一个不带客户的路线方案
720 % 给定DIJKSTRA算法的跟踪索引，返回一个路径
721 plan_without_cus = zeros(1, times);
722 ind = p_start ;
723 plan_without_cus(1) = ind;
724
725 count = 2;
726 while 1
727     if preceding(ind) == p_end
728         break
729     end
730

```

```

731 plan_without_cus(count) = preceding(ind);
732 ind = preceding(ind);
733
734 count = count+1;
735 end
736 plan_without_cus = plan_without_cus(end-1:1); % 逆转
737 plan_without_cus(plan_without_cus==0) = []; % 除0
738 end
739
740 function [ lr_ub, plan, trans_cost ] = ub_dfs(I, bar_J, location, plan, trans_cost,
741 data, q, max_try)
742 %UB_DFS 上界DFS搜索
743 pi = data.price(2,1); % 惩罚成本
744 cus_mu = zeros(1,length(bar_J)); % 上界没有乘子
745 dmd = data.dmd;
746 price = data.price;
747 parfor j = 1:length(I)
748     cus = I(j);
749     best_r = plan(j,:); % 当前最优路径
750
751     ub = trans_cost(j); % 当前上界
752     cus_dmd = dmd(j); % 客户的需求
753
754     fac = find(location==1); % 已经建设的设施
755
756     R = coder.ignoreConst(max_try-1);
757     coder.varsize('cus', [1 100], [0 1]);
758     coder.varsize('best_r');
759     coder.varsize('bar_J');
760     coder.varsize('price');
761     coder.varsize('q');
762     coder.varsize('cus_mu');
763
764     [ub, best_r] = lb_dfs(cus, best_r, ub, 0, 1, fac, cus_dmd, price, R, pi, q,
765     cus_mu); % 深度优先搜索
766     trans_cost(j) = ub; % 更新上界
767     plan(j,:) = 0;
768     if length(best_r) < max_try+1
769         plan(j,:) = [best_r, zeros(1, max_try+1-length(best_r))];
770     else
771         plan(j,:) = best_r; % 记录方案
772     end
773 end
774
775 fix_cost = data.fix(location); % 计算固定成本
776 lr_ub = sum(trans_cost) + sum(fix_cost); % 上界
777 end

```

Gurobi 建模源码 (Python)

```

1 import gurobipy as gp
2 import numpy as np
3 import math
4 import copy
5 import os
6 import sys
7
8
9 currentPath = os.getcwd().replace('\\\\','/')
10 print( currentPath )
11
12
13 def previous(i, j, J):
14     before = copy.deepcopy(J)
15     before = [i] + before
16     if j != 0:
17         before.remove(j)
18     return before
19
20
21 def later (i, j, bar_J):
22     after = copy.deepcopy(bar_J)
23     if i != j:
24         after.remove(j)
25     return after
26
27 def get_sol(x, I, bar_J, max_visit_num):
28     plan = np.zeros((len(I), max_visit_num+1))
29     for i in I:
30         plan[i-I[0], 0] = i
31         for k in later (i,i,bar_J):
32             if math.isclose (x[i,i,k].X, 1, rel_tol=0.05):
33                 plan[i-I[0], 1] = k
34                 ind = k
35                 break
36
37         count = 1
38         flag = False
39         while True:
40             for k in later (i, ind, bar_J):
41                 if ind == bar_J[0]:
42                     flag = True
43                     break
44                 if math.isclose (x[i,ind,k].X, 1, rel_tol=0.05):
45                     count = count+1
46                     plan[i-I[0], count] = k
47                     ind = k
48
49             if flag:
50                 break

```

```

51     return plan
52
53 # 导入数据
54 np.set_printoptions(suppress=True)      # 取消numpy打印的科学计数法
55
56
57 # cost 矩阵的第一索引位置是0 默认为虚拟设施
58 root = './ data/SnyderData/49nodes/'
59 cost = np.loadtxt(root+'cost.csv',    # 相对路径下的csv文件
60                   dtype=None,        # 数据类型默认
61                   encoding='UTF-8',  # 注意此文件为UTF-8格式且取消BOM
62                   delimiter=',')    # 分隔符
63
64 dmd = np.loadtxt(root+'dmd.csv',   # 相对路径下的csv文件
65                   dtype=None,        # 数据类型默认
66                   encoding='UTF-8',  # 注意此文件为UTF-8格式且取消BOM
67                   delimiter=',')    # 分隔符
68
69 fc = np.loadtxt(root+'fc.csv',    # 相对路径下的csv文件
70                   dtype=None,        # 数据类型默认
71                   encoding='UTF-8',  # 注意此文件为UTF-8格式且取消BOM
72                   delimiter=',')    # 分隔符
73
74
75 rho = 0.05 # 损坏概率参数
76 max_visit_num = 5 # 客户的最大尝试次数
77
78 q = rho * np.exp(-fc/200000) # 损坏概率
79 q[0] = 1
80
81 cus_num = len(dmd) # 客户数
82 node_num = cost.shape[0] # 节点数（虚拟设施 客户）
83 fac_num = node_num - cus_num - 1 # 设施数
84
85 # 集合设置
86 J = [j for j in range(1, fac_num+1)] # 设施集合
87 I = [i for i in range(fac_num+1, node_num)] # 客户集合
88 bar_J = [j for j in range(0, fac_num+1)] # 设施拓展集合
89 R = [r for r in range(1,max_visit_num)] # 等级
90
91 # 常数集合
92 lmd = {i : dmd[i-fac_num-1] for i in I} # lambda需求
93 c = {(i,j) : cost[i,j] for i in bar_J for j in bar_J} # 价格
94 f = {j : fc[j] for j in J} # 建设成本
95
96 m = gp.Model()
97
98 y = m.addVars((j for j in J), vtype=gp.GRB.BINARY, name='y')
99 x = m.addVars(((i, j, j_p) for i in I for j in J+[i] for j_p in later(i, j, bar_J)), vtype=gp.GRB.BINARY, name='x') # 弧(j,j_p)属于客户i

```

```

100 p = m.addVars(((i, j, j_p) for i in I for j in J+[i] for j_p in later(i,j,bar_J)), lb =
101     0, ub = 1, vtype = gp.GRB.CONTINUOUS,name = 'p') # 属于客户i的弧(j,j_p)的概率
102 w = m.addVars(((i, j, j_p) for i in I for j in J+[i] for j_p in later(i,j,bar_J)), lb =
103     0, ub = 1, vtype = gp.GRB.CONTINUOUS,name = 'w') # 期望价格
104
105 item_1 = gp.quicksum(f[j] * y[j] for j in J)
106 item_2 = gp.quicksum((lmd[i] * c[k,j] * w[i,k,j]) for i in I for j in bar_J for k in
107     previous(i,j,J))
108 m.setObjective(item_1 + item_2)
109
110 m.addConstrs((gp.quicksum(x[i,k,j] for k in previous(i,j,J)) <= y[j] for i in I for j in
111     J, name='assign2open')
112 m.addConstrs((gp.quicksum(x[i,i,j] for j in later(i,i,bar_J)) == 1 for i in I), name =
113     'flowin')
114 m.addConstrs((gp.quicksum(x[i,j,0] for j in previous(i,0,J)) == 1 for i in I), name =
115     'flowout')
116
116 m.addConstrs((gp.quicksum(x[i,j,k] for k in later(i,j,bar_J))
117     == gp.quicksum(x[i,k,j] for k in previous(i,j,J)) for i in I for j in J), name =
118     'balance')
119
120 m.addConstrs((p[i,i,j] == x[i,i,j] for i in I for j in later(i,i,bar_J)), name =
121     'probinit')
122
123 m.addConstrs(((q[j] * gp.quicksum(w[i,k,j] for k in previous(i,j,J))
124     == p[i,j,j_p]) for i in I for j in J for j_p in later(i,j,bar_J)), name =
125     'probbalance')
126
126 m.addConstrs(((gp.quicksum(x[i,j,k] for j in J+[i] for k in later(i,j,bar_J))) <=
127     max_visit_num for i in I), name = 'maxtry')
128
129 m.addConstrs((w[i,j,k] <= p[i,j,k] for i in I for j in J+[i] for k in later(i,j,bar_J)),
130     name='u1b')
131 m.addConstrs((w[i,j,k] <= x[i,j,k] for i in I for j in J+[i] for k in later(i,j,bar_J)),
132     name='u2b')
133 m.addConstrs((w[i,j,k] >= p[i,j,k] + x[i,j,k] -1 for i in I for j in J+[i] for k in later
134     (i,j,bar_J)), name='lb')
135
136
137 # m.computeIIS()
138 # m.write('Model.ilp')

```

```
139 # m.write('Model.lp')
140
141
142 # m.write('Solution.sol')
143
144 print('求解完成')
```

附录 B

附表 1 数值实验结果

Appendix Table 1 Computational results

数据集	ρ	R	上界值	下界值	时长 (s)	gap
49	0.01	2	1135517.37	1124561.86	0.22	0.96%
	0.01	3	966308.71	956750.61	0.14	0.99%
	0.01	4	965071.19	955472.87	0.15	0.99%
	0.01	5	965061.90	955531.65	0.21	0.99%
	0.01	6	965061.83	956135.16	0.24	0.92%
	0.01	7	964525.92	955048.14	0.24	0.98%
	0.01	8	964525.92	954970.74	0.25	0.99%
	0.01	9	965061.83	956252.50	0.29	0.91%
	0.01	10	965436.69	956132.91	0.31	0.96%
	0.05	2	1841659.64	1823442.97	0.32	0.99%
49	0.05	3	1048789.38	1038387.54	0.41	0.99%
	0.05	4	1019247.36	1009373.96	0.29	0.97%
	0.05	5	1018144.58	1009101.44	0.40	0.89%
	0.05	6	1018104.60	1008243.89	0.41	0.97%
	0.05	7	1018103.21	1008268.14	0.44	0.97%
	0.05	8	1018103.21	1008109.60	0.60	0.98%
	0.05	9	1018103.21	1008247.42	0.63	0.97%
	0.05	10	1018103.21	1008065.32	0.65	0.99%
	0.10	2	2602511.64	2585660.00	0.06	0.65%
	0.10	3	1196572.46	1184761.34	0.28	0.99%
49	0.10	4	1084946.72	1074286.41	0.60	0.98%
	0.10	5	1076487.05	1065795.41	0.53	0.99%
	0.10	6	1075870.87	1065324.53	0.68	0.98%
	0.10	7	1075830.41	1065130.61	0.80	0.99%
	0.10	8	1075827.73	1065089.46	1.01	1.00%
	0.10	9	1075827.73	1065093.24	1.01	1.00%
	0.10	10	1075827.73	1065481.76	1.29	0.96%
	0.20	2	4071733.67	4033081.73	0.02	0.95%
	0.20	3	1643187.64	1628596.57	0.28	0.89%
	0.20	4	1256671.79	1244146.88	2.60	1.00%
49	0.20	5	1198200.17	1184365.67	13.27	1.15%
	0.20	6	1189876.86	1175360.95	12.16	1.22%
	0.20	7	1188558.90	1173857.09	11.94	1.24%
	0.20	8	1188384.90	1174055.52	12.22	1.21%
	0.20	9	1188361.85	1173754.72	13.84	1.23%
	0.20	10	1188361.85	1174390.61	14.91	1.18%
	0.30	2	5439070.59	5387362.03	0.02	0.95%
	0.30	3	2211300.38	2198699.26	0.19	0.57%
	0.30	4	1544002.77	1524732.75	26.22	1.25%

(接续附表 1)

(续附表 1)

数据集	ρ	R	上界值	下界值	时长 (s)	gap
49	0.30	5	1362413.82	1338269.29	42.51	1.77%
	0.30	6	1320464.96	1296444.54	49.98	1.82%
	0.30	7	1310922.82	1286668.08	35.42	1.85%
	0.30	8	1308868.83	1284814.26	28.78	1.84%
	0.30	9	1308450.80	1283843.49	33.26	1.88%
	0.30	10	1308368.90	1284260.26	37.09	1.84%
	0.40	2	6758263.04	6697925.28	0.01	0.89%
	0.40	3	2854887.46	2826953.56	0.11	0.98%
	0.40	4	1933012.97	1913706.13	24.98	1.00%
	0.40	5	1622648.73	1571175.89	126.49	3.17%
49	0.40	6	1502927.38	1457148.04	191.09	3.05%
	0.40	7	1472517.63	1423103.26	154.34	3.36%
	0.40	8	1452528.35	1412965.08	121.86	2.72%
	0.40	9	1449549.09	1409394.51	97.60	2.77%
	0.40	10	1451904.13	1408976.37	150.24	2.96%
	0.50	2	8070033.29	7992304.15	0.01	0.96%
	0.50	3	3605678.11	3570134.58	0.13	0.99%
	0.50	4	2371811.16	2355908.15	5.24	0.67%
	0.50	5	1949936.79	1894944.23	345.39	2.82%
	0.50	6	1755114.28	1681887.68	704.90	4.17%
49	0.50	7	1675363.70	1596072.80	875.71	4.73%
	0.50	8	1643806.94	1564328.18	653.71	4.84%
	0.50	9	1620854.17	1552979.28	474.68	4.19%
	0.50	10	1609302.71	1547928.47	354.23	3.81%
	0.01	2	1673394.90	1656674.56	0.73	1.00%
	0.01	3	1364030.87	1350428.05	0.84	1.00%
	0.01	4	1361575.48	1347965.15	0.58	1.00%
	0.01	5	1361557.62	1347972.16	0.76	1.00%
	0.01	6	1361557.48	1348555.69	0.82	0.95%
88	0.01	7	1361557.48	1348856.53	0.99	0.93%
	0.01	8	1361557.48	1349135.09	1.35	0.91%
	0.01	9	1361557.48	1347974.70	1.38	1.00%
	0.01	10	1361557.48	1348016.92	1.86	0.99%
	0.05	2	2799849.13	2777195.62	0.24	0.81%
	0.05	3	1499033.58	1484056.77	2.02	1.00%
	0.05	4	1442229.04	1427813.80	0.89	1.00%
	0.05	5	1440083.62	1425830.33	1.18	0.99%
	0.05	6	1440000.80	1426528.16	1.65	0.94%
	0.05	7	1441512.14	1427336.44	2.61	0.98%
88	0.05	8	1440291.86	1426118.86	2.86	0.98%
	0.05	9	1438257.78	1424266.28	3.9	0.97%
	0.05	10	1441234.69	1426857.46	4.17	1.00%
	0.1	2	3934049.93	3895307.39	0.06	0.98%

(接续附表 1)

(续附表 1)

数据集	ρ	R	上界值	下界值	时长 (s)	gap
88	0.1	3	1751971.02	1734658.00	5.04	0.99%
	0.1	4	1542803.21	1525863.32	15.65	1.10%
	0.1	5	1527361.61	1513243.13	6.67	0.92%
	0.1	6	1527612.80	1512748.74	13.73	0.97%
	0.1	7	1523398.09	1508304.21	6.26	0.99%
	0.1	8	1523389.90	1508197.76	9.57	1.00%
	0.1	9	1523389.29	1505232.83	29.26	1.19%
	0.1	10	1523389.25	1508223.17	13.09	1.00%
	0.2	2	6005078.65	5956442.00	0.03	0.81%
	0.2	3	2409126.67	2385045.21	2.21	1.00%
88	0.2	4	1844357.46	1796160.25	80.47	2.61%
	0.2	5	1721135.80	1674692.24	98.52	2.70%
	0.2	6	1699944.26	1654016.26	96.03	2.70%
	0.2	7	1696749.37	1652673.49	84.02	2.60%
	0.2	8	1696325.16	1651132.02	79.95	2.66%
	0.2	9	1696262.13	1652565.32	92.02	2.58%
	0.2	10	1696253.54	1649649.61	103.35	2.75%
	0.3	2	7865967.01	7815012.06	0.03	0.65%
	0.3	3	3144224.79	3113096.17	3.46	0.99%
	0.3	4	2241490.44	2192516.16	181.49	2.18%
88	0.3	5	1986701.61	1902873.69	297.82	4.22%
	0.3	6	1903030.98	1822750.61	341.46	4.22%
	0.3	7	1885198.80	1802537.60	286.47	4.38%
	0.3	8	1879179.38	1794012.12	203.22	4.53%
	0.3	9	1882784.33	1794525.95	191.90	4.69%
	0.3	10	1882784.33	1794525.95	191.90	4.69%
	0.4	2	9376632.49	9297798.62	0.01	0.84%
	0.4	3	3910424.64	3891209.00	0.95	0.49%
	0.4	4	2697337.04	2667818.22	333.89	1.09%
	0.4	5	2321884.16	2224766.91	1001.43	4.18%
88	0.4	6	2159103.30	2046769.35	1005.02	5.20%
	0.4	7	2133113.90	1969316.33	1000.63	7.68%
	0.4	8	2081157.76	1980071.95	1000.79	4.86%
	0.4	9	2064194.68	1958792.65	1000.29	5.11%
	0.4	10	2045614.42	1945806.67	1000.37	4.88%
	0.5	2	10796071.02	10757998.28	0.01	0.35%
	0.5	3	4758985.94	4719571.15	1.02	0.83%
	0.5	4	3253177.21	3224507.59	179.69	0.88%
	0.5	5	2708226.56	2619231.35	1000.52	3.29%
	0.5	6	2474703.53	2332739.09	1000.32	5.74%
88	0.5	7	2447684.34	2178110.08	1000.76	11.01%
	0.5	8	2477928.92	2123260.30	1001.36	14.31%
	0.5	9	2318882.18	2109306.80	1002.29	9.04%

(接续附表 1)

(续附表 1)

数据集	ρ	R	上界值	下界值	时长 (s)	gap
			0.5	10	2229697.00	2113633.56
150	0.01	2	2544807.53	2521837.58	1.21	0.90%
	0.01	3	2178652.92	2156895.94	3.05	1.00%
	0.01	4	2176505.14	2154764.59	3.98	1.00%
	0.01	5	2176492.30	2154727.90	13.63	1.00%
	0.01	6	2176492.22	2154730.38	13.00	1.00%
	0.01	7	2178258.68	2157312.95	14.20	0.96%
	0.01	8	2176492.22	2153289.75	30.90	1.07%
	0.01	9	2176492.22	2154127.28	32.58	1.03%
	0.01	10	2176492.22	2145085.81	49.18	1.44%
	0.05	2	3874369.09	3844889.81	0.61	0.76%
150	0.05	3	2333513.38	2300024.43	15.24	1.44%
	0.05	4	2281044.78	2244820.48	16.90	1.59%
	0.05	5	2279427.84	2247701.89	19.26	1.39%
	0.05	6	2279375.95	2239137.69	36.50	1.77%
	0.05	7	2279374.46	2241391.16	33.14	1.67%
	0.05	8	2279374.42	2231988.57	66.57	2.08%
	0.05	9	2279374.42	2235367.94	62.36	1.93%
	0.05	10	2279374.42	2228257.34	96.36	2.24%
	0.10	2	4670884.11	4624345.77	1.34	1.00%
	0.10	3	2591989.01	2544668.04	31.59	1.83%
150	0.10	4	2402425.55	2352136.53	91.73	2.09%
	0.10	5	2391069.73	2344868.48	46.93	1.93%
	0.10	6	2390372.58	2337730.26	101.45	2.20%
	0.10	7	2390332.05	2342281.99	112.36	2.01%
	0.10	8	2390330.04	2325003.15	129.39	2.73%
	0.10	9	2390329.90	2329218.84	125.01	2.56%
	0.10	10	2390329.90	2326724.82	135.11	2.66%
	0.20	2	5104457.39	5053451.42	1.43	1.00%
	0.20	3	3258304.19	3189773.05	97.70	2.10%
	0.20	4	2696035.59	2622663.80	145.61	2.72%
150	0.20	5	2611120.81	2524599.01	184.14	3.31%
	0.20	6	2600405.99	2516612.88	174.74	3.22%
	0.20	7	2600027.49	2502794.94	177.08	3.74%
	0.20	8	2599035.77	2512958.71	185.29	3.31%
	0.20	9	2599022.19	2510626.78	205.28	3.40%
	0.20	10	2599020.11	2496214.00	217.33	3.96%
	0.30	2	5449215.89	5379119.35	7.55	1.29%
	0.30	3	3760072.43	3704380.51	91.71	1.48%
	0.30	4	3096900.63	2970440.23	378.52	4.08%
	0.30	5	2870208.46	2760987.34	472.86	3.81%
150	0.30	6	2834877.35	2714850.08	389.11	4.23%
	0.30	7	2821476.11	2700787.31	384.20	4.28%

(接续附表 1)

(续附表 1)

数据集	ρ	R	上界值	下界值	时长 (s)	gap
150	0.30	8	2819638.79	2695397.14	407.46	4.41%
	0.30	9	2819333.71	2678220.25	462.89	5.01%
	0.30	10	2819281.64	2677995.29	539.67	5.01%
	0.40	2	5699684.69	5640420.76	7.66	1.04%
	0.40	3	4093710.69	4057739.55	17.81	0.88%
	0.40	4	3585032.60	3373126.27	794.69	5.91%
	0.40	5	3194788.49	3027146.65	1000.31	5.25%
	0.40	6	3087747.84	2918073.69	1000.18	5.50%
	0.40	7	3035702.53	2880124.73	1000.56	5.12%
	0.40	8	3029697.66	2875142.85	1000.10	5.10%
150	0.40	9	3020351.27	2877893.15	1000.01	4.72%
	0.40	10	3024438.35	2866995.57	1000.72	5.21%
	0.50	2	5922099.59	5862883.07	2.94	1.00%
	0.50	3	4368153.33	4325108.99	11.76	0.99%
	0.50	4	3915245.86	3692113.09	1000.50	5.70%
	0.50	5	3633533.29	3313987.06	1000.46	8.79%
	0.50	6	3405115.82	3131086.20	1001.27	8.05%
	0.50	7	3317441.28	3063547.65	1001.49	7.65%
	0.50	8	3274026.65	3057234.64	1000.78	6.62%
	0.50	9	3308522.57	3045479.93	1000.75	7.95%
	0.50	10	3282775.65	3037413.15	1000.01	7.47%

附表 2 下界求解性能
Appendix Table 2 Performance on solving lower bound

乘子 取值	ρ	R	HEU-DFS		HEU			GRB	
			上界值	时间 (s)	上界值	时间 (s)	gap*	上界值	时间 (s)
0.1	0.1	2	1633944.99	0.001	1670376.12	0.003	2.23%	1633944.99	22.540
		3	141464.04	0.001	142770.19	0.001	0.92%	141464.04	34.160
		4	41267.75	0.001	41451.49	0.000	0.45%	41267.75	63.760
		5	34443.24	0.001	34456.31	0.000	0.04%	34443.24	70.980
		6	33965.45	0.001	33973.26	0.001	0.02%	33960.48	85.820
	0.2	7	33931.63	0.002	33938.50	0.001	0.02%	33928.38	88.460
		8	33929.30	0.005	33936.05	0.001	0.02%	33928.26	121.570
		9	33929.14	0.013	33935.89	0.001	0.02%	33928.21	141.110
		10	33929.13	0.057	33935.88	0.001	0.02%	33928.28	138.940
		2	3200666.60	0.001	3340752.24	0.000	4.38%	3200666.60	20.050
全零	0.3	3	492574.63	0.001	508533.33	0.000	3.24%	492574.63	51.610
		4	130824.03	0.003	134009.88	0.000	2.44%	130824.03	191.340
		5	81884.59	0.003	82319.67	0.000	0.53%	81884.59	214.070
		6	75049.11	0.003	75204.92	0.001	0.21%	75049.09	238.290
		7	74091.10	0.004	74178.41	0.001	0.12%	74091.03	288.120
	0.4	8	73959.27	0.008	74033.23	0.001	0.10%	73943.76	248.100
		9	73941.16	0.020	74013.56	0.001	0.10%	73936.64	278.980
		10	73938.68	0.068	74010.90	0.001	0.10%	73936.47	311.060
		2	4694376.52	0.001	5011128.36	0.000	6.75%	4694376.52	19.470
		3	1029633.96	0.003	1097289.43	0.000	6.57%	1029633.96	268.060
0.5	0.5	4	305850.62	0.005	323802.24	0.000	5.87%	305850.62	253.100
		5	159508.53	0.008	163763.07	0.000	2.67%	159508.53	551.460
		6	129433.27	0.008	130852.86	0.001	1.10%	129433.27	548.520
		7	123189.92	0.010	123709.38	0.001	0.42%	123189.92	461.720
		8	121869.57	0.016	122188.68	0.001	0.26%	121869.62	481.730
	0.6	9	121599.96	0.030	121879.79	0.001	0.23%	121595.17	453.270
		10	121544.86	0.089	121817.20	0.001	0.22%	121528.77	454.860
		2	1723122.41	0.001	1754270.70	0.000	1.81%	1723122.41	21.500
		3	298683.51	0.001	301568.78	0.000	0.97%	298683.51	25.590
		4	234311.50	0.002	235004.85	0.000	0.30%	234311.50	25.610
0.7	0.7	5	234311.50	0.002	235004.85	0.000	0.30%	234311.50	24.860
		6	234311.50	0.002	235004.85	0.001	0.30%	234311.50	24.910
		7	234311.50	0.002	235004.85	0.000	0.30%	234311.50	24.580
		8	234311.50	0.002	235004.85	0.000	0.30%	234311.50	24.400
		9	234311.50	0.002	235004.85	0.000	0.30%	234311.50	24.800
	0.8	10	234311.50	0.002	235004.85	0.000	0.30%	234311.50	24.440

(接续附表 2)

(续附表 2)

乘子 取值	ρ	R	HEU-DFS		HEU			GRB	
			上界值	时间	上界值	时间(s)	gap*	上界值	时间(s)
初始	0.2	2	3297990.49	0.001	3427611.75	0.000	3.93%	3297990.49	20.210
		3	666412.48	0.001	693534.12	0.000	4.07%	666412.48	34.910
		4	356701.30	0.004	360987.90	0.000	1.20%	356701.30	210.220
		5	334265.11	0.008	338760.76	0.000	1.34%	334265.11	143.990
		6	334265.11	0.008	338760.76	0.000	1.34%	334265.11	139.020
		7	334265.11	0.010	338760.76	0.001	1.34%	334265.11	134.830
		8	334265.11	0.009	338760.76	0.001	1.34%	334265.11	135.350
		9	334265.11	0.009	338760.76	0.000	1.34%	334265.11	135.120
		10	334265.11	0.009	338760.76	0.000	1.34%	334265.11	132.770
		2	4800015.31	0.001	5100952.80	0.000	6.27%	4800015.31	19.410
0.3	0.3	3	1220766.16	0.004	1320667.45	0.000	8.18%	1220766.16	324.400
		4	557306.00	0.008	579407.23	0.000	3.97%	557306.00	381.170
		5	446252.57	0.018	453822.51	0.000	1.70%	446252.57	386.480
		6	437074.61	0.033	444558.57	0.001	1.71%	437074.61	383.050
		7	437074.61	0.034	444966.36	0.001	1.81%	437074.61	404.630
		8	437074.61	0.035	444966.36	0.001	1.81%	437074.61	428.760
		9	437074.61	0.034	444966.36	0.001	1.81%	437074.61	413.180
		10	437074.61	0.036	444966.36	0.001	1.81%	437074.61	401.340
		2	3108786.52	0.000	3133465.73	0.000	0.79%	3108786.52	21.330
		3	1962014.55	0.001	1964487.80	0.000	0.13%	1962014.55	14.220
0.1	0.1	4	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.570
		5	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.330
		6	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.710
		7	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.870
		8	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.330
		9	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.410
		10	1947138.07	0.001	1949611.32	0.000	0.13%	1947138.07	14.750
		2	4643757.79	0.000	4748735.73	0.000	2.26%	4643757.79	20.230
		3	2444464.63	0.001	2454110.49	0.000	0.39%	2444464.63	14.710
		4	2294633.28	0.001	2318998.23	0.000	1.06%	2294633.28	14.570
随机	0.2	5	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	14.800
		6	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	15.190
		7	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	14.420
		8	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	14.990
		9	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	14.760
		10	2291633.33	0.001	2314385.20	0.000	0.99%	2291633.33	14.870
		2	6127605.57	0.001	6364005.73	0.000	3.86%	6127605.57	19.400

(接续附表 2)

(续附表 2)

乘子 取值	ρ	R	HEU-DFS		HEU			GRB	
			上界值	时间	上界值	时间 (s)	gap*	上界值	时间 (s)
0.3	3	3089066.16	0.001	3139701.37	0.000	1.64%	3089066.16	23.630	
	4	2648117.83	0.001	2696479.94	0.000	1.83%	2648117.83	23.030	
	5	2622185.75	0.002	2676739.86	0.000	2.08%	2622185.75	23.140	
	6	2622185.75	0.002	2676739.86	0.001	2.08%	2622185.75	23.050	
	7	2622185.75	0.002	2676739.86	0.001	2.08%	2622185.75	23.030	
	8	2622185.75	0.002	2676739.86	0.000	2.08%	2622185.75	23.130	
	9	2622185.75	0.002	2676739.86	0.001	2.08%	2622185.75	23.020	
	10	2622185.75	0.002	2676739.86	0.000	2.08%	2622185.75	23.220	

附表 3 上界求解性能
Appendix Table 3 Performance on solving upper bound

$ J^* $	ρ	R	DIJK-DFS		DIJK			GRB	
			上界值	时间 (s)	上界值	时间 (s)	gap*	上界值	时间 (s)
2	0.1	2	3012246.66	0.001	3014368.04	0.044	0.07%	3012246.66	2.220
		3	1530528.73	0.001	1759968.32	0.001	14.99%	1530528.73	5.360
		4	1429483.20	0.001	1452475.48	0.001	1.61%	1429483.20	4.850
		5	1422385.37	0.001	1424496.67	0.001	0.15%	1422385.37	5.110
		6	1421956.49	0.001	1422981.45	0.001	0.07%	1421962.06	5.510
		7	1421924.37	0.001	1422696.62	0.001	0.05%	1421927.84	5.740
		8	1421922.23	0.001	1422673.48	0.001	0.05%	1421927.84	5.700
		9	1421922.08	0.001	1422671.94	0.001	0.05%	1421927.84	5.770
		10	1421922.07	0.001	1422671.69	0.001	0.05%	1421927.84	5.740
		2	4680993.49	0.001	4688272.98	0.001	0.16%	4680993.49	2.120
3	0.2	3	1946579.67	0.001	2547957.00	0.001	30.89%	1946579.67	4.290
		4	1578457.03	0.001	1712022.64	0.001	8.46%	1578457.03	5.660
		5	1525765.22	0.001	1551073.90	0.001	1.66%	1525765.22	5.500
		6	1519486.69	0.001	1525966.49	0.001	0.43%	1519496.93	6.340
		7	1518541.36	0.001	1521053.90	0.001	0.17%	1518553.11	6.830
		8	1518416.06	0.001	1520337.86	0.001	0.13%	1518389.21	7.070
		9	1518397.63	0.001	1520240.16	0.001	0.12%	1518388.24	7.100
		10	1518395.28	0.001	1520219.50	0.001	0.12%	1518388.98	7.320
		2	6316980.28	0.001	6362177.93	0.001	0.72%	6316980.28	2.290
		3	2585602.02	0.001	3548432.76	0.001	37.24%	2585602.02	5.110
4	0.3	4	1831720.83	0.001	2232271.92	0.001	21.87%	1831720.83	5.840
		5	1671213.78	0.001	1834075.97	0.001	9.75%	1671213.78	5.830
		6	1642616.91	0.001	1687393.89	0.001	2.73%	1642616.91	6.860
		7	1636041.81	0.001	1652511.38	0.001	1.01%	1636041.59	7.290
		8	1634732.26	0.001	1646061.74	0.001	0.69%	1634738.06	7.620
		9	1634443.55	0.001	1644048.22	0.001	0.59%	1634443.54	8.010
		10	1634389.85	0.001	1643659.73	0.001	0.57%	1634371.50	8.210
		2	3433597.35	0.001	3461440.13	0.001	0.81%	3433597.35	4.020
		3	1989538.64	0.001	1993724.84	0.001	0.21%	1989538.64	8.330
		4	1889570.80	0.001	1906337.44	0.001	0.89%	1889570.80	10.150
5	0.1	5	1882630.96	0.001	1884238.17	0.001	0.09%	1882630.96	11.740
		6	1882170.96	0.001	1882267.73	0.001	0.01%	1882167.30	12.930
		7	1882137.79	0.001	1882144.81	0.001	0.00%	1882144.04	12.620
		8	1882135.58	0.001	1882136.20	0.001	0.00%	1882146.84	12.940
		9	1882135.44	0.001	1882135.52	0.001	0.00%	1882144.04	12.980
		10	1882135.43	0.001	1882135.48	0.001	0.00%	1882144.04	13.280

(接续附表 3)

(续附表 3)

$ J^* $	ρ	R	DIJK-DFS		DIJK			GRB	
			上界值	时间 (s)	上界值	时间 (s)	gap*	上界值	时间 (s)
20	0.2	2	4985136.54	0.001	5094165.57	0.001	2.19%	4985136.54	4.240
		3	2355514.33	0.001	2444003.06	0.001	3.76%	2355514.33	8.410
		4	2000839.16	0.001	2101479.43	0.001	5.03%	2000839.16	10.160
		5	1952091.00	0.001	1972412.17	0.001	1.04%	1952091.00	11.100
		6	1945406.13	0.001	1949177.49	0.001	0.19%	1945413.53	13.310
		7	1944483.43	0.001	1946063.19	0.001	0.08%	1944491.31	14.280
		8	1944357.26	0.002	1945659.43	0.001	0.07%	1944348.92	13.970
		9	1944340.44	0.004	1945569.22	0.001	0.06%	1944343.84	14.260
		10	1944338.20	0.009	1945557.25	0.001	0.06%	1944338.77	14.640
		2	6479779.20	0.001	6726891.01	0.001	3.81%	6479779.20	3.890
0.3	0.3	3	2915432.67	0.001	3157915.61	0.001	8.32%	2915432.67	10.030
		4	2203573.62	0.001	2511244.87	0.001	13.96%	2203573.62	15.020
		5	2055954.65	0.001	2168868.42	0.001	5.49%	2055954.65	14.630
		6	2025553.43	0.001	2083844.92	0.001	2.88%	2025553.43	15.430
		7	2019479.29	0.001	2032890.46	0.001	0.66%	2019487.13	16.970
		8	2018178.88	0.002	2029254.48	0.001	0.55%	2018187.58	17.470
		9	2017920.80	0.004	2022587.06	0.001	0.23%	2017904.60	19.830
		10	2017869.24	0.007	2021140.73	0.001	0.16%	2017850.79	18.740
		2	4130314.17	0.001	4193400.81	0.001	1.53%	4130314.17	7.640
		3	2672873.84	0.001	2681180.02	0.001	0.31%	2672873.84	14.540
0.1	0.1	4	2573618.61	0.001	2577219.65	0.001	0.14%	2573603.04	18.040
		5	2566919.96	0.001	2567435.47	0.001	0.02%	2566907.27	21.180
		6	2566440.07	0.001	2566514.22	0.001	0.00%	2566419.67	22.910
		7	2566407.20	0.001	2566423.90	0.001	0.00%	2566391.06	21.850
		8	2566404.89	0.002	2566416.30	0.001	0.00%	2566391.06	22.680
		9	2566404.72	0.004	2566415.68	0.001	0.00%	2566391.06	22.340
		10	2566404.71	0.016	2566415.57	0.001	0.00%	2566391.06	21.960
		2	5665616.27	0.001	5866269.11	0.001	3.54%	5665616.27	7.430
		3	3035341.56	0.001	3359616.11	0.001	10.68%	3035341.56	15.200
		4	2676440.07	0.001	2855391.19	0.001	6.69%	2676440.07	21.580
30	0.2	5	2627953.26	0.001	2663893.10	0.001	1.37%	2627927.78	24.340
		6	2621219.54	0.002	2636688.29	0.001	0.59%	2621208.11	29.120
		7	2620286.27	0.003	2629745.91	0.001	0.36%	2620276.85	32.890
		8	2620154.31	0.005	2623295.82	0.001	0.12%	2620117.87	32.960
		9	2620135.35	0.013	2622408.95	0.001	0.09%	2620117.87	33.320
		10	2620132.67	0.029	2621870.67	0.001	0.07%	2620117.87	33.260
		2	7138310.69	0.001	7539137.40	0.001	5.62%	7138310.69	7.270

(接续附表 3)

(续附表 3)

$ J^* $	ρ	R	DIJK-DFS		DIJK			GRB	
			上界值	时间(s)	上界值	时间(s)	gap*	上界值	时间(s)
0.3	0.3	3	3593063.07	0.001	4136931.33	0.001	15.14%	3593063.07	20.000
		4	2868351.60	0.002	3330131.96	0.001	16.10%	2868351.6	29.790
		5	2721043.28	0.003	2869801.27	0.001	5.47%	2721043.28	44.460
		6	2691218.14	0.004	2748234.41	0.001	2.12%	2691192.76	44.270
		7	2685009.59	0.005	2717092.59	0.001	1.19%	2684992.32	48.720
		8	2683695.78	0.006	2694338.66	0.001	0.40%	2683695.78	79.550
		9	2683413.89	0.010	2689324.20	0.001	0.22%	2683398.51	82.520
		10	2683354.48	0.026	2688113.60	0.001	0.18%	2683313.12	57.400
		2	4714111.30	0.001	4771681.94	0.001	1.22%	4714111.30	13.260
		3	3240363.56	0.001	3249096.75	0.001	0.27%	3240363.56	20.690
0.1	0.1	4	3142850.48	0.001	3147452.07	0.001	0.15%	3142850.48	27.760
		5	3136163.42	0.001	3140377.19	0.001	0.13%	3136181.22	36.500
		6	3135710.10	0.001	3136071.16	0.001	0.01%	3135702.79	69.450
		7	3135678.97	0.001	3135711.51	0.001	0.00%	3135675.16	60.810
		8	3135676.91	0.002	3135685.68	0.001	0.00%	3135675.14	69.350
		9	3135676.77	0.006	3135681.20	0.001	0.00%	3135675.22	71.960
		10	3135676.76	0.018	3135680.44	0.001	0.00%	3135675.16	70.040
		2	6277780.12	0.001	6444693.93	0.001	2.66%	6277780.12	13.340
		3	3589432.99	0.001	3802986.76	0.001	5.95%	3589432.99	22.650
		4	3234297.42	0.001	3365449.67	0.001	4.06%	3234297.42	66.950
40	0.2	5	3186336.46	0.002	3228211.24	0.001	1.31%	3186336.46	72.890
		6	3179818.58	0.003	3200729.13	0.001	0.66%	3179821.39	103.590
		7	3178943.94	0.003	3183790.23	0.001	0.15%	3178943.86	102.400
		8	3178824.65	0.007	3181344.08	0.001	0.08%	3178810.17	101.360
		9	3178808.58	0.008	3179422.57	0.001	0.02%	3178811.46	131.820
		10	3178806.30	0.025	3179114.53	0.001	0.01%	3178805.75	145.670
		2	7765849.13	0.001	8117705.93	0.001	4.53%	7765849.13	12.700
		3	4123701.63	0.001	4582631.86	0.001	11.13%	4123701.63	36.460
		4	3409967.52	0.003	3782957.28	0.001	10.94%	3409967.52	144.740
		5	3266011.16	0.004	3443964.45	0.001	5.45%	3266011.16	202.240
0.3	0.3	6	3237015.81	0.009	3339599.19	0.001	3.17%	3237015.81	187.980
		7	3231194.82	0.017	3291825.61	0.001	1.88%	3231194.82	170.260
		8	3230026.51	0.021	3252392.28	0.001	0.69%	3230018.85	243.760
		9	3229790.47	0.030	3238230.27	0.001	0.26%	3229780.56	202.880
		10	3229742.82	0.071	3234940.44	0.001	0.16%	3229729.27	203.700

(本页留空)

作者简历及攻读硕士学位期间取得的研究成果

一、作者简历

2021 年，毕业于北京交通大学，交通运输学院，物流工程专业，获工学学士学位；
2021 年至 2023 年，就读于北京交通大学，交通运输学院，交通运输专业，导师员丽芬副教授；

二、发表论文

- [1] **RUNFENG YU, LIFEN YUN, CHEN CHEN, YUANJIE TANG, HONGQIANG FAN, YI QIN**, Vehicle Routing Optimization for Vaccine Distribution Considering Reducing Energy Consumption [J]. Sustainability, 2023, 15(2): 1252.(SCI)
- [2] **RUNFENG YU, LIFEN YUN, HONGQINAG FAN, YANXI LIU, MINYU JIN**. Optimization of Vehicle Routing Problem for Vaccine Distribution [C], Washington DC, United States: Transportation Research Board 101st Annual Meeting, 2022.(会议)
- [3] **RUNFENG YU, LIFEN YUN, HONGQINAG FAN, YANXI LIU, MINYU JIN**. Optimization of Vehicle Routing Problem for Vaccine Distribution [C], 武汉: 世界交通大会, 2022.(会议)
- [4] 员丽芬, 余润峰, 范宏强, 张蛰. 不完美信息下考虑设施独立损坏概率的固定费用选址模型及算法 [C], 中国物流学术年会论文, 2022.(会议)
- [5] **YANXI LIU, LIFEN YUN, HONGQINAG FAN, RUNFENG YU, MINYU JIN**. Location-Routing Problem of Pharmaceutical Facilities with Soft Time Window [C], Washington DC, United States: Transportation Research Board 101st Annual Meeting, 2022.(会议)

三、专利

- [1] 员丽芬, 张钰儒, 余润峰. 仓店一体模式下前置仓选址与路径联合优化方法: CN202210529946 [P]. 2022.08.23.

四、获奖情况

- (1) 一等奖学金, 北京交通大学, 2022
- (2) 神州控股校园极客大赛(全国第一名), 神州控股, 2022
- (3) 日日顺创客训练营(银奖), 青岛日日顺, 2022

(本页留空)

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的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。

学位论文作者签名： 签字日期： 年 月 日

(本页留空)

学位论文数据集

表 1.1 数据集页

关键词 *	密级 *	中图分类号	UDC	论文资助
物流节点选址; 选址问题; 拉格朗日松弛; 迭代局部搜索	公开			
学位授予单位名称 *		学位授予单位代码 *	学位类别 *	学位级别 *
北京交通大学		10004	交通运输专业学位	硕士
论文题名 *		并列题名 *		论文语种 *
基于迭代局部搜索改进的拉格朗日松弛算法设计		无		中文
作者姓名 *	余润峰		学号 *	21125806
培养单位名称 *		培养单位代码 *	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西直门外上园村 3 号	100044
专业学位类别 (领域) *		研究方向 *	学制 *	学位授予年 *
交通运输		运筹优化	2 年	2023
论文提交日期 *	2023 年 6 月			
导师姓名 *	员丽芬		职称 *	副教授
评阅人	答辩委员会主席 *		答辩委员会成员	
	姜超峰			
电子版论文提交格式 文本 0 图像 0 视频 0 音频 0 多媒体 0 其他 0 推荐格式: application/msword; application/pdf				
电子版论文出版 (发布) 者		电子版论文出版 (发布) 地		权限声明
论文总页数 *	105			
共 33 项, 其中带 * 为必填数据, 为 21 项。				