武汉理工大学毕业设计（论文）

利用智能混合算法解决大规模路径优化问题

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：

年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密□，在　　年解密后适用本授权书；

2、不保密□ 。

（请在以上相应方框内打“√”）

作者签名：　　年　　月　　日

导师签名：　　年　　月　　日

摘　要

运输成本在企业、行业乃至社会的总物流成本中都占最高比例，达 50%以上。数理决策作为一种通过对车辆行驶路径进行合理规划从而有效降低运输成本的方法，长期以来是相关企业进行物流管理的重要手段之一。同时，与运输决策相关的问题往往需要在满足某些特定条件下对有限资源进行优化配置，问题的本质结构有很高的理论研究价值。其中最具代表性的车辆路径优化问题(Vehicle Routing Problem,VRP)一直是相关科学领域的研究热点。除运输外，该问题在互联网领域中路由寻址的优化，计算机领域中数据的分布式存储结构的设计等任何涉及到网络的领域都有着重要应用。

随着经济全球化以及物流行业服务质量的不断提高，实际应用中的车辆路径问题也变得越来越复杂，比如全球最大的快递公司 FedEx 每天就有超过 650 万件的快递运往世界 200 多个不同的国家和地区。对于这种大规模的路径优化问题，是否具备操作性强的高性能优化方法往往成为企业盈利的关键。

基于以上动机，论文对近 40 年来车辆路径问题的经典文献进行了详细综述，尤其对最近 10 年出现的新的技术和方法进行了深入的总结和对比；在此基础上设计开发了一种用于解决带资源约束 VRP 问题的智能混合算法；算法以局部搜索为基础，结合了模拟退火和禁忌搜索的思想。相比于单一的方法，该混合算法在求解时间和解的质量方面均占有优势。为了验证算法的有效性，本文从权威的 VRP 基准测试库 TSPLIB 中选取了三种基准测试包进行了严密的计算与对比实验。研究结果表明，智能混合搜索算法既解决了模拟退火法搜索效率低下的问题，又改善了禁忌搜索易于陷入局优的缺陷。尤其在大规模问题中，该算法优势较为明显。

关键词：路径优化问题；局部搜索；模拟退火法；禁忌搜索；混合算法

Abstract

Transportation cost possesses the highest proportion of 50% in enterprise, profession and even the social total logistic cost. Mathematical decision which can be used to decrease the transportation cost effectively through making logical planning for vehicle path has always been one of the important ways for related companies to implement logistic management. At the meantime the problem concerned with transportation decision have been required to make optimized configuration for limited resources under some particular conditions. The essential structure of the problem is worthy of being researched in theoretical field. Of which the most characteristic one is the Vehicle Routing Problem, the research hotspot in related scientific field. Except for transportation, the VRP has also been applied in many other fields which are concerned with network such as the allocation of goods in logistics; the addressing of router; the design of distributed databases storage structures in computer science and so on.

With the globalization of the world economy and the continuous improvement of the quality of service in logistic, the VRP in real application is also becoming increasingly complex. For instance, FedEx, the largest express company in the world, has to distribute more than 6,500,000 goods to more than 200 different countries and regions in the world. Facing this kind of large-scale VRP, the key point of the enterprise's profit is whether or not the optimizing method has high maneuverability and capacity.

Base on the above motivation, the paper makes a detail generalization of the recent 40 years classical references about VRP, especially summarizes and makes a comparison in depth for some advances and new technologies in a decade. On the basis of that a hybrid intelligent algorithm which aim at solving the VRP with capacity constrain is developed. The algorithm based on local search conjoins the simulated annealing with tabu-search. Compared with the single algorithm, the hybrid one has the advantage of on two aspects: the solving time and the quality of the solution. In order to validate the effectiveness of the hybrid, the paper takes 3 kinds benchmark form the TSPLIB which is a well-known VRP benchmark library to implement the rigorous computation and comparative experiments. The result shows that the hybrid intelligent algorithm not only improve the efficiency but also avoid the deficiency of tending to stuck into the local minima.

Key words: Vehicle Routing Problem, local search, simulated annealing, tabu search, hybrid algorithm

目 录

第 1 章 绪论

1.1 研究背景与意义

车辆路径问题 VRP（Vehicle Routing Problem）是经典的组合优化问题，几乎它的所有变形在一般意义上都是 NP-hard。该问题最基本形式可描述如下：设有一配送中心，共有 M 辆规格一致的货车，车辆容量为 C，有 N 位顾客，每位顾客有其需求量 D。车辆从配送中心出发对客户进行配送服务最后返回场站，要求所有顾客都被配送，每位顾客一次配送完成，且不能违反车辆容量的限制，目的是使所有车辆行驶路线的总距离最小。

车辆路径问题是许多领域的热点研究问题，一方面是因为其拥有广泛的应用基础。现实生产和生活中，飞机航班安排、铁路车辆编组、邮政投递问题、码头调运、货车车队规划、水运船舶调运、燃油运输、公共汽车调度问题以及电力调度问题等都可以抽象为 VRP 问题来进行解决。比如在交通运输系统中客货运输领域。交通运输系统作为国民经济发展的支柱，其运输费用已经在国民经济支出中占有重要比重。有研究表明，在欧洲，货品的年运输周转费用大约为 1680 亿美元[2]。英国、法国、丹麦三国在交通运输费用上面约占国民经济支出比例为 15%，9%，15%[3][4]。根据我国第三产业普查资料，我国在物流行业的成本费用之和约占国民生产总值的 15%。而近几年，随着互联网的重新崛起，各个行业都面临着新的机遇与挑战。物流行业尤甚。在网络化的影响下，交通运输系统、物流配送系统，快递收发系统等行业在实时性与高效性方面开始面临更严苛挑战。物流配送路径优化，即车辆路径问题，是当今物流配送优化中关键的一环，也是电子商务活动不可缺少的内容。运输路线是否合理直接影响到配送速度、成本和效益。选取恰当的车辆路径，不仅可以加快对客户需求的响应速度，提高服务质量，还可以增强客户对物流环节的满意度，降低服务商运作成本。在国内，以阿里、京东为代表的电商企业都开始了新一轮“圈地”运动，意图建立自己的物流配送中心。以阿里为例，其最近的对海尔旗下日日顺物流进行了高达 28.22 亿港元的投资。而京东在物流方面的费用也逐年递增，2009 年到 2012 年，京东物流费用率依次为 4.9%、5.6%、7.2%、7.4%；2013 年前三季度，该占比降为 5.8%。而横向对比，苏宁的该项费用率为 5.32%，亚马逊的该项费用率也为 11.9%。因此，随着传统销售渠道的改变以及新的销售渠道的崛起，作为物流运作的核心，车辆路径问题（VRP）在实际应用领域的研究将变得非常重要。在降低成本之外，优化运输路线，在缓解交通阻塞、减少交通事故和运输工具尾气污染等方面页具有重要的社会意义。

VRP 问题受到越来越多的学者和研究人员的重视的另一个重要原因是其求解的计算复杂性。由于 VRP 是一个典型的 NP 难题，随着客户访问点的增加其解的规模也呈指数增长。分支定界法、动态规划等精确算法在客户访问点超过 150 个后会出现指数爆炸的问题。而在实际应用过程中，随着物流行业快速的发展，大规模的客户访问点的存在几乎不可避免，在有限时间内求得满意解的启发式算法是求解大规模车辆路径问题的重要途径。在对 VRP 问题近 40 年的研究中，产生了许多有代表性的启发式算法，如基于搜索思想的模拟退火法

（Simulated Annealing)、禁忌搜索法(Tabu Search)、领域搜索法(Neighborhood Search）以及基于仿生思想的遗传算法（genetic algorithms)、粒子群算法（swarm-based algorithms）等。由于 VRP 问题具有高度的特别（ad hoc）性,不同的启发式算法往往在求解某一类具有特定结构的问题时显得非常有效。然而，由于各类算法在设计上都有一定的局限性，一旦问题的条件出现变化，单一依靠某一特定的方法进行求解往往会在求解时间或质量上受到限制。这一现象随着社会经济的发展和物联网技术的普及将变得更加明显，因为不断有新的约束引入到标准车辆路径问题，使之变得越来越复杂。

近年来，来自计算机人工智能领域的学者和传统运筹学领域的学者提出了混合优化（Integrated Optimization）的思想，旨在将不同的优化方法（如约束规划、数学规划以及元启发算法等）融入统一的算法框架，实现扬长避短、优势互补。基于这种思想，本论文选择最基本的带资源约束的路径优化问题为研究对象，通过对几个经典启发式算法的深入研究，设计和开发可以将这几种算法的优势相结合的混合优化算法。通过该研究，对解决更加复杂更加具有实际意义的 VRP 问题，改进或者发现新的可以构建高质量并且有良好鲁棒性的解的算法都具有重要的理论意义和实践意义。

1.2 国内外研究现状

1.2.1 国外路径优化问题研究现状

VRP 问题是组合优化领域被最广泛研究的问题之一，它在 1959 年最先由 Dantzig and Ramser 提出[5]。该问题有很多种变体，最常见的是带资源约束的 VRP 问题（CVRP）以及带有时间窗的 VRP 问题（VRPTW)。其后，人们将更多地注意力放在对 VRP 的复杂变体的研究上，其被称为“丰富的”VRPs(“rich” VRPs)。这些问题更接近于实际分销问题，在具体问题中可能会有多个配送中心，每辆车可能会执行多条不同的路线，车的类型也会有所不同等等情况。而近些年，由于信息和通信技术的进步使得信息可以被实时获取并处理，有关动态车辆路径问题(DVRP)的研究开始兴起。图 1.1 列出了与 VRP 问题相关的研究方向：

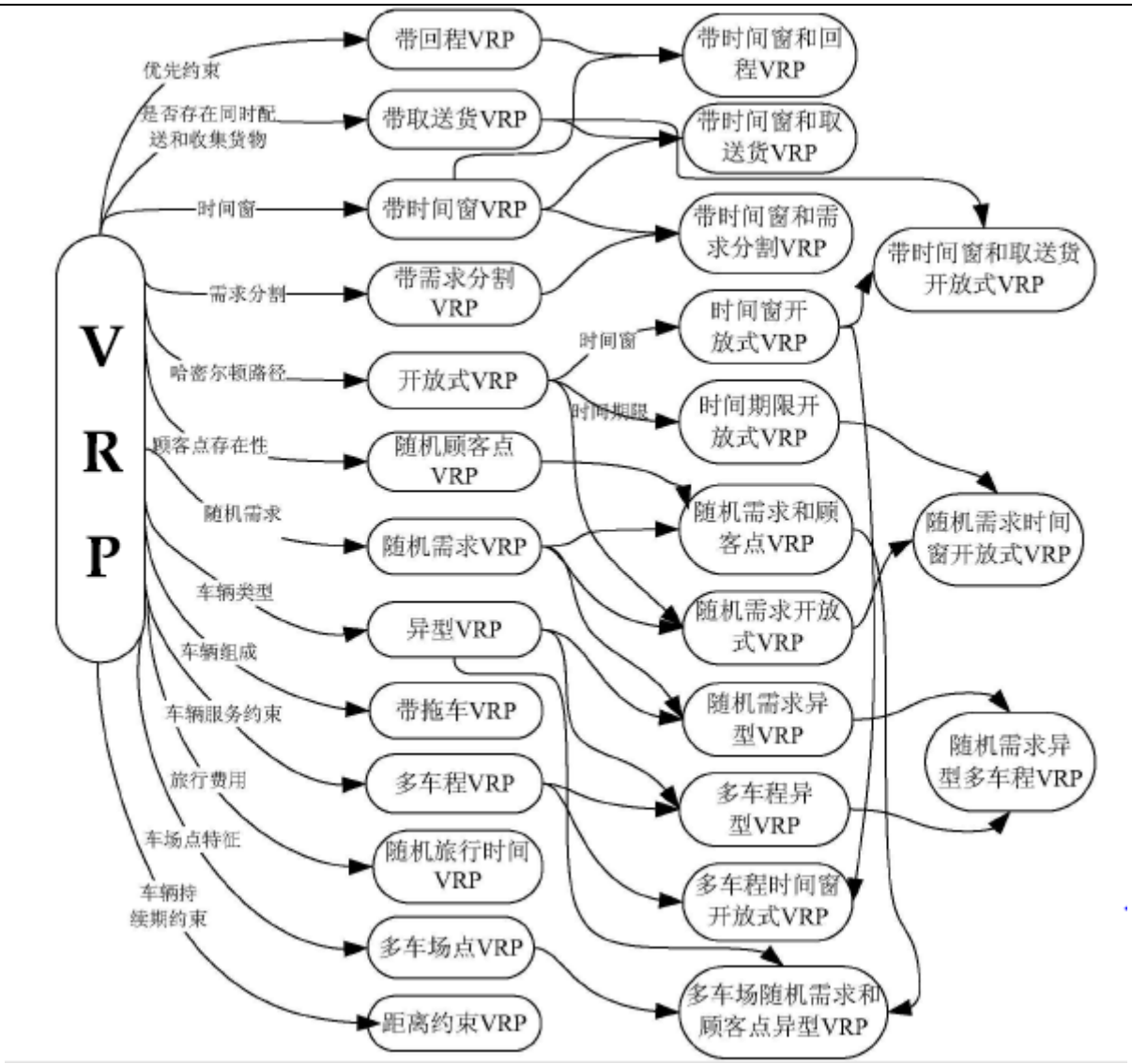


图 1.1 VRP 问题相关研究方向

车辆路径问题的求解方法一般可分为两类：精确算法与启发式算法。下面分别对其研究现状进行介绍：

1）精确算法：

由于最初 VRP 问题涉及结点较少,规模也较小,所以众多学者关注于对精确算法的研究,提出的主要精确算法有:割平面法、动态规划法、K 度中心树算法、三下标车辆流、二下标车辆流、分枝定界法等。近期众多国外学者的研究成果按时间顺序可列表 1.1 如下。

表 1.1 近期国外学者在精确算法领域的研究成果

时间	作者	成果
2007	Ropke,Cordeau,Vigo	提出用于解决带有二维运输限制的CVRP问题的分支定界定价法（Branch-and-Cut-and-Price）[6]
2009	Qureshi,Taniguchi,Yamada	提出一种新的列型分代收集精确优化算法（column generation-based algorithm）可以求解带有半软时间窗的 VRP 问题。[7]
2010	Gutiérrez-Jarpa, Desaulniers 等人	提出了用于解决带有时间窗的复合车辆路径问题的算法。[8]
2010	M. Gendreau,J.-Y. Potvin	提出一种新的列型分代收集精确优化算法（column generation-based algorithm）可以求解带有半软时间窗的 VRP 问题。[9]

精确算法诞生较早,是基于运筹学的优化算法,可以求得最优解。但是该方法只适用于小规模 VRP 的求解,并且通常只针对特定问题才能进行设计。如果问题规模扩大,该方法的计算量将呈指数形式爆炸增长。因此当 VRP 规模增大时,精确算法不具有实用性。

2）启发算法：

随着 VRP 问题研究的深入,VRP 问题涉及的因素越来越多,分类也越来越细,由此目标函数以及约束条件也越来越多。有些约束条件数学构造虽然简单,但实际求解极为困难,已被证明为 NP 难题。因此,近年来专家们开始关注启发式算法的应用。

传统启发式算法主要包括有:节约法、扫描法、插入法、最邻近法、两阶段法等。目前,传统启发式算法在 VRP 问题中已有广泛应用。传统启发式算法相比于精确算法,不再局限于小规模 VRP 问题的求解,应用范围更广,运算时间和复杂度也大大降低。

20 世纪 70 年代以来,随着仿生学、遗传学和人工智能科学的发展,许多学者开始将智能算法应用到优化领域以解决相关问题。关于车辆路径问题的启发式算法的研究在 90 年代左右兴起，早期的算法很零散，并且主要集中在对禁忌搜索的研究上（比如 1993 的 Taillard，1994 的 Gendreau，2003 的 Toth and Vigo[10]），不过近些年研究开始趋于理性。最好的智能启发算法是那些在搜索空间上及有深度也有广度，同时可以解决该问题的多种变体。他们既可以用于几种变换中，比如用于可调式的大规模邻域搜索（Pisinger and Ropke，2007 年提出[10]），也可以用于将局部搜索和基因搜索结合的问题，比如混合遗传算法（Vidal 在 2010 年提出[10]）

常见的智能算法有:禁忌搜索算法(Tabu Search,TS)、遗传算法(Genetic Algorithm,GA)、模拟退火算法(Simulated AnneaHng,SA)、蚁群算法(Ant Colony Algorithm,ACA) 等。1986 年,Glover 首先提出禁忌搜索算法,其思想是全局逐步寻优。而 Gendreau 等人最先将该算法应用到 VRP 的研究中,随后,Renaud、Willard、Barbarosoglu、Taillard、Duhamel、Philippe 等人利用该算法求解不同的 VRP 问题,取得了大量研究成果。

模拟退火算法最早由Kirkpatrick应用于组合优化领域。该算法是基于蒙特卡洛迭代求解策略的一种随机寻优算法,其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性.[1]最近几年,Teodorovic 、Laporte通过将原子获得能量等同于分配最优结点,将原子振动等同于随机搜索线路的寻优空间，利用模拟退火算法求解VRP问题。遗传算法(Genetic Algorithm)最初是由美国Michigan大学J.Holland教授于1975年第一位提出,它是一种模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型,是通过模拟自然进化过程搜索最优解的方法。Blanton和Wainwright、Potvin、Thangiah、J.Lawrence等人都曾利用该算法解决了有时间窗口的VRP问题, Cheng和Gen利用该算法解决了模糊VRP问题。蚁群算法是由Dorigo等人首次提出,该算法是一种新型仿生类算法,其由蚂奴觅食原理的搜索机制设计而成。近期国外学者的研究成果按时间顺序可列表 1.2如下。

表1.2 近期国外启发式算法研究现状

时间	作者	成果
2001	Berger,Barkaoui等人	提出了用以解决带有时间窗的路径优化问题的混合搜索算法。
2003	Hoong，Sim等人	提出了路径导向的混合遗传算法用以解决带有时间窗的路径优化问题。
2005	Montemanni，Gambardella	使用蚁群系统解决动态车辆路径问题。[11]
2005	Y.-J.Cho，S.-D.Wang	提出了一种阈值接受的元启发算法用以解决带有取回策略和时间窗约束的车辆路径问题。[12]
2006	Gandreau, Guertin, Seguin等人	提出了基于Ejection chain的局部搜索算法用以解决带有取送要求的实时DVRP问题。[13]
2009	Yuichi,Olli	提出了一个强大的路径最小化的启发式算法用来解决带有时间窗的车辆路径问题。[14]
2011	G.-N.Abel,Bullinaria John	提出了改进的多目标进化算法用以解决带有时间窗的车辆路径问题[15]
2011	K.-W.Peng	提出了可调式的并行路径构建启发式算法用来解决带有时间窗的车辆路径问题[16]
2011	S.R.Balseiro,I.Loiseau and J.Ramone	提出了蚁群算法与插入法混合的算法用以解决带有时间窗的依赖时间

		的车辆路径问题。[17]

1.2.2 国内路径优化问题研究现状

近些年,随着物流行业的迅速发展,国内许多学者开始研究 VRP 问题。国内 VRP 问题受到越来越多的关注,虽然国内学者对 VRP 问题的研究时间较短,但是已取得很多很好的研究成果。现阶段,针对 VRP 问题,学者们主要通过改进各种智能算法或者将多种算法取其优点相混合的策略来实现具体的求解。研究成果按时间顺序可列表 1.3 如下。

表 1.3 近期国内启发式算法研究现状

时间	作者	成果
2000	肖鹏等人	构造 VRP 的染色体表达，采用基因换算子进行染色体重组，实现了单亲遗传算法。[18]
2000	李军	设计了基于自然数编码的遗传算法，并将其用于求解非满载的 VRP 问题。[19]
2000	袁健等人	采用神经网络求解了 VRP 问题[20]
2000	祝崇俊。刘民、吴澄等人	通过建立三下标流模型，同时结合 2-opt 算法解决了较大规模的 VRP 问题。[21]
2001	刘浩等人	应用模拟退火算法求解了两车型随机需求的 VRP 问题。[22]
2002	张丽萍等人	通过引入新颖交叉算子，构造了一种改进遗传算法，该算法可以有效求得 VRP 的优化解。[23]
2002	郎茂祥	利用遗传算法求解了一般 VRP 问题。而后其又改进了传统的遗传算法以解决对 VRP 问题搜索效率低等缺陷。[24]
2004	崔雪丽等人	基于人工蚂蚁系统，结合 2-opt 交换策略局部搜索机制，给出了可快速求解 VRP 问题的蚁群算法。[25]
2004	刘云忠等人	应用蚁群算法求解了 VRP 问题。[26]
2005	张建勇等人	运用混合遗传算法结合模糊可能性理论讨论了需求不确定的 VRP 问题。[27]
2009	崔雪丽等人	设计了混合改进型蚂蚁算法来求解 VRP 问题。[28]
2011	于斌等人	设计了一个蚁群最优模型，用以解决带有时间窗的路径优化问题。[29]

近期在该领域的成就主要有以下几个方面：1）对取送问题进行了优化，在该问题中，货物从供应商处被运输至分销中心，在交接设施处进行运输整合。其用美国的一个大型零售商的数据进行了测试。2）为各种路径问题引入了模型和有效不等式。该应用主要起于外勤工作和家庭护理交付 3）概率型的带有时间窗的 TSP 问题，其中顾客的出现为概率事件。作者提出了一个资源模型以及一个可变的邻域递减搜索 4）引入一个军用航空器计划问题，其中一对飞机要攻击许多地面目标。问题可被归纳为带有同步和优先约束的 VRP 问题。它

可由线性规划和整数规划的混合来表示。在实际中可用 Cplex 求解。[10]

随着人们对 VRP 问题研究的深入以及对 VRP 问题解的质量要求的提高，人们开始研究如何在算法中加进人的主观判断以提高解的质量，比如如何在行驶过程中判断到仓库补货的时机，这归结为补货策略问题；另外，人们也开始研究如何结合顾客库存的情况来制定运输策略的问题，这归结为库存运输问题；诸如此类的研究是当前研究的重点。

1.3 论文的研究目标与内容

本论文选择三个基于智能搜索思想的经典启发式算法-局部搜索、禁忌搜索和模拟退火进行研究。通过对算法本质思想的掌握和理解，设计结合三种算法优势的新的混合智能优化算法，并通过计算实验，对国际通用的基准测试数据包 TSPLIB 进行测试，验证和分析混合优化算法在求解大规模 VRP 问题上的有效性。

论文总体上的组织结构可分为三个部分：第一部分给出 VRP 问题的研究背景并对其研究现状进行综述；第二部分为论文的核心部分，主要研究 CVRP 的数学模型，三种启发式算法及改进的智能混合算法；第三部分为实验数据的分析以及讨论并对本次论文进行总结。全文共分五章，主要内容如下：

第二章背景介绍 规范定义了带资源约束的路径优化问题（CVRP），给出了该问题的一般数学模型，对三种智能搜索算法的本质思想进行了综述，并讨论了各算法的优缺点。

第三章算法设计与实现 设计了智能混合搜索算法框架，从不同角度阐述各算法在混合优化策略中的作用，通过对算法输入和输出对象的具体定义，描述了利用面向对象编程方法对该智能混合算法进行开发的过程。

第四章计算实验 首先介绍了算法实验的运行环境和基准测试包 TSPLIB，然后从问题结构、数据规模以及参数设置等几个方面设计了计算机实验，最后通过实验结果，对不同算法进行了全面的比较和分析，系统地研究了智能混合算法的性能和求解质量。

第五章总结与展望 总结全文，给出论文研究结论和成果，并对将来可拓展的研究方向进行了讨论。

1.4 论文创新点

论文的创新点主要由以下几个方面：

1）通过对研究不同温度下是否有距离约束对解的质量的影响，提出了对模拟退火法的改进。根据是否有距离约束而选取不同的温度，改善解的质量的同时增强了解的稳定性。

3）使用智能的迭代方法，在两阶段法的基础上引入随机因素。第一阶段使用禁忌搜索与模拟退火法的混合算法进行 diversification，第二阶段在第一阶段找到邻域最小值的基础上在该解附近进行使用下山法进行 intensification。

2）将模拟退火法与禁忌搜索进行混合，提出智能的混合搜索算法。并使用数值进行对比，验证了算法的有效性以及改进性。

第 2 章 带资源约束的车辆路径问题与元启发算法

本章对论文研究的主要问题和算法设计中用到的元启发算法思想进行综述。文章 2.1 节规范地定义了带资源约束的车辆路径问题的图形问题（graph problem），并讨论了该问题的数学模型。接下来，2.2 节对几个有代表性的元启发算法分别进行了介绍，并讨论了各方法单独使用时的优缺点。

2.1 带资源约束的车辆路径问题

在带资源约束的车辆路径问题（Capacitated Vehicle Routing Problem, CVRP）中，与每位顾客对应的运输量以及每位顾客的需求都是确定的，或者说已知的，而且不能被切分。每辆车都是相同的并且从同一个配送中心出发，而且每辆车只有容量限制。目标是满足所有顾客的需求并且使得总花费最小。

CVRP 问题可以被描述为一个图论问题。设$\mathrm{G}=(\mathrm{V},\mathrm{A})$是一个完全图，$\mathrm{V}=\{0,\ldots,\mathrm{n}\}$为点集，A 为弧集。点集的点对应顾客，0 点为配送点。定义一个非负花费c_{ij}，其为在其对应弧集 A 中的一条弧(i,j)中，从点 i 至 j 的花费，一般为从 i 至 j 的距离。花费矩阵必须满足三角不等式：

$$c_{ik}+c_{kj}\geq c_{ij}\text{对于所有的}\mathrm{i},\mathrm{j},\mathrm{k}\in\mathrm{V} \qquad (2.1)$$

车辆具有的最大装载能力为 C，有 N 位顾客，每位顾客有其需求量 D。车辆从配送中心出发对客户进行配送服务最后返回场站，要求所有顾客都被配送，每位顾客一次配送完成，且不能违反车辆容量的限制，目的是使所有车辆路线的总距离最小。

2.1.1 问题假设

1）每个需求点的需求已知

2）每个需求点的位置已知

3）需求点之间的距离按直线最短距离计算

4）每个需求点只由一辆车服务一次

5）每辆车由配送中心出发再回到配送中心，且每辆车的最大容量相同且已知。

2.1.2 参数定义

该问题一半由图论定义。$\mathrm{G}=(\mathrm{V},\mathrm{A})$为一个完全图，V 是点集，代表顾客和总站，A 是弧集代表路径。c_{ij}表示从顶点 i 到顶点 j 的花费（该花费一般代表两点之间的距离）。D_i表示每个顾客的需求。对于一个包含于 V 的点集 S，令$\mathrm{d}(\mathrm{S})$表示 S 中所有顾客的需求。K 表示车辆数，C 表示车的容量。K_{min}表示需要服务所有顾客的车辆数。$\mathrm{r}(\mathrm{S})$为需要服务 S 中所有顾客的最小车辆数。这样 CVRP 即为寻找最小花费的前提下满足所有顾客的 K 个简单圈。并有如下约束：

- 每个圈必过总站点
- 每个顾客访问点只被一个圈访问

• 一个圈的顾客总需求不能超过车辆容量。

2.1.3 模型建立

基于上面的设置，可以得出 CVRP 的基本模型，为 0-1 整数规划。定义变量：

$$x_{ij}\begin{cases}1 & \text{若}x_{ij}\text{属于最优解}\\ 0 & \text{否则}\end{cases}$$

$$c_{ij}\begin{cases}1 & \text{若}c_{ij}\text{属于最优解}\\ 0 & \text{否则}\end{cases}$$

其数学模型为：

$$\min \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{2.2}$$

Subject to

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.4}$$

$$\sum_{i\in V} x_{i0} = K \tag{2.5}$$

$$\sum_{j\in V} x_{0j} = K \tag{2.6}$$

$$\sum_{i\notin S}\sum_{j\in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \phi \tag{2.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{2.8}$$

(2.2)目标函数表示令每条路的花费的总和最小。

(2.3)(2.4)表示从每个点进入与出去的选择只有一个。

(2.5)(2.6)表示从总站进入的与从总站出去的必须等于总车辆数。

(2.7)表示对于某点集 S，穿过该集的弧的个数一定要不小于服务该集的最小车辆数。（容量切割限制）

2.2 现代启发式算法

由于 VRP 问题涉及的因素很多，目前的解决方法只能对特定的问题运用不同的算法求解。经过国内外专家、学者的归纳总结，求解 VRP 问题时，可以将问题分类为几个简单的组合优化的基本原型，如最短路径问题、旅行商问题(TSP)、中国邮递员问题等，再用与之对应的理论和方法进行求解，最终得到模型最优解或较优解。当前在求解 VRP 问题时可以将所使用的算法大致分为两类，一类是精确算法，另一类是启发式算法。图 2.1 列出了各种启发式算法及其分类。

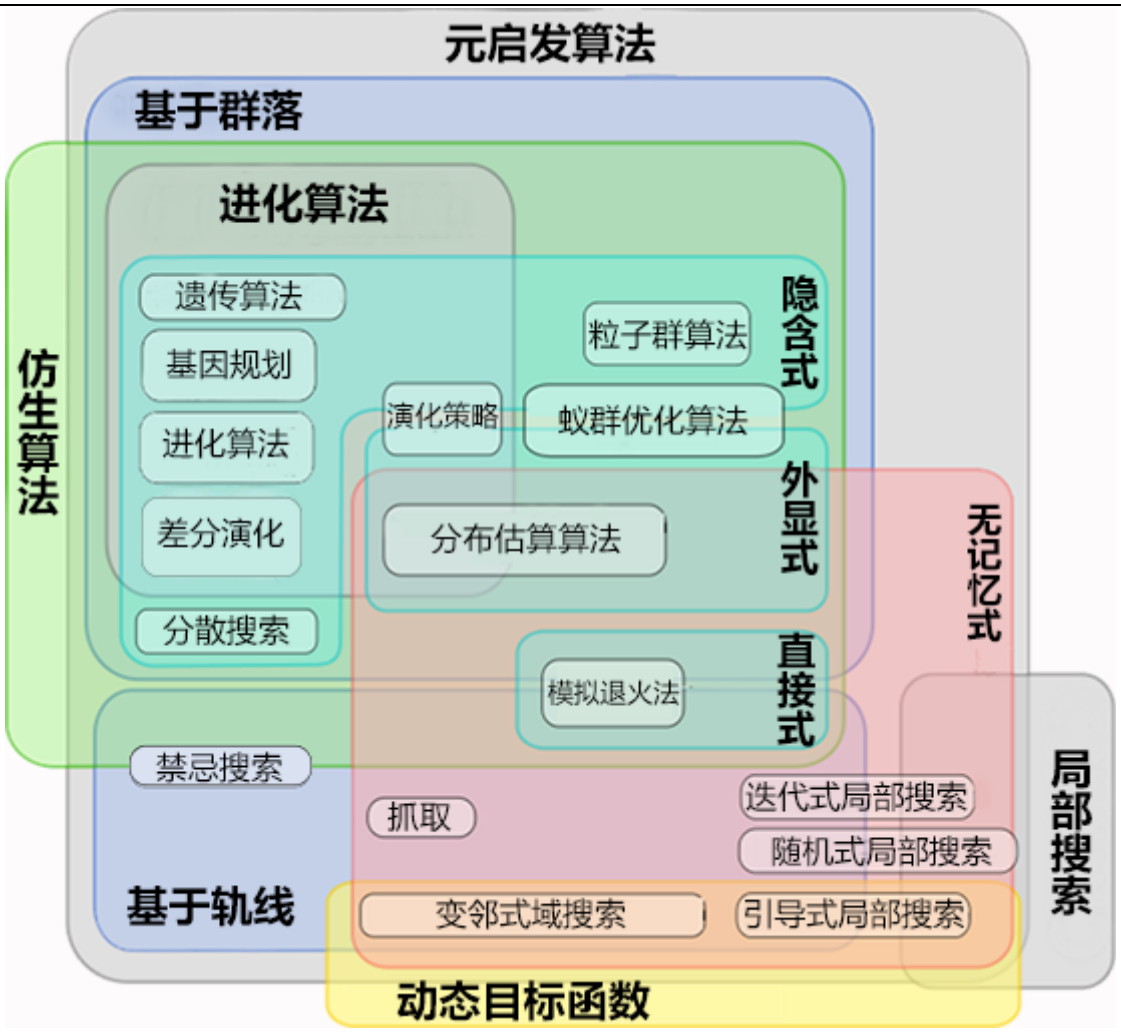


图 2.1 启发式算法分类图

根据 Stutzle 的定义，现代启发式算法是指一种典型的高级策略，该策略引导基本的问题使用特定的启发式算法来增强其性能。大多数现代启发式算法的搜索过程是一种概率决策过程，但是这种搜索与纯粹的随机搜索的最大的不同在于它不是盲目地随机搜索，而是以一种智能的形式进行随机搜索。[30]

有上述定义可以看出，现代启发式算法是一种利用一组不同的启发式算法对搜索空间进行探索的高级策略。它的本质思想就是根据不同的问题实现多样化搜索（diversification）和集中搜索（intensification）之间的动态平衡。多样化搜索是指在整个搜索空间进行大范围的搜索，而集中搜索则是指充分利用多样化搜索是中搜集到的信息对特定的区域进行深度挖掘。这种策略的优势在于，既可以快速的探索包含高质量的解的区域又不必浪费太多资源在探索过的区域或者不太可能包含高质量解的区域进行探索。下面综述一些具有代表性的启发式算法。包括模拟退火法、禁忌搜索、引导式邻域搜索。其中的模拟退火法为本实验的基础，并在此基础上运用禁忌搜索与模拟退火法的结合对问题进行求解。

2.2.1 模拟退火法

模拟退火是一种贪心算法，但是它的搜索过程引入了随机因素。以一定概率接受劣解是其最重要的特性。模拟退火算法与爬山法类似，但是它没有选择最佳的移动，而是选择随机的移动。如果该移动使情况得到改善，那么接受该移动；如果并没有改善，算法以某个概率接受该移动，概率的计算方式为：$\exp(-\frac{\Delta t}{T})$。模拟退火算法要求在算法进行的过程中，逐步降低其跳出局部最优的概率，使其越来越趋于稳定，最终达到全局的最优解。图2.2给出了模拟退火法算法框架。

模拟退火法具有鲁棒性强，计算过程简单，易于与其它算法相结合的优点。但其同时具有收敛速度慢，搜索效率低下，易受初始解以及相关参数的影响的缺点。

Step 1. 参数设置和初始化
    s = **GenerateInitialSoltion**( )
    InitializeAnnealingParameter
    s_{best}=s
    n=0
Step 2. 当算法外部循环满足终止条件时，转到 Step3，否则执行下列过程：
    Step 2.1 当算法内部循环满足终止条件时，转到 Step 2.2，否则执行下列过程：
        s' = **GenerateNeighbor**(s)
        s = **AcceptSolution**(T_n, s, s')
        若 f(s)<f(s_{best})，则s_{best} = s
    Step 2.2 **UpdateTemp**(n)；n = n+1，转到 Step 2.
Step 3. 输出最优解s_{best}

图2.2 模拟退火法算法框架

2.2.2 禁忌搜索算法

禁忌搜索是一种带有记忆功能的启发式算法，它使用一个禁忌列表来记录最近实施的搜索过程，通过将局部最优解的某些特征列为禁忌从而在接下来的若干步迭代中禁止触碰带有这些特征的解。该长度成为禁忌长度。这样就避免搜索多次回到同一解上，以便提高算法跳出最优解的能力。但当搜索到优于当前最优解的解时则破除禁忌，从而提高搜索效率。在选择禁忌对象时，禁忌搜索会形成一个候选解集，从该解集中选择节省路径最多而没有被禁忌的对象。图 2.3 给出了简单的禁忌搜索的算法框架。

禁忌搜索算法的特点是其在搜索过程中采用了禁忌技术。其核心即禁止重复已进行的搜索，这样就避免了局部搜索易陷入局部最优的不足。通过使用一个禁忌表来记录下搜索到的局部最优点，从而在以后的搜索避开这些点进而逃出局部最优。该算法的缺陷在于对初始解具有较强的依赖性，当初始解较好时，禁忌搜索可以得到更好的解；而当初始解较差时，则算法的收敛速度会变慢且解的质量也较差。

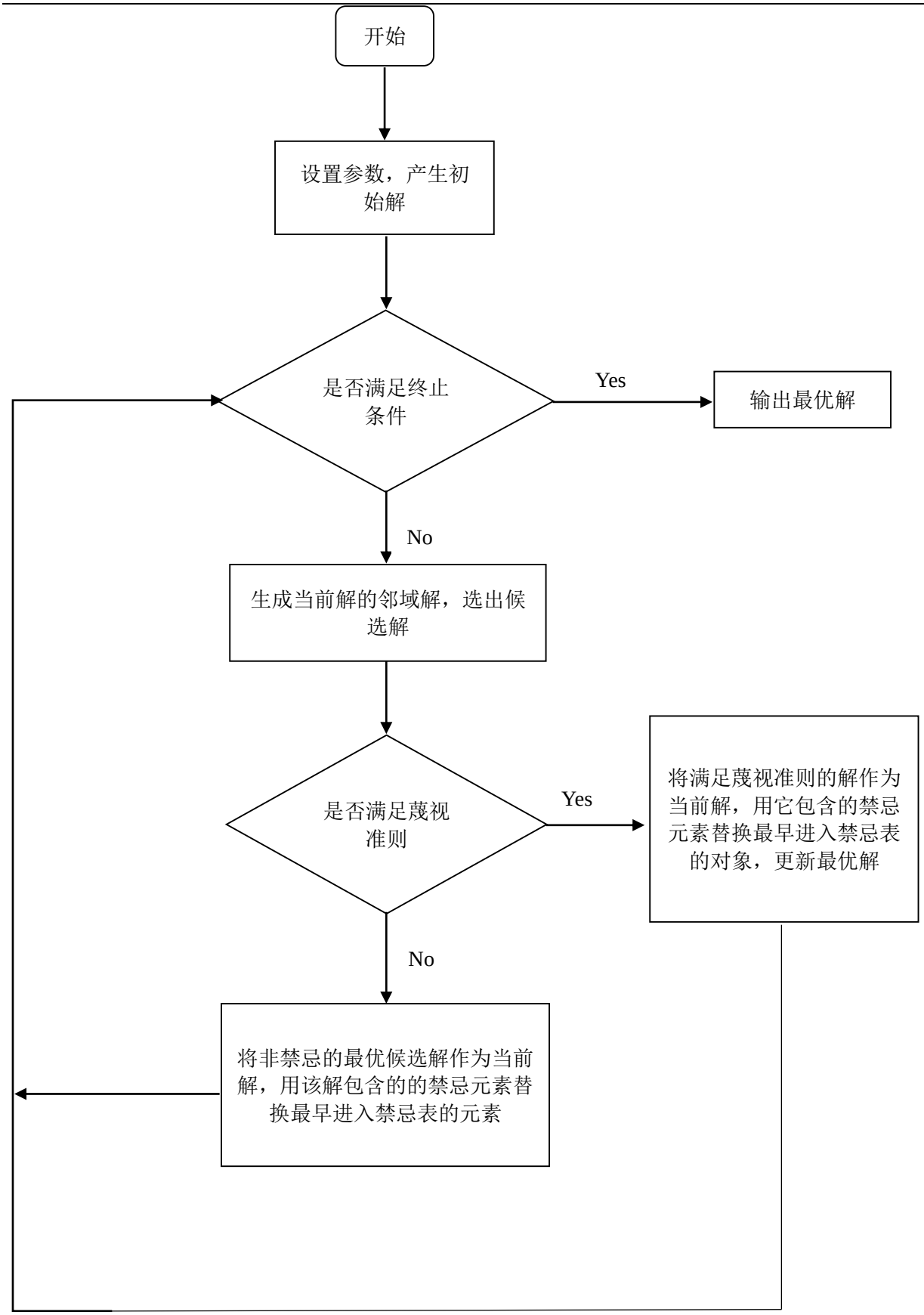


图 2.3 禁忌搜索算法框架

2.2.3 引导式邻域搜索

引导式局部搜索是基于惩罚的通用启发式算法。该算法的工作机理是，基于搜索获得的经验为目标函数增加一个惩罚系数。大致来说，如果搜索游离到距离之前已被访问的局部最小值太近的话，它将会被惩罚。

引导式局部搜索通过惩罚特殊解的特征从而移出局部最小值，该特征被认为不应该出现在近似最优解中。该算法定义了一个被修正的目标函数，该函数带有一系列与解的特征有关的惩罚项的参数。然后我们会调用传统的局部搜索改善参数目标函数。

引导式局部搜索需要如下部分：

1）一组特征F。对于特征$i \in F$，令该特征的指标函数为f_i。如果特征 i 在解 S 中，则令$f_i(S) = 1$，否则令$f_i(S) = 0$。

2）一个成本矢量为c，c_i表示特征 i 的成本。

3）一个惩罚系数 λ 。本实验中 λ 取值为 0.2。

引导式局部搜索通过一个惩罚矢量 p 追踪被执行的惩罚。P_i是特征 i 目前为止被惩罚的次数（整数）。假设O(S)是问题的原始目标函数，引导式局部搜索定义如下参数目标函数：

$$O'(S) = O(S) + \lambda \sum_{i \in F} f_i(S) p_i c_i \quad (2.9)$$

在本实验中，被惩罚的特征元素为边，构建一个惩罚矩阵p[i][j],为每条边的惩罚项。每确定进行一次移动，便将该次移动所涉及的边的惩罚项加 1。在进行节约值的比较时，使用带有惩罚项的节约函数：Saving()。即每条边的距离值变为：$d[i][j] + 0.2 * d[i][j] * p[i][j]$。

第 3 章 求解带资源约束的 VRP 问题的智能混合算法

在上一章中论文给出了带资源约束的 VRP 问题的定义和数学描述，并对几个常用现代元启发算法思想进行了介绍。本章重点阐述论文研究的主要内容，既针对带资源约束的 VRP 问题设计和开发的智能混合算法。在论文的 3.1 节，首先介绍了求解 CVRP 启发式搜索算法的一般过程和智能混合算法框架，然后在此基础之上，通过依次增加模拟退火、自适应性模拟退火以及混合模拟退火与禁忌搜索等策略对算法进行改进和变形；在 3.2 节，通过面向对象的编程思想，对算法的计算机实现进行了详细解说，包括输入输出的数据类型以及各函数模块之间的信息传递与交互过程等。

3.1 算法设计

3.1.1 算法的基本思路与智能混合算法框架

基于搜索的启发式算法求解带资源约束的 VRP 问题的一般过程如图 3.1 所示，主要分为四个部分：

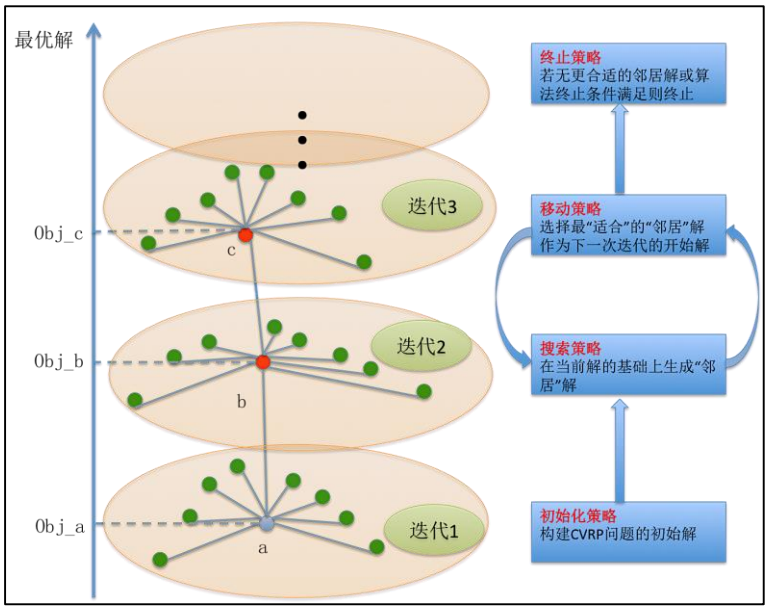


图 3.1 基于搜索的启发式算法梗概

1） 依据构建策略构建初始解

由于带资源约束的 VRP 问题的决策变量与条件约束很多，对于一个给定的 CVRP 问题，在各变量的求解区域内找到满足所有条件的可行解本身就是一个挑战，且初始解的好坏往往对接下来的建立搜索空间、循环替代当前最优解直至算法终止的整个过程都有重要影响。图 3.1 中迭代 1 中的蓝色节点代表一个初始解。

2） 依据邻域搜索在当前解的基础上建立“更优”解集合

在得到了初始可行解之后，基于搜索的启发式算法的下一个步骤就是搜索可行解区域以期得到比当前解的目标函数值更优越的解。然而，如前面章节讨论过的，CVRP 的可行解个数常常是以问题规模（如节点个数）的指数倍存在的，完全搜索（complete search）在

计算时间和空间的约束下往往是无法实现的。基于建立当前解邻域的搜索是解决该问题的一种思路。一个解的领域一般通过制定某种规则，对当前解的某一个或一组决策变量的值进行改变而生成。邻域搜索的出发点在于如果当前解已比较令人满意，那么最优解与当前解的差别应该不会太大。通过这种布局的搜索，可以大大缩短求解的时间。但是，在进行邻域搜索时要特别注意两个问题。一是邻节点的个数不能过大，否则就和完全搜索一样，达不到节省时间的效果；二是邻居的类型要多样化，才能较快的完成从初始解向最优解的进化。图 3.1 中每一次迭代中与当前节点相邻的节点都代表当前解的一个“邻居”，而这些“邻居”的集合构成了当前解的一个“邻域”。

3） 依据移动策略在邻域中选择下一个解继续搜索

邻域搜索之后，通常会生成一系列比当前解的目标值或优或差的候选解，接下来要在这些候选者中选出一个解进行移动，替代当前解。移动策略对搜索算法的性能也有很重要的影响，因为它决定了求解的方向。图 3.1 展示了基于贪婪思想的移动策略，在每一次迭代中总是选取目标值最优的邻节点进行移动。好的移动策略可以使搜索范围快速向最优解靠拢，而某些单一的移动策略虽然在短时间内可以快速优化目标值，但容易陷入局部最优的圈套，无法逃出。

4） 依据终止策略终止搜索

搜索算法的另一个优点就是具有灵活的终止机制，算法可以根据运行时间、迭代次数、当前最优解的质量等不同参数选择终止，具有较强的可控性，有利于实际生产中的应用。

通过以上讨论和对各种现代启发式算法的深入理解和分析，本论文提出智能混合搜索算法框架，其基本结构如图 3.2 所示。

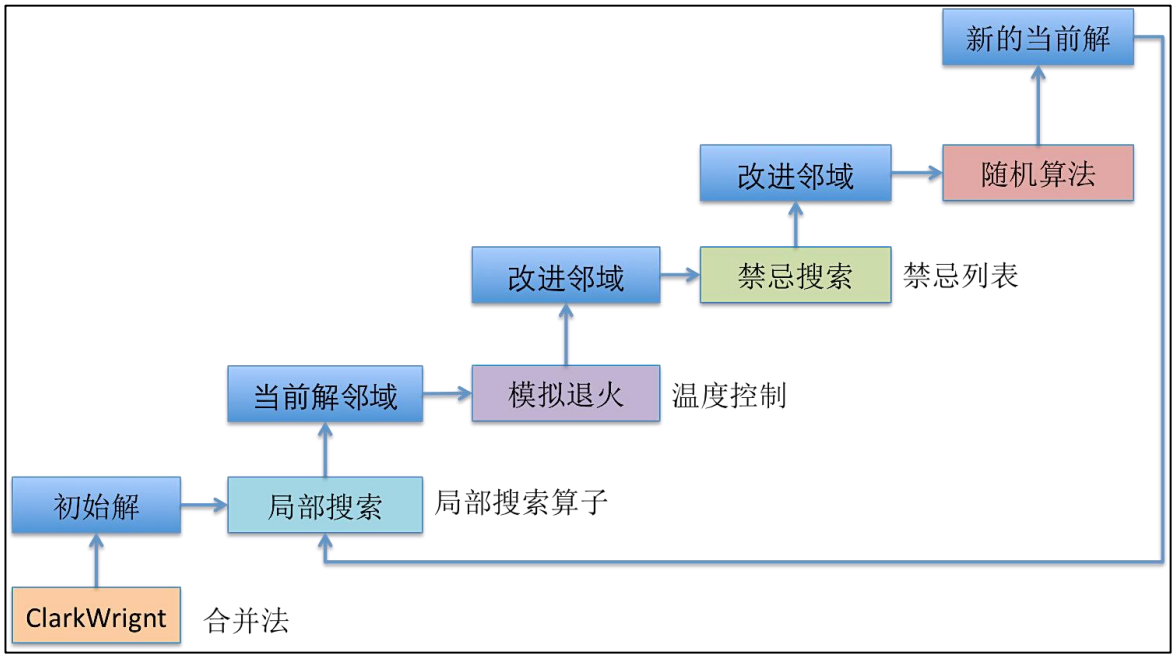


图 3.2 基于多种启发式方法的智能混合搜索算法框架

该混合算法集中了多个启发式算法的优势。ClarkWright 算法有助于产生高质量的初始可行解，局部搜索、禁忌搜索和随机算法有利于增加解的多样性，而模拟退火算法有助

于摆脱算法陷入局部最优的困境。下面对该混合算法的各个部分进行细致的说明，详细介绍论文在研究不同策略的过程中进行的算法设计。

3.1.2 基于 ClarkWright 的初始解构建策略

该方法用于建立初始解。该算法的核心思想是节约算法，即将两条路径合并为一条从而“节约”路径。如图3.3.

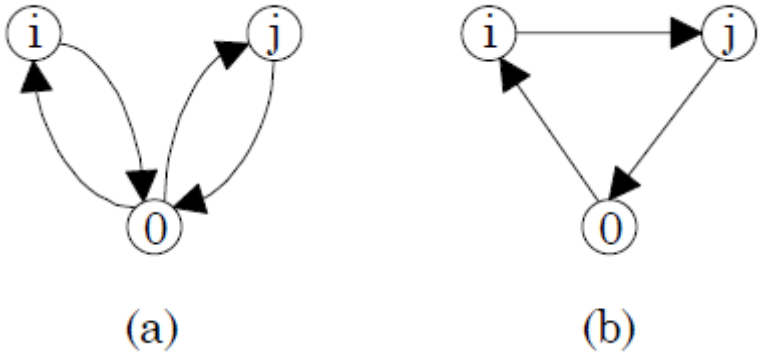


图3.3 节约法

(a)方案的花费为： $C_{0i}+C_{oj}+C_{i0}+C_{j0}$

(b)方案的花费为： $C_{0i}+C_{j0}-C_{ij}$

节约值:$S_{ij}=C_{i0}+C_{0j}-C_{ij}$

其步骤大致如下：

首先，计算每个客户点之间的节约值 savings，然后按照降序排列。接着从序列顶部抽出一对点进行检查，要保证如下两点：

1）将 i-j 相连后不会删除原先已经建立的客户点之间的联系。

2）不得超出容量限制和距离限制。

3）上述步骤完成后，重新进行排列并检查。

算法框架如图3.4所示：

1：**for all**(i, j) with 0 < i < j <= n **do** 计算$s_{ij}=d_{0i}+\lambda d_{ij}+d_{0j}$

2：**end for**

3：创建一个三维列表 L(i, j, s_{ij})，根据s_{ij}的值对表 L 进行降序排列

4：**While** L ≠∅ **do**

5：　选择列表 L(i, j, s_{ij})最顶端的值。

6：　**if** 如果客户点 i 和 j 与配送点相邻并且在不同的路径中 **Then**

7：　　**if** 可以通过合并 i 和 j 所在路径获得新路径 **then**

8：　　通过增加一条与 i 和 j 相连的边将这两条路并进行合并。

9：　　**end if**

10：　**end if**

11：　将(i, j, s_{ij})从 L 中移除。

12：**end while**

图3.4 ClarkWright 算法框架

图 3.5、3.6、3.7 展示了该算法不同阶段的解。图 3.8 展示了在该初始解基础上使用智能混合算法的最终解。

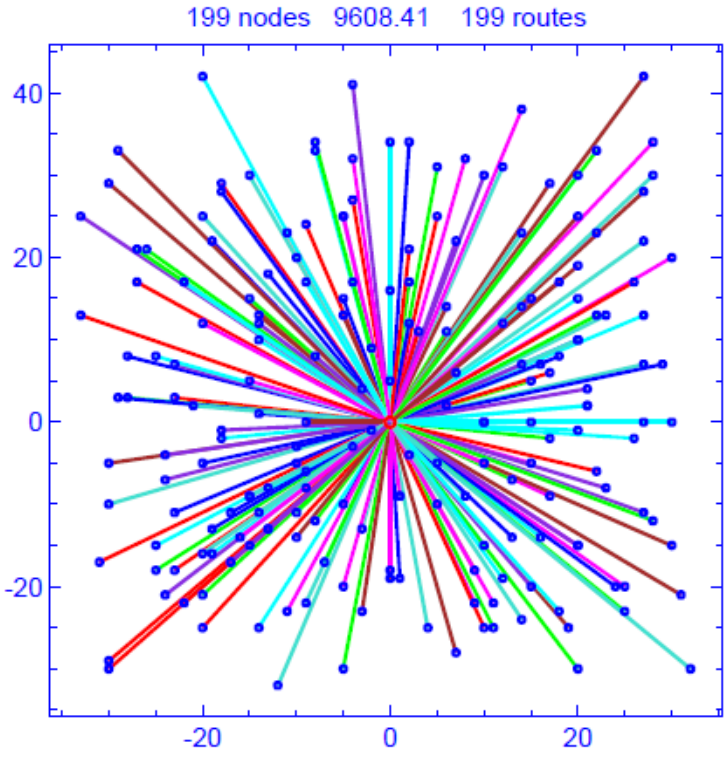


图 3.5 初始解

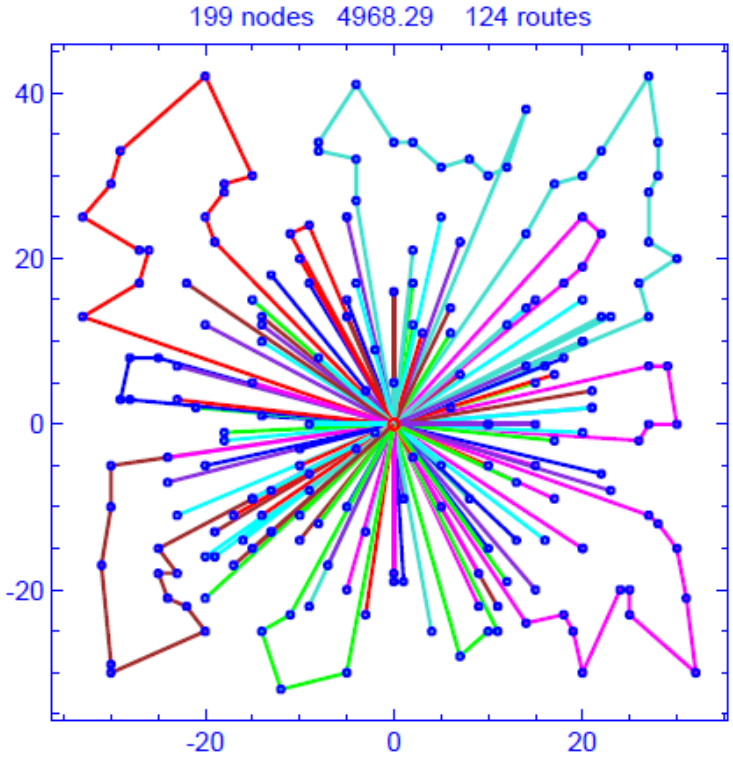


图 3.6 合并 100 次之后的解

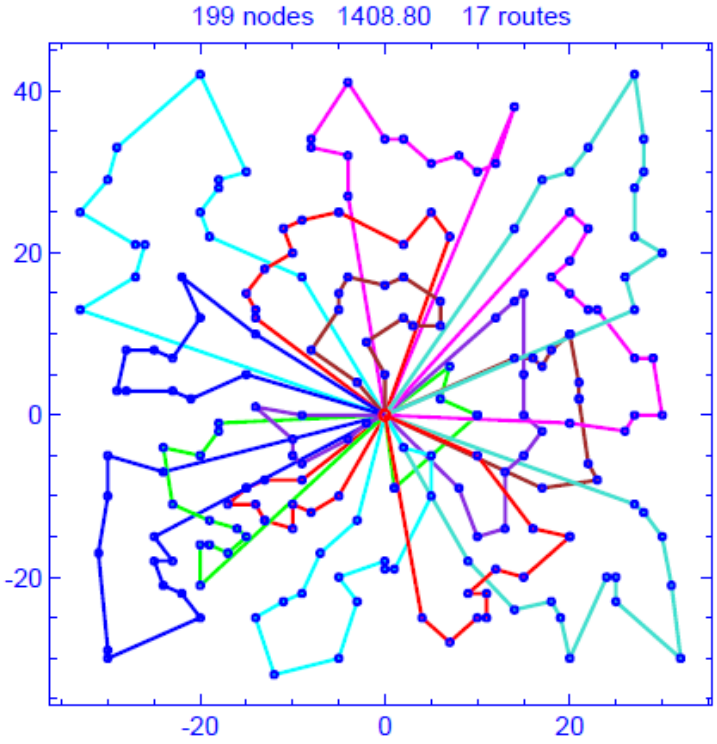
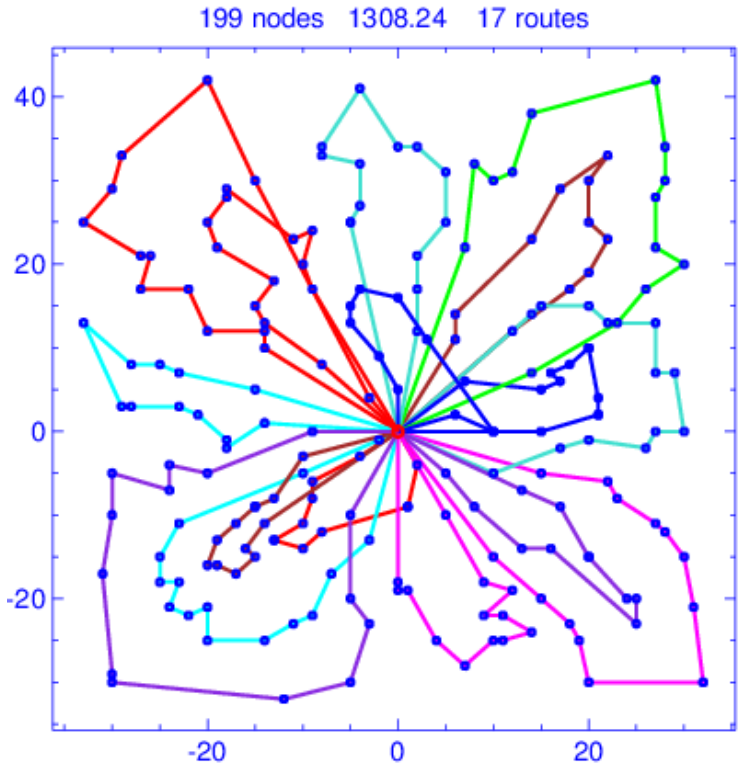


图 3.7 初始解构建完成



3.8 混合算法最终解

3.1.3 基于局部搜索的邻域搜索策略

局部搜索时组合优化问题中最常用的技巧之一。所谓局部搜索，其实质是变换。是指使用特定的操作方式对一定区域内的元素进行变换。对于 VRP 问题来说，所谓的邻域实际上是跟据局部搜索策略决定的，即解中的一个点被移动至一个新的位置或者几条边被移除或者被新边取代。换句话说，对于特定的解和局部搜索策略，邻域即对当前解使用局部搜索策略所能得到全部解。局部搜索已被证明对处理 VRP 问题十分有效，并且这项技术已被用于许多 VRP 元启发算法中。

比较常用的局部搜索策略有有 2 最优，交换，单点移动，两点移动等等。通过比较搜索前后的解决定是否接受该解，若接受，则再在该解的基础上进行下一步搜索。但是这种方法最大的缺陷就是容易陷入局部最优解，即搜索过程陷入某种循环。为此，需要使用其他算法对其进行改进。本研究使用的局部搜索有单点移动、两点移动、2 最优移动。

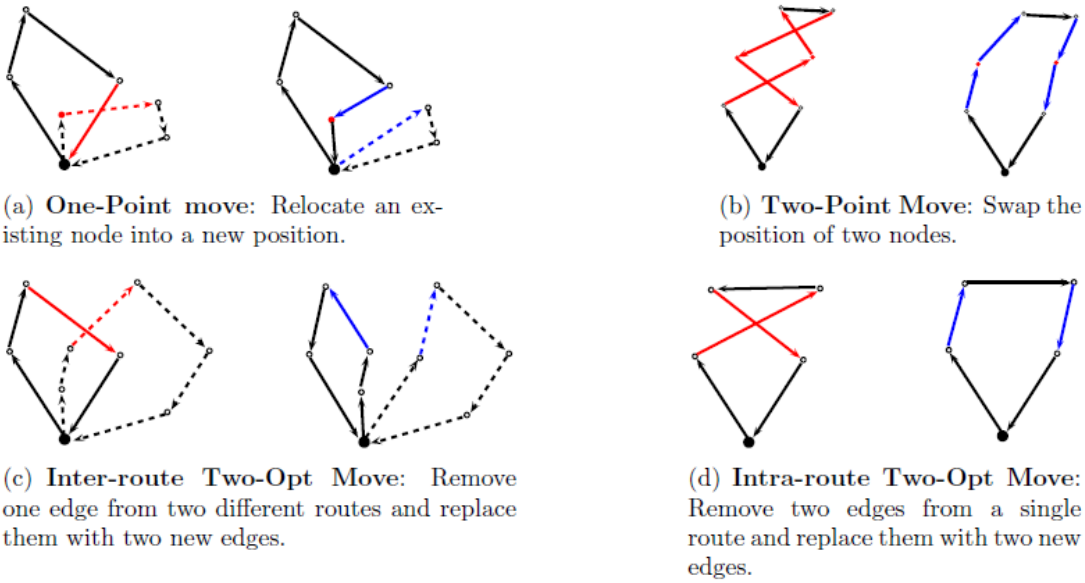
对于 VRP 问题，大量文献证明：在局部搜索中加入禁忌搜索策略可大幅提高搜索效率。[31]因此在图 4.1 中混合算法局部搜索阶段，考虑使用禁忌搜索算法。禁忌搜索是由 Glover 提出的一种带有记忆的局部搜索策略，它利用这种记忆式算法跳出局部最优解。本算法使用具有短期记忆的禁忌搜索。本算法的禁忌搜索阶段包含以下几个重要组成：

1）初始解：为了开始进行搜索，禁忌搜索需要一个初始解，本实验使用由 ClarkWright 产生的解作为初始解。

2）解的评价：本实验以模拟退火机制对解进行接受。

3）邻域结构：本实验在执行禁忌搜索算法时，对当前解依次使用三种算子来产生邻居

解。即 2-opt，单点移动，两点移动。具体操作如图 3.9 所示：



(a) **One-Point move**: Relocate an existing node into a new position.

(b) **Two-Point Move**: Swap the position of two nodes.

(c) **Inter-route Two-Opt Move**: Remove one edge from two different routes and replace them with two new edges.

(d) **Intra-route Two-Opt Move**: Remove two edges from a single route and replace them with two new edges.

图 3.9 局部搜索算子

4）禁忌对象和禁忌表：本实验将路径作为禁忌元素，即将每一个被接受的解中受影响的路径作为禁忌元素加以禁忌并记录在禁忌表中。根据文献[31]所述，这里采用随机禁忌表长度。禁忌表的长度在[5,10]之间随机选取。

5）特赦准则：在禁忌搜索过程中，可能出现全部邻居解被禁或者目前为止的最优解被禁的情况，因此本算法设置以目标函数值为判断标准的特设准则。

3.1.4 基于模拟退火与自适应模拟退火的移动策略

该策略的主要过程为：在建立初始解之前，首先对问题的类型进行判断。若问题仅有容量约束则将模拟退火法的初始温度设置为 2，若问题同时具有容量和距离约束，则将初始温度设置为 6。然后在由 ClarkWright 初始化的解的基础上对每一个点在其邻域内依次进行三种邻域搜索，每进行一次搜索则首先判断当前解是否满足约束，若满足约束则再判断当前解的节约值是否小于 0，若小于 0 则将该次移动标记为可行，若大于 0 则以一定概率决定是否将该移动标记为可行。概率公式为为：

$$\exp(-\frac{\Delta t'}{T}) > \text{random}[0,1) \tag{3.1}$$

其中 $\Delta t'$ 表示节约值，T 表示当前温度。从该公式可以看出，当 T 越高或者当前解与最优解的差距越小则接受劣解的概率越大。该策略的伪码如图 3.10 所示。其中 K 为总循环次数，D 为每次循环迭代的次数。

1. 判断问题类型并对参数进行设置
   若问题仅带有容量约束，则 **Initial Temperature**=2
   若问题带有容量和距离约束，则 **Initial Temperature**=6
2. 产生一个随机的$\lambda \in (0.5,2)$，然后使用 ClarkWright 算法生成一个初始可行解
3. 设置 k = 0
4. **while** k < K **do**
5. **for** i = 1 to D **do**
6. **for** 每种搜索算子 **do**
7. **for** 所有当前解中的点 j **do**
8. **if** 移动后的解优于最优解，接受该移动
9. **else if** 移动后的解劣于最优解，则根据概率

$$\exp\left(-\frac{\Delta t'}{T}\right) > \text{random}[0,1)$$

决定是否接受该移动

10. **end if**
11. **end for**
12. **end for**
13. **end for**
14. k++
15. **end while**
16. 返回发现的最优解

图 3.10 基于自适应模拟退火法的伪码

3.1.5 基于模拟退火与禁忌搜索混合的移动策略

该策略以 3.1.3 中的自适应模拟退火算法为基础，在此基础上融合禁忌搜索。

本实验采用的禁忌搜索为，由 ClarkWright 产生初始解 S，在初始解的基础上进行邻域搜索，生成当前解的所有邻域解$\{S_i\}$，然后根据节约值排序选出最优的五个候选解$\{S_i'\}$。对候选解进行判断：1）是否被禁忌 2）是否满足蔑视准则。没有被禁则接受该解，并将该解的禁忌元素添加入禁忌列表中。若被禁则检查是否满足蔑视准则。若满足蔑视准则，则将该解作为当前最优解并破除禁忌，重新将该解的禁忌元素添加入禁忌列表。若不满足则不接受该解。

禁忌对象：本实验将每次移动影响到的路径作为禁忌对象。

禁忌表的设计：

本实验的禁忌表中的元素为被禁忌路径 r 对应的 hash 值。每次检查当前路径是否为禁忌时，只需检查该路径的 hash 值是否已存在与禁忌列表中。Hash 值由一个 hash 函数产生。产生步骤如下：

1）检查路径的是否为有向的（即路径中第一个客户点的索引值小于最后一个的，否则对路径进行颠倒）

2）将路径中相邻的两点的索引值相加产生一个新的数组。比如：路径 r：0-2-4-5-6-0。经变换后为{6,9,11}

3）选择三个随机数与这三个数相关联：{Y[6], Y[9], Y[1]}

4）做 XOR 和（$Y[6] \oplus Y[9] \oplus Y[1]$）

5）取该值的对数后得到 hash 值。

禁忌表长的设计：

禁忌表长是T_s中的一个关键因素。一般根据实际问题的不同而不同。表长过短会导致搜索过程中出现循环，过长会导致太多的禁忌，降低搜索的作用，本文后面的实例初始禁忌长度为 3。

候选解：候选解为每个点邻域搜索后得到的 5 组最优移动。

该移动策略采用的混合机制主要通过以下三点实现：

1）在搜索过程中，每进行一次移动之前先对该移动进行评价，评价过程主要包括：检查移动是否满足容量约束，检查各个客户点的需求是否满足，如果有距离约束检查距离约束是否满足，检查移动后的解是否优于最优解，若优于最优解则接受该移动；若不优于最优解，则通过判断该解是否满足退火概率决定是否接受本次移动。

2）若评价过程接受本次移动，则将移动后的解作为候选解储存到候选解列表中。候选解列表中只保留最优的五次移动。

3）从候选解中随机选择一个移动，判断该移动是否包含禁忌元素从而决定是否执行移动命令。

该移动策略通过将模拟退火法与禁忌搜索的混合实现了在扩大邻域搜索范围的同时避免进行迂回搜索从而既提高解的质量又提高求解速度。同时为了使算法在迭代过程中具有智能，该算法同时使用了两阶段法。即用来 diversification 的 uphill 阶段以及用来 intensification 的 downhill。其核心思想是，在爬山阶段使用混合算法接受一些劣解并以最大效率搜索解空间。搜索完毕找到局部最优解，在该解附近进行下山法的搜索。在多次尝试后无法跳出特定的局部最小值后算法停止。算法框架如图 3.11 所示：

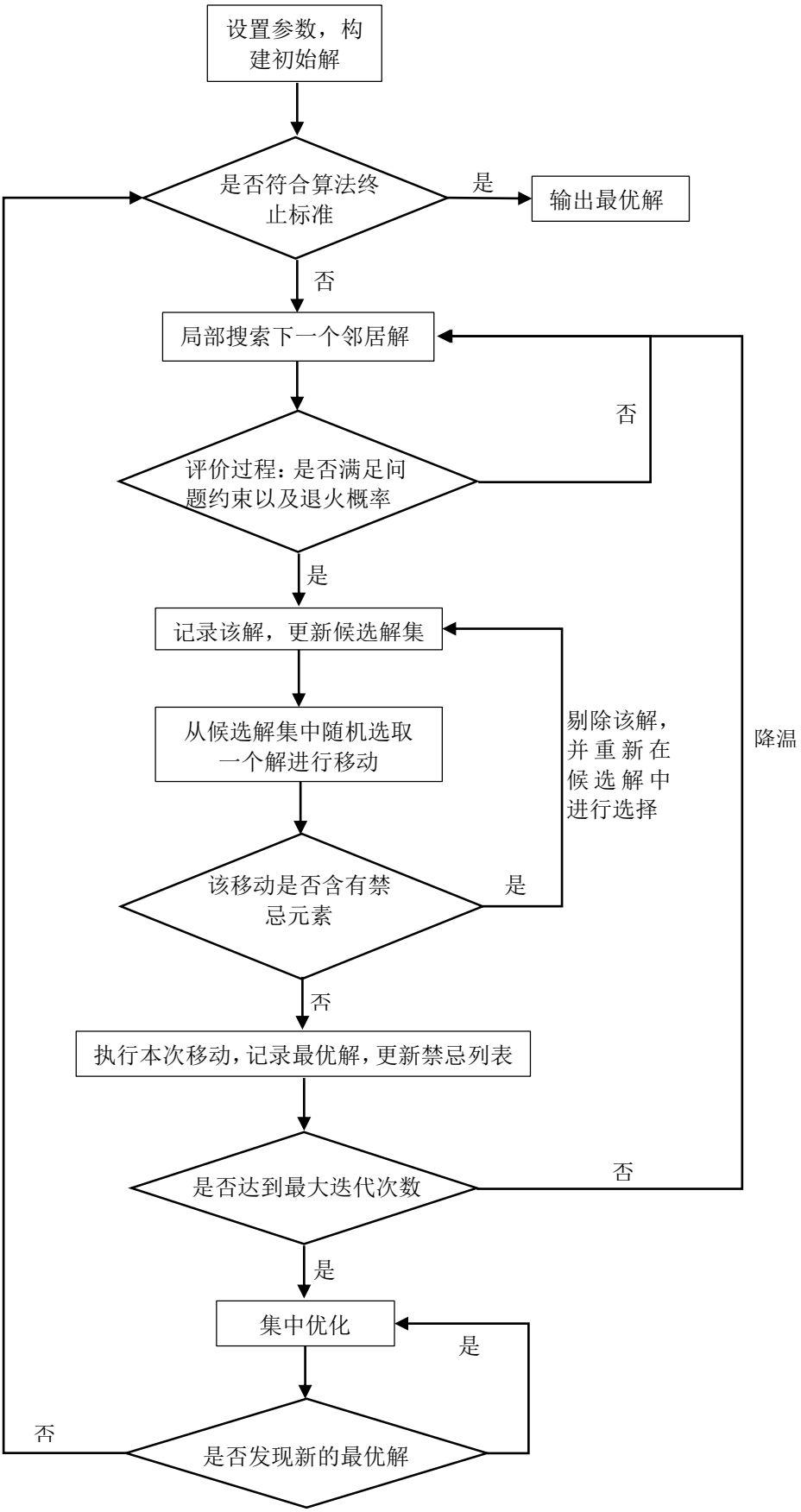


图 3.11 智能混合算法框架图

3.2 算法实现

本论文研究的启发式智能混合算法属于一类实验算法（Experimental Algorithms），必须通过计算机实验（computational experiments）来验证其有效性（effectiveness）和效率（efficiency）。论文使用面向对象的编程语言 C++对算法进行了开发。本节对计算机程序中使用到的主要数据结构、解的存储方式以及各功能模块间的交互关系等进行详细的介绍。

3.2.1 数据结构

VRP

该类为该程序的主类。

该类中包含程序调用内部数据的函数以及进行算法的主要函数。

包含描述一个 VRP 问题的各个数据：已知最优解，客户点数，每个点的坐标，每个点之间的距离矩阵，各个点的需求，容量限制以及路长限制；包含进行算法时需要的数据：最优解和当前解的缓存，邻域大小以及邻域矩阵，模拟退火法中的温度及退火率；其他类：VRPMove，VRPNode，VRPRoute，VRPTabulist 等。

VRPMove

该类用于记录每次移动后的信息。

包含一个用于比较两次移动的函数，该函数用于实现 BestAccept 策略。

包含的数据有：被影响的路径总数，路径编号，路径路长，路径需求，路径客户数，节约值，总路径个数，移动后路径长度，移动类型，受影响的参数的个数等。

VRPNode

该类用于记录每个点的信息

包含点的坐标值，点的编号，点的需求量，点的邻域。

VRPRoute

该类用于记录路径的信息

包含一个哈希函数，用于生成一条路径的哈希值，该哈希值被用于禁忌搜索。

包含一条路的起点、终点、长度、容量、客户数。路径中客户点的排列顺序，该路径的哈希值。

VRPTabulist

该类用于禁忌搜索算法。

包含禁忌列表类以及候选解类。

包含更新禁忌列表的函数，清空禁忌列表的函数，更新候选解以及清空候选解的函数。

包含哈希值，VRPMove 类等。

3.2.2 解的存储方式

当前解被储存在一个 double linked list 中，其由一对数组pred_array[i]和next_array[i] 构成。这一对数组中的元素为指针，通过将pred_array[i]和next_array[i]的指针指向特定的 node 来储存一个解（即路径中节点的连接顺序），其中 negative indices 表示一个新的路径的开始。可由图 3.12 和 3.13 说明：

Route	Ordering
1	0-2-5-8-0
2	0-1-3-4-0
3	0-6-7-9-10-0

图 3.12 一个十个节点的 VRP 实例

i	0	1	2	3	4	5	6	7	8	9	10
$next\_array[i]$	-2	3	5	4	-6	8	7	9	-1	10	0
$pred\_array[i]$	-10	-8	0	1	3	2	-4	6	5	7	9

图 3.13 与图 3.12 中的解相对应的pred_array[i]和next_array[i]

3.2.3 哈希表示

本程序中的禁忌搜索使用一对哈希值表示一个禁忌元素，这样既节省了存储空间同时可以快速查看某元素是否被禁忌，加快了搜索效率。该方法的主要思想是，利用一个“hash_function”将一个解转换成一个“标准形式”再将其存入一个缓存中，最后将该缓存与一个整数关联。已知我们有一个由 n 个点的 VRP 问题产生的具有 R 条路径的解。对于 1 至 R 的每条路径 j，我们用一个有序列代r_j表第 j 条路。此外，我们储存了一列 n 个随机 32 位整数，$Y_0, Y_1, \cdots, Y_{n-1}$。然后通过执行“hash_function”我们将每一个解同一个正整数关联。图 3.14 表示如何利用一个“hash_function”将解转换成一个标准形式。

1：**for** j = 1 to R **do**

2：　　**if** $a_j > z_j$ then 倒转路径 j 的客户点排列顺序

3：　　**end if**

4：**end for**

5：创建一个列表 L，其中包含路径组成的解，可能有倒转的路径
然后根据每条路径第一个被访问的点的序号进行排序

6：则 L=$\{r_1, r_2, \ldots, r_R\}$，将这些路径合并为一个单独的列表
$\{m_1, m_2, \ldots, m_n\}$

7：返回 H=$\bigoplus_{i=1}^{n-1} Y_{(m_i+m_{i+1})modn} mod 2^h$

图 3.14 将解转换成一个标准形式

在步骤 2 中，我们保证每条路都是有向的以便每条路的第一个点的序号值小于该条路中最有一个点的序号值。然后我们在步骤 5，6 将这些路径合并成一个单独的列。在步骤 7 中，我们将序列表中的每对点与一个 32 位的整数$\bigoplus_{i=1}^{n-1} Y_{\left(m_i+m_{i+1}\right)modn} mod 2^h$相关联并且将

他们加和。（我们没有使用正常的加和操作而是按位加和 XOR）

3.2.4 各模块之间的关系

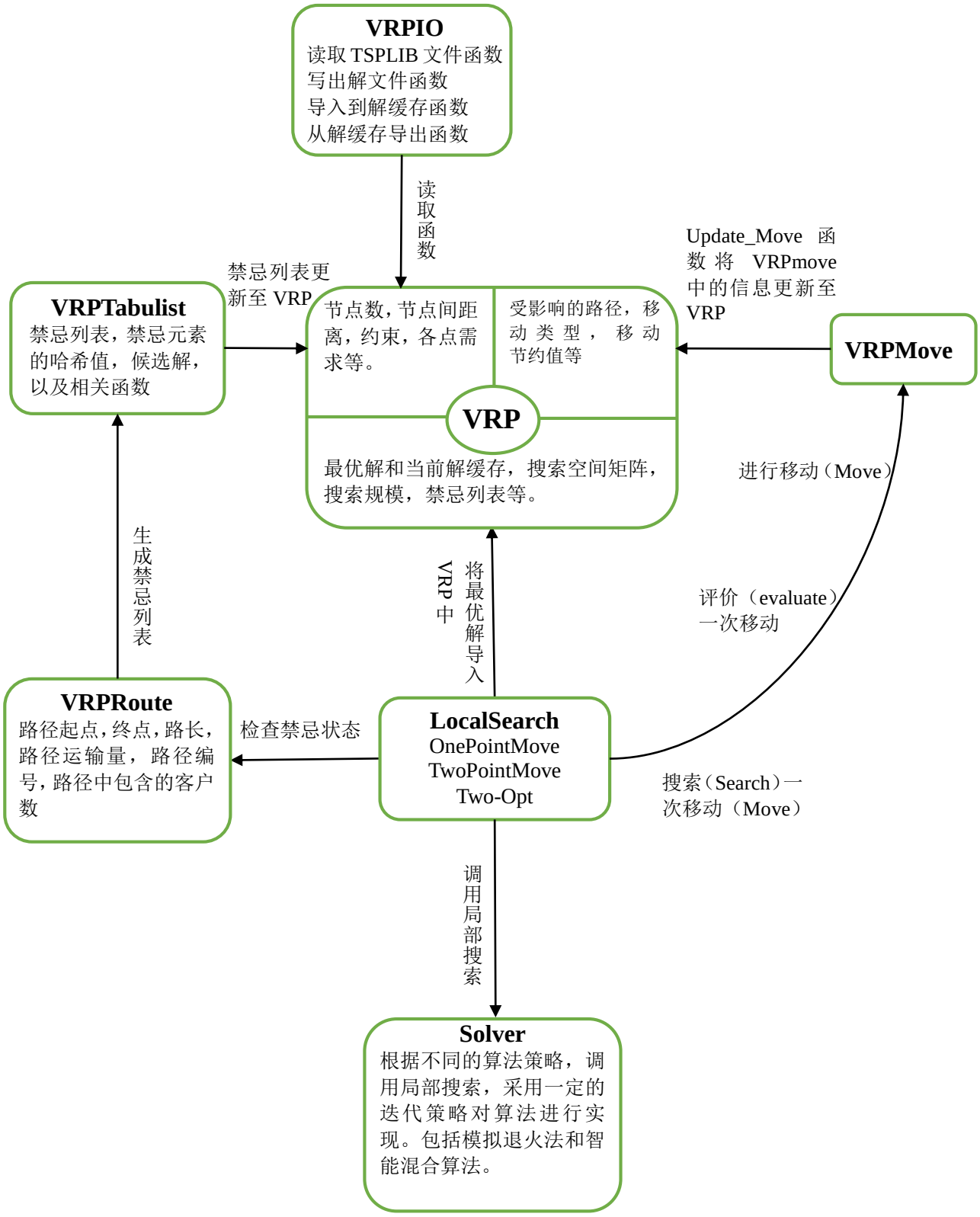


图 3.15 各模块间关系

3.3 本章小结

本章主要分为两部分，分别对现代式启发算法和本实验所使用的程序的数据结构进行了说明。

在第一部分中，详细介绍了本文会用到的算法：模拟退火法及禁忌搜索。并对引导式邻域搜索进行了简单的介绍。在第二部分中，说明了本实验使用的程序的数据类型，解的存储方式，禁忌搜索使用到的哈希表示。并通过图形的方式说明了程序各个模块之间的关系。

第 4 章 实验设计与结果分析

4.1 实验设计

本章通过计算机实验系统地研究第三章提出的智能混合算法在求解带资源约束的大规模车辆路径问题的算法性能。实验的目的主要包括研究不同的混合策略以及相同混合策略下不同的算法参数对算法性能的影响。文章 4.1 节介绍实验的设计环节，包括实验数据的选择，算法的运行环境以及参数的选择；接下来的 4.2 节对实验的具体过程和结果进行了细致说明和分析。

4.1.1 实验数据

本文采用取自 TSPLIB 的三类算例进行试验，分别为 Christofides、Li、Goledn。TSPLIB 是一个实例库，它是世界上被使用最广泛的实例库之一，由 G.Reinelt 在 1991 年提出。[32] 其中有来自不同资源的不同类型的 VRP 实例。本文的实例取自网站 TSPLIB，其中包含适用于各种 TSP 相关问题的实例。与程序适配的所有问题均可从网站 https://sites.google.com/site/vrphlibrary/benchmark-por 下载。问题的点数分布从 50 至 1200，所有问题使用的距离均为欧几里得距离。表 A1 给出了测试问题的特征信息。

本文根据客户点数对算例进行了分类。共分为三类：C1 为客户点数 50-200，包含 14 个算例，其中六个有距离约束。C2 为客户点数 241-484，包含 20 个算例，其中八个有距离约束。C3 为客户点数 560-1200，包含十二个算例，均有距离约束。

4.1.2 实验环境

实验运行环境为 Intel®Core™i5-2410MCPU@2.30GHz/4000M RAM PC。操作系统 Windows 7 Ultimate 64 位。所有实验程序均有 C++语言编写。

4.1.3 实验参数

本文所使用的智能混合搜索算法以模拟退火法为基础，将禁忌搜索嵌入局部搜索。故需要进行一些基本的参数设置。本节通过一些基本实验实现最优参数组合。智能混合算法需要设置的参数有：

T - 模拟退火法初始温度

r - 退火率

N - 局部搜索时每个点邻域的规模

D - 控制在 diversification 阶段的主循环规模。

K - 尝试突破局部最小值的次数。

C - 禁忌搜索中候选解的个数

L - 禁忌长度

本算法根据来自文献[33]的经验性结论，当退火率在 0.8-0.99 时产生的结果较好，且该

值越接近 1 越好。故本实验将退火率设置为 0.99。经过数次参数调整实验，发现当 D=30，K=5，N=10 时可以在合理地时间内得到优质的解。根据文献[31]，禁忌搜索候选解个数为 5 时产生的解较理想。禁忌表的长度在[5,10]中随机进行选择。模拟退火法算法中的初始温度将通过实验确定。

4.2 实验和结果

下面分别对模拟退火法和智能混合算法进行测试。不同算法的迭代方式不同。初始解由 ClarkWright 从三个[0.5,2]内随机选择的 λ 值生成。目标函数值为所有车辆行驶的总距离。各算法的其他参数在求解的各自部分说明。

4.2.1 实验

本实验分别对 C1 组，C2 组和 C3 组在不同温度下进行测试。得到的结果如表 A2、A3、A4 所示（见附录），我们将图 A1、A2、A3 中有距离约束的实例标记为红色，没有距离约束的实例标记为蓝色。这样，我们从表中数据和对图的观察可以得出以下结论：

（1） 从表 A2 及图 A1 可以看出。当客户点数较小时，温度的变化对解的质量基本没有影响。且没有距离约束的情况下算例的解的质量要优于距离约束存在时的算例。

（2） 根据表 A3，从同一温度下不同算例的均值来看，当温度在 2 左右时解的质量最好，而当温度不断升高后，解的质量也随之下降。而从图 A2 上可以看出，有距离约束的算例 1,8,4,3,2,6,7,5 随着温度的升高解的质量逐渐变好并趋于稳定，而其他没有距离约束的算例，随着温度的升高解的质量逐渐变坏。而温度 2,3 便是有距离约束和无距离约束的两类算例解的质量上升和下降的交点，所以平均来看在此处解的质量最好。

（3） 由图 A3 可以看出，随着温度的增加，解的质量在改善并趋于稳定。并且，当温度大于 5 后解的均值趋于稳定。

因此我们根据上述观察对模拟退火机制进行了改进，引入了变温机制使得算法对距离约束有更好的适应性。具体来说，当没有距离约束时，温度为 2，当有距离约束时，温度为 6。数据如表 4.1。

表 4.1 引入变温机制后的结果

Name	Best	Name	Best	Name	Best
Christofied_01	1	Golden_01	1.03	Li_21	1.026
Christofied_02	1.005	Golden_02	1.01	Li_22	1.025
Christofied_03	1.002	Golden_03	1.015	Li_23	1.027
Christofied_04	1.007	Golden_04	1.022	Li_24	1.01
Christofied_05	1.016	Golden_05	1.03	Li_25	1.027
Christofied_06	0.945	Golden_06	1.021	Li_26	1.011
Christofied_07	0.919	Golden_07	1.013	Li_27	1.03
Christofied_08	0.955	Golden_08	1.022	Li_28	1.042
Christofied_09	0.892	Golden_09	1.029	Li_29	1.031
Christofied_10	0.95	Golden_10	1.031	Li_30	1.052
Christofied_11	1	Golden_11	1.034	Li_31	1.04
Christofied_12	1	Golden_12	1.037	Li_32	1.021
Christofied_13	0.677	Golden_13	1.021	AVG	1.0285
Christofied_14	0.946	Golden_14	1.024		
Avg	0.951	Golden_15	1.021		
		Golden_16	1.025		
		Golden_17	1.013		
		Golden_18	1.026		
		Golden_19	1.019		
		Golden_20	1.019		
		Avg	1.0231		

将上述数据与恒温下的最好数据进行对比，可以发现对于 C1 组的算例，恒温组的平均最优解出现在温度为 2 时，平均最优解为 0.951571。而此时变温组的最优解为 0.951。对于 C2 组的算例，恒温组的平均最优解出现在温度为 2 时，为 1.031。而变温组的平均最优解为 1.0231。对于 C3 组的算例，由于该组算例均有距离约束所以平均最优解相同。

上述分析说明变温机制的引入可以改进解的质量。至此参数设置完毕。

4.2.2 实验结果

本算法包含三个随机因素：1）使用 ClarkWright 法创建初始解时 λ 的选择。2）产生候选解后在候选解中随机进行选择，并在此解的基础上进行迭代。3）禁忌表长的选择。由于包含随机因素，故本次实验对每组数据进行十次运算后取平均值。实验数据见表 A6、A7、A8（见附录）。此外，本文在附录中列出分别列出客户点数为 50、100、200、400、800、1000 时，由混合算法产生的解的图像。

本文智能混合算法与单纯的模拟退火法在时间与解的质量的比较如图 4.1-4.6 及表 4.2、4.3、4.4。

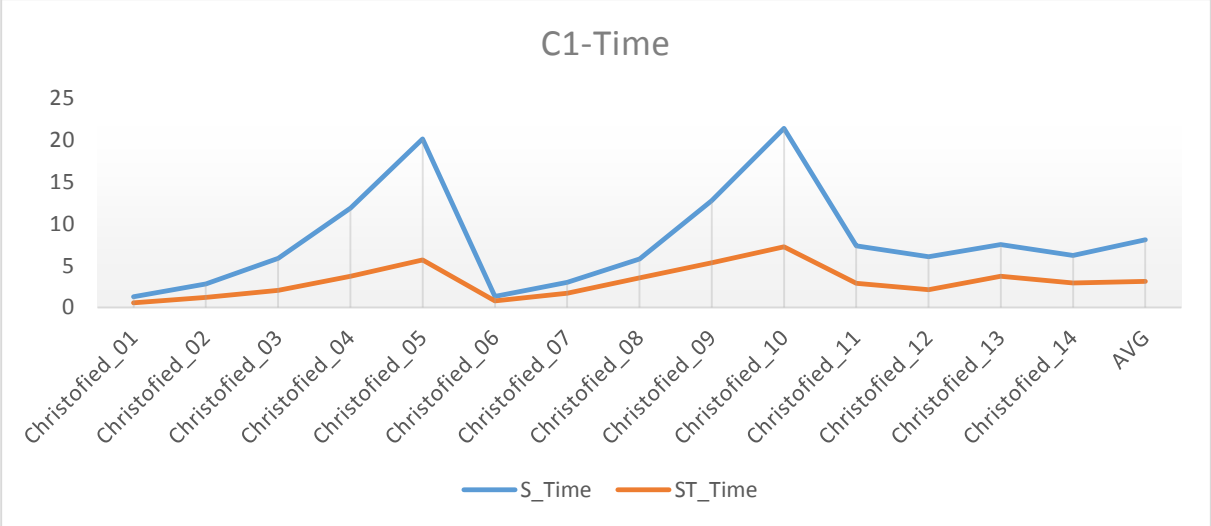


图 4.1 C1 组混合算法与模拟退火法在时间上的比较

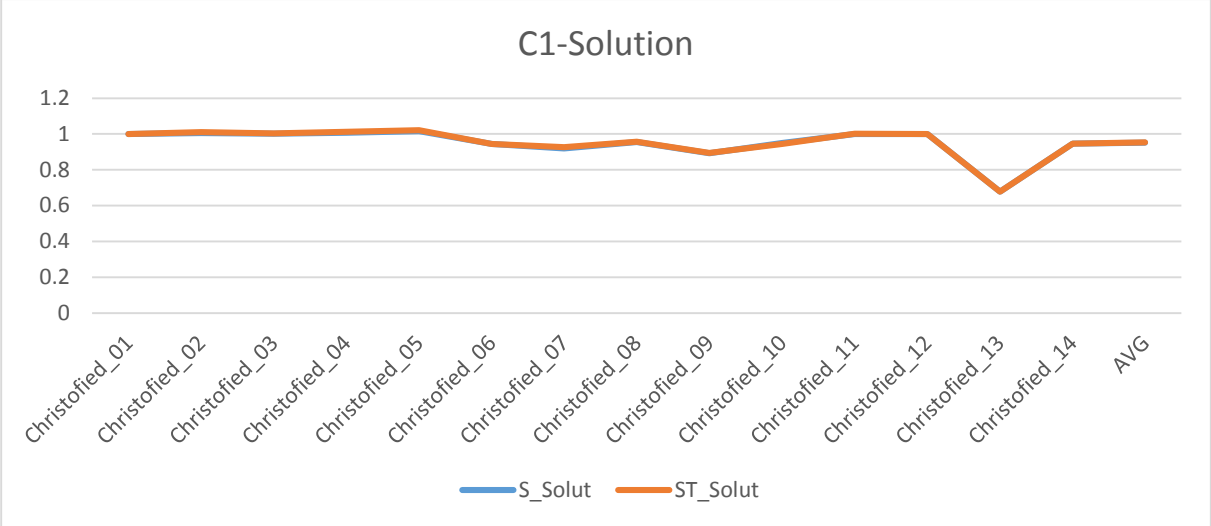


图 4.2 C1 组混合算法与模拟退火法解的质量上的比较

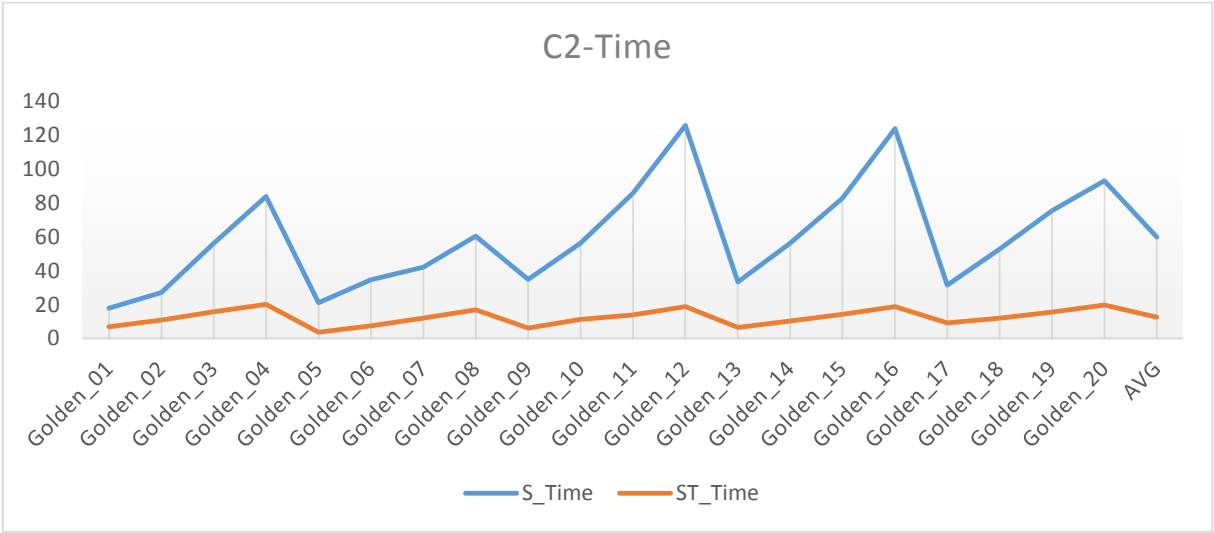


图 4.3 C2 组混合算法与模拟退火法在时间上的比较

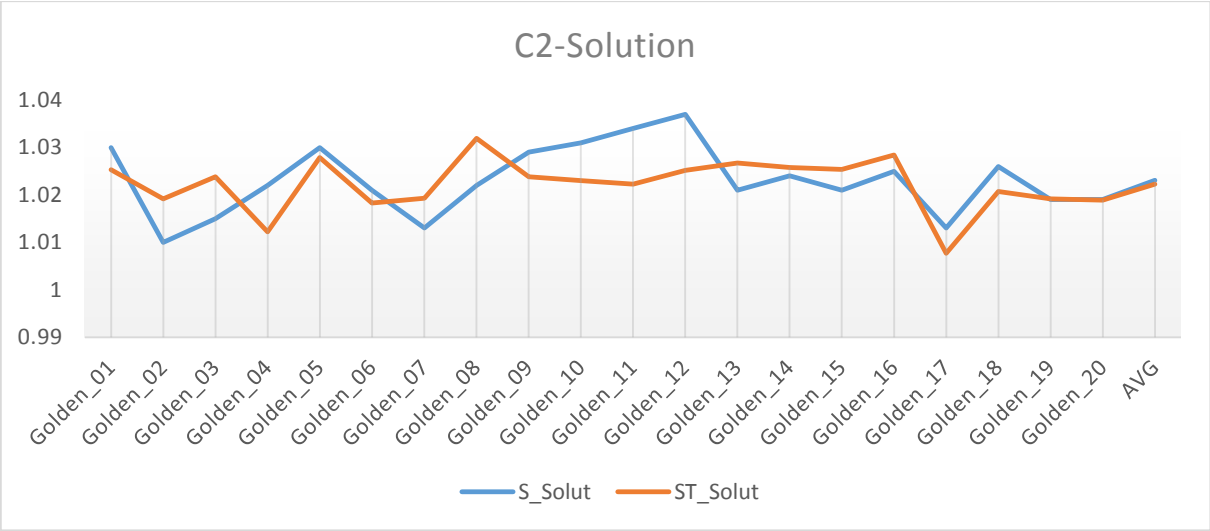


图 4.4 C2 组混合算法与模拟退火法在解的质量上的比较

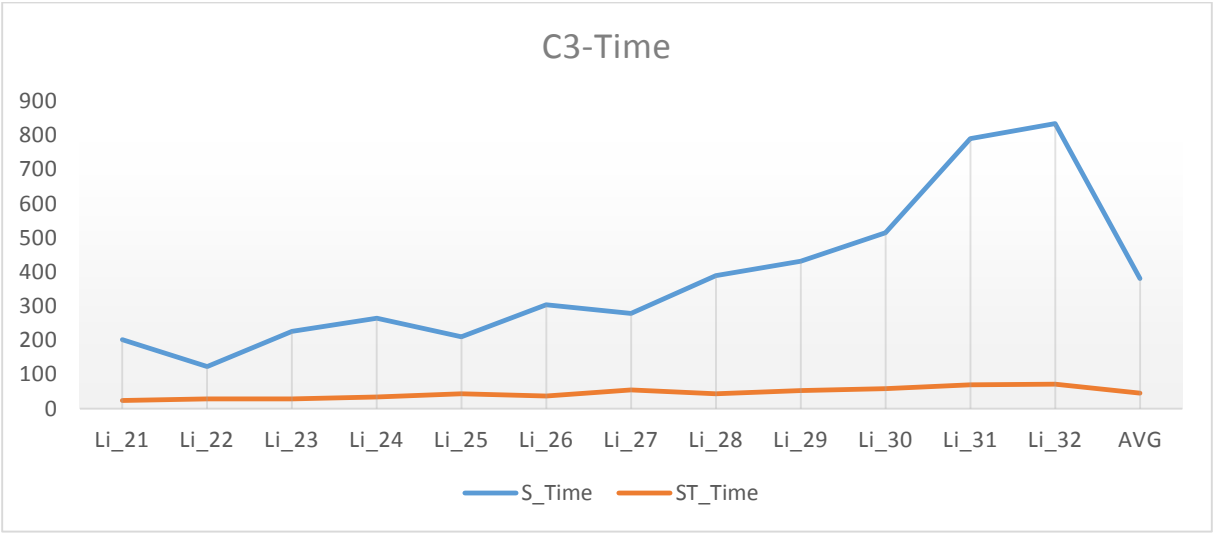


图 4.5 C3 组混合算法与模拟退火法在时间上的比较

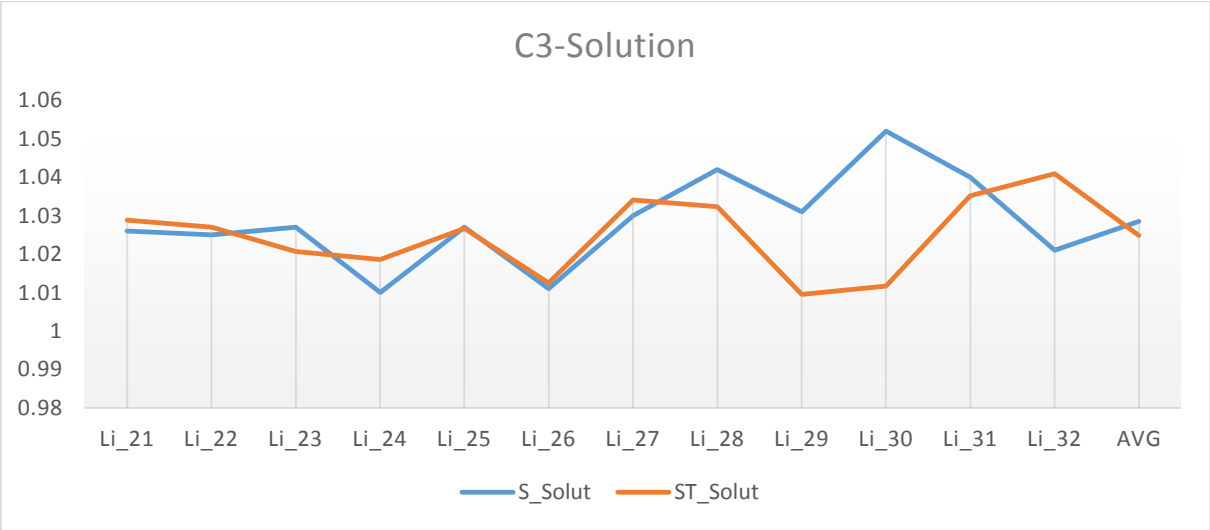


图 4.6 C3 组混合算法与模拟退火法在解的质量上的比较

表 4.2 C1 组混合算法与模拟退火法的比较

Name	S_Time	ST_Time	S_Solut	ST_Solut
Christofied_01	1.29	0.572	1	1
Christofied_02	2.84	1.219	1.005	1.0102
Christofied_03	5.9	2.07	1.002	1.0035
Christofied_04	11.87	3.741	1.007	1.0123
Christofied_05	20.14	5.672	1.016	1.0211
Christofied_06	1.33	0.806	0.945	0.945
Christofied_07	3.01	1.732	0.919	0.9259
Christofied_08	5.82	3.534	0.955	0.9565
Christofied_09	12.74	5.352	0.892	0.8941
Christofied_10	21.4	7.249	0.95	0.9435
Christofied_11	7.38	2.897	1	1.0012
Christofied_12	6.08	2.131	1	1
Christofied_13	7.53	3.755	0.677	0.6772
Christofied_14	6.21	2.958	0.946	0.946
AVG	8.11	3.120571	0.951	0.952607
DEV			7.67E-03	7.78E-03

表 4.3 C2 组混合算法与模拟退火法的比较

Name	S_Time	ST_Time	S_Solut	ST_Solut
Golden_01	17.85	6.9	1.03	1.0253
Golden_02	27.1	10.858	1.01	1.0192
Golden_03	56.19	15.861	1.015	1.0238
Golden_04	83.86	20.032	1.022	1.0122
Golden_05	21.03	3.719	1.03	1.0279
Golden_06	34.69	7.411	1.021	1.0183
Golden_07	42.07	11.949	1.013	1.0193
Golden_08	60.33	16.99	1.022	1.0319
Golden_09	34.82	6.151	1.029	1.0238
Golden_10	56.19	11.314	1.031	1.023
Golden_11	85.75	13.881	1.034	1.0223
Golden_12	125.74	18.725	1.037	1.0252
Golden_13	33.32	6.469	1.021	1.0267
Golden_14	56.27	10.242	1.024	1.0258
Golden_15	82.85	14.22	1.021	1.0254
Golden_16	123.96	18.847	1.025	1.0284
Golden_17	31.51	9.106	1.013	1.0077
Golden_18	52.77	12.01	1.026	1.0207
Golden_19	75.44	15.661	1.019	1.0192
Golden_20	93.04	19.765	1.019	1.0189
AVG	59.739	12.50555	1.0231	1.02225
DEV			5.30421E-05	3.15E-05

表 4.4 C3 组混合算法与模拟退火法的比较

Name	S_Time	ST_Time	S_Solut	ST_Solut
Li_21	201.66	23.565	1.026	1.0288
Li_22	123.16	28.944	1.025	1.027
Li_23	226.25	28.381	1.027	1.0207
Li_24	264.61	33.961	1.01	1.0186
Li_25	210.43	43.638	1.027	1.0267
Li_26	304.37	36.879	1.011	1.0125
Li_27	278.66	55.156	1.03	1.0341
Li_28	389.81	43.462	1.042	1.0323
Li_29	431.7	52.484	1.031	1.0095
Li_30	515.01	58.188	1.052	1.0117
Li_31	790.74	69.509	1.04	1.0352
Li_32	834.42	71.775	1.021	1.0409
AVG	380.90167	45.49517	1.0285	1.024833
DEV			1.46E-04	1.04E-04

从上面的图像与数据中我们可以得出以下几个结论：

（1） 从图 4.1-4.6 我们可以看出，在解的质量方面，智能混合算法与模拟退火法的差距虽然很小，但是在求解时间方面，智能混合算法要远远优于模拟退火法算法。尤其当客户规模较大时，改进算法在求解时间方面的优势十分明显。

（2） 通过对表 4.2-4.4 中方差的比较可以发现智能混合算法产生的解的稳定性更好。在 C1 组中，模拟退火算法的方差为 7.67E-03，混合算法为 7.78E-03，基本一致。在 C2 组中，模拟退火算法为 5.30421E-05，混合算法为 3.15E-05，大大优于模拟退火法。在 C3 组中，模拟退火法为 1.46E-04，混合算法为 1.04E-04。稳定性提高的原因可能与采用变温策略有关。

（3） 从表 4.2-4.4 的结果可以看出，随着客户规模的增大，两种算法的求解时间均有所上升，解的质量有所下降。但是，在时间方面，混合算法求解时间的增加要远远小于模拟退火法；在解的质量方面，混合算法的解质量的下降程度要小于模拟退火法。

4.3 本章小结

在物流配送日益重要的今天，作为该问题最基本的模型，CVRP 的研究是解决这类问题的基础。本章首先对 CVRP 的提出和数学模型进行了说明，接着提出了一种智能混合算法，并在实验的基础上对其中的某些参数进行了调整与设置。之后通过与相同条件下由模拟退火法产生的结果的比较，验证了算法的有效性与稳定性。此外，在参数设置阶段发现了距离约束与模拟退火机制中温度的关系。从结果来看，混合算法在解的质量方面并没有大的突破，这可能与局部搜索部分没有涉及对提高解的质量的优化有关。因此未来的研究工作将进一步研究如何改善解的质量。并将通过大规模的实际应用问题来改善算法的求解性能。

第 5 章 总结与展望

随着全球经济一体化，国家贸易的更深入开展，国与国、区域与区域之间的各种货物往来会越来越多，物流行业的重要性越来越突出，而 VRP 问题作为物流行业调度环节的关键一环必须要解决好。在物流配送系统中，合理安排车辆路线是减少浪费、提高经济效益的重要手段，然而由于本身问题的 NP 完全性质，精确求解非常困难，研究启发式算法不失为一种可行的方向。通过对车辆路径问题的分析，本文首先给出了经典 VRP 问题的数学模型，现实生活中大多数 VRP 都可以由这个基本模型进行衍生。接着本文主要基于模拟退火算法的启发式算法求解了三种规模的 VRP 算例，并根据所得数据对算法进行了优化，将禁忌搜索与模拟退火法进行了混合。本文的创新之处有以下几点：一、在模拟退火算法中，通过数据分析发现是否有距离约束会在不同的温度下有不同的反应，所以对模拟退火进行了变温改进。二、通过采用更智能的迭代方式，并将模拟退火法与禁忌搜索进行混合，提高了搜索效率，在不降低解的质量的基础上大大减少了求解时间。本文的特色在于对算法使用 C++程序的实现，并绘制出解的路径图，使其具有更强的实际意义。

目前，模拟退火算法已经在很多领域得到了应用，结合其他算法的混合算法更具有应用性，随着 VRP 问题的深入研究,相信很多问题都可以基于模拟退火的启发式算法来进行求解。

参考文献

[1] 宋燕子.[基于模拟退火法的启发式算法在VRP中的应用].武汉：华中师范大学，2013.

[2] O.Braysy and M.Gendreau.Vehiele routing Problem with time windows, Partl: Route construction and local search algorithms.Transportation Science, 39(l): 104-118, 2005.

[3] T.G.Grainic and G.Laporte, Planning models for freight transportation. European Journal of Operational Reasearch, 97:409-438, 1997.

[4] J.Larsen.Parallelization of the Vehicel Routing Problem with Time window. PhD thesis, Technical University of Denmark, 1999.

[5] G.Dantzig and J.Ramser. The truck dispatching problem. Management Science, 6:80-91, 1959.

[6] S.Ropke, J.-F.Cordeau, M.Iori and D.Vigo, “Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem with Two-Dimensional Loading Constraints.” Proceedings of ROUTE, Jekyll Island, 13 May 2007.

[7] A.G.Qureshi, E.Taniguchi and T.Yamada, “An Exact Solution Approach for Vehicle Routing and Scheduling Problems with Soft Time Windows,” Transprotation Research Part E: Logistics and Transportation Review, Vol.45, No.6, 2009, pp. 960-977. doi:10.1016/j.tre.2009.04.007.

[8] G.Gutiérrez-Jarpa, G.Desaulniers, G.Laporte and V.Ma- rianov, “A Branch-and-Price Algorithm for the Vehicle Routing Problem with Deliveries, Selective Pickups and Time Windows,” European Journal of Operational Re- search, Vol. 206, No.12, 2010, pp. 341-349.

[9] N.Azi, M.Gendreau and J.-Y.Potvin, “An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles,” European Journal of Operational Research, Vol.202, No.3, 2010, pp.756-763.

[10] Gilbert Laporte, Paolo Toth, Daniele Vigo, “Vehicle routing: historical perspective and recent contributions” EURO J Transp Logist (2013) 2:1–4.

[11] R. Montemanni, L. M. Gambardella, A. E. Rizzoli and A. V. Donati, “Ant Colony System for a Dynamic Vehicle Routing Pronlem,” Journal of Combinatorial Optimization, Vol. 10, No. 4, 2005, pp. 327-343.

[12] Y.-J. Cho and S.-D. Wang, “A Threshold Accepting Meta- Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows,” Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, 2005, pp. 3022-3037.

[13] M. Gandreau, F. Guertin, J.-Y. Potvin and R. Seguin, “Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-Ups and Deliveries,” Transportation Research Part E: Logistics and Transportation Review, Vol. 14, No. 3, 2006, pp. 157-174.

[14] N. Yuichi and B. Olli, “A Powerful Route Minimization Heuristic for the Vehicle Routing Problem with Time Windows,” Operations Research Letters, Vol. 37, No. 5, 2009, pp. 333-338.

[15] G.-N. Abel and Bullinaria John A, “An Improved Multi-Objective Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows,” Computers & Operations Research, Vol. 38, No. 1, 2011, pp. 287-300.

[16] K.-W. Peng, “An Adaptive Parallel Route Construction Heuristic for the Vehicle Routing Problem with Time Windows Constraints,” Expert System with Applications, Vol. 38, No. 9, 2011, pp.11939-11946.

[17] S. R. Balseiro, I. Loiseau and J. Ramone, “An Ant Col-ony Algorithm Hybridized with Insertion Heuristics for the Time Dependent Vehicle Routing Problem with Time Windows,” Computers & Operations Research, Vol.38, 2011, pp. 954-966.

[18] 肖朋等.车辆路径问题的单亲遗传算法[J].计算机技术与自动化,2000.

[19] 李军,谢秉晶,郭耀煌.非满载车辆调度问题的遗传算法[M].系统工程理论方法应用,2000,9(3): 235-239.

[20] 袁健,刘晋.随机需求情形 VRP 的 Hopfield 神经网络解法[J].南京航空航大大学学报,2000,32 (5): 579-585.

[21] Zhu Chongjun, Liu Min’Wu Cheng. The effect of fuzzy demand possibility distribution and confidence level to the VRP solution. The First International Conference on Mechanical Engineering. ICME 2000.

[22] 刘浩,袁健,卢厚清.两种类型车辆随机需求路由问题[J].南京航空航天大学学报,2001,33(2).

[23] 张丽萍,柴跃廷,曹瑞.有时间窗车辆路径问题的改进遗传算法[J].计算机集成制造系统——CIMS.2002, 8 (6): 451-454.

[24] 郎茂祥.用单亲遗传算法求解配送车辆调度问题的研究[J].交通与计算机,2006,128 (24): 119-122.

[25] 崔雪丽,马良,范炳全.车辆路径问题(VRP)的蚂蚁搜索算法[J].系统工程学报,2004,19 (4): 418-422.

[26] 刘云忠,宣慧玉.蚂蚁算法在车辆路径问题中的应用研究[J].信息与控制,2004,33 (2): 249-252.

[27] 张建勇,李军.模糊车辆路径问题的一种混合遗传算法[J].管理工程学报,2005,19 (2): 23-26.

[28] 崔雪丽,朱振刚.VRP 的混合改进型蚂蚁算法求解[J].苏州科技学院学报(工程技术版),2009,4(22): 62-66.

[29] Yu Bin and Yang Zhong Zhen, “An Ant Colony Optimization Model: The Period Vehicle Routing Problem with Time Windows,” Transportation Research Part E, Vol. 47, 2011, pp. 166-181.

[30] T.Stutzle. Local Search Algorithms for Combinatorial Problem - Analysis, Improvement New Applications. Infix, Augustin, 1999.

[31] Jean-Francois Cordeau, Gilbert Laporte. “Tabu Search Heuristics for the Vehicle Routing Problem,” Canada Research Chair in Distribution Management and GERAD, Canada H3T 2A7.

[32] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. ORSA Journal on Computing, 3:376－384, 1991.

[33] Dimitris Bertsimas and John Tsitsiklis. “Simulated Annealing,” Statistical Science, Vol. 8, No. 1, 10-15, 1993.

[34] 王沛栋,唐功友,李扬. 带容量约束车辆路由问题的改进蚁群算法-控制与决策,2012.

附录 A 实验结果相关表格与图像

表 A1 算例特征信息

Name	Dimension	Capacity	Distance	Name	Dimension	Capacity	Distance	Name	Dimension	Capacity	Distance
Christofied_01	51	160		Golden_01	241	550	650	Li_21	561	1200	1800
Christofied_02	76	140		Golden_02	321	700	900	Li_22	601	900	1000
Christofied_03	101	200		Golden_03	401	900	1200	Li_23	641	1400	2200
Christofied_04	151	200		Golden_04	481	1000	1600	Li_24	721	1500	2400
Christofied_05	200	200		Golden_05	201	900	1800	Li_25	761	900	900
Christofied_06	51	160	200	Golden_06	281	900	1500	Li_26	801	1700	2500
Christofied_07	76	140	160	Golden_07	361	900	1300	Li_27	841	900	900
Christofied_08	101	200	230	Golden_08	441	900	1200	Li_28	881	1800	2800
Christofied_09	151	200	200	Golden_09	256	1000		Li_29	961	2000	3000
Christofied_10	200	200	200	Golden_10	324	1000		Li_30	687.9	2100	3200
Christofied_11	121	200		Golden_11	400	1000		Li_31	1121	2300	3500
Christofied_12	101	200		Golden_12	484	1000		Li_32	1201	2500	3600
Christofied_13	121	200	720	Golden_13	253	1000					
Christofied_14	101	200	1040	Golden_14	321	1000					
				Golden_15	397	1000					
				Golden_16	481	1000					
				Golden_17	241	200					
				Golden_18	301	200					
				Golden_19	361	200					
				Golden_20	421	200					

表 A2 C1 组变温实验

Name	Best_1	Best_2	Best_3	Best_4	Best_5	Best_6	Best_7	Best_8	Best_9	Best_10
Chris_01	1.031	1	1	1	1	1	1	1	1	1
Chris_02	1.031	1.005	1.017	1.008	1.011	1.001	1.003	1.003	1.013	1.003
Chris_03	1.005	1.002	1.002	1.002	1.005	1.006	1.008	1.009	1.011	1.007
Chris_04	1.028	1.007	1.013	1.014	1.006	1.009	1.008	1.016	1.017	1.025
Chris_05	1.015	1.016	1.018	1.022	1.021	1.023	1.019	1.036	1.046	1.047
Chris_06	0.974	0.945	0.945	0.945	0.951	0.945	0.945	0.945	0.945	0.945
Chris_07	0.933	0.929	0.921	0.932	0.919	0.919	0.925	0.926	0.924	0.92
Chris_08	0.958	0.958	0.958	0.954	0.959	0.955	0.959	0.963	0.967	0.962
Chris_09	0.903	0.898	0.893	0.894	0.892	0.892	0.905	0.897	0.905	0.914
Chris_10	0.947	0.94	0.944	0.947	0.952	0.95	0.952	0.952	0.962	0.97
Chris_11	1.005	1	1	1.001	1.001	1.001	1.003	1.004	1.005	1.004
Chris_12	1	1	1	1	1.001	1	1	1	1.003	1
Chris_13	0.68	0.676	0.676	0.677	0.677	0.677	0.678	0.679	0.679	0.679
Chris_14	0.946	0.946	0.946	0.946	0.947	0.946	0.946	0.946	0.949	0.946
AVG	0.964	0.951	0.952	0.953	0.953	0.951	0.953	0.955	0.959	0.958

表 A3 C2 组变温实验

Name	Best_1	Best_2	Best_3	Best_4	Best_5	Best_6	Best_7	Best_8	Best_9	Best_10
Golden_01	1.027	1.026	1.017	1.015	1.015	1.03	1.024	1.018	1.034	1.034
Golden_02	1.04	1.042	1.018	1.013	1.011	1.01	1.023	1.011	1.012	1.012
Golden_03	1.059	1.047	1.025	1.029	1.011	1.015	1.014	1.006	1.012	1.011
Golden_04	1.092	1.056	1.026	1.011	1.017	1.022	1.03	1.013	1.013	1.015
Golden_05	1.066	1.033	1.03	1.031	1.001	1.03	1.021	1.03	1	1.001
Golden_06	1.042	1.041	1.021	1.036	1.021	1.021	1.007	1.021	1.021	1.007
Golden_07	1.055	1.036	1.029	1.016	1.005	1.013	1.025	1.016	1.023	1.002
Golden_08	1.076	1.049	1.037	1.029	1.026	1.022	1.088	1.022	1.025	1.023
Golden_09	1.023	1.029	1.056	1.063	1.079	1.097	1.085	1.094	1.085	1.085
Golden_10	1.021	1.031	1.055	1.077	1.098	1.111	1.109	1.089	1.103	1.114
Golden_11	1.016	1.034	1.055	1.089	1.079	1.084	1.109	1.11	1.104	1.095
Golden_12	1.024	1.037	1.063	1.09	1.105	1.114	1.072	1.122	1.127	1.118
Golden_13	1.024	1.021	1.034	1.041	1.058	1.067	1.084	1.075	1.07	1.071
Golden_14	1.016	1.024	1.034	1.055	1.065	1.07	1.093	1.085	1.087	1.081
Golden_15	1.02	1.021	1.028	1.047	1.062	1.087	1.095	1.09	1.095	1.086
Golden_16	1.019	1.025	1.034	1.06	1.089	1.083	1.095	1.097	1.097	1.103
Golden_17	1.015	1.013	1.042	1.029	1.053	1.053	1.059	1.05	1.05	1.056
Golden_18	1.021	1.026	1.031	1.051	1.056	1.051	1.054	1.069	1.056	1.058
Golden_19	1.014	1.019	1.025	1.043	1.047	1.054	1.058	1.06	1.063	1.064
Golden_20	1.015	1.019	1.021	1.026	1.041	1.046	1.047	1.054	1.051	1.065
AVG	1.03425	1.03145	1.03405	1.04255	1.04695	1.054	1.0596	1.0566	1.0564	1.05505

表 A4 C3 组变温实验

Name	Best_1	Best_2	Best_3	Best_4	Best_5	Best_6	Best_7	Best_8	Best_9	Best_10
Li_21	1.079	1.062	1.029	1.028	1.019	1.026	1.027	1.018	1.027	1.027
Li_22	1.034	1.023	1.025	1.026	1.023	1.025	1.024	1.024	1.027	1.027
Li_23	1.062	1.041	1.031	1.031	1.018	1.027	1.02	1.027	1.02	1.027
Li_24	1.072	1.04	1.015	1.003	1.014	1.01	1.013	1.028	1.028	1.007
Li_25	1.029	1.027	1.025	1.027	1.027	1.027	1.025	1.043	1.032	1.041
Li_26	1.112	1.079	1.053	1.028	1.012	1.011	1.011	1.006	1.007	1.012
Li_27	1.052	1.034	1.027	1.026	1.028	1.03	1.03	1.036	1.043	1.039
Li_28	1.087	1.074	1.034	1.03	1.03	1.042	1.031	1.01	1.027	1.031
Li_29	1.085	1.042	1.044	1.032	1.03	1.031	1.03	1.03	1.03	1.03
Li_30	1.123	1.09	1.099	1.076	1.035	1.052	1.031	1.031	1.031	1.03
Li_31	1.104	1.033	1.041	1.028	1.043	1.04	1.012	1.031	1.034	1.034
Li_32	1.097	1.063	1.056	1.046	1.048	1.021	1.062	1.043	1.042	1.045
AVG	1.078	1.050667	1.039917	1.03175	1.02725	1.0285	1.026333	1.02725	1.029	1.029167

表 A6 C1 组混合算法结果

Name/Tempra	1	2	3	4	5	6	7	8	9	10	AVG
Christofied_01	1	1	1	1	1	1	1	1	1	1	1
Time	0.53	0.61	0.56	0.53	0.58	0.51	0.61	0.64	0.51	0.64	0.572
Christofied_02	1.01	1.012	1.013	1.014	1.016	1.009	1.004	1.01	1.001	1.013	1.0102
Time	1.22	1.2	1.31	1.28	1.42	1.03	1.16	0.94	1.37	1.26	1.219
Christofied_03	1.003	1.003	1.002	1.002	1.004	1.004	1.004	1.004	1.006	1.003	1.0035
Time	2.17	2.18	2.15	1.97	2.18	1.81	2.11	2.37	1.7	2.06	2.07
Christofied_04	1.012	1.012	1.014	1.008	1.015	1.01	1.016	1.017	1.007	1.012	1.0123
Time	3.28	3.84	3.84	4.1	3.77	3.65	3.4	3.81	4.01	3.71	3.741
Christofied_05	1.019	1.02	1.019	1.017	1.021	1.024	1.023	1.02	1.023	1.025	1.0211
Time	4.96	4.48	5.8	5.27	6.18	6.38	5.82	5.49	6.38	5.96	5.672
Christofied_06	0.945	0.945	0.945	0.945	0.945	0.945	0.945	0.945	0.945	0.945	0.945
Time	0.95	0.81	0.76	0.76	0.81	0.89	0.8	0.94	0.72	0.62	0.806
Christofied_07	0.919	0.92	0.933	0.929	0.919	0.935	0.927	0.924	0.934	0.919	0.9259
Time	1.81	1.78	1.64	1.7	1.9	1.65	1.9	1.73	1.37	1.84	1.732
Christofied_08	0.958	0.955	0.958	0.958	0.954	0.955	0.955	0.958	0.959	0.955	0.9565
Time	3.6	3.45	3.59	3.81	3.48	3.62	3.38	3.52	3.54	3.35	3.534
Christofied_09	0.896	0.894	0.897	0.892	0.895	0.897	0.892	0.893	0.894	0.891	0.8941
Time	5.43	5.54	6.08	5.19	4.84	5.21	5.72	5.1	5.12	5.29	5.352
Christofied_10	0.949	0.944	0.94	0.945	0.944	0.945	0.944	0.939	0.946	0.939	0.9435
Time	6.05	8.1	7.88	7.49	7.66	7.52	7.27	7.99	6.54	5.99	7.249
Christofied_11	1	1.004	1.001	1	1.001	1	1	1.004	1.001	1.001	1.0012
Time	2.54	2.81	3.04	2.9	3	3.23	2.78	2.85	3	2.82	2.897
Christofied_12	1	1	1	1	1	1	1	1	1	1	1
Time	2.26	1.95	2.35	2.4	2.11	2.03	1.97	2.37	1.87	2	2.131
Christofied_13	0.679	0.676	0.679	0.676	0.677	0.676	0.678	0.676	0.679	0.676	0.6772
Time	3.23	4.2	4.07	4.18	4.37	2.81	3.63	3.35	3.62	4.09	3.755
Christofied_14	0.946	0.946	0.946	0.946	0.946	0.946	0.946	0.946	0.946	0.946	0.946
Time	2.85	2.75	3.32	3.59	2.64	2.98	2.79	2.84	2.82	3	2.958

表 A7 C2 组混合算法

Name	1	2	3	4	5	6	7	8	9	10	AVG
Golden_01	1.028	1.03	1.015	1.027	1.018	1.033	1.025	1.028	1.022	1.027	1.0253
Time	7.44	7.19	6.55	6.46	7.91	7.13	6.1	6.63	8.02	5.57	6.9
Golden_02	1.019	1.013	1.013	1.017	1.014	1.03	1.016	1.017	1.031	1.022	1.0192
Time	10.44	10	10.53	10.69	12.67	13.12	10.53	10.27	9.89	10.44	10.858
Golden_03	1.013	1.026	1.027	1.015	1.022	1.029	1.03	1.02	1.024	1.032	1.0238
Time	15.96	15.15	16.96	12.84	16.82	18.77	16.4	16.6	13.35	15.76	15.861
Golden_04	1.015	1.011	1.014	1.017	1.012	1.011	1.015	1.008	1.007	1.012	1.0122
Time	20.51	20.17	21.18	20.62	18.94	19.75	19.59	19.53	18.78	21.25	20.032
Golden_05	1.031	1.031	1.002	1.029	1.03	1.031	1.032	1.031	1.031	1.031	1.0279
Time	4.17	4.07	3.35	4.07	3.23	2.93	3.38	3.88	4.07	4.04	3.719
Golden_06	1.021	1.022	1.021	1.007	1.021	1.007	1.021	1.021	1.021	1.021	1.0183
Time	7.29	7.55	7.63	8.06	9.28	7.72	6.43	7.05	7.33	5.77	7.411
Golden_07	1.021	1.019	1.027	1.007	1.015	1.016	1.019	1.025	1.021	1.023	1.0193
Time	12.62	11.26	13.26	11.36	11.95	11.78	12.48	12.32	11.65	10.81	11.949
Golden_08	1.032	1.026	1.038	1.028	1.038	1.031	1.034	1.036	1.03	1.026	1.0319
Time	15.21	17.99	18.72	17.6	17.29	15.01	17.49	18.38	14.38	17.83	16.99
Golden_09	1.021	1.029	1.026	1.024	1.02	1.027	1.022	1.023	1.022	1.024	1.0238
Time	7.19	5.54	5.45	7.18	5.69	5.19	6.96	5.46	5.83	7.02	6.151
Golden_10	1.023	1.024	1.023	1.029	1.02	1.023	1.024	1.02	1.022	1.022	1.023
Time	14.24	10.28	9.98	18.98	10.7	10.22	9.52	9.45	11.86	7.91	11.314
Golden_11	1.019	1.024	1.02	1.026	1.019	1.025	1.03	1.022	1.017	1.021	1.0223
Time	9.89	14.02	14.62	12.78	15.6	16.5	13.62	13.95	13.53	14.3	13.881
Golden_12	1.033	1.028	1.026	1.028	1.022	1.019	1.027	1.02	1.023	1.026	1.0252
Time	17.88	18.14	20.03	18.24	19.3	19.84	19.41	18.14	18	18.27	18.725
Golden_13	1.03	1.03	1.026	1.024	1.028	1.023	1.025	1.028	1.024	1.029	1.0267
Time	4.99	7.6	6.16	7.27	5.65	5.87	5.91	7.1	7.14	7	6.469
Golden_14	1.029	1.03	1.025	1.022	1.026	1.025	1.026	1.026	1.026	1.023	1.0258
Time	9.47	9.84	9.77	10.53	10.31	10.19	11.22	10.25	9.97	10.87	10.242
Golden_15	1.028	1.027	1.022	1.022	1.021	1.025	1.029	1.031	1.024	1.025	1.0254
Time	14.99	14.13	13.46	14.73	14.68	13.4	14.1	13.91	14.96	13.84	14.22

Golden_16	1.031	1.029	1.029	1.03	1.027	1.028	1.023	1.029	1.03	1.028	1.0284
Time	19.33	19.09	18.41	17.88	19.14	18.61	18.88	20.16	18.84	18.13	18.847
Golden_17	1.009	1.007	1.013	1.004	1.008	1.008	1.008	1.007	1.006	1.007	1.0077
Time	8.44	9.53	9.86	8.69	8.53	9.27	9.88	8.78	8.91	9.17	9.106
Golden_18	1.019	1.026	1.021	1.018	1.021	1.02	1.021	1.022	1.018	1.021	1.0207
Time	12.54	12.95	11.76	12.12	11.82	11.84	11.9	11.75	12.25	11.17	12.01
Golden_19	1.02	1.021	1.019	1.018	1.021	1.018	1.017	1.02	1.021	1.017	1.0192
Time	15.04	15.26	14.87	15.93	15.46	14.65	16.89	16.93	16.71	14.87	15.661
Golden_20	1.018	1.021	1.019	1.016	1.021	1.023	1.016	1.018	1.018	1.019	1.0189
Time	19.73	19.59	18.19	20.09	20.44	19.59	19.91	20.47	20.12	19.52	19.765

表 A8 C3 组混合算法

Name	1	2	3	4	5	6	7	8	9	10	AVG
Li_21	1.033	1.028	1.028	1.028	1.03	1.031	1.029	1.028	1.03	1.023	1.0288
Time	24.01	23.98	20.25	25.43	22.89	24.88	24.38	24.35	23.41	22.07	23.565
Li_22	1.028	1.028	1.026	1.028	1.03	1.027	1.024	1.025	1.027	1.027	1.027
Time	27.88	29.08	28.78	28.03	30.73	28.92	28.67	30.08	28.75	28.52	28.944
Li_23	1.023	1.027	1.017	1.021	1.021	1.027	1.018	1.022	1.013	1.018	1.0207
Time	26.79	28.88	27.05	27.11	32.26	29.38	28.56	25.63	30.4	27.75	28.381
Li_24	1.013	1.028	1.028	1.028	1.028	1.007	1.006	1.013	1.013	1.022	1.0186
Time	33.06	30.53	34.21	28.84	36.4	36.88	36.33	37.07	33.23	33.06	33.961
Li_25	1.027	1.025	1.028	1.027	1.029	1.025	1.026	1.028	1.025	1.027	1.0267
Time	43.62	38.49	43.12	42.54	46.75	43.91	44.46	48.72	42.31	42.46	43.638
Li_26	1.011	1.011	1.011	1.012	1.016	1.011	1.012	1.017	1.013	1.011	1.0125
Time	36.77	33.12	30.81	42.2	44.38	35.37	32.56	38.98	34.93	39.67	36.879
Li_27	1.033	1.033	1.028	1.032	1.034	1.034	1.044	1.029	1.043	1.031	1.0341
Time	51.85	61.37	56.6	55.47	53.52	54.3	55.54	56.78	49.22	56.91	55.156
Li_28	1.029	1.03	1.034	1.034	1.029	1.034	1.035	1.029	1.034	1.035	1.0323
Time	46.61	42.46	46.71	37.5	43.63	38.58	44.57	53.49	40.53	40.54	43.462
Li_29	1.009	1.009	1.009	1.009	1.014	1.009	1.009	1.009	1.004	1.014	1.0095
Time	48.58	56.18	48.41	46.18	57.99	48.66	61.01	57.35	47.56	52.92	52.484
Li_30	1.013	1.014	1.013	1.013	1.013	1.008	1.013	1.009	1.008	1.013	1.0117
Time	56.8	55.02	61.25	62.91	57.92	56.21	51.73	59.83	58.23	61.98	58.188
Li_31	1.037	1.033	1.038	1.035	1.036	1.033	1.037	1.033	1.037	1.033	1.0352
Time	78.69	68.05	69.34	64.05	64.35	74.85	73.34	64.23	65.66	72.53	69.509
Li_32	1.039	1.043	1.04	1.041	1.038	1.04	1.039	1.045	1.043	1.041	1.0409
Time	71.96	65.61	70.28	71.95	70.59	71.81	70.11	78	71.7	75.74	71.775

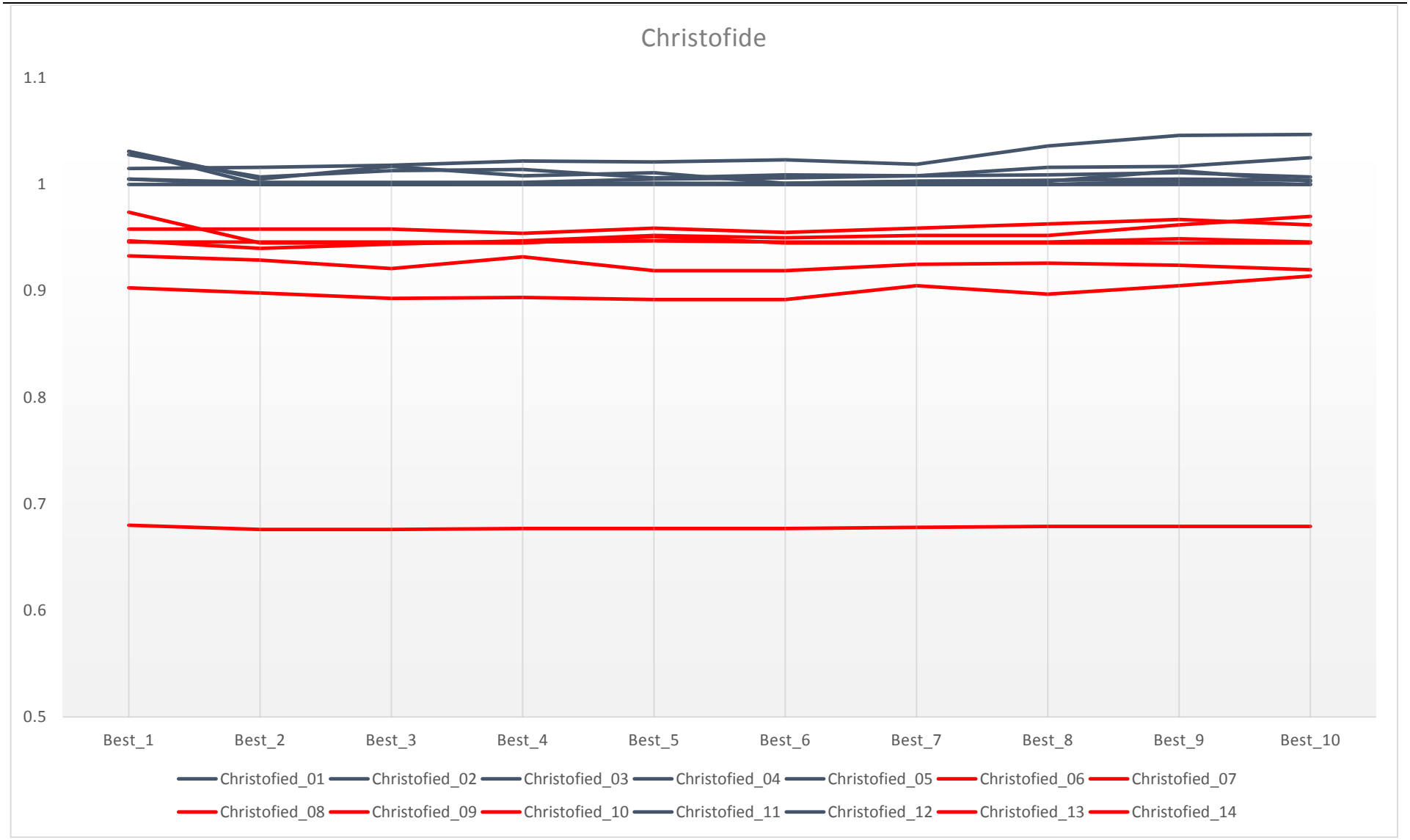

Christofide
1.1
1
0.9
0.8
0.7
0.6
0.5
Best_1
Best_2
Best_3
Best_4
Best_5
Best_6
Best_7
Best_8
Best_9
Best_10
Christofied_01
Christofied_02
Christofied_03
Christofied_04
Christofied_05
Christofied_06
Christofied_07
Christofied_08
Christofied_09
Christofied_10
Christofied_11
Christofied_12
Christofied_13
Christofied_14

图 A1 C1 组变温实验

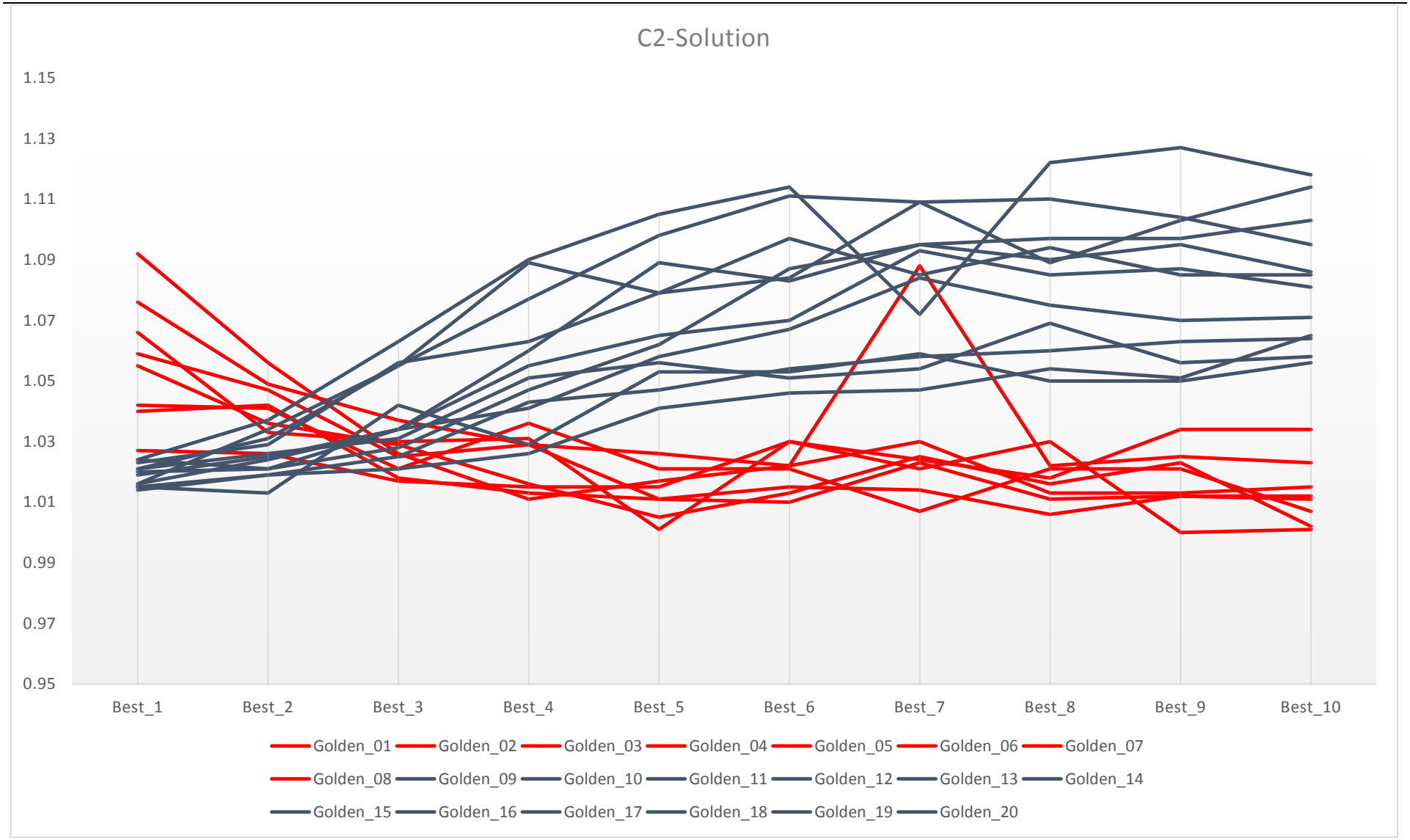

C2-Solution
1.15
1.13
1.11
1.09
1.07
1.05
1.03
1.01
0.99
0.97
0.95
Best_1
Best_2
Best_3
Best_4
Best_5
Best_6
Best_7
Best_8
Best_9
Best_10
Golden_01
Golden_02
Golden_03
Golden_04
Golden_05
Golden_06
Golden_07
Golden_08
Golden_09
Golden_10
Golden_11
Golden_12
Golden_13
Golden_14
Golden_15
Golden_16
Golden_17
Golden_18
Golden_19
Golden_20

图 A2 C2 组变温实验

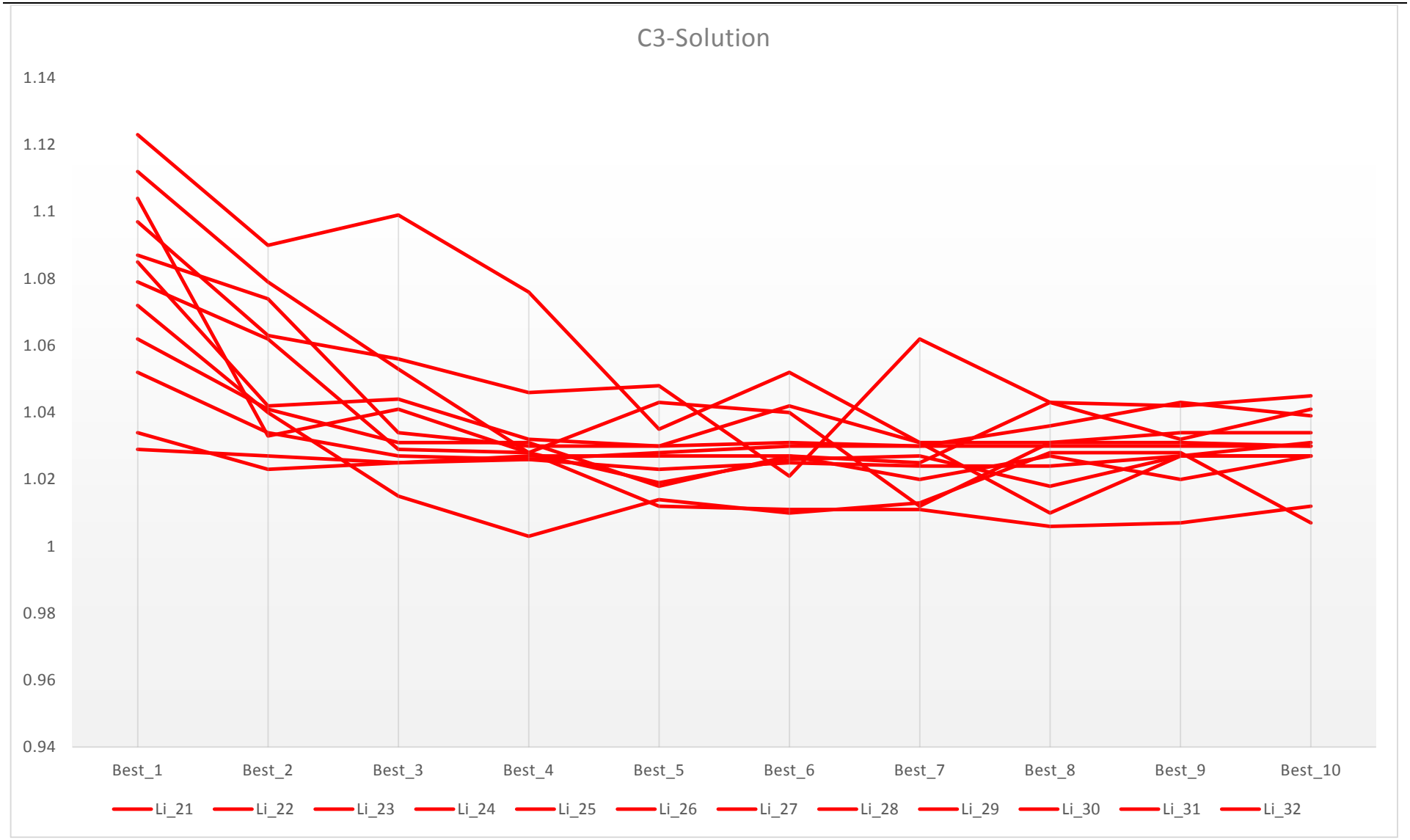

C3-Solution
1.14
1.12
1.1
1.08
1.06
1.04
1.02
1
0.98
0.96
0.94
Best_1
Best_2
Best_3
Best_4
Best_5
Best_6
Best_7
Best_8
Best_9
Best_10
Li_21
Li_22
Li_23
Li_24
Li_25
Li_26
Li_27
Li_28
Li_29
Li_30
Li_31
Li_32

图 A3 C3 组变温实验

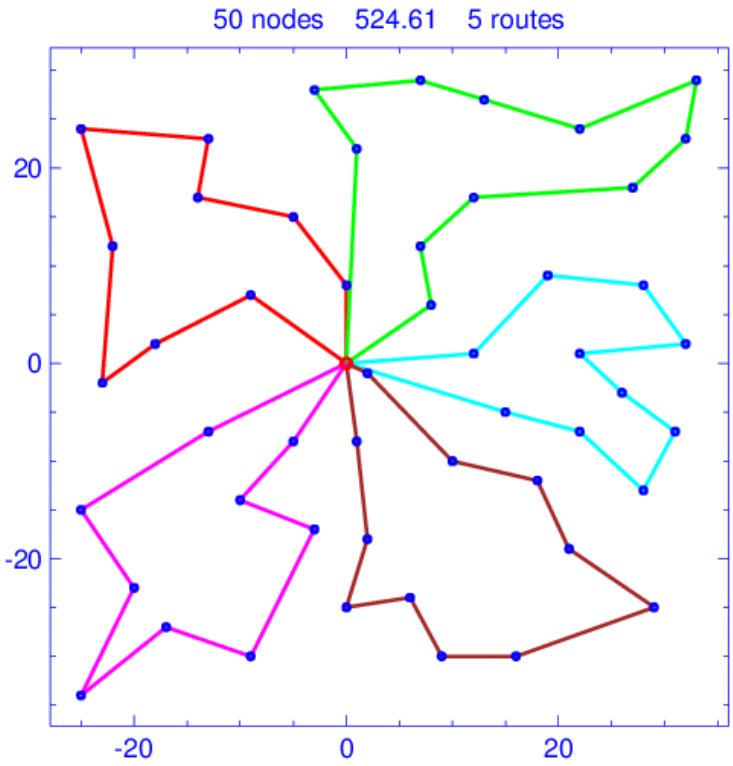


图 A5 50 个点路径图

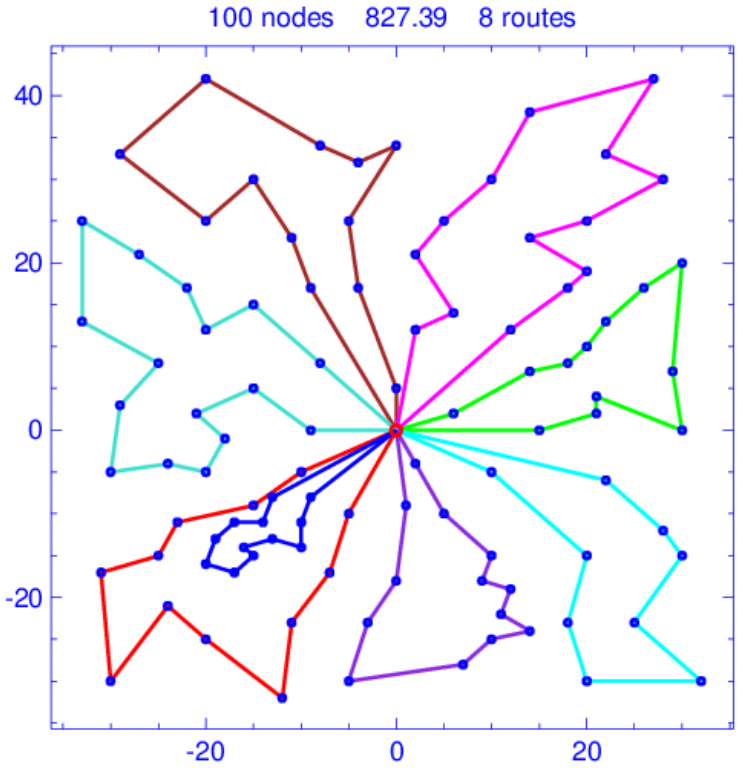


图 A6 100 个点路径图

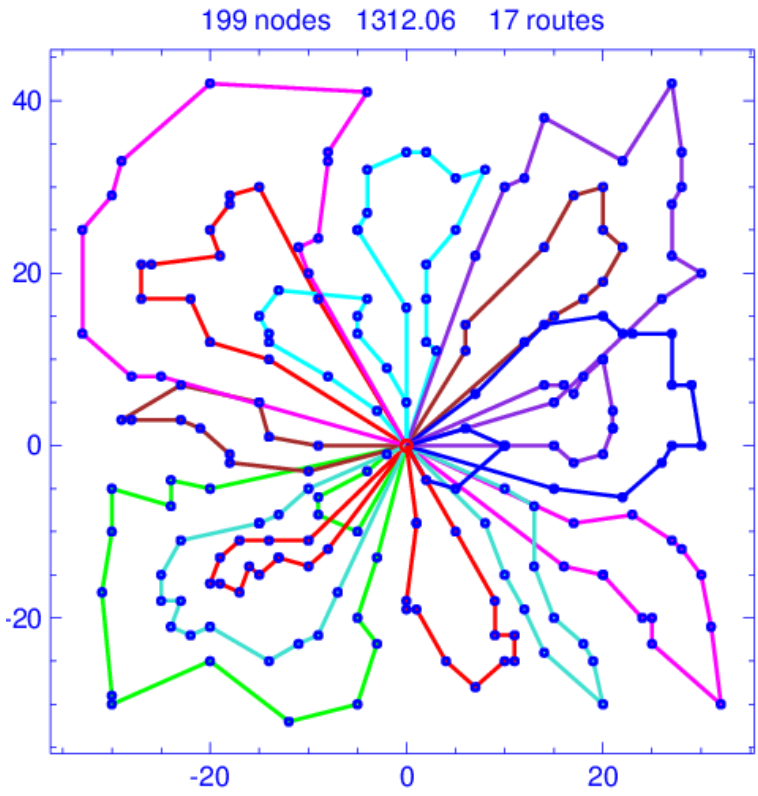


图 A7 200 个点路径图

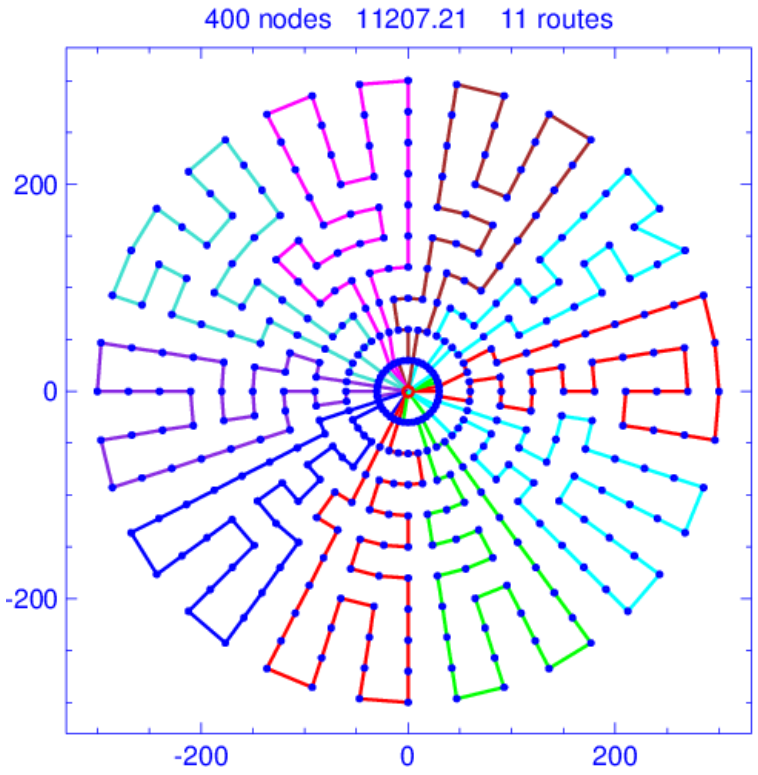


图 A8 400 个点路径图

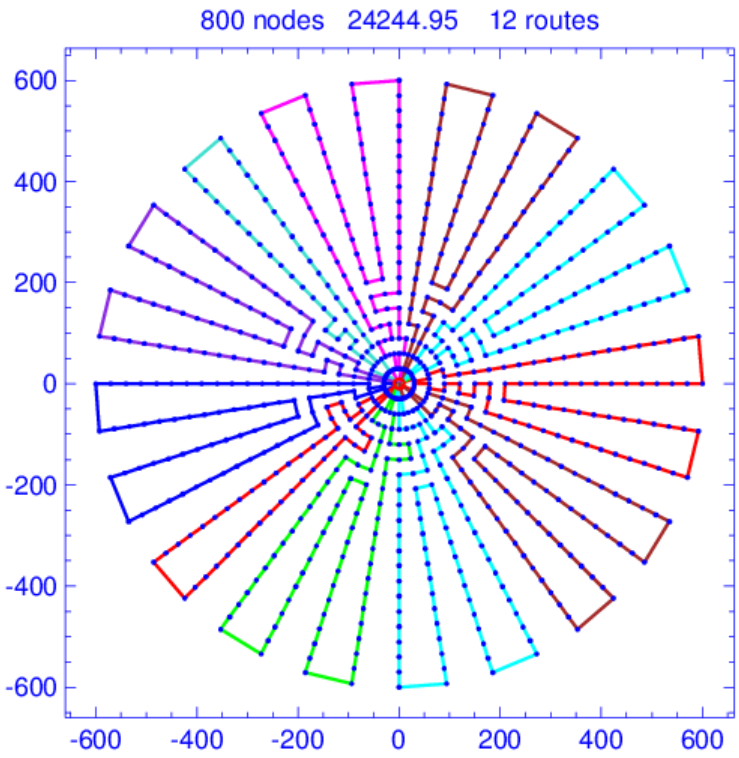


图 A9 800 个点路径图

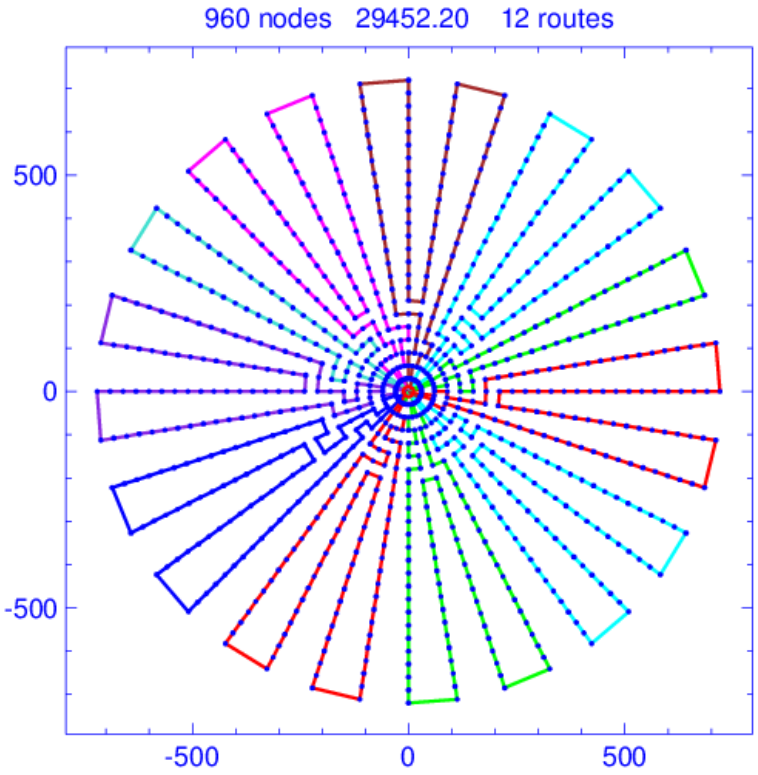


图 A10 960 个点路径图

附录 B 源代码

求解模块：

```
double VRP::Solver(int heuristics, int intensity, int max_try, int list_size,int accept_type, bool
verbose)
{
    int ctr, n, j, i, R, neighbor_list, objective;

    this->cooling_ratio = .99;
    if(this->max_route_length==VRP_INFINITY)
        this->temperature=2;
    else
        this->temperature=6;
    // changed temperature strategy

    neighbor_list=0;

    if(heuristics & VRPH_USE_NEIGHBOR_LIST)
        neighbor_list=VRPH_USE_NEIGHBOR_LIST;

    objective=VRPH_SAVINGS_ONLY;
    // default strategy

    if(heuristics & VRPH_TABU)
        this->tabu_list->empty();
    // reset the tabu list

    n=num_nodes;

    OnePointMove OPM;
    TwoPointMove TPM;
    TwoOpt          TO;
    // Define the heuristics we will use

    double start_val;
    int *perm;
    perm=new int[this->num_nodes+1];

    j=VRPH_ABS(this->next_array[VRPH_DEPOT]);

    for(i=0;i<this->num_nodes;i++)
    {
        perm[i]=j;
        if(!routed[j])
            report_error("%s: Unrouted node in solution!!\n");

        j=VRPH_ABS(this->next_array[j]);
    }

    if(j!=VRPH_DEPOT)
```

```
        report_error("%s: VRPH_DEPOT is not last node in solution!!\n");

    int rules;

    neighbor_list_size=VRPH_MIN(nlist_size, this->num_nodes);
    // Set the neighbor list size used in the improvement search

    record=this->total_route_length;
    this->best_total_route_length=this->total_route_length;
    this->export_solution_buff(this->current_sol_buff);
    this->export_solution_buff(this->best_sol_buff);

    normalize_route_numbers();

    ctr=0;


uphill:

    // Start an uphill phase using the following "rules":

    double beginning_best=this->best_total_route_length;

rules=VRPH_BEST_ACCEPT+VRPH_SIMULATED_ANNEALING+objective+neighbor_list+
VRPH_TABU;

    for(int k=1;k<intensity;k++)
    {
        start_val=total_route_length;
        this->temperature = this->cooling_ratio * this->temperature;

        if(heuristics & ONE_POINT_MOVE)
        {
            for(i=1;i<=n;i++)
            {
                OPM.search(this,perm[i-1],rules);
            }
        }

        if(heuristics & TWO_POINT_MOVE)
        {
            for(i=1;i<=n;i++)
                TPM.search(this,perm[i-1],rules);
        }

        if(heuristics & TWO_OPT)
        {
            for(i=1;i<=n;i++)
                TO.search(this,perm[i-1],rules);
        }
```

```
    }

    if(total_route_length<record)
        record = total_route_length;

    if(this->best_total_route_length<beginning_best-VRPH_EPSILON)
    {
        this->import_solution_buff(this->best_sol_buff);
    }

downhill:

    double orig_val=total_route_length;

    for(i=1;i<=this->get_total_number_of_routes();i++)
    {
        this->clean_route(i,ONE_POINT_MOVE+TWO_POINT_MOVE+TWO_OPT);
    }


    // Repeat the downhill phase until we find no more improvements
    if(total_route_length<orig_val-VRPH_EPSILON)
        goto downhill;

    if(total_route_length < record-VRPH_EPSILON)
    {
        // New record - reset ctr
        ctr=1;
        record=total_route_length;
    }
    else
        ctr++;

    if(ctr<max_stuck)
        goto uphill;


    delete [] perm;

    this->import_solution_buff(best_sol_buff);
    // Import the best solution found
        return best_total_route_length;
}
```

禁忌搜索模块：

```
VRPTabuList::VRPTabuList()
{
	///
	/// Default constructor for the VRPTabuList.
	///

	this->max_entries=0;
	this->num_entries=0;
	this->full=false;
	this->start_index=0;
	this->hash_vals1=NULL;
	this->hash_vals2=NULL;


}

VRPTabuList::VRPTabuList(int t)
{
	/// Constructor for the VRPTabuList with t tabu routes.

	this->max_entries=t;
	this->num_entries=0;
	this->full=false;
	this->start_index=0;
	this->hash_vals1=new int[t];
	this->hash_vals2=new int[t];

	int i;
	for(i=0;i<this->max_entries;i++)
	{
		this->hash_vals1[i]=-1;
		this->hash_vals2[i]=-1;
	}


}

VRPTabuList::~VRPTabuList()
{
	/// Destructor for the VRPTabuList.

	if(this->hash_vals1)
		delete [] hash_vals1;

	if(this->hash_vals2)
		delete [] hash_vals2;
}

void VRPTabuList::update_list(VRPRoute *r)
{
```

```
    /// Updates the tabu list by adding the route r.

    r->hash_val=r->hash(SALT_1);   //计算该路径的一对 hash 值
    r->hash_val2=r->hash(SALT_2);

    if(this->num_entries < this->max_entries)
    {
        // Update the lists of hash_vals
        this->hash_vals1[this->num_entries]=r->hash_val;
        this->hash_vals2[this->num_entries]=r->hash_val2;

        this->num_entries++;

        return;

    }

    // The list is full - overwrite the current start_index entry
    // and increment start_index

    this->hash_vals1[this->start_index]=r->hash_val;
    this->hash_vals2[this->start_index]=r->hash_val2;
    this->start_index = ((this->start_index + 1) % (this->num_entries));
    this->full=true;

    return;
}

void VRPTabuList::empty()
{
    /// Removes all entries from the tabu list.

    int i;
    for(i=0;i<this->max_entries;i++)
    {
        this->hash_vals1[i]=-1;
        this->hash_vals2[i]=-1;
    }

    this->start_index=0;
    this->num_entries=0;
    this->full=false;

    return;

}

Candidates::Candidates()
{
    VRPMove Mc;
    this->num_can=0;
```

```
	this->max_can=0;
	this->worst_obj=0;
}

Candidates::Candidates(int t)
{
	VRPMove Mc;
	this->num_can=0;
	this->max_can=t;
	this->can = new VRPMove[t];
	for(int i=0; i<t;i++)
	{
		this->can[i]=Mc;
	}
	this->worst_obj=0;
}

Candidates::~Candidates()
{
	if(this->can)
		delete [] can;
}


void Candidates::update_candidates(VRPMove* M)
{
	///update the candidates by adding Move M
	//fill up the candilist
	if(this->num_can < this->max_can)
	{
		this->can[num_can]=*M;
		///here is the problem
		this->num_can++;
		qsort(this->can,this->num_can,sizeof(VRPMove),VRPCondidateCompare);
		return;
	}

	//find the worst move
	this->worst_obj = this->can[this->max_can-1].savings;
	//if the new one if better than the worst on then replace it

	if(M->savings < this->worst_obj)
	{
		this->can[this->max_can-1]=*M;
		qsort(this->can,this->num_can,sizeof(VRPMove),VRPCondidateCompare);
	}
	//remember: before we use this function, we should sort it first

	return;

}
```

```
void Candidates::empty_can()
{
	VRPMove Mc;
	for(int i=0; i < this->max_can; i++)
	{
		this->can[i]=Mc;
	}

	this->num_can=0;
	this->worst_obj=0;
	return;
}
```

致　　谢

在本次毕业设计的过程中，非常感谢我的导师的悉心指导。不论是在每周一次的例会，还是在电话上的沟通，她耐心而专业的指导都给了我很大的帮助。最令我欣赏的是她传道授业的方式，不是单纯的告诉我该做什么怎么做，而是帮助我自我发现，引导我自己去摸索。我觉得这很重要，这不仅体现了老师的智慧，同时也是对学生的一种尊重。所以，在这个过程中我不仅学到了专业方面的知识。更重要的是，我学会了如何思考；学会了面对一件困难的事从何处下手，如何推进；明白了做事要追求专业；懂得了要勇于尝试。总之，能遇到一位良师确为我的幸运。最后，还是要再一次感谢我的导师 。

此外，值此毕业之际，感谢陪同度过这四年的老师和同学，丰富了我生命的色彩。

2014 年 5 月