运筹优化常用模型、算法及案例实战

Python+Java 实现

教材免费附赠代码手册

温馨提示：为了方便读者查看，本手册保持了原书的所有目录结构。
一些章节仅有标题，但是没有代码，这是正常的。
没有代码的章节，i) 本章节本来就不涉及代码实现；ii) 代码已经在本书正文提供了。
望读者周知。
2022 年 12 月 1 日

刘兴禄 **主编**

熊望祺 臧永森 段宏达 曾文佳 陈伟坚 **编著**

清华大学出版社

北 京

目录

插图

表格

序章

0.1 编写组成员

编委：本书各位编委的单位及联系方式如下：

刘兴禄　清华大学，清华-伯克利深圳学院，博士在读。
联系方式：hsinglul@163.com

熊望祺　毕业于清华大学工业工程系，硕士。
联系方式：xiongwq16@tsinghua.org.cn

臧永森　清华大学，工业工程系，博士在读。
联系方式：zangys15@tsinghua.org.cn

段宏达　清华大学，工业工程系，博士在读。
联系方式：dhd18@mails.tsinghua.edu.cn

曾文佳　毕业于清华大学工业工程系，硕士。
联系方式：zwj19@mails.tsinghua.edu.cn

陈伟坚　清华大学深圳国际研究生院副院长、清华-伯克利深圳学院副院长、
清华大学教授、博士生导师。联系方式：chanw@sz.tsinghua.edu.cn

顾问：

戚铭尧　清华大学，清华大学深圳国际研究生院，物流与交通学部副教授、
博士生导师。联系方式：qimy@sz.tsinghua.edu.cn

秦虎　华中科技大学，管理学院，管理科学与信息管理系教授，博士生导师。
联系方式：tigerqin@hotmail.com

0.2 内容简介

本书主要介绍了运筹优化领域常用的数学模型、精确算法以及相应的代码实现。本书首先简要介绍了大量基本理论，然后用丰富的配套案例讲解了多个经典的精确算法框架，最后结合常用的优化求解器 (CPLEX 和 Gurobi)，介绍了如何用 Python 和 Java 语言实现书中提到的所有精确算法。

全书共分为 3 部分。第 1 部分 (第 1~4 章) 为运筹优化常用模型及建模技巧。该部分着重介绍了整数规划的建模技巧和常见的经典模型。第 2 部分 (第 5~7 章) 为常用求解器 API 详解及应用案例。该部分主要介绍了两款常用的商业求解器 (CPLEX 和 Gurobi) 的使用方法，包括 Python 和 Java 的 API 详解、简单案例以及复杂案例。第 3 部分 (第 8~17 章) 为运筹优化常用算法及实战。该部分详细的介绍了几个经典的精确算法的理论、相关案例、伪代码以及相应的代码实现。

本书适合作为高等院校工业工程、管理科学与工程、信息管理与信息系统、数学与应用数学、物流工程、物流管理、控制科学与工程等开设运筹学相关课程的高年级本科生、研究生教材，同时可供在物流与供应链、交通、互联网、制造业、医疗、金融、能源等相关企业中从事有关运筹优化的开发人员、以及广大科技工作者和研究人员参考。

0.3 前言

0.3.1 为什么要写这本书

近年来，国内从事运筹优化学术研究的科研人员和工业界的运筹优化算法工程师日益增多，运筹优化逐渐得到国内各行各业的重视，这也是为广大运筹从业者所喜闻乐见的。物流、交通、供应链、电商、零售业、制造业、航空、金融、能源、定价与收益管理等各个领域，都有大量的运筹优化应用场景，同时，也有不少复杂的实际问题亟待解决。这对于国内从事运筹学研究的学者和算法工程师而言，无疑是巨大的挑战和机遇，对于该领域的在校博士生、硕士生，甚至本科生而言，亦是如此。

“问渠那得清如许，为有源头活水来”。一个行业要想长期欣欣向荣，就需要源源不断地涌入优质的行业人才，而行业人才的一个最重要的来源，就是在校博士生、硕士生和本科生。拥有高水平的运筹优化领域的研究生、本科生教育，是培养出优质行业人才的重要条件。打造丰富多样的优质教材是提高一个领域的教育水平的重要举措。笔者在硕士阶段，一直留心调研国内运筹优化教材的现状，发现到目前为止，市面上面向本科生教育的优质教材比较多，这些教材在基础理论的讲解上做的非常到位。但是，国内市面上面向研究生，甚至是已经从业的运筹优化算法工程师的优质教材并不多见，至于聚焦在有针对性的、详细的介绍运筹优化常用算法及其编程实战的教材，更是屈指可数。目前国内运筹优化领域的研究生教育所采用的高级运筹学教材，也大多都使用国外的课本，这些课本虽然在基本理论讲解方面非常详尽透彻，但是往往在实战方面，却少有涉及。大部分现有的教材都聚焦在讲解基本概念、基本理论、公式推导等方面，而不涉及具体代码实现层面的细节和技巧。国内的很多教材，也都聚焦在一些晦涩的理论推导及证明上，这让很多初学者望而却步。作为一名已经掌握了一些本科阶段运筹优化知识的学生或从业者而言，要找到一本合适的进阶版中文教材，以满足科研或者工作的需要，是非常困难的。本书就是一本同时能满足本科生课外拓展、研究生科研需要和工业界从业人员项目需要的教材。

“纸上得来终觉浅，绝知此事要躬行”。算法是一个非常讲究实战的领域，仅仅了解理论，而不能动手将其实现，很多时候并不能真正地掌握算法的精髓，达到融会贯通的境界。对于从事科研工作的硕士生和博士生，以及业界算法工程师而言，不能编程实现算法，意味着科研工作或者企业项目不能被推进。因此，笔者认为编写一本既能简洁清楚地讲解基本理论，又能有非常详细的案例和配套的代码的运筹优化算法教材，是非常有必要的。笔者在硕士阶段的初期，研究课题为车辆路径规划 (VRP)。为了推进这个研究，我阅读了大量的文献和教材，找到了论文的创新点，然后自以为可以较快的推进研究。但是，真正着手做研究的时候，发现要推进这个课题，需要掌握多方面的知识和技能，包括：第一，熟练掌握至少一种编程语言，这是为了能实现涉及到的算法。第二，懂得并且熟练使用整数规划的各种高级建模技巧。第三，掌握一些常用的精确算法或者启发式算法。第四，掌握至少一个优化求解器的使用方法。第五，将所选算法应用到自己课题的模型中去，并将其完整的实现。这还只是完整的复现一遍，不包括模型创新和算法创新的部分。

整个过程听起来似乎并不很困难，但是执行起来让我倍觉吃力。阅读文献，理解论文主旨和其中的算法原理，这部分比较轻松。可是到了算法实现方面，我遇到了一系列的困难。包括算法细节、数据结构、编程语言等，尤其是后两个部分，让我不知所措。虽然我本科也学过一些编程，但是由于学习并不深入，缺乏针对性的练习，编程能力非常薄弱。正因为如此，我在独立实现这些算法的过程中困难重重。于是我去 github 等平台到处搜集资料，希望能得到一些非常对口的代码或文档。但是在 2016、2017 年左右，这些平台上并没有非常相关的高质量参考资料，其他同类平台上也是类似的情况，资料零零散散，逐个筛选非常费时费力。

一个偶然的机会，我在朋友圈看到了一个微信公众号发布的技术科普文章，名为 《分支定界法解带时间窗的车辆路径规划问题》，我滑到推文顶部一看，这个公众号叫 **“数据魔术师”**。令我兴奋的是，该文章提供了完整的代码。我兴高采烈地下载了这篇文章的详细代码，如获至宝。之后我仔细研读，反复学习，最终找到了常见的精确算法实现的窍门。

随后的一段时间，我就接连开始攻关 Branch and Cut，Branch and Price 等精确算法，并在课题组组会上展示和交流。我硕士阶段的导师，清华大学深圳国际研究生院物流与交通学部副教授，戚铭尧老师也非常高兴，非常支持我钻研这些算法，在理论和实践层面都给予了我很多指导。尤其是关于算法的一些细节方面，老师跟我有多次深入的讨论，很多困惑也是在这个过程中得到解决的。另外，戚老师课题组内研究物流网络规划、车辆路径规划，智能仓储系统等方向的师兄师姐发表的论文中，也在频繁的使用 Branch and Cut，Branch and Price，Column Generation 等精确算法。总之，经过长时间的认真钻研，我终于大致掌握了上述算法。

后来，在研究 Dantzig-Wolfe 分解和拉格朗日松弛的时候，我已经快要硕士毕业了。当时我照例去 github 寻找参考资料，并且发现了一个非常高质量的代码。非常幸运，我联系到了这个代码的作者，他叫伍健，是西安交通大学毕业的硕士，现在是杉数科技的算法工程师。在这个高质量的代码的帮助下，我很快按照自己的理解完成了上述两个算法的实现。另外在实现的过程中，他也解答了我在算法细节方面的诸多问题。由于他的代码框架远胜于我自己的代码框架，所以这部分的代码，我就参照他原来代码的大框架，重新写了一版，放在本书相应章节中。不久前我开始撰写这两章，联系他阐明此事，他欣然同意，并且非常支持我，这让我非常感动。在这里，我也代表本书的作者们以及将来的读者们向伍健表示感谢！

在整个算法的学习过程中，我和同课题组的熊望祺 (也是本书的作者之一) 经常讨论探讨一个问题：从零开始学习这些算法，然后一一实现他们，难度究竟如何？我们的观点非常一致：实属不易，参考资料很少很杂，质量不高！我们不止一次的感叹，为什么没有一本能解决我大部分疑惑的资料呢？当我们学习理论时，可以很轻松地找到好的参考资料，但是在尝试去实现这些算法时，却难以获取很详细的资料。要么就只能钻研数百页的较为复杂的用户手册，自己慢慢筛选有用信息。要么就只能找到一些零散的代码资料和文档，大部分时候这些代码甚至没有任何注释，就连代码实现的具体模型、具体算法都没有做解释说明，更不用说细节方面的解释了。通常的情况是，读者看懂了 A 模型的算法，经过筛选大量资

料，却搜集到了 B 模型的代码，而 B 模型的代码对应的算法、注释很不齐全，每个函数的具体功能也没有提供有用的文档。如果读者要硬着头皮研读，可能会花费大量时间，并且很有可能是做无用功。相信经常编程的同行都了解，阅读别人的代码是多么痛苦的事，更何况是没有伪代码，没有注释，也没有 README 的代码。

在读博以后，我和本书的另外两名作者，臧永森和段宏达在闲聊的时候，常常聊到运筹优化方向参考资料匮乏这件事。慢慢的，我萌生了把这些资料整理成书出版，供想要入门和进阶的博士生、硕士生、本科生或者业界的算法工程师学习、查阅的想法。我也将这个想法告诉了熊望祺，他非常赞成，并表示愿意一起合作将这本书整理出版，于是他将平时完成的部分精确算法的代码提供给了我，并将这些内容整合在本书相应章节中。之后，我也向臧永森和段宏达表达了合作意愿，他们也都认为这是一件有价值的事，并且不久后，他们也正式加入了编写团队，为本书做出了非常重要的贡献。

读博期间写书需要花费大量时间，个别时候会耽误一点科研进度，因此我非常担心我导师不同意我做这件事。但是当我向我的导师，清华-伯克利深圳学院副院长陈伟坚老师表明了我写书的计划以后，陈老师非常支持，并鼓励我多做调研，明确本书的受众，构思好书的脉络，要坚持做完整，不要半途而废。然后多次和我讨论如何设计本书的框架，如何编排各个章节，一些细节相关的内容如何取舍，以及针对不同受众如何侧重，还对本书未来的拓展方向、需要改进完善的资源提出了很多非常有帮助的建设性意见。这些意见对本书的顺利完成起到了不可或缺的作用。

目前正在攻读博士的我，虽然对运筹优化有极大的热情，但是回想起过去两三年前的“小白”阶段的学习过程，还是觉得比较痛苦。很多次，我都心力交瘁，濒临放弃的边缘。我个人觉得，像我一样有同样困惑的同行，兴许有不少吧。我不希望每个像我一样的运筹优化爱好者 (初学者)，都去经历那样的过程，把大量时间浪费在甄别杂乱、不系统的资料中去。我认为，国内应当有一些完整、系统的资料，帮助运筹优化爱好者们突破入门，走上进阶，把主要精力放在探索更新的理论中去，或者将这些理论应用于解决新的问题上，真正创造价值。这也是我们花了大量时间和精力，将一些常用技巧、模型、算法原理、算法的伪代码以及算法的完整代码实现通过系统的整理，以通俗易懂的语言串联在一起，最后编成这本书的初衷。

“不积跬步，无以至千里；不积小流，无以成江海”。笔者从硕士一年级开始慢慢探索积累，直到博士三年级，终于完成了全书的撰写。如今这本书得以面世，心中又喜又忧。喜自不必说，忧在怕自己只是一个博士在读生，见解浅薄，功夫远不到家，并且书中一定存在一些自己还没发现的错误，不够完美，影响读者阅读……总之，走过整个过程，才越来越深入理解厚积薄发的含义。回想过去几年，真是感慨万千。我曾一次次在实验室 debug 到凌晨，直到逮到 bug，代码调通，紧锁的眉头顿时舒展。然后我潇洒的将电脑合上，慢悠悠插上耳机，听着自己喜欢的歌，撒着欢儿，披着学术长廊的灯光，傍着我的影子就溜达回宿舍了。那种成就感和幸福感，着实是一种享受。虽然那段时间我经常宿舍、食堂、实验室三点一线，听起来这种生活似乎毫无亮点，枯燥乏味，但是我每天都在快速进步，我很喜欢这样的状态。这种状态很像陶渊明先生在《五柳先生传》中的描述：“好读书，不求甚解；每有会

意，便欣然忘食”。不求甚解在这里不必纠结其具体观点，单就欣然忘食这种喜悦感，我认为是相通的。当自己真正有收获的时候，内心的快乐是油然而生的。也正由于那段时间的积累，才有了现在这本书。博观约取，厚积薄发，希望低年级的同学们坚持积累，不断提升，最终达到梦想的顶峰！

0.3.2 本书内容安排

为了能够让读者系统的学习运筹优化算法，我们将本书分成以下三个部分：运筹优化常用模型及建模技巧、常用优化求解器 API 详解及应用案例、运筹优化常用算法及实战。

运筹优化常用模型及建模技巧部分不需要读者有任何的编程基础，只需要掌握本科的运筹学线性规划相关理论和基本的对偶理论以及整数规划基础知识即可顺利读懂。本部分分为 4 章。第 1 章介绍了一些数学规划模型的分类和后面章节的精确算法会涉及到的一些凸优化领域的概念，包括凸集、凸包等。由于本书的重点不在此，因此本章介绍的非常简略。第 2 章介绍了一些非常常见的运筹学问题，例如指派问题、旅行商问题、车辆路径规划问题和多商品网络流问题。这些问题非常经典，并且具有代表性，当下很多实际问题都可以转化为这些问题的变种和拓展。第 3 章讲解了整数规划中常用的建模技巧，包括逻辑约束等约束的写法、非线性项的线性化方法等，这些技巧频繁的出现在业内顶级期刊发表的论文中，因此本书专门设置了一章来介绍这些内容。第 4 章讲解了运筹优化中一个非常重要的理论——对偶理论。不同于其他教材，本书中这一章主要聚焦如何写出大规模线性规划的对偶问题，是作者通过自己探索总结出的方法，目前国内外的各种教材和网站，鲜有看到类似的方法介绍。

第一部分是为后续的算法部分做铺垫，之后章节中会多次用到这一章介绍的概念和模型。

常用优化求解器 API 详解及应用案例 部分分为 3 章，主要是介绍常用优化求解器 (CPLEX 和 Gurobi) 及其应用案例。这部分将为之后的算法实战作好技术铺垫，本书第三部分的章节，都需要用到这一章的内容。要顺利读懂这一部分，需要读者有一定的编程基础，至少需要掌握 Java 或者 Python 中的一门语言。第 5 章详细地介绍了 CPLEX 的 Java 接口的用法。这部分的主要内容是根据 CPLEX 提供的用户手册整理而来，包括基本的类、callback、以及例子库中部分例子代码的解读。该章能够帮助读者快速的掌握 CPLEX 的 Java 接口的使用，读者无需去研读厚厚的英文版用户手册。第 6 章系统的介绍了 Gurobi 的算法框架以及其 Python 接口的用法。包括常用类、Python 调用 Gurobi 的完整建模过程、日志信息、callback 等部分。最后还附以简单的案例帮助读者理解。第 7 章提供了带时间窗的车辆路径规划 (VRPTW) 的代码实现。详细的给出了如何调用求解器，建立 VRPTW 的模型并求解。这个案例略微复杂，掌握了这个案例，其余更高难度的案例也就可以迎刃而解。但是本章的案例都是直接调用求解器得到模型的解，并没有涉及到自己实现精确算法的内容。

运筹优化常用算法及实战部分是本书最为重要，干货最多的部分。该部分全面、系统的将运筹优化中常用的精确算法以通俗易懂的方式讲解给读者，尽量避免晦涩的解读。为

了方便读者自己实现算法，我们为每一个算法，都提供了详细完整的伪代码，这些伪代码可以帮助读者自己从 0 到 1 动手实现相应的算法。在第 8 章，我们简要回顾了单纯形法，并给出了伪代码和 Python 代码实现。第 9 章，我们介绍了求解最短路问题的 Dijkstra 算法及其实现。Dijkstra 算法也是一个非常常用的基础算法。第 10 章到第 17 章，我们以理论 + 详细小案例 + 伪代码 + 复杂大案例 + 完整代码实现的方式，为读者介绍了 Branch and Bound，Branch and Cut，Column Generation，Dynamic Programming，Branch and Price，Dantzig-Wolfe Decomposition，Benders Decomposition，Lagrangian Relaxation 这 8 个经典且常用的精确算法。这些算法经常出现在运筹学领域各个期刊的文章中，以及在工业界的具体项目之中。为了便于读者理解，我们尽量避免复杂的数学推导，着重讲解基本原理和算法迭代步骤，真正意义上帮助读者从理论到实践，一步到位，无需到处寻找零散资料，做重复性的整合工作。

相信认真研读这本教材的读者，一定会大有收获。尤其是对刚入门的硕士生和博士生们，可以凭借这本教材，系统的掌握本领域的精确算法，更好的胜任自己的科研工作。对于已经从业的运筹优化算法工程师，本书也可以作为一本非常详尽的学习工具书。

这里需要做一点特别说明，本书不同章节代码的继承性不大，大部分章节的代码都是针对该章节独立编写的。这种做法是为了方便初学者较快的理解每一章的代码，更好的消化每个单独的算法。但是这种编排也有一些弊端，即不利于拓展和改进。实际上，大部分精确算法之间都是有联系的，科研过程中也经常将它们组合使用，如果将所有章节的代码做成以一个集成的算法包，既方便管理，又便于拓展。但是，集成较好的代码，其结构一般都很复杂，初学者面对这样的代码，往往不知所措。复杂的函数调用关系，众多的类和属性，难免会让初学者望而生畏。基于此，本书中各个章节的代码还是保持了较高的独立性，今后如果有机会，笔者会考虑提供两个版本的代码，即独立版本和集成版本，前者主要面向初学者，后者主要面向较为熟练的读者。

为了方便读者更容易的对照本领域内的英文文献，本书对涉及到的专业名词 (概念、问题名称和算法名称等)，在其第一次出现的时候，给出相应的中文翻译，其他时候，我们均使用英文名称。

此外，为了方便读者理解，我们为书中的代码添加了详细注释，同时也将一些等价的操作方法或者辅助调试的部分直接放在代码中，并未将其删除，这也许并不是很标准的做法，但是为了方便读者学习，我们仍然将其保留。本书中代码的主要作用是为读者提供一个样例，帮助读者了解如何实现书中涉及到的算法，部分章节的代码虽然有重复，但是为了保证自解释性，我们仍然提供了完整的代码。还有一部分代码，我们并没有做精心优化，其性能也得不到较好的保证，追求性能的读者需要自己对其进行优化。

由于笔者水平和时间有限，在整理的过程中，难免有疏漏和错误，很多参考了网上的资料也由于时间久远不能找到出处，希望具体位置的原作者看到以后，及时与笔者联系。再版之际，我们将会做出修改。也希望广大热心读者为本书提供宝贵的改进意见。

本书的一些相关资料以及所有章节的拓展讲解会在微信公众号 **“运小筹”** 持续更新，欢迎各位读者关注。

0.3.3 本书内容的先修课

运筹优化常用模型及建模技巧部分：先修课程为《运筹学》。另外，最好修过《凸优化》这门课 (没有选修这门课也不影响阅读学习)。

常用优化求解器 API 详解及应用案例部分：先修课程为《Java 编程基础》，或者《Python 编程基础》。

运筹优化常用算法及实战部分：修过以上两部分的先修课程的同学，也可以无障碍阅读学习。如果还修过《高级运筹学》或者《整数规划》，学习会更加轻松。如果没有修过上述课程，也不影响学习，我们的教材内容非常通俗易懂。

0.3.4 致谢

非常感谢我博士阶段的指导老师，清华大学清华-伯克利深圳学院陈伟坚教授。陈老师对本书整体框架、章节安排等方面提出了诸多建设性意见，也参与了本书的校对、完善等工作。而且，陈老师从读者角度出发，对全书内容的精炼等方面提出了若干宝贵修改意见，有效的提升了本书的可读性和可拓展性。另外，陈老师对本书后续的完善、以及图书习题部分的补充也给出了明确方向。

感谢我硕士阶段的指导老师，清华大学深圳国际研究生院，物流与交通学部戚铭尧教授。戚老师对本书算法理论介绍部分以及内容完整性方面提出了诸多宝贵的修改意见。

感谢我《高级运筹学》课程的授课老师张灿荣教授。张老师精彩的讲授，激起了我对运筹优化浓厚的学习兴趣。此外，该课程充足、有针对性的课下阅读材料也为我深入的学习提供了有效的帮助。

感谢运筹优化领域微信公众号 **“数据魔术师”** 给予我的莫大帮助。该公众号发布的优化算法介绍文章以及完整代码资料为我实现精确算法提供了非常有用的参考。特别的，在 Branch and Bound 的实现过程中，我参考了该公众号分享的部分代码。由衷感谢该公众号的运营者：华中科技大学管理学院秦虎教授以及他的团队。秦虎老师的团队发布的算法科普文章以及举办的精确算法系列讲座，犹如雪中送炭，为很多运筹领域的初学者提供了巨大帮助。

感谢伍健在 DW-分解算法、拉格朗日松弛算法方面提供的资源。也非常感激在算法实现过程中，伍健对我诸多疑惑的耐心解答，同时也非常感谢伍健对这本书出版的大力支持。

此外，笔者还很荣幸的邀请到了我的老师、师兄、师姐、师弟、师妹们参与了本书后期的读稿和校对工作。他们是：陈名华 (香港中文大学教授，我的老师)，张莹、黄一潇、陈锐、王祖健、游锦涛、何彦东、王美芹、王梦彤、段淇耀、贺静、张文修、周鹏翔、王丹、夏旸、李怡、郝欣茹、朱泉、修宇璇、王涵民、张祎、梁伯田、陈琰钰、王基光、徐璐、左齐茹仪、张婷、李良涛、赖克凡、曹可欣、金广、席好宁、俞佳莉、陈梦玄。非常感谢他们宝贵的意见和建议。

感谢运筹优化领域公众号 **“运筹 OR 帷幄”**。该公众号发布的高质量的理论介绍文章、讲座预告等给予了我大量的信息，在拓宽视野，了解运筹领域交叉学科的发展现状等方面

给了我莫大的帮助。

感谢本书中所有参考文献的作者，是你们的研究成果让我学到了许多新的知识，获得了不少新的启发。本书中部分内容参考或者翻译自这些文献，在此向这些研究者们致以崇高的敬意。

希望本书能得到广大读者的喜爱，为大家提供有效的帮助。

刘兴禄

2020 年 12 月于清华大学深圳国际研究生院

0.4 作者贡献

本书所有章节的贡献者如下。

表 0.1: 作者贡献

具体内容	贡献
第 1 章: 运筹优化算法相关概念介绍	**段宏达**: 凸集、极点、多面体、超平面与半平面等概念介绍 (刘兴禄修改) **刘兴禄**: 几类常见的数学规划模型概念介绍
第 2 章: 运筹优化经典问题数学模型	**刘兴禄、陈伟坚**
第 3 章: 整数规划建模技巧	**段宏达**: 逻辑约束、线性化部分 **刘兴禄**: 分段函数线性化部分及 Gurobi 代码实现部分
第 4 章: 大规模线性规划的对偶	**刘兴禄、陈伟坚**
第 5 章: CPLEX 的 Java API 详解及简单案例	**熊望祺**: 全章内容撰写 **臧永森、刘兴禄**: 全章修改
第 6 章: Gurobi 的 Python API 详解及简单案例	**刘兴禄**：第 6.2.1-6.2.8, 6.2.10-6.2.11 节及整章修改 **臧永森**: 第 6.2.9 节 **曾文佳**: Python 调用 Gurobi 可以求解的问题类型部分的代码和 Gurobi 算法介绍部分
第 7 章: 调用 CPLEX 和 Gurobi 求解 MIP 的复杂案例: VRP	**刘兴禄**
写在算法之前	**臧永森、刘兴禄**
第 8 章: 单纯形法	**刘兴禄、陈伟坚**
第 9 章: Dijkstra 算法	**刘兴禄**
第 10 章: 分支定界算法	**刘兴禄**: 理论介绍、算法伪代码、整章修改、Branch and bound 求解 TSP、Branch and bound 求解 VRPTW 的全部 Python 代码 **数据魔术师公众号 & 黄楠 & 刘兴禄**: Branch and bound 的 Java 代码（版本 2） **数据魔术师公众号 & 黄楠**：Branch and bound 的 Java 代码（版本 1）
第 11 章: 分支切割算法	**刘兴禄**: Cutting plane、Branch and cut 理论介绍伪代码和 Branch and cut 求解 VRPTW 的 Java 代码，Branch and cut 求解 VRPTW 的 Python 代码，以及整章修改 **曾文佳**: Branch and cut 求解 CVRP 和 VRPTW 的理论介绍和 Java+callback 实现
第 12 章: 拉格朗日松弛	**刘兴禄**: 理论介绍、伪代码以及修改版本的 Lagrangian relaxation 求解 Location Transport Problem 的 Python 代码及整章的修改 **伍健**: Lagrangian relaxation 求解 Location Transport Problem 的 Python 代码 (代码原作者)

具体内容	贡献
第 13 章：列生成算法	**刘兴禄**：理论介绍及相应 Column Generation 的 Java 、Python 版本的代码及整章修改 **熊望祺**：Column Generation 的版本 2 的 Java 代码
第 14 章：动态规划	**臧永森**：第 1 节理论介绍及相应案例的 Java 代码 **刘兴禄**：Dynamic Programming 求解 TSP 的理论介绍、伪代码和 Python 代码及整章修改 **刘兴禄**：SPPRC 的 Labelling algorithm 的理论介绍、伪代码和 Python 代码 **刘兴禄**：Python 实现用 SPPRC 的 Labelling Algorithm 加 Column Generation 求解 VRPTW 的理论介绍和代码 **熊望祺**：Java 实现大规模 SPPRC 的 Labelling Algorithm 的代码
第 15 章：分支定价算法	**刘兴禄**：理论介绍、算法框架、伪代码、Branch and Price 的 Python 代码及整章修改 **熊望祺**：Java 实现 Branch and Price 的代码
第 16 章：Dantzig-Wolfe 分解算法	**段宏达**：理论介绍、具体案例和相应伪代码 **刘兴禄**：部分理论介绍、简单案例的 Python 代码和 CG 结合 D-W Decomposition 的 Python 代码及整章修改 **伍健**：Dantzig-Wolfe Decomposition 求解 Multicommodity Network Flow 问题的 Python 代码 (代码原作者)
第 17 章：Benders 分解算法	**刘兴禄** ：理论介绍、伪代码、第一个版本的 Benders Decomposition 的 Java 代码、修改版本的 Benders Decomposition 求解 Fixed Charge Transportation Problem 的 Python 代码及整章的修改 **熊望祺**：Fixed Charge Transportation Problem 和 Uncapacitated Facility Location Problem 的 Benders Decomposition 部分的 Java 代码 **数据魔术师公众号 & 黄楠 & 刘兴禄**：第一个版本的 Benders Decomposition 的 Java 代码 **伍健**：Benders Decomposition 求解 Fixed Charge Transportation Problem 的 Python 代码 (代码原作者)，Benders Decomposition 求解 Uncapacitated Facility Location Problem 的 Python 代码 (代码原作者)
术语与缩写对照表	**曾文佳**
全书框架、章节安排、内容精炼	**陈伟坚**
所有附赠代码审查、修改和排版	**刘兴禄、陈伟坚**

第 I 部分

运筹优化常用模型及建模技巧

第 1 章　运筹优化算法相关概念介绍

本章我们主要介绍一些运筹优化相关的概念，包括常见的数学规划模型以及凸集、极点、多面体、超平面与半平面等。本章旨在为后续章节做铺垫，因此仅对涉及到的概念做简要介绍，需要深入了解的读者请参考文献Boyd et al. 2004。

1.1 几类常见的数学规划模型

1.1.1 线性规划

1.1.2 混合整数规划

1.1.3 二次规划

1.1.4 二次约束规划

1.1.5 二次约束二次规划

1.1.6 二阶锥规划

1.2 凸集和极点

1.2.1 凸集

1.2.2 极点

1.3 多面体、超平面与半平面

1.3.1 多面体

1.3.2 超平面与半平面

1.4 凸组合和凸包

1.4.1 凸组合和凸包

1.4.2 一些结论

第 2 章　运筹优化经典问题数学模型

2.1 指派问题

2.2 最短路问题

2.3 最大流问题

2.3.1 问题描述

2.3.2 问题建模及最优解

2.3.3 最大流问题的一般模型

2.3.4 Ford–Fulkerson 算法求解最大流问题

2.3.5 Java 实现 Ford–Fulkerson 算法求解最大流问题

FordFulkerson.java

FordFulkerson.java

```java
package maxflow;

import java.io.IOException;
import java.util.LinkedList;
import java.util.Queue;

import graph.FlowEdge;
import graph.FlowNetwork;

/**
 * Ford-Fulkerson algorithm to compute max-flow/min-cut.
 *
 * @author Xiong Wangqi
 * @version V1.0
 * @since JDK1.8
 */
public class FordFulkerson {
    private static final double EPS = 1e-10;
    /** marked[v] = true iff s->v path in residual graph. */
    private boolean[] marked;
    /** edgeTo[v] = last edge on shortest residual s->v path. */
    private FlowEdge[] edgeTo;
    /** current value of max flow. */
    private double value;

    /**
     * Compute a max flow and min cut in the network g from vertex s to t: <br>
     * The key is increasing flow along augmenting paths.
     *
     * @param g the flow network
     * @param s the source vertex
     * @param t the sink vertex
     */
    public FordFulkerson(FlowNetwork g, int s, int t) {
        validate(s, g.getVertexNum());
        validate(t, g.getVertexNum());

        if (s == t) {
            throw new IllegalArgumentException("Source equals sink");
        }

```

```
42          if (!isFeasible(g, s, t)) {
43              throw new IllegalArgumentException("Initial flow is infeasible");
44          }
45
46          // while there exists an augmenting path, use it(the value)
47          value = excess(g, s);
48          while (hasAugmentingPath(g, s, t)) {
49              // 1 compute bottleneck capacity
50              double bottleneck = Double.POSITIVE_INFINITY;
51              for (int v = t; v != s; v = edgeTo[v].other(v)) {
52                  bottleneck = Math.min(bottleneck, edgeTo[v].residualCapacityTo(v));
53              }
54
55              // 2 update the flow on the augment path
56              for (int v = t; v != s; v = edgeTo[v].other(v)) {
57                  edgeTo[v].addResidualFlowTo(v, bottleneck);
58              }
59
60              // 3 update current "max-flow"
61              value += bottleneck;
62          }
63          // If there is no augmenting path, then the max-flow/min-cut is found.
64      }
65
66      /**
67       * is v in the s side of the min s-t cut?/is v reachable from s in residual network?.
68       *
69       * @param v the vertex
70       * @return true if vertex is on the side of mincut, false otherwise
71       */
72      public boolean inCut(int v) {
73          validate(v, marked.length);
74          return marked[v];
75      }
76
77      /**
78       * Return the value of the max flow.
79       *
80       * @return the value of the max flow
81       */
82      public double value() {
83          return value;
84      }
85
86      /**
87       * Is there an augmenting path? <br>
88       * if so, upon termination edgeTo[] will contain a parent-link representation of such a path <br>
89       * this implementation finds a shortest augmenting path (fewest number of edges), <br>
90       * which performs well both in theory and in practice <br>
91       * The augmenting path: <br>
92       * 1 can increase flow on forward edges (not full) <br>
93       * 2 can decrease flow on backward edge (not empty).
94       *
95       * @param g the flow network
96       * @param s the source vertex
97       * @param t the sink vertex
98       * @return True if there is an augmenting path, false otherwise
99       */
100     private boolean hasAugmentingPath(FlowNetwork g, int s, int t) {
```

```
        int vertexNum = g.getVertexNum();
        edgeTo = new FlowEdge[vertexNum];
        marked = new boolean[vertexNum];

        // breadth-first search
        Queue<Integer> queue = new LinkedList<>();
        queue.add(s);
        marked[s] = true;
        while (!queue.isEmpty() && !marked[t]) {
            int v = queue.poll();

            for (FlowEdge e: g.adj(v)) {
                int w = e.other(v);
                if (e.residualCapacityTo(w) > 0) {
                    if (!marked[w]) {
                        edgeTo[w] = e;
                        marked[w] = true;
                        queue.add(w);
                    }
                }
            }
        }

        // Is there an augmenting path?
        return marked[t];
    }

    private boolean isFeasible(FlowNetwork g, int s, int t) {
        // check that capacity constraints are satisfied
        int vertexNum = g.getVertexNum();
        for (int v = 0; v < vertexNum; v++) {
            for (FlowEdge e: g.adj(v)) {
                if (e.getFlow() < -EPS || e.getFlow() > e.getCapacity()) {
                    System.err.println("Edge does not satisfy capacity constraints: " + e);
                    return false;
                }
            }
        }

        // check that net flow into a vertex equals zero, except at source and sink
        if (Math.abs(value + excess(g, s)) > EPS) {
            System.err.println("Excess at source = " + excess(g, s));
            System.err.println("Max flow         = " + value);
            return false;
        }
        if (Math.abs(value - excess(g, t)) > EPS) {
            System.err.println("Excess at sink = " + excess(g, t));
            System.err.println("Max flow         = " + value);
            return false;
        }
        for (int v = 0; v < vertexNum; v++) {
            if (v == s || v == t) {
                continue;
            }

            if (Math.abs(excess(g, v))> EPS) {
                return false;
            }
        }
```

```
160
161             return true;
162         }
163
164         /**
165          * Return excess flow(in flow - out flow) at vertex v.
166          *
167          * @param g the flow network
168          * @param v the vertex
169          * @return excess flow at vertex v
170          */
171         private double excess(FlowNetwork g, int v) {
172             double excess = 0.0;
173             for (FlowEdge e: g.adj(v)) {
174                 if (v == e.from()) {
175                     // out-flow
176                     excess -= e.getFlow();
177                 } else {
178                     // in-flow
179                     excess += e.getFlow();
180                 }
181             }
182
183             return excess;
184         }
185
186         private void validate(int v, int n)  {
187             if (v < 0 || v >= n) {
188                 throw new IndexOutOfBoundsException("vertex " + v + " is not between 0 and " + (n - 1));
189             }
190         }
191
192         public static void main(String[] args) {
193             // filename of the graph
194             String filename = "./data/tinyFN.txt";
195
196             FlowNetwork g = null;
197             try {
198                 g = new FlowNetwork(filename);
199             } catch (IOException e) {
200                 e.printStackTrace();
201             }
202
203             int s = 0;
204             int vertexNum = g.getVertexNum();
205             int t = vertexNum - 1;
206             FordFulkerson maxFlow = new FordFulkerson(g, s, t);
207
208             StringBuilder sb = new StringBuilder(String.format("Max flow from %d to %d:\n", s, t));
209             for (int v = 0; v < vertexNum; v++) {
210                 for (FlowEdge e: g.adj(v)) {
211                     if ((v == e.from()) && e.getFlow() > 0) {
212                         sb.append(e);
213                         sb.append('\n');
214                     }
215                 }
216             }
217
218             // print min-cut
```

```
219             sb.append("Min cut: ");
220             for (int v = 0; v < vertexNum; v++) {
221                 if (maxFlow.inCut(v)) {
222                     sb.append(v + " ");
223                 }
224             }
225             sb.append('\n');
226             sb.append("Max flow value = " +  maxFlow.value());
227 
228             System.out.print(sb.toString());
229         }
230 
231     }
```

FlowEdge.java

FlowEdge.java

```
1   package graph;
2   
3   /**
4    * Flow edge class.
5    *
6    * @author Xiong Wangqi
7    * @version V1.0
8    * @since JDK1.8
9    */
10  public class FlowEdge {
11      private final int from;
12      private final int to;
13      private final double capacity;
14      private double flow;
15  
16      public FlowEdge(int from, int to, double capacity) {
17          if (from < 0 || to < 0) {
18              throw new IndexOutOfBoundsException("Vertex name must be a nonnegative integer");
19          }
20          if (capacity < 0.0) {
21              throw new IllegalArgumentException("Edge capacity must be nonnegaitve");
22          }
23  
24          this.from = from;
25          this.to = to;
26          this.capacity = capacity;
27          this.flow = 0.0;
28      }
29  
30      public FlowEdge(int from, int to, double capacity, double flow) {
31          if (from < 0 || to < 0) {
32              throw new IndexOutOfBoundsException("Vertex name must be a nonnegative integer");
33          }
34          if (capacity < 0.0) {
35              throw new IllegalArgumentException("Edge capacity must be nonnegaitve");
36          }
37  
38          if (flow < 0.0) {
39              throw new IllegalArgumentException("Flow must be nonnnegative");
40          }
41          if (flow > capacity) {
```

```
            throw new IllegalArgumentException("Flow exceeds capacity");
        }

        this.from = from;
        this.to = to;
        this.capacity = capacity;
        this.flow = flow;
    }

    public FlowEdge(FlowEdge e) {
        this.from = e.from;
        this.to = e.to;
        this.capacity = e.capacity;
        this.flow = e.flow;
    }

    /**
     * Return the residual capacity of the edge in the direction to the given vertex: <br>
     * if vertex is the head vertex, the residual capacity equals {@link #flow} <br>
     * if vertex is the tail vertex, the residual capacity equals {@link #capacity} - {@link #flow}.
     *
     * @param vertex one endpoint of the edge
     * @return the residual capacity of the edge in the direction to the given vertex
     */
    public double residualCapacityTo(int vertex) {
        if (vertex == from) {
            return flow;
        } else if (vertex == to) {
            return capacity - flow;
        } else {
            throw new IllegalArgumentException("Illegal endpoint");
        }
    }

    /**
     * Increases the flow on the edge in the direction to the given vertex: <br>
     * if vertex is the head vertex, this decreases the flow on the edge by delta <br>
     * if vertex is the tail vertex, this increases the flow on the edge by delta.
     *
     * @param vertex one endpoint of the edge
     */
    public void addResidualFlowTo(int vertex, double delta) {
        if (vertex == from) {
            flow -= delta;
        } else if (vertex == to) {
            flow += delta;
        } else {
            throw new IllegalArgumentException("Illegal endpoint");
        }

        if (flow < 0.0) {
            throw new IllegalArgumentException("Flow must be nonnnegative");
        }
        if (flow > capacity) {
            throw new IllegalArgumentException("Flow exceeds capacity");
        }
    }

    public int other(int vertex) {
```

```
        if (vertex == from) {
            return to;
        } else if (vertex == to) {
            return from;
        } else {
            throw new IllegalArgumentException("Illegal endpoint");
        }
    }

    public int from() {
        return from;
    }

    public int to() {
        return to;
    }

    public double getCapacity() {
        return capacity;
    }

    public double getFlow() {
        return flow;
    }

    @Override
    public String toString() {
        return from + "->" + to + " " + flow + "/" + capacity;
    }

}
```

FlowNetwork.java

FlowNetwork.java

```
package graph;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.LinkedList;
import java.util.Scanner;


/**
 * The FlowNetwork class, representing a capacitated network with vertexes, <br>
 * and directed flow edge {@link FlowEdge} with a real-valued capacity and flow.
 *
 * @author Xiong Wangqi
 * @version V1.0
 * @since JDK1.8
 */
public class FlowNetwork {
    private final int vertexNum;
    private int edgeNum;
    private LinkedList<FlowEdge>[] adj;

    @SuppressWarnings("unchecked")
    public FlowNetwork(int vertexNum) {
```

```
24          // TODO Auto-generated constructor stub
25          if (vertexNum < 0) {
26              throw new IllegalArgumentException("Number of vertices in a Digraph must be nonnegative");
27          }
28          this.vertexNum = vertexNum;
29
30          edgeNum = 0;
31          adj = new LinkedList[vertexNum];
32          for (int v = 0; v < vertexNum; v++) {
33              adj[v] = new LinkedList<>();
34          }
35      }
36
37      @SuppressWarnings("unchecked")
38      public FlowNetwork(String filename) throws FileNotFoundException {
39          Scanner scanner = new Scanner(new FileInputStream(filename));
40
41          // 前两行为节点数和边数
42          vertexNum = scanner.nextInt();
43          adj = new LinkedList[vertexNum];
44          for (int i = 0; i < vertexNum; i++) {
45              adj[i] = new LinkedList<>();
46          }
47
48          int n = scanner.nextInt();
49          for (int i = 0; i < n; i++) {
50              int from = scanner.nextInt();
51              int to = scanner.nextInt();
52              double capacity = scanner.nextDouble();
53              addEdge(new FlowEdge(from, to, capacity));
54          }
55
56          scanner.close();
57      }
58
59      public void addEdge(FlowEdge e) {
60          int v = e.from();
61          int w = e.to();
62          validateVertex(v);
63          validateVertex(w);
64
65          adj[v].push(e);
66          adj[w].push(e);
67          edgeNum++;
68      }
69
70      public Iterable<FlowEdge> edges() {
71          LinkedList<FlowEdge> list = new LinkedList<>();
72          for (int v = 0; v < vertexNum; v++) {
73              for (FlowEdge e: adj[v]) {
74                  if (e.to() != v) {
75                      list.push(e);
76                  }
77              }
78          }
79
80          return list;
81      }
82
```

```
83      public Iterable<FlowEdge> adj(int v) {
84          validateVertex(v);
85          return adj[v];
86      }
87
88      /** adj[v] = adjacency list for vertex v. */
89      public int getVertexNum() {
90          return vertexNum;
91      }
92
93      public int getEdgeNum() {
94          return edgeNum;
95      }
96
97      @Override
98      public String toString() {
99          StringBuilder sb = new StringBuilder();
100         sb.append(vertexNum + " vertices, " + edgeNum + " edges \n");
101         for (int v = 0; v < vertexNum; v++) {
102             sb.append(v + ": ");
103             for (FlowEdge e : adj[v]) {
104                 if (e.to() != v) {
105                     sb.append(e + " ");
106                 }
107             }
108             sb.append('\n');
109         }
110
111         return sb.toString();
112     }
113
114     /**
115      * throw an IllegalArgumentException unless 0 <= v < V.
116      *
117      * @param given integer
118      */
119     private void validateVertex(int v) {
120         if (v < 0 || v >= vertexNum) {
121             throw new IllegalArgumentException("vertex " + v + " is not between 0 and " + (vertexNum - 1));
122         }
123     }
124 }
```

2.4 最优整数解特性和幺模矩阵

2.5 多商品网络流问题

2.6 多商品流运输问题

2.7 设施选址问题

2.8 旅行商问题

2.9 车辆路径规划问题

第 3 章　整数规划建模技巧

第 4 章　大规模线性规划的对偶

第 II 部分

常用优化求解器 API 详解及应用案例

第 5 章　CPLEX 的 Java API 详解及简单案例

第 6 章　Gurobi 的 Python API 详解及简单案例

第 7 章　调用 CPLEX 和 Gurobi 求解 MIP 的复杂案例：VRPTW 和 TSP

第 III 部分

运筹优化常用算法及实战

第 8 章　单纯形法

第 9 章　Dijkstra 算法

9.1 Dijkstra 算法求解最短路问题详解

9.2 Dijkstra 算法步骤及伪代码

9.3 Python 实现 Dijkstra 算法

9.4 Java 实现 Dijkstra 算法

本部分的完整代码如下。

9.4.1 DijkstraSp.java

DijkstraSp.java

```java
package shortestpath;

import java.io.IOException;
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.PriorityQueue;
import java.util.Queue;

import graph.DirectedEdge;
import graph.EdgeWeightedDigraph;


/**
 * Dijkstra algorithm in a edge-weighted digraph.
 *
 * @author Xiong Wanggi
 * @version V1.0
 * @since JDK1.8
 */
public class DijkstraSp {
    /** distTo[v] = distance of shortest path from source vertex to v. */
    private double[] distTo;
    /** edgeTo[v] = last edge on shortest s->v path. */
    private DirectedEdge[] edgeTo;
    /** queue of vertices to relax. */
    private Queue<Integer> pq;

    /**
     * compute shortest paths from source vertex to other vertices.
     *
     * @param g the edge-weighted digraph
     * @param s source vertex
     */
    public DijkstraSp(EdgeWeightedDigraph g, int s) {
        for (DirectedEdge e : g.edges()) {
            if (e.getWeight() < 0) {
                throw new IllegalArgumentException("edge " + e + " has negative weight");
            }
        }

        int vertexNum = g.getVertexNum();
```

```
42          distTo = new double[vertexNum];
43          edgeTo =  new DirectedEdge[vertexNum];
44
45          validateVertex(s);
46
47          // initalize the distTo
48          for (int v = 0; v < vertexNum; v++) {
49              distTo[v] = Double.POSITIVE_INFINITY;
50          }
51          distTo[s] = 0.0;
52
53          /**
54           *  Dijkstra algorithm:
55           *  1 Consider vertices(non-tree vertex) in increasing order of distance from s
56           *  2 Choose the vertex with the lowest distTo[] value) among vertices above
57           *  3 Add vertex to tree and relax all edges pointing from that vertex
58           */
59          // 注意这里的比较器 Comparator 的写法
60          pq = new PriorityQueue<Integer>(vertexNum, (v, w) -> Double.compare(distTo[v], distTo[w]));
61          pq.offer(s);
62          while (!pq.isEmpty()) {
63              int v = pq.poll();
64              for (DirectedEdge e: g.adj(v)) {
65                  relax(e);
66              }
67          }
68
69      }
70
71      /**
72       * Is there a path from source vertex to vertex {@code v}?.
73       *
74       * @param v the vertex
75       * @return {@code true} if there is a path from source vertex to vertex {@code v}, or {@code false} otherwise
76       */
77      public boolean hasPathTo(int v) {
78          validateVertex(v);
79          return distTo[v] < Double.POSITIVE_INFINITY;
80      }
81
82      /**
83       * Return a shortest path between source vertex and vertex {@code v}, or {@code null} if no such path.
84       *
85       * @param v the vertex
86       * @return shortest path between source vertex and vertex {@code v} as an Iterable, or {@code null} if no such
    ↪  path.
87       */
88      public Iterable<DirectedEdge> pathTo(int v) {
89          validateVertex(v);
90
91          if (!hasPathTo(v)) {
92              return null;
93          }
94
95          Deque<DirectedEdge> path = new ArrayDeque<DirectedEdge>();
96          for (DirectedEdge e = edgeTo[v]; e != null; e = edgeTo[e.from()]) {
97              path.push(e);
98          }
99          return path;
```

```
100     }
101
102     public double distTo(int v) {
103         validateVertex(v);
104         return distTo[v];
105     }
106
107     /**
108      * relax edge e and update {@link #pq} if changed.
109      *
110      * @param e directed edge
111      */
112     private void relax(DirectedEdge e) {
113         int v = e.from();
114         int w = e.to();
115
116         if (distTo[w] > distTo[v] + e.getWeight()) {
117             // 更新 distTo[w]，非常重要，后续会根据 distTo[w] 更新队列
118             distTo[w] = distTo[v] + e.getWeight();
119             edgeTo[w] = e;
120
121             /**
122              * 如果 w 是否在队列中，更新 w 在队列中的位置（先移除再插入）；否则，直接将 w 添加到队列中
123              * remove 函数：如果 w 是否在队列中，删除；否则不进行任何操作
124              */
125             pq.remove(w);
126             pq.offer(w);
127         }
128
129     }
130
131     private void validateVertex(int v) {
132         int vertexNum = distTo.length;
133         if (v < 0 || v >= vertexNum) {
134             throw new IllegalArgumentException("vertex " + v + " is not between 0 and " + (vertexNum - 1));
135         }
136     }
137
138     public static void main(String[] args) {
139         // filename of the graph
140         String filename = "./data/tinyEWD.txt";
141         // source vertex
142         int s = 7;
143
144         EdgeWeightedDigraph g = null;
145         try {
146             g = new EdgeWeightedDigraph(filename);
147         } catch (IOException e) {
148             e.printStackTrace();
149         }
150         DijkstraSp sp = new DijkstraSp(g, s);
151
152         int n = g.getVertexNum();
153         StringBuilder sb = new StringBuilder();
154         for (int t = 0; t < n; t++) {
155             if (sp.hasPathTo(t)) {
156                 sb.append(String.format("%d to %d (%.2f):  ", s, t, sp.distTo(t)));
157                 for (DirectedEdge e : sp.pathTo(t)) {
158                         sb.append(e + "   ");
```

```
159                 }
160                 sb.append('\n');
161             } else {
162                 sb.append(String.format("%d to %d:  not connected\n", s, t));
163             }
164 
165         }
166         System.out.print(sb.toString());
167     }
168 
169 }
```

9.4.2 DirectedEdge.java

DirectedEdge.java

```
1  //package graph;
2  
3  /**
4   * 有向加权边.
5   *
6   * @author Xiong Wangqi
7   * @version V1.0
8   * @since JDK1.8
9   */
10 public class DirectedEdge {
11     private final int from;
12     private final int to;
13     private final double weight;
14 
15     public DirectedEdge(int from, int to, double weight) {
16         this.from = from;
17         this.to = to;
18         this.weight = weight;
19     }
20 
21     public int from() {
22         return from;
23     }
24 
25     public int to() {
26         return to;
27     }
28 
29     public double getWeight() {
30         return weight;
31     }
32 
33     @Override
34     public String toString() {
35         return from + "->" + to + " " + String.format("%5.2f", weight);
36     }
37 }
```

9.4.3 EdgeWeightedDigraph.java

EdgeWeightedDigraph.java

```java
package graph;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.LinkedList;
import java.util.Scanner;


/**
 * edge-weighted digraph.
 *
 * @author Xiong Wanggi
 * @version V1.0
 * @since JDK1.8
 */
public class EdgeWeightedDigraph {
    private final int vertexNum;
    private int edgeNum;
    /** adj[v] = adjacency list for vertex v. */
    private LinkedList<DirectedEdge>[] adj;

    @SuppressWarnings("unchecked")
    public EdgeWeightedDigraph(int vertexNum) {
        if (vertexNum < 0) {
            throw new IllegalArgumentException("Number of vertices in a Digraph must be nonnegative");
        }
        this.vertexNum = vertexNum;

        edgeNum = 0;
        adj = new LinkedList[vertexNum];
        for (int v = 0; v < vertexNum; v++) {
            adj[v] = new LinkedList<>();
        }

    }

    public EdgeWeightedDigraph(EdgeWeightedDigraph g) {
        this(g.vertexNum);
        this.edgeNum = g.edgeNum;

        for (int v = 0; v < vertexNum; v++) {
            // reverse so that adjacency list is in same order as original
            Deque<DirectedEdge> reverse = new ArrayDeque<>();
            for (DirectedEdge e: g.adj[v]) {
                reverse.push(e);
            }

            for (DirectedEdge e: reverse) {
                adj[v].push(e);
            }
        }
    }

    @SuppressWarnings("unchecked")
```

```java
    public EdgeWeightedDigraph(String filename) throws FileNotFoundException {
        Scanner scanner = new Scanner(new FileInputStream(filename));

        // 前两行为节点数和边数
        vertexNum = scanner.nextInt();
        adj = new LinkedList[vertexNum];
        for (int i = 0; i < vertexNum; i++) {
            adj[i] = new LinkedList<>();
        }

        int n = scanner.nextInt();
        for (int i = 0; i < n; i++) {
            int from = scanner.nextInt();
            int to = scanner.nextInt();
            double weight = scanner.nextDouble();
            addEdge(new DirectedEdge(from, to, weight));
        }

        scanner.close();
    }

    public void addEdge(DirectedEdge e) {
        int from = e.from();
        int to = e.to();
        validateVertex(from);
        validateVertex(to);

        adj[from].push(e);
        edgeNum++;
    }

    /**
     * Returns the edges incident on vertex v.
     *
     * @param  v the vertex
     * @return the edges incident on vertex v as an Iterable
     */
    public Iterable<DirectedEdge> adj(int v) {
        validateVertex(v);
        return adj[v];
    }

    /**
     * Returns all edges in this edge-weighted graph as an Iterable.
     *
     * @return all edges in this edge-weighted graph as an iterable
     */
    public Iterable<DirectedEdge> edges() {
        LinkedList<DirectedEdge> list = new LinkedList<>();
        for (int v = 0; v < vertexNum; v++) {
            for (DirectedEdge e: adj[v]) {
                list.push(e);
            }
        }

        return list;
    }

    public int getVertexNum() {
```

```
116         return vertexNum;
117     }
118
119     public int getEdgeNum() {
120         return edgeNum;
121     }
122
123     @Override
124     public String toString() {
125         StringBuilder sb = new StringBuilder();
126         sb.append(vertexNum + " vertices, " + edgeNum + " edges \n");
127         for (int v = 0; v < vertexNum; v++) {
128             sb.append(v + ": ");
129             for (DirectedEdge e : adj[v]) {
130                 sb.append(e + " ");
131             }
132             sb.append('\n');
133         }
134
135         return sb.toString();
136     }
137
138     private void validateVertex(int v) {
139         if (v < 0 || v >= vertexNum) {
140             throw new IndexOutOfBoundsException("vertex " + v + " is not between 0 and " + (vertexNum - 1));
141         }
142     }
143
144 }
```

9.5 Java 实现 BellmanFordSp 求解 SPP

9.5.1 BellmanFordSp.java

BellmanFordSp.java

```
1  package shortestpath;
2
3  import java.io.IOException;
4  import java.util.ArrayDeque;
5  import java.util.Deque;
6  import java.util.LinkedList;
7  import java.util.Queue;
8
9  import graph.DirectedEdge;
10 import graph.EdgeWeightedDigraph;
11 import graph.EdgeWeightedDirectedCycle;
12
13
14 /**
15  * Queue-based implementation of BellmanFord algorithm in a edge-weighted digraph, and negative cycle is considered.
16  *
17  * @author Xiong Wangqi
18  * @version V1.0
19  * @since JDK1.8
20  */
21 public class BellmanFordSp {
```

```
22      /** distTo[v] = distance of shortest path from source vertex to v. */
23      private double[] distTo;
24      /** edgeTo[v] = last edge on shortest s->v path. */
25      private DirectedEdge[] edgeTo;
26
27      /** queue of vertices to relax. */
28      private Queue<Integer> queue;
29      /** onQueue[v] = is v currently on the queue. */
30      private boolean[] onQueue;
31
32      /** number of calls to relax(). */
33      private int cost;
34      /** negative cycle (or null if no such cycle). */
35      private Iterable<DirectedEdge> cycle;
36
37      /**
38       * Computes a shortest paths tree from source vertex to every other vertex in the edge-weighted digraph: <br>
39       * Repeat V times: relax all E edges(queue is used to improve the algorithm).
40       *
41       * @param g the edge-weighted digraph
42       * @param s source vertex
43       */
44      public BellmanFordSp(EdgeWeightedDigraph g, int s) {
45          int vertexNum = g.getVertexNum();
46          distTo = new double[vertexNum];
47          edgeTo = new DirectedEdge[vertexNum];
48          onQueue = new boolean[vertexNum];
49          for (int v = 0; v < vertexNum; v++) {
50              distTo[v] = Double.POSITIVE_INFINITY;
51          }
52          distTo[s] = 0.0;
53
54          /**
55           * Bellman-Ford algorithm, repeat V times: relax all E edges
56           * If distTo[v] does not change during pass i, no need to relax any edge pointing from v in pass i+1
57           * Maintain queue of vertices whose distTo[] changed
58           */
59          queue = new LinkedList<>();
60          queue.offer(s);
61          onQueue[s] = true;
62          while (!queue.isEmpty() && !hasNegativeCycle()) {
63              int v = queue.poll();
64              onQueue[v] = false;
65              relax(g, v);
66          }
67      }
68
69      public boolean hasNegativeCycle() {
70          return cycle != null;
71      }
72
73      public Iterable<DirectedEdge> negativeCycle() {
74          return cycle;
75      }
76
77      /**
78       * Is there a path from source vertex to vertex {@code v}?.
79       *
80       * @param v the vertex
```

```java
     * @return {@code true} if there is a path from source vertex to vertex {@code v}, or {@code false} otherwise
     */
    public boolean hasPathTo(int v) {
        validateVertex(v);
        return distTo[v] < Double.POSITIVE_INFINITY;
    }

    /**
     * Return the shortest path between source vertex and vertex {@code v}
     *
     * @param v the vertex
     * @return shortest path between source vertex and vertex {@code v} as an Iterable, or {@code null} if no such path.
     */
    public Iterable<DirectedEdge> pathTo(int v) {
        validateVertex(v);

        if (hasNegativeCycle()) {
            throw new UnsupportedOperationException("Negative cost cycle exists");
        }

        if (!hasPathTo(v)) {
            return null;
        }

        Deque<DirectedEdge> path = new ArrayDeque<DirectedEdge>();
        for (DirectedEdge e = edgeTo[v]; e != null; e = edgeTo[e.from()]) {
            path.push(e);
        }
        return path;
    }

    public double distTo(int v) {
        validateVertex(v);
        if (hasNegativeCycle()) {
            throw new UnsupportedOperationException("Negative cost cycle exists");
        }

        return distTo[v];
    }

    private void relax(EdgeWeightedDigraph g, int v) {
        // TODO 完成松弛操作
        for (DirectedEdge e: g.adj(v)) {
            int w = e.to();
            if (distTo[w] > distTo[v] + e.getWeight()) {
                distTo[w] = distTo[v] + e.getWeight();
                edgeTo[w] = e;

                if (!onQueue[w]) {
                    queue.add(w);
                    onQueue[w] = true;
                }
            }

            // If any vertex v is updated in pass V, there exists a negative cycle
            // So, if we can find a cycle in the digraph based on edgeTo[v] entries, then there is a negative cycle
            // We can check for negative cycles more frequently, every vertexNum call of relax for example
            cost++;
```

```
            if (cost % g.getVertexNum() == 0) {
                findNegativeCycle();
                if (hasNegativeCycle()) {
                    return;
                }
            }

        }
    }

    /**
     * finding a cycle in predecessor graph.
     */
    private void findNegativeCycle() {
        int vertexNum = distTo.length;
        // predecessor graph
        EdgeWeightedDigraph spt = new EdgeWeightedDigraph(vertexNum);
        for (int v = 0; v < vertexNum; v++) {
            if (edgeTo[v] != null) {
                spt.addEdge(edgeTo[v]);
            }
        }

        EdgeWeightedDirectedCycle finder = new EdgeWeightedDirectedCycle(spt);
        cycle = finder.cycle();
    }

    private void validateVertex(int v) {
        int vertexNum = distTo.length;
        if (v < 0 || v >= vertexNum) {
            throw new IllegalArgumentException("vertex " + v + " is not between 0 and " + (vertexNum - 1));
        }
    }

    public static void main(String[] args) {
        // filename of the graph
        // tinyEWDn.txt has negative edges
        // tinyEWDnc.txt has negative cycle
        String filename = "./data/tinyEWDnc.txt";
        // source vertex
        int s = 0;

        EdgeWeightedDigraph g = null;
        try {
            g = new EdgeWeightedDigraph(filename);
        } catch (IOException e) {
            e.printStackTrace();
        }
        BellmanFordSp sp = new BellmanFordSp(g, s);

        if (sp.hasNegativeCycle()) {
            StringBuilder sb = new StringBuilder("The graph contains negative cycle:\n");
            for (DirectedEdge e: sp.negativeCycle()) {
                sb.append(e);
                sb.append('\n');
            }
            System.out.print(sb.toString());
            return;
        }
```

```
198
199         int n = g.getVertexNum();
200         StringBuilder sb = new StringBuilder();
201         for (int t = 0; t < n; t++) {
202             if (sp.hasPathTo(t)) {
203                 sb.append(String.format("%d to %d (%.2f):  ", s, t, sp.distTo(t)));
204                 for (DirectedEdge e : sp.pathTo(t)) {
205                         sb.append(e + "   ");
206                 }
207                 sb.append('\n');
208             } else {
209                 sb.append(String.format("%d to %d:  not connected\n", s, t));
210             }
211
212         }
213         System.out.print(sb.toString());
214     }
215
216 }
```

9.5.2 EdgeWeightedDirectedCycle.java

EdgeWeightedDirectedCycle.java

```
1  package graph;
2
3  import java.util.Stack;
4
5  /**
6   * Detect whether there is cycle in the edge-weighted digraph.
7   *
8   * @author Xiong Wanggi
9   * @version V1.0
10  * @since JDK1.8
11  */
12 public class EdgeWeightedDirectedCycle {
13     /** marked[v] = has vertex v been marked?. */
14     private boolean[] marked;
15     /** edgeTo[v] = previous edge on path to v. */
16     private DirectedEdge[] edgeTo;
17     /** directed cycle (or null if no such cycle). */
18     private Stack<DirectedEdge> cycle;
19     /** onStack[v] = is vertex on the stack?. */
20     private boolean[] onStack;
21
22     public EdgeWeightedDirectedCycle(EdgeWeightedDigraph g) {
23         int vertexNum = g.getVertexNum();
24         marked = new boolean[vertexNum];
25         onStack = new boolean[vertexNum];
26         edgeTo = new DirectedEdge[vertexNum];
27
28         for (int v = 0; v < vertexNum; v++) {
29             if (!marked[v]) {
30                 dfs(g, v);
31             }
32         }
33     }
34
35     public boolean hasCycle() {
```

```
36          return cycle != null;
37      }
38
39      public Iterable<DirectedEdge> cycle() {
40          return cycle;
41      }
42
43      /**
44       * check that algorithm computes either the topological order or finds a directed cycle.
45       *
46       * @param g edge-weighted digraph
47       * @param v vertex to start depth first search
48       */
49      private void dfs(EdgeWeightedDigraph g, int v) {
50          onStack[v] = true;
51          marked[v] = true;
52
53          for (DirectedEdge e: g.adj(v)) {
54              if (cycle != null) {
55                  return;
56              }
57
58              int w = e.to();
59              // found new vertex, so recur
60              if (!marked[w]) {
61                  edgeTo[w] = e;
62                  dfs(g, w);
63              }
64
65              // trace back directed cycle
66              else if (onStack[w]) {
67                  cycle = new Stack<>();
68                  while (e.from() != w) {
69                      cycle.push(e);
70                      e = edgeTo[e.from()];
71                  }
72                  cycle.push(e);
73                  return;
74              }
75
76          }
77
78          onStack[v] = false;
79      }
80
81  }
```

9.6 Java 实现 FloydSp 求解 SPP

9.6.1 FloydSp.java

FloydSp.java

```
1  package shortestpath;
2
3  import java.io.IOException;
4  import java.util.ArrayDeque;
```

```java
import java.util.Deque;

import graph.DirectedEdge;
import graph.EdgeWeightedDigraph;
import graph.EdgeWeightedDirectedCycle;

/**
 * Floyd-Warshall algorithm to find all-pairs shortest path.
 *
 * @author Xiong Wangqi
 * @version V1.0
 * @since JDK1.8
 */
public class FloydSp {
    private boolean hasNegativeCycle;
    /** distTo[v][w] = length of shortest v->w path. */
    private double[][] distTo;
    /** edgeTo[v][w] = last edge on shortest v->w path. */
    private DirectedEdge[][] edgeTo;


    public FloydSp(EdgeWeightedDigraph g) {
        // TODO Auto-generated constructor stub
        int vertexNum = g.getVertexNum();
        distTo = new double[vertexNum][vertexNum];
        edgeTo = new DirectedEdge[vertexNum][vertexNum];

        // initialize distances to infinity
        for (int v = 0; v < vertexNum; v++) {
            for (int w = 0; w < vertexNum; w++) {
                distTo[v][w] = Double.POSITIVE_INFINITY;
            }
        }

        // initialize distances between end points by using edge weight
        for (int v = 0; v < vertexNum; v++) {
            for (DirectedEdge e: g.adj(v)) {
                distTo[e.from()][e.to()] = e.getWeight();
                edgeTo[e.from()][e.to()] = e;
            }

            // in case of self-loops
            if (distTo[v][v] >= 0) {
                distTo[v][v] = 0.0;
                edgeTo[v][v] = null;
            }
        }

        // Floyd algorithm
        for (int i = 0; i < vertexNum; i++) {
            // compute shortest paths using only 0, 1, ..., i as intermediate vertices
            for (int v = 0; v < vertexNum; v++) {
                if (edgeTo[v][i] == null) {
                    continue;
                }
                for (int w = 0; w < vertexNum; w++) {
                    if (distTo[v][w] > distTo[v][i] + distTo[i][w]) {
                        distTo[v][w] = distTo[v][i] + distTo[i][w];
                        edgeTo[v][w] = edgeTo[i][w];
```

```
                    }
                }

                if (distTo[v][v] < 0.0) {
                    hasNegativeCycle = true;
                    return;
                }
            }

        }

    }

    public boolean hasNegativeCycle() {
        return hasNegativeCycle;
    }

    public Iterable<DirectedEdge> negativeCycle() {
        for (int v = 0; v < distTo.length; v++) {
            // negative cycle in v's predecessor graph
            if (distTo[v][v] < 0.0) {
                int vertexNum = edgeTo.length;
                EdgeWeightedDigraph spt = new EdgeWeightedDigraph(vertexNum);
                for (int w = 0; w < vertexNum; w++) {
                    if (edgeTo[v][w] != null) {
                        spt.addEdge(edgeTo[v][w]);
                    }
                }

                EdgeWeightedDirectedCycle finder = new EdgeWeightedDirectedCycle(spt);
                return finder.cycle();
            }
        }
        return null;
    }

    /**
     * Is there a path from the vertex {@code s} to vertex {@code t}?.
     *
     * @param s the source vertex
     * @param t the destination vertex
     * @return {@code true} if there is a path from vertex {@code s} to vertex {@code t}, or {@code false}
↪   otherwise.
     */
    public boolean hasPath(int s, int t) {
        validateVertex(s);
        validateVertex(t);

        return distTo[s][t] < Double.POSITIVE_INFINITY;
    }

    /**
     * Return the shortest path between {@link #s} and vertex {@code v}
     *
     * @param s the source vertex
     * @param t the destination vertex
     * @return shortest path between {@code s} and {@code v} as an Iterable, or {@code null} if no such path.
     */
    public Iterable<DirectedEdge> path(int s, int t) {
```

```
122             validateVertex(s);
123             validateVertex(t);
124
125             if (hasNegativeCycle()) {
126                 throw new UnsupportedOperationException("Negative cost cycle exists");
127             }
128
129             if (!hasPath(s, t)) {
130                 return null;
131             }
132
133             Deque<DirectedEdge> path = new ArrayDeque<DirectedEdge>();
134             for (DirectedEdge e = edgeTo[s][t]; e != null; e = edgeTo[s][e.from()]) {
135                 path.push(e);
136             }
137             return path;
138         }
139
140         public double dist(int s, int t) {
141             validateVertex(s);
142             validateVertex(t);
143
144             if (hasNegativeCycle()) {
145                 throw new UnsupportedOperationException("Negative cost cycle exists");
146             }
147
148             return distTo[s][t];
149         }
150
151         private void validateVertex(int v) {
152             int vertexNum = distTo.length;
153             if (v < 0 || v >= vertexNum) {
154                 throw new IllegalArgumentException("vertex " + v + " is not between 0 and " + (vertexNum - 1));
155             }
156         }
157
158         public static void main(String[] args) {
159             // filename of the graph
160             // tinyEWD.txt has no negative edges
161             // tinyEWDn.txt has negative edges
162             // tinyEWDnc.txt has negative cycle
163             String filename = "./data/tinyEWD.txt";
164
165             EdgeWeightedDigraph g = null;
166             try {
167                 g = new EdgeWeightedDigraph(filename);
168             } catch (IOException e) {
169                 e.printStackTrace();
170             }
171             FloydSp sp = new FloydSp(g);
172
173             if (sp.hasNegativeCycle()) {
174                 StringBuilder sb = new StringBuilder("The graph contains negative cycle:\n");
175                 for (DirectedEdge e: sp.negativeCycle()) {
176                     sb.append(e);
177                     sb.append('\n');
178                 }
179                 System.out.print(sb.toString());
180                 return;
```

```
        }

        // print all-pairs shortest paths
        int n = g.getVertexNum();
        StringBuilder sb = new StringBuilder();
        for (int s = 0; s < n; s++) {
            for (int t = 0; t < n; t++) {
                if (sp.hasPath(s, t)) {
                    sb.append(String.format("%d to %d (%.2f):  ", s, t, sp.dist(s, t)));
                    for (DirectedEdge e : sp.path(s, t)) {
                            sb.append(e + "   ");
                    }
                    sb.append('\n');
                } else {
                    sb.append(String.format("%d to %d:  not connected\n", s, t));
                }

            }
            sb.append('\n');
        }
        System.out.print(sb.toString());

    }

}
```

9.7 拓展

第 10 章　分支定界算法

我们对线性规划都非常熟悉了，但是实际生产活动中，许多问题都是离散的，因此我们更常见的是整数规划或者混合整数规划。从本章开始，我们就聚焦在混合整数规划的精确求解算法及其实现上。本书介绍的所有精确算法中，分支定界算法是最基本，最底层的算法，其他若干算法都是以分支定界算法为基础的拓展。分支定界算法由伦敦政治经济学院 Ailsa Land 和 Alison Doig 于 1960 年提出 (Land and Doig 1960)，但是当时她们并没有称其为分支定界算法。而后 John D. C. Little 等在 1963 年发表的关于 TSP 的研究中，首次使用了 Branch and Bound Method 的名称 (Little et al. 1963)。到了今天，分支定界算法已经成为运筹优化领域最著名的算法之一。

10.1 整数规划和混合整数规划

10.2 分支定界算法求解混合整数规划

10.3 分支定界算法的一般步骤和伪代码

10.4 分支定界算法执行过程的直观展示

10.5 分支定界算法的分支策略

10.6 分支定界算法的搜索策略

10.7 分支定界算法的剪枝策略

10.8 Python 调用 Gurobi 实现分支定界算法的简单案例

为方便读者理解，我们首先以前文中的简单整数规划为例，来实现 Branch and bound，然后再以 VRPTW 为例，介绍 Branch and Bound 求解复杂 (混合) 整数规划的代码实现。

整数规划模型如下所示。

$$\max\ Z = 100x_1 + 150x_2 \tag{10.8.1}$$

$$s.t.\quad 2x_1 + x_2 \leqslant 10 \tag{10.8.2}$$

$$3x_1 + 6x_2 \leqslant 40 \tag{10.8.3}$$

$$x_1, x_2 \geqslant 0 \tag{10.8.4}$$

$$x_1, x_2 \text{ integer} \tag{10.8.5}$$

下面我们给出 Branch and Bound 算法求解上述整数规划模型的 Python 代码 (调用 Gurobi)。为了方便读者理解，我们首先给出下面代码的详细伪代码。该伪代码与前面章节介绍的伪代码大体相同，只是细节上略有改动。

Algorithm 1 Branch and Bound 算法 (简单案例)

初始化: 根据模型 (10.8.1) 构建初始模型, 并根据 IP 的 LP 松弛创建根节点 S
$UB \leftarrow \infty, LB \leftarrow 0$
设置节点集合 $Q \leftarrow \{S\}$, incumbent $x^* \leftarrow$ Null
while Q 非空且 $UB - LB > \epsilon$ **do**
    选择当前节点 $S^i \leftarrow Q$.pop() (对应的模型为 P^i)
    status ← 求解 P^i 的 LP 松弛
    对偶边界 $\bar{z}^i \leftarrow LP$ 的目标函数值
    $x^i(LP) \leftarrow LP$ 的解
    if status不是 Optimal **then**
        根据不可行性剪枝
    else if $z^i \leqslant LB$ **then**
        根据界限剪枝
    else if $x(LP)$ 是整数可行解 **then**
        更新原边界 $LB \leftarrow \bar{z}^i$
        更新 incument $x^* \leftarrow x^i(LP)$
        根据最优性剪枝
    else if status 是 Optimal 且 $x(LP)$ 是小数解 **then**
        将当前解 $x^i(LP)$ 圆整为整数, 即, $x^i_{int}(LP)$
        if 圆整后解对应的目标函数 $x^i_{int}(LP) > LB$ **then**
            更新 $LB \leftarrow x^i_{int}(LP)$ 对应的目标函数
            更新 incument $x^* \leftarrow x^i_{int}(LP)$
        end if
        选择 $x^i(LP)$ 中第一个小数变量作为分支变量
        创建两个子节点 S^i_1 和 S^i_2，其模型为 P^i_1 和 P^i_2
        更新节点集合 $Q \leftarrow Q \cup \{S^i_1, S^2_1\}$
        tempUB ← Q 中所有叶子节点的 LP 松弛的目标函数的最大值
        更新 $UB \leftarrow$ tempUB
    end if
end while
return incumbent x^* optimal

在上述伪代码中，为了快速得到一些可行解，我们在每个小数解的节点执行了向下圆整的操作，向下圆整操作对于该模型来讲一定可以得到一个整数可行解，这对更新 LB 很有帮助。但是请注意，向下圆整获得可行解仅对该算例成立，对于其他问题不一定成立，对于其他问题，读者需要自行设计相应的算法。 另外，为了更快的更新 UB，我们在分支操作后求解了所有叶子节点的线性松弛模型。

下面是完整的 Python 代码。

Branch and bound

```python
# @author: Liu Xinglu
# @e-mail: hsinglul@163.com
# @institute: Tsinghua University
# @date : 2021-3-11

# Branch and Bound Algorithm
# max 100x_1 + 150x_2
# 2x_1 + x_2 <= 10
# 3x_1 + 6x_2 <= 40
# x_1, x_2 >= 0, and integer

# Creat LP
from gurobipy import *
import numpy as np
import copy
import matplotlib.pyplot as plt

RLP = Model('relaxed MIP')
x = {}
for i in range(2):
    x[i] = RLP.addVar(lb = 0
                      , ub = GRB.INFINITY
                      , vtype = GRB.CONTINUOUS
                      , name = 'x_' + str(i)
                      )

RLP.setObjective(100 * x[0] + 150 * x[1], GRB.MAXIMIZE)

RLP.addConstr(2 * x[0] + x[1] <= 10, name = 'c_1')
RLP.addConstr(3 * x[0] + 6 * x[1] <= 40, name = 'c_2')

RLP.optimize()

# Node class
class Node:
    # this class defines the node
    def __init__(self):
        self.local_LB = 0
        self.local_UB = np.inf
        self.x_sol = {}
        self.x_int_sol = {}
        self.branch_var_list = []
        self.model = None
        self.cnt = None
        self.is_integer = False

    def deepcopy_node(node):
        new_node = Node()
        new_node.local_LB = 0
```

```
50          new_node.local_UB = np.inf
51          new_node.x_sol = copy.deepcopy(node.x_sol)
52          new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
53          new_node.branch_var_list = []
54          new_node.model = node.model.copy()
55          new_node.cnt = node.cnt
56          new_node.is_integer = node.is_integer
57
58          return new_node
59
60  # Branch and Bound
61  def Branch_and_bound(RLP):
62      # initialize the initial node
63      RLP.optimize()
64      global_UB = RLP.ObjVal
65      global_LB = 0
66      eps = 1e-3
67      incumbent_node = None
68      Gap = np.inf
69
70      '''
71         Branch and Bound starts
72      '''
73      # creat initial node
74      Queue = []
75      node = Node()
76      node.local_LB = 0
77      node.local_UB = global_UB
78      node.model = RLP.copy()
79      node.model.setParam("OutputFlag", 0)
80      node.cnt = 0
81      Queue.append(node)
82
83      cnt = 0
84      Global_UB_change = []
85      Global_LB_change = []
86      while (len(Queue) > 0 and global_UB - global_LB > eps):
87          # select the current node
88          current_node = Queue.pop()
89          cnt += 1
90
91          # solve the current model
92          current_node.model.optimize()
93          Solution_status = current_node.model.Status
94
95          '''
96          OPTIMAL = 2
97          INFEASIBLE = 3
98          UNBOUNDED = 5
99          '''
100
101         # check whether the current solution is integer and execute prune step
102         '''
103             is_integer : mark whether the current solution is integer solution
104             Is_Pruned  : mark whether the current solution is pruned
105         '''
106         is_integer = True
107         Is_Pruned = False
108         if (Solution_status == 2):
```

```
            for var in current_node.model.getVars():
                current_node.x_sol[var.varName] = var.x
                print(var.VarName, ' = ', var.x)

                # # round the solution to get an integer solution
                current_node.x_int_sol[var.varName] = (int)(var.x)  # round the solution to get an integer solution
                if (abs((int)(var.x) - var.x) >= eps):
                    is_integer = False
                    current_node.branch_var_list.append(var.VarName)  # to record the candidate branch variables

            # update the LB and UB
            if (is_integer == True):
                # For integer solution node, update the LB and UB
                current_node.is_integer = True
                current_node.local_LB = current_node.model.ObjVal
                current_node.local_UB = current_node.model.ObjVal
                if (current_node.local_LB > global_LB):
                    global_LB = current_node.local_LB
                    incumbent_node = Node.deepcopy_node(current_node)
            if (is_integer == False):
                # For integer solution node, update the LB and UB also
                current_node.is_integer = False
                current_node.local_UB = current_node.model.ObjVal
                current_node.local_LB = 0
                for var_name in current_node.x_int_sol.keys():
                    var = current_node.model.getVarByName(var_name)
                    current_node.local_LB += current_node.x_int_sol[var_name] * var.Obj
                if (current_node.local_LB > global_LB or (
                        current_node.local_LB == global_LB and current_node.is_integer == True)):
                    global_LB = current_node.local_LB
                    incumbent_node = Node.deepcopy_node(current_node)
                    incumbent_node.local_LB = current_node.local_LB
                    incumbent_node.local_UB = current_node.local_UB

            '''
                PRUNE step
            '''
            # prune by optimility
            if (is_integer == True):
                Is_Pruned = True

            # prune by bound
            if (is_integer == False and current_node.local_UB < global_LB):
                Is_Pruned = True

            Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
            print('\n ------------ \n', cnt, '\t Gap  = ', Gap, '  %')
        elif (Solution_status != 2):
            # the current node is infeasible or unbound
            is_integer = False
            '''
                PRUNE step
            '''
            # prune by infeasiblity
            Is_Pruned = True
            continue

        '''
          BRANCH STEP
```

```
        '''
        if (Is_Pruned == False):
            # selecte the branch variable
            branch_var_name = current_node.branch_var_list[0]
            left_var_bound = (int)(current_node.x_sol[branch_var_name])
            right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1

            # creat two child nodes
            left_node = Node.deepcopy_node(current_node)
            right_node = Node.deepcopy_node(current_node)

            # creat left child node
            temp_var = left_node.model.getVarByName(branch_var_name)
            left_node.model.addConstr(temp_var <= left_var_bound, name='branch_left_' + str(cnt))
            left_node.model.setParam("OutputFlag", 0)
            left_node.model.update()
            cnt += 1
            left_node.cnt = cnt

            # creat right child node
            temp_var = right_node.model.getVarByName(branch_var_name)
            right_node.model.addConstr(temp_var >= right_var_bound, name='branch_right_' + str(cnt))
            right_node.model.setParam("OutputFlag", 0)
            right_node.model.update()
            cnt += 1
            right_node.cnt = cnt

            Queue.append(left_node)
            Queue.append(right_node)

            # update the global UB, explore all the leaf nodes
            temp_global_UB = 0
            for node in Queue:
                node.model.optimize()
                if(node.model.status == 2):
                    if(node.model.ObjVal >= temp_global_UB):
                        temp_global_UB = node.model.ObjVal

            global_UB = temp_global_UB
            Global_UB_change.append(global_UB)
            Global_LB_change.append(global_LB)

    # all the nodes are explored, update the LB and UB
    global_UB = global_LB
    Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
    Global_UB_change.append(global_UB)
    Global_LB_change.append(global_LB)

    print('\n\n\n\n')
    print('----------------------------------------')
    print('         Branch and Bound terminates     ')
    print('         Optimal solution found          ')
    print('----------------------------------------')
    print('\nFinal Gap  = ', Gap, '  %')
    print('Optimal Solution:', incumbent_node.x_int_sol)
    print('Optimal Obj:', global_LB)

    return incumbent_node, Gap, Global_UB_change, Global_LB_change

```

```
227  # Solve the IP model by branch and bound
228  incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_bound(RLP)
229
230  # plot the results
231  # fig = plt.figure(1)
232  # plt.figure(figsize=(15,10))
233  font_dict = {"family":'Arial',    #"Kaiti",
234         "style":"oblique",
235          "weight":"normal",
236          "color":"green",
237          "size": 20
238         }
239
240  plt.rcParams['figure.figsize'] = (12.0, 8.0) # 单位是 inches
241  plt.rcParams["font.family"] = 'Arial' #"SimHei"
242  plt.rcParams["font.size"] = 16
243
244  x_cor = range(1, len(Global_LB_change) + 1)
245  plt.plot(x_cor, Global_LB_change, label = 'LB')
246  plt.plot(x_cor, Global_UB_change, label = 'UB')
247  plt.legend()
248  plt.xlabel('Iteration', fontdict=font_dict)
249  plt.ylabel('Bounds update', fontdict=font_dict)
250  plt.title('Bounds update during branch and bound procedure \n', fontsize = 23)
251  plt.savefig('Bound_updates.eps')
252  plt.show()
```

算法迭代过程中 Bounds 的更新过程如下图所示。

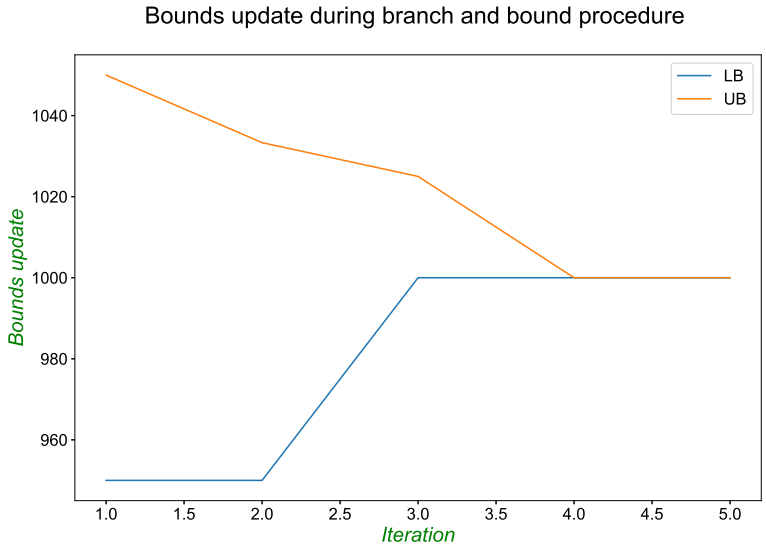


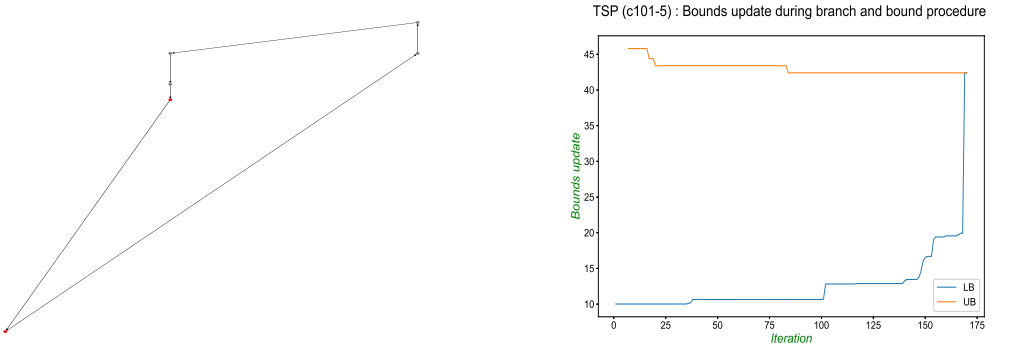
图 10.1: *Bounds 更新过程*

10.9 Python 调用 Gurobi 实现分支定界算法求解 TSP

本小节尝试使用 Branch and bound 求解 TSP。基本的 Branch and bound 求解 TSP 的效率并不高。为了方便，我们仍然用 Solomon VRP Benchmark(Solomon and Marius 1987)

中的数据作为 TSP 的测试数据。

选取 C101 的前 5 个客户点为测试数据，加上仓库点，共 6 个点。求解过程中的 UB 和 LB 的更新如下图所示。



(a) 最优解路径图　　(b) Branch and bound 迭代过程中的 bound 更新

图 10.2: *6 个点的 TSP 算例测试结果*

当客户点为 10 个及以上时，求解就比较困难了。因此 Branch and Bound 求解 TSP 效率并不高。不过本小节的主要目的是学习 Branch and Bound 算法本身，而不是如何提升求解效率。感兴趣的读者可以继续深入探索如何加速求解。关于加快 Branch and Bound 求解 TSP 的求解速度，一个比较可行的加速方法就是利用拉格朗日松弛提供较好的下界，具体介绍见第 12 章。

下面是 Python 调用 Gurobi 求解 TSP 的完整代码。

算例的下载地址为:https://www.sintef.no/projectweb/top/vrptw/100-customers/.

Branch and Bound TSP

```
1
2   #!/usr/bin/env python
3   # coding: utf-8
4
5   #  * author :  Liu Xinglu
6   #  * Institute: Tsinghua University
7   #  * Date : 2020-7-11
8   #  * E-mail : hsinglul@163.com
9
10  # # Python_Call_Gurobi_Solve_TSP
11  # # Prepare Data
12
13  # _*_coding:utf-8 _*_
14  from __future__ import print_function
15  from gurobipy import *
16  import re
17  import math
18  import matplotlib.pyplot as plt
19  import numpy
20  import pandas as pd
21  import networkx as nx
22  import copy
23  import random
24
25  class Data:
```

```
26      def __init__(self):
27          self.customerNum = 0
28          self.nodeNum     = 0
29          self.vehicleNum  = 0
30          self.capacity    = 0
31          self.cor_X       = []
32          self.cor_Y       = []
33          self.demand      = []
34          self.serviceTime = []
35          self.readyTime   = []
36          self.dueTime     = []
37          self.disMatrix   = [[]]   # 读取数据
38          self.arcs        = {}
39
40      # function to read data from .txt files
41      def readData(data, path, customerNum):
42          data.customerNum = customerNum
43          data.nodeNum = customerNum + 1
44          f = open(path, 'r')
45          lines = f.readlines()
46          count = 0
47          # read the info
48          for line in lines:
49              count = count + 1
50              if(count == 5):
51                  line = line[:-1].strip()
52                  str = re.split(r" +", line)
53                  data.vehicleNum = int(str[0])
54                  data.capacity = float(str[1])
55              elif(count >= 10 and count <= 10 + customerNum):
56                  line = line[:-1]
57                  str = re.split(r" +", line)
58                  data.cor_X.append(float(str[2]))
59                  data.cor_Y.append(float(str[3]))
60                  data.demand.append(float(str[4]))
61                  data.readyTime.append(float(str[5]))
62                  data.dueTime.append(float(str[6]))
63                  data.serviceTime.append(float(str[7]))
64
65          # compute the distance matrix
66          data.disMatrix = [([0] * data.nodeNum) for p in range(data.nodeNum)] # 初始化距离矩阵的维度，防止浅拷贝
67          # data.disMatrix = [[0] * nodeNum] * nodeNum]; 这个是浅拷贝，容易重复
68          for i in range(0, data.nodeNum):
69              for j in range(0, data.nodeNum):
70                  temp = (data.cor_X[i] - data.cor_X[j])**2 + (data.cor_Y[i] - data.cor_Y[j])**2
71                  data.disMatrix[i][j] = round(math.sqrt(temp), 1)
72                  temp = 0
73
74          # initialize the arc
75          for i in range(data.nodeNum):
76              for j in range(data.nodeNum):
77                  if(i == j):
78                      data.arcs[i,j] = 0
79                  else:
80                      data.arcs[i,j] = 1
81          return data
82
83      def preprocess(data):
84          # preprocessing for ARCS
```

```
85          # 除去不符合时间窗和容量约束的边
86          for i in range(data.nodeNum):
87              for j in range(data.nodeNum):
88                  if(i == j):
89                      data.arcs[i,j] = 0
90                  elif(data.readyTime[i] + data.serviceTime[i] + data.disMatrix[i][j] > data.dueTime[j]
91                    or data.demand[i] + data.demand[j] > data.capacity):
92                      data.arcs[i,j] = 0
93                  elif(data.readyTime[0] + data.serviceTime[i] + data.disMatrix[0][i] +
                    ↪  data.disMatrix[i][data.nodeNum-1] > data.dueTime[data.nodeNum-1]):
94                      print("the calculating example is false")
95                  else:
96                      data.arcs[i,j] = 1
97          for i in range(data.nodeNum):
98              data.arcs[data.nodeNum - 1, i] = 0
99              data.arcs[i, 0] = 0
100         return data
101
102     def printData(data, customerNum):
103         print(" 下面打印数据\n")
104         print("vehicle number = %4d" % data.vehicleNum)
105         print("vehicle capacity = %4d" % data.capacity)
106         for i in range(len(data.demand)):
107             print('{0}\t{1}\t{2}\t{3}'.format(data.demand[i], data.readyTime[i],data.dueTime[i],
                ↪  data.serviceTime[i]))
108
109         print("-------距离矩阵-------\n")
110         for i in range(data.nodeNum):
111             for j in range(data.nodeNum):
112                 #print("%d   %d" % (i, j));
113                 print("%6.2f" % (data.disMatrix[i][j]), end = " ")
114             print()
115
116 # # Read Data
117 data = Data()
118 path = 'c101.txt'
119 customerNum = 5
120 data = Data.readData(data, path, customerNum)
121 data.vehicleNum = 1
122 Data.printData(data, customerNum)
123 data = Data.preprocess(data)
124
125 # # Build Graph
126 # 构建有向图对象
127 Graph = nx.DiGraph()
128 cnt = 0
129 pos_location = {}
130 nodes_col = {}
131 nodeList = []
132 for i in range(data.nodeNum):
133     X_coor = data.cor_X[i]
134     Y_coor = data.cor_Y[i]
135     name = str(i)
136     nodeList.append(name)
137     nodes_col[name] = 'gray'
138     node_type = 'customer'
139     if(i == 0):
140         node_type = 'depot'
141     Graph.add_node(name
```

```
                , ID = i
                , node_type = node_type
                , time_window = (data.readyTime[i], data.dueTime[i])
                , arrive_time = 10000   # 这个是时间标签 1
                , demand = data.demand
                , serviceTime = data.serviceTime
                , x_coor = X_coor
                , y_coor = Y_coor
                , min_dis = 0           # 这个是距离标签 2
                , previous_node = None  # 这个是前序结点标签 3
                )

    pos_location[name] = (X_coor, Y_coor)
# add edges into the graph
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        if(i == j or (i == 0 and j == data.nodeNum - 1) or (j == 0 and i == data.nodeNum - 1)):
            pass
        else:
            Graph.add_edge(str(i), str(j)
                           , travelTime = data.disMatrix[i][j]
                           , length = data.disMatrix[i][j]
                          )

nodes_col['0'] = 'red'
nodes_col[str(data.nodeNum-1)] = 'red'
plt.rcParams['figure.figsize'] = (10, 10) # 单位是 inches
nx.draw(Graph
        , pos=pos_location
        , with_labels = True
        , node_size = 50
        , node_color = nodes_col.values()   #'y'
        , font_size = 15
        , font_family = 'arial'
#         , edge_color = 'grey'   #'grey'  # b, k, m, g,
        , edgelist = [] #edge_list #[]
#         , nodelist = nodeList
       )
fig_name = 'network_' + str(customerNum) + '_1000.jpg'
plt.savefig(fig_name, dpi=600)
plt.show()

# # getRoute
def getValue(var_dict, nodeNum):
    x_value = np.zeros([nodeNum + 1, nodeNum + 1])
    for key in var_dict.keys():
        a = key[0]
        b = key[1]
        x_value[a][b] = var_dict[key].x

    return x_value
#     '''
#     input: x_value 的矩阵
#     output: 一条路径，[0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 0]，像这样
#     '''
#     # 假如是 5 个点的算例，我们的路径会是 1-4-2-3-5-6 这样的，因为我们加入了一个虚拟点
#     # 也就是当路径长度为 6 的时候，我们就停止，这个长度和 x_value 的长度相同
def getRoute(x_value):
    x = copy.deepcopy(x_value)
```

```
#     route_temp.append(0)
    previousPoint = 0
    route_temp = [previousPoint]
    count = 0
    while(len(route_temp) < len(x)):
        #print('previousPoint: ', previousPoint )
        if(x[previousPoint][count] > 0):
            previousPoint = count
            route_temp.append(previousPoint)
            count = 0
            continue
        else:
            count += 1
    route_temp.append(0)
    return route_temp


# # Build and solve TSP
big_M = data.nodeNum

nodeNum = data.nodeNum
# creat the model
model = Model('TSP')
# creat decision variables
x = {}
mu = {}
for i in range(nodeNum + 1):
    mu[i] = model.addVar(lb = 0.0
                         , ub = 100 #GRB.INFINITY
                          # , obj = distance_initial
                         , vtype = GRB.CONTINUOUS
                         , name = "mu_" + str(i)
                        )

    for j in range(nodeNum + 1):
        if(i != j):
            x[i, j] = model.addVar(vtype = GRB.BINARY
                                   , name = 'x_' + str(i) + '_' + str(j)
                                   )

# set objective function
obj = LinExpr(0)
for key in x.keys():
    i = key[0]
    j = key[1]
    if(i < nodeNum and j < nodeNum):
        obj.addTerms(data.disMatrix[key[0]][key[1]], x[key])
    elif(i == nodeNum):
        obj.addTerms(data.disMatrix[0][key[1]], x[key])
    elif(j == nodeNum):
        obj.addTerms(data.disMatrix[key[0]][0], x[key])

model.setObjective(obj, GRB.MINIMIZE)

# add constraints 1
for j in range(1, nodeNum + 1):
    lhs = LinExpr(0)
    for i in range(0, nodeNum):
        if(i != j):
```

```
260             lhs.addTerms(1, x[i, j])
261     model.addConstr(lhs == 1, name = 'visit_' + str(j))
262 
263 # add constraints 2
264 for i in range(0, nodeNum):
265     lhs = LinExpr(0)
266     for j in range(1, nodeNum + 1):
267         if(i != j):
268             lhs.addTerms(1, x[i, j])
269     model.addConstr(lhs == 1, name = 'visit_' + str(j))
270 
271 # add MTZ constraints
272 # for key in X.keys():
273 #     org = key[0]
274 #     des = key[1]
275 #     if(org != 0 or des != 0):
276 # #         pass
277 #         model.addConstr(mu[org] - mu[des] + 100 * X[key] <= 100 - 1)
278 for i in range(0, nodeNum):
279     for j in range(1, nodeNum + 1):
280         if(i != j):
281             model.addConstr(mu[i] - mu[j] + 100 * x[i, j] <= 100 - 1)
282 
283 # model.setParam('MIPGap', 0)
284 model.write('model.lp')
285 model.optimize()
286 
287 print('Obj:', model.ObjVal)
288 x_value = getValue(x, nodeNum)
289 route = getRoute(x_value)
290 print('optimal route:', route)
291 
292 print("\n\n-----optimal value-----")
293 print(model.ObjVal)
294 
295 edge_list = []
296 for key in x.keys():
297     if(x[key].x > 0):
298         print(x[key].VarName, ' = ', x[key].x)
299         arc = None
300         if((int)(key[1]) == data.nodeNum):
301             arc = (str(key[0]), '0')
302         else:
303             arc = (str(key[0]), str(key[1]))
304 
305         edge_list.append(arc)
306 
307 nodes_col['0'] = 'red'
308 # nodes_col[str(data.nodeNum-1)] = 'red'
309 plt.rcParams['figure.figsize'] = (15, 15) # 单位是 inches
310 nx.draw(Graph
311         , pos=pos_location
312 #         , with_labels = True
313         , node_size = 50
314         , node_color = nodes_col.values()   #'y'
315         , font_size = 15
316         , font_family = 'arial'
317 #         , edge_color = 'grey'   #'grey'  # b, k, m, g,
318         , edgelist = edge_list
```

```
#         , nodelist = nodeList
        )
fig_name = 'TSP_solution_' + str(customerNum) + '_1000.jpg'
plt.savefig(fig_name, dpi=600)
plt.show()

# # Node class
class Node:
    # this class defines the node
    def __init__(self):
        self.local_LB = 0
        self.local_UB = np.inf
        self.x_sol = {}
        self.x_int_sol = {}
        self.branch_var_list = []
        self.model = None
        self.cnt = None
        self.is_integer = False
        self.var_LB = {}
        self.var_UB = {}

    def deepcopy_node(node):
        new_node = Node()
        new_node.local_LB = 0
        new_node.local_UB = np.inf
        new_node.x_sol = copy.deepcopy(node.x_sol)
        new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
        new_node.branch_var_list = []
        new_node.model = node.model.copy()
        new_node.cnt = node.cnt
        new_node.is_integer = node.is_integer

        return new_node

class Node_2:
    # this class defines the node
    def __init__(self):
        self.local_LB = 0
        self.local_UB = np.inf
        self.x_sol = {}
        self.x_int_sol = {}
        self.branch_var_list = []
        self.cnt = None
        self.is_integer = False
        self.var_LB = {}
        self.var_UB = {}

    def deepcopy_node(node):
        new_node = Node()
        new_node.local_LB = 0
        new_node.local_UB = np.inf
        new_node.x_sol = copy.deepcopy(node.x_sol)
        new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
        new_node.branch_var_list = []
        new_node.cnt = node.cnt
        new_node.is_integer = node.is_integer
        new_node.var_LB = copy.deepcopy(node.var_LB)
        new_node.var_UB = copy.deepcopy(node.var_UB)

```

```
378             return new_node
379
380     # # TSP Branch and Bound framework -v1: copy model at node
381     def Branch_and_bound(TSP_model, summary_interval):
382         Relax_TSP_model = TSP_model.relax()
383         # initialize the initial node
384         Relax_TSP_model.optimize()
385         global_UB = np.inf
386         global_LB = Relax_TSP_model.ObjVal
387         eps = 1e-6
388         incumbent_node = None
389         Gap = np.inf
390         feasible_sol_cnt = 0
391
392         '''
393             Branch and Bound starts
394         '''
395
396         # creat initial node
397         Queue = []
398         node = Node()
399         node.local_LB = global_LB
400         node.local_UB = np.inf
401         node.model = Relax_TSP_model.copy()
402         node.model.setParam("OutputFlag", 0)
403         node.cnt = 0
404         Queue.append(node)
405
406         cnt = 0
407         branch_cnt = 0
408         Global_UB_change = []
409         Global_LB_change = []
410         while (len(Queue) > 0 and global_UB - global_LB > eps):
411             # select the current node
412             current_node = Queue.pop()
413             cnt += 1
414
415             # solve the current model
416             current_node.model.optimize()
417             Solution_status = current_node.model.Status
418
419             '''
420             OPTIMAL = 2
421             INFEASIBLE = 3
422             UNBOUNDED = 5
423             '''
424
425             # check whether the current solution is integer and execute prune step
426             '''
427                 is_integer : mark whether the current solution is integer solution
428                 Is_Pruned  : mark whether the current solution is pruned
429             '''
430             is_integer = True
431             Is_Pruned = False
432             if (Solution_status == 2):
433                 for var in current_node.model.getVars():
434                     if(var.VarName.startswith('x')):
435                         current_node.x_sol[var.varName] = copy.deepcopy(current_node.model.getVarByName(var.VarName).x)
436                         # print(var.VarName, ' = ', var.x)
```

```
437                     # record the branchable variable
438                     #  if(abs((int)(var.x) - var.x) >= eps):
439                     if(abs(round(var.x, 0) - var.x) >= eps):
440                     # if(round(var.x, 0) != var.x):  # 如果改成在一定精度范围内，就会出现错误，结果不是最优解
441                         is_integer = False
442                         current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
                            ↪  variables
443 #             print(current_node.x_sol)
444 
445             # update the LB and UB
446             if (is_integer == True):
447                 feasible_sol_cnt +=  1
448                 # For integer solution node, update the LB and UB
449                 current_node.is_integer = True
450                 current_node.local_LB = current_node.model.ObjVal
451                 current_node.local_UB = current_node.model.ObjVal
452                 # if the solution is integer, uodate the UB of global and update the incumbent
453                 if (current_node.local_UB < global_UB):
454                     global_UB = current_node.local_UB
455                     incumbent_node = Node.deepcopy_node(current_node)
456             if (is_integer == False):
457                 # For integer solution node, update the LB and UB also
458                 current_node.is_integer = False
459                 current_node.local_UB = global_UB
460                 current_node.local_LB = current_node.model.ObjVal
461 
462             '''
463                 PRUNE step
464             '''
465             # prune by optimility
466             if (is_integer == True):
467                 Is_Pruned = True
468 
469             # prune by bound
470             if (is_integer == False and current_node.local_LB > global_UB):
471                 Is_Pruned = True
472 
473             Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
474 
475         elif (Solution_status != 2):
476             # the current node is infeasible or unbound
477             is_integer = False
478 
479             '''
480                 PRUNE step
481             '''
482             # prune by infeasiblity
483             Is_Pruned = True
484 
485             continue
486 
487         '''
488           BRANCH STEP
489         '''
490         if (Is_Pruned == False):
491             branch_cnt += 1
492             # selecte the branch variable: choose the value which is closest to 0.5
493             branch_var_name = None
494 
```

```
495             min_diff = 100
496             for var_name in current_node.branch_var_list:
497                 if(abs(current_node.x_sol[var_name] - 0.5) < min_diff):
498                     branch_var_name = var_name
499                     min_diff = abs(current_node.x_sol[var_name] - 0.5)
500 #             branch_var_name = current_node.branch_var_list[0]
501 
502             # choose the variable cloest to 0 or 1
503 #             min_diff = 100
504 #             for var_name in current_node.branch_var_list:
505 #                 diff = max(abs(current_node.x_sol[var_name] - 1), abs(current_node.x_sol[var_name] - 0))
506 #                 if(min_diff >= diff):
507 #                     branch_var_name = var_name
508 #                     min_diff = diff
509 
510 
511 #             branch_var_name = current_node.branch_var_list[0]
512             if(branch_cnt % summary_interval == 0):
513                 print('Branch var name :', branch_var_name, '\t, Branch var value :',
                    ↪  current_node.x_sol[branch_var_name])
514             left_var_bound = (int)(current_node.x_sol[branch_var_name])
515             right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1
516 
517             # creat two child nodes
518             left_node = Node.deepcopy_node(current_node)
519             right_node = Node.deepcopy_node(current_node)
520 
521             # creat left child node
522             temp_var = left_node.model.getVarByName(branch_var_name)
523             left_node.model.addConstr(temp_var <= left_var_bound, name='branch_left_' + str(cnt))
524             left_node.model.setParam("OutputFlag", 0)
525             left_node.model.update()
526             cnt += 1
527             left_node.cnt = cnt
528 
529             # creat right child node
530             temp_var = right_node.model.getVarByName(branch_var_name)
531             right_node.model.addConstr(temp_var >= right_var_bound, name='branch_right_' + str(cnt))
532             right_node.model.setParam("OutputFlag", 0)
533             right_node.model.update()
534             cnt += 1
535             right_node.cnt = cnt
536 
537             Queue.append(left_node)
538             Queue.append(right_node)
539 
540             # update the global LB, explore all the leaf nodes
541             temp_global_LB = np.inf
542             for node in Queue:
543                 node.model.optimize()
544                 if(node.model.status == 2):
545                     if(node.model.ObjVal <= temp_global_LB and node.model.ObjVal <= global_UB):
546                         temp_global_LB = node.model.ObjVal
547 
548             if(temp_global_LB == np.inf):
549                 temp_global_LB = Global_LB_change[-1]
550 
551             global_LB = temp_global_LB
552             Global_UB_change.append(global_UB)
```

```
553                 Global_LB_change.append(global_LB)
554
555             if( (cnt - 2) % summary_interval == 0):
556                 print('\n\n====================')
557                 print('Queue length :', len(Queue))
558                 print('\n ------------ \n', cnt, ' UB  = ', global_UB, '   LB = ', global_LB, '\t Gap  = ', Gap, '  %',
                    ↪  'feasible_sol_cnt :', feasible_sol_cnt)
559
560         # all the nodes are explored, update the LB and UB
561         incumbent_node.model.optimize()
562         global_UB = incumbent_node.model.ObjVal
563         global_LB = global_UB
564         Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
565         Global_UB_change.append(global_UB)
566         Global_LB_change.append(global_LB)
567
568         print('\n\n\n\n')
569         print('----------------------------------------')
570         print('         Branch and Bound terminates     ')
571         print('         Optimal solution found          ')
572         print('----------------------------------------')
573         print('\nIter cnt  = ', cnt, ' \n\n')
574         print('\nFinal Gap  = ', Gap, '  % \n\n')
575         # print('Optimal Solution:', incumbent_node.x_sol)
576         print(' ---- Optimal Solution ------')
577         for key in incumbent_node.x_sol.keys():
578             if(incumbent_node.x_sol[key] > 0):
579                 print(key, ' = ', incumbent_node.x_sol[key])
580         print('\nOptimal Obj:', global_LB)
581
582         return incumbent_node, Gap, Global_UB_change, Global_LB_change
583
584
585     # # Solve the IP model by branch and bound
586     incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_bound(model, summary_interval = 50)
587
588
589     # # TSP Branch and Bound V2:do not copy model at node
590     def Branch_and_bound_v2(TSP_model, summary_interval):
591         Relax_TSP_model = TSP_model.relax()
592         Relax_TSP_model.setParam("OutputFlag", 0)
593         # Relax_TSP_model.setParam("OptimalityTol", 1e-3)
594         # Relax_TSP_model.Params.Presolve = 0
595         # Relax_TSP_model.Params.FeasibilityTol = 1e-2
596         # initialize the initial node
597         Relax_TSP_model.optimize()
598         global_UB = np.inf
599         global_LB = Relax_TSP_model.ObjVal
600         eps = 1e-6
601         eps_bound = 1e-3
602         incumbent_node = None
603         Gap = np.inf
604         feasible_sol_cnt = 0
605
606         '''
607            Branch and Bound starts
608         '''
609
610         # creat initial node
```

```
    Queue = []
    node = Node_2()
    node.local_LB = global_LB
    node.local_UB = np.inf
    node.cnt = 0
    for var in Relax_TSP_model.getVars():
        if(var.VarName.startswith('x')):
            node.var_LB[var.VarName] = 0
            node.var_UB[var.VarName] = 1
    if (Relax_TSP_model.status == 2):
        for var in Relax_TSP_model.getVars():
            if(var.VarName.startswith('x')):
                # print(var.VarName, ' = ', var.x)
                # record the branchable variable
                # if(abs(round(var.x, 0) - var.x) >= eps):
                if(round(var.x, 0) != var.x):
                    node.branch_var_list.append(var.VarName)  # to record the candidate branch variables
                    node.x_sol[var.varName] = var.x
                else:
                    node.x_sol[var.varName] = round(var.x, 0) # 精度问题，直接圆整成整数解

    Queue.append(node)

    cnt = 0
    Global_UB_change = []
    Global_LB_change = []
    while (len(Queue) > 0 and global_UB - global_LB > eps):
        # select the current node
        current_node = Queue.pop()
        cnt += 1

        for var_name in current_node.var_LB.keys():
            Relax_TSP_model.getVarByName(var_name).lb = current_node.var_LB[var_name]
            Relax_TSP_model.getVarByName(var_name).ub = current_node.var_UB[var_name]
        Relax_TSP_model.update()
        # solve the current model
        Relax_TSP_model.optimize()
        Solution_status = Relax_TSP_model.Status

        '''
        OPTIMAL = 2
        INFEASIBLE = 3
        UNBOUNDED = 5
        '''
        # check whether the current solution is integer and execute prune step
        '''
            is_integer : mark whether the current solution is integer solution
            Is_Pruned  : mark whether the current solution is pruned
        '''
        is_integer = True
        Is_Pruned = False
        if (Solution_status == 2):
            for var in Relax_TSP_model.getVars():
                if(var.VarName.startswith('x')):
                    # print(var.VarName, ' = ', var.x)
                    # record the branchable variable
                    if(abs(round(var.x, 0) - var.x) >= eps):
                        is_integer = False
```

```
669                         current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
                            ↪ variables
670                         current_node.x_sol[var.VarName] = var.x
671                     else:
672                         current_node.x_sol[var.VarName] = round(var.x, 0) # 精度问题，直接圆整成整数解
673
674             # update the LB and UB
675             if (is_integer == True):
676                 feasible_sol_cnt +=  1
677                 # For integer solution node, update the LB and UB
678                 current_node.is_integer = True
679                 current_node.local_LB = Relax_TSP_model.ObjVal
680                 current_node.local_UB = Relax_TSP_model.ObjVal
681                 # if the solution is integer, uodate the UB of global and update the incumbent
682                 if (current_node.local_UB < global_UB):
683                     global_UB = current_node.local_UB
684                     incumbent_node = Node_2.deepcopy_node(current_node)
685             if (is_integer == False):
686                 # For integer solution node, update the LB and UB also
687                 current_node.is_integer = False
688                 current_node.local_UB = global_UB
689                 current_node.local_LB = Relax_TSP_model.ObjVal
690
691             '''
692                 PRUNE step
693             '''
694             # prune by optimility
695             if (is_integer == True):
696                 Is_Pruned = True
697
698             # prune by bound
699             if (is_integer == False and current_node.local_LB > global_UB):
700                 Is_Pruned = True
701
702             Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
703
704         elif (Solution_status != 2):
705             # the current node is infeasible or unbound
706             is_integer = False
707
708             '''
709                 PRUNE step
710             '''
711             # prune by infeasiblity
712             Is_Pruned = True
713
714             continue
715
716         '''
717           BRANCH STEP
718         '''
719         if (Is_Pruned == False):
720             # selecte the branch variable: choose the value which is closest to 0.5
721             branch_var_name = None
722 #             min_diff = 100
723 #             for var_name in current_node.branch_var_list:
724 #                 if(abs(current_node.x_sol[var_name] - 0.5) < min_diff):
725 #                     branch_var_name = var_name
726 #                     min_diff = abs(current_node.x_sol[var_name] - 0.5)
```

```
727  #                   print(var_name, ' = ', current_node.x_sol[branch_var_name])
728  #             branch_var_name = current_node.branch_var_list[0]
729
730              # choose the variable cloest to 0 or 1
731              min_diff = 100
732              for var_name in current_node.branch_var_list:
733                  diff = max(abs(current_node.x_sol[var_name] - 1), abs(current_node.x_sol[var_name] - 0))
734                  if(min_diff >= diff):
735                      branch_var_name = var_name
736                      min_diff = diff
737
738              # branch_var_name = current_node.branch_var_list[0]
739              if(cnt % summary_interval == 0):
740                  print('Branch var name :', branch_var_name, '\t, value :', current_node.x_sol[branch_var_name])
741              left_var_bound = (int)(current_node.x_sol[branch_var_name])
742              right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1
743
744              # creat two child nodes
745              left_node = Node_2.deepcopy_node(current_node)
746              right_node = Node_2.deepcopy_node(current_node)
747
748              # creat left child node (update the bound data for left node)
749              left_node.var_LB[branch_var_name] = 0
750              left_node.var_UB[branch_var_name] = 0
751              cnt += 1
752              left_node.cnt = cnt
753
754              # creat right child node (update the bound data for right node)
755              right_node.var_LB[branch_var_name] = 1
756              right_node.var_UB[branch_var_name] = 1
757              cnt += 1
758              right_node.cnt = cnt
759
760              Queue.append(left_node)
761              Queue.append(right_node)
762
763              # update the global LB, explore all the leaf nodes
764              temp_global_LB = np.inf
765              for node in Queue:
766                  for var_name in node.var_LB.keys():
767                      Relax_TSP_model.getVarByName(var_name).lb = node.var_LB[var_name]
768                      Relax_TSP_model.getVarByName(var_name).ub = node.var_UB[var_name]
769
770                  Relax_TSP_model.update()
771                  Relax_TSP_model.optimize()
772                  if(Relax_TSP_model.status == 2):
773                      if(Relax_TSP_model.ObjVal <= temp_global_LB and Relax_TSP_model.ObjVal <= global_UB):
774                          temp_global_LB = Relax_TSP_model.ObjVal
775
776              if(temp_global_LB == np.inf):
777                  temp_global_LB = Global_LB_change[-1]
778
779              global_LB = temp_global_LB
780              Global_UB_change.append(global_UB)
781              Global_LB_change.append(global_LB)
782
783          if(cnt % summary_interval == 0):
784              print('\n\n====================')
785              print('Queue length :', len(Queue))
```

```
786             print('\n ------------ \n', cnt, ' UB  = ', global_UB, '   LB = ', global_LB, '\t Gap  = ', Gap, '  %',
                ↪  'feasible_sol_cnt :', feasible_sol_cnt)
787
788     # all the nodes are explored, update the LB and UB
789     for var_name in incumbent_node.var_LB.keys():
790         Relax_TSP_model.getVarByName(var_name).lb = incumbent_node.var_LB[var_name]
791         Relax_TSP_model.getVarByName(var_name).ub = incumbent_node.var_UB[var_name]
792     Relax_TSP_model.update()
793     Relax_TSP_model.optimize()
794     global_UB = Relax_TSP_model.ObjVal
795     global_LB = global_UB
796     Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
797     Global_UB_change.append(global_UB)
798     Global_LB_change.append(global_LB)
799
800     print('\n\n\n\n')
801     print('-----------------------------------------')
802     print('         Branch and Bound terminates     ')
803     print('         Optimal solution found          ')
804     print('-----------------------------------------')
805     print('\nFinal Gap  = ', Gap, '  %')
806 #     print('Optimal Solution:', incumbent_node.x_sol)
807     print('Optimal Solution: \n')
808     for key in incumbent_node.x_sol.keys():
809         if(incumbent_node.x_sol[key] > 0):
810             print(key, ' = ', incumbent_node.x_sol[key])
811     print('Optimal Obj:', global_LB)
812
813     return incumbent_node, Gap, Global_UB_change, Global_LB_change
814
815 incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_bound_v2(model, summary_interval = 500)
816
817 # # plot the results
818 # fig = plt.figure(1)
819 # plt.figure(figsize=(15,10))
820 font_dict = {"family":'Arial',    #"Kaiti",
821        "style":"oblique",
822         "weight":"normal",
823         "color":"green",
824         "size": 20
825        }
826
827 plt.rcParams['figure.figsize'] = (12.0, 8.0) # 单位是 inches
828 plt.rcParams["font.family"] = 'Arial' #"SimHei"
829 plt.rcParams["font.size"] = 16
830 # plt.xlim(0, len(Global_LB_change) + 1000)
831
832 x_cor = range(1, len(Global_LB_change) + 1)
833 plt.plot(x_cor, Global_LB_change, label = 'LB')
834 plt.plot(x_cor, Global_UB_change, label = 'UB')
835 plt.legend()
836 plt.xlabel('Iteration', fontdict=font_dict)
837 plt.ylabel('Bounds update', fontdict=font_dict)
838 plt.title('TSP (c101-5) : Bounds update during branch and bound procedure \n', fontsize = 23)
839 plt.savefig('Bound_updates_TSP_c101_5.eps')
840 plt.savefig('Bound_updates_TSP_c101_5.pdf')
841 plt.show()
```

10.10 Python 调用 Gurobi 实现分支定界算法求解 VRPTW

本小节尝试用 Branch and bound 求解 VRPTW。本章代码使用深度优先对 BB tree 进行搜索。在每个小数解的节点，选择取值最接近 0.5 的变量 x_{ij} 进行分支。

我们选取 solomon VRP benchmark 中的 C101 的前 10 个点作为算例进行测试，设置车辆数为 2。Branch and bound 算法迭代过程中，UB 和 LB 的迭代如图所示。

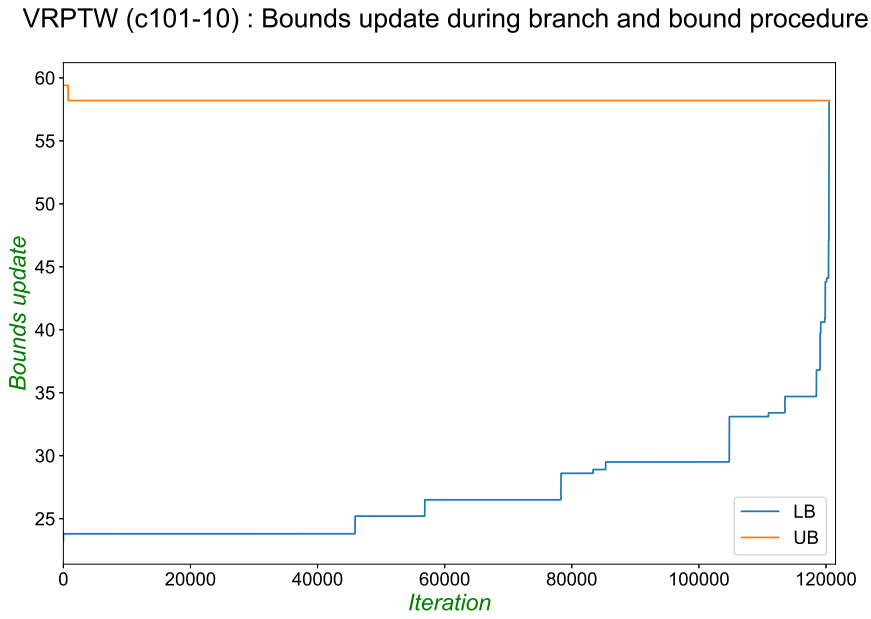


图 10.3: *VRPTW: 算例 c101-10 的 Bounds 更新过程*

由上图可知，最基本的 Branch and bound 求解 VRPTW 是非常困难的，即使是 10 个客户点，2 辆车的小算例，迭代次数也要将近 12 万次，求解时间为 17 分钟左右。此外，算法迭代过程中，LB 更新非常缓慢。

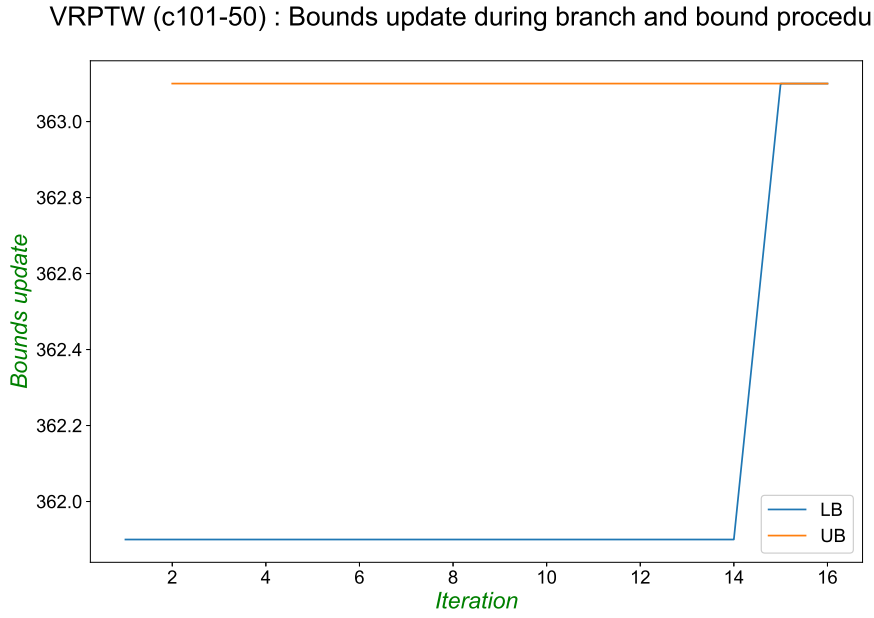


图 10.4: *VRPTW: 算例 c101-50 的 Bounds 更新过程*

为了加速求解，在建模之前，可以先对网络拓扑图进行预处理，删除容量、时间窗明显不满足要求的弧。经过预处理之后，我们发现，基本的 Branch and bound 求解 VRPTW 的效率有显著的提升。就 C101 而言，提取前 25、前 50 个客户点作为测试算例，求解都比较快。对于 100 个客户点的算例，也可以在几小时之内得到最优解。其中 50 个点的算例迭代过程中的 UB 和 LB 更新如下图所示 (设置车辆数为 6)，求解时间仅为 4.3 秒。求解较快的原因是，C 类算例容易求解，预处理之后变量个数明显减少。对于 C101-50，原本的 $0-1$ 变量个数为 $52 \times 51 \times 6 = 15912$ 个，经过预处理之后，缩减为 7308 个。

下面是 Python 调用 Gurobi 实现 Branch and bound 求解 VRPTW 的完整代码。

算例的下载地址为:https://www.sintef.no/projectweb/top/vrptw/100-customers/.

Branch and Bound

```python
#!/usr/bin/env python
# coding: utf-8

#  * author :  Liu Xinglu
#  * Institute: Tsinghua University
#  * Date : 2020-7-11
#  * E-mail : hsinglul@163.com

# # Python_Call_Gurobi_Solve_VRPTW

# # Prepare Data
# _*_coding:utf-8 _*_
from __future__ import print_function
from gurobipy import *
import re
import math
import matplotlib.pyplot as plt
import numpy
import pandas as pd
```

```python
import networkx as nx
import copy
import random

class Data:
    def __init__(self):
        self.customerNum = 0
        self.nodeNum     = 0
        self.vehicleNum  = 0
        self.capacity    = 0
        self.cor_X       = []
        self.cor_Y       = []
        self.demand      = []
        self.serviceTime = []
        self.readyTime   = []
        self.dueTime     = []
        self.disMatrix   = [[]]   # 读取数据
        self.arcs        = {}

    # function to read data from .txt files
    def readData(data, path, customerNum):
        data.customerNum = customerNum
        data.nodeNum = customerNum + 2
        f = open(path, 'r')
        lines = f.readlines()
        count = 0
        # read the info
        for line in lines:
            count = count + 1
            if(count == 5):
                line = line[:-1].strip()
                str = re.split(r" +", line)
                data.vehicleNum = int(str[0])
                data.capacity = float(str[1])
            elif(count >= 10 and count <= 10 + customerNum):
                line = line[:-1]
                str = re.split(r" +", line)
                data.cor_X.append(float(str[2]))
                data.cor_Y.append(float(str[3]))
                data.demand.append(float(str[4]))
                data.readyTime.append(float(str[5]))
                data.dueTime.append(float(str[6]))
                data.serviceTime.append(float(str[7]))

        data.cor_X.append(data.cor_X[0])
        data.cor_Y.append(data.cor_Y[0])
        data.demand.append(data.demand[0])
        data.readyTime.append(data.readyTime[0])
        data.dueTime.append(data.dueTime[0])
        data.serviceTime.append(data.serviceTime[0])


        # compute the distance matrix
        data.disMatrix = [([0] * data.nodeNum) for p in range(data.nodeNum)] # 初始化距离矩阵的维度，防止浅拷贝
        # data.disMatrix = [[0] * nodeNum] * nodeNum]; 这个是浅拷贝，容易重复
        for i in range(0, data.nodeNum):
            for j in range(0, data.nodeNum):
                temp = (data.cor_X[i] - data.cor_X[j])**2 + (data.cor_Y[i] - data.cor_Y[j])**2
                data.disMatrix[i][j] = round(math.sqrt(temp), 1)
```

```
                temp = 0

        # initialize the arc
        for i in range(data.nodeNum):
            for j in range(data.nodeNum):
                if(i == j):
                    data.arcs[i,j] = 0
                else:
                    data.arcs[i,j] = 1
        return data

    def preprocess(data):
        # preprocessing for ARCS
        # 除去不符合时间窗和容量约束的边
        for i in range(data.nodeNum):
            for j in range(data.nodeNum):
                if(i == j):
                    data.arcs[i,j] = 0
                elif(data.readyTime[i] + data.serviceTime[i] + data.disMatrix[i][j] > data.dueTime[j]
                  or data.demand[i] + data.demand[j] > data.capacity):
                    data.arcs[i,j] = 0
                elif(data.readyTime[0] + data.serviceTime[i] + data.disMatrix[0][i] +
                ↪  data.disMatrix[i][data.nodeNum-1] > data.dueTime[data.nodeNum-1]):
                    print("the calculating example is false")
                else:
                    data.arcs[i,j] = 1
        for i in range(data.nodeNum):
            data.arcs[data.nodeNum - 1, i] = 0
            data.arcs[i, 0] = 0
        return data


    def printData(data, customerNum):
        print(" 下面打印数据\n")
        print("vehicle number = %4d" % data.vehicleNum)
        print("vehicle capacity = %4d" % data.capacity)
        for i in range(len(data.demand)):
            print('{0}\t{1}\t{2}\t{3}'.format(data.demand[i], data.readyTime[i],data.dueTime[i],
            ↪  data.serviceTime[i]))

        print("-------距离矩阵-------\n")
        for i in range(data.nodeNum):
            for j in range(data.nodeNum):
                #print("%d   %d" % (i, j));
                print("%6.2f" % (data.disMatrix[i][j]), end = " ")
            print()


# # Read Data
data = Data()
path = 'rc101.txt'
customerNum = 20
data = Data.readData(data, path, customerNum)
data.vehicleNum = 5
# Data.printData(data, customerNum)
# data = Data.preprocess(data)

# # Build Graph
# 构建有向图对象
```

```
Graph = nx.DiGraph()
cnt = 0
pos_location = {}
nodes_col = {}
nodeList = []
for i in range(data.nodeNum):
    X_coor = data.cor_X[i]
    Y_coor = data.cor_Y[i]
    name = str(i)
    nodeList.append(name)
    nodes_col[name] = 'gray'
    node_type = 'customer'
    if(i == 0):
        node_type = 'depot'
    Graph.add_node(name
                , ID = i
                , node_type = node_type
                , time_window = (data.readyTime[i], data.dueTime[i])
                , arrive_time = 10000   # 这个是时间标签 1
                , demand = data.demand
                , serviceTime = data.serviceTime
                , x_coor = X_coor
                , y_coor = Y_coor
                , min_dis = 0           # 这个是距离标签 2
                , previous_node = None  # 这个是前序结点标签 3
                )

    pos_location[name] = (X_coor, Y_coor)
# add edges into the graph
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        if(i == j or (i == 0 and j == data.nodeNum - 1) or (j == 0 and i == data.nodeNum - 1)):
            pass
        else:
            Graph.add_edge(str(i), str(j)
                           , travelTime = data.disMatrix[i][j]
                           , length = data.disMatrix[i][j]
                          )

nodes_col['0'] = 'red'
nodes_col[str(data.nodeNum-1)] = 'red'
plt.rcParams['figure.figsize'] = (10, 10) # 单位是 inches
nx.draw(Graph
        , pos=pos_location
        , with_labels = True
        , node_size = 50
        , node_color = nodes_col.values()   #'y'
        , font_size = 15
        , font_family = 'arial'
#         , edge_color = 'grey'   #'grey'  # b, k, m, g,
        , edgelist = [] #edge_list #[]
#         , nodelist = nodeList
       )

fig_name = 'network_' + str(customerNum) + '_1000.jpg'
plt.savefig(fig_name, dpi=600)
plt.show()

# # Build and solve VRPTW
```

```
195  big_M = 100000
196
197  # creat the model
198  model = Model('VRPTW')
199
200  # decision variables
201  x = {}
202  s = {}
203  x_var = {}
204  for i in range(data.nodeNum):
205      for k in range(data.vehicleNum):
206          name = 's_' + str(i) + '_' + str(k)
207          s[i, k] = model.addVar(lb = data.readyTime[i] # 0 # data.readyTime[i]
208                                 , ub = data.dueTime[i]    #  1e15 # data.dueTime[i]
209                                 , vtype = GRB.CONTINUOUS
210                                 , name = name
211                                 )
212          for j in range(data.nodeNum):
213              if(i != j and data.arcs[i,j] == 1):
214                  name = 'x_' + str(i) + '_' + str(j) + '_' + str(k)
215                  x[i, j, k] = model.addVar(lb = 0
216                                            , ub = 1
217                                            , vtype = GRB.BINARY
218                                            , name = name)
219                  x_var[i, j, k] = model.addVar(lb = 0
220                                            , ub = 1
221                                            , vtype = GRB.CONTINUOUS
222                                            , name = name)
223
224  # Add constraints
225  # create the objective expression
226  obj = LinExpr(0)
227  for i in range(data.nodeNum):
228      for j in range(data.nodeNum):
229          if(i != j and data.arcs[i,j] == 1):
230              for k in range(data.vehicleNum):
231                  obj.addTerms(data.disMatrix[i][j], x[i,j,k])
232  #print(model.getObjective()); # 这个可以打印出目标函数
233  # add the objective function into the model
234  model.setObjective(obj, GRB.MINIMIZE)
235
236  # constraint (1)
237  for i in range(1, data.nodeNum - 1): # 这里需要注意，i 的取值范围，否则可能会加入空约束
238      expr = LinExpr(0)
239      for j in range(data.nodeNum):
240          if(i != j and data.arcs[i,j] == 1):
241              for k in range(data.vehicleNum):
242                  if(i != 0 and i != data.nodeNum - 1):
243                      expr.addTerms(1, x[i,j,k])
244
245      model.addConstr(expr == 1, "c1")
246      expr.clear()
247
248  # constraint (2)
249  for k in range(data.vehicleNum):
250      expr = LinExpr(0);
251      for i in range(1, data.nodeNum - 1):
252          for j in range(data.nodeNum):
253              if(i != 0 and i != data.nodeNum - 1 and i != j and data.arcs[i,j] == 1):
```

```
                expr.addTerms(data.demand[i], x[i,j,k])
    model.addConstr(expr <= data.capacity, "c2")
    expr.clear()

# constraint (3)
for k in range(data.vehicleNum):
    expr = LinExpr(0)
    for j in range(1, data.nodeNum): # 处处注意，不能有 i == j 的情况出现
        if(data.arcs[0,j] == 1):
            expr.addTerms(1.0, x[0,j,k])
    model.addConstr(expr == 1.0, "c3")
    expr.clear()

# constraint (4)
for k in range(data.vehicleNum):
    for h in range(1, data.nodeNum - 1):
        expr1 = LinExpr(0)
        expr2 = LinExpr(0)
        for i in range(data.nodeNum):
            if(h != i and data.arcs[i,h] == 1):
                expr1.addTerms(1, x[i,h,k])

        for j in range(data.nodeNum):
            if(h != j and data.arcs[h,j] == 1):
                expr2.addTerms(1, x[h,j,k])

        model.addConstr(expr1 == expr2, "c4")
        expr1.clear()
        expr2.clear()

# constraint (5)
for k in range(data.vehicleNum):
    expr = LinExpr(0)
    for i in range(data.nodeNum - 1): # 这个地方也要注意，是 data.nodeNum - 1, 不是 data.nodeNum
        if(data.arcs[i,data.nodeNum - 1] == 1):
            expr.addTerms(1, x[i,data.nodeNum - 1,k])
    model.addConstr(expr == 1, "c5")
    expr.clear()

# constraint (6)
big_M = 0
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        big_M = max(data.dueTime[i] + data.disMatrix[i][j] - data.readyTime[i], big_M)

for k in range(data.vehicleNum):
    for i in range(data.nodeNum):
        for j in range(data.nodeNum):
            if(i != j and data.arcs[i,j] == 1):
                model.addConstr(s[i,k] + data.disMatrix[i][j] - s[j,k] <= big_M  - big_M * x[i,j,k], "c6")

# model.setParam('MIPGap', 0)
model.optimize()

print("\n\n-----optimal value-----")
print(model.ObjVal)

edge_list = []
for key in x.keys():
```

```
313         if(x[key].x > 0):
314             print(x[key].VarName, ' = ', x[key].x)
315             arc = (str(key[0]), str(key[1]))
316             edge_list.append(arc)
317 
318     nodes_col['0'] = 'red'
319     nodes_col[str(data.nodeNum-1)] = 'red'
320     plt.rcParams['figure.figsize'] = (15, 15) # 单位是 inches
321     nx.draw(Graph
322             , pos=pos_location
323     #         , with_labels = True
324             , node_size = 50
325             , node_color = nodes_col.values()   #'y'
326             , font_size = 15
327             , font_family = 'arial'
328     #         , edge_color = 'grey'   #'grey'  # b, k, m, g,
329             , edgelist = edge_list
330     #         , nodelist = nodeList
331           )
332     fig_name = 'network_' + str(customerNum) + '_1000.jpg'
333     plt.savefig(fig_name, dpi=600)
334     plt.show()
335 
336     # # Node class
337     class Node:
338         # this class defines the node
339         def __init__(self):
340             self.local_LB = 0
341             self.local_UB = np.inf
342             self.x_sol = {}
343             self.x_int_sol = {}
344             self.branch_var_list = []
345             self.model = None
346             self.cnt = None
347             self.is_integer = False
348             self.var_LB = {}
349             self.var_UB = {}
350 
351         def deepcopy_node(node):
352             new_node = Node()
353             new_node.local_LB = 0
354             new_node.local_UB = np.inf
355             new_node.x_sol = copy.deepcopy(node.x_sol)
356             new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
357             new_node.branch_var_list = []
358             new_node.model = node.model.copy()
359             new_node.cnt = node.cnt
360             new_node.is_integer = node.is_integer
361 
362             return new_node
363 
364     class Node_2:
365         # this class defines the node
366         def __init__(self):
367             self.local_LB = 0
368             self.local_UB = np.inf
369             self.x_sol = {}
370             self.x_int_sol = {}
371             self.branch_var_list = []
```

```
372             self.cnt = None
373             self.is_integer = False
374             self.var_LB = {}
375             self.var_UB = {}
376 
377         def deepcopy_node(node):
378             new_node = Node()
379             new_node.local_LB = 0
380             new_node.local_UB = np.inf
381             new_node.x_sol = copy.deepcopy(node.x_sol)
382             new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
383             new_node.branch_var_list = []
384             new_node.cnt = node.cnt
385             new_node.is_integer = node.is_integer
386             new_node.var_LB = copy.deepcopy(node.var_LB)
387             new_node.var_UB = copy.deepcopy(node.var_UB)
388 
389             return new_node
390 
391 
392     # # Branch and Bound framework -v1: copy model at each node
393     def Branch_and_bound(VRPTW_model, summary_interval):
394         Relax_VRPTW_model = VRPTW_model.relax()
395         # initialize the initial node
396         Relax_VRPTW_model.optimize()
397         global_UB = np.inf
398         global_LB = Relax_VRPTW_model.ObjVal
399         eps = 1e-6
400         incumbent_node = None
401         Gap = np.inf
402         feasible_sol_cnt = 0
403 
404         '''
405            Branch and Bound starts
406         '''
407         # creat initial node
408         Queue = []
409         node = Node()
410         node.local_LB = global_LB
411         node.local_UB = np.inf
412         node.model = Relax_VRPTW_model.copy()
413         node.model.setParam("OutputFlag", 0)
414         node.cnt = 0
415         Queue.append(node)
416 
417         cnt = 0
418         branch_cnt = 0
419         Global_UB_change = []
420         Global_LB_change = []
421         while (len(Queue) > 0 and global_UB - global_LB > eps):
422             # select the current node
423             current_node = Queue.pop()
424             cnt += 1
425 
426             # solve the current model
427             current_node.model.optimize()
428             Solution_status = current_node.model.Status
429 
430             '''
```

```
        OPTIMAL = 2
        INFEASIBLE = 3
        UNBOUNDED = 5
        '''
        # check whether the current solution is integer and execute prune step
        '''
            is_integer : mark whether the current solution is integer solution
            Is_Pruned  : mark whether the current solution is pruned
        '''
        is_integer = True
        Is_Pruned = False
        if (Solution_status == 2):
            for var in current_node.model.getVars():
                if(var.VarName.startswith('x')):
                    current_node.x_sol[var.varName] = copy.deepcopy(current_node.model.getVarByName(var.VarName).x)
                    print(var.VarName, ' = ', var.x)
                    # record the branchable variable
                    if(abs(round(var.x, 0) - var.x) >= eps):
                    #if(round(var.x, 0) != var.x):  # 如果改成在一定精度范围内，就会出现错误，结果不是最优解
                        is_integer = False
                        current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
                        ↪  variables
            # print(current_node.x_sol)

            # update the LB and UB
            if (is_integer == True):
                feasible_sol_cnt +=  1
                # For integer solution node, update the LB and UB
                current_node.is_integer = True
                current_node.local_LB = current_node.model.ObjVal
                current_node.local_UB = current_node.model.ObjVal
                # if the solution is integer, uodate the UB of global and update the incumbent
                if (current_node.local_UB < global_UB):
                    global_UB = current_node.local_UB
                    incumbent_node = Node.deepcopy_node(current_node)
            if (is_integer == False):
                # For integer solution node, update the LB and UB also
                current_node.is_integer = False
                current_node.local_UB = global_UB
                current_node.local_LB = current_node.model.ObjVal

            '''
                PRUNE step
            '''
            # prune by optimility
            if (is_integer == True):
                Is_Pruned = True

            # prune by bound
            if (is_integer == False and current_node.local_LB > global_UB):
                Is_Pruned = True

            Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
        elif (Solution_status != 2):
            # the current node is infeasible or unbound
            is_integer = False

            '''
                PRUNE step
```

```
            '''
            # prune by infeasiblity
            Is_Pruned = True

            continue

        '''
          BRANCH STEP
        '''
        if (Is_Pruned == False):
            branch_cnt += 1
            # selecte the branch variable: choose the value which is closest to 0.5
            branch_var_name = None
            #
            min_diff = 100
            for var_name in current_node.branch_var_list:
                if(abs(current_node.x_sol[var_name] - 0.5) < min_diff):
                    branch_var_name = var_name
                    min_diff = abs(current_node.x_sol[var_name] - 0.5)
            # branch_var_name = current_node.branch_var_list[0]

            # choose the variable cloest to 0 or 1
#             min_diff = 100
#             for var_name in current_node.branch_var_list:
#                 diff = max(abs(current_node.x_sol[var_name] - 1), abs(current_node.x_sol[var_name] - 0))
#                 if(min_diff >= diff):
#                     branch_var_name = var_name
#                     min_diff = diff

#             branch_var_name = current_node.branch_var_list[0]
            if(branch_cnt % summary_interval == 0):
                print('Branch var name :', branch_var_name, '\t, Branch var value :',
                ↪  current_node.x_sol[branch_var_name])
            left_var_bound = (int)(current_node.x_sol[branch_var_name])
            right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1

            # creat two child nodes
            left_node = Node.deepcopy_node(current_node)
            right_node = Node.deepcopy_node(current_node)

            # creat left child node
            temp_var = left_node.model.getVarByName(branch_var_name)
            left_node.model.addConstr(temp_var <= left_var_bound, name='branch_left_' + str(cnt))
            left_node.model.setParam("OutputFlag", 0)
            left_node.model.update()
            cnt += 1
            left_node.cnt = cnt

            # creat right child node
            temp_var = right_node.model.getVarByName(branch_var_name)
            right_node.model.addConstr(temp_var >= right_var_bound, name='branch_right_' + str(cnt))
            right_node.model.setParam("OutputFlag", 0)
            right_node.model.update()
            cnt += 1
            right_node.cnt = cnt

            Queue.append(left_node)
            Queue.append(right_node)

```

```
547                 # update the global LB, explore all the leaf nodes
548                 temp_global_LB = np.inf
549                 for node in Queue:
550                     node.model.optimize()
551                     if(node.model.status == 2):
552                         if(node.model.ObjVal <= temp_global_LB and node.model.ObjVal <= global_UB):
553                             temp_global_LB = node.model.ObjVal
554 
555 
556                 global_LB = temp_global_LB
557                 Global_UB_change.append(global_UB)
558                 Global_LB_change.append(global_LB)
559 
560             if( (cnt - 2) % summary_interval == 0):
561                 print('\n\n====================')
562                 print('Queue length :', len(Queue))
563                 print('\n ------------ \n', cnt, ' UB  = ', global_UB, '   LB = ', global_LB, '\t Gap  = ', Gap, '  %',
                    ↪  'feasible_sol_cnt :', feasible_sol_cnt)
564 
565         # all the nodes are explored, update the LB and UB
566         incumbent_node.model.optimize()
567         global_UB = incumbent_node.model.ObjVal
568         global_LB = global_UB
569         Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
570         Global_UB_change.append(global_UB)
571         Global_LB_change.append(global_LB)
572 
573         print('\n\n\n\n')
574         print('-----------------------------------------')
575         print('         Branch and Bound terminates     ')
576         print('         Optimal solution found          ')
577         print('-----------------------------------------')
578         print('\nIter cnt  = ', cnt, ' \n\n')
579         print('\nFinal Gap  = ', Gap, '  % \n\n')
580     #     print('Optimal Solution:', incumbent_node.x_sol)
581         print(' ---- Optimal Solution ------')
582         for key in incumbent_node.x_sol.keys():
583             if(incumbent_node.x_sol[key] > 0):
584                 print(key, ' = ', incumbent_node.x_sol[key])
585         print('\nOptimal Obj:', global_LB)
586 
587         return incumbent_node, Gap, Global_UB_change, Global_LB_change
588 
589 
590     # # Solve the IP model by branch and bound
591     incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_bound(model, summary_interval = 50)
592 
593     for key in incumbent_node.x_sol.keys():
594         if(incumbent_node.x_sol[key] > 0):
595             print(key, ' = ', incumbent_node.x_sol[key])
596 
597 
598     # # Branch and Bound V2:do not copy model at node
599     def Branch_and_bound_v2(VRPTW_model, summary_interval):
600         Relax_VRPTW_model = VRPTW_model.relax()
601         Relax_VRPTW_model.setParam("OutputFlag", 0)
602         # Relax_VRPTW_model.setParam("OptimalityTol", 1e-3)
603         # Relax_VRPTW_model.Params.Presolve = 0
604         # Relax_VRPTW_model.Params.FeasibilityTol = 1e-2
```

```
    # initialize the initial node
    Relax_VRPTW_model.optimize()
    global_UB = np.inf
    global_LB = Relax_VRPTW_model.ObjVal
    eps = 1e-6
    eps_bound = 1e-3
    incumbent_node = None
    Gap = np.inf
    feasible_sol_cnt = 0

    '''
       Branch and Bound starts
    '''

    # creat initial node
    Queue = []
    node = Node_2()
    node.local_LB = global_LB
    node.local_UB = np.inf
    node.cnt = 0
    for var in Relax_VRPTW_model.getVars():
        if(var.VarName.startswith('x')):
            node.var_LB[var.VarName] = 0
            node.var_UB[var.VarName] = 1
    if (Relax_VRPTW_model.status == 2):
        for var in Relax_VRPTW_model.getVars():
            if(var.VarName.startswith('x')):
                # print(var.VarName, ' = ', var.x)
                # record the branchable variable
                # if(abs(round(var.x, 0) - var.x) >= eps):
                if(round(var.x, 0) != var.x):
                    node.branch_var_list.append(var.VarName)  # to record the candidate branch variables
                    node.x_sol[var.varName] = var.x
                else:
                    node.x_sol[var.varName] = round(var.x, 0) # 精度问题，直接圆整成整数解

    Queue.append(node)

    cnt = 0
    Global_UB_change = []
    Global_LB_change = []
    while (len(Queue) > 0 and global_UB - global_LB > eps):
        # select the current node
        current_node = Queue.pop()
        cnt += 1

        for var_name in current_node.var_LB.keys():
            Relax_VRPTW_model.getVarByName(var_name).lb = current_node.var_LB[var_name]
            Relax_VRPTW_model.getVarByName(var_name).ub = current_node.var_UB[var_name]
        Relax_VRPTW_model.update()
        # solve the current model
        Relax_VRPTW_model.optimize()
        Solution_status = Relax_VRPTW_model.Status

        '''
        OPTIMAL = 2
        INFEASIBLE = 3
        UNBOUNDED = 5
        '''
```

```
664
665         # check whether the current solution is integer and execute prune step
666         '''
667             is_integer : mark whether the current solution is integer solution
668             Is_Pruned  : mark whether the current solution is pruned
669         '''
670         is_integer = True
671         Is_Pruned = False
672         if (Solution_status == 2):
673             for var in Relax_VRPTW_model.getVars():
674                 if(var.VarName.startswith('x')):
675                     # print(var.VarName, ' = ', var.x)
676                     # record the branchable variable
677                     if(abs(round(var.x, 0) - var.x) >= eps):
678                         is_integer = False
679                         current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
                            ↪  variables
680                         current_node.x_sol[var.VarName] = var.x
681                     else:
682                         current_node.x_sol[var.VarName] = round(var.x, 0) # 精度问题，直接圆整成整数解
683
684             # update the LB and UB
685             if (is_integer == True):
686                 feasible_sol_cnt +=  1
687                 # For integer solution node, update the LB and UB
688                 current_node.is_integer = True
689                 current_node.local_LB = Relax_VRPTW_model.ObjVal
690                 current_node.local_UB = Relax_VRPTW_model.ObjVal
691                 # if the solution is integer, uodate the UB of global and update the incumbent
692                 if (current_node.local_UB < global_UB):
693                     global_UB = current_node.local_UB
694                     incumbent_node = Node_2.deepcopy_node(current_node)
695             if (is_integer == False):
696                 # For integer solution node, update the LB and UB also
697                 current_node.is_integer = False
698                 current_node.local_UB = global_UB
699                 current_node.local_LB = Relax_VRPTW_model.ObjVal
700
701             '''
702                 PRUNE step
703             '''
704             # prune by optimility
705             if (is_integer == True):
706                 Is_Pruned = True
707
708             # prune by bound
709             if (is_integer == False and current_node.local_LB > global_UB):
710                 Is_Pruned = True
711
712             Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
713
714         elif (Solution_status != 2):
715             # the current node is infeasible or unbound
716             is_integer = False
717
718             '''
719                 PRUNE step
720             '''
721             # prune by infeasiblity
```

```
722                 Is_Pruned = True
723
724                 continue
725
726             '''
727               BRANCH STEP
728             '''
729             if (Is_Pruned == False):
730                 # selecte the branch variable: choose the value which is closest to 0.5
731                 branch_var_name = None
732     #             min_diff = 100
733     #             for var_name in current_node.branch_var_list:
734     #                 if(abs(current_node.x_sol[var_name] - 0.5) < min_diff):
735     #                     branch_var_name = var_name
736     #                       min_diff = abs(current_node.x_sol[var_name] - 0.5)
737     #                     print(var_name, ' = ', current_node.x_sol[branch_var_name])
738     #             branch_var_name = current_node.branch_var_list[0]
739
740                 # choose the variable cloest to 0 or 1
741                 min_diff = 100
742                 for var_name in current_node.branch_var_list:
743                     diff = max(abs(current_node.x_sol[var_name] - 1), abs(current_node.x_sol[var_name] - 0))
744                     if(min_diff >= diff):
745                         branch_var_name = var_name
746                         min_diff = diff
747
748                 # branch_var_name = current_node.branch_var_list[0]
749                 if(cnt % summary_interval == 0):
750                     print('Branch var name :', branch_var_name, '\t, value :', current_node.x_sol[branch_var_name])
751                 left_var_bound = (int)(current_node.x_sol[branch_var_name])
752                 right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1
753
754                 # creat two child nodes
755                 left_node = Node_2.deepcopy_node(current_node)
756                 right_node = Node_2.deepcopy_node(current_node)
757
758                 # creat left child node (update the bound data for left node)
759                 left_node.var_LB[branch_var_name] = 0
760                 left_node.var_UB[branch_var_name] = 0
761                 cnt += 1
762                 left_node.cnt = cnt
763
764                 # creat right child node (update the bound data for right node)
765                 right_node.var_LB[branch_var_name] = 1
766                 right_node.var_UB[branch_var_name] = 1
767                 cnt += 1
768                 right_node.cnt = cnt
769
770                 Queue.append(left_node)
771                 Queue.append(right_node)
772
773                 # update the global LB, explore all the leaf nodes
774                 temp_global_LB = np.inf
775                 for node in Queue:
776                     for var_name in node.var_LB.keys():
777                         Relax_VRPTW_model.getVarByName(var_name).lb = node.var_LB[var_name]
778                         Relax_VRPTW_model.getVarByName(var_name).ub = node.var_UB[var_name]
779
780                     Relax_VRPTW_model.update()
```

```
781                 Relax_VRPTW_model.optimize()
782                 if(Relax_VRPTW_model.status == 2):
783                     if(Relax_VRPTW_model.ObjVal <= temp_global_LB and Relax_VRPTW_model.ObjVal <= global_UB):
784                         temp_global_LB = Relax_VRPTW_model.ObjVal
785
786             global_LB = temp_global_LB
787             Global_UB_change.append(global_UB)
788             Global_LB_change.append(global_LB)
789
790         if(cnt % summary_interval == 0):
791             print('\n\n====================')
792             print('Queue length :', len(Queue))
793             print('\n ------------ \n', cnt, ' UB  = ', global_UB, '   LB = ', global_LB, '\t Gap  = ', Gap, '  %',
                ↪  'feasible_sol_cnt :', feasible_sol_cnt)
794
795     # all the nodes are explored, update the LB and UB
796     for var_name in incumbent_node.var_LB.keys():
797         Relax_VRPTW_model.getVarByName(var_name).lb = incumbent_node.var_LB[var_name]
798         Relax_VRPTW_model.getVarByName(var_name).ub = incumbent_node.var_UB[var_name]
799     Relax_VRPTW_model.update()
800     Relax_VRPTW_model.optimize()
801     global_UB = Relax_VRPTW_model.ObjVal
802     global_LB = global_UB
803     Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
804     Global_UB_change.append(global_UB)
805     Global_LB_change.append(global_LB)
806
807     Global_UB_change.append(global_UB)
808     Global_LB_change.append(global_UB)
809
810     print('\n\n\n\n')
811     print('-----------------------------------------')
812     print('         Branch and Bound terminates     ')
813     print('         Optimal solution found          ')
814     print('-----------------------------------------')
815     print('\nFinal Gap  = ', Gap, '  %')
816     # print('Optimal Solution:', incumbent_node.x_sol)
817     print('Optimal Solution: \n')
818     for key in incumbent_node.x_sol.keys():
819         if(incumbent_node.x_sol[key] > 0):
820             print(key, ' = ', incumbent_node.x_sol[key])
821     print('Optimal Obj:', global_LB)
822
823     return incumbent_node, Gap, Global_UB_change, Global_LB_change
824
825 # # Solve the IP model by branch and bound version 2
826 incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_bound_v2(model, summary_interval = 500)
827
828 # # plot the results
829 # fig = plt.figure(1)
830 # plt.figure(figsize=(15,10))
831 font_dict = {"family":'Arial',    #"Kaiti",
832        "style":"oblique",
833         "weight":"normal",
834         "color":"green",
835         "size": 20
836        }
837
838 plt.rcParams['figure.figsize'] = (12.0, 8.0) # 单位是 inches
```

```
839  plt.rcParams["font.family"] = 'Arial' #"SimHei"
840  plt.rcParams["font.size"] = 16
841
842  x_cor = range(1, len(Global_LB_change) + 1)
843  plt.plot(x_cor, Global_LB_change, label = 'LB')
844  plt.plot(x_cor, Global_UB_change, label = 'UB')
845  plt.legend()
846  plt.xlabel('Iteration', fontdict=font_dict)
847  plt.ylabel('Bounds update', fontdict=font_dict)
848  plt.title('VRPTW (c101-60) : Bounds update during branch and bound procedure \n', fontsize = 23)
849  plt.savefig('Bound_updates_VRPTW_c101_60_2.eps')
850  plt.savefig('Bound_updates_VRPTW_c101_60_2.pdf')
851  plt.show()
```

10.11 Java 调用 CPLEX 实现分支定界算法求解 VRPTW：介绍

VRPTW 的分支定界算法中分支操作的实现方式主要有以下 3 种：

1. 用设置变量上下界的方法实现分支；
2. 用添加约束的方法实现分支；
3. 用 callback 实现分支。

其中，设置变量上下界和添加约束的方法一般需要自行实现分支定界，而使用 callback 则不需要实现分支定界，只需要调用求解器的分支定界即可。使用 callback 实现分支定界的代码量相对较少，但是灵活性差，且对求解器的种类有一定要求。CPLEX 可以通过 callback 指定分支变量，但是 Gurobi 等求解器却暂时不支持该功能。

本节将提供 3 个版本的分支定界算法代码，分别对应上述 3 种实现分支的方式。其中，第一种实现分支的方法对应的代码原作者为黄楠博士，毕业于华中科技大学，相应的推文发布在微信公众号” 数据魔术师” 上，题目为《分支定界法解带时间窗的车辆路径规划问题》[1]。该代码相对简单易懂，因此本小节采用该代码作为示例代码，供读者学习。在忠于原始代码的基础上，本书作者进行了一些修改，并增加了部分注释。

以添加约束的形式实现分支定界的代码是基于第一版代码修改而来，代码中分支操作部分发生了较大改动，其余部分改动较小。

用 callback 实现分支的代码较为简单，本书不再做过多阐述。

10.12 Java 调用 CPLEX 实现分支定界算法求解 VRPTW(版本 1)：通过设置变量界限实现分支

本代码中有Node类、Data类、Check类和BaB_VRPTW_v1类。其中：

[1] https://mp.weixin.qq.com/s/_AwEYJylTeAcwC2W5jIjQw

- Node类: 为 BB tree 中的分支节点类，分支节点类中需要存储该节点的深度、节点的目标函数、LP 解、解对应的路径以及与分支相关的信息等。其中
  - node_cost：为节点的 LP 的目标函数。
  - node_x_map：是一个三维数组，索引与 x_{ijk} 一一对应，用来辅助分支；如果 node_x_map[i][j][k] 为 0, 表示在该分支中 $x_{ijk} \in \{0,1\}$；如果为 1，表示在该节点的模型中必须约束 $x_{ijk} = 1$；如果为 -1，则表示在该节点的模型中必须约束 $x_{ijk} = 0$。由于 VRPTW 中，决策变量 $x_{ijk} \in \{0,1\}$，因此我们可以通过设置节点模型的 Bound 来完成分支，而不用显式地加入一条分支约束 (当然我们也提供了加入分支约束版本的代码)。在每个分支节点，我们只需要利用 node_x_map 的值，通过与函数 set_bound1(Node node) 的配合，即可完成分支。
  - node_x：二维数组，索引与弧 $A = \{(i,j)\}$ 对应。表示弧 (i,j) 在最优解中是否被访问。该数组也是用来辅助分支，只不过该分支更高效，是一种基于弧的分支。如果 node_x[i][j] 为 0, 表示在该分支中弧 (i,j) 可被访问，也可不被访问，因此可以不做任何操作；如果为 1，表示在该节点的模型中必须约束 $\sum_k x_{ijk} = 1$；如果为 -1，则表示在该节点的模型中必须约束 $\sum_k x_{ijk} = 0$，即将所有的 $x_{ijk}, \forall k$ 的 UB 均设置为 0 即可。可以通过调用函数 set_bound(Node node) 实现。
- Data类: 用于存储算例数据。其中函数 Read_data() 用于从 txt 文档中读取算例数据，并存储到 Data 类对象中，函数 double_truncate() 用于截断小数，只保留小数点后一位。
- Check类: 用于检查当前解是否是可行解。
- BaB_VRPTW_v1类: 实现 Branch and Bound 算法的类。其中
  - 函数build_model()用于构建根节点的 LP 模型。
  - get_value()用于将当前模型的解提取出来，包括决策变量 x_{ijk} 的取值以及对应的路径 routes 和在每个节点的开始服务时间 servetimes。
  - 函数init(BaB_VRPTW_v1 lp)函数用于初始化 BranchAndBound_v1 类的对象 lp。包括构建模型，求解模型并提取对应的解等。当然，还可以删除不必要的车辆。
  - 函数branch_and_bound()即为 Branch and Bound 算法的主要部分，其中在分左支和分右支的时候分别调用了函数 branch_left_arc() 和 branch_right_arc()，用于添加左支和右支的分支约束。在 branch_left_arc() 和 branch_right_arc() 中，分支约束又是通过调用函数 set_bound() 实现的。

10.12.1 Node 类

Node.java

```
1  package VRPTW_BrandhAndBound;
2
3  import java.util.ArrayList;
4
```

```java
public class Node implements Comparable{
    /*
     * 这里继承了 Comparable 接口，是因为在实现分支定界的时候，该代码使用了优先队列
     * 存储未探索的节点，即 PriorityQueue.
     */

    Data            data;               // 算例数据
    int               d;                    // 节点的深度
    double           node_cost;           // 该节点的 LP 的目标值
    double[][][]    lp_x;               // 该节点的 LP 的小数解 (x[i][j][k])
    int[][][]        node_x_map;          // node_x[i][j]=1 时，node_x_map[i][j][k]=1 表示必须访问，
    ↪ node_x_map[i][j][k]=0 表示不能访问
    int[][]           node_x;              // node_x[i][j]=0 表示弧 (i, j) 可以访问，1 表示弧 (i, j) 必须访问，-1
    ↪ 表示弧 (i, j) 不能访问
    ArrayList<ArrayList<Integer>> node_routes;        // 车辆路径
    ArrayList<ArrayList<Double>> node_servetimes;    // 顾客的开始服务时间

    public Node(Data data) {
        super();
        this.data = data;
        node_cost = data.big_num;
        lp_x = new double [data.vertex_num][data.vertex_num][data.veh_num];
        node_x_map = new int[data.vertex_num][data.vertex_num][data.veh_num];
        node_x = new int[data.vertex_num][data.vertex_num];
        node_routes = new ArrayList<ArrayList<Integer>>();
        node_servetimes = new ArrayList<ArrayList<Double>>();
    }


    /**
     * node 的深度拷贝函数
     *
     * 这里使用了 clone 方法
     * Computer c=new Computer("dell", "4G 内存");
     * Computer c1=c.Clone();
     * 在这两句代码中有两个 Computer 类型的对象 c 和 c1，其中 c1 就是通过 Clone 方法复制的 c，
     * 我们可以使用 System.out.println() 方法将两个对象的内存地址打印出来，会发现是两个不同的值。
     *
     * @return new_node
     */
    public Node node_copy() {
        Node new_node = new Node(data);
        new_node.d = d;
        new_node.node_cost = node_cost;
        for (int i = 0; i < lp_x.length; i++) {
            for (int j = 0; j < lp_x[i].length; j++) {
                new_node.lp_x[i][j] = lp_x[i][j].clone();
            }
        }
        for (int i = 0; i < node_x.length; i++) {
            new_node.node_x[i] = node_x[i].clone();
        }
        for (int i = 0; i < node_x_map.length; i++) {
            for (int j = 0; j < node_x_map[i].length; j++) {
                new_node.node_x_map[i][j] = node_x_map[i][j].clone();
            }
        }
        for (int i = 0; i < node_routes.size(); i++) {
            new_node.node_routes.add((ArrayList<Integer>) node_routes.get(i).clone());
```

```
62          }
63          for (int i = 0; i < node_servetimes.size(); i++) {
64              new_node.node_servetimes.add((ArrayList<Double>) node_servetimes.get(i).clone());
65          }
66          return new_node;
67      }
68
69
70      /**
71       * compareTo:
72       *     比较两个节点实例的大小的函数。如果对象 o 更小，则返回 1
73       *  用于优先队列，是为了找到下一个要处理的节点。
74       */
75      public int compareTo(Object o){
76          Node node = (Node) o;
77          if(node_cost < node.node_cost)
78              return -1;
79          else if(node_cost == node.node_cost)
80              return 0;
81          else
82              return 1;
83      }
84  }
```

10.12.2 Data 类

Data.java

```
1   package VRPTW_Branch_and_Bound;
2
3   import java.io.BufferedReader;
4   import java.io.FileReader;
5   import java.util.Scanner;
6
7   //定义参数
8   class Data{
9       int vertex_num;            //所有点集合 n（包括配送中心和客户点，首尾（0 和 n）为配送中心）
10      double E;                  //配送中心时间窗开始时间
11      double    L;                 //配送中心时间窗结束时间
12      int veh_num;             //车辆数
13      double cap;              //车辆载荷
14      int[][] vertexs;         //所有点的坐标 x,y
15      int[] demands;             //需求量
16      int[] vehicles;            //车辆编号
17      double[] a;                //时间窗开始时间【a[i],b[i]】
18      double[] b;                //时间窗结束时间【a[i],b[i]】
19      double[] s;                //客户点的服务时间
20      int[][] arcs;             //arcs[i][j] 表示 i 到 j 点的弧
21      double[][] dist;         //距离矩阵，满足三角关系，暂用距离表示花费 C[i][j]=dist[i][j]
22      double gap= 1e-6;
23      double big_num = 100000;
24
25      //截断小数 3.26434-->3.2
26      public double double_truncate(double v){
27          int iv = (int) v;
28          if(iv+1 - v <= gap)
29              return iv+1;
30          double dv = (v - iv) * 10;
31          int idv = (int) dv;
```

```
32          double rv = iv + idv / 10.0;
33          return rv;
34      }
35      public Data() {
36          super();
37      }
38
39      //函数功能：从 txt 文件中读取数据并初始化参数
40      public void Read_data(String path,Data data,int vertexnum) throws Exception{
41          String line = null;
42          String[] substr = null;
43          Scanner cin = new Scanner(new BufferedReader(new FileReader(path)));  //读取文件
44          for(int i =0; i < 4;i++){
45              line = cin.nextLine();  //读取一行
46          }
47          line = cin.nextLine();
48          line.trim(); //返回调用字符串对象的一个副本，删除起始和结尾的空格
49          substr = line.split(("\\s+")); //以空格为标志将字符串拆分
50          //初始化参数
51          data.vertex_num = vertexnum;
52          data.veh_num = Integer.parseInt(substr[1]);
53          data.cap = Integer.parseInt(substr[2]);
54          data.vertexs =new int[data.vertex_num][2];                //所有点的坐标 x,y
55          data.demands = new int[data.vertex_num];                    //需求量
56          data.vehicles = new int[data.veh_num];                    //车辆编号
57          data.a = new double[data.vertex_num];                        //时间窗开始时间
58          data.b = new double[data.vertex_num];                        //时间窗结束时间
59          data.s = new double[data.vertex_num];                        //服务时间
60          data.arcs = new int[data.vertex_num][data.vertex_num];
61          //距离矩阵，满足三角关系，用距离表示 cost
62          data.dist = new double[data.vertex_num][data.vertex_num];
63          for(int i =0; i < 4;i++){
64              line = cin.nextLine();
65          }
66          //读取 vetexnum-1 行数据
67          for (int i = 0; i < data.vertex_num - 1; i++) {
68              line = cin.nextLine();
69              line.trim();
70              substr = line.split("\\s+");
71              data.vertexs[i][0] = Integer.parseInt(substr[2]);
72              data.vertexs[i][1] = Integer.parseInt(substr[3]);
73              data.demands[i] = Integer.parseInt(substr[4]);
74              data.a[i] = Integer.parseInt(substr[5]);
75              data.b[i] = Integer.parseInt(substr[6]);
76              data.s[i] = Integer.parseInt(substr[7]);
77          }
78          cin.close();//关闭流
79          //初始化配送中心参数
80          data.vertexs[data.vertex_num-1] = data.vertexs[0];
81          data.demands[data.vertex_num-1] = 0;
82          data.a[data.vertex_num-1] = data.a[0];
83          data.b[data.vertex_num-1] = data.b[0];
84          data.E = data.a[0];
85          data.L = data.b[0];
86          data.s[data.vertex_num-1] = 0;
87          double min1 = 1e15;
88          double min2 = 1e15;
89          //距离矩阵初始化
90          for (int i = 0; i < data.vertex_num; i++) {
```

```
            for (int j = 0; j < data.vertex_num; j++) {
                if (i == j) {
                    data.dist[i][j] = 0;
                    continue;
                }
                data.dist[i][j] =
                    Math.sqrt((data.vertexs[i][0]-data.vertexs[j][0])
                            *(data.vertexs[i][0]-data.vertexs[j][0])+
                    (data.vertexs[i][1]-data.vertexs[j][1])
                    *(data.vertexs[i][1]-data.vertexs[j][1]));
                data.dist[i][j]=data.double_truncate(data.dist[i][j]);
            }
        }
        data.dist[0][data.vertex_num-1] = 0;
        data.dist[data.vertex_num-1][0] = 0;
        //距离矩阵满足三角关系
        for (int  k = 0; k < data.vertex_num; k++) {
            for (int i = 0; i < data.vertex_num; i++) {
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.dist[i][j] > data.dist[i][k] + data.dist[k][j]) {
                        data.dist[i][j] = data.dist[i][k] + data.dist[k][j];
                    }
                }
            }
        }
        //初始化为完全图
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (i != j) {
                    data.arcs[i][j] = 1;
                }
                else {
                    data.arcs[i][j] = 0;
                }
            }
        }
        /**
         *
         * 预处理，除去不符合时间窗和容量约束的边
         */
        //除去不符合时间窗和容量约束的边
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (i == j) {
                    continue;
                }
                if (data.a[i]+data.s[i]+data.dist[i][j]>data.b[j] ||
                        data.demands[i]+data.demands[j]>data.cap) {
                    data.arcs[i][j] = 0;
                }
                if (data.a[0]+data.s[i]+data.dist[0][i]+data.dist[i][data.vertex_num-1]>
                data.b[data.vertex_num-1]) {
                    System.out.println("the calculating example is false");

                }
            }
        }
        for (int i = 1; i < data.vertex_num-1; i++) {
            if (data.b[i] - data.dist[0][i] < min1) {
```

```
150                 min1 = data.b[i] - data.dist[0][i];
151             }
152             if (data.a[i] + data.s[i] + data.dist[i][data.vertex_num-1] < min2) {
153                 min2 = data.a[i] + data.s[i] + data.dist[i][data.vertex_num-1];
154             }
155         }
156         if (data.E > min1 || data.L < min2) {
157             System.out.println("Duration false!");
158             System.exit(0);//终止程序
159         }
160 
161         //初始化配送中心 0, n+1 两点的参数
162         data.arcs[data.vertex_num-1][0] = 0;
163         data.arcs[0][data.vertex_num-1] = 1;
164         for (int i = 1; i < data.vertex_num-1; i++) {
165             data.arcs[data.vertex_num-1][i] = 0;
166         }
167         for (int i = 1; i < data.vertex_num-1; i++) {
168             data.arcs[i][0] = 0;
169         }
170     }
171 
172     public static void printData(Data data){
173         System.out.println("vehicleNum" + "\t\t: " + data.veh_num);
174         System.out.println("vehicleCapacity" + "\t\t: " + data.cap);
175         for(int i = 0; i < data.vertex_num - 1; i++){  //这里用了增强 for
176             System.out.print(i + 1 + "\t");
177             System.out.print(data.vertexs[i][0] + "\t");
178             System.out.print(data.vertexs[i][1] + "\t");
179             System.out.print(data.demands[i] + "\t");
180             System.out.print(data.a[i] + "\t");
181             System.out.print(data.b[i] + "\t");
182             System.out.print(data.s[i] + "\n");
183         }
184         for(int i = 0; i < data.vertex_num - 1; i++) {
185             for(int j = 0; j < data.vertex_num - 1; j++) {
186                 System.out.print(data.dist[i][j] + "\t");
187             }
188             System.out.println();
189         }
190 
191     }
192 }
```

10.12.3 Check 类

Check.java

```
1  package VRPTW_BrandhAndBound;
2 
3  import java.util.ArrayList;
4  import ilog.concert.IloException;
5 
6  /**
7   * Check 类功能：解的可行性判断（可直接跳过此类）
8   */
9  class Check{
10     double epsilon = 0.0001;
11     Data data = new Data();
```

```
12      ArrayList<ArrayList<Integer>> routes = new ArrayList<>();
13      ArrayList<ArrayList<Double>> servetimes = new ArrayList<>();
14
15      /**
16       * 构造函数
17       * @param lp: BaB_Vrptw 的实例
18       */
19      public Check(BaB_VRPTW_v1 lp) {
20          super();
21          this.data = lp.data;
22          this.routes = lp.routes;
23          this.servetimes = lp.servetimes;
24      }
25
26      /**
27       * double_compare
28       *     函数功能：比较两个数的大小
29       */
30      public int double_compare(double v1,double v2) {
31          if (v1 < v2 - epsilon) {
32              return -1;
33          }
34          if (v1 > v2 + epsilon) {
35              return 1;
36          }
37          return 0;
38      }
39
40      /**
41       * 函数功能：解的可行性判断
42       * @throws IloException
43       */
44      public void fesible() throws IloException {
45          //车辆数量可行性判断
46          if (routes.size() > data.veh_num) {
47              System.out.println("error: vecnum!!!");
48              System.exit(0);
49          }
50          //车辆载荷可行性判断
51          for (int k = 0; k < routes.size(); k++) {
52              ArrayList<Integer> route = routes.get(k);
53              double capacity = 0;
54              for (int i = 0; i < route.size(); i++) {
55                  capacity += data.demands[route.get(i)];
56              }
57              if (capacity > data.cap) {
58                  System.out.println("error: cap!!!");
59                  System.exit(0);
60              }
61          }
62          //时间窗、车容量可行性判断
63          for (int k = 0; k < routes.size(); k++) {
64              ArrayList<Integer> route = routes.get(k);
65              ArrayList<Double> servertime = servetimes.get(k);
66              double capacity = 0;
67              for (int i = 0; i < route.size()-1; i++) {
68                  int origin = route.get(i);
69                  int destination = route.get(i+1);
70                  double si = servertime.get(i);
```

```
                double sj = servertime.get(i+1);
                if (si < data.a[origin] && si >  data.b[origin]) {
                    System.out.println("error: servertime!");
                    System.exit(0);
                }
        if (double_compare(si + data.dist[origin][destination],data.b[destination]) > 0) {
                    System.out.println(origin + ": [" + data.a[origin]
                            + ","+data.b[origin]+"]"+ " "+ si);
                    System.out.println(destination + ": [" +
                            data.a[destination] + ","+data.b[destination]+"]"+ " "+ sj);
                    System.out.println(data.dist[origin][destination]);
                    System.out.println(destination + ":" );
                    System.out.println("error: forward servertime!");
                    System.exit(0);
                }
            if (double_compare(sj - data.dist[origin][destination],data.a[origin]) < 0) {
                    System.out.println(origin + ": [" + data.a[origin]
                            + ","+data.b[origin]+"]"+ " "+ si);
                    System.out.println(destination + ": [" + data.a[destination]
                            + ","+data.b[destination]+"]"+ " "+ sj);
                    System.out.println(data.dist[origin][destination]);
                    System.out.println(destination + ":" );
                    System.out.println("error: backward servertime!");
                    System.exit(0);
                }
            }
            if (capacity > data.cap) {
                System.out.println("error: capacity!!!");
                System.exit(0);
            }
        }
    }
}
```

10.12.4 BaB_VRPTW_v1 类：以setUB和setLB的方式实现

BaB_VRPTW_v1.java

```
package VRPTW_BrandhAndBound;

import java.util.ArrayList;
import java.util.PriorityQueue;
import ilog.concert.IloException;
import ilog.concert.IloNumExpr;
import ilog.concert.IloNumVar;
import ilog.concert.IloNumVarType;
import ilog.cplex.IloCplex;

/**
 * @source: 本代码的原始版本来自于公众号" 数据魔术师", 原作者: 黄楠, 华中科技大学
 * @comment: 我在原始版本的基础上做了一些注释和一部分修改
 * @author: 黄楠, 华中科技大学
 * @revise: 刘兴禄, 清华大学
 * @date: 2018-8-2
 * @ 操作说明: 读入不同的文件前要手动修改 vetexnum 参数, 参数值为所有点个数, 包括配送中心 0 和 n+1, 代码算例截取于
↪  Solomon 测试算例
 *
 *
 * BaB_VRPTW_v1 的类功能: 建立 VRPTW 模型, 并使用 Branch and bound 算法求解
```

```
 */

public class BaB_VRPTW_v1 {

    static double gap = 1e-6;
    Data             data;          // 定义类 Data 的对象
    Node             node1;         // 分支左节点
    Node             node2;       // 分支右节点
    int              deep;          // 深度
    Node             best_node;     // 当前最好节点
    double             cur_best;      // 当前最好解
    int[]             record_arc; // 记录需要分支的变量下标 x[i][j][k] 的 [i][j][k]
    double             x_gap;         // 计算精度容差
    IloCplex         model;         // 模型实例
    double             cost;          // 目标值
    double[][][]     x_map;         // 记录解 x[i][j][k]
    public IloNumVar[][][]     x;  // x[i][j][k] 表示弧 arcs[i][j] 被车辆 k 访问
    public IloNumVar[][]     w;  // 车辆访问所有点的时间矩阵 ** 其实就是 s_ik，就是第 i 个点被第 k 辆车访问的时间
    public PriorityQueue<Node>         queue; // 分支队列
    ArrayList<ArrayList<Integer>>     routes; // 车辆路径
    ArrayList<ArrayList<Double>>     servetimes; // 顾客的开始服务时间

    /**
     * Branch_and_Bound_VRPTW 类的构造函数
     * @param data: 算例数据
     */
    public BaB_VRPTW_v1(Data data) {
        this.data = data;
        x_gap = data.gap;
        routes = new ArrayList<>(); // 定义车辆路径链表
        servetimes = new ArrayList<>(); // 定义花费时间链表
        // 初始化车辆路径和花费时间 list，list 长度为车辆数 k
        for (int k = 0; k < data.veh_num; k++) {
            ArrayList<Integer> r = new ArrayList<>();
            ArrayList<Double> t = new ArrayList<>();
            routes.add(r);
            servetimes.add(t);
        }
        x_map = new double[data.vertex_num][data.vertex_num][data.veh_num];
    }

    /**
     * 将 lp 中的数据清除
     */
    public void clear_lp() {
        data = null;
        routes.clear();
        servetimes.clear();
        x_map = null;
    }

    /**
     * 将 lp 解拷贝到 node
     *
     * @param lp
     * @param node
     */
    public void copy_lp_to_node(BaB_VRPTW_v1 lp, Node node) {
        // 首先清除 node 里面的数据
```

```
80          node.node_routes.clear();
81          node.node_servetimes.clear();
82
83          // 然后把 lp 里面的数据 copy 到 node 里面
84          node.node_cost = lp.cost;
85          for (int i = 0; i < lp.x_map.length; i++) {
86              for (int j = 0; j < lp.x_map[i].length; j++) {
87                  node.lp_x[i][j] = lp.x_map[i][j].clone();
88              }
89          }
90          for (int i = 0; i < lp.routes.size(); i++) {
91              node.node_routes.add((ArrayList<Integer>) lp.routes.get(i).clone());
92          }
93          for (int i = 0; i < lp.servetimes.size(); i++) {
94              node.node_servetimes.add((ArrayList<Double>) lp.servetimes.get(i)
95                      .clone());
96          }
97      }
98
99      /**
100      * 函数功能：建立 VRPTW 的 cplex 模型
101      * @throws IloException
102      *
103      * 建立模型：这里我们将 VRPTW 标准模型中的整数约束松弛掉，建立 VRPTW 的线性松弛
104      */
105     private void build_model() throws IloException {
106         // create model instance
107         model = new IloCplex();
108         // model.setOut(null);    // 关闭 CPLEX 的 log 信息
109         // model.setParam(IloCplex.DoubleParam.EpOpt, 1e-9);  // 设置 tolerance
110         // model.setParam(IloCplex.DoubleParam.EpGap, 1e-9);  // 设置 tolerance
111
112         // create decision variables
113         x = new IloNumVar[data.vertex_num][data.vertex_num][data.veh_num];
114         w = new IloNumVar[data.vertex_num][data.veh_num]; // 车辆访问点的时间
115
116         // 创建决策变量 x 和 w，设置其类型及取值范围
117         for (int i = 0; i < data.vertex_num; i++) {
118             for (int k = 0; k < data.veh_num; k++) {
119                 // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
120                 // w[i][k] = model.numVar(data.a[i], data.b[i], IloNumVarType.Float, "w" + i + "," + k);
121                 System.out.println(data.a[i] + " ---- " + data.b[i]);
122                 w[i][k] = model.numVar(0, 1e15, IloNumVarType.Float, "w" + i
123                         + "," + k);
124             }
125
126             for (int j = 0; j < data.vertex_num; j++) {
127                 if (data.arcs[i][j] == 0) {
128                     // 如果 i=j，则该条弧不通
129                     x[i][j] = null;
130                 } else {
131                     // x_ijk
132                     for (int k = 0; k < data.veh_num; k++) {
133                         // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
134                         x[i][j][k] = model.numVar(0, 1, IloNumVarType.Float,
135                                 "x" + i + "," + j + "," + k);
136                     }
137                 }
138             }
```

```
        }

        // 加入目标函数
        // 表达式 (7.1.1)
        IloNumExpr obj = model.numExpr();
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] == 0) {
                    System.out.println("delected arc : " + i + ", " + j);
                    continue;
                }
                for (int k = 0; k < data.veh_num; k++) {
                    obj = model.sum(obj,
                            model.prod(data.dist[i][j], x[i][j][k]));
                }
            }
        }
        model.addMinimize(obj);
        // 加入约束 1
        // 表达式 (7.1.2)
        for (int i = 1; i < data.vertex_num - 1; i++) {
            IloNumExpr expr1 = model.numExpr();
            for (int k = 0; k < data.veh_num; k++) {
                for (int j = 1; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr1 = model.sum(expr1, x[i][j][k]);
                    }
                }
            }
            model.addEq(expr1, 1);
        }
        // 加入约束 2
        // 表达式 (7.1.3)
        for (int k = 0; k < data.veh_num; k++) {
            IloNumExpr expr2 = model.numExpr();
            for (int j = 1; j < data.vertex_num; j++) {
                if (data.arcs[0][j] == 1) {
                    expr2 = model.sum(expr2, x[0][j][k]);
                }
            }
            model.addEq(expr2, 1);
        }
        // 加入约束 3
        // 表达式 (7.1.4)
        for (int k = 0; k < data.veh_num; k++) {
            for (int j = 1; j < data.vertex_num - 1; j++) {
                IloNumExpr expr3 = model.numExpr();
                IloNumExpr subExpr1 = model.numExpr();
                IloNumExpr subExpr2 = model.numExpr();
                for (int i = 0; i < data.vertex_num; i++) {
                    if (data.arcs[i][j] == 1) {
                        subExpr1 = model.sum(subExpr1, x[i][j][k]);
                    }
                    if (data.arcs[j][i] == 1) {
                        subExpr2 = model.sum(subExpr2, x[j][i][k]);
                    }
                }
                expr3 = model.sum(subExpr1, model.prod(-1, subExpr2));
                model.addEq(expr3, 0);
```

```
            }
        }
        // 加入约束 4
        // 表达式 (7.1.5)
        for (int k = 0; k < data.veh_num; k++) {
            IloNumExpr expr4 = model.numExpr();
            for (int i = 0; i < data.vertex_num - 1; i++) {
                if (data.arcs[i][data.vertex_num - 1] == 1) {
                    expr4 = model.sum(expr4, x[i][data.vertex_num - 1][k]);
                }
            }
            model.addEq(expr4, 1);
        }
        // 加入约束 5
        // 表达式 (7.1.6)
        for (int k = 0; k < data.veh_num; k++) {
            IloNumExpr expr8 = model.numExpr();
            for (int i = 1; i < data.vertex_num - 1; i++) {
                IloNumExpr expr9 = model.numExpr();
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr9 = model.sum(expr9, x[i][j][k]);
                    }
                }
                expr8 = model.sum(expr8, model.prod(data.demands[i], expr9));
            }
            model.addLe(expr8, data.cap);
            model.exportModel("VRPTW_LP.lp");
        }
        // 加入约束 6
        // 表达式 (7.1.7)
        double M = 1e5;
        for (int k = 0; k < data.veh_num; k++) {
            for (int i = 0; i < data.vertex_num; i++) {
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        IloNumExpr expr5 = model.numExpr();
                        IloNumExpr expr6 = model.numExpr();
                        expr5 = model.sum(w[i][k], data.s[i] + data.dist[i][j]);
                        expr5 = model.sum(expr5, model.prod(-1, w[j][k]));
                        expr6 = model.prod(M,
                                model.sum(1, model.prod(-1, x[i][j][k])));
                        model.addLe(expr5, expr6);
                    }
                }
            }
        }
        // 加入约束 7
        // 时间窗约束
        for (int k = 0; k < data.veh_num; k++) {
            for (int i = 1; i < data.vertex_num - 1; i++) {
                IloNumExpr expr7 = model.numExpr();
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr7 = model.sum(expr7, x[i][j][k]);
                    }
                }
                model.addLe(model.prod(data.a[i], expr7), w[i][k]);
                model.addLe(w[i][k], model.prod(data.b[i], expr7));
```

```
257                 }
258             }
259             // 加入约束 7
260             // 表达式 (7.1.8)
261             for (int k = 0; k < data.veh_num; k++) {
262                 model.addLe(data.E, w[0][k]);
263                 model.addLe(data.E, w[data.vertex_num - 1][k]);
264                 model.addLe(w[0][k], data.L);
265                 model.addLe(w[data.vertex_num - 1][k], data.L);
266             }
267 
268         }
269 
270 
271         /**
272          * 函数功能：解模型，并生成车辆路径和得到目标值
273          * @throws IloException
274          *
275          * 获取 VRPTW 模型的线性松弛的解，在根节点或者在分支节点处被调用
276          */
277         public void get_value() throws IloException {
278             routes.clear();
279             servetimes.clear();
280             cost = 0;
281 
282             // 初始化车辆路径和花费时间 list (空 list)，list 长度为车辆数 k
283             for (int k = 0; k < data.veh_num; k++) {
284                 ArrayList<Integer> r = new ArrayList<>();
285                 ArrayList<Double> t = new ArrayList<>();
286                 routes.add(r);
287                 servetimes.add(t);
288             }
289 
290             // x_map[i][j][k]，其实就相当于 x[i][j][k]，是为了记录解
291             for (int i = 0; i < data.vertex_num; i++) {
292                 for (int j = 0; j < data.vertex_num; j++) {
293                     for (int k = 0; k < data.veh_num; k++) {
294                         x_map[i][j][k] = 0.0; //首先将 x[i][j][k] 初始化为 0
295                     }
296                     if (data.arcs[i][j] > 0.5) { // data.arcs[i][j] 取值为 0 或者 1，这是判断 i 和 j 是都连接，是否是一条
                    ↪ 弧
297                         for (int k = 0; k < data.veh_num; k++) {
298                             x_map[i][j][k] = model.getValue(x[i][j][k]); // 将 x[i][j][k] 拷贝到 x_map[i][j][k] 中去
299                         }
300                     }
301                 }
302             }
303 
304             // 模型可解，从解 x[i][j][k] 中循环提取出车辆路径
305             for (int k = 0; k < data.veh_num; k++) {
306                 boolean terminate = true;
307                 int i = 0;
308                 routes.get(k).add(0); // 加入 0，拼成初始的 0-...
309                 servetimes.get(k).add(0.0);
310                 while (terminate) {
311                     for (int j = 0; j < data.vertex_num; j++) {
312                         if (doubleCompare(x_map[i][j][k], 0) == 1) {  // 如果 x_map[i][j][k] > 0
313                             routes.get(k).add(j);
314                             servetimes.get(k).add(model.getValue(w[j][k]));
```

```
                        i = j;
                        break;
                    }
                }
                if (i == data.vertex_num - 1) {
                    terminate = false;
                }
            }
            routes.get(k).set(routes.get(k).size() - 1, 0);
        }
        cost = model.getObjValue();
    }

    /**
     * init() 函数是确定有合法解的最小车辆数，由于直接求解解空间太大，且有很多车辆不能使用
     * 因此，我们删除无用的车辆，来缩小解空间（这是一个小优化，能够加快程序速度）
     *
     * 具体做法就是建立一个松弛了的 cplex 模型，并计算使用的车辆数
     * 如果有 aa 辆未使用车辆就减少 aa 辆可用车辆，否则减少一辆知道没有可行解
     * 当然，最后我们可使用的车辆就是最小的车辆了
     */
    public BaB_VRPTW_v1 init(BaB_VRPTW_v1 lp) throws IloException {
        lp.build_model();

        /*  使用这一部分可以进行预处理，减少车辆数。注释掉这一部分，相当于不做预处理。如果
        // 如果要使用预处理，需要将这一部分打开，并注释掉下方 lp.model.solve(); 这一行
        if (lp.model.solve()) {
            lp.get_value();
            int aa = 0;
            for (int i = 0; i < lp.routes.size(); i++) {
                if (lp.routes.get(i).size() == 2) { //如果路径是 0-102 这样的，就是从 depot 出发，到 depot 结束的，这
↪ 就要删除这个车
                    aa++;
                }
            }
            System.out.println(" 未使用的车辆数是: " + aa);

            if (aa == 0) {
                data.veh_num -= 1;      // 如果未使用的是 0
                lp.model.clearModel();  // clearModel() 是 IloCplex 类中的一个方法
                lp = new Branch_and_Bound_VRPTW(data);
                return init(lp);
            } else {
                data.veh_num -= aa; // 如果未使用的车辆数 >0，那么就要将车辆数减去 aa，然后返回原模型
                lp.model.clearModel();
                lp = new Branch_and_Bound_VRPTW(data); // 更新了 data 中的车辆数，因此重新构建模型
                return init(lp);   // 递归调用函数
            }
        } else {
            data.veh_num += 1;  // 如果 lp 问题不可解，就将车辆数增加一个
            System.out.println("vehicle number: " + data.veh_num);
            lp.model.clearModel();
            lp = new Branch_and_Bound_VRPTW(data);
            lp.build_model();
            if (lp.model.solve()) {
                lp.get_value();
                return lp;
            } else {
                System.out.println("error init");
```

```
373                 return null;
374             }
375 
376         }
377         */
378 
379         lp.model.solve();
380 
381 
382         System.out.println("====================");
383         System.out.println(" root node solution : Obj :" + lp.model.getObjValue());
384         // x_map[i][j][k], 其实就相当于 x[i][j][k], 是为了记录解
385         for (int i = 0; i < data.vertex_num; i++) {
386             for (int j = 0; j < data.vertex_num; j++) {
387                 if (data.arcs[i][j] > 0.5) { // data.arcs[i][j] 取值为 0 或者 1, 这是判断 i 和 j 是都连接, 是否是一条
                    ↪ 弧
388                     for (int k = 0; k < data.veh_num; k++) {
389                         System.out.println("x[" + i + ',' + j + ',' + k + "] = " + model.getValue(x[i][j][k]));
390                         // x_map[i][j][k] = model.getValue(x[i][j][k]); // 将 x[i][j][k] 拷贝到 x_map[i][j][k] 中去
391                     }
392                 }
393             }
394         }
395         lp.get_value();
396 
397         return lp;
398     }
399 
400 
401 
402     /**
403      * branch and bound 算法主体流程
404      *
405      * @param lp:  BaB_VRPTW_v1 的实例
406      * @throws IloException
407      */
408     public void branch_and_bound(BaB_VRPTW_v1 lp) throws IloException {
409         /*
410          * 这个版本的代码没有区分 global_UB 和 local_UB
411          * 直接使用的是 global_UB, 这种做法也是没问题的
412          * 本代码中的 cur_best 就是 global_UB
413          */
414 
415         // 初始化全局的 UB = inf
416         cur_best = Double.MAX_VALUE;
417         deep = 0;
418         record_arc = new int[3];  // 这个是记录可以分支的变量 x[i][j][k]
419         node1 = new Node(data);
420         best_node = null;
421         queue = new PriorityQueue<Node>();
422         // 初始解 (非法解)
423         for (int i = 0; i < lp.routes.size(); i++) {
424             ArrayList<Integer> r = lp.routes.get(i);
425             System.out.println();
426             for (int j = 0; j < r.size(); j++) {
427                 System.out.print(r.get(j) + " ");
428             }
429         }
430         System.out.println("\n\n----branch and bound-----\n");
```

```
431             lp.copy_lp_to_node(lp, node1);
432
433             // node1.node_cost = lp.cost;
434             // node1.lp_x = lp.x_map.clone();
435             // node1.node_routes =lp.routes;
436             // node1.node_servetimes = lp.servetimes;
437
438             node2 = node1.node_copy();
439             deep = 0;
440             node1.d = deep;
441
442             //首先把 node1，也就是初始的 lp 加入到 queue 里面去
443             queue.add(node1);
444
445             // branch and bound 过程
446             int cnt = 0;
447             while (!queue.isEmpty()) {
448                 cnt++;
449                 /*
450                  * remove() 和 poll() 方法的语义也完全相同，都是获取并删除队首元素，
451                  * 区别是当方法失败时前者抛出异常，后者返回 null。
452                  * 由于删除操作会改变队列的结构，为维护小顶堆的性质，需要进行必要的调整。
453                  */
454                 System.out.println("\n\n==============================");
455                 System.out.println(cnt + " Queue length:" + queue.size());
456                 // 弹出队首元素
457                 Node node = queue.poll();
458
459                 /*
460                  * 接下来就是分支定界的过程
461                  * 分支定界的流程是：
462                  * 1. 确定一个下界（初始解 LB），上界 UB 设置为无穷大，或者一个已知的上界。
463                  * 2. 把初始问题构建一个节点加入优先队列（我们使用 best first search，也就是每一次都选择下界最好的节点进行探
    ↪ 索最前搜索）。
464                  * 3. 判断队列是否为空，如果为空跳转至 7，否则取出并弹出队首元素，计算该节点的目标值 P。
465                  * 4. 如果 P > UB，返回 3。否则判断当前节点是否是合法解（对于任意 i,j,k,x_ijk 均为整数），如果是，跳转 5 否则
    ↪ 跳转 6。
466                  * 5. 如果 P < UB，记录 UB = P，当前节点为当前最优解 BS。返回 3.
467                  * 6. 设置两个子节点 L，R。L，R 的建立方式如上，如果 L 的目标值 L.P <= UB，把 L 加入队列，如果 R 的目标值
    ↪ R.P <= UB，把 R 加入队列。返回 3.
468                  * 7. 结束，返回记录的最优节点 BS。如果 BS 为空则无解。
469                  */
470                 // 某支最优解大于（也就是劣于）当前最好可行解，删除（因为 poll() 方法就是弹出队首元素，并将其删除）
471                 if (doubleCompare(node.node_cost, cur_best) > 0) {
472                     continue;
473                 } else {
474                     // 找到可以分支的变量 x[i][j][k]
475                     // record_arc = lp.find_arc(node.lp_x);
476                     record_arc = lp.find_arc1(node.lp_x);
477                     System.out.println("branch variable is : x["
478                             + record_arc[0] + ","
479                             + record_arc[1] + ","
480                             + record_arc[2] + "]");
481                     // 某支的合法解,0,1 组合的解，当前分支最好解
482                     if (record_arc[0] == -1) {
483                         // 如果比当前最好解 cur_best 好，更新当前解
484                         if (doubleCompare(node.node_cost, cur_best) == -1) {
485                             lp.cur_best = node.node_cost;
486                             System.out.println(node.d + "  cur_best:" + cur_best);
```

```
487                         lp.best_node = node;
488                     }
489                     continue;
490                 } else {// 可以分支
491                     node1 = lp.branch_left_arc(lp, node, record_arc);// 左支
492                     if(lp.deep == 0) {
493                         lp.model.exportModel("left.lp");
494                     }
495                     node2 = lp.branch_right_arc(lp, node, record_arc);// 右支
496                     if(lp.deep == 1) {
497                         lp.model.exportModel("right.lp");
498                     }
499 
500                     if (node1 != null
501                             && doubleCompare(node1.node_cost, cur_best) <= 0) {
502                         // 如果 node1 的成本 <= cur_best, 那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
503                         queue.add(node1);
504                     }
505                     if (node2 != null
506                             && doubleCompare(node2.node_cost, cur_best) <= 0) {
507                         // 如果 node1 的成本 <= cur_best, 那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
508                         queue.add(node2);
509                     }
510                     System.out.println(" 当前最优 = " + lp.cur_best);
511                 }
512             }
513         }
514     }
515 
516     /**
517      * 分支设置，使用 setLB 和 setLB 的方式，实现了分支约束的添加
518      * 基于弧 (i, j) 的分支
519      *
520      * @param node
521      * @throws IloException
522      */
523     public void set_bound(Node node) throws IloException {
524         //System.out.println("data.veh_num = " + data.veh_num);
525         for (int i = 0; i < data.vertex_num; i++) {
526             for (int j = 0; j < data.vertex_num; j++) {
527                 if (data.arcs[i][j] > 0.5) {
528                     if (node.node_x[i][j] == 0) {
529                         // 如果 node_x[i][j] = 0, 弧 (i, j) 可以被访问，所以上界为 1，下界为 0
530                         for (int k = 0; k < data.veh_num; k++) {
531                             x[i][j][k].setLB(0.0);
532                             x[i][j][k].setUB(1.0);
533                         }
534                     } else if (node.node_x[i][j] == -1) {
535                         // 如果 node_x[i][j] = -1, 弧 (i, j) 不能被访问，所以上界为 0，下界为 0
536                         for (int k = 0; k < data.veh_num; k++) {
537                             x[i][j][k].setLB(0.0);
538                             x[i][j][k].setUB(0.0);
539                         }
540                     } else {
541                         for (int k = 0; k < data.veh_num; k++) {
542                             // 如果 node_x[i][j] = 1, 弧 (i, j) 必须被访问，所以上界为 1，下界为 1
543                             if (node.node_x_map[i][j][k] == 1) {
544                                 x[i][j][k].setLB(1.0);
545                                 x[i][j][k].setUB(1.0);
```

```
                            } else {
                                x[i][j][k].setLB(0.0);
                                x[i][j][k].setUB(0.0);
                            }
                        }
                    }
                }
            }
        }

        /**
         * 分支设置，使用 setLB 和 setLB 的方式，实现了分支约束的添加
         *     基于变量 x[i][j][k] 的分支
         *
         * @param node
         * @throws IloException
         */
        public void set_bound1(Node node) throws IloException {
            for (int i = 0; i < data.vertex_num; i++) {
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] > 0.5) {
                        for (int k = 0; k < data.veh_num; k++) {
                            if (node.node_x_map[i][j][k] == 0) {
                                x[i][j][k].setLB(0.0);
                                x[i][j][k].setUB(1.0);
                            } else if (node.node_x_map[i][j][k] == -1) {
                                x[i][j][k].setLB(0.0);
                                x[i][j][k].setUB(0.0);
                            } else {
                                x[i][j][k].setLB(1.0);
                                x[i][j][k].setUB(1.0);
                            }
                        }
                    }
                }
            }
        }

        /**
         * 设置左分支
         *
         * @param lp
         * @param father_node
         * @param record
         * @return
         * @throws IloException
         */
        public Node branch_left_arc(BaB_VRPTW_v1 lp, Node father_node, int[] record)
                throws IloException {
            if (record[0] == -1) {
                return null;
            }
            Node new_node = new Node(data);

            // 首先将 father_node 拷贝一份成 new_node
            new_node = father_node.node_copy();

```

```
604             // node_x[i][j] = 1 表示弧 (i, j) 必须被访问， node_x[i][j] = -1 表示不能访问， node_x[i][j] = 0 表示可以访问
                ↪ 也可以不访问
605             // 由于是左支，因此设置 (i, j) 不能被访问
606             new_node.node_x[record[0]][record[1]] = -1;
607
608             // 由于是左支，因此将 x[i][j][k] 设置成 0
609             for (int k = 0; k < data.veh_num; k++) {
610                 new_node.node_x_map[record[0]][record[1]][k] = 0;
611             }
612             // new_node.node_x_map[record[0]][record[1]][record[2]]=-1;
613             // 设置左支
614             lp.set_bound(new_node);
615
616             // 根据 new_node 对应的接的解，更新完了 lp 的上下界之后，就继续求解 lp
617             if (lp.model.solve()) {
618                 lp.get_value();
619                 deep++;
620                 new_node.d = deep;
621                 // 将 lp 的解 copy 到 new_node 中，然后返回 new_node
622                 lp.copy_lp_to_node(lp, new_node);
623                 System.out.println(new_node.d + " left" + " " + lp.cost);
624
625                 // print put the solution of the node
626                 // x_map[i][j][k]，其实就相当于 x[i][j][k]，是为了记录解
627                 /*
628                 System.out.println("\n\n The solution ");
629                 for (int i = 0; i < data.vertex_num; i++) {
630                     for (int j = 0; j < data.vertex_num; j++) {
631                         for (int k = 0; k < data.veh_num; k++) {
632                             if(lp.x_map[i][j][k] > 0) {
633                                 System.out.println(lp.x_map[i][j][k]);
634                             }
635                         }
636                     }
637                 }
638                 */
639             } else {
640                 System.out.println("left branch -- 无解");
641                 new_node.node_cost = data.big_num;
642             }
643             return new_node;
644         }
645
646         /**
647          * 设置右分支
648          * @param lp
649          * @param father_node
650          * @param record
651          * @return
652          * @throws IloException
653          */
654         public Node branch_right_arc(BaB_VRPTW_v1 lp, Node father_node, int[] record)
655                 throws IloException {
656             if (record[0] == -1) {
657                 return null;
658             }
659             Node new_node = new Node(data);
660             new_node = father_node.node_copy();
661
```

```
662             // 由于是左支，因此设置 (i, j) 必须被访问
663             new_node.node_x[record[0]][record[1]] = 1;
664             // new_node.node_x_map[record[0]][record[1]][record[2]]=1;
665
666             // 由于是右支，则设置 x[i][j][k] = 1
667             for (int k = 0; k < data.veh_num; k++) {
668                 if (k == record[2]) {
669                     new_node.node_x_map[record[0]][record[1]][k] = 1;
670                 } else {
671                     new_node.node_x_map[record[0]][record[1]][k] = 0;
672                 }
673             }
674             // 设置右支
675             lp.set_bound(new_node);
676             if (lp.model.solve()) {
677                 lp.get_value();
678                 deep++;
679                 new_node.d = deep;
680                 System.out.println(new_node.d + " right: " + lp.cost);
681                 lp.copy_lp_to_node(lp, new_node);
682
683                 // print put the solution of the node
684                 // x_map[i][j][k], 其实就相当于 x[i][j][k], 是为了记录解
685                 /*
686                 System.out.println("\n\n The solution ");
687                 for (int i = 0; i < data.vertex_num; i++) {
688                     for (int j = 0; j < data.vertex_num; j++) {
689                         for (int k = 0; k < data.veh_num; k++) {
690                             if(lp.x_map[i][j][k] > 0) {
691                                 System.out.println(lp.x_map[i][j][k]);
692                             }
693                         }
694                     }
695                 }
696                 */
697
698             } else {
699                 System.out.println("right branch -- 无解");
700                 new_node.node_cost = data.big_num;
701             }
702             return new_node;
703         }
704
705         /**
706          * 找到需要分支的决策变量 x[i][j][k]
707          *
708          * @param x
709          * @return
710          */
711         public int[] find_arc1(double[][][] x) {
712             int record[] = new int[3];// 记录分支顶点
713             double cur_dif = 0;
714             double min_dif = Double.MAX_VALUE;
715             double branch_var_value = 0;
716
717             // 找出最接近 0.5 的弧
718             for (int i = 1; i < data.vertex_num - 1; i++) {
719                 for (int j = 1; j < data.vertex_num - 1; j++) {
720                     if (data.arcs[i][j] > 0.5) {
```

```
                    for (int k = 0; k < data.veh_num; k++) {
                        // 若该弧值为 0 或 1，则继续
                        if (is_one_zero(x[i][j][k])) {
                            continue;
                        }
                        cur_dif = get_dif(x[i][j][k]);
                        if (doubleCompare(cur_dif, min_dif) == -1) {
                            record[0] = i;
                            record[1] = j;
                            record[2] = k;
                            min_dif = cur_dif;
                            branch_var_value = x[i][j][k];
                        }
                    }
                }
            }
        }

        // depot
        if (doubleCompare(min_dif, Double.MAX_VALUE) == 0) {
            for (int i = 1; i < data.vertex_num - 1; i++) {
                if (data.arcs[0][i] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        if (is_fractional(x[0][i][k])) {
                            cur_dif = get_dif(x[0][i][k]);
                            if (doubleCompare(cur_dif, min_dif) == -1) {
                                record[0] = 0;
                                record[1] = i;
                                record[2] = k;
                                min_dif = cur_dif;
                                branch_var_value = x[0][i][k];
                            }
                        }
                    }
                }
                if (data.arcs[i][data.vertex_num - 1] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        if (is_fractional(x[i][data.vertex_num - 1][k])) {
                            cur_dif = get_dif(x[i][data.vertex_num - 1][k]);
                            if (doubleCompare(cur_dif, min_dif) == -1) {
                                record[0] = i;
                                record[1] = data.vertex_num - 1;
                                record[2] = k;
                                min_dif = cur_dif;
                                branch_var_value = x[i][data.vertex_num - 1][k];
                            }
                        }
                    }
                }
            }
        }

        if (doubleCompare(min_dif, data.big_num) == 1) {
            record[0] = -1;
            record[1] = -1;
            record[2] = -1;
        }
        System.out.println("branch variable value:" + branch_var_value);
        return record;
```

```
780         }
781
782         /**
783          * 找到需要分支的决策变量 x[i][j][k]
784          *
785          * @param x
786          * @return
787          */
788         public int[] find_arc(double[][][] x) {
789             int record[] = new int[3];// 记录分支顶点
790             boolean is_integer = true;
791             for (int i = 0; i < data.vertex_num; i++) {
792                 for (int j = 0; j < data.vertex_num; j++) {
793                     if (data.arcs[i][j] > 0.5) { // if this arc is not 1-1, 2-2 etc.
794                         for (int k = 0; k < data.veh_num; k++) {
795                             // 若该弧值为 0 或 1，则继续
796                             if (is_one_zero(x[i][j][k])) {
797                                 continue;
798                             }
799                             // cur_dif = get_dif(x[i][j][k]);
800                             record[0] = i;
801                             record[1] = j;
802                             record[2] = k;
803                             is_integer = false;
804                             return record;
805                         }
806                     }
807                 }
808             }
809             is_integer = true;
810             System.out.println("The solution is integer!");
811             record[0] = -1;
812             record[1] = -1;
813             record[2] = -1;
814             return record;
815         }
816
817         /**
818          * 比较两个 double 数值的大小
819          *
820          * @param a
821          * @param b
822          * @return
823          */
824         public int doubleCompare(double a, double b) {
825             if (a - b > x_gap)
826                 return 1;
827             if (b - a > x_gap)
828                 return -1;
829             return 0;
830         }
831
832         /**
833          * 判断是否为 0 到 1 之间的小数
834          *
835          * @param v
836          * @return
837          */
838         public boolean is_fractional(double v) {
```

```
839             if (v > (int) v + x_gap && v < (int) v + 1 - x_gap)
840                 return true;
841             else
842                 return false;
843         }
844 
845         /**
846          * 判断是否为 0 或者 1
847          *
848          * @param temp
849          * @return
850          */
851         public boolean is_one_zero(double temp) {
852             if (doubleCompare(temp, 0) == 0 || doubleCompare(temp, 1) == 0) {
853                 return true;
854             } else {
855                 return false;
856             }
857         }
858 
859         /**
860          * 获取到 0.5 的距离
861          *
862          * @param temp
863          * @return
864          */
865         public double get_dif(double temp) {
866             double v = (int) temp + 0.5;
867             if (v > temp) {
868                 return v - temp;
869             } else {
870                 return temp - v;
871             }
872         }
873 
874         /**
875          * 截断小数 3.26434-->3.2
876          *
877          * @param v
878          * @return
879          */
880         public static double double_truncate(double v){
881             int iv = (int) v;
882             if(iv+1 - v <= gap)
883                 return iv+1;
884             double dv = (v - iv) * 10;
885             int idv = (int) dv;
886             double rv = iv + idv / 10.0;
887             return rv;
888         }
889 
890     }
```

10.12.5 run_this 类：算法测试

该类用于测试算法。

run_this.java

```java
package VRPTW_BrandhAndBound;

import java.util.ArrayList;

public class run_this {
    public static void main(String[] args) throws Exception {

        double start = System.currentTimeMillis();
        Data data = new Data();
        int vetexnum = 30 + 2;  // 所有点个数，包括 0，n+1 两个 depot，这两个 depot 是同一个点，只不过 copy 了一份

        // 读入不同的文件前要手动修改 vetexnum 参数，参数值等于所有点个数，包括两个 depot
        String path = "F:\\MyCode\\JavaCode\\JavaCallCplex\\src\\VRPTW_Branch_and_Bound\\c101.txt";// 算例地址

        // 根据 path 和 vetexnum，读取文件中的数据到空的 data 对象中
        /*
         * 读取数据的时候，代码中做了一部分预处理，也就是将一些明显会导致不可行的弧段 (i,j) 删除了
         * 预处理会加快求解速度，减小模型规模
         */
        data.Read_data(path, data, vetexnum);
        data.veh_num = 25;

        Data.printData(data);
        System.out.println("Read data finished!");
        System.out.println(" ------ Branch and Bound ------ ");
        BaB_VRPTW_v1 lp = new BaB_VRPTW_v1(data);
        double cplex_time1 = System.nanoTime();

        /**
         * 删除未用的车辆，减少车辆数，缩小解空间，也就是执行了预处理
         */
        lp = lp.init(lp);
        System.out.println(":   " + lp.data.veh_num);

        /**
         * 尝试不删除未使用的车辆，也就是不执行预处理
         */
        /*
        lp.build_model();
        lp.model.solve();
        lp.get_value();
        */

        // 调用 branch and bound 算法求解 VRPTW
        lp.branch_and_bound(lp);

        // 检验解的合法性
        System.out.println("\n\n================================");
        System.out.println("------ Check the feasibility of the solution---- ");
        Check check = new Check(lp);
        check.fesible();
        double cplex_time2 = System.nanoTime();
        double cplex_time = (cplex_time2 - cplex_time1) / 1e9;// 求解时间，单位 s

        System.out.println("cplex_time : " + cplex_time + " \n"
                + "bestcost : " + lp.cur_best);
        for (int i = 0; i < lp.best_node.node_routes.size(); i++) {
            ArrayList<Integer> r = lp.best_node.node_routes.get(i);
```

```
59              System.out.println();
60              for (int j = 0; j < r.size(); j++) {
61                  System.out.print(r.get(j) + " ");
62              }
63          }
64
65          double end = System.currentTimeMillis();
66          System.out.println("\n\n 程序运行时间: " + (end - start) / 1000.0 + " 秒");
67      }
68  }
```

经过测试，算例 c101-25 的结果如下：

results.java

```
1  bestcost : 191.30000000000004
2
3  0 5 3 7 8 10 11 9 6 4 2 1 0
4  0 20 24 25 23 22 21 0
5  0 13 17 18 19 15 16 14 12 0
6  程序运行时间: 1.385秒
```

10.12.6 算例格式

Solomon VRP Benchmark 算例 C101

```
1  C101
2
3  VEHICLE
4  NUMBER     CAPACITY
5    25         200
6
7  CUSTOMER
8  CUST NO.  XCOORD.   YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE   TIME
9
10     0      40         50          0          0       1236          0
11     1      45         68         10        912        967         90
12     2      45         70         30        825        870         90
13     3      42         66         10         65        146         90
14     4      42         68         10        727        782         90
15     5      42         65         10         15         67         90
16     6      40         69         20        621        702         90
17     7      40         66         20        170        225         90
18     8      38         68         20        255        324         90
19     9      38         70         10        534        605         90
20    10      35         66         10        357        410         90
21    11      35         69         10        448        505         90
22    12      25         85         20        652        721         90
23    13      22         75         30         30         92         90
24    14      22         85         10        567        620         90
25    15      20         80         40        384        429         90
26    16      20         85         40        475        528         90
27    17      18         75         20         99        148         90
28    18      15         75         20        179        254         90
29    19      15         80         10        278        345         90
30    20      30         50         10         10         73         90
31    21      30         52         20        914        965         90
32    22      28         52         20        812        883         90
33    23      28         55         10        732        777         90
```

34	24	25	50	10	65	144	90
35	25	25	52	40	169	224	90
36	26	25	55	10	622	701	90
37	27	23	52	10	261	316	90
38	28	23	55	20	546	593	90
39	29	20	50	10	358	405	90
40	30	20	55	10	449	504	90
41	31	10	35	20	200	237	90
42	32	10	40	30	31	100	90
43	33	8	40	40	87	158	90
44	34	8	45	20	751	816	90
45	35	5	35	10	283	344	90
46	36	5	45	10	665	716	90
47	37	2	40	20	383	434	90
48	38	0	40	30	479	522	90
49	39	0	45	20	567	624	90
50	40	35	30	10	264	321	90
51	41	35	32	10	166	235	90
52	42	33	32	20	68	149	90
53	43	33	35	10	16	80	90
54	44	32	30	10	359	412	90
55	45	30	30	10	541	600	90
56	46	30	32	30	448	509	90
57	47	30	35	10	1054	1127	90
58	48	28	30	10	632	693	90
59	49	28	35	10	1001	1066	90
60	50	26	32	10	815	880	90
61	51	25	30	10	725	786	90
62	52	25	35	10	912	969	90
63	53	44	5	20	286	347	90
64	54	42	10	40	186	257	90
65	55	42	15	10	95	158	90
66	56	40	5	30	385	436	90
67	57	40	15	40	35	87	90
68	58	38	5	30	471	534	90
69	59	38	15	10	651	740	90
70	60	35	5	20	562	629	90
71	61	50	30	10	531	610	90
72	62	50	35	20	262	317	90
73	63	50	40	50	171	218	90
74	64	48	30	10	632	693	90
75	65	48	40	10	76	129	90
76	66	47	35	10	826	875	90
77	67	47	40	10	12	77	90
78	68	45	30	10	734	777	90
79	69	45	35	10	916	969	90
80	70	95	30	30	387	456	90
81	71	95	35	20	293	360	90
82	72	53	30	10	450	505	90
83	73	92	30	10	478	551	90
84	74	53	35	50	353	412	90
85	75	45	65	20	997	1068	90
86	76	90	35	10	203	260	90
87	77	88	30	10	574	643	90
88	78	88	35	20	109	170	90
89	79	87	30	10	668	731	90
90	80	85	25	10	769	820	90
91	81	85	35	30	47	124	90
92	82	75	55	20	369	420	90

```
93    83    72    55    10    265    338    90
94    84    70    58    20    458    523    90
95    85    68    60    30    555    612    90
96    86    66    55    10    173    238    90
97    87    65    55    20     85    144    90
98    88    65    60    30    645    708    90
99    89    63    58    10    737    802    90
100   90    60    55    10     20     84    90
101   91    60    60    10    836    889    90
102   92    67    85    20    368    441    90
103   93    65    85    40    475    518    90
104   94    65    82    10    285    336    90
105   95    62    80    30    196    239    90
106   96    60    80    10     95    156    90
107   97    60    85    30    561    622    90
108   98    58    75    20     30     84    90
109   99    55    80    10    743    820    90
110  100    55    85    20    647    726    90
```

10.13 Java 调用 CPLEX 实现分支定界算法求解 VRPTW(版本 2)：通过添加约束实现分支

在这个版本的代码中，我们使用添加约束的形式，来实现分支定界算法中的分支操作。这里，我们仍然对弧 (i,j) 进行分支。分支过程是通过分左支函数branch_left_arc()和分右支的函数 branch_right_arc()实现的，我们分别在左支和右支对应的节点的模型中，添加一条分支约束。

其它函数功能与之前介绍的类似，只是在分支部分的实现细节上有所不同。

首先，Node类需要做一些小的改变。我们加入了 cut 这个成员变量，用来存储到目前为止，一个分支节点处的所有分支约束。完整代码如下。

10.13.1 更新后的 Node.java

Node.java

```java
package VRPTW_BrandhAndBound_addCut;

import java.util.ArrayList;
import ilog.concert.*;
import ilog.cplex.*;

public class Node implements Comparable{
    /*
     * 这里继承了 Comparable 接口，是因为在实现分支定界的时候，该代码使用了优先队列
     * 存储未探索的节点，即 PriorityQueue.
     */

    Data            data;               // 算例数据
    int               d;                  // 节点的深度
    double          node_cost;          // 该节点的 LP 的目标值
    double[][][]    lp_x;               // 该节点的 LP 的小数解 (x[i][j][k])
    int[][][]       node_x_map;         // node_x[i][j]=1 时，node_x_map[i][j][k]=1 表示必须访问，
    ↪ node_x_map[i][j][k]=0 表示不能访问
```

```
18      int[][]             node_x;                 // node_x[i][j]=0 表示弧 (i, j) 可以访问，1 表示弧 (i, j) 必须访问，-1
        ↪  表示弧 (i, j) 不能访问
19      ArrayList<ArrayList<Integer>> node_routes;        // 车辆路径
20      ArrayList<ArrayList<Double>> node_servetimes;     // 顾客的开始服务时间
21
22      // 新加入的元素
23      IloNumVarArray                  varSet;
24      protected IloRange[]        cuts;
25
26      public Node(Data data) {
27          super();
28          this.data = data;
29          node_cost = data.big_num;
30          lp_x = new double [data.vertex_num][data.vertex_num][data.veh_num];
31          node_x_map = new int[data.vertex_num][data.vertex_num][data.veh_num];
32          node_x = new int[data.vertex_num][data.vertex_num];
33          node_routes = new ArrayList<ArrayList<Integer>>();
34          node_servetimes = new ArrayList<ArrayList<Double>>();
35      }
36
37
38      /**
39       * node 的深度拷贝函数
40       *
41       * 这里使用了 clone 方法
42       * Computer c=new Computer("dell", "4G 内存");
43       * Computer c1=c.Clone();
44       * 在这两句代码中有两个 Computer 类型的对象 c 和 c1，其中 c1 就是通过 Clone 方法复制的 c，
45       * 我们可以使用 System.out.println() 方法将两个对象的内存地址打印出来，会发现是两个不同的值。
46       *
47       * @return new_node
48       */
49      public Node node_copy() {
50          Node new_node = new Node(data);
51          new_node.d = d;
52          new_node.node_cost = node_cost;
53          for (int i = 0; i < lp_x.length; i++) {
54              for (int j = 0; j < lp_x[i].length; j++) {
55                  new_node.lp_x[i][j] = lp_x[i][j].clone();
56              }
57          }
58          for (int i = 0; i < node_x.length; i++) {
59              new_node.node_x[i] = node_x[i].clone();
60          }
61          for (int i = 0; i < node_x_map.length; i++) {
62              for (int j = 0; j < node_x_map[i].length; j++) {
63                  new_node.node_x_map[i][j] = node_x_map[i][j].clone();
64              }
65          }
66          for (int i = 0; i < node_routes.size(); i++) {
67              new_node.node_routes.add((ArrayList<Integer>) node_routes.get(i).clone());
68          }
69          for (int i = 0; i < node_servetimes.size(); i++) {
70              new_node.node_servetimes.add((ArrayList<Double>) node_servetimes.get(i).clone());
71          }
72          return new_node;
73      }
74
75
```

```
76      /**
77       * compareTo:
78       *     比较两个节点实例的大小的函数。如果对象 o 更小，则返回 1
79       *  用于优先队列，是为了找到下一个要处理的节点。
80       */
81      public int compareTo(Object o){
82          Node node = (Node) o;
83          if(node_cost < node.node_cost)
84              return -1;
85          else if(node_cost == node.node_cost)
86              return 0;
87          else
88              return 1;
89      }
90
91
92      /**
93       * 统计变量值与变量数量 (这个函数主要是为了输出求解结果时使用，不写这个函数也是可以的)
94       */
95      static class IloNumVarArray {
96          int _num = 0;   // // _num 标识目前数组中有多少个决策变量
97          IloNumVar[] _array = new IloNumVar[32];
98
99          // 数组不够就增加成两倍长度
100         void add(IloNumVar ivar) {
101             if (_num >= _array.length) {
102                 IloNumVar[] array = new IloNumVar[2 * _array.length];
103                 System.arraycopy(_array, 0, array, 0, _num);
104                 _array = array;
105             }
106             _array[_num++] = ivar;
107         }
108
109         IloNumVar getElement(int i) {
110             return _array[i];
111         }
112
113         IloNumVar getVar(int i) {
114             return _array[i];
115         }
116
117         int getSize() {
118             return _num;   // 获得目前变量数组中有多少个决策变量
119         }
120     }
121
122 }
```

10.13.2 BaB_VRPTW_addCut.java

之前的 BaB_VRPTW_v1.java 类，改为本小节中的 BaB_VRPTW_addCut.java 类。

BaB_VRPTW_addCut.java

```
1  package VRPTW_BrandhAndBound_addCut;
2
3  import java.util.ArrayList;
4  import java.util.PriorityQueue;
5
```

```java
import ilog.concert.*;
import ilog.cplex.*;

/**
 * @source: 本代码的原始版本来自于公众号" 数据魔术师", 原作者: 黄楠, 华中科技大学
 * @comment: 我在原始版本的基础上做了一些注释和一部分修改
 * @author: 黄楠, 华中科技大学
 * @revise: 刘兴禄, 清华大学
 * @date: 2018-8-2
 * @ 操作说明: 读入不同的文件前要手动修改 vetexnum 参数, 参数值为所有点个数, 包括配送中心 0 和 n+1, 代码算例截取于 Solomon 测试算例
 *
 *
 * BaB_VRPTW_v1 的类功能: 建立 VRPTW 模型, 并使用 Branch and bound 算法求解
 */

public class BaB_VRPTW_addCut {
    static double gap = 1e-6;
    Data             data;          // 定义类 Data 的对象
    Node             node1;         // 分支左节点
    Node             node2;       // 分支右节点
    int              deep;          // 深度
    Node             best_node;     // 当前最好节点
    double             cur_best;      // 当前最好解
    int[]             record_arc; // 记录需要分支的变量下标 x[i][j][k] 的 [i][j][k]
    double             x_gap;         // 计算精度容差
    IloCplex         model;          // 模型实例
    double             cost;          // 目标值
    double[][][]     x_map;         // 记录解 x[i][j][k]
    public IloNumVar[][][]     x;  // x[i][j][k] 表示弧 arcs[i][j] 被车辆 k 访问
    public IloNumVar[][]     w;  // 车辆访问所有点的时间矩阵 ** 其实就是 s_ik, 就是第 i 个点被第 k 辆车访问的时间
    public PriorityQueue<Node>         queue; // 分支队列
    ArrayList<ArrayList<Integer>>     routes; // 车辆路径
    ArrayList<ArrayList<Double>>     servetimes; // 顾客的开始服务时间

    // 新加的内容
    IloNumVarArray varSet = new IloNumVarArray();


    /**
     * BaB_VRPTW_addCut 类的构造函数
     * @param data: 算例数据
     */
    public BaB_VRPTW_addCut(Data data) {
        this.data = data;
        x_gap = data.gap;
        routes = new ArrayList<>(); // 定义车辆路径 list
        servetimes = new ArrayList<>(); // 定义花费时间 list
        // 初始化车辆路径和花费时间 list, list 长度为车辆数 k
        for (int k = 0; k < data.veh_num; k++) {
            ArrayList<Integer> r = new ArrayList<>();
            ArrayList<Double> t = new ArrayList<>();
            routes.add(r);
            servetimes.add(t);
        }
        x_map = new double[data.vertex_num][data.vertex_num][data.veh_num];
    }

    /**
```

```
64          * 将 lp 中的数据清除
65          */
66         public void clear_lp() {
67             data = null;
68             routes.clear();
69             servetimes.clear();
70             x_map = null;
71         }
72 
73         /**
74          * 将 lp 解拷贝到 node
75          *
76          * @param lp
77          * @param node
78          */
79         public void copy_lp_to_node(BaB_VRPTW_addCut lp, Node node) {
80             // 首先清除 node 里面的数据
81             node.node_routes.clear();
82             node.node_servetimes.clear();
83 
84             // 然后把 lp 里面的数据 copy 到 node 里面
85             node.node_cost = lp.cost;
86             for (int i = 0; i < lp.x_map.length; i++) {
87                 for (int j = 0; j < lp.x_map[i].length; j++) {
88                     node.lp_x[i][j] = lp.x_map[i][j].clone();
89                 }
90             }
91             for (int i = 0; i < lp.routes.size(); i++) {
92                 node.node_routes.add((ArrayList<Integer>) lp.routes.get(i).clone());
93             }
94             for (int i = 0; i < lp.servetimes.size(); i++) {
95                 node.node_servetimes.add((ArrayList<Double>) lp.servetimes.get(i)
96                         .clone());
97             }
98         }
99 
100        /**
101         * 函数功能：建立 VRPTW 的 cplex 模型
102         * @throws IloException
103         *
104         * 建立模型：这里我们将 VRPTW 标准模型中的整数约束松弛掉，建立 VRPTW 的线性松弛
105         */
106        private void build_model() throws IloException {
107            // creat model
108            model = new IloCplex();
109            // model.setOut(null);    // 关闭 CPLEX 的 log 信息
110            // model.setParam(IloCplex.DoubleParam.EpOpt, 1e-9);  // 设置 tolerance
111            // model.setParam(IloCplex.DoubleParam.EpGap, 1e-9);  // 设置 tolerance
112 
113            // creat decision variables
114            x = new IloNumVar[data.vertex_num][data.vertex_num][data.veh_num];
115            w = new IloNumVar[data.vertex_num][data.veh_num]; // 车辆访问点的时间
116            // 创建决策变量 x 和 w，设置其类型及取值范围
117            for (int i = 0; i < data.vertex_num; i++) {
118                for (int k = 0; k < data.veh_num; k++) {
119                    // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
120                    // w[i][k] = model.numVar(data.a[i], data.b[i], IloNumVarType.Float, "w" + i
121                    //         + "," + k);
122                    System.out.println(data.a[i] + " ---- " + data.b[i]);
```

```
                w[i][k] = model.numVar(0, 1e15, IloNumVarType.Float, "w" + i
                        + "," + k);
            }
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] == 0) {
                    // 如果 i=j, 则该条狐不通
                    x[i][j] = null;
                } else {
                    // x_ijk
                    for (int k = 0; k < data.veh_num; k++) {
                        // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
                        x[i][j][k] = model.numVar(0, 1, IloNumVarType.Float,
                                "x" + i + "," + j + "," + k);
                    }
                }
            }
        }
        // 加入目标函数
        // 表达式 (7.1.1)
        IloNumExpr obj = model.numExpr();
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] == 0) {
                    System.out.println("delected arc : " + i + ", " + j);
                    continue;
                }
                for (int k = 0; k < data.veh_num; k++) {
                    obj = model.sum(obj,
                            model.prod(data.dist[i][j], x[i][j][k]));
                }
            }
        }
        model.addMinimize(obj);
        // 加入约束 1
        // 表达式 (7.1.2)
        for (int i = 1; i < data.vertex_num - 1; i++) {
            IloNumExpr expr1 = model.numExpr();
            for (int k = 0; k < data.veh_num; k++) {
                for (int j = 1; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr1 = model.sum(expr1, x[i][j][k]);
                    }
                }
            }
            model.addEq(expr1, 1);
        }
        // 加入约束 2
        // 表达式 (7.1.3)
        for (int k = 0; k < data.veh_num; k++) {
            IloNumExpr expr2 = model.numExpr();
            for (int j = 1; j < data.vertex_num; j++) {
                if (data.arcs[0][j] == 1) {
                    expr2 = model.sum(expr2, x[0][j][k]);
                }
            }
            model.addEq(expr2, 1);
        }
        // 加入约束 3
        // 表达式 (7.1.4)
```

```
182             for (int k = 0; k < data.veh_num; k++) {
183                 for (int j = 1; j < data.vertex_num - 1; j++) {
184                     IloNumExpr expr3 = model.numExpr();
185                     IloNumExpr subExpr1 = model.numExpr();
186                     IloNumExpr subExpr2 = model.numExpr();
187                     for (int i = 0; i < data.vertex_num; i++) {
188                         if (data.arcs[i][j] == 1) {
189                             subExpr1 = model.sum(subExpr1, x[i][j][k]);
190                         }
191                         if (data.arcs[j][i] == 1) {
192                             subExpr2 = model.sum(subExpr2, x[j][i][k]);
193                         }
194                     }
195                     expr3 = model.sum(subExpr1, model.prod(-1, subExpr2));
196                     model.addEq(expr3, 0);
197                 }
198             }
199             // 加入约束 4
200             // 表达式 (7.1.5)
201             for (int k = 0; k < data.veh_num; k++) {
202                 IloNumExpr expr4 = model.numExpr();
203                 for (int i = 0; i < data.vertex_num - 1; i++) {
204                     if (data.arcs[i][data.vertex_num - 1] == 1) {
205                         expr4 = model.sum(expr4, x[i][data.vertex_num - 1][k]);
206                     }
207                 }
208                 model.addEq(expr4, 1);
209             }
210             // 加入约束 5
211             // 表达式 (7.1.6)
212             for (int k = 0; k < data.veh_num; k++) {
213                 IloNumExpr expr8 = model.numExpr();
214                 for (int i = 1; i < data.vertex_num - 1; i++) {
215                     IloNumExpr expr9 = model.numExpr();
216                     for (int j = 0; j < data.vertex_num; j++) {
217                         if (data.arcs[i][j] == 1) {
218                             expr9 = model.sum(expr9, x[i][j][k]);
219                         }
220                     }
221                     expr8 = model.sum(expr8, model.prod(data.demands[i], expr9));
222                 }
223                 model.addLe(expr8, data.cap);
224                 model.exportModel("VRPTW_LP.lp");
225             }
226             // 加入约束 6
227             // 表达式 (7.1.7)
228             double M = 1e5;
229             for (int k = 0; k < data.veh_num; k++) {
230                 for (int i = 0; i < data.vertex_num; i++) {
231                     for (int j = 0; j < data.vertex_num; j++) {
232                         if (data.arcs[i][j] == 1) {
233                             IloNumExpr expr5 = model.numExpr();
234                             IloNumExpr expr6 = model.numExpr();
235                             expr5 = model.sum(w[i][k], data.s[i] + data.dist[i][j]);
236                             expr5 = model.sum(expr5, model.prod(-1, w[j][k]));
237                             expr6 = model.prod(M,
238                                     model.sum(1, model.prod(-1, x[i][j][k])));
239                             model.addLe(expr5, expr6);
240                         }
```

```
                }
            }
        }
        // 加入约束 7
        // 时间窗约束
        for (int k = 0; k < data.veh_num; k++) {
            for (int i = 1; i < data.vertex_num - 1; i++) {
                IloNumExpr expr7 = model.numExpr();
                for (int j = 0; j < data.vertex_num; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr7 = model.sum(expr7, x[i][j][k]);
                    }
                }
                model.addLe(model.prod(data.a[i], expr7), w[i][k]);
                model.addLe(w[i][k], model.prod(data.b[i], expr7));
            }
        }
        // 加入约束 7
        // 表达式 (7.1.8)
        for (int k = 0; k < data.veh_num; k++) {
            model.addLe(data.E, w[0][k]);
            model.addLe(data.E, w[data.vertex_num - 1][k]);
            model.addLe(w[0][k], data.L);
            model.addLe(w[data.vertex_num - 1][k], data.L);
        }

    }


    /**
     * 函数功能：解模型，并生成车辆路径和得到目标值
     * @throws IloException
     *
     * 获取 VRPTW 模型的线性松弛的解，在根节点或者在分支节点处被调用
     */
    public void get_value() throws IloException {
        routes.clear();
        servetimes.clear();
        cost = 0;

        // 初始化车辆路径和花费时间 list (空 list)，list 长度为车辆数 k
        for (int k = 0; k < data.veh_num; k++) {
            ArrayList<Integer> r = new ArrayList<>();
            ArrayList<Double> t = new ArrayList<>();
            routes.add(r);
            servetimes.add(t);
        }

        // x_map[i][j][k]，其实就相当于 x[i][j][k]，是为了记录解
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                for (int k = 0; k < data.veh_num; k++) {
                    x_map[i][j][k] = 0.0; //首先将 x[i][j][k] 初始化为 0
                }
                if (data.arcs[i][j] > 0.5) { // data.arcs[i][j] 取值为 0 或者 1，这是判断 i 和 j 是都连接，是否是一条
                ↪ 弧
                    for (int k = 0; k < data.veh_num; k++) {
                        x_map[i][j][k] = model.getValue(x[i][j][k]); // 将 x[i][j][k] 拷贝到 x_map[i][j][k] 中去
                    }
```

```
                }
            }
        }

        // 模型可解，从解 x[i][j][k] 中循环提取出车辆路径
        for (int k = 0; k < data.veh_num; k++) {
            boolean terminate = true;
            int i = 0;
            routes.get(k).add(0); // 加入 0，拼成初始的 0-...
            servetimes.get(k).add(0.0);
            while (terminate) {
                for (int j = 0; j < data.vertex_num; j++) {
                    if (doubleCompare(x_map[i][j][k], 0) == 1) {  // 如果 x_map[i][j][k] > 0
                        routes.get(k).add(j);
                        servetimes.get(k).add(model.getValue(w[j][k]));
                        i = j;
                        break;
                    }
                }
                if (i == data.vertex_num - 1) {
                    terminate = false;
                }
            }
            routes.get(k).set(routes.get(k).size() - 1, 0);
        }
        cost = model.getObjValue();
    }

    /**
     * init() 函数是确定有合法解的最小车辆数，由于直接求解解空间太大，且有很多车辆不能使用
     * 因此，我们删除无用的车辆，来缩小解空间（这是一个小优化，能够加快程序速度）
     *
     * 具体做法就是建立一个松弛了的 cplex 模型，并计算使用的车辆数
     * 如果有 aa 辆未使用车辆就减少 aa 辆可用车辆，否则减少一辆知道没有可行解
     * 当然，最后我们可使用的车辆就是最小的车辆了
     */
    public BaB_VRPTW_addCut init(BaB_VRPTW_addCut lp) throws IloException {
        lp.build_model();
        if (lp.model.solve()) {
            lp.get_value();
            int aa = 0;
            for (int i = 0; i < lp.routes.size(); i++) {
                if (lp.routes.get(i).size() == 2) { //如果路径是 0-102 这样的，就是从 depot 出发，到 depot 结束的，这
                ↪  就要删除这个车
                    aa++;
                }
            }
            System.out.println(" 未使用的车辆数是：" + aa);

            if (aa == 0) {
                data.veh_num -= 1;      // 如果未使用的是 0
                lp.model.clearModel();  // clearModel() 是 IloCplex 类中的一个方法
                lp = new BaB_VRPTW_addCut(data);
                return init(lp);
            } else {
                data.veh_num -= aa; // 如果未使用的车辆数 >0，那么就要将车辆数减去 aa，然后返回原模型
                lp.model.clearModel();
                lp = new BaB_VRPTW_addCut(data); // 更新了 data 中的车辆数，因此重新构建模型
                return init(lp);   // 递归调用函数
```

```
357                 }
358             } else {
359                 data.veh_num += 1;  // 如果 lp 问题不可解，就将车辆数增加一个
360                 System.out.println("vehicle number: " + data.veh_num);
361                 lp.model.clearModel();
362                 lp = new BaB_VRPTW_addCut(data);
363                 lp.build_model();
364                 if (lp.model.solve()) {
365                     lp.get_value();
366                     return lp;
367                 } else {
368                     System.out.println("error init");
369                     return null;
370                 }
371             }
372         }
373
374
375
376         /**
377          * branch and bound 算法主体流程
378          *
379          * @param lp:  BaB_VRPTW_v1 的实例
380          * @throws IloException
381          */
382         public void branch_and_bound(BaB_VRPTW_addCut lp) throws IloException {
383             /*
384              * 这个版本的代码没有区分 global_UB 和 local_UB
385              * 直接使用的是 global_UB，这种做法也是没问题的
386              * 本代码中的 cur_best 就是 global_UB
387              */
388
389             // 初始化全局的 UB = inf
390             cur_best = Double.MAX_VALUE;
391             deep = 0;
392             record_arc = new int[3];  // 这个是记录可以分支的变量 x[i][j][k]
393             node1 = new Node(data);
394             best_node = null;
395             queue = new PriorityQueue<Node>();
396             // 初始解（非法解）
397             for (int i = 0; i < lp.routes.size(); i++) {
398                 ArrayList<Integer> r = lp.routes.get(i);
399                 System.out.println();
400                 for (int j = 0; j < r.size(); j++) {
401                     System.out.print(r.get(j) + " ");
402                 }
403             }
404             System.out.println("\n\n");
405             lp.copy_lp_to_node(lp, node1);
406
407             node2 = node1.node_copy();
408             deep = 0;
409             node1.d = deep;
410
411             //首先把 node1，也就是初始的 lp 加入到 queue 里面去
412             queue.add(node1);
413             // branch and bound 过程
414             while (!queue.isEmpty()) {
415                 /*
```

```
416                 * remove() 和 poll() 方法的语义也完全相同，都是获取并删除队首元素，
417                 * 区别是当方法失败时前者抛出异常，后者返回 null。
418                 * 由于删除操作会改变队列的结构，为维护小顶堆的性质，需要进行必要的调整。
419                 */
420                // 弹出队首元素
421                Node node = queue.poll();
422
423                /*
424                 * 接下来就是分支定界的过程
425                 * 分支定界的流程是:
426                 * 1. 确定一个下界（初始解 LB），上界 UB 设置为无穷大，或者一个已知的上界。
427                 * 2. 把初始问题构建一个节点加入优先队列（我们使用 best first search，也就是每一次都选择下界最好的节点进行探索最前搜索）。
428                 * 3. 判断队列是否为空，如果为空跳转至 7，否则取出并弹出队首元素，计算该节点的目标值 P。
429                 * 4. 如果 P > UB，返回 3。否则判断当前节点是否是合法解（对于任意 i,j,k,x_ijk 均为整数），如果是，跳转 5 否则跳转 6。
430                 * 5. 如果 P < UB，记录 UB = P，当前节点为当前最优解 BS。返回 3.
431                 * 6. 设置两个子节点 L，R。L，R 的建立方式如上，如果 L 的目标值 L.P <= UB，把 L 加入队列，如果 R 的目标值 R.P <= UB，把 R 加入队列。返回 3.
432                 * 7. 结束，返回记录的最优节点 BS。如果 BS 为空则无解。
433                 */
434                // 某支最优解大于（也就是劣于）当前最好可行解，删除（因为 poll() 方法就是弹出队首元素，并将其删除）
435                if (doubleCompare(node.node_cost, cur_best) > 0) {
436                    continue;
437                } else {
438                    // 找到可以分支的变量 x[i][j][k]
439                    record_arc = lp.find_arc(node.lp_x);
440                    /*
441                    System.out.println("branch variable = "
442                                       + record_arc[0] + "-"
443                                       + record_arc[1] + "-"
444                                       + record_arc[2] );
445                    */
446                    // 某支的合法解,0,1 组合的解，当前分支最好解
447                    if (record_arc[0] == -1) {
448                        // 如果比当前最好解 cur_best 好，更新当前解
449                        if (doubleCompare(node.node_cost, cur_best) == -1) {
450                            lp.cur_best = node.node_cost;
451                            System.out.println(node.d + "  cur_best:" + cur_best);
452                            lp.best_node = node;
453                        }
454                        continue;
455                    } else {// 可以分支
456                        node1 = lp.branch_left_arc(lp, node, record_arc);// 左支
457                        node2 = lp.branch_right_arc(lp, node, record_arc);// 右支
458                        if(lp.deep == 0) {
459                            lp.model.exportModel("right_cut.lp");
460                        }
461
462                        if (node1 != null
463                                && doubleCompare(node1.node_cost, cur_best) <= 0) {
464                            // 如果 node1 的成本 <= cur_best，那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
465                            queue.add(node1);
466                        }
467                        if (node2 != null
468                                && doubleCompare(node2.node_cost, cur_best) <= 0) {
469                            // 如果 node1 的成本 <= cur_best，那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
470                            queue.add(node2);
471                        }
```

```
                }
            }
        }
    }

    // 分支设置：如果采用设置变量 Bound 的做法，就是调用 set_bound 函数
    /*
    public void set_bound(Node node) throws IloException {
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] > 0.5) {
                    if (node.node_x[i][j] == 0) {
                        // 如果 node_x[i][j] = 0，弧 (i, j) 可以被访问，所以上界为 1，下界为 0
                        for (int k = 0; k < data.veh_num; k++) {
                            x[i][j][k].setLB(0.0);
                            x[i][j][k].setUB(1.0);
                        }
                    } else if (node.node_x[i][j] == -1) {
                        // 如果 node_x[i][j] = -1，弧 (i, j) 不能被访问，所以上界为 0，下界为 0
                        for (int k = 0; k < data.veh_num; k++) {
                            x[i][j][k].setLB(0.0);
                            x[i][j][k].setUB(0.0);
                        }
                    } else {
                        for (int k = 0; k < data.veh_num; k++) {
                            // 如果 node_x[i][j] = 1，弧 (i, j) 必须被访问，所以上界为 1，下界为 1
                            if (node.node_x_map[i][j][k] == 1) {
                                x[i][j][k].setLB(1.0);
                                x[i][j][k].setUB(1.0);
                            } else {
                                x[i][j][k].setLB(0.0);
                                x[i][j][k].setUB(0.0);
                            }
                        }
                    }
                }
            }
        }
     */
    /*
    public void set_bound1(Node node) throws IloException {
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        if (node.node_x_map[i][j][k] == 0) {
                            x[i][j][k].setLB(0.0);
                            x[i][j][k].setUB(1.0);
                        } else if (node.node_x_map[i][j][k] == -1) {
                            x[i][j][k].setLB(0.0);
                            x[i][j][k].setUB(0.0);
                        } else {
                            x[i][j][k].setLB(1.0);
                            x[i][j][k].setUB(1.0);
                        }
                    }
                }
            }
```

```
531             }
532         }
533         */
534 
535         // 设置左支
536         public Node branch_left_arc(BaB_VRPTW_addCut lp, Node father_node, int[] record)
537                 throws IloException {
538             if (record[0] == -1) {
539                 return null;
540             }
541             Node new_node = new Node(data);
542 
543             // 首先将 father_node 拷贝一份成 new_node
544             new_node = father_node.node_copy();
545 
546             // node_x[i][j] = 1 表示弧 (i, j) 必须被访问， node_x[i][j] = -1 表示不能访问， node_x[i][j] = 0 表示可以访问
                ↪ 也可以不访问
547             // 由于是左支，因此设置 (i, j) 不能被访问
548             new_node.node_x[record[0]][record[1]] = -1;
549 
550             // 由于是左支，因此将 x[i][j][k] 设置成 0
551             for (int k = 0; k < data.veh_num; k++) {
552                 new_node.node_x_map[record[0]][record[1]][k] = 0;
553             }
554             new_node.node_x_map[record[0]][record[1]][record[2]]=-1;
555             // 设置左支
556             // lp.set_bound(new_node);  如果不加 cut 就是这样的语句
557 
558             // ----------------下面是加 cut 的做法-----------------------
559             // 左支：需要令所有 x[i][j][k] == 0
560             //System.out.println(" 父节点 cut 数量" + father_node.cuts.length);
561 
562             //System.out.println("\n 添加 left 支 cut 之前 lp 中的约束个数" + lp.model.getNrows());
563             // 拷贝父节点的 cuts （这里可以用深度拷贝，结合 getExpr 函数实现，但是暂时没有必要）
564 
565             if(father_node.cuts != null) {
566                 new_node.cuts = new IloRange[father_node.cuts.length + lp.data.veh_num];
567                 for(int i = 0; i < father_node.cuts.length; i++) {
568                     new_node.cuts[i] = father_node.cuts[i];
569                     // 先要把父节点的 cut 加入到 lp.model 中去
570                     //lp.model.addCut(father_node.cuts[i]);
571                     lp.model.addRange(father_node.cuts[i].getLB(), father_node.cuts[i].getExpr(),
                        ↪ father_node.cuts[i].getUB());
572                 }
573                 // 增加新的 cuts
574                 for(int i = father_node.cuts.length; i < new_node.cuts.length; i++) {
575                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                        ↪ father_node.cuts.length], 0));
576                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i - father_node.cuts.length], 0);
577                 }
578 
579             }else {
580                 // 增加新的 cuts
581                 new_node.cuts = new IloRange[lp.data.veh_num];
582                 for(int i = 0; i < new_node.cuts.length; i++) {
583                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i], 0));
584                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i], 0);
585                 }
586             }
```

```
587
588
589             System.out.println("copy 成功");
590             System.out.println(" 添加 cut 结束");
591             System.out.println(" 添加 left 支 cut 之后 lp 中的约束个数" + lp.model.getNrows());
592             // ---------------------cut 添加结
                ↪ 束----------------------------------------------------------------------------------
593
594             if(lp.deep == 0) {
595                 lp.model.exportModel("left_cut.lp");
596             }
597
598             // 根据 new_node 对应的接的解，更新完了 lp 的上下界之后，就继续求解 lp
599             if (lp.model.solve()) {
600                 lp.get_value();
601                 deep++;
602                 new_node.d = deep;
603                 // 将 lp 的解 copy 到 new_node 中，然后返回 new_node
604                 lp.copy_lp_to_node(lp, new_node);
605                 System.out.println(new_node.d + " left" + " " + lp.cost);
606
607                 // 清理 cuts
608                 //lp.model.clearCuts();
609                 for(int i = 0; i < new_node.cuts.length; i++) {
610                     lp.model.remove(new_node.cuts[i]);
611                 }
612                 //System.out.println(new_node.cuts.length);
613
614                 //System.out.println(" 左支分完删除后模型的约束个数" + lp.model.getNrows());
615             } else {
616                 System.out.println(" 分左支求解无解");
617                 for(int i = 0; i < new_node.cuts.length; i++) {
618                     lp.model.remove(new_node.cuts[i]);
619                 }
620                 //System.out.println("\n 左支分完删除后模型的约束个数" + lp.model.getNrows());
621                 //System.out.println("\n");
622                 new_node.node_cost = data.big_num;
623             }
624             return new_node;
625         }
626
627         // 设置右支
628         public Node branch_right_arc(BaB_VRPTW_addCut lp, Node father_node, int[] record)
629                 throws IloException {
630             if (record[0] == -1) {
631                 return null;
632             }
633             Node new_node = new Node(data);
634             new_node = father_node.node_copy();
635
636             // 由于是左支，因此设置 (i, j) 必须被访问
637             new_node.node_x[record[0]][record[1]] = 1;
638             new_node.node_x_map[record[0]][record[1]][record[2]]=1;
639
640             // 由于是右支，则设置 x[i][j][k] = 1
641             for (int k = 0; k < data.veh_num; k++) {
642                 if (k == record[2]) {
643                     new_node.node_x_map[record[0]][record[1]][k] = 1;
644                 } else {
```

```
                new_node.node_x_map[record[0]][record[1]][k] = 0;
            }
        }



        // 设置右支
        //lp.set_bound(new_node);

        // ---------------------下面是加 cut 的做法-----------------------------------------------------------------------
        // 左支：需要令所有 x[i][j][k] == 1，其余的均为 0
        // 首先初始化长度

        // 拷贝父节点的 cuts （这里可以用深度拷贝，结合 getExpr 函数实现，但是暂时没有必要）
        //System.out.println("**************");
        //System.out.println(" 添加 right 支 cut 之前 lp 中的约束个数" + lp.model.getNrows());
        if(father_node.cuts != null) {
            //System.out.println(" 进入 if");
            new_node.cuts = new IloRange[father_node.cuts.length + lp.data.veh_num];
            for(int i = 0; i < father_node.cuts.length; i++) {
                new_node.cuts[i] = father_node.cuts[i];
                // 先要把父节点的 cut 加入到 lp.model 中去
                //lp.model.addCut(father_node.cuts[i]);
                lp.model.addRange(father_node.cuts[i].getLB(), father_node.cuts[i].getExpr(),
                    father_node.cuts[i].getUB());
            }
            // 增加新的 cuts
            for(int i = father_node.cuts.length; i < new_node.cuts.length; i++) {
                if(i - father_node.cuts.length == record[2]) {
                    //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][record[2]], 1));
                    new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][record[2]], 1);
                }else {
                    //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                    father_node.cuts.length], 0));
                    new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i - father_node.cuts.length], 0);
                }

            }

        }else {
            // 增加新的 cuts
            //System.out.println(" 进入 else");
            new_node.cuts = new IloRange[lp.data.veh_num];

            // 增加新的 cuts
            for(int i = 0; i < new_node.cuts.length; i++) {
                //System.out.println(i);
                if(i == record[2]) {
                    //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][record[2]], 1));
                    new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][record[2]], 1);
                    //System.out.println(new_node.cuts[i].toString());
                }else {
                    //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i], 0));
                    new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i], 0);
                    //System.out.println(new_node.cuts[i].toString());
                }

            }
```

```
701             }
702             System.out.println("copy 成功");
703 
704             System.out.println(" 添加 cut 结束");
705             System.out.println("**************");
706             System.out.println(" 添加 right 支 cut 之后 lp 中的约束个数" + lp.model.getNrows());
707             // ---------------------cut 添加结
                ↪ 束----------------------------------------------------------------------------------
708             System.out.println("lp.deep" + lp.deep);
709             if(lp.deep == 0) {
710                 System.out.println(" 导出模型");
711                 lp.model.exportModel("right_cut.lp");
712             }
713             lp.model.solve();
714 
715 
716             if (lp.model.solve()) {
717                 lp.get_value();
718                 deep++;
719                 new_node.d = deep;
720                 System.out.println(new_node.d + " right: " + lp.cost);
721                 lp.copy_lp_to_node(lp, new_node);
722 
723                 // 清理 cuts
724                 //lp.model.clearCuts();
725                 for(int i = 0; i < new_node.cuts.length; i++) {
726                     lp.model.remove(new_node.cuts[i]);
727                 }
728                 //System.out.println("right 支分完删除后模型的约束个数" + lp.model.getNrows());
729 
730             } else {
731                 System.out.println(" 分 right 支求解无解");
732                 for(int i = 0; i < new_node.cuts.length; i++) {
733                     lp.model.remove(new_node.cuts[i]);
734                 }
735                 //System.out.println("right 支求解无解删除后模型的约束个数" + lp.model.getNrows());
736                 new_node.node_cost = data.big_num;
737             }
738             //System.out.println("====== 分支完毕------");
739             return new_node;
740         }
741 
742         // 找到需要分支的支点位置
743         public int[] find_arc1(double[][][] x) {
744             int record[] = new int[3];// 记录分支顶点
745             double cur_dif = 0;
746             double min_dif = Double.MAX_VALUE;
747             // 找出最接近 0.5 的弧
748             for (int i = 1; i < data.vertex_num - 1; i++) {
749                 for (int j = 1; j < data.vertex_num - 1; j++) {
750                     if (data.arcs[i][j] > 0.5) {
751                         for (int k = 0; k < data.veh_num; k++) {
752                             // 若该弧值为 0 或 1，则继续
753                             if (is_one_zero(x[i][j][k])) {
754                                 continue;
755                             }
756                             cur_dif = get_dif(x[i][j][k]);
757                             if (doubleCompare(cur_dif, min_dif) == -1) {
758                                 record[0] = i;
```

```
                            record[1] = j;
                            record[2] = k;
                            min_dif = cur_dif;
                        }
                    }
                }
            }
        }
        // depot
        if (doubleCompare(min_dif, Double.MAX_VALUE) == 0) {
            for (int i = 1; i < data.vertex_num - 1; i++) {
                if (data.arcs[0][i] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        if (is_fractional(x[0][i][k])) {
                            cur_dif = get_dif(x[0][i][k]);
                            if (doubleCompare(cur_dif, min_dif) == -1) {
                                record[0] = 0;
                                record[1] = i;
                                record[2] = k;
                                min_dif = cur_dif;
                            }
                        }
                    }
                }
                if (data.arcs[i][data.vertex_num - 1] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        if (is_fractional(x[i][data.vertex_num - 1][k])) {
                            cur_dif = get_dif(x[i][data.vertex_num - 1][k]);
                            if (doubleCompare(cur_dif, min_dif) == -1) {
                                record[0] = i;
                                record[1] = data.vertex_num - 1;
                                record[2] = k;
                                min_dif = cur_dif;
                            }
                        }
                    }
                }
            }
        }
        if (doubleCompare(min_dif, data.big_num) == 1) {
            record[0] = -1;
            record[1] = -1;
            record[2] = -1;
        }
        return record;
    }

    // 找到要分支的弧
    public int[] find_arc(double[][][] x) {
        int record[] = new int[3];// 记录分支顶点
        for (int i = 0; i < data.vertex_num; i++) {
            for (int j = 0; j < data.vertex_num; j++) {
                if (data.arcs[i][j] > 0.5) {
                    for (int k = 0; k < data.veh_num; k++) {
                        // 若该弧值为 0 或 1，则继续
                        if (is_one_zero(x[i][j][k])) {
                            continue;
                        }
                        // cur_dif = get_dif(x[i][j][k]);
```

```
                        record[0] = i;
                        record[1] = j;
                        record[2] = k;
                        return record;
                    }
                }
            }
        }
        record[0] = -1;
        record[1] = -1;
        record[2] = -1;
        return record;
    }

    // 比较两个 double 数值的大小
    public int doubleCompare(double a, double b) {
        if (a - b > x_gap)
            return 1;
        if (b - a > x_gap)
            return -1;
        return 0;
    }

    // 判断是否为 0 到 1 之间的小数
    public boolean is_fractional(double v) {
        if (v > (int) v + x_gap && v < (int) v + 1 - x_gap)
            return true;
        else
            return false;
    }

    // 判断是否为 0 或者 1
    public boolean is_one_zero(double temp) {
        if (doubleCompare(temp, 0) == 0 || doubleCompare(temp, 1) == 0) {
            return true;
        } else {
            return false;
        }
    }

    // 获取到 0.5 的距离
    public double get_dif(double temp) {
        double v = (int) temp + 0.5;
        if (v > temp) {
            return v - temp;
        } else {
            return temp - v;
        }
    }

    // 截断小数 3.26434-->3.2
    public static double double_truncate(double v){
        int iv = (int) v;
        if(iv+1 - v <= gap)
            return iv+1;
        double dv = (v - iv) * 10;
        int idv = (int) dv;
        double rv = iv + idv / 10.0;
        return rv;
```

```
877         }
878
879         // 统计变量值与变量数量（这个函数主要是为了输出求解结果时使用，不写这个函数也是可以的）
880         static class IloNumVarArray {
881             int _num = 0;   // _num 标识目前数组中有多少个决策变量
882             IloNumVar[] _array = new IloNumVar[32];
883
884             // 数组不够就增加成两倍长度
885             void add(IloNumVar ivar) {
886                 if (_num >= _array.length) {
887                     IloNumVar[] array = new IloNumVar[2 * _array.length];
888                     System.arraycopy(_array, 0, array, 0, _num);
889                     _array = array;
890                 }
891                 _array[_num++] = ivar;
892             }
893
894             IloNumVar getElement(int i) {
895                 return _array[i];
896             }
897
898             IloNumVar getVar(int i) {
899                 return _array[i];
900             }
901
902             int getSize() {
903                 return _num;   // 获得目前变量数组中有多少个决策变量
904             }
905         }
906     }
```

第 11 章　分支切割算法

分支切割算法 (Branch and Cut Algorithm) 是非常强大的求解混合整数规划的精确算法，本书涉及的两款求解器 Gurobi 和 CPLEX 的主体算法框架都是分支切割算法。该算法由 Padberg, Manfred 和 Rinaldi, Giovanni 于 1991 年提出，用于求解大规模 STSP 问题 (Padberg and Rinaldi 1991)。目前为止，分支切割算法是求解一般的混合整数规划 (即不依赖具体问题特性的混合整数规划) 问题的最有效的精确算法。

11.1 什么是分支切割算法

11.2 有效不等式

11.3 割平面算法

11.3.1 Gomory's 分数割平面算法

Gomory's 分数割平面算法由 R.E. Gomory 于 1958 年首次提出 (Gomory 1958)。

11.3.2 其他割平面算法

11.4 分支切割算法: 分支定界 + 割平面

11.4.1 分支切割算法伪代码

11.4.2 分支切割算法: 一个详细的例子

11.5 Java 调用 CPLEX 实现分支切割算法求解 VRPTW：用户实现分支和割平面的版本

11.5.1 分支定界

11.5.2 割平面

11.6 Java 调用 CPLEX 实现分支切割算法求解 VRPTW 完整代码：介绍

介绍完分支策略和 Cutting Plane 之后，我们来讨论具体的代码实现。在代码中，我们采用第一种分支策略，即在每个具有小数解的 BB tree 的子节点处，找到取值最接近 0.5 的 x_{ijk} 进行分支。在生成 k-path cuts 的过程中，我们随机地选择一些点，构成集合 S，然后构建 k-path cuts。在每一次分支过程中，我们最多添加 5 条 Cut。

本节的 Branch and Cut 代码中的关于 Branch and Bound 的部分与第 10.11节中的相同（即与数据魔术师公众号中黄楠博士提供的代码基本相同）。本书作者在此基础上新增了如下内容，方便读者进行对照。

- `VRPTW_Branch_and_Cut`类中的 Cutting plane 的部分为新增内容；
- `MyRandom`类为新增的类。

其余的类文件改动较小。此外，Cutting plane 的部分旨在给出实现的样例，并不保证高效。对求解效率有要求的读者可以进行相应的优化。

实现 Branch and cut 一般有 2 种方法：

1. 自行实现 branch and bound 和 cutting plane 的部分；
2. branch and bound 和通用的 cutting plane 部分调用求解器的算法，但是定制化的 cutting plane 用 callback 实现。

一般来讲，第二种实现方法更加高效，代码量也较小，因为求解器的 branch and bound 和 cutting plane 算法已经经过了充分的优化和提速。自行实现所有模块的方式，需要撰写大量实现 branch and bound 算法的代码，实现较为困难。本书针对以上两种方式，各提供了一套基础代码，供读者学习使用。

下面是 Java 调用 CPLEX 实现 Branch and cut 求解 VRPTW 的完整 Java 代码。

11.7 Java 调用 CPLEX 实现分支切割算法求解 VRPTW 完整代码：自行实现 branch and bound 和 cutting plane 的版本

11.7.1 Node 类

Node.java

```java
package BranchAndCut_addCut;

import java.util.ArrayList;
import ilog.concert.*;
import ilog.cplex.*;

public class Node implements Comparable{
    /*
     * 这里继承了 Comparable 接口，是因为在实现分支定界的时候，该代码使用了优先队列
     * 存储未探索的节点，即 PriorityQueue.
     */

    Data            data;               // 算例数据
    int               d;                    // 节点的深度
    double           node_cost;           // 该节点的 LP 的目标值
    double[][][]    lp_x;               // 该节点的 LP 的小数解 (x[i][j][k])
    int[][][]        node_x_map;           // node_x[i][j]=1 时，node_x_map[i][j][k]=1 表示必须访问，
    ↪ node_x_map[i][j][k]=0 表示不能访问
    int[][]           node_x;               // node_x[i][j]=0 表示弧 (i, j) 可以访问，1 表示弧 (i, j) 必须访问，-1
    ↪ 表示弧 (i, j) 不能访问
    ArrayList<ArrayList<Integer>> node_routes;        // 车辆路径
    ArrayList<ArrayList<Double>> node_servetimes;    // 顾客的开始服务时间

    // 新加入的元素
    IloNumVarArray                  varSet;
    protected IloRange[]        cuts;

    public Node(Data data) {
        super();
        this.data = data;
        node_cost = data.big_M;
        lp_x = new double [data.vertexNum][data.vertexNum][data.vehicleNum];
        node_x_map = new int[data.vertexNum][data.vertexNum][data.vehicleNum];
        node_x = new int[data.vertexNum][data.vertexNum];
        node_routes = new ArrayList<ArrayList<Integer>>();
        node_servetimes = new ArrayList<ArrayList<Double>>();
    }


    /**
     * node 的深度拷贝函数
     *
     * 这里使用了 clone 方法
     * Computer c=new Computer("dell", "4G 内存");
     * Computer c1=c.Clone();
     * 在这两句代码中有两个 Computer 类型的对象 c 和 c1，其中 c1 就是通过 Clone 方法复制的 c，
     * 我们可以使用 System.out.println() 方法将两个对象的内存地址打印出来，会发现是两个不同的值。
     *
```

```
47          * @return new_node
48          */
49         public Node node_copy() {
50             Node new_node = new Node(data);
51             new_node.d = d;
52             new_node.node_cost = node_cost;
53             for (int i = 0; i < lp_x.length; i++) {
54                 for (int j = 0; j < lp_x[i].length; j++) {
55                     new_node.lp_x[i][j] = lp_x[i][j].clone();
56                 }
57             }
58             for (int i = 0; i < node_x.length; i++) {
59                 new_node.node_x[i] = node_x[i].clone();
60             }
61             for (int i = 0; i < node_x_map.length; i++) {
62                 for (int j = 0; j < node_x_map[i].length; j++) {
63                     new_node.node_x_map[i][j] = node_x_map[i][j].clone();
64                 }
65             }
66             for (int i = 0; i < node_routes.size(); i++) {
67                 new_node.node_routes.add((ArrayList<Integer>) node_routes.get(i).clone());
68             }
69             for (int i = 0; i < node_servetimes.size(); i++) {
70                 new_node.node_servetimes.add((ArrayList<Double>) node_servetimes.get(i).clone());
71             }
72             return new_node;
73         }
74 
75 
76         /**
77          * compareTo:
78          *     比较两个节点实例的大小的函数。如果对象 o 更小，则返回 1
79          *  用于优先队列，是为了找到下一个要处理的节点。
80          */
81         public int compareTo(Object o){
82             Node node = (Node) o;
83             if(node_cost < node.node_cost)
84                 return -1;
85             else if(node_cost == node.node_cost)
86                 return 0;
87             else
88                 return 1;
89         }
90 
91         /**
92          * 统计变量值与变量数量 (这个函数主要是为了输出求解结果时使用，不写这个函数也是可以的)
93          */
94         static class IloNumVarArray {
95             int _num = 0;   // // _num 标识目前数组中有多少个决策变量
96             IloNumVar[] _array = new IloNumVar[32];
97 
98             // 数组不够就增加成两倍长度
99             void add(IloNumVar ivar) {
100                 if (_num >= _array.length) {
101                     IloNumVar[] array = new IloNumVar[2 * _array.length];
102                     System.arraycopy(_array, 0, array, 0, _num);
103                     _array = array;
104                 }
105                 _array[_num++] = ivar;
```

```
106         }
107
108         IloNumVar getElement(int i) {
109             return _array[i];
110         }
111
112         IloNumVar getVar(int i) {
113             return _array[i];
114         }
115
116         int getSize() {
117             return _num;   // 获得目前变量数组中有多少个决策变量
118         }
119     }
120
121 }
```

11.7.2 Check 类

Check.java

```
1  package BranchAndCut_addCut;
2
3  import java.util.ArrayList;
4  import ilog.concert.IloException;
5
6  /**
7   * Check 类功能：解的可行性判断（可直接跳过此类）
8    */
9  class Check{
10     double epsilon = 0.0001;
11     Data data = new Data();
12     ArrayList<ArrayList<Integer>> routes = new ArrayList<>();
13     ArrayList<ArrayList<Double>> servetimes = new ArrayList<>();
14
15     /**
16      * 构造函数
17      * @param lp: BaB_Vrptw 的实例
18      */
19     public Check(BranchAndCut_addCut lp) {
20         super();
21         this.data = lp.data;
22         this.routes = lp.routes;
23         this.servetimes = lp.servetimes;
24     }
25
26     /**
27      * double_compare
28      *     函数功能：比较两个数的大小
29      */
30     public int double_compare(double v1,double v2) {
31         if (v1 < v2 - epsilon) {
32             return -1;
33         }
34         if (v1 > v2 + epsilon) {
35             return 1;
36         }
37         return 0;
38     }
```

```
39
40      /**
41       * 函数功能：解的可行性判断
42       * @throws IloException
43       */
44      public void fesible() throws IloException {
45          //车辆数量可行性判断
46          if (routes.size() > data.vehicleNum) {
47              System.out.println("error: vecnum!!!");
48              System.exit(0);
49          }
50          //车辆载荷可行性判断
51          for (int k = 0; k < routes.size(); k++) {
52              ArrayList<Integer> route = routes.get(k);
53              double capacity = 0;
54              for (int i = 0; i < route.size(); i++) {
55                  capacity += data.demands[route.get(i)];
56              }
57              if (capacity > data.cap) {
58                  System.out.println("error: cap!!!");
59                  System.exit(0);
60              }
61          }
62          //时间窗、车容量可行性判断
63          for (int k = 0; k < routes.size(); k++) {
64              ArrayList<Integer> route = routes.get(k);
65              ArrayList<Double> servertime = servetimes.get(k);
66              double capacity = 0;
67              for (int i = 0; i < route.size()-1; i++) {
68                  int origin = route.get(i);
69                  int destination = route.get(i+1);
70                  double si = servertime.get(i);
71                  double sj = servertime.get(i+1);
72                  if (si < data.a[origin] && si >  data.b[origin]) {
73                      System.out.println("error: servertime!");
74                      System.exit(0);
75                  }
76          if (double_compare(si + data.dist[origin][destination],data.b[destination]) > 0) {
77                      System.out.println(origin + ": [" + data.a[origin]
78                              + ","+data.b[origin]+"]"+ " "+ si);
79                      System.out.println(destination + ": [" +
80                              data.a[destination] + ","+data.b[destination]+"]"+ " "+ sj);
81                      System.out.println(data.dist[origin][destination]);
82                      System.out.println(destination + ":" );
83                      System.out.println("error: forward servertime!");
84                      System.exit(0);
85                  }
86              if (double_compare(sj - data.dist[origin][destination],data.a[origin]) < 0) {
87                      System.out.println(origin + ": [" + data.a[origin]
88                              + ","+data.b[origin]+"]"+ " "+ si);
89                      System.out.println(destination + ": [" + data.a[destination]
90                              + ","+data.b[destination]+"]"+ " "+ sj);
91                      System.out.println(data.dist[origin][destination]);
92                      System.out.println(destination + ":" );
93                      System.out.println("error: backward servertime!");
94                      System.exit(0);
95                  }
96              }
97              if (capacity > data.cap) {
```

```
98                  System.out.println("error: capacity!!!");
99                  System.exit(0);
100             }
101         }
102     }
103 }
```

11.7.3 MyRandom 类

MyRandom.java

```
1  package BranchAndCut_addCut;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Random;
6
7  public class MyRandom {
8
9      /**
10      * 根据 min 和 max 随机生成一个范围在 [min,max] 的随机数，包括 min 和 max
11      * @param min
12      * @param max
13      * @return int
14      */
15     public static int getRandom(int min, int max){
16         Random random = new Random();
17         return random.nextInt( max - min + 1 ) + min;
18     }
19
20     /**
21      * 根据 min 和 max 随机生成 count 个不重复的随机数组
22      * @param min
23      * @param max
24      * @param count
25      * @return int[]
26      */
27     public static int[] getRandoms(int min, int max, int count){
28         int[] randoms = new int[count];
29         List<Integer> listRandom = new ArrayList<Integer>();
30
31         if( count > ( max - min + 1 )){
32             return null;
33         }
34         // 将所有的可能出现的数字放进候选 list
35         for(int i = min; i <= max; i++){
36             listRandom.add(i);
37         }
38         // 从候选 list 中取出放入数组，已经被选中的就从这个 list 中移除
39         for(int i = 0; i < count; i++){
40             int index = getRandom(0, listRandom.size()-1);
41             randoms[i] = listRandom.get(index);
42             listRandom.remove(index);
43         }
44
45         return randoms;
46     }
47
48
```

```
49      public static ArrayList<ArrayList<Integer>> getRandomsList(int min, int max, int count){
50          ArrayList<ArrayList<Integer>> info = new ArrayList<ArrayList<Integer>>();
51          int[] randoms = new int[count];
52          ArrayList<Integer> listRandom = new ArrayList<Integer>();
53          ArrayList<Integer> inListRandom = new ArrayList<Integer>();
54
55          if( count > ( max - min + 1 )){
56              return null;
57          }
58          // 将所有的可能出现的数字放进候选 list
59          for(int i = min; i <= max; i++){
60              listRandom.add(i);
61          }
62          // 从候选 list 中取出放入数组，已经被选中的就从这个 list 中移除
63          for(int i = 0; i < count; i++){
64              int index = getRandom(0, listRandom.size()-1);
65              randoms[i] = listRandom.get(index);
66              listRandom.remove(index);
67          }
68
69          for(int i = 0; i < randoms.length; i++) {
70              inListRandom.add(randoms[i]);
71          }
72
73          info.add(inListRandom);
74          info.add(listRandom);
75
76          return info;
77      }
78  }
```

11.7.4 Data 类

Data.java

```
1   package BranchAndCut_addCut;
2
3   import java.io.BufferedReader;
4   import java.io.FileReader;
5   import java.util.Scanner;
6
7   //定义参数
8   class Data{
9       int vertexNum;            //所有点集合 n（包括配送中心和客户点，首尾（0 和 n）为配送中心）
10      double E;                  //配送中心时间窗开始时间
11      double    L;                 //配送中心时间窗结束时间
12      int vehicleNum;            //车辆数
13      double cap;              //车辆载荷
14      int[][] vertexs;         //所有点的坐标 x,y
15      int[] demands;             //需求量
16      int[] vehicles;            //车辆编号
17      double[] a;                //时间窗开始时间【a[i],b[i]】
18      double[] b;                //时间窗结束时间【a[i],b[i]】
19      double[] s;                //客户点的服务时间
20      int[][] arcs;             //arcs[i][j] 表示 i 到 j 点的弧
21      double[][] dist;         //距离矩阵，满足三角关系，暂用距离表示花费 C[i][j]=dist[i][j]
22      double gap= 1e-6;
23      double big_M = 100000;
24      int cutNum;                    // 每一次最多添加的 cutting plane 的数量
```

```
25
26      //截断小数 3.26434-->3.2
27      public double double_truncate(double v){
28          int iv = (int) v;
29          if(iv+1 - v <= gap)
30              return iv+1;
31          double dv = (v - iv) * 10;
32          int idv = (int) dv;
33          double rv = iv + idv / 10.0;
34          return rv;
35      }
36
37
38      public Data() {
39          super();
40      }
41
42      //函数功能：从 txt 文件中读取数据并初始化参数
43      public void Read_data(String path,Data data,int vertexnum) throws Exception{
44          String line = null;
45          String[] substr = null;
46          Scanner cin = new Scanner(new BufferedReader(new FileReader(path)));  //读取文件
47          for(int i =0; i < 4;i++){
48              line = cin.nextLine();  //读取一行
49          }
50          line = cin.nextLine();
51          line.trim(); //返回调用字符串对象的一个副本，删除起始和结尾的空格
52          substr = line.split(("\\s+")); //以空格为标志将字符串拆分
53          //初始化参数
54          data.vertexNum = vertexnum;
55          data.vehicleNum = Integer.parseInt(substr[1]);
56          data.cap = Integer.parseInt(substr[2]);
57          data.vertexs =new int[data.vertexNum][2];                //所有点的坐标 x,y
58          data.demands = new int[data.vertexNum];                    //需求量
59          data.vehicles = new int[data.vehicleNum];                    //车辆编号
60          data.a = new double[data.vertexNum];                        //时间窗开始时间
61          data.b = new double[data.vertexNum];                        //时间窗结束时间
62          data.s = new double[data.vertexNum];                        //服务时间
63          data.arcs = new int[data.vertexNum][data.vertexNum];
64          //距离矩阵，满足三角关系，用距离表示 cost
65          data.dist = new double[data.vertexNum][data.vertexNum];
66          for(int i =0; i < 4;i++){
67              line = cin.nextLine();
68          }
69          //读取 vetexnum-1 行数据
70          for (int i = 0; i < data.vertexNum - 1; i++) {
71              line = cin.nextLine();
72              line.trim();
73              substr = line.split("\\s+");
74              data.vertexs[i][0] = Integer.parseInt(substr[2]);
75              data.vertexs[i][1] = Integer.parseInt(substr[3]);
76              data.demands[i] = Integer.parseInt(substr[4]);
77              data.a[i] = Integer.parseInt(substr[5]);
78              data.b[i] = Integer.parseInt(substr[6]);
79              data.s[i] = Integer.parseInt(substr[7]);
80          }
81          cin.close();//关闭流
82          //初始化配送中心参数
83          data.vertexs[data.vertexNum-1] = data.vertexs[0];
```

```
84          data.demands[data.vertexNum-1] = 0;
85          data.a[data.vertexNum-1] = data.a[0];
86          data.b[data.vertexNum-1] = data.b[0];
87          data.E = data.a[0];
88          data.L = data.b[0];
89          data.s[data.vertexNum-1] = 0;
90          double min1 = 1e15;
91          double min2 = 1e15;
92          //距离矩阵初始化
93          for (int i = 0; i < data.vertexNum; i++) {
94              for (int j = 0; j < data.vertexNum; j++) {
95                  if (i == j) {
96                      data.dist[i][j] = 0;
97                      continue;
98                  }
99                  data.dist[i][j] =
100                     Math.sqrt((data.vertexs[i][0]-data.vertexs[j][0])
101                             *(data.vertexs[i][0]-data.vertexs[j][0])+
102                     (data.vertexs[i][1]-data.vertexs[j][1])
103                     *(data.vertexs[i][1]-data.vertexs[j][1]));
104                 data.dist[i][j]=data.double_truncate(data.dist[i][j]);
105             }
106         }
107         data.dist[0][data.vertexNum-1] = 0;
108         data.dist[data.vertexNum-1][0] = 0;
109         //距离矩阵满足三角关系
110         for (int  k = 0; k < data.vertexNum; k++) {
111             for (int i = 0; i < data.vertexNum; i++) {
112                 for (int j = 0; j < data.vertexNum; j++) {
113                     if (data.dist[i][j] > data.dist[i][k] + data.dist[k][j]) {
114                         data.dist[i][j] = data.dist[i][k] + data.dist[k][j];
115                     }
116                 }
117             }
118         }
119         //初始化为完全图
120         for (int i = 0; i < data.vertexNum; i++) {
121             for (int j = 0; j < data.vertexNum; j++) {
122                 if (i != j) {
123                     data.arcs[i][j] = 1;
124                 }
125                 else {
126                     data.arcs[i][j] = 0;
127                 }
128             }
129         }
130         /**
131          *
132          * 预处理，除去不符合时间窗和容量约束的边
133          */
134         //除去不符合时间窗和容量约束的边
135         for (int i = 0; i < data.vertexNum; i++) {
136             for (int j = 0; j < data.vertexNum; j++) {
137                 if (i == j) {
138                     continue;
139                 }
140                 if (data.a[i]+data.s[i]+data.dist[i][j]>data.b[j] ||
141                         data.demands[i]+data.demands[j]>data.cap) {
142                     data.arcs[i][j] = 0;
```

```
143                 }
144                 if (data.a[0]+data.s[i]+data.dist[0][i]+data.dist[i][data.vertexNum-1]>
145                 data.b[data.vertexNum-1]) {
146                     System.out.println("the calculating example is false");
147 
148                 }
149             }
150         }
151         for (int i = 1; i < data.vertexNum-1; i++) {
152             if (data.b[i] - data.dist[0][i] < min1) {
153                 min1 = data.b[i] - data.dist[0][i];
154             }
155             if (data.a[i] + data.s[i] + data.dist[i][data.vertexNum-1] < min2) {
156                 min2 = data.a[i] + data.s[i] + data.dist[i][data.vertexNum-1];
157             }
158         }
159         if (data.E > min1 || data.L < min2) {
160             System.out.println("Duration false!");
161             System.exit(0);//终止程序
162         }
163 
164         //初始化配送中心 0, n+1 两点的参数
165         data.arcs[data.vertexNum-1][0] = 0;
166         data.arcs[0][data.vertexNum-1] = 1;
167         for (int i = 1; i < data.vertexNum-1; i++) {
168             data.arcs[data.vertexNum-1][i] = 0;
169         }
170         for (int i = 1; i < data.vertexNum-1; i++) {
171             data.arcs[i][0] = 0;
172         }
173     }
174 
175     public static void printData(Data data){
176         System.out.println("vehicleNum" + "\t\t: " + data.vehicleNum);
177         System.out.println("vehicleCapacity" + "\t\t: " + data.cap);
178         for(int i = 0; i < data.vertexNum - 1; i++){  //这里用了增强 for
179             System.out.print(i + 1 + "\t");
180             System.out.print(data.vertexs[i][0] + "\t");
181             System.out.print(data.vertexs[i][1] + "\t");
182             System.out.print(data.demands[i] + "\t");
183             System.out.print(data.a[i] + "\t");
184             System.out.print(data.b[i] + "\t");
185             System.out.print(data.s[i] + "\n");
186         }
187         for(int i = 0; i < data.vertexNum - 1; i++) {
188             for(int j = 0; j < data.vertexNum - 1; j++) {
189                 System.out.print(data.dist[i][j] + "\t");
190             }
191             System.out.println();
192         }
193 
194     }
195 }
```

11.7.5 BranchAndCut_addCut 类

BranchAndCut_addCut.java

```java
package BranchAndCut_addCut;

import java.util.ArrayList;
import java.util.PriorityQueue;
import ilog.concert.*;
import ilog.cplex.*;

/**
 * @reference: 本代码的 Branch and Bound 部分来自于"数据魔术师"公众号 (Branch and Bound 代码原作者为黄楠博士，毕业于华中科技大学)
 * @reviseAndExtension: 刘兴禄
 * @Institute: 清华大学
 * @ 操作说明：读入不同的文件前要手动修改 vetexNum 参数，参数值为所有点个数，包括配送中心 0 和 n+1，代码算例截取于 Solomon 测试算例
 * @date: 2018.9.5
 *
 */
// 类功能：建立模型并求解
public class BranchAndCut_addCut {
    static double gap = 1e-6;
    Data             data;         // 定义类 Data 的对象
    Node             node1;        // 分支左节点
    Node             node2;       // 分支右节点
    int              deep;         // 深度
    Node             best_node;     // 当前最好节点
    double             cur_best;      // 当前最好解
    int[]             record_arc; // 记录需要分支的变量下标 x[i][j][k] 的 [i][j][k]
    double             x_gap;         // 计算精度容差
    IloCplex         model;          // 模型实例
    double             cost;          // 目标值
    double[][][]     x_map;         // 记录解 x[i][j][k]
    public IloNumVar[][][]     x;  // x[i][j][k] 表示弧 arcs[i][j] 被车辆 k 访问
    public IloNumVar[][]     w;  // 车辆访问所有点的时间矩阵 ** 其实就是 s_ik，就是第 i 个点被第 k 辆车访问的时间
    public PriorityQueue<Node>         queue; // 分支队列
    ArrayList<ArrayList<Integer>>     routes; // 车辆路径
    ArrayList<ArrayList<Double>>     servetimes; // 顾客的开始服务时间

    // 新加的东西
    IloNumVarArray varSet = new IloNumVarArray();


    /**
     *  BranchAndCut_addCut 的构造方法
     * @param data
     */
    public BranchAndCut_addCut(Data data) {
        this.data = data;
        x_gap = data.gap;
        routes = new ArrayList<>();     // 定义车辆路径链表
        servetimes = new ArrayList<>(); // 定义花费时间链表
        // 初始化车辆路径和花费时间链表，链表长度为车辆数 k
        for (int k = 0; k < data.vehicleNum; k++) {
            ArrayList<Integer> r = new ArrayList<>();
            ArrayList<Double> t = new ArrayList<>();
            routes.add(r);
            servetimes.add(t);
```

```
55          }
56          x_map = new double[data.vertexNum][data.vertexNum][data.vehicleNum];
57      }
58
59      /**
60       *
61       * 将 lp 中的数据清除
62       */
63      public void clear_lp() {
64          data = null;
65          routes.clear();
66          servetimes.clear();
67          x_map = null;
68      }
69
70      /**
71       * 将 lp 的解拷贝到 node
72       *
73       * @param lp
74       * @param node
75       */
76      public void copy_lp_to_node(BranchAndCut_addCut lp, Node node) {
77          // 首先清除 node 里面的数据
78          node.node_routes.clear();
79          node.node_servetimes.clear();
80
81          // 然后把 lp 里面的数据 copy 到 node 里面
82          node.node_cost = lp.cost;
83          for (int i = 0; i < lp.x_map.length; i++) {
84              for (int j = 0; j < lp.x_map[i].length; j++) {
85                  node.lp_x[i][j] = lp.x_map[i][j].clone();
86              }
87          }
88          for (int i = 0; i < lp.routes.size(); i++) {
89              node.node_routes.add((ArrayList<Integer>) lp.routes.get(i).clone());
90          }
91          for (int i = 0; i < lp.servetimes.size(); i++) {
92              node.node_servetimes.add((ArrayList<Double>) lp.servetimes.get(i)
93                      .clone());
94          }
95      }
96
97      /**
98       * 函数功能：建立 VRPTW 的 cplex 模型
99       *
100      * 这里我们将 VRPTW 标准模型中的整数约束松弛掉，建立 VRPTW 的线性松弛
101      *
102      * @throws IloException
103      */
104     private void build_model() throws IloException {
105         // creat model
106         model = new IloCplex();
107         // model.setOut(null);    // 关闭 CPLEX 的 log 信息
108         // model.setParam(IloCplex.DoubleParam.EpOpt, 1e-9);  // 设置 tolerance
109         // model.setParam(IloCplex.DoubleParam.EpGap, 1e-9);  // 设置 tolerance
110
111         // creat decision variables
112         x = new IloNumVar[data.vertexNum][data.vertexNum][data.vehicleNum];
113         w = new IloNumVar[data.vertexNum][data.vehicleNum]; // 车辆访问点的时间
```

```
114             // 创建决策变量 x 和 w，设置其类型及取值范围
115             for (int i = 0; i < data.vertexNum; i++) {
116                 for (int k = 0; k < data.vehicleNum; k++) {
117                     // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
118                     // w[i][k] = model.numVar(data.a[i], data.b[i], IloNumVarType.Float, "w" + i
119                     //         + "," + k);
120                     System.out.println(data.a[i] + " ---- " + data.b[i]);
121                     w[i][k] = model.numVar(0, 1e15, IloNumVarType.Float, "w" + i
122                             + "," + k);
123                 }
124                 for (int j = 0; j < data.vertexNum; j++) {
125                     if (data.arcs[i][j] == 0) {
126                         // 如果 i=j，则该条狐不通
127                         x[i][j] = null;
128                     } else {
129                         // x_ijk
130                         for (int k = 0; k < data.vehicleNum; k++) {
131                             // 注意，这里由于要建立 lp 松弛问题，因此都是 numVar 类型的
132                             x[i][j][k] = model.numVar(0, 1, IloNumVarType.Float,
133                                     "x" + i + "," + j + "," + k);
134                         }
135                     }
136                 }
137             }
138             // 加入目标函数
139             // 表达式 (7.1.1)
140             IloNumExpr obj = model.numExpr();
141             for (int i = 0; i < data.vertexNum; i++) {
142                 for (int j = 0; j < data.vertexNum; j++) {
143                     if (data.arcs[i][j] == 0) {
144                         System.out.println("delected arc : " + i + ", " + j);
145                         continue;
146                     }
147                     for (int k = 0; k < data.vehicleNum; k++) {
148                         obj = model.sum(obj,
149                                 model.prod(data.dist[i][j], x[i][j][k]));
150                     }
151                 }
152             }
153             model.addMinimize(obj);
154             // 加入约束 1
155             // 表达式 (7.1.2)
156             for (int i = 1; i < data.vertexNum - 1; i++) {
157                 IloNumExpr expr1 = model.numExpr();
158                 for (int k = 0; k < data.vehicleNum; k++) {
159                     for (int j = 1; j < data.vertexNum; j++) {
160                         if (data.arcs[i][j] == 1) {
161                             expr1 = model.sum(expr1, x[i][j][k]);
162                         }
163                     }
164                 }
165                 model.addEq(expr1, 1);
166             }
167             // 加入约束 2
168             // 表达式 (7.1.3)
169             for (int k = 0; k < data.vehicleNum; k++) {
170                 IloNumExpr expr2 = model.numExpr();
171                 for (int j = 1; j < data.vertexNum; j++) {
172                     if (data.arcs[0][j] == 1) {
```

```
                    expr2 = model.sum(expr2, x[0][j][k]);
                }
            }
            model.addEq(expr2, 1);
        }
        // 加入约束 3
        // 表达式 (7.1.4)
        for (int k = 0; k < data.vehicleNum; k++) {
            for (int j = 1; j < data.vertexNum - 1; j++) {
                IloNumExpr expr3 = model.numExpr();
                IloNumExpr subExpr1 = model.numExpr();
                IloNumExpr subExpr2 = model.numExpr();
                for (int i = 0; i < data.vertexNum; i++) {
                    if (data.arcs[i][j] == 1) {
                        subExpr1 = model.sum(subExpr1, x[i][j][k]);
                    }
                    if (data.arcs[j][i] == 1) {
                        subExpr2 = model.sum(subExpr2, x[j][i][k]);
                    }
                }
                expr3 = model.sum(subExpr1, model.prod(-1, subExpr2));
                model.addEq(expr3, 0);
            }
        }
        // 加入约束 4
        // 表达式 (7.1.5)
        for (int k = 0; k < data.vehicleNum; k++) {
            IloNumExpr expr4 = model.numExpr();
            for (int i = 0; i < data.vertexNum - 1; i++) {
                if (data.arcs[i][data.vertexNum - 1] == 1) {
                    expr4 = model.sum(expr4, x[i][data.vertexNum - 1][k]);
                }
            }
            model.addEq(expr4, 1);
        }
        // 加入约束 5
        // 表达式 (7.1.6)
        for (int k = 0; k < data.vehicleNum; k++) {
            IloNumExpr expr8 = model.numExpr();
            for (int i = 1; i < data.vertexNum - 1; i++) {
                IloNumExpr expr9 = model.numExpr();
                for (int j = 0; j < data.vertexNum; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr9 = model.sum(expr9, x[i][j][k]);
                    }
                }
                expr8 = model.sum(expr8, model.prod(data.demands[i], expr9));
            }
            model.addLe(expr8, data.cap);
            model.exportModel("VRPTW_LP.lp");
        }
        // 加入约束 6
        // 表达式 (7.1.7)
        double M = 1e5;
        for (int k = 0; k < data.vehicleNum; k++) {
            for (int i = 0; i < data.vertexNum; i++) {
                for (int j = 0; j < data.vertexNum; j++) {
                    if (data.arcs[i][j] == 1) {
                        IloNumExpr expr5 = model.numExpr();
```

```
                        IloNumExpr expr6 = model.numExpr();
                        expr5 = model.sum(w[i][k], data.s[i] + data.dist[i][j]);
                        expr5 = model.sum(expr5, model.prod(-1, w[j][k]));
                        expr6 = model.prod(M,
                                model.sum(1, model.prod(-1, x[i][j][k])));
                        model.addLe(expr5, expr6);
                    }
                }
            }
        }
        // 加入约束 7
        // 时间窗约束
        for (int k = 0; k < data.vehicleNum; k++) {
            for (int i = 1; i < data.vertexNum - 1; i++) {
                IloNumExpr expr7 = model.numExpr();
                for (int j = 0; j < data.vertexNum; j++) {
                    if (data.arcs[i][j] == 1) {
                        expr7 = model.sum(expr7, x[i][j][k]);
                    }
                }
                model.addLe(model.prod(data.a[i], expr7), w[i][k]);
                model.addLe(w[i][k], model.prod(data.b[i], expr7));
            }
        }
        // 加入约束 7
        // 表达式 (7.1.8)
        for (int k = 0; k < data.vehicleNum; k++) {
            model.addLe(data.E, w[0][k]);
            model.addLe(data.E, w[data.vertexNum - 1][k]);
            model.addLe(w[0][k], data.L);
            model.addLe(w[data.vertexNum - 1][k], data.L);
        }

        //model.exportModel("lp.lp");
    }

    /**
     * 函数功能：解模型，并生成车辆路径和得到目标值
     * 获取 VRPTW 模型的线性松弛的解，在根节点或者在分支节点处被调用
     *
     * @throws IloException
     */
    public void get_value() throws IloException {
        routes.clear();
        servetimes.clear();
        cost = 0;
        // 初始化车辆路径和花费时间链表（空链表），链表长度为车辆数 k
        for (int k = 0; k < data.vehicleNum; k++) {
            ArrayList<Integer> r = new ArrayList<>();
            ArrayList<Double> t = new ArrayList<>();
            routes.add(r);
            servetimes.add(t);
        }
        // x_map[i][j][k]，其实就相当于 x[i][j][k]，是为了记录解
        for (int i = 0; i < data.vertexNum; i++) {
            for (int j = 0; j < data.vertexNum; j++) {
                for (int k = 0; k < data.vehicleNum; k++) {
                    x_map[i][j][k] = 0.0; //首先将 x[i][j][k] 初始化为 0
                }
```

```
291                 if (data.arcs[i][j] > 0.5) { // data.arcs[i][j] 取值为 0 或者 1，这是判断 i 和 j 是都连接，是否是一条
                    ↪ 弧
292                     for (int k = 0; k < data.vehicleNum; k++) {
293                         x_map[i][j][k] = model.getValue(x[i][j][k]); // 将 x[i][j][k] 拷贝到 x_map[i][j][k] 中去
294                     }
295                 }
296             }
297         }
298         // 模型可解，从解 x[i][j][k] 中循环提取出车辆路径
299         for (int k = 0; k < data.vehicleNum; k++) {
300             boolean terminate = true;
301             int i = 0;
302             routes.get(k).add(0); // 加入 0，拼成初始的 0-...
303             servetimes.get(k).add(0.0);
304             while (terminate) {
305                 for (int j = 0; j < data.vertexNum; j++) {
306                     if (doubleCompare(x_map[i][j][k], 0) == 1) {  // 如果 x_map[i][j][k] > 0
307                         routes.get(k).add(j);
308                         servetimes.get(k).add(model.getValue(w[j][k]));
309                         i = j;
310                         break;
311                     }
312                 }
313                 if (i == data.vertexNum - 1) {
314                     terminate = false;
315                 }
316             }
317             routes.get(k).set(routes.get(k).size() - 1, 0);
318         }
319         cost = model.getObjValue();
320     }
321 
322     /**
323      *
324      * init() 函数是确定有合法解的最小车辆数，由于直接求解解空间太大，且有很多车辆不能使用
325      * 因此，我们删除无用的车辆，来缩小解空间（这是一个小优化，能够加快程序速度）
326      *
327      * 具体做法就是建立一个松弛了的 cplex 模型，并计算使用的车辆数
328      * 如果有 aa 辆未使用车辆就减少 aa 辆可用车辆，否则减少一辆知道没有可行解
329      * 当然，最后我们可使用的车辆就是最小的车辆了
330      */
331     public BranchAndCut_addCut init(BranchAndCut_addCut lp) throws IloException {
332         lp.build_model();
333         if (lp.model.solve()) {
334             lp.get_value();
335             int aa = 0;
336             for (int i = 0; i < lp.routes.size(); i++) {
337                 if (lp.routes.get(i).size() == 2) { //如果路径是 0-102 这样的，就是从 depot 出发，到 depot 结束的，这
                    ↪ 就要删除这个车
338                     aa++;
339                 }
340             }
341             System.out.println(" 未使用的车辆数是: " + aa);
342 
343             if (aa == 0) {
344                 data.vehicleNum -= 1;      // 如果未使用的是 0
345                 lp.model.clearModel();  // clearModel() 是 IloCplex 类中的一个方法
346                 lp = new BranchAndCut_addCut(data);
347                 return init(lp);
```

```
348                 } else {
349                     data.vehicleNum -= aa; // 如果未使用的车辆数 >0，那么就要将车辆数减去 aa，然后返回原模型
350                     lp.model.clearModel();
351                     lp = new BranchAndCut_addCut(data); // 更新了 data 中的车辆数，因此重新构建模型
352                     return init(lp);   // 递归调用函数
353                 }
354             } else {
355                 data.vehicleNum += 1;  // 如果 lp 问题不可解，就将车辆数增加一个
356                 System.out.println("vehicle number: " + data.vehicleNum);
357                 lp.model.clearModel();
358                 lp = new BranchAndCut_addCut(data);
359                 lp.build_model();
360                 if (lp.model.solve()) {
361                     lp.get_value();
362                     return lp;
363                 } else {
364                     System.out.println("error init");
365                     return null;
366                 }
367             }
368         }
369
370
371
372         /**
373          *  branch and cut 算法的主体流程
374          *
375          * @param lp
376          * @throws IloException
377          */
378         public void branch_and_cut(BranchAndCut_addCut lp) throws IloException {
379             cur_best = 3000;// 预设初始上界
380             deep = 0;
381             record_arc = new int[3];  // 这个是记录可以分支的变量 x[i][j][k]
382             node1 = new Node(data);
383             best_node = null;
384             queue = new PriorityQueue<Node>();
385             // 初始解（非法解）
386             for (int i = 0; i < lp.routes.size(); i++) {
387                 ArrayList<Integer> r = lp.routes.get(i);
388                 System.out.println();
389                 for (int j = 0; j < r.size(); j++) {
390                     System.out.print(r.get(j) + " ");
391                 }
392             }
393             System.out.println("\n\n");
394             lp.copy_lp_to_node(lp, node1);
395
396             node2 = node1.node_copy();
397             deep = 0;
398             node1.d = deep;
399
400             //首先把 node1，也就是初始的 lp 加入到 queue 里面去
401             queue.add(node1);
402             // branch and bound 过程
403             while (!queue.isEmpty()) {
404                 /*
405                  * remove() 和 poll() 方法的语义也完全相同，都是获取并删除队首元素，
406                  * 区别是当方法失败时前者抛出异常，后者返回 null。
```

```
407                 * 由于删除操作会改变队列的结构，为维护小顶堆的性质，需要进行必要的调整。
408                 */
409                // 弹出队首元素
410                Node node = queue.poll();
411
412                /*
413                 * 接下来就是分支定界的过程
414                 * 分支定界的流程是：
415                 * 1. 确定一个下界（初始解 LB），上界 UB 设置为无穷大，或者一个已知的上界。
416                 * 2. 把初始问题构建一个节点加入优先队列（我们使用 best first search，也就是每一次都选择下界最好的节点进行探索最前搜索）。
417                 * 3. 判断队列是否为空，如果为空跳转至 7，否则取出并弹出队首元素，计算该节点的目标值 P。
418                 * 4. 如果 P > UB，返回 3。否则判断当前节点是否是合法解（对于任意 i,j,k,x_ijk 均为整数），如果是，跳转 5 否则跳转 6。
419                 * 5. 如果 P < UB，记录 UB = P，当前节点为当前最优解 BS。返回 3.
420                 * 6. 设置两个子节点 L，R。L，R 的建立方式如上，如果 L 的目标值 L.P <= UB，把 L 加入队列，如果 R 的目标值 R.P <= UB，把 R 加入队列。返回 3.
421                 * 7. 结束，返回记录的最优节点 BS。如果 BS 为空则无解。
422                 */
423                // 某支最优解大于（也就是劣于）当前最好可行解，删除（因为 poll() 方法就是弹出队首元素，并将其删除）
424                if (doubleCompare(node.node_cost, cur_best) > 0) {
425                    continue;
426                } else {
427                    // 找到可以分支的变量 x[i][j][k]
428                    record_arc = lp.find_arc(node.lp_x);
429                    /*
430                    System.out.println("branch variable = "
431                                       + record_arc[0] + "-"
432                                       + record_arc[1] + "-"
433                                       + record_arc[2] );
434                    */
435
436                    // 某支的合法解,0,1 组合的解，当前分支最好解
437                    if (record_arc[0] == -1) {
438                        // 如果比当前最好解 cur_best 好，更新当前解
439                        if (doubleCompare(node.node_cost, cur_best) == -1) {
440                            lp.cur_best = node.node_cost;
441                            System.out.println(node.d + "  cur_best:" + cur_best);
442                            lp.best_node = node;
443                        }
444                        continue;
445                    } else {// 可以分支
446                        node1 = lp.branch_left_arc(lp, node, record_arc);// 左支
447                        node2 = lp.branch_right_arc(lp, node, record_arc);// 右支
448                        if(lp.deep == 0) {
449                            lp.model.exportModel("right_cut.lp");
450                        }
451
452                        if (node1 != null
453                                && doubleCompare(node1.node_cost, cur_best) <= 0) {
454                            // 如果 node1 的成本 <= cur_best，那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
455                            queue.add(node1);
456                        }
457                        if (node2 != null
458                                && doubleCompare(node2.node_cost, cur_best) <= 0) {
459                            // 如果 node1 的成本 <= cur_best，那就添加进来，否则就说明是个劣于当前最好解的解，就将其删除
460                            queue.add(node2);
461                        }
462                    }
```

```
463             }
464         }
465     }
466 
467     // 分支设置
468     /*
469     public void set_bound(Node node) throws IloException {
470         for (int i = 0; i < data.vertexNum; i++) {
471             for (int j = 0; j < data.vertexNum; j++) {
472                 if (data.arcs[i][j] > 0.5) {
473                     if (node.node_x[i][j] == 0) {
474                         // 如果 node_x[i][j] = 0，弧 (i, j) 可以被访问，所以上界为 1，下界为 0
475                         for (int k = 0; k < data.vehicleNum; k++) {
476                             x[i][j][k].setLB(0.0);
477                             x[i][j][k].setUB(1.0);
478                         }
479                     } else if (node.node_x[i][j] == -1) {
480                         // 如果 node_x[i][j] = -1，弧 (i, j) 不能被访问，所以上界为 0，下界为 0
481                         for (int k = 0; k < data.vehicleNum; k++) {
482                             x[i][j][k].setLB(0.0);
483                             x[i][j][k].setUB(0.0);
484                         }
485                     } else {
486                         for (int k = 0; k < data.vehicleNum; k++) {
487                             // 如果 node_x[i][j] = 1，弧 (i, j) 必须被访问，所以上界为 1，下界为 1
488                             if (node.node_x_map[i][j][k] == 1) {
489                                 x[i][j][k].setLB(1.0);
490                                 x[i][j][k].setUB(1.0);
491                             } else {
492                                 x[i][j][k].setLB(0.0);
493                                 x[i][j][k].setUB(0.0);
494                             }
495                         }
496                     }
497                 }
498             }
499         }
500     }
501      */
502     /*
503     public void set_bound1(Node node) throws IloException {
504         for (int i = 0; i < data.vertexNum; i++) {
505             for (int j = 0; j < data.vertexNum; j++) {
506                 if (data.arcs[i][j] > 0.5) {
507                     for (int k = 0; k < data.vehicleNum; k++) {
508                         if (node.node_x_map[i][j][k] == 0) {
509                             x[i][j][k].setLB(0.0);
510                             x[i][j][k].setUB(1.0);
511                         } else if (node.node_x_map[i][j][k] == -1) {
512                             x[i][j][k].setLB(0.0);
513                             x[i][j][k].setUB(0.0);
514                         } else {
515                             x[i][j][k].setLB(1.0);
516                             x[i][j][k].setUB(1.0);
517                         }
518                     }
519                 }
520             }
521         }
```

```
522     }
523     */
524
525     /**
526      *  向左分支
527      *
528      * @param lp
529      * @param father_node
530      * @param record
531      * @return
532      * @throws IloException
533      */
534     public Node branch_left_arc(BranchAndCut_addCut lp, Node father_node, int[] record)
535             throws IloException {
536         if (record[0] == -1) {
537             return null;
538         }
539         Node new_node = new Node(data);
540
541         // 首先将 father_node 拷贝一份成 new_node
542         new_node = father_node.node_copy();
543
544         // node_x[i][j] = 1 表示弧 (i, j) 必须被访问， node_x[i][j] = -1 表示不能访问， node_x[i][j] = 0 表示可以访问
            ↪ 也可以不访问
545         // 由于是左支，因此设置 (i, j) 不能被访问
546         new_node.node_x[record[0]][record[1]] = -1;
547
548         // 由于是左支，因此将 x[i][j][k] 设置成 0
549         for (int k = 0; k < data.vehicleNum; k++) {
550             new_node.node_x_map[record[0]][record[1]][k] = 0;
551         }
552         new_node.node_x_map[record[0]][record[1]][record[2]]=-1;
553         // 设置左支
554         // lp.set_bound(new_node);  如果不加 cut 就是这样的语句
555
556         // ---------------------下面是加 cut 的做
            ↪ 法--------------------------------------------------------------------------------
557         // 左支：需要令所有 x[i][j][k] == 0
558         // 首先初始化长度
559
560         //System.out.println("\n 添加 left 支 cut 之前 lp 中的约束个数" + lp.model.getNrows());
561         // 拷贝父节点的 cuts （这里可以用深度拷贝，结合 getExpr 函数实现，但是暂时没有必要）
562
563         if(father_node.cuts != null) {
564             new_node.cuts = new IloRange[father_node.cuts.length + lp.data.vehicleNum + data.cutNum];
565             for(int i = 0; i < father_node.cuts.length; i++) {
566                 new_node.cuts[i] = father_node.cuts[i];
567                 // 先要把父节点的 cut 加入到 lp.model 中去
568                 //lp.model.addCut(father_node.cuts[i]);
569                 lp.model.addRange(father_node.cuts[i].getLB(), father_node.cuts[i].getExpr(),
                    ↪ father_node.cuts[i].getUB());
570             }
571             // 增加新的 cuts(关于 bound 的 cuts)
572             for(int i = father_node.cuts.length; i < new_node.cuts.length - data.cutNum; i++) {
573                 //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                    ↪ father_node.cuts.length], 0));
574                 new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i - father_node.cuts.length], 0);
575             }
576
```

```
577                 // 增加新的 cuts(2-cys 的 cuts)
578                 IloRange[] k_path_cuts = k_path_cuts(data);
579                 for(int i = new_node.cuts.length - data.cutNum; i < new_node.cuts.length; i++) {
580                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                    ↪  father_node.cuts.length], 0));
581                     new_node.cuts[i] = k_path_cuts[i - (new_node.cuts.length - data.cutNum)];
582                 }
583
584             }else {
585                 // 增加新的 cuts
586                 new_node.cuts = new IloRange[lp.data.vehicleNum + data.cutNum];
587                 for(int i = 0; i < new_node.cuts.length - data.cutNum; i++) {
588                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i], 0));
589                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i], 0);
590                 }
591
592                 // 增加新的 cuts(2-cys 的 cuts)
593                 IloRange[] k_path_cuts = k_path_cuts(data);
594                 for(int i = new_node.cuts.length - data.cutNum; i < new_node.cuts.length; i++) {
595                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                    ↪  father_node.cuts.length], 0));
596                     //System.out.print("i = " + i);
597                     //System.out.println("     i2 = " + (i - (new_node.cuts.length - data.cutNum)));
598                     new_node.cuts[i] = k_path_cuts[i - (new_node.cuts.length - data.cutNum)];
599                 }
600             }
601
602             System.out.println("copy 成功");
603
604             System.out.println(" 添加 cut 结束");
605             System.out.println(" 添加 left 支 cut 之后 lp 中的约束个数" + lp.model.getNrows());
606             // ---------------------cut 添加结
                ↪  束----------------------------------------------------------------------------------------
607
608             if(lp.deep == 0) {
609                 lp.model.exportModel("left_cut.lp");
610             }
611
612             // 根据 new_node 对应的接的解，更新完了 lp 的上下界之后，就继续求解 lp
613             if (lp.model.solve()) {
614                 lp.get_value();
615                 deep++;
616                 new_node.d = deep;
617                 // 将 lp 的解 copy 到 new_node 中，然后返回 new_node
618                 lp.copy_lp_to_node(lp, new_node);
619                 System.out.println(new_node.d + " left" + " " + lp.cost);
620
621                 // 清理 cuts
622                 //lp.model.clearCuts();
623                 for(int i = 0; i < new_node.cuts.length; i++) {
624                     lp.model.remove(new_node.cuts[i]);
625                 }
626                 //System.out.println(new_node.cuts.length);
627
628                 //System.out.println(" 左支分完删除后模型的约束个数" + lp.model.getNrows());
629             } else {
630                 System.out.println(" 分左支求解无解");
631                 for(int i = 0; i < new_node.cuts.length; i++) {
632                     lp.model.remove(new_node.cuts[i]);
```

```
633             }
634             //System.out.println("\n 左支分完删除后模型的约束个数" + lp.model.getNrows());
635             //System.out.println("\n");
636             new_node.node_cost = data.big_M;
637         }
638         return new_node;
639     }
640 
641     /**
642      * 生成 k_path-cuts 的函数
643      *
644      * @param data
645      * @return
646      * @throws IloException
647      */
648     private IloRange[] k_path_cuts(Data data) throws IloException {
649         // 首先新建一个 IloRange 对象
650         IloRange[] cuts = new IloRange[data.cutNum];
651 
652         //循环加割
653         int count = 0;
654         while(count < data.cutNum) {
655             System.out.println(" 第" + count + " 个割");
656             //然后我们随机的生成 9-15 个点
657             int pointSetSize = (int) (9  + Math.round(6 * Math.random()));
658             //int selectedNodes[] = MyRandom.getRandoms(1, vertexnum, pointSetSize);
659             ArrayList<ArrayList<Integer>> selectedNodes
660                                 = MyRandom.getRandomsList(1, data.vertexNum - 2, pointSetSize);
661 
662             //计算 k(s)
663             double totalDemand = 0;
664             for(int i = 0; i < pointSetSize; i++) {
665                 totalDemand += data.demands[selectedNodes.get(0).get(i)];
666             }
667 
668 
669             int K_s = (int)Math.ceil(totalDemand/data.cap);
670             System.out.println("totalDemand = " + totalDemand);
671             System.out.println("K_s = " + K_s);
672 
673             if(K_s <= 1) {
674                 continue;
675             }
676 
677             System.out.println(" 车辆数" + data.vehicleNum);
678             //System.out.println(" 第 1 维" + x.length);
679             //System.out.println(" 第 2 维" + x[0].length);
680             //System.out.println(" 第 3 维" + model.getValue(x[0][0][0]));
681 
682             // 下面生成 cut
683             IloNumExpr expr = model.numExpr();
684             for(int i = 0; i < pointSetSize; i++) {
685                 int index1 = selectedNodes.get(0).get(i);
686                 //循环拼凑 expr
687                 // sum X_ij, i 不在 set 中, j 在 set 中
688                 for(int j = 0; j < selectedNodes.get(1).size(); j++) {
689                     int index2 = selectedNodes.get(1).get(j);
690                     for(int k = 0; k < data.vehicleNum; k++) {
691                         /*
```

```
692                         System.out.print("index 1 = " + index1);
693                         System.out.print("      index 2 = " + index2);
694                         System.out.println("      k = " + k);
695                         */
696
697                         // 这里会出现问题：因为 CPLEX 在预处理的时候，会删除一些变量，导致添加的时候出现空指针异常的情况
698                         // 此时我们可以用 try catch 来写代码
699                         /*
700                         if(x[index1][index2][k] != null) {
701                             expr = model.sum(expr, x[index1][index2][k]);
702                         }*/
703
704                         try{
705                             expr = model.sum(expr, x[index1][index2][k]);
706                         }catch(Exception e) {
707                             //e.printStackTrace();
708                         }
709                         //System.out.println("   " + expr.toString());
710                     }
711
712                 }
713
714             }
715             System.out.println("new cut : " + K_s + " <= "   + expr.toString() + "<=" + Integer.MAX_VALUE);
716             cuts[count] = model.addRange(0, Integer.MAX_VALUE);
717             cuts[count].setLB(K_s);
718             cuts[count].setExpr(expr);
719             count = count + 1;
720             System.out.println("count = " + count);
721             System.out.println("\n\n\n 一个 cut 添加完毕\n\n\n");
722
723         }
724
725         return cuts;
726     }
727
728     /**
729      *  向右分支
730      *
731      * @param lp
732      * @param father_node
733      * @param record
734      * @return
735      * @throws IloException
736      */
737     public Node branch_right_arc(BranchAndCut_addCut lp, Node father_node, int[] record)
738             throws IloException {
739         if (record[0] == -1) {
740             return null;
741         }
742         Node new_node = new Node(data);
743         new_node = father_node.node_copy();
744
745         // 由于是左支，因此设置 (i, j) 必须被访问
746         new_node.node_x[record[0]][record[1]] = 1;
747         new_node.node_x_map[record[0]][record[1]][record[2]]=1;
748
749         // 由于是右支，则设置 x[i][j][k] = 1
750         for (int k = 0; k < data.vehicleNum; k++) {
```

```
751             if (k == record[2]) {
752                 new_node.node_x_map[record[0]][record[1]][k] = 1;
753             } else {
754                 new_node.node_x_map[record[0]][record[1]][k] = 0;
755             }
756         }
757 
758 
759 
760         // 设置右支
761         //lp.set_bound(new_node);
762 
763         //*****************************************************************************
764         if(father_node.cuts != null) {
765             //System.out.println(" 进入 if");
766             new_node.cuts = new IloRange[father_node.cuts.length + lp.data.vehicleNum];
767             for(int i = 0; i < father_node.cuts.length; i++) {
768                 new_node.cuts[i] = father_node.cuts[i];
769                 // 先要把父节点的 cut 加入到 lp.model 中去
770                 //lp.model.addCut(father_node.cuts[i]);
771                 lp.model.addRange(father_node.cuts[i].getLB(), father_node.cuts[i].getExpr(),
                    ↪  father_node.cuts[i].getUB());
772             }
773             // 增加新的 cuts
774             for(int i = father_node.cuts.length; i < new_node.cuts.length; i++) {
775                 if(i - father_node.cuts.length == record[2]) {
776                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][record[2]], 1));
777                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][record[2]], 1);
778                 }else {
779                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i -
                        ↪  father_node.cuts.length], 0));
780                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i - father_node.cuts.length], 0);
781                 }
782 
783             }
784 
785         }else {
786             // 增加新的 cuts
787             //System.out.println(" 进入 else");
788             new_node.cuts = new IloRange[lp.data.vehicleNum];
789 
790             // 增加新的 cuts
791             for(int i = 0; i < new_node.cuts.length; i++) {
792                 //System.out.println(i);
793                 if(i == record[2]) {
794                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][record[2]], 1));
795                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][record[2]], 1);
796                     System.out.println(new_node.cuts[i].toString());
797                 }else {
798                     //new_node.cuts[i] = lp.model.addCut(lp.model.eq(x[record[0]][record[1]][i], 0));
799                     new_node.cuts[i] = lp.model.addEq(x[record[0]][record[1]][i], 0);
800                     //System.out.println(new_node.cuts[i].toString());
801                 }
802 
803             }
804         }
805 
806 
807         System.out.println("copy 成功");
```

```
808
809
810             System.out.println(" 添加 cut 结束");
811             System.out.println("**************");
812             System.out.println(" 添加 right 支 cut 之后 lp 中的约束个数" + lp.model.getNrows());
813             // ---------------------cut 添加结
                ↪ 束----------------------------------------------------------------------------------
814             System.out.println("lp.deep" + lp.deep);
815             if(lp.deep == 0) {
816                 System.out.println(" 导出模型");
817                 lp.model.exportModel("right_cut.lp");
818             }
819             lp.model.solve();
820
821
822             if (lp.model.solve()) {
823                 lp.get_value();
824                 deep++;
825                 new_node.d = deep;
826                 System.out.println(new_node.d + " right: " + lp.cost);
827                 lp.copy_lp_to_node(lp, new_node);
828
829                 // 清理 cuts
830                 //lp.model.clearCuts();
831                 for(int i = 0; i < new_node.cuts.length; i++) {
832                     lp.model.remove(new_node.cuts[i]);
833                 }
834                 //System.out.println(new_node.cuts.length);
835
836                 //System.out.println(" 左支分完删除后模型的约束个数" + lp.model.getNrows());
837
838             } else {
839                 System.out.println(" 分 right 支求解无解");
840                 for(int i = 0; i < new_node.cuts.length; i++) {
841                     lp.model.remove(new_node.cuts[i]);
842                 }
843                 //System.out.println("right 支求解无解删除后模型的约束个数" + lp.model.getNrows());
844                 new_node.node_cost = data.big_M;
845             }
846             //System.out.println("====== 分支完毕------");
847             return new_node;
848         }
849
850
851         /**
852          * 找到需要分支的决策变量 x[i][j][k]
853          *
854          * @param x
855          * @return
856          */
857         public int[] find_arc1(double[][][] x) {
858             int record[] = new int[3];// 记录分支顶点
859             double cur_dif = 0;
860             double min_dif = Double.MAX_VALUE;
861             // 找出最接近 0.5 的弧
862             for (int i = 1; i < data.vertexNum - 1; i++) {
863                 for (int j = 1; j < data.vertexNum - 1; j++) {
864                     if (data.arcs[i][j] > 0.5) {
865                         for (int k = 0; k < data.vehicleNum; k++) {
```

```
866                         // 若该弧值为 0 或 1，则继续
867                         if (is_one_zero(x[i][j][k])) {
868                             continue;
869                         }
870                         cur_dif = get_dif(x[i][j][k]);
871                         if (doubleCompare(cur_dif, min_dif) == -1) {
872                             record[0] = i;
873                             record[1] = j;
874                             record[2] = k;
875                             min_dif = cur_dif;
876                         }
877                     }
878                 }
879             }
880         }
881         // depot
882         if (doubleCompare(min_dif, Double.MAX_VALUE) == 0) {
883             for (int i = 1; i < data.vertexNum - 1; i++) {
884                 if (data.arcs[0][i] > 0.5) {
885                     for (int k = 0; k < data.vehicleNum; k++) {
886                         if (is_fractional(x[0][i][k])) {
887                             cur_dif = get_dif(x[0][i][k]);
888                             if (doubleCompare(cur_dif, min_dif) == -1) {
889                                 record[0] = 0;
890                                 record[1] = i;
891                                 record[2] = k;
892                                 min_dif = cur_dif;
893                             }
894                         }
895                     }
896                 }
897                 if (data.arcs[i][data.vertexNum - 1] > 0.5) {
898                     for (int k = 0; k < data.vehicleNum; k++) {
899                         if (is_fractional(x[i][data.vertexNum - 1][k])) {
900                             cur_dif = get_dif(x[i][data.vertexNum - 1][k]);
901                             if (doubleCompare(cur_dif, min_dif) == -1) {
902                                 record[0] = i;
903                                 record[1] = data.vertexNum - 1;
904                                 record[2] = k;
905                                 min_dif = cur_dif;
906                             }
907                         }
908                     }
909                 }
910             }
911         }
912         if (doubleCompare(min_dif, data.big_M) == 1) {
913             record[0] = -1;
914             record[1] = -1;
915             record[2] = -1;
916         }
917         return record;
918     }
919
920
921     /**
922      * 找到需要分支的决策变量 x[i][j][k]
923      *
924      * @param x
```

```
925         * @return
926         */
927        public int[] find_arc(double[][][] x) {
928            int record[] = new int[3];// 记录分支顶点
929            for (int i = 0; i < data.vertexNum; i++) {
930                for (int j = 0; j < data.vertexNum; j++) {
931                    if (data.arcs[i][j] > 0.5) {
932                        for (int k = 0; k < data.vehicleNum; k++) {
933                            // 若该弧值为 0 或 1，则继续
934                            if (is_one_zero(x[i][j][k])) {
935                                continue;
936                            }
937                            // cur_dif = get_dif(x[i][j][k]);
938                            record[0] = i;
939                            record[1] = j;
940                            record[2] = k;
941                            return record;
942                        }
943                    }
944                }
945            }
946            record[0] = -1;
947            record[1] = -1;
948            record[2] = -1;
949            return record;
950        }
951
952        /**
953         *  比较两个 double 数值的大小
954         * @param a
955         * @param b
956         * @return
957         */
958        public int doubleCompare(double a, double b) {
959            if (a - b > x_gap)
960                return 1;
961            if (b - a > x_gap)
962                return -1;
963            return 0;
964        }
965
966        /**
967         * 判断是否为 0 到 1 之间的小数
968         * @param v
969         * @return
970         */
971        public boolean is_fractional(double v) {
972            if (v > (int) v + x_gap && v < (int) v + 1 - x_gap)
973                return true;
974            else
975                return false;
976        }
977
978        /**
979         * 判断是否为 0 或者 1
980         *
981         * @param temp
982         * @return
983         */
```

```
984     public boolean is_one_zero(double temp) {
985         if (doubleCompare(temp, 0) == 0 || doubleCompare(temp, 1) == 0) {
986             return true;
987         } else {
988             return false;
989         }
990     }
991 
992     /**
993      * 获取到 0.5 的距离
994      *
995      * @param temp
996      * @return
997      */
998     public double get_dif(double temp) {
999         double v = (int) temp + 0.5;
1000        if (v > temp) {
1001            return v - temp;
1002        } else {
1003            return temp - v;
1004        }
1005    }
1006
1007    /**
1008     *  截断小数 3.26434-->3.2
1009     *
1010     * @param v
1011     * @return
1012     */
1013    public static double double_truncate(double v){
1014        int iv = (int) v;
1015        if(iv+1 - v <= 1e-6)
1016            return iv+1;
1017        double dv = (v - iv) * 10;
1018        int idv = (int) dv;
1019        double rv = iv + idv / 10.0;
1020        return rv;
1021    }
1022
1023
1024    /**
1025     * 统计变量值与变量数量 (这个函数主要是为了输出求解结果时使用，不写这个函数也是可以的)
1026     */
1027    static class IloNumVarArray {
1028        int _num = 0;   // // _num 标识目前数组中有多少个决策变量
1029        IloNumVar[] _array = new IloNumVar[32];
1030
1031        // 数组不够就增加成两倍长度
1032        void add(IloNumVar ivar) {
1033            if (_num >= _array.length) {
1034                IloNumVar[] array = new IloNumVar[2 * _array.length];
1035                System.arraycopy(_array, 0, array, 0, _num);
1036                _array = array;
1037            }
1038            _array[_num++] = ivar;
1039        }
1040
1041        IloNumVar getElement(int i) {
1042            return _array[i];
```

```
1043            }
1044
1045            IloNumVar getVar(int i) {
1046                return _array[i];
1047            }
1048
1049            int getSize() {
1050                return _num;   // 获得目前变量数组中有多少个决策变量
1051            }
1052        }
1053
1054    }
```

11.7.6 run_this 类：算法测试

该类用于测试算法。

run_this.java

```
1   package BranchAndCut_addCut;
2
3   import java.util.ArrayList;
4
5   public class run_this {
6       public static void main(String[] args) throws Exception {
7
8           double gap = 1e-6;            // 计算的容差
9           int cutNum = 5;               // 每一次最多添加的割平面数量
10          int vertexNum = 50 + 2;     // 所有点个数，包括 0，n+1 两个配送中心点
11
12          double start = System.currentTimeMillis();
13          Data data = new Data();
14          data.cutNum = cutNum;
15
16          // 读入不同的文件前要手动修改 vetexNum 参数，参数值等于所有点个数，包括配送中心
17          String path = "F:\\MyCode\\JavaCode\\Java_CPLEX_Algorithm\\src\\VRPTW_Branch_and_Cut\\c102.txt";// 算例地址
18          data.Read_data(path, data, vertexNum);
19
20          System.out.println("Read data finished!");
21          System.out.println(" ------ Branch and Cut ------ ");
22          BranchAndCut_addCut lp = new BranchAndCut_addCut(data);
23          double cplex_time1 = System.nanoTime();
24
25          ///*
26          // 删除未用的车辆，缩小解空间
27          lp = lp.init(lp);
28          System.out.println(":   " + lp.data.vehicleNum);
29          //*/
30
31          /*
32          // 尝试不删除未使用的车辆
33          lp.build_model();
34          lp.model.solve();
35          lp.get_value();
36          */
37
38          // branch_and_cut
39          System.out.println("-----进入了 branch_and_cut-----");
40          System.out.println();
```

```
        lp.branch_and_cut(lp);

        // 检验解的合法性
        Check check = new Check(lp);
        check.fesible();
        double cplex_time2 = System.nanoTime();
        double cplex_time = (cplex_time2 - cplex_time1) / 1e9;// 求解时间，单位 s

        // 输出最优解
        System.out.println("cplex_time : " + cplex_time + " \n"
                + "bestcost : " + lp.cur_best);
        if(lp.best_node.node_routes != null) {
            for (int i = 0; i < lp.best_node.node_routes.size(); i++) {
                ArrayList<Integer> r = lp.best_node.node_routes.get(i);
                System.out.println();
                for (int j = 0; j < r.size(); j++) {
                    System.out.print(r.get(j) + " ");
                }
            }
        }


        double end = System.currentTimeMillis();
        System.out.println("\n\n 程序运行时间: " + (end - start) / 1000.0 + " 秒");
    }

}
```

11.7.7 算例格式

Solomon VRPTW Benchmark C101

```
C101

VEHICLE
NUMBER     CAPACITY
  25         200

CUSTOMER
CUST NO.  XCOORD.   YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE   TIME

    0      40         50          0          0       1236          0
    1      45         68         10        912        967         90
    2      45         70         30        825        870         90
    3      42         66         10         65        146         90
    4      42         68         10        727        782         90
    5      42         65         10         15         67         90
    6      40         69         20        621        702         90
    7      40         66         20        170        225         90
    8      38         68         20        255        324         90
    9      38         70         10        534        605         90
   10      35         66         10        357        410         90
   11      35         69         10        448        505         90
   12      25         85         20        652        721         90
   13      22         75         30         30         92         90
   14      22         85         10        567        620         90
   15      20         80         40        384        429         90
   16      20         85         40        475        528         90
   17      18         75         20         99        148         90
```

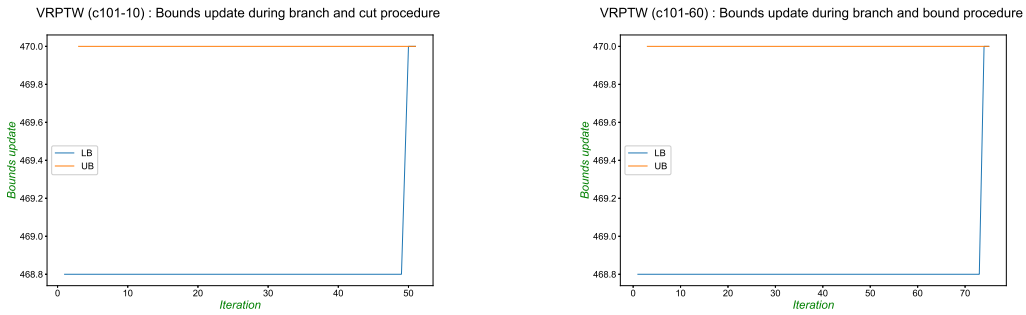
28	18	15	75	20	179	254	90
29	19	15	80	10	278	345	90
30	20	30	50	10	10	73	90
31	21	30	52	20	914	965	90
32	22	28	52	20	812	883	90
33	23	28	55	10	732	777	90
34	24	25	50	10	65	144	90
35	25	25	52	40	169	224	90
36	26	25	55	10	622	701	90
37	27	23	52	10	261	316	90
38	28	23	55	20	546	593	90
39	29	20	50	10	358	405	90
40	30	20	55	10	449	504	90
41	31	10	35	20	200	237	90
42	32	10	40	30	31	100	90
43	33	8	40	40	87	158	90
44	34	8	45	20	751	816	90
45	35	5	35	10	283	344	90
46	36	5	45	10	665	716	90
47	37	2	40	20	383	434	90
48	38	0	40	30	479	522	90
49	39	0	45	20	567	624	90
50	40	35	30	10	264	321	90
51	41	35	32	10	166	235	90
52	42	33	32	20	68	149	90
53	43	33	35	10	16	80	90
54	44	32	30	10	359	412	90
55	45	30	30	10	541	600	90
56	46	30	32	30	448	509	90
57	47	30	35	10	1054	1127	90
58	48	28	30	10	632	693	90
59	49	28	35	10	1001	1066	90
60	50	26	32	10	815	880	90
61	51	25	30	10	725	786	90
62	52	25	35	10	912	969	90
63	53	44	5	20	286	347	90
64	54	42	10	40	186	257	90
65	55	42	15	10	95	158	90
66	56	40	5	30	385	436	90
67	57	40	15	40	35	87	90
68	58	38	5	30	471	534	90
69	59	38	15	10	651	740	90
70	60	35	5	20	562	629	90
71	61	50	30	10	531	610	90
72	62	50	35	20	262	317	90
73	63	50	40	50	171	218	90
74	64	48	30	10	632	693	90
75	65	48	40	10	76	129	90
76	66	47	35	10	826	875	90
77	67	47	40	10	12	77	90
78	68	45	30	10	734	777	90
79	69	45	35	10	916	969	90
80	70	95	30	30	387	456	90
81	71	95	35	20	293	360	90
82	72	53	30	10	450	505	90
83	73	92	30	10	478	551	90
84	74	53	35	50	353	412	90
85	75	45	65	20	997	1068	90
86	76	90	35	10	203	260	90

87	77	88	30	10	574	643	90
88	78	88	35	20	109	170	90
89	79	87	30	10	668	731	90
90	80	85	25	10	769	820	90
91	81	85	35	30	47	124	90
92	82	75	55	20	369	420	90
93	83	72	55	10	265	338	90
94	84	70	58	20	458	523	90
95	85	68	60	30	555	612	90
96	86	66	55	10	173	238	90
97	87	65	55	20	85	144	90
98	88	65	60	30	645	708	90
99	89	63	58	10	737	802	90
100	90	60	55	10	20	84	90
101	91	60	60	10	836	889	90
102	92	67	85	20	368	441	90
103	93	65	85	40	475	518	90
104	94	65	82	10	285	336	90
105	95	62	80	30	196	239	90
106	96	60	80	10	95	156	90
107	97	60	85	30	561	622	90
108	98	58	75	20	30	84	90
109	99	55	80	10	743	820	90
110	100	55	85	20	647	726	90

11.8 Python 调用 Gurobi 实现分支切割算法求解 VRPTW 完整代码

我们也提供了 Python 版本的 Branch and Cut 代码，包括手动实现 Branch and bound 以及 Cutting plane 的版本和使用 callback 添加 Cutting plane 的版本。同样，我们选取取值最接近 0.5 的决策变量 x 进行分支，在每个子节点处，最多添加 5 条 k-path Cuts。

我们取 C101 中前 60 个客户点作为测试算例，设置车辆数为 8，对比 Branch and Bound 和 Branch and Cut 的求解效果。如下图所示。Branch and Cut 的迭代次数较少，探索的节点数也更少。



(a) Branch and Cut (b) Branch and Bound

图 11.1: *Branch and Cut 和 Branch and Bound 的 UB 和 LB 更新比较 (C101-60)*

在 callback 版本的代码中，首先要注意，一定要将参数`PreCrush`设置为 1，语法为

model.Params.PreCrush = 1。因为在 Gurobi 中，想要添加自己设计的 Cut，必须要设置该参数。构造自己设计的 Cut 之前，首先需要在 callback 函数中判断 where 等于 GRB.Callback.MIPNODE，也就是我们要在 BB tree 的节点处添加 Cut。然后我们判断该节点的线性松弛模型是有最优解的，即用语句 model.cbGet(GRB.Callback.MIPNODE_STATUS) 得到该点的求解状态，当状态为GRB.OPTIMAL 时，我们就可以添加 Cut 了。此时，先通过调用函数 cbGetNodeRel获得当前节点的松弛解，然后根据松弛解和外部变量model._vars 构造 Cut 的左端线性表达式，最后调用函数 cbCut将该 Cut 添加到当前模型中。注意，函数cbCut的参数中没有变量名这个参数。按照上述操作，就完成了使用 callback 构造用户自己设计的 Cut。

下面是 Python 调用 Gurobi 实现分支切割算法求解 VRPTW 完整代码。

下面是 Python 调用 Gurobi 实现分支切割算法求解 VRPTW 完整代码。

11.8.1 Python 调用 Gurobi 实现分支切割算法求解 VRPTW 完整代码 (Branch and bound 过程为手动实现的版本)

BnC VRPTW

```python
#!/usr/bin/env python
# coding: utf-8

#  * author :  Liu Xinglu
#  * Institute: Tsinghua University
#  * Date : 2020-7-11
#  * E-mail : hsinglul@163.com

# # Python_Call_Gurobi_Solve_VRPTW

# # Prepare Data
# _*_coding:utf-8 _*_
from __future__ import print_function
from gurobipy import *
import re
import math
import matplotlib.pyplot as plt
import numpy
import pandas as pd
import networkx as nx
import copy
import random

class Data:
    def __init__(self):
        self.customerNum = 0
        self.nodeNum     = 0
        self.vehicleNum  = 0
        self.capacity    = 0
        self.cor_X       = []
        self.cor_Y       = []
        self.demand      = []
        self.serviceTime = []
        self.readyTime   = []
        self.dueTime     = []
        self.disMatrix   = [[]]   # 读取数据
        self.arcs        = {}
```

```
38
39      # function to read data from .txt files
40      def readData(data, path, customerNum):
41          data.customerNum = customerNum
42          data.nodeNum = customerNum + 2
43          f = open(path, 'r')
44          lines = f.readlines()
45          count = 0
46          # read the info
47          for line in lines:
48              count = count + 1
49              if(count == 5):
50                  line = line[:-1].strip()
51                  str = re.split(r" +", line)
52                  data.vehicleNum = int(str[0])
53                  data.capacity = float(str[1])
54              elif(count >= 10 and count <= 10 + customerNum):
55                  line = line[:-1]
56                  str = re.split(r" +", line)
57                  data.cor_X.append(float(str[2]))
58                  data.cor_Y.append(float(str[3]))
59                  data.demand.append(float(str[4]))
60                  data.readyTime.append(float(str[5]))
61                  data.dueTime.append(float(str[6]))
62                  data.serviceTime.append(float(str[7]))
63
64          data.cor_X.append(data.cor_X[0])
65          data.cor_Y.append(data.cor_Y[0])
66          data.demand.append(data.demand[0])
67          data.readyTime.append(data.readyTime[0])
68          data.dueTime.append(data.dueTime[0])
69          data.serviceTime.append(data.serviceTime[0])
70
71
72          # compute the distance matrix
73          data.disMatrix = [([0] * data.nodeNum) for p in range(data.nodeNum)] # 初始化距离矩阵的维度，防止浅拷贝
74          # data.disMatrix = [[0] * nodeNum] * nodeNum]; 这个是浅拷贝，容易重复
75          for i in range(0, data.nodeNum):
76              for j in range(0, data.nodeNum):
77                  temp = (data.cor_X[i] - data.cor_X[j])**2 + (data.cor_Y[i] - data.cor_Y[j])**2
78                  data.disMatrix[i][j] = round(math.sqrt(temp), 1)
79                  # print("%6.2f" % (math.sqrt(temp)), end = " ");
80                  temp = 0
81
82          # initialize the arc
83          for i in range(data.nodeNum):
84              for j in range(data.nodeNum):
85                  if(i == j):
86                      data.arcs[i,j] = 0
87                  else:
88                      data.arcs[i,j] = 1
89          return data
90
91      def preprocess(data):
92          # preprocessing for ARCS
93          # 除去不符合时间窗和容量约束的边
94          for i in range(data.nodeNum):
95              for j in range(data.nodeNum):
96                  if(i == j):
```

```
                    data.arcs[i,j] = 0
                elif(data.readyTime[i] + data.serviceTime[i] + data.disMatrix[i][j] > data.dueTime[j]
                  or data.demand[i] + data.demand[j] > data.capacity):
                    data.arcs[i,j] = 0
                elif(data.readyTime[0] + data.serviceTime[i] + data.disMatrix[0][i] +
                ↪  data.disMatrix[i][data.nodeNum-1] > data.dueTime[data.nodeNum-1]):
                    print("the calculating example is false")
                else:
                    data.arcs[i,j] = 1
        for i in range(data.nodeNum):
            data.arcs[data.nodeNum - 1, i] = 0
            data.arcs[i, 0] = 0
        return data


    def printData(data, customerNum):
        print(" 下面打印数据\n")
        print("vehicle number = %4d" % data.vehicleNum)
        print("vehicle capacity = %4d" % data.capacity)
        for i in range(len(data.demand)):
            print('{0}\t{1}\t{2}\t{3}'.format(data.demand[i], data.readyTime[i],data.dueTime[i],
            ↪  data.serviceTime[i]))

        print("-------距离矩阵-------\n")
        for i in range(data.nodeNum):
            for j in range(data.nodeNum):
                #print("%d   %d" % (i, j));
                print("%6.2f" % (data.disMatrix[i][j]), end = " ")
            print()


# # Read Data
data = Data()
path = 'r101.txt'
customerNum = 30
data = Data.readData(data, path, customerNum)
data.vehicleNum = 8
Data.printData(data, customerNum)
# data = Data.preprocess(data)

# # Build Graph
# 构建有向图对象
Graph = nx.DiGraph()
cnt = 0
pos_location = {}
nodes_col = {}
nodeList = []
for i in range(data.nodeNum):
    X_coor = data.cor_X[i]
    Y_coor = data.cor_Y[i]
    name = str(i)
    nodeList.append(name)
    nodes_col[name] = 'gray'
    node_type = 'customer'
    if(i == 0):
        node_type = 'depot'
    Graph.add_node(name
                , ID = i
                , node_type = node_type
```

```
                , time_window = (data.readyTime[i], data.dueTime[i])
                , arrive_time = 10000   # 这个是时间标签 1
                , demand = data.demand
                , serviceTime = data.serviceTime
                , x_coor = X_coor
                , y_coor = Y_coor
                , min_dis = 0           # 这个是距离标签 2
                , previous_node = None  # 这个是前序结点标签 3
                )

    pos_location[name] = (X_coor, Y_coor)
# add edges into the graph
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        if(i == j or (i == 0 and j == data.nodeNum - 1) or (j == 0 and i == data.nodeNum - 1)):
            pass
        else:
            Graph.add_edge(str(i), str(j)
                           , travelTime = data.disMatrix[i][j]
                           , length = data.disMatrix[i][j]
                          )

nodes_col['0'] = 'red'
nodes_col[str(data.nodeNum-1)] = 'red'
plt.rcParams['figure.figsize'] = (10, 10) # 单位是 inches
nx.draw(Graph
        , pos=pos_location
        , with_labels = True
        , node_size = 50
        , node_color = nodes_col.values()   #'y'
        , font_size = 15
        , font_family = 'arial'
#         , edge_color = 'grey'   #'grey'  # b, k, m, g,
        , edgelist = [] #edge_list #[]
#         , nodelist = nodeList
       )

fig_name = 'network_' + str(customerNum) + '_1000.jpg'
plt.savefig(fig_name, dpi=600)
plt.show()

# # Build and solve VRPTW
big_M = 100000

# creat the model
model = Model('VRPTW')

# decision variables
x = {}
x_var = {}
s = {}
for i in range(data.nodeNum):
    for k in range(data.vehicleNum):
        name = 's_' + str(i) + '_' + str(k)
        s[i, k] = model.addVar(lb = data.readyTime[i] # 0 # data.readyTime[i]
                               , ub = data.dueTime[i]    #  1e15 # data.dueTime[i]
                               , vtype = GRB.CONTINUOUS
                               , name = name
                               )
```

```
        for j in range(data.nodeNum):
            if(i != j and data.arcs[i,j] == 1):
                name = 'x_' + str(i) + '_' + str(j) + '_' + str(k)
                x[i, j, k] = model.addVar(lb = 0
                                          , ub = 1
                                          , vtype = GRB.BINARY
                                          , name = name)
                x_var[i, j, k] = model.addVar(lb = 0
                                          , ub = 1
                                          , vtype = GRB.CONTINUOUS
                                          , name = name)

# Add constraints
# create the objective expression
obj = LinExpr(0)
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        if(i != j and data.arcs[i,j] == 1):
            for k in range(data.vehicleNum):
                obj.addTerms(data.disMatrix[i][j], x[i,j,k])
#print(model.getObjective()); # 这个可以打印出目标函数
# add the objective function into the model
model.setObjective(obj, GRB.MINIMIZE)

# constraint (1)
for i in range(1, data.nodeNum - 1): # 这里需要注意，i 的取值范围，否则可能会加入空约束
    expr = LinExpr(0)
    for j in range(data.nodeNum):
        if(i != j and data.arcs[i,j] == 1):
            for k in range(data.vehicleNum):
                if(i != 0 and i != data.nodeNum - 1):
                    expr.addTerms(1, x[i,j,k])

    model.addConstr(expr == 1, "c1")
    expr.clear()

# constraint (2)
for k in range(data.vehicleNum):
    expr = LinExpr(0);
    for i in range(1, data.nodeNum - 1):
        for j in range(data.nodeNum):
            if(i != 0 and i != data.nodeNum - 1 and i != j and data.arcs[i,j] == 1):
                expr.addTerms(data.demand[i], x[i,j,k])
    model.addConstr(expr <= data.capacity, "c2")
    expr.clear()

# constraint (3)
for k in range(data.vehicleNum):
    expr = LinExpr(0)
    for j in range(1, data.nodeNum): # 处处注意，不能有 i == j 的情况出现
        if(data.arcs[0,j] == 1):
            expr.addTerms(1.0, x[0,j,k])
    model.addConstr(expr == 1.0, "c3")
    expr.clear()

# constraint (4)
for k in range(data.vehicleNum):
    for h in range(1, data.nodeNum - 1):
        expr1 = LinExpr(0)
```

```
272             expr2 = LinExpr(0)
273             for i in range(data.nodeNum):
274                 if(h != i and data.arcs[i,h] == 1):
275                     expr1.addTerms(1, x[i,h,k])
276 
277             for j in range(data.nodeNum):
278                 if(h != j and data.arcs[h,j] == 1):
279                     expr2.addTerms(1, x[h,j,k])
280 
281             model.addConstr(expr1 == expr2, "c4")
282             expr1.clear()
283             expr2.clear()
284 
285     # constraint (5)
286     for k in range(data.vehicleNum):
287         expr = LinExpr(0)
288         for i in range(data.nodeNum - 1): # 这个地方也要注意，是 data.nodeNum - 1, 不是 data.nodeNum
289             if(data.arcs[i,data.nodeNum - 1] == 1):
290                 expr.addTerms(1, x[i,data.nodeNum - 1,k])
291         model.addConstr(expr == 1, "c5")
292         expr.clear()
293 
294     # constraint (6)
295     big_M = 0
296     for i in range(data.nodeNum):
297         for j in range(data.nodeNum):
298             big_M = max(data.dueTime[i] + data.disMatrix[i][j] - data.readyTime[i], big_M)
299 
300     for k in range(data.vehicleNum):
301         for i in range(data.nodeNum):
302             for j in range(data.nodeNum):
303                 if(i != j and data.arcs[i,j] == 1):
304                     model.addConstr(s[i,k] + data.disMatrix[i][j] - s[j,k] <= big_M  - big_M * x[i,j,k], "c6")
305 
306     # model.setParam('MIPGap', 0)
307     model.optimize()
308 
309     print("\n\n-----optimal value-----")
310     print(model.ObjVal)
311 
312     edge_list = []
313     for key in x.keys():
314         if(x[key].x > 0):
315             print(x[key].VarName, ' = ', x[key].x)
316             arc = (str(key[0]), str(key[1]))
317             edge_list.append(arc)
318 
319     nodes_col['0'] = 'red'
320     nodes_col[str(data.nodeNum-1)] = 'red'
321     plt.rcParams['figure.figsize'] = (15, 15) # 单位是 inches
322     nx.draw(Graph
323             , pos=pos_location
324     #         , with_labels = True
325             , node_size = 50
326             , node_color = nodes_col.values()   #'y'
327             , font_size = 15
328             , font_family = 'arial'
329     #         , edge_color = 'grey'   #'grey'  # b, k, m, g,
330             , edgelist = edge_list
```

```
#         , nodelist = nodeList
        )
fig_name = 'network_' + str(customerNum) + '_1000.jpg'
plt.savefig(fig_name, dpi=600)
plt.show()

# # Node class
class Node:
    # this class defines the node
    def __init__(self):
        self.local_LB = 0
        self.local_UB = np.inf
        self.x_sol = {}
        self.x_int_sol = {}
        self.branch_var_list = []
        self.model = None
        self.cnt = None
        self.is_integer = False
        self.var_LB = {}
        self.var_UB = {}

    def deepcopy_node(node):
        new_node = Node()
        new_node.local_LB = 0
        new_node.local_UB = np.inf
        new_node.x_sol = copy.deepcopy(node.x_sol)
        new_node.x_int_sol = copy.deepcopy(node.x_int_sol)
        new_node.branch_var_list = []
        new_node.model = node.model.copy()
        new_node.cnt = node.cnt
        new_node.is_integer = node.is_integer

        return new_node

# # Branch and Cut framework
def Branch_and_Cut(VRPTW_model, x_var, summary_interval):
    Relax_VRPTW_model = VRPTW_model.relax()
    # initialize the initial node
    Relax_VRPTW_model.optimize()
    global_UB = np.inf
    global_LB = Relax_VRPTW_model.ObjVal
    eps = 1e-6
    incumbent_node = None
    Gap = np.inf
    feasible_sol_cnt = 0

    # initialize the cuts pool, this dict aims to store all the cuts generated during branch and bound procedure
    Cuts_pool = {}
    Cuts_LHS = {}
    Cut_cnt = 0

    '''
       Branch and Cut starts
    '''

    # creat initial node
    Queue = []
    node = Node()
    node.local_LB = global_LB
```

```
390         node.local_UB = np.inf
391         node.model = Relax_VRPTW_model.copy()
392         node.model.setParam("OutputFlag", 0)
393         node.cnt = 0
394         Queue.append(node)
395 
396         cnt = 0
397         Global_UB_change = []
398         Global_LB_change = []
399         while (len(Queue) > 0 and global_UB - global_LB > eps):
400             # select the current node
401             current_node = Queue.pop()
402             cnt += 1
403 
404             # solve the current model
405             current_node.model.optimize()
406             Solution_status = current_node.model.Status
407 
408             '''
409             OPTIMAL = 2
410             INFEASIBLE = 3
411             UNBOUNDED = 5
412             '''
413 
414             # check whether the current solution is integer and execute prune step
415             '''
416                 is_integer : mark whether the current solution is integer solution
417                 Is_Pruned  : mark whether the current solution is pruned
418             '''
419             is_integer = True
420             Is_Pruned = False
421             if (Solution_status == 2):
422                 for var in current_node.model.getVars():
423                     if(var.VarName.startswith('x')):
424                         current_node.x_sol[var.varName] = var.x
425                         # record the branchable variable
426                         if(abs(round(var.x, 0) - var.x) >= eps):
427                         # if(round(var.x, 0) != var.x):  # 如果改成在一定精度范围内，就会出现错误，结果不是最优解
428                             is_integer = False
429 #                           current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
    ↪ variables
430 #                           print(var.VarName, ' = ', var.x)
431 #             print('Before x_sol :', current_node.x_sol)
432 #             print('Before:', current_node.model.NumConstrs)
433                 '''
434                     Cuts Generation
435                 '''
436                 # cut generation
437                 temp_cut_cnt = 0
438                 if (is_integer == False):
439                     # generate cuts
440                     # generate at most 5 cuts on each node
441                     customer_set = list(range(data.nodeNum))[1:-1]  # 去掉了两个 depot 点
442 
443                     while(temp_cut_cnt <= 5):
444                         sample_num = random.choice(customer_set[3:])
445 #                       sample_num = 15
446                         selected_customer_set = random.sample(customer_set, sample_num) # 每次运行结果不同。
447 #                       print('selected_customer_set :', selected_customer_set)
```

```
                    estimated_veh_num = 0
                    total_demand = 0
                    for customer in selected_customer_set:
                        total_demand += data.demand[customer]
                    estimated_veh_num = math.ceil(total_demand / data.capacity)

                    current_node.model._vars = x_var
                    # creat cut
                    cut_lhs = LinExpr(0)
                    for key in x_var.keys():
                        key_org = key[0]
                        key_des = key[1]
                        key_vehicle = key[2]
                        if(key_org not in selected_customer_set and key_des in selected_customer_set):
                            var_name = 'x_' + str(key_org) + '_' + str(key_des) + '_' + str(key_vehicle)
#                             print(type(current_node.model.getVarByName(var_name)))
                            cut_lhs.addTerms(1, current_node.model.getVarByName(var_name))
                    Cut_cnt += 1
                    temp_cut_cnt += 1
                    cut_name = 'Cut_' + str(Cut_cnt)
                    Cuts_pool[cut_name] = current_node.model.addConstr(cut_lhs >= estimated_veh_num, name =
                    ↪  cut_name)
                    Cuts_LHS[cut_name] = cut_lhs
                # lazy update
                current_node.model.update()
            '''
                Cuts Generation ends
            '''
#             print('After:', current_node.model.NumConstrs)
            # solve the added cut model
            current_node.model.optimize()
#             current_node.branch_var_list = []
#             current_node.x_sol = {}
#             print('Status:', current_node.model.status)
            is_integer = True
            if (current_node.model.status == 2):
                for var in current_node.model.getVars():
                    if(var.VarName.startswith('x')):
                        current_node.x_sol[var.VarName] =
                        ↪  copy.deepcopy(current_node.model.getVarByName(var.VarName).x)
#                         current_node.x_sol.update({var.VarName:var.x})
#                         print(var.VarName, ' = ***', var.x)
#                         print(var.VarName, ' = ***', current_node.x_sol[var.VarName])
    #                     print(var.VarName, ' = ', var.x)
                        # record the branchable variable
    #                     if(abs((int)(var.x) - var.x) >= eps):
                        if(abs(round(var.x, 0) - var.x) >= eps):
    #                     if(round(var.x, 0) != var.x):  # 如果改成在一定精度范围内，就会出现错误，结果不是最优解
                            is_integer = False
                            current_node.branch_var_list.append(var.VarName)  # to record the candidate branch
                            ↪  variables
#                             print(var.VarName, ' = ***', var.x)
            else:
                continue

#                 print('branch_var_list:', current_node.branch_var_list)
#                 print('x_sol:', current_node.x_sol)

            # update the LB and UB
```

```
504             if (is_integer == True):
505                 feasible_sol_cnt +=  1
506                 # For integer solution node, update the LB and UB
507                 current_node.is_integer = True
508                 current_node.local_LB = current_node.model.ObjVal
509                 current_node.local_UB = current_node.model.ObjVal
510                 # if the solution is integer, uodate the UB of global and update the incumbent
511                 if (current_node.local_UB < global_UB):
512                     global_UB = current_node.local_UB
513                     incumbent_node = Node.deepcopy_node(current_node)
514             if (is_integer == False):
515                 # For integer solution node, update the LB and UB also
516                 current_node.is_integer = False
517                 current_node.local_UB = global_UB
518                 current_node.local_LB = current_node.model.ObjVal
519 
520             '''
521                 PRUNE step
522             '''
523             # prune by optimility
524             if (is_integer == True):
525                 Is_Pruned = True
526 
527             # prune by bound
528             if (is_integer == False and current_node.local_LB > global_UB):
529                 Is_Pruned = True
530 
531             Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
532 
533         elif (Solution_status != 2):
534             # the current node is infeasible or unbound
535             is_integer = False
536 
537             '''
538                 PRUNE step
539             '''
540             # prune by infeasiblity
541             Is_Pruned = True
542 
543             continue
544 
545         '''
546           BRANCH STEP
547         '''
548         if (Is_Pruned == False):
549             # selecte the branch variable: choose the value which is closest to 0.5
550             branch_var_name = None
551 
552             min_diff = 100
553             for var_name in current_node.branch_var_list:
554                 if(abs(current_node.x_sol[var_name] - 0.5) < min_diff):
555                     branch_var_name = var_name
556                     min_diff = abs(current_node.x_sol[var_name] - 0.5)
557 #                    print('branch:', var_name, ' = ', current_node.x_sol[branch_var_name])
558 #             branch_var_name = current_node.branch_var_list[0]
559 
560             # choose the variable cloest to 0 or 1
561 #             min_diff = 100
562 #             for var_name in current_node.branch_var_list:
```

```
563  #                 diff = max(abs(current_node.x_sol[var_name] - 1), abs(current_node.x_sol[var_name] - 0))
564  #                 if(min_diff >= diff):
565  #                     branch_var_name = var_name
566  #                     min_diff = diff
567
568
569  #             branch_var_name = current_node.branch_var_list[0]
570              if(cnt % summary_interval == 0):
571                  print('Branch var name :', branch_var_name, '\t, branch var value :',
                     current_node.x_sol[branch_var_name])
572              left_var_bound = (int)(current_node.x_sol[branch_var_name])
573              right_var_bound = (int)(current_node.x_sol[branch_var_name]) + 1
574
575              # creat two child nodes
576              left_node = Node.deepcopy_node(current_node)
577              right_node = Node.deepcopy_node(current_node)
578
579              # creat left child node
580              temp_var = left_node.model.getVarByName(branch_var_name)
581              left_node.model.addConstr(temp_var <= left_var_bound, name='branch_left_' + str(cnt))
582              left_node.model.setParam("OutputFlag", 0)
583              left_node.model.update()
584              cnt += 1
585              left_node.cnt = cnt
586
587              # creat right child node
588              temp_var = right_node.model.getVarByName(branch_var_name)
589              right_node.model.addConstr(temp_var >= right_var_bound, name='branch_right_' + str(cnt))
590              right_node.model.setParam("OutputFlag", 0)
591              right_node.model.update()
592              cnt += 1
593              left_node.cnt = cnt
594
595              Queue.append(left_node)
596              Queue.append(right_node)
597
598              # update the global LB, explore all the leaf nodes
599              temp_global_LB = np.inf
600              for node in Queue:
601                  node.model.optimize()
602                  if(node.model.status == 2):
603                      if(node.model.ObjVal <= temp_global_LB and node.model.ObjVal <= global_UB):
604                          temp_global_LB = node.model.ObjVal
605
606
607              global_LB = temp_global_LB
608              Global_UB_change.append(global_UB)
609              Global_LB_change.append(global_LB)
610
611          if(cnt % summary_interval == 0):
612              print('\n\n====================')
613              print('Queue length :', len(Queue))
614              print('\n ------------ \n', cnt, ' UB  = ', global_UB, '   LB = ', global_LB, '\t Gap  = ', Gap, '  %',
                     'feasible_sol_cnt :', feasible_sol_cnt)
615              print('Cut pool size :', len(Cuts_pool))
616              NAME = list(Cuts_LHS.keys())[-1]
617  #             print('last cut :', Cuts_LHS[cut_name])
618              print('RHS :', estimated_veh_num)
619              print('Cons Num :', current_node.model.NumConstrs)
```

```
620
621
622         # all the nodes are explored, update the LB and UB
623         incumbent_node.model.optimize()
624         global_UB = incumbent_node.model.ObjVal
625         global_LB = global_UB
626         Gap = round(100 * (global_UB - global_LB) / global_LB, 2)
627         Global_UB_change.append(global_UB)
628         Global_LB_change.append(global_LB)
629
630         print('\n\n\n\n')
631         print('----------------------------------------')
632         print('         Branch and Cut terminates       ')
633         print('         Optimal solution found          ')
634         print('----------------------------------------')
635         print('\nIter cnt  = ', cnt, ' \n\n')
636         print('\nFinal Gap  = ', Gap, '  % \n\n')
637         # print('Optimal Solution:', incumbent_node.x_sol)
638         print(' ---- Optimal Solution ------')
639         for key in incumbent_node.x_sol.keys():
640             if(incumbent_node.x_sol[key] > 0):
641                 print(key, ' = ', incumbent_node.x_sol[key])
642         print('\nOptimal Obj:', global_LB)
643
644         return incumbent_node, Gap, Global_UB_change, Global_LB_change
645
646
647     # # Branch and Cut Solve the IP model
648
649     incumbent_node, Gap, Global_UB_change, Global_LB_change = Branch_and_Cut(model, x_var, summary_interval = 100)
650
651     for key in incumbent_node.x_sol.keys():
652         if(incumbent_node.x_sol[key] > 0):
653             print(key, ' = ', incumbent_node.x_sol[key])
654
655
656     # # plot the results
657
658     # fig = plt.figure(1)
659     # plt.figure(figsize=(15,10))
660     font_dict = {"family":'Arial',    #"Kaiti",
661            "style":"oblique",
662             "weight":"normal",
663             "color":"green",
664             "size": 20
665            }
666
667     plt.rcParams['figure.figsize'] = (12.0, 8.0) # 单位是 inches
668     plt.rcParams["font.family"] = 'Arial' #"SimHei"
669     plt.rcParams["font.size"] = 16
670     # plt.xlim(0, len(Global_LB_change) + 1000)
671
672     x_cor = range(1, len(Global_LB_change) + 1)
673     plt.plot(x_cor, Global_LB_change, label = 'LB')
674     plt.plot(x_cor, Global_UB_change, label = 'UB')
675     plt.legend()
676     plt.xlabel('Iteration', fontdict=font_dict)
677     plt.ylabel('Bounds update', fontdict=font_dict)
678     plt.title('VRPTW (c101-10) : Bounds update during branch and cut procedure \n', fontsize = 23)
```

```
plt.savefig('BnC_Bound_updates_VRPTW_c101_60.eps')
plt.savefig('BnC_Bound_updates_VRPTW_c101_60.pdf')
plt.show()
```

11.8.2 Python 调用 Gurobi 实现分支切割算法求解 VRPTW 完整代码：callback 添加 cut 的版本

callback 版本的代码，只需要在构建完 VRPTW 的 MIP 模型后，直接求解 VRPTW 模型，并且将 callback 函数 Cutting_plane_callback作为参数即可。完整代码如下 (读取数据部分的代码与上一小节完全相同)。

BnC VRPTW callback

```

# Callback use callback to add cutting planes
def Cutting_plane_callback(model, where):
    if (where == GRB.Callback.MIPNODE):
        status = model.cbGet(GRB.Callback.MIPNODE_STATUS)
        if(status == GRB.OPTIMAL):
            # obtain the current solution of current node
            # initialize the cuts pool, this dict aims to store all the cuts generated during branch and bound
            ↪ procedure
            Cuts_pool = {}
            Cuts_LHS = {}
            Cut_cnt = 0

            # print('Execute Callback')

            is_integer = False
            eps = 1e-5
            x_sol = {}
            Vars = model._vars

            for key in Vars.keys():
                var = Vars[key]
                x_sol[key] = copy.deepcopy(model.cbGetNodeRel ( var ) )  # 这里不能用 var.x 会报错 AttributeError:
                var_value = model.cbGetNodeRel ( var )
                if(abs(round(var_value, 0) - var_value) >= eps):
                    is_integer = False

            # cut generation
            temp_cut_cnt = 0
            if (is_integer == False):
                # generate cuts
                # generate at most 5 cuts on each node
                customer_set = list(range(data.nodeNum))[1:-1]  # 去掉了两个 depot 点

                while(temp_cut_cnt <= 5):
                    sample_num = random.choice(customer_set[3:])
                    sample_num = 15
                    selected_customer_set = random.sample(customer_set, sample_num) # 每次运行结果不同。
    #                     print('selected_customer_set :', selected_customer_set)
                    estimated_veh_num = 0
                    total_demand = 0
                    for customer in selected_customer_set:
                        total_demand += data.demand[customer]
                    estimated_veh_num = math.ceil(total_demand / data.capacity)
```

```
44
45                      # creat cut
46                      cut_lhs = LinExpr(0)
47                      for key in Vars.keys():
48                          key_org = (int)(key[0])
49                          key_des = (int)(key[1])
50                          key_vehicle = (int)(key[2])
51                          if(key_org not in selected_customer_set and key_des in selected_customer_set):
52  #                               var_name = 'x_' + str(key_org) + '_' + str(key_des) + '_' + str(key_vehicle)
53      #                               print(type(current_node.model.getVarByName(var_name)))
54                              cut_lhs.addTerms(1, model._vars[key])
55                      Cut_cnt += 1
56                      temp_cut_cnt += 1
57                      cut_name = 'Cut_' + str(Cut_cnt)
58                      Cuts_pool[cut_name] = model.cbCut(cut_lhs >= estimated_veh_num)
59                      Cuts_LHS[cut_name] = cut_lhs
60                      print('cut_name = ', cut_name)
61                      print('LinExpr = ', cut_lhs)
62                      print('RHS = ', estimated_veh_num)
63
64              '''
65                  Cuts Generation ends
66              '''
67  # Build VRPTW model
68  big_M = 100000
69
70  # creat the model
71  model = Model('VRPTW')
72
73  # decision variables
74  x = {}
75  x_var = {}
76  s = {}
77  for i in range(data.nodeNum):
78      for k in range(data.vehicleNum):
79          name = 's_' + str(i) + '_' + str(k)
80          s[i, k] = model.addVar(lb = data.readyTime[i] # 0 # data.readyTime[i]
81                                 , ub = data.dueTime[i]    #  1e15 # data.dueTime[i]
82                                 , vtype = GRB.CONTINUOUS
83                                 , name = name
84                                 )
85          for j in range(data.nodeNum):
86              if(i != j and data.arcs[i,j] == 1):
87                  name = 'x_' + str(i) + '_' + str(j) + '_' + str(k)
88                  x[i, j, k] = model.addVar(lb = 0
89                                            , ub = 1
90                                            , vtype = GRB.BINARY
91                                            , name = name)
92                  x_var[i, j, k] = model.addVar(lb = 0
93                                            , ub = 1
94                                            , vtype = GRB.CONTINUOUS
95                                            , name = name)
96
97  # Add constraints
98  # create the objective expression
99  obj = LinExpr(0)
100 for i in range(data.nodeNum):
101     for j in range(data.nodeNum):
102         if(i != j and data.arcs[i,j] == 1):
```

```
            for k in range(data.vehicleNum):
                obj.addTerms(data.disMatrix[i][j], x[i,j,k])
#print(model.getObjective()); # 这个可以打印出目标函数
# add the objective function into the model
model.setObjective(obj, GRB.MINIMIZE)

# constraint (1)
for i in range(1, data.nodeNum - 1): # 这里需要注意，i 的取值范围，否则可能会加入空约束
    expr = LinExpr(0)
    for j in range(data.nodeNum):
        if(i != j and data.arcs[i,j] == 1):
            for k in range(data.vehicleNum):
                if(i != 0 and i != data.nodeNum - 1):
                    expr.addTerms(1, x[i,j,k])

    model.addConstr(expr == 1, "c1")
    expr.clear()

# constraint (2)
for k in range(data.vehicleNum):
    expr = LinExpr(0);
    for i in range(1, data.nodeNum - 1):
        for j in range(data.nodeNum):
            if(i != 0 and i != data.nodeNum - 1 and i != j and data.arcs[i,j] == 1):
                expr.addTerms(data.demand[i], x[i,j,k])
    model.addConstr(expr <= data.capacity, "c2")
    expr.clear()

# constraint (3)
for k in range(data.vehicleNum):
    expr = LinExpr(0)
    for j in range(1, data.nodeNum): # 处处注意，不能有 i == j 的情况出现
        if(data.arcs[0,j] == 1):
            expr.addTerms(1.0, x[0,j,k])
    model.addConstr(expr == 1.0, "c3")
    expr.clear()

# constraint (4)
for k in range(data.vehicleNum):
    for h in range(1, data.nodeNum - 1):
        expr1 = LinExpr(0)
        expr2 = LinExpr(0)
        for i in range(data.nodeNum):
            if(h != i and data.arcs[i,h] == 1):
                expr1.addTerms(1, x[i,h,k])

        for j in range(data.nodeNum):
            if(h != j and data.arcs[h,j] == 1):
                expr2.addTerms(1, x[h,j,k])

        model.addConstr(expr1 == expr2, "c4")
        expr1.clear()
        expr2.clear()

# constraint (5)
for k in range(data.vehicleNum):
    expr = LinExpr(0)
    for i in range(data.nodeNum - 1): # 这个地方也要注意，是 data.nodeNum - 1, 不是 data.nodeNum
        if(data.arcs[i,data.nodeNum - 1] == 1):
```

```
            expr.addTerms(1, x[i,data.nodeNum - 1,k])
    model.addConstr(expr == 1, "c5")
    expr.clear()

# constraint (6)
big_M = 0
for i in range(data.nodeNum):
    for j in range(data.nodeNum):
        big_M = max(data.dueTime[i] + data.disMatrix[i][j] - data.readyTime[i], big_M)

for k in range(data.vehicleNum):
    for i in range(data.nodeNum):
        for j in range(data.nodeNum):
            if(i != j and data.arcs[i,j] == 1):
                model.addConstr(s[i,k] + data.disMatrix[i][j] - s[j,k] <= big_M  - big_M * x[i,j,k], "c6")




model._vars = x
model.Params.PreCrush = 1  # You must turn this parameter on when you are using callbacks to add your own cuts.
# model.Params.CliqueCuts = 0

# model.setParam('CliqueCuts', 0)
# model.setParam('CoverCuts', 0)
# model.setParam('FlowCoverCuts', 0)
# model.setParam('FlowPathCuts', 0)
# model.setParam('GUBCoverCuts', 0)
# model.setParam('ImpliedCuts', 0)
# model.setParam('InfProofCuts', 0)
# model.setParam('MIPSepCuts', 0)
# model.setParam('MIRCuts', 0)
# model.setParam('ModKCuts', 0)
# model.setParam('NetworkCuts', 0)
# model.setParam('ProjImpliedCuts', 0)
# model.setParam('RelaxLiftCuts', 0)
# model.setParam('RLTCuts', 0)
# model.setParam('StrongCGCuts', 0)
# model.setParam('SubMIPCuts', 0)
# model.setParam('ZeroHalfCuts', 0)

model.setParam('Cuts', 0)    # Global cut aggressiveness setting. Use value 0 to shut off cuts,
model.update()

model.optimize(Cutting_plane_callback)
```

11.9 Java 调用 CPLEX 实现分支切割算法求解 CVRP: 回调函数添加割平面的版本

本小节我们提供实现 Branch and Cut 的另外一种方法：基于求解器提供的 callback 函数的实现。这种方法省去了自己实现 Branch and Bound 算法的过程，便于研究者将主要精力集中在如何构建 Cut 上，并且实现起来更容易。

我们以 Capacitated Vehicle Routing Problem (CVRP) 为例，来介绍结合 callback 的

Branch and Cut 的实现。本代码通过简单的修改之后，也可以用于求解 VRPTW。CVRP 的模型和本节将要实现的 Cutting Plane 的详细介绍见教材正文。

11.9.1 CVRP 的基本模型

11.9.2 割平面

11.9.3 Java 调用 CPLEX 实现分支切割算法求解 CVRP 完整代码

下面是 Java 调用 CPLEX 实现分支切割算法求解 CVRP 完整代码。

Instance 类

Instance.java

```java
package VRP;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class Instance {

    private int line_location=0;
    private int nodeNum = 0;
    private int car_num;
    private int capacity;
    private ArrayList<Node> nodeList = new ArrayList<Node>();
    private double [][]distance;

    public void initInstance(File filename) {
        try {
            InputStreamReader read = new InputStreamReader(new FileInputStream(filename));
            BufferedReader br = new BufferedReader(read);
            String lineTxt = "";
            while ((lineTxt = br.readLine()) != null) {
                this.line_location += 1;
                    if(this.line_location == 2) {
                    this.car_num = Integer.parseInt(lineTxt.substring(lineTxt.indexOf("trucks") + 8,
                    ↪  lineTxt.indexOf(", Optimal value")));
                    }
                    else if(this.line_location == 4) {
                    this.nodeNum =Integer.parseInt(lineTxt.split(" ")[2]);
                    }
                    else if(this.line_location == 6) {
                    this.capacity=Integer.parseInt(lineTxt.split(" ")[2]);
                 }
                    else if(this.line_location >= 8 && this.line_location < 8 + this.nodeNum) {
                        Node node0 = new Node();
                        node0.setIndex(Integer.parseInt(lineTxt.split(" ")[1]));
                        node0.setX(Double.parseDouble(lineTxt.split(" ")[2]));
                        node0.setY(Double.parseDouble(lineTxt.split(" ")[3]));
                    this.nodeList.add(node0);
                    }
```

```
42                      else if(this.line_location >= 9 + this.nodeNum && this.line_location < 9 + 2 * this.nodeNum){
43                      this.nodeList.get(this.line_location - 9 -
                        ↪  this.nodeNum).setDemand(Double.parseDouble(lineTxt.split(" ")[1]));
44                      }
45              }
46              read.close();
47              calDistance(this.nodeList);
48          }catch (Exception e) {
49              e.printStackTrace();
50          }
51      }
52
53      public void calDistance(List<Node> nodeList) {
54          this.distance = new double[nodeNum][nodeNum];
55          for(int i = 0; i < nodeNum; i++) {
56              for(int j = 0; j < nodeNum; j++) {
57                  this.distance[i][j] = Math.round(10*Math.sqrt((nodeList.get(i).getX()-nodeList.get(j).getX())
58                  *(nodeList.get(i).getX()-nodeList.get(j).getX())+(nodeList.get(i).getY()
59                  -nodeList.get(j).getY())*(nodeList.get(i).getY()-nodeList.get(j).getY())))
60                  /10.0;
61              }
62          }
63      }
64
65      public int getNodeNum() {
66          return nodeNum;
67      }
68
69      public int getCar_num() {
70          return car_num;
71      }
72
73      public List<Node> getNodeList() {
74          return nodeList;
75      }
76
77      public int getCapacity() {
78          return capacity;
79      }
80
81      public double[][] getDistance() {
82          return distance;
83      }
84
85
86  }
```

Node 类

Node.java

```
1   package VRP;
2
3   public class Node {
4       private int index;
5       private double x;
6       private double y;
7       private double demand;
8
```

```
    public double getDemand() {
        return demand;
    }
    public void setDemand(double demand) {
        this.demand = demand;
    }
    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
    }
    public double getY() {
        return y;
    }
    public void setY(double y) {
        this.y = y;
    }
    public int getIndex() {
        return index;
    }
    public void setIndex(int index) {
        this.index = index;
    }

}
```

CVRP 类

CVRP.java

```
package VRP;


import java.io.File;

import ilog.concert.IloException;

public class CVRP {
    public static void main(String[] args) throws IloException {
        long startTime = System.nanoTime();

        String pathname="dataset/A-n32-k5.txt";
        File filename = new File(pathname);

        Instance instance = new Instance();
        instance.initInstance(filename);
        BranchCutAlgo algo = new BranchCutAlgo();
        algo.buildModel(instance.getNodeNum(), instance.getDistance(), instance.getCar_num(),instance);

        long endTime = System.nanoTime();
        System.out.println("\nduration time = "+(endTime - startTime)/1000000000.0+"s");
    }
}
```

Cut 类

Cut.java

```java
package VRP;

import java.util.ArrayList;

import ilog.concert.IloNumExpr;

public class Cut {
    ArrayList<IloNumExpr> cuts;
    ArrayList<Integer>right;
    public ArrayList<IloNumExpr> getLhsExprs() {
        return cuts;
    }
    public void setLhsExprs(ArrayList<IloNumExpr> cuts) {
        this.cuts = cuts;
    }
    public ArrayList<Integer> getRight() {
        return right;
    }
    public void setRight(ArrayList<Integer> right) {
        this.right = right;
    }

}
```

BranchCutAlgo 类

BranchCutAlgo.java

```java
package VRP;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import ilog.concert.*;
import ilog.cplex.IloCplex;

public class BranchCutAlgo {

    public static int pointtoedge(int i, int j, int nCus) {
        if(i==j) {
            return -1;
        }
        else if(i>j) {
            int temp=i;
            i=j;
            j=temp;
        }
        return nCus*i - i*(i + 1) / 2 + (j - i - 1);
    }

    public static boolean isCustomerInSet(int num, ArrayList<Integer> S) {
        for (int i = 0; i < S.size(); i++) {
            if(S.get(i)==num) {
                return true;
            }
```

```
29          }
30          return false;
31      }
32
33      public static int identifyMaxWeightCustomer(ArrayList<Integer> superVertex, double[] xSol, boolean[] cusLabels,
        ↪  int nCus) {
34          double maxWeight=0;
35          int customer = -1;
36
37          for(int j=1; j < nCus;j++) {
38              // filter customer that has checked
39              if(!cusLabels[j]){
40                  continue;
41              }
42              //find max weight
43              double weight=0;
44              for(int i=0;i<superVertex.size();i++) {
45                  weight+=xSol[pointtoedge(superVertex.get(i), j, nCus)];
46              }
47              if (weight > maxWeight){
48                  customer = j;
49                  maxWeight = weight;
50              }
51          }
52          return customer;
53      }
54
55      public static Cut makecuts(IloNumVar[] x, double[] xSol, int nCus, IloModeler ilcplex, Instance instance)
        ↪  throws IloException {
56          ArrayList<Integer>superVertex = new ArrayList<Integer>();
57          ArrayList<Integer>right = new ArrayList<Integer>();
58          Cut cut =new Cut();
59          ArrayList<IloNumExpr> cutLhs = new ArrayList<IloNumExpr>();
60
61          // for each customer
62          boolean[] cusLabels = new boolean[nCus];
63          Arrays.fill(cusLabels,true);
64
65          while (true)
66          {
67              // find the open edge with the highest weight
68              int iMax=-1, jMax=-1;
69              double wMax = 0.001; // a very small positive real;
70              for (int i = 1; i < nCus; i++)
71              {
72                  for (int j = i + 1; j < nCus; j++)
73                  {
74                      if (cusLabels[i] == false || cusLabels[j] == false) // exclude the closed customers
75                          continue;
76
77                      double weight = xSol[pointtoedge(i, j, nCus)];
78                      if (weight > wMax )
79                      {
80                          iMax = i;
81                          jMax = j;
82                          break;
83                      }
84                  }
85              }
```

```
            // if no positive edge is found, the whole procedure ends.
            if (iMax == -1 && jMax == -1)
            {
                break;
            }

            // merge two customers iMax and jMax as a super vertex;
            superVertex.clear();
            superVertex.add(iMax);
            superVertex.add(jMax);
            // mark the two customers as closed
            cusLabels[iMax] = false;
            cusLabels[jMax] = false;

            // while there is any open vertex
            while (true)
            {
                // find the customer from all open customers with maximum weight with SV;
                int cus = identifyMaxWeightCustomer(superVertex, xSol, cusLabels, nCus);
                if (cus != -1)
                {
                    //label cus as closed;
                    cusLabels[cus] = false;
                    // merge SV and cus as a supervertex
                    superVertex.add(cus);
                }
                else
                {
                    break; // from while(true)
                }
            }
            //check if superVertex violates the capacity constraint, if yes, use it to generate a constraint;
            double lhs = 0, rhs = 0;

            IloLinearNumExpr lhsExpr=ilcplex.linearNumExpr();
            for (int i = 0; i < superVertex.size(); i++)
            {
                for (int j = 1; j < nCus; j++)
                {
                    if (isCustomerInSet(j,superVertex))
                        continue;
                    lhs += xSol[pointtoedge(superVertex.get(i), j, nCus)];
                    lhsExpr.addTerm(1, x[pointtoedge(superVertex.get(i), j, nCus)]);

                }
                lhs += xSol[pointtoedge(superVertex.get(i), 0, nCus)];
                lhsExpr.addTerm(1, x[pointtoedge(superVertex.get(i), 0, nCus)]);
                rhs += instance.getNodeList().get(superVertex.get(i)).getDemand();
            }
            rhs = 2*Math.ceil(rhs/instance.getCapacity());    // round up

            if (lhs < rhs) //violated
            {
                cutLhs.add(lhsExpr);
                right.add((int)rhs);
            }
            //check if superVertex violates the volume constraint, if yes, use it to generate a constraint;
        }
        cut.setLhsExprs(cutLhs);
```

```
145             cut.setRight(right);
146
147             return cut;
148         }
149
150         public void buildModel(int nodeNum, double[][] distance, int car_num, Instance instance) throws IloException {
151             IloCplex ilcplex =new IloCplex();
152             IloNumVar[] x;
153
154             //variables
155             int variable_num=nodeNum*(nodeNum-1)/2;
156             x = new IloNumVar[variable_num];
157             //set range
158             for(int i = 0; i < variable_num; i ++) {
159                 if(i < nodeNum - 1){
160                     x[i]=ilcplex.intVar(0,2);
161                 }
162                 else {
163                     x[i]=ilcplex.intVar(0,1);
164                 }
165             }
166
167             //objective
168             double[] OneDimensionDistance=new double[variable_num];
169             for(int i=0;i<nodeNum;i++) {
170                 for(int j=i+1;j<nodeNum;j++) {
171                     OneDimensionDistance[pointtoedge(i, j, nodeNum)]=distance[i][j];
172                 }
173             }
174             IloLinearNumExpr obj=ilcplex.linearNumExpr();
175             obj.addTerms(OneDimensionDistance, x);
176             ilcplex.addMinimize(obj);
177
178             //constraints
179             for(int i=0;i<nodeNum;i++) {
180                 IloLinearNumExpr left=ilcplex.linearNumExpr();
181                 for(int j = i + 1; j < nodeNum; j ++) {
182                     left.addTerm(1, x[pointtoedge(i, j, nodeNum)]); //edges from i
183                 }
184                 for(int j = i - 1; j >= 0; j --) {
185                     left.addTerm(1, x[pointtoedge(j, i, nodeNum)]); //edges to i
186                 }
187                 if(i==0) {
188                     ilcplex.addLe(left, 2*car_num);//对每个 i 加入约束
189                 }
190                 else {
191                     ilcplex.addEq(left, 2);//对每个 i 加入约束
192                 }
193             }
194             ilcplex.exportModel("CVRPModel.lp");
195
196             //callback
197             ilcplex.use(new Callback(x,ilcplex,nodeNum,instance));
198             ilcplex.use(new LazyCallback(x,ilcplex,nodeNum,instance));
199
200             if(ilcplex.solve()) {
201                 System.out.println("The objective value is:" + ilcplex.getObjValue());
202                 double[] xVal = ilcplex.getValues(x);
203
```

```
            ArrayList<Integer> S = new ArrayList<>(); //Prevent an edge from being searched repeatedly
            for (int i = 1; i < nodeNum; i++) {
                if (!isCustomerInSet(i, S) && xVal[pointtoedge(0, i, nodeNum)] > 1 - 1e-3) {
                    System.out.print("[0-" + i);
                    int currNode = i;
                    S.add(currNode);
                    boolean flag = true;
                    while (flag) {
                        flag = false;
                        for (int j = 1; j < nodeNum; j++) {
                            if (!isCustomerInSet(j, S) && Math.abs(xVal[pointtoedge(currNode, j, nodeNum)] - 1) <
                            ↪  1e-3) {
                                System.out.print("-" + j);
                                currNode = j;
                                S.add(currNode);
                                flag = true;
                                break;
                            }
                        }
                    }
                    System.out.println("-0]");
                }
            }
        }
    }

    public static class Callback extends IloCplex.UserCutCallback{
        Cut cut;
        ArrayList<IloNumExpr> cutLhs;
        ArrayList<Integer> cutRhs;
        IloNumVar[] x;
        IloCplex ilcplex;
        int nCus;
        Instance instance;
        Callback(IloNumVar[] x0,IloCplex ilcplex0,int nCus0,Instance instance0){
            x=x0;
            ilcplex=ilcplex0;
            nCus=nCus0;
            instance=instance0;
        }

        public void main() throws IloException{
            double[] xSol = getValues(x);
            //生成 cut
            cut = makecuts(x, xSol, nCus, ilcplex, instance);
            //添加
            cutLhs = cut.getLhsExprs();
            cutRhs= cut.getRight();
            for(int i = 0; i< cutLhs.size(); i++) {
                addLocal(ilcplex.ge(cutLhs.get(i), cutRhs.get(i)));
            }
        }
    }


    public static class LazyCallback extends IloCplex.LazyConstraintCallback {
        Cut cut;
        ArrayList<IloNumExpr> cutLhs;
        ArrayList<Integer> cutRhs;
```

```
        IloNumVar[] x;
        IloCplex ilcplex;
        int nCus;
        Instance instance;
        LazyCallback(IloNumVar[] x0,IloCplex ilcplex0,int nCus0,Instance instance0){
            x=x0;
            ilcplex=ilcplex0;
            nCus=nCus0;
            instance=instance0;
        }

        public void main() throws IloException{
            double[] xSol = getValues(x);
            cut = makecuts(x, xSol, nCus, ilcplex, instance);
            cutLhs = cut.getLhsExprs();
            cutRhs= cut.getRight();
            for(int i = 0; i< cutLhs.size(); i++) {
                add(ilcplex.ge(cutLhs.get(i), cutRhs.get(i)));
            }
        }
    }

}
```

测试算例

下面为本代码涉及到的测试算例 `A-n32-k5.txt`。

A-n32-k5.txt

```
NAME : A-n32-k5
COMMENT : (Augerat et al, Min no of trucks: 5, Optimal value: 784)
TYPE : CVRP
DIMENSION : 32
EDGE_WEIGHT_TYPE : EUC_2D
CAPACITY : 100
NODE_COORD_SECTION
 1 82 76
 2 96 44
 3 50 5
 4 49 8
 5 13 7
 6 29 89
 7 58 30
 8 84 39
 9 14 24
 10 2 39
 11 3 82
 12 5 10
 13 98 52
 14 84 25
 15 61 59
 16 1 65
 17 88 51
 18 91 2
 19 19 32
 20 93 3
 21 50 93
 22 98 14
```

```
 23 5 42
 24 42 9
 25 61 62
 26 9 97
 27 80 55
 28 57 69
 29 23 15
 30 20 70
 31 85 60
 32 98 5
DEMAND_SECTION
1 0
2 19
3 21
4 6
5 19
6 7
7 12
8 16
9 6
10 16
11 8
12 14
13 21
14 16
15 3
16 22
17 18
18 19
19 1
20 24
21 8
22 12
23 4
24 8
25 24
26 24
27 2
28 20
29 15
30 2
31 14
32 9
DEPOT_SECTION
 1
 -1
EOF
```

我们运行上述代码，结果如下。

Result

```
The objective value is:786.7
[0-6-3-2-23-4-11-28-14-0]
[0-12-1-16-30-0]
[0-18-8-9-22-15-29-10-25-5-20-0]
[0-21-31-19-17-13-7-26-0]
[0-24-27-0]

```

```
8 duration time = 16.0399973s
```

第 12 章　拉格朗日松弛

Lagrangian Relaxation(拉格朗日松弛) 是一个非常重要的算法。该算法在运筹优化领域通常与 Branch and Bound、Branch and Cut、Branch and Price 等联合使用，一般用来为算法提供较好的 Upper Bound (上界) 或者 Lower Bound (下界)，以加快算法收敛。本章我们首先介绍最优性 (Optimality) 和松弛 (Relaxation) 的相关理论，然后详细介绍 Lagrangian Relaxation 的原理、伪代码和代码实现。本章的大部分内容均参考自文献Wolsey 1998第 2 章和第 10 章，感兴趣的读者可以移步相应章节阅读完整内容。另外，本章中涉及较多的定理，为了提高可读性，我们仅给出了部分定理的证明。

12.1 最优性和松弛

12.2 对偶

12.3 拉格朗日松弛

12.4 拉格朗日对偶的加强

12.5 求解拉格朗日对偶

12.6 如何选择拉格朗日松弛

12.7 Python 调用 Gurobi 实现拉格朗日求解选址-运输问题

12.7.1 拉格朗日松弛应用案例：选址-运输问题

12.7.2 Python 代码实现：版本 1

下面是 Python 调用 Gurobi 实现拉格朗日松弛算法求解 LTP 的完整代码。

本代码参考自杉数科技的算法工程伍健在 Github 上公开的代码[1]。在这里，本书作者对其进行了微小调整。

数据类型和读取数据

ReadData

```
from __future__ import division, print_function

from gurobipy import *
import copy
import pandas as pd

class Data:
    facilityNumLimit = 0
    customerNum = 0
    supply = []
    demand = []
    travelCost = []
    def __init__(self):
        # initialize data
        self.facilityNumLimit = 0
        self.customerNum = 0
        self.supply = []
        self.demand = []
        self.travelCost = []


import re
def readData(data, filename):
    f = open(filename, 'r')
    lines = f.readlines()
    cnt = 0
    for line in lines:
        cnt += 1
        if(cnt == 1):
            data.facilityNumLimit = int(line)
        if(cnt == 2):
            data.customerNum = int(line)
        if(cnt == 3):
            line = line[:-1]   # 去除换行符
            array = re.split(r" +", line)
            for i in range(data.customerNum):
                data.supply.append(float(array[i]))
        if(cnt == 4):
            line = line[:-1]   # 去除换行符
            array = re.split(r" +", line)
            for i in range(data.customerNum):
```

[1]Github 链接：https://github.com/wujianjack/optimizationmodels/tree/master/gurobi。

```
                data.demand.append(float(array[i]))
        if(cnt >= 5 and cnt <= data.customerNum + 4):
            line = line[:-1]   # 去除换行符
            array = re.split(r" +", line)
            temp_cost = []
            for j in range(data.customerNum):
                temp_cost.append(float(array[j]))
            data.travelCost.append(temp_cost)
    return data
```

初始化模型

creatModel

```
def creatModel(data, var_x, var_y, relaxedCons):
    try:
        LTP_model = Model('Location Transport Problem')

        # close output log
        LTP_model.setParam("OutputFlag", 0)

        # creat Location Transport Problem  model
        for i in range(data.customerNum):
            var_x_temp = []
            for j in range(data.customerNum):
                var_x_temp.append(LTP_model.addVar(lb = 0.0, ub = data.demand[j], obj = 0.0, vtype = GRB.INTEGER))
            var_x.append(var_x_temp)

        for i in range(data.customerNum):
            var_y.append(LTP_model.addVar(lb = 0.0, ub = 1.0, obj = 0.0, vtype = GRB.BINARY))

        # logic constraints
        for i in range(data.customerNum):
            LTP_model.addConstr(quicksum(var_x[i][j] for j in range(data.customerNum)) <= data.supply[i] *
            ↪  var_y[i])

        # logic constraints
        LTP_model.addConstr(quicksum(var_y[i] for i in range(data.customerNum)) <= data.facilityNumLimit)

        # meet the customer's demand constraints
        for j in range(data.customerNum):
            relaxedCons.append(LTP_model.addConstr(quicksum(var_x[i][j] for i in range(data.customerNum)) >=
            ↪  data.demand[j]))

        LTP_model.setObjective(quicksum(var_x[i][j] * data.travelCost[i][j] for i in range(data.customerNum) for j
        ↪  in range(data.customerNum)), GRB.MINIMIZE)

        # update model if necessary
        LTP_model.update()

        return LTP_model, var_x, var_y, relaxedCons

    except GurobiError as e:
        print('Error code' + str(e.errno) + ': ' + str(e))
    except AttributeError as e:
        print('Encountered an attribute error: ' + str(e))append(temp_cost)
```

拉格朗日松弛：次梯度算法

subGradient

```python
def subGradientSolve(data, LTP_model, var_x, var_y, relaxedCons, maxIter, noChangeCntLimit, LBlog, UBlog,thetaLog, stepSizeLog):

    # set parameters of Lagrangian Relaxation algorithm
    noChangeCnt = 0
    squareSum = 0.0
    stepSize = 0.0
    theta = 2.0
    LB = 0.0
    UB = 0.0
    Lag_multiplier = [0.0] * data.customerNum
    slack = [0.0] * data.customerNum

    # initial lower bound LB (via LP relaxation)
#     LB = relaxUB(LTP_model)
    LTP_model_copy = LTP_model.copy()
    relaxed_LTP_model = LTP_model_copy.relax()
    relaxed_LTP_model.optimize()
    LB = relaxed_LTP_model.objval
    print('LB:', LB)

    # initial UB (via sum all max travelCost)
    for i in range(data.customerNum):
        UB += max(data.travelCost[i])
    print('UB:', UB)

    # temporary linear expression
    obj_totalTravelCost = quicksum(var_x[i][j] * data.travelCost[i][j] for i in range(data.customerNum) for j in range(data.customerNum))

    # indidate that whether the current model is lagrangian relaxation version
    isModelLagrangianRelaxed = False

    # main 'Lagrangian Relaxation' loop
    for iter in range(maxIter):
        # solve lower bound
        if(isModelLagrangianRelaxed == False):
            isModelLagrangianRelaxed = True

            relaxedConsNum = len(relaxedCons)
            for i in range(relaxedConsNum):
                LTP_model.remove(relaxedCons[i])
            relaxedCons = []

        # lagrangian relaxation term :  ∑_{j∈C} μ_j (d_j − ∑_{i∈D} x_ij)
        obj_lagrangian_relaxed_term = quicksum(Lag_multiplier[j] * (data.demand[j] -
                                  quicksum(var_x[i][j] for i in range(data.customerNum)))
                                  for j in range(data.customerNum))

        # lagrangian relaxation objective :  ∑_{i∈D} ∑_{j∈C} c_ij x_ij + ∑_{j∈C} μ_j (d_j − ∑_{i∈D} x_ij)
        LTP_model.setObjective(obj_totalTravelCost + obj_lagrangian_relaxed_term, GRB.MINIMIZE)

        # solve relaxed model and obtain lower bound
        LTP_model.update()
        LTP_model.optimize()
        print('LTP_model.objval:', LTP_model.objval)
```

```
55
56          # calculate slacks for each relaxed constraints by the solution x
57          for j in range(data.customerNum):
58              # slacks for each relaxed constraints :  ∑_{i∈D} x_ij − d_j
59              slack[j] = sum(var_x[i][j].x for i in range(data.customerNum)) - data.demand[j]
60          print(slack)
61          # update lower bound if there has any improvement
62          if(LTP_model.objval > LB + 1e-6):
63              LB = LTP_model.objval
64              noChangeCnt = 0
65          else:
66              noChangeCnt += 1
67
68          # update scale theta if theta does not change for a number of iterations(noChangeCntLimit)
69          if(noChangeCnt == noChangeCntLimit):
70              theta = theta / 2.0
71              noChangeCnt = 0
72
73          # calculate '2-norm'
74          squareSum = sum(slack[i]**2.0 for i in range(data.customerNum))
75  #          if(squareSum == 0):
76  #              squareSum = 1
77
78          # update step size
79          stepSize = theta * (UB - LTP_model.objval) / squareSum
80
81          # update lagrangian multipliers with update equations
82          for i in range(data.customerNum):
83              if(Lag_multiplier[i] > stepSize * slack[i]):
84                  Lag_multiplier[i] = Lag_multiplier[i] - stepSize * slack[i]
85              else:
86                  Lag_multiplier[i] = 0.0
87
88          # get an upper bound of original model
89          '''
90          we relax the demand constraints, thus the relaxed model may select more facility so that the supply
91          will ecxceed the demand
92          '''
93          selected_facility_supply = sum(data.supply[i] * var_y[i].x for i in range(data.customerNum))
94          demand_sum_all = sum(data.demand)
95
96          print('selected_facility_supply = ', selected_facility_supply)
97          print('demand_sum_all = ', demand_sum_all)
98          if(selected_facility_supply - demand_sum_all >= 1e-6):
99              isModelLagrangianRelaxed = False
100
101             # add relaxed constraints into LTP model and fix y, so that the model is easy to solve
102             # this LTP model is same as the original model
103             # but with fixed y
104             # in lagrangian relaxation version, we relax these constraints
105             # in this version, we add them back, aiming to obtain an UB
106             for j in range(data.customerNum):
107                 relaxedCons.append(LTP_model.addConstr(quicksum(var_x[i][j]
108                                                                 for i in range(data.customerNum)) >= data.demand[j]))
109
110             # retrieve solution from LB model and fix it
111             '''
112             fix facility location variable and get an upper bound of original model
113             fix the value of y (via revise the lb and ub)
```

```
            '''
            for i in range(data.customerNum):
                var_y[i].lb = var_y[i].x
                var_y[i].ub = var_y[i].x

            LTP_model.setObjective(obj_totalTravelCost, GRB.MINIMIZE)

            # solve the revised model with fixed y and get an upper bound
            LTP_model.update()
            LTP_model.optimize()

            # update UB
            UB = min(UB, LTP_model.objval)

            # reset the facility location variable y's bound to 0-1
            for i in range(data.customerNum):
                var_y[i].lb = 0.0
                var_y[i].ub = 1.0

            LTP_model.update()

        # update 'LBlog', 'UBlog', 'stepSizelog', 'thetalog'
        LBlog.append(LB)
        UBlog.append(UB)
        stepSizeLog.append(stepSize)
        thetaLog.append(theta)

    # report the information
    print("\n   --------------  Iteration log information --------------    \n")
    print("  Iter        LB              UB          theta        stepSize")

    for i in range(len(LBlog)):
        print("  %3d    %12.6f    %12.6f    %8.6f    %8.6f" \
              % (i, LBlog[i], UBlog[i], thetaLog[i], stepSizeLog[i]))
```

算例格式

Location Transport Problem 算例格式

```
8
30
19 13 20 18 23 25 22 28 18 16 17 14 22 13 18 20 15 23 25 19 18 19 22 22 21 19 17 21 15 25
4 1 2 7 9 9 3 9 5 2 4 1 9 3 1 7 5 9 2 4 9 6 5 4 5 6 8 9 5 6
0 61 35 35 53 16 11 58 28 34 43 18 48 30 68 15 27 53 26 32 13 41 54 36 51 68
↪ 29 20 49 48
61 0 96 88 29 77 50 92 57 75 48 47 76 32 77 74 34 75 87 68 70 38 8 87 71 82
↪ 39 78 104 106
35 96 0 18 88 21 45 72 58 53 65 49 51 66 93 22 61 76 14 40 33 76 88 43 76 91
↪ 64 17 43 33
35 88 18 0 87 27 42 84 62 62 51 42 33 62 100 20 55 85 26 25 39 76 81 55 83 100
↪ 63 16 59 51
53 29 88 87 0 67 45 66 37 54 62 47 85 26 49 68 35 48 77 73 57 14 27 67 44 54
↪ 24 73 85 90
16 77 21 27 67 0 27 57 36 34 55 32 51 46 73 10 43 57 10 37 12 54 69 29 57 73
↪ 43 12 38 34
11 50 45 42 45 27 0 64 29 40 35 8 46 20 68 24 16 55 37 31 23 35 43 44 52 69
↪ 22 29 60 59
58 92 72 84 66 57 64 0 37 24 99 72 106 65 31 67 72 19 60 90 47 54 87 30 23 26
↪ 56 69 38 49
28 57 58 62 37 36 29 37 0 18 62 36 74 28 40 41 34 26 45 59 24 23 51 30 24 41
↪ 19 46 48 54
```

```
14  34  75  53  62  54  34  40  24  18  0  75  48  81  45  41  43  50  24  39  66  23  40  69  15  24  39
    ↪ 37  46  32  39
15  43  48  65  51  62  55  35  99  62  75  0  27  29  39  99  46  30  88  63  27  57  59  41  79  85  101
    ↪ 48  49  92  89
16  18  47  49  42  47  32  8  72  36  48  27  0  40  21  74  27  13  62  42  27  31  39  39  52  60  77
    ↪ 26  31  67  65
17  48  76  51  33  85  51  46  106  74  81  29  40  0  60  114  41  51  100  55  15  59  79  69  80  98  115
    ↪ 66  40  89  82
18  30  32  66  62  26  46  20  65  28  45  39  21  60  0  60  44  10  51  56  48  38  19  25  55  48  63
    ↪ 10  49  72  74
19  68  77  93  100  49  73  68  31  40  41  99  74  114  60  0  80  69  17  79  99  61  42  75  53  17  6
    ↪ 51  85  67  77
20  15  74  22  20  68  10  24  67  41  43  46  27  41  44  80  0  39  65  17  26  20  56  66  39  63  80
    ↪ 44  5  48  43
21  27  34  61  55  35  43  16  72  34  50  30  13  51  10  69  39  0  59  53  39  38  29  27  58  56  72
    ↪ 18  44  74  74
22  53  75  76  85  48  57  55  19  26  24  88  62  100  51  17  65  59  0  63  85  45  37  70  37  4  15
    ↪ 42  69  51  60
23  26  87  14  26  77  10  37  60  45  39  63  42  55  56  79  17  53  63  0  42  21  64  80  30  63  78
    ↪ 53  15  34  27
24  32  68  40  25  73  37  31  90  59  66  27  27  15  48  99  26  39  85  42  0  44  65  61  65  83  100
    ↪ 52  27  75  69
25  13  70  33  39  57  12  23  47  24  23  57  31  59  38  61  20  38  45  21  44  0  44  63  22  44  61
    ↪ 34  24  36  36
26  41  38  76  76  14  54  35  54  23  40  59  39  79  19  42  56  29  37  64  65  44  0  34  53  33  46
    ↪ 13  61  71  76
27  54  8  88  81  27  69  43  87  51  69  41  39  69  25  75  66  27  70  80  61  63  34  0  80  67  79
    ↪ 32  71  97  99
28  36  87  43  55  67  29  44  30  30  15  79  52  80  55  53  39  58  37  30  65  22  53  80  0  38  51
    ↪ 48  41  18  25
29  51  71  76  83  44  57  52  23  24  24  85  60  98  48  17  63  56  4  63  83  44  33  67  38  0  17
    ↪ 38  68  53  62
30  68  82  91  100  54  73  69  26  41  39  101  77  115  63  6  80  72  15  78  100  61  46  79  51  17  0
    ↪ 54  84  63  73
31  29  39  64  63  24  43  22  56  19  37  48  26  66  10  51  44  18  42  53  52  34  13  32  48  38  54
    ↪ 0  49  66  69
32  20  78  17  16  73  12  29  69  46  46  49  31  40  49  85  5  44  69  15  27  24  61  71  41  68  84
    ↪ 49  0  48  42
33  49  104  43  59  85  38  60  38  48  32  92  67  89  72  67  48  74  51  34  75  36  71  97  18  53  63
    ↪ 66  48  0  12
34  48  106  33  51  90  34  59  49  54  39  89  65  82  74  77  43  74  60  27  69  36  76  99  25  62  73
    ↪ 69  42  12  0
```

注意，在算例文件 `location_transport_instance.txt` 中：

- 第一行为可选设施的数量上限；
- 第二行为客户个数；
- 第三行为 30 个候选设施 (配送中心或者仓库) 的供应量；
- 第四行为 30 个客户点的需求量；
- 第 5-34 行为配送费用矩阵 $c_{ij}, \forall i \in D, \forall j \in C$。

算例测试

test instance

```
1  data = Data()
2  data = readData(data, 'location_transport_instance.txt')
3
4  # initialize parameters
5  maxIter = 200
6  noChangeCntLimit = 5
7  stepSizeLog = []
```

```
thetaLog = []
LBlog = []
UBlog = []

var_x = []
var_y = []
relaxedCons = []   # relaxed constraints

LTP_model, var_x, var_y, relaxedCons = creatModel(data, var_x, var_y, relaxedCons)

subGradientSolve(data, LTP_model, var_x, var_y, relaxedCons, maxIter, noChangeCntLimit, LBlog, UBlog,thetaLog,
↪  stepSizeLog)
```

运行结果如下

test instance

```
[Out]:

   --------------  Iteration log information --------------

  Iter        LB              UB          theta        stepSize
  ..........................................................
    8     1084.122883     2684.000000    1.000000    1.259523
    9     1084.122883     2684.000000    1.000000    0.679221
   10     1093.244186     2684.000000    1.000000    2.367196
   11     1093.244186     2684.000000    1.000000    1.279174
   12     1093.244186     2684.000000    1.000000    0.622132
   13     1258.238295     2684.000000    1.000000    1.087538
  ..........................................................
   20     1317.915751     2684.000000    0.500000    1.112446
   21     1317.915751     2684.000000    0.500000    1.062753
   22     1317.915751     2684.000000    0.500000    0.296870
   23     1465.210699     1592.000000    0.500000    2.199981
   24     1465.210699     1592.000000    0.500000    0.191505
   25     1465.210699     1592.000000    0.500000    0.150506
  ..........................................................
  177     1547.479347     1592.000000    0.000244    0.000111
  178     1547.479347     1592.000000    0.000244    0.000069
  179     1547.479347     1592.000000    0.000244    0.000061
  180     1547.479347     1592.000000    0.000244    0.000101
  181     1547.479347     1592.000000    0.000122    0.000049
  182     1547.480336     1592.000000    0.000122    0.000062
  ..........................................................
  193     1547.481684     1592.000000    0.000122    0.000051
  194     1547.481684     1592.000000    0.000122    0.000049
  195     1547.481941     1592.000000    0.000122    0.000020
  196     1547.482079     1592.000000    0.000122    0.000057
  197     1547.482079     1592.000000    0.000122    0.000055
  198     1547.482171     1592.000000    0.000122    0.000042
  199     1547.482171     1592.000000    0.000122    0.000019
```

12.7.3 Python 代码实现：版本 2

这个版本的代码是杉数科技的算法工程伍健在 Github 上公开的代码的原版。

locationtransport.py

locationtransport.py

```
from __future__ import division, print_function

import gurobipy as GRBPY


class LocationTransport:
    def __init__(self, name=None):
        # initialize data
        self.buildlimit = 0
        self.ncites = 0
        self.supply = []
        self.demand = []
        self.shipcost = []

        # initialize parameters
        self.iterlimit = 100
        self.samelimit = 3
        self.steplog = []
        self.scalelog = []
        self.xLBlog = []
        self.xUBlog = []

        if name is not None:
            self.name = name
        else:
            self.name = "demo"

        self.vship = []
        self.vbuild = []
        self.crelax = []

    def read(self, filename):
        with open(filename, "r") as data:
            self.buildlimit = int(data.readline())
            self.ncites = int(data.readline())

            column = data.readline().split()
            for i in range(self.ncites):
                self.supply.append(float(column[i]))

            column = data.readline().split()
            for i in range(self.ncites):
                self.demand.append(float(column[i]))

            for i in range(self.ncites):
                column = data.readline().split()
                lshipcost = []
                for j in range(self.ncites):
                    lshipcost.append(float(column[j]))
                self.shipcost.append(lshipcost)

    def build(self):
        try:
            self.mtrans = GRBPY.Model(self.name)

            # discard output information
```

```
            self.mtrans.setParam("OutputFlag", 0)

            # construct model
            for i in range(self.ncites):
                shipvar = []
                for j in range(self.ncites):
                    shipvar.append(self.mtrans.addVar(0.0, self.demand[j], 0.0, GRBPY.GRB.INTEGER))
                self.vship.append(shipvar)

            for i in range(self.ncites):
                self.vbuild.append(self.mtrans.addVar(0.0, 1.0, 0.0, GRBPY.GRB.BINARY))

            for i in range(self.ncites):
                self.mtrans.addConstr(GRBPY.quicksum(self.vship[i][j] for j in range(self.ncites)) \
                                      <= self.supply[i] * self.vbuild[i])

            self.mtrans.addConstr(GRBPY.quicksum(self.vbuild[i] for i in range(self.ncites)) \
                                  <= self.buildlimit)

            for j in range(self.ncites):
                self.crelax.append(self.mtrans.addConstr(GRBPY.quicksum(self.vship[i][j] \
                                   for i in range(self.ncites)) \
                                   >= self.demand[j]))

            self.mtrans.setObjective(GRBPY.quicksum(self.vship[i][j] * self.shipcost[i][j] \
                                     for i in range(self.ncites) \
                                     for j in range(self.ncites)), GRBPY.GRB.MINIMIZE)

            # update is necessary
            self.mtrans.update()
        except GRBPY.GurobiError as e:
            print('Error code' + str(e.errno) + ': ' + str(e))
        except AttributeError as e:
            print('Encountered an attribute error: ' + str(e))

    def solve(self):
        # 'Lagrangian Relaxation' parameters
        same = 0
        norm = 0.0
        step = 0.0
        scale = 1.0
        xLB = 0.0
        xUB = 0.0
        xlambda = [0.0] * self.ncites
        slack = [0.0] * self.ncites

        # build model
        self.build()

        # initial 'xLB'
        xLB = self.relaxUB(self.mtrans)

        # initial 'xUB'
        for i in range(self.ncites):
            xUB += max(self.shipcost[i])

        # temporary linear expression
        obj_shipcost = GRBPY.quicksum(self.vship[i][j] * self.shipcost[i][j] \
                                      for i in range(self.ncites) \
```

```
                                    for j in range(self.ncites))

        # sentinel flag
        lbmodel = 0

        # main 'Lagrangian Relaxation' loop
        for iter in range(self.iterlimit):
            # solve lower bound
            if lbmodel == 0:
                lbmodel = 1

                lenrelax = len(self.crelax)
                for i in range(lenrelax):
                    self.mtrans.remove(self.crelax[i])
                self.crelax = []

            obj_lagrangian = GRBPY.quicksum(xlambda[j] * (self.demand[j] - \
                                        GRBPY.quicksum(self.vship[i][j] for i in range(self.ncites))) \
                                        for j in range(self.ncites))
            self.mtrans.setObjective(obj_shipcost + obj_lagrangian, GRBPY.GRB.MINIMIZE)

            # 'LB' model
            self.mtrans.optimize()

            # calculate 'slack'
            for j in range(self.ncites):
                slack[j] = sum(self.vship[i][j].x for i in range(self.ncites)) - self.demand[j]

            # improve lower bound
            if self.mtrans.objval > xLB + 1e-6:
                xLB = self.mtrans.objval
                same = 0
            else:
                same += 1

            # update 'scale' if no improvement in 'samelimit' iteration
            if same == self.samelimit:
                scale /= 2.0
                same = 0

            # calculate 'norm'
            norm = sum(slack[i]**2.0 for i in range(self.ncites))

            # update 'step'
            step = scale * (xUB - self.mtrans.objval) / norm

            # update 'lambda'
            for i in range(self.ncites):
                if xlambda[i] > step * slack[i]:
                    xlambda[i] -= step * slack[i]
                else:
                    xlambda[i] = 0.0

            # solve upper bound
            sumsbval = sum(self.supply[i] * self.vbuild[i].x for i in range(self.ncites))
            sumdemand = sum(self.demand)

            if sumsbval - sumdemand >= 1e-6:
                lbmodel = 0
```

```
175
176                 for j in range(self.ncites):
177                     self.crelax.append(self.mtrans.addConstr(GRBPY.quicksum(self.vship[i][j] \
178                                                             for i in range(self.ncites)) >= self.demand[j]))
179
180                 # retrieve solution from LB model and fix it
181                 for i in range(self.ncites):
182                     self.vbuild[i].lb = self.vbuild[i].x
183                     self.vbuild[i].ub = self.vbuild[i].x
184
185                 self.mtrans.setObjective(obj_shipcost, GRBPY.GRB.MINIMIZE)
186
187                 self.mtrans.optimize()
188
189                 xUB = min(xUB, self.mtrans.objval)
190
191                 # reset to initial bound
192                 for i in range(self.ncites):
193                     self.vbuild[i].lb = 0.0
194                     self.vbuild[i].ub = 1.0
195
196             # update 'xLBlog', 'xUBlog', 'steplog', 'scalelog'
197             self.xLBlog.append(xLB)
198             self.xUBlog.append(xUB)
199             self.steplog.append(step)
200             self.scalelog.append(scale)
201
202     def relaxUB(self, mtrans):
203         mrelax = mtrans.relax()
204
205         mrelax.optimize()
206
207         return mrelax.objval
208
209     def report(self):
210         print("\n               *** Summary Report ***               \n")
211         print("  Iter        LB              UB          scale        step")
212
213         for i in range(len(self.xLBlog)):
214             print("  %3d    %12.6f    %12.6f    %8.6f    %8.6f" \
215                   % (i, self.xLBlog[i], self.xUBlog[i], self.scalelog[i], self.steplog[i]))
216
217 if __name__ == "__main__":
218     loctrans = LocationTransport()
219     loctrans.read("loctrans.dat")
220     loctrans.solve()
221     loctrans.report()
```

第 13 章　列生成算法

Column Generation Algorithm (列生成算法) 是混合整数规划中一个非常强大的精确算法。该算法由 Gilmore 和 Gomory 于 1961 年在研究下料问题 (Cutting Stock Problem) 的文章中首次提出 (Gilmore and Gomory 1961)，其基本原理与 Simplex Algorithm 迭代过程中选择入基变量的原理基本相同。Column Generation 也经常与之前章节介绍过的 Branch and Bound，还有之后章节将要介绍的 Dantzig-Wolfe Decomposition(DW 分解算法) 组合使用，即所谓的 Branch and Price(分支定价算法)。本章我们就来详细介绍 Column Generation Algorithm。

13.1 为什么用列生成算法

13.2 下料问题

13.3 列生成求解下料问题的实现

13.3.1 Python 调用 Gurobi 实现列生成求解下料问题示例算例

13.3.2 Python 调用 Gurobi 实现列生成求解下料问题示例算例 (以人工变量为初始列的方式)

13.3.3 Python 调用 Gurobi 实现列生成求解下料问题: 版本 3

13.3.4 Java 调用 CPLEX 实现列生成求解下料问题：官方文档示例代码解读

Java 调用 CPLEX 实现 Column Generation 求解 Cutting Stock Problem 在 CPLEX 提供的官方文档中是有示例代码的，在此，我们对其加以注释和解读。

下面是完整代码。

Data 类

Data.java

```java
package ColumnGeneration_Hsinglu;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

public class Data {
    double rollwidth;
    double[] size;
    double[] amount;

    public static Data readData(String path) throws Exception{
        Data data = new Data();
        BufferedReader br = new BufferedReader(new FileReader(path));
        String line;
```

```
        int count = 1;
        while((line = br.readLine()) != null){
//            line.trim();
            if(count == 1){
                String[] str = line.split("\\s+");
                //System.out.println(Integer.parseInt(str[0]));
                data.rollwidth = Integer.parseInt(str[0]);
                count += 1;
                continue;
            }else if(count == 2){
                String[] str = line.split("\\s+");
                data.size = new double[str.length];
                for(int i = 0; i < str.length; i++){
                    data.size[i] = Integer.parseInt(str[i]);
                }
                count += 1;
                continue;
            }else if(count == 3){
                String[] str = line.split("\\s+");
                data.amount = new double[str.length];
                for(int i = 0; i < str.length; i++){
                    data.amount[i] = Integer.parseInt(str[i]);
                }
                continue;
            }
        }
        br.close();
        return data;
    }

    public static void printData(Data data){
        System.out.println("rollwidth:\t" + data.rollwidth);
        System.out.print("size:\t\t");
        for(int i = 0; i < data.size.length; i++){
            System.out.print(data.size[i] + "\t");
        }
        System.out.print("\namount:\t\t");
        for(int i = 0; i < data.amount.length; i++){
            System.out.print(data.amount[i] + "\t");
        }
        System.out.println();
    }
}
```

Cutstock 类

Cutstock.java

```
package ColumnGeneration_Hsinglu;

/**
 * @author:  Hsinglu Liu
 * @School: Tsinghua University
 * @ 操作说明: 修改读取文件的文件名，求解 cutstock1 和 cutstock2 两个算例，
 *          cutstock1 来自于 CPLEX12.6 版本的附带例子
 *          cutstock2 来自于 CPLEX12.8 版本的附带例子
 * 算例来源: CPLEX 官方文档
 *
 */
```

```
12
13  import ilog.concert.IloColumn;
14  import ilog.concert.IloCopyManager;
15  import ilog.concert.IloCopyable;
16  import ilog.concert.IloException;
17  import ilog.concert.IloNumExpr;
18  import ilog.concert.IloNumVar;
19  import ilog.concert.IloNumVarType;
20  import ilog.concert.IloObjective;
21  import ilog.concert.IloRange;
22  import ilog.concert.IloCopyManager.Check;
23  import ilog.cplex.IloCplex;
24
25  public class Cutstock {
26
27      // 定义容差
28      static double eps = 1.0e-6;
29
30      // 首先定义初始数据
31      static double rollwidth;    // 棒材长度            17
32      static double[] size;       // 不同需求的长度数组    3,5,9
33      static double[] amount;       // 不同需求的数量        25,20,15
34
35      public static void main(String[] args) throws Exception {
36          // 获取文件路径的字符串
37          String path = "E:\\MyCode\\JavaCode\\JavaCallCplex\\"
38                  + "src\\ColumnGeneration_Hsinglu\\cutstock.txt";
39
40          // 读取数据
41          Data data = Data.readData(path);
42          rollwidth = data.rollwidth;
43          size = data.size;
44          amount = data.amount;
45
46          data.printData(data);
47
48          // =============== 创建 Cplex 模型对象 ======================
49          IloCplex masterProblem = new IloCplex();
50
51          // =========== 建立 masterProblem 的目标函数 ======================
52          /*
53           * Interface IloObjective: 这是一个接口
54           * IloObjective:   An objective function is defined by an objective expression
55           *                   and an optimization sense.
56           *
57           * ---------IloModeler 下的方法  -----------
58           * 返回值类型                |   方法名                                          | 描述
59           * IloObjective      addMinimize()           Creates and returns an empty minimization objective
60           *                                           function and adds it to the invoking model.
61           * IloObjective      addMinimize(java.lang.String name)
62           *                                           Creates and returns an empty minimization objective
63           *                                           function with the specified name and adds the
64           *                                           empty objective to the invoking model.
65           *
66           */
67          IloObjective RollsUsed = masterProblem.addMinimize();
68
69          // ================ 加入模型的约束 ============================
70          /*
```

```
71          * 由于问题的特殊性，下料问题的约束个数等于不同长度需求的个数。本问题中为三个约束
72          *   min  z  =   x1   +   x2   +   x3
73          *            5*x1                    >=  25  (3-ft 的需求)
74          *                    3*x2            >=  20  (5-ft 的需求)
75          *                            x3      >=  15  (9-ft 的需求)
76          *
77          *   由于约束的个数是一定的，也就是行数一定，因此可以用加入矩阵的方式来加入约束。
78          *   这里用 IloRange 这个接口（类）
79          *   Interface IloRange
80          *       : This is the interface for modeling objects representing
81          *       ranged constraints of the format:   lb <= expr <= ub.
82          *     ** lb and ub are double values, referred to as the
83          *        lower bound and upper bound of the ranged constraint,
84          *        and expr is an expression. Values +- infinity can be used
85          *        as bounds. This allows you to use IloRange objects to represent
86          *        more commonly used constraints:
87          *
88          *     ## for expr == rhs, set lb = ub = rhs
89          *     ## for expr <= rhs, set lb = -infinity and ub = rhs
90          *     ## for expr >= rhs, set lb = rhs and ub = infinity
91          *
92          *   //一些常用的方法
93          *   IloModeler.addRange(double, IloNumExpr, double): 即 lb <= expr <= ub
94          *   IloModeler.addEq(IloNumExpr, double)             : 即 expr = rhs
95          *   IloModeler.addGe(IloNumExpr, double)             : 即 expr >= rhs
96          *   IloModeler.addLe(IloNumExpr, double)           : 即 expr <= rhs
97          *
98          *   以及 IloCplex 类中的方法:
99          *
100         *   返回值类型  |   方法名                                        | 描述
101         *   IloRange     addCut(IloRange cut):   Adds the constraint cut as a cut to the
102         *                                          invoking IloCplex object.
103         *   IloRange[]   addCuts(IloRange[] cut): Adds the constraints given in cut as
104         *                                         cuts to the invoking IloCplex object.
105         *
106         *   //-----------下面两个方法是 IloModeler 下的方法----------------
107         *
108         *   返回值类型  |   方法名                                        | 描述
109         *   IloRange     addRange(double lb, IloNumExpr expr, double ub):  // 相当于一个一维数组
110         *                                  ** Creates and returns an instance of IloRange initialized to
111         *                                  represent the constraint lb <= expr <= ub and added to
112         *                                  the invoking instance of IloModeler.
113         *
114         *   IloRange[]   addRange(double lb, IloNumExpr expr, double ub, java.lang.String name):  // 相当于二维数组
115         *                                    **Creates and returns an instance of IloRange initialized to
116         *                                 represent the constraint lb <= expr <= ub and added to the
117         *                                 invoking instance of IloModeler.
118         *
119         *
120         *   //-----------下面两个方法是 Interface IloMPModeler 下的方法----------------
121         *
122         *   返回值类型       |   方法名                                                  | 描述
123         *   IloRange   addRange(double lb, double ub):   Creates, returns, and adds to the
124         *                                              invoking model an empty IloRange object.
125         *
126         *   IloRange   addRange(double lb, double ub, java.lang.String name):
127         *                                          Creates, returns, and adds to the invoking model
128         *                                          an empty IloRange object with the specified name
129         *                                          and upper and lower bounds.
```

```
130          *
131          */
132         IloRange[] Fill = new IloRange[data.amount.length]; // 首先构造对应于约束的 IloRang 类型数组，也就是 3 个约束，
        ↪  对应 3 行
133         for(int f = 0; f < amount.length; f++){
134             /*
135              * 下面是给出 IloRange 的上下界，但是内容是空的对象
136              * 返回一个空的 IloRange 对象，并将其加入到 IloCplex 模型中去
137              * addRange 方法会创建并返回一个 IloRange 实例，并将其加入到 masterProblem 中去，当做约束
138              */
139             Fill[f] = masterProblem.addRange(amount[f], Double.MAX_VALUE); // 25 <= Fill[f] <= inf
140         }
141 
142         //================ 下面来初始化模型 ==================
143         /*
144          * 这里有一个比较关键的问题，就是列生成的问题的变量个数是动态变化的，会随着迭代次数的增加而增加
145          * 但是在基本的调用过程当中，我们是给定了长度的调用，比如说：
146          *         IloNumVar[][] S = new IloNumVar[instance.nodeNum][instance.vehicleNum];
147          * 因此，为了适应长度变化的需求，我们需要定义一个类，来实现动态改变决策变量长度的功能。
148          * 这个类就是： IloNumVarArray （当然这个类也可以不写）
149          * IloNumVarArray 类完全是为了将新添加的决策变量放到一个长度可变化的数组里面。这个功能也可以用 HashMap 等数据结
    ↪  构来实现。
150          */
151         int nWidth = size.length;
152         IloNumVarArray Cut = new IloNumVarArray();
153 
154         // 列生成，首先要初始化模型，将模型中加入若干初始列，然后再循环添加新生成的列。
155         // 下面的循环语句就是现在模型中加入 3 列，以完成模型的初始化
156         for(int i = 0; i < nWidth; i++){
157             /*
158              * ---------Interface IloMPModeler 下的几个常用方法---------
159              * 返回值类型        |   方法名
    ↪  | 描述
160              * IloColumn    column(IloObjective obj, double val)
161              *                                 Creates an IloColumn object suitable for adding a new
162              *                                 variable to the objective obj as a linear term with
163              *                                 coefficient val.
164              * IloColumn    column(IloRange rng, double val)
165              *                                 Creates an IloColumn object suitable for adding a new
166              *                                 variable to constraint rng as a linear term with
167              *                                 coefficient val.
168              *
169              *
170              *  --------------Class IloColumn 的常用方法-------------------
171              * 返回值类型        |   方法名                                                    | 描述
172              * IloColumn     and(IloColumn column)        Links two column objects.
173              *
174              *  --------------Interface IloMPModeler 的常用方法-------------------
175              * 返回值类型        |   方法名                                                    | 描述
176              * IloNumVar     numVar(IloColumn column, double lb, double ub)
177              *                                         Creates a continuous modeling variable, of type
178              *                                         Float with upper bound and lower bound as
179              *                                         specified, for column-wise modeling.
180              *
181              * IloNumVar    numVar(IloColumn column, double lb, double ub, IloNumVarType type)
182              *                                         Creates and returns a new modeling variable
183              *                                         for column-wise modeling.
184              */
185 
```

```
            IloColumn col_1 = masterProblem.column(RollsUsed, 1.0);   // 目标函数中的系数
            IloColumn col_2 = masterProblem.column(Fill[i], Math.floor(rollwidth/size[i])); // 约束中的系数
            IloColumn col = col_1.and(col_2);  // 拼成一列

            // 创造基于这一列的决策变量，并将这一列加入到主问题中，以达到初始化主问题模型的目的
            IloNumVar ivar = masterProblem.numVar(col, 0, Double.MAX_VALUE);

            Cut.add(ivar);  // 这一步只是把新加入的列对应的决策变量放入到 Cut 这个容器中，方便输出求解结果

        }
        masterProblem.solve();

        // ===== 构建子问题 (pattern generation problem) ============
        IloCplex subProblem = new IloCplex();

        // 创建子问题的目标函数
        /*
         * 子问题; min : 1 - (1/5)*a3 - (1/3)*a5 - a9 (这个就对应 Reduced Cost)
         */
        IloObjective ReduceCost = subProblem.addMinimize();

        // -----定义子问题的决策变量 (均为整数)--------
        // 详见 numVarArray 方法的解释。该函数功能为：创建一个长度为 nWidth，上下界分别为 0，inf 的 int 型变量
        IloNumVar[] Use = subProblem.numVarArray(nWidth, 0, Double.MAX_VALUE, IloNumVarType.Int);

        // -----加入子问题的约束条件----------------
        /*
         * -Double.MAX_VALUE<patSolver.scalProd(_size, Use)<= _rollWidth
         * 即：
         *         -inf <= 3 * a3 + 5 * a5 + 9 * a9 <= 17，这就是子问题的约束条件
         */
        subProblem.addRange(-Double.MAX_VALUE, subProblem.scalProd(size, Use), rollwidth);

        // ========== 下面进行列生成的全部过程 ============================
        // 首先存储新的切割的类型（便于更新主问题，给决策变量赋系数）
        double[] newPatt = new double[nWidth];
        for(;;){
            // ---  首先求解主问题 --------
            masterProblem.solve();

            // 输出共切了多少根 17 英尺的木材以及每种切法用了多少木材
            report1(masterProblem, Cut, Fill);

            // ---- 获得主问题中各个约束的对偶变量（/也就是 Reduced Cost） -----
            double[] price = masterProblem.getDuals(Fill);

            // ---- 设置子问题的目标函数的系数 -----(diff 表示减法)
            ReduceCost.setExpr(subProblem.diff(1.0, subProblem.scalProd(price, Use)));

            // ---- 求解子问题 ------
            subProblem.solve();

            // 输出影子价格以及新的切法
            report2(subProblem, Use);

            // ---- 判断是否存在新的列可以加进来 ----（如果没有，这说明主问题已经是最终形式，直接跳出去求解最终的主问题）
            if(subProblem.getObjValue() > -eps){
                break;
            }
```

```
245
246             // 得到新的切割类型的数据
247             newPatt = subProblem.getValues(Use);
248
249             // ---- 更新主问题 -------
250             // ---首先创建列，并更新目标函数系数
251             IloColumn col = masterProblem.column(RollsUsed, 1.0);
252
253             // ---更新新列的约束系数，并将其与目标系数列连在一起-------
254             for(int i = 0; i < amount.length; i++){
255                 IloColumn col_1 = masterProblem.column(Fill[i], newPatt[i]);
256                 col = col.and(col_1);
257             }
258
259             // 将新的列加入到 masterProblem 中，并将其对应的决策变量加入存储所有决策变量的 Cut 中去
260             IloNumVar ivar = masterProblem.numVar(col, 0, Double.MAX_VALUE);
261             Cut.add(ivar);
262         }
263
264         // 将所有变量更新成为 int 型（之前为了获得新列，并没有将其设置成 int 型）
265         for(int i = 0; i < Cut.getSize(); i++){
266             masterProblem.add(masterProblem.conversion(Cut.getElement(i),  IloNumVarType.Int));
267         }
268
269         // 所有列均已生成并加入到最终模型中，求解最终模型
270         masterProblem.solve();
271
272         // 输出最优切法所需木材数以及每种切法所需木材数
273         report3(masterProblem, Cut);
274         System.out.println("Solution status: " + masterProblem.getStatus());
275
276         // 关闭资源，列生成到此结束
277          masterProblem.end();
278          subProblem.end();
279
280     }
281
282     // 统计变量值与变量数量（这个函数主要是为了输出求解结果时使用，不写这个函数也是可以的）
283     static class IloNumVarArray {
284         int _num = 0;   // _num 标识目前数组中有多少个决策变量
285         IloNumVar[] _array = new IloNumVar[32];
286
287         // 数组不够就增加成两倍长度
288         void add(IloNumVar ivar) {
289             if (_num >= _array.length) {
290                 IloNumVar[] array = new IloNumVar[2 * _array.length];
291                 System.arraycopy(_array, 0, array, 0, _num);
292                 _array = array;
293             }
294             _array[_num++] = ivar;
295         }
296
297         IloNumVar getElement(int i) {
298             return _array[i];
299         }
300
301         int getSize() {
302             return _num;   // 获得目前变量数组中有多少个决策变量
303         }
```

```
    }

    // 下面是输出求解结果的 3 个函数
    static void report1(IloCplex masterProblem, IloNumVarArray Cut, IloRange[] Fill)
            throws IloException {
        System.out.println();
        System.out.println("Using " + masterProblem.getObjValue() + " rolls");

        System.out.println();
        for (int j = 0; j < Cut.getSize(); j++) {
            System.out.println("  Cut" + j + " = "
                    + masterProblem.getValue(Cut.getElement(j)));
        }
        System.out.println();

        for (int i = 0; i < Fill.length; i++)
            System.out.println("  Fill" + i + " = "
                    + masterProblem.getDual(Fill[i]));
        System.out.println();
    }

    static void report2(IloCplex subProblem, IloNumVar[] Use)
            throws IloException {
        System.out.println();
        System.out.println("Reduced cost is " + subProblem.getObjValue());

        System.out.println();
        if (subProblem.getObjValue() <= -eps) {
            for (int i = 0; i < Use.length; i++)
                System.out.println("  Use" + i + " = "
                        + subProblem.getValue(Use[i]));
            System.out.println();
        }
    }

    static void report3(IloCplex masterProblem, IloNumVarArray Cut)
            throws IloException {
        System.out.println();
        System.out.println("Best integer solution uses "
                + masterProblem.getObjValue() + " rolls");
        System.out.println();
        for (int j = 0; j < Cut.getSize(); j++)
            System.out.println("  Cut" + j + " = "
                    + masterProblem.getValue(Cut.getElement(j)));
    }

}
```

算例 1

算例 1

```
17
[3,5,9]
[25,20,15]
```

算例 2

算例 2

```
5600
[1380,1520,1560,1710,1820,1880,1930,2000,2050,2100,2140,2150,2200]
[22,25,12,14,18,18,20,10,12,14,16,18,20]
```

13.4 列生成求解 TSP

13.4.1 TSP 的 1-tree 建模及列生成求解

13.4.2 主问题

13.4.3 子问题

13.5 列生成求解 VRPTW

13.5.1 主问题

13.5.2 子问题

13.5.3 详细案例演示

第 14 章　动态规划

Dynamic Programming(动态规划, DP) 是运筹学中的一种重要的最优化数学方法，主要用来求解多阶段决策问题。20 世纪 50 年代初，美国数学家 R.Bellman 等人在研究多阶段决策过程的优化问题时，提出了著名的最优化原理，从而创立了动态规划 (Bellman 1952, Bellman 1966)。动态规划算法可以非常高效的求解 SPPRC，从而加快 VRPTW 的求解。同时，也可以用于提高其他子问题为 SPPRC 的相关问题的求解效率。

14.1 动态规划

14.1.1 动态规划求解最短路问题

14.1.2 问题建模和求解

14.1.3 一个较大的例子

14.2 动态规划的实现

14.3 动态规划求解 TSP

14.4 标签算法求解带资源约束的最短路问题

14.5 Python 实现标签算法结合列生成求解 VRPTW

14.5.1 初始化 RMP

14.5.2 标签算法求解子问题

第 15 章　分支定价算法

之前我们介绍了 Column Generation Algorithm，这是一种强大的求解大规模整数规划的算法。但是，我们也指出了 Column Generation 的缺陷。就是我们在生成新列的过程中，暂时将 RMP 松弛成 LP(即 RLMP)，但是在最终形式的 RMP 中，我们将所有变量均设置成 $0-1$ 变量，然后进行求解。我们提到，求解整数规划版本的 RMP 得到的整数解，是原问题最优解的一个上界，但并不一定是原问题的最优解。因此仅仅用 Column Generation，并不能保证得到原问题的全局最优解。不过在下面这种情况下，是可以确定 Column Generation 的解同时也是全局最优解的，那就是当我们将最终形式 RMP 的整数约束去掉，将其松弛成 RLMP，如果 RLMP 的最优解同时也是整数解，则此时 Column Generation 得到的解就是全局最优解。

不幸的是，最终形式的 RMP 的线性松弛 RLMP 不总是存在整数最优解。那么，我们不禁要问，如何能够保证总是得到全局最优解呢？一个比较好的解决方法就是将 Column Generation Algorithm 与 Branch and Bound 嵌套在一起使用。当最终 RLMP 的解是小数时，我们对取值为小数的变量进行分支，然后在 BB tree 的每一个叶子节点处，在执行了分支操作的基础上，继续执行 Column Generation，得到新的最终 RMP，紧接着，再次求解这个新的 RMP 对应的 RLMP。我们不断地分支、更新 Upper Bound 和 Lower Bound，直到算法结束，最终得到的解一定是原问题的最优解。上面的思想，正是著名的 Branch and Price Algorithm (分支定价算法)。本章就来详细介绍该算法。

15.1 分支定价算法基本原理概述

15.2 分支定价算法求解 VRPTW

本小节我们提供一版用 Java 实现分支定价算法求解 VRPTW 的开源代码。完整代码可前往 https://github.com/dengfaheng/BPVRPTW下载。

第 16 章　Dantzig-Wolfe 分解算法

第 17 章　Benders 分解算法

Benders 分解算法是另外一种强大的算法，该算法由 Benders, JF 于 1962 年首次提出。区别于 Column Generation 的不断添加新列，Benders 分解是不断的添加新的行，是一种 Row Generation(行生成) 的方法。当然，在一些鲁棒优化问题中 (两阶段鲁棒优化)，还会有同时使用 Column Generation 和 Row Generation 的方法，叫做 Column and Constraint Generation(C&CG, 列与约束生成)。掌握了本章的读者，可以继续深入研究该算法。本章我们就来详细的介绍这种算法。我们首先介绍 Benders Decomposition 的基本原理，然后以一个非常具体的例子来帮助大家理解该算法的完整过程。本章内容主要参考文献Taşkin 2011和Kalvelagen 2002。

17.1 分解方法

17.2 详细案例

17.3 Benders 分解应用案例

17.4 Java 调用 CPLEX 实现 Benders 分解求解 FCTP

本节提供的 Java 版本的 Benders 分解代码来源于微信公众号” 数据魔术师”，代码原作者为黄楠博士，毕业于华中科技大学。对应的推文题目为《运筹学教学 |Benders Decomposition （二）应用实例及算法实现（附源代码及详细的代码注释）》[1]，为了提升代码的可读性，本书作者添加了大量的详细注释，并对代码结构做了相应的调整。

本书作者在微信公众号” 运小筹” 上也发布了原创的 Python 版本的 Benders 分解的相关代码，读者可以前往获取学习[2]。

本章提供的份代码包含三部分：

1. 直接调用 CPLEX 建立模型求解：`SolveMIPModel` 类；
2. 使用 Benders 分解算法求解模型，用 CPLEX 的 `LazyConstraintCallback` 实现：`BendersDecomposition_callback` 类；
3. 使用 Benders 分解算法求解模型，通过自己的方法实现：`BendersDecomposition` 类；

其他类文件的功能分别为:

- `Data` 类；存储测试算例数据；
- `Solution` 类；存储解的信息；
- `run_this` 类；运行算法，测试算例数据。

此外，我们提供了两个测试算例：test1 和 test2。

具体代码如下。

17.4.1 Data 类

Data.java

```java
package BendersDecomposition_FCTP;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

public class Data {
    int SourcesSize;    // 供应点数量
    int DemandsSize;    // 需求点数量
    double []supply;    // 供应量
    double []demand;    // 需求量
    double [][]c;        // 供应单位资源的成本
    double [][]fixed_c;    // 固定成本
    double [][] big_M;

    //读入数据
    public void read_data(String path) throws Exception {
        String line = null;
        String[] substr = null;
```

[1] https://mp.weixin.qq.com/s/gXMNReKgBY-hL-27bJeE_A
[2] https://mp.weixin.qq.com/s/7LpHBPvedDknWP7iRAS_zA

```
        Scanner cin = new Scanner(new BufferedReader(new FileReader(path)));
        for (int i = 0; i < 2; i++) {
            line = cin.nextLine();
        }
        line.trim();
        substr = line.split(("\\s+"));
        SourcesSize = Integer.parseInt(substr[0]);
        DemandsSize = Integer.parseInt(substr[1]);
        supply = new double[SourcesSize];
        demand = new double[DemandsSize];
        c = new double[SourcesSize][DemandsSize];
        fixed_c = new double[SourcesSize][DemandsSize];
        big_M = new double[SourcesSize][DemandsSize];
        for (int i = 0; i < 2; i++) {
            line = cin.nextLine();
        }
        line.trim();
        substr = line.split(("\\s+"));
        for (int i = 0; i < SourcesSize; i++) {
            supply[i] = Integer.parseInt(substr[i]);
        }
        for (int i = 0; i < 2; i++) {
            line = cin.nextLine();
        }
        line.trim();
        substr = line.split(("\\s+"));
        for (int i = 0; i < DemandsSize; i++) {
            demand[i] = Integer.parseInt(substr[i]);
        }
        line = cin.nextLine();
        for (int i = 0; i < SourcesSize; i++) {
            line = cin.nextLine();
            line.trim();
            substr = line.split(("\\s+"));
            for (int j = 0; j < DemandsSize; j++) {
                c[i][j] = Integer.parseInt(substr[j]);
            }
        }

        line = cin.nextLine();
        for (int i = 0; i < SourcesSize; i++) {
            line = cin.nextLine();
            line.trim();
            substr = line.split(("\\s+"));
            for (int j = 0; j < DemandsSize; j++) {
                fixed_c[i][j] = Integer.parseInt(substr[j]);
            }
        }
        cin.close();

        //设置 M 参数
         for (int i = 0; i < SourcesSize; i++) {
             for (int j = 0; j < DemandsSize; j++) {
             big_M[i][j] = Math.min(supply[i], demand[j]);
             }
         }
    }
}
```

17.4.2 Solution 类

Solution.java

```java
package BendersDecomposition_FCTP;

import ilog.cplex.IloCplex;

public class Solution {
    public double cost;          // 总成本
    public double[][] ship;      // 运载量
    public double[][] link_y;    // y
    public IloCplex.CplexStatus status;  // 模型的求解状态，即是否为最优，无界，无可行解等，是 CPLEX 的参数

    //输出 ship 的值
    public void print_ship() {
        for (int i = 0; i < ship.length; i++) {
            for (int j = 0; j < ship[i].length; j++) {
                System.out.printf("\t%d -> %d: %f", i, j, ship[i][j],"  ");
            }
            System.out.println();
        }
    }
}
```

17.4.3 BendersDecomposition_callback 类

BendersDecomposition_callback.java

```java
package BendersDecomposition_FCTP;

import java.util.Arrays;
import ilog.concert.*;
import ilog.cplex.*;

public class BendersDecomposition_callback {
    Data data;
    protected IloCplex subProblem;
    protected IloCplex masterProblem;
    IloObjective subObj;            //记录子问题目标函数
    IloLinearNumExpr subObj_expr;

    //对偶变量
    protected IloNumVar[] u;    // 资源约束的对偶变量
    protected IloNumVar[] v;    // 需求约束的对偶变量
    protected IloNumVar[][] w;    // x,y 约束的对偶变量
    double[] uSource;    //子问题中目标函数里对偶变量 u 对应系数
    double[] vDemand;    //子问题中目标函数里对偶变量 v 对应系数
    double[][] wM;        //子问题中目标函数里对偶变量 w 对应系数

    protected IloRange[][] subCon;    //子问题的约束方程
    double[][] xValue;                //记录原问题的 x 值

    protected IloNumVar subcost;    //子问题中的目标值，对应松弛的主问题模型中 q
    protected IloNumVar[][] y;        //主问题中的变量
    public static final double eps = 1.0e-7;

    int[][] yInit;                        //子问题中变量 y 初始值

    /**
```

```
32          * 构造函数
33          * @param d
34          */
35         public BendersDecomposition_callback(Data d) {
36             this.data = d;
37         }
38 
39         /**
40          * 置 1 函数：将数组中的数字全部设置为 1
41          */
42         void setOne(int[][] a) {
43             for (int i = 0; i < a.length; i++) {
44                 for (int j = 0; j < a[i].length; j++) {
45                     a[i][j] = 1;
46                 }
47             }
48         }
49 
50         /**
51          * 建立主问题和子问题的模型
52          * @throws IloException
53          */
54         public void buildBendersModels() throws IloException {
55 
56             subProblem = new IloCplex();         // 子问题
57             masterProblem = new IloCplex();         // 主问题
58             subProblem.setOut(null);             // 关闭子问题的求解日志
59             masterProblem.setOut(null);             // 关闭主问题的求解日志
60 
61             //参数初始化
62             yInit = new int[data.SourcesSize][data.DemandsSize];
63             setOne(yInit);             // 初始化参数 y=[1] ，主问题的目标值首先需要确定
64             u = new IloNumVar[data.SourcesSize];
65             v = new IloNumVar[data.DemandsSize];
66             w = new IloNumVar[data.SourcesSize][data.DemandsSize];
67             uSource = new double[data.SourcesSize];
68             vDemand = new double[data.DemandsSize];
69             wM = new double[data.SourcesSize][data.DemandsSize];
70             y = new IloNumVar[data.SourcesSize][data.DemandsSize];
71             subCon = new IloRange[data.SourcesSize][data.DemandsSize];
72             xValue = new double[data.SourcesSize][data.DemandsSize];
73 
74             /*
75              * 创建决策变量
76              */
77             subcost = masterProblem.numVar(0.0, Double.MAX_VALUE, IloNumVarType.Float, "subcost");
78             for (int i = 0; i < data.SourcesSize; i++) {
79                 u[i] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "u_" + i);
80             }
81             for (int i = 0; i < data.DemandsSize; i++) {
82                 v[i] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "v_" + i);
83             }
84             for (int i = 0; i < data.SourcesSize; i++) {
85                 for (int j = 0; j < data.DemandsSize; j++) {
86                     y[i][j] = masterProblem.numVar(0, 1, IloNumVarType.Int, "y_" + i + "_" + j);
87                     w[i][j] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "w_" + i + "_" + j);
88                 }
89             }
90 
```

```
91          /*
92           * 构建主问题的模型
93           */
94          IloNumExpr expr0 = masterProblem.numExpr();
95          for (int i = 0; i < data.SourcesSize; i++) {
96              for (int j = 0; j < data.DemandsSize; j++) {
97                  expr0 = masterProblem.sum(expr0, masterProblem.prod(data.fixed_c[i][j], y[i][j]));
98              }
99          }
100         masterProblem.addMinimize(masterProblem.sum(subcost, expr0), "TotalCost");
101
102         /*
103          * 什么是回调函数？这个偷来的解释讲的很清楚：
104          * 对普通函数的调用：调用程序发出对普通函数的调用后，程序执行立即转向被调用函数执行，
105          *      直到被调用函数执行完毕后，再返回调用程序继续执行。
106          *      从发出调用的程序的角度看，这个过程为“调用--> 等待被调用函数执行完毕--> 继续执行”
107          * 对回调函数调用：调用程序发出对回调函数的调用后，不等函数执行完毕，立即返回并继续执行。
108          *   这样，调用程序执和被调用函数同时在执行。
109          *   当被调函数执行完毕后，被调函数会反过来调用某个事先指定函数，以通知调用程序：函数调用结束。
110          *   这个过程称为回调 (Callback)，这正是回调函数名称的由来。
111          *   所以回调函数的特点就是占用时间比较长 (比如 I/O，http 请求等)，
112          *   使用回调函数就不需要等待回调函数的结果 (系统帮你去执行了)，程序直接执行你接下来的代码。
113          */
114
115         /* 回调函数
116          * public void use(IloCplex.Callback cb) throws IloException
117          *
118          *   Installs a user-written callback.
119          *      Callbacks are objects with a user-written method main that
120          *      are called regularly during the optimization of the active model.
121          *      This object must be implemented as a class derived from
122          *      a subclass of IloCplex.
123          *      Callback class, and the abstract method main must be
124          *      implemented for this class.
125          *
126          *      There are several places where the IloCplex algorithms
127          *      call a callback. IloCplex provides several different types
128          *      of callbacks, and each is implemented as a specific
129          *      subclass of IloCplex.Callback.
130          *
131          *      IloCplex can use only one callback of a given type at a time.
132          *      Thus, when calling method use several times with callbacks
133          *      of the same type, only the callback passed at the last call
134          *      of method use will be executed during the optimization.
135          *      However, callbacks of different types can be used simultaneously.
136          *
137          * Parameters:
138          *      cb - The callback to be used from now on.
139          *           The type of the callback object being passed
140          *           determines which callback is being installed.
141          *           If a callback of the same type has previously
142          *           been installed, the new callback will replace the old one.
143          */
144
145         // attach a Benders callback to the masterProblem
146         /*
147          * use 的过程中，BendersCallback 是一个回调函数，里面只有主函数，因此会直接运行主函数
148          * 这两个过程是同步的
149          */
```

```
150         masterProblem.use(new BendersCallback());
151
152         /*
153          * 构建子问题
154          */
155
156         // 子问题目标函数
157         subObj_expr = subProblem.linearNumExpr();
158         IloLinearNumExpr obj = subProblem.linearNumExpr();
159         for (int i = 0; i < data.SourcesSize; i++) {
160             uSource[i] = -data.supply[i];
161             obj.addTerm(uSource[i], u[i]);
162             subObj_expr.addTerm(uSource[i], u[i]);
163         }
164         for (int i = 0; i < data.DemandsSize; i++) {
165             vDemand[i] = data.demand[i];
166             obj.addTerm(vDemand[i], v[i]);
167             subObj_expr.addTerm(vDemand[i], v[i]);
168         }
169         for (int i = 0; i < data.SourcesSize; i++) {
170             for (int j = 0; j < data.DemandsSize; j++) {
171                 wM[i][j] = -data.big_M[i][j];
172                 obj.addTerm(wM[i][j] * yInit[i][j], w[i][j]);
173             }
174         }
175         subObj = subProblem.addMaximize(obj, "dualSubCost");
176
177         // 子问题约束
178         for (int i = 0; i < data.SourcesSize; i++) {
179             for (int j = 0; j < data.DemandsSize; j++) {
180                 IloNumExpr expr = subProblem.numExpr();
181                 IloNumExpr expr1 = subProblem.numExpr();
182                 expr = subProblem.sum(subProblem.prod(-1, u[i]), v[j]);
183                 expr1 = subProblem.sum(expr, subProblem.prod(-1, w[i][j]));
184                 subCon[i][j] = subProblem.addLe(expr1, data.c[i][j],"C" + i+ "_" +j);
185             }
186         }
187         // turn off the presolver for the main model
188         subProblem.setParam(IloCplex.BooleanParam.PreInd, false);
189         subProblem.setParam(IloCplex.Param.RootAlgorithm, IloCplex.Algorithm.Primal);
190     }
191
192     /*
193      * ===== LazyConstraintCallback == 的解释
194      * Callback class for lazy constraints.
195      * This is an advanced class
196      *
197      * This is the constructor for user-written lazy constraint callbacks.
198      */
199
200     /*
201      * benders decomposition 的实现：使用 LazyConstraintCallback 添加 feasibility cut 和 optimality cut
202      */
203     private class BendersCallback extends IloCplex.LazyConstraintCallback {
204         public void main() throws IloException {
205
206             // 由于主问题由 subcost 和另外一项组成，因此可以获得 subcost 的值
207             double zmasterProblem = getValue(subcost);//从主问题中获得 subcost 参数的值
208
```

```
209             //获得主问题中的变量 y 值
210             int[][] y2 = new int[data.SourcesSize][data.DemandsSize];
211             for (int i = 0; i < data.SourcesSize; i++) {
212                 for (int j = 0; j < data.DemandsSize; j++) {
213                     double aa = getValue(y[i][j]);
214                     if (aa > 0.5) {
215                         y2[i][j] = 1;
216                     } else {
217                         y2[i][j] = 0;
218                     }
219                 }
220             }
221             //根据松弛的主问题中的 变量 y 值重置子问题目标函数
222             // subProblem.remove(subObj);
223              IloLinearNumExpr subObj_expr0 = subProblem.linearNumExpr();
224              for (int i = 0; i < data.SourcesSize; i++) {
225                     subObj_expr0.addTerm(uSource[i], u[i]);
226                 }
227                 for (int i = 0; i < data.DemandsSize; i++) {
228                     subObj_expr0.addTerm(vDemand[i], v[i]);
229                 }
230                 for (int i = 0; i < data.SourcesSize; i++) {
231                     for (int j = 0; j < data.DemandsSize; j++) {
232                         subObj_expr0.addTerm(wM[i][j] * y2[i][j], w[i][j]);
233                     }
234                 }
235
236             // subObj_expr1 = (IloLinearNumExpr) subProblem.sum(subObj_expr,subObj_expr1);
237             // subObj = subProblem.addMaximize(subObj_expr0, "dualSubCost");
238             subObj.setExpr(subObj_expr0);//重置子问题目标函数
239
240             subProblem.solve();
241
242             IloCplex.Status status = subProblem.getStatus();
243
244             //判断子问题的求解状态
245             if (status == IloCplex.Status.Unbounded) {
246                 // 如果子问题求解状态是" 无界" 的
247                 // 获得射线
248                 /*
249                  * 返回值类型                          方法名称
250                  * IloLinearNumExpr          getRay()
251                  *           Returns a linear expression of the unbounded
252                  *           direction of a model proven unbounded by a
253                  *           simplex method.
254                  *
255                  * IloLinerNumExpr 还可以调用 toString() 方法将其表达式转换成字符串
256                  */
257                 //获得子问题的一条极射线 (子问题的一条射线是一个线性表达式: 表示无界的方向)
258                 // 如 1 * x1 - 1 * x2, 表示 45 度斜向上方向
259                 IloLinearNumExpr ray = subProblem.getRay();
260                 System.out.println("getRay returned " + ray.toString());
261
262                 //记录极射线的参数
263                 IloLinearNumExprIterator it = ray.linearIterator();
264                 double[] ucoef = new double[data.SourcesSize];    //极射线中 u 的系数
265                 double[] vcoef = new double[data.DemandsSize];    //极射线中 v 的系数
266                 double[][] wcoef = new double[data.SourcesSize][data.DemandsSize];//极射线中 w 的系数
267                 while (it.hasNext()) {
```

```
                    // 提取出 IloLinearNumExpr 中的决策变量，用 linearIterator 方法
                    IloNumVar var = it.nextNumVar();
                    boolean varFound = false;
                    for (int i = 0; i < data.SourcesSize && !varFound; i++) {
                        if (var.equals(u[i])) {
                            ucoef[i] = it.getValue(); // 得到极射线中变量的具体值
                            varFound = true;
                        }
                        for (int j = 0; j < data.DemandsSize && !varFound; j++) {
                            if (var.equals(w[i][j])) {
                                wcoef[i][j] = it.getValue() * wM[i][j];
                                varFound = true;
                            }
                        }
                    }
                    for (int i = 0; i < data.DemandsSize && !varFound; i++) {
                        if (var.equals(v[i])) {
                            vcoef[i] = it.getValue();
                            varFound = true;
                        }
                    }
                }

                //构造要添加到约束方程
                IloNumExpr expr1 = masterProblem.numExpr();
                double expr2 = 0;
                for (int i = 0; i < data.SourcesSize; i++) {
                    // 极射线的值 * 原问题的右端常数
                    expr2 += ucoef[i] * uSource[i];
                    expr1 = masterProblem.sum(expr1, masterProblem.scalProd(wcoef[i], y[i]));
                }
                for (int j = 0; j < data.DemandsSize; j++) {
                    expr2 += vcoef[j] * vDemand[j];
                }

                //创建，并添加约束方程到主问题中 (相当于在主问题中添加 cut)
                /*
                 * IloModeler.le(IloNumExpr e, double v, String name)
                 *
                 * Creates and returns a named range that forces the specified numeric
                 * expression to be less than or equal to the specified value
                 * -------
                 * 返回值类型                          方法名
                 * IloConstraint      add(IloConstraint object)
                 *         This method adds the constraint to the invoking
                 *         and-constraint.
                 */

                IloConstraint r = add(masterProblem.le(masterProblem.sum(expr1, expr2), 0));
                /*
                  // 或者也可以这么写
                IloRange range = masterProblem.le(masterProblem.sum(expr1, expr2), 0);
                IloConstraint r = add(range);
                */
                System.out.println("\n>>> Adding feasibility cut: " + r + "\n");


            } else if (status == IloCplex.Status.Optimal) {
                // 如果子问题最优，但是并不和主问题目标函数相等。则需要继续迭代
```

```
327                 if (zmasterProblem < subProblem.getObjValue() - eps) {
328                     //子问题有解，则最优解即一个极值点
329                     double[] ucoef = new double[data.SourcesSize];//极点中 u 的系数
330                     double[] vcoef = new double[data.DemandsSize];//极点中 v 的系数
331                     double[][] wcoef = new double[data.SourcesSize][data.DemandsSize];//极点中 w 的系数
332 
333                     // 得到极值点的值（也就是解）
334                     ucoef = subProblem.getValues(u);
335                     vcoef = subProblem.getValues(v);
336                     for (int i = 0; i < data.SourcesSize; i++) {
337                         wcoef[i] = subProblem.getValues(w[i]);
338                     }
339 
340                     //构造要添加到约束方程
341                     double expr3 = 0;
342                     IloNumExpr expr4 = masterProblem.numExpr();
343                     for (int i = 0; i < data.SourcesSize; i++) {
344                         expr3 += ucoef[i] * uSource[i];
345                         for (int j = 0; j < data.DemandsSize; j++) {
346                             wcoef[i][j] = wcoef[i][j] * wM[i][j];
347                         }
348                     }
349                     for (int j = 0; j < data.DemandsSize; j++) {
350                         expr3 += vcoef[j] * vDemand[j];
351                     }
352                     for (int i = 0; i < data.SourcesSize; i++) {
353                         expr4 = masterProblem.sum(expr4, masterProblem.scalProd(wcoef[i], y[i]));
354                     }
355                     //添加约束方程到主问题中
356                     IloConstraint r = add((IloRange) masterProblem.le(masterProblem.sum(expr3, expr4), subcost));
357                     System.out.println("\n>>> Adding optimality cut: " + r + "\n");
358                 } else {
359                     // 如果子问题最优，并且和主问题目标函数相等。则达到最优解
360                     System.out.println("\n>>> Accepting new incumbent with value " + getObjValue() + "\n");
361                     // the masterProblem and subproblem flow costs match
362                     // -- record the subproblem flows in case this proves to be the
363                     // winner (saving us from having to solve the LP one more time
364                     // once the masterProblem terminates)
365                     //主问题中 subcost 值和子问题中的最优解值相等
366                     for (int i = 0; i < data.SourcesSize; i++) {
367                         // 子问题的对偶变量就是原问题的解
368                         xValue[i] = subProblem.getDuals(subCon[i]); // 获得子问题约束 [i] 对应的对偶变量
369                     }
370                 }
371             } else {
372                 // unexpected status -- report but do nothing
373                 //出现不希望出现的状态，非法
374                 System.err.println("\n!!! Unexpected subproblem solution status: " + status + "\n");
375             }
376         }
377     }
378     //求解 benders 模型并复制结果
379     public final Solution solve() throws IloException {
380         Solution s = new Solution();
381         //记录结果
382         if (masterProblem.solve()) {
383             s.cost = masterProblem.getObjValue();
384             s.ship = new double[data.SourcesSize][];
385             s.link_y = new double[data.SourcesSize][];
```

```
386             for (int i = 0; i < data.SourcesSize; i++) {
387                 s.link_y[i] = masterProblem.getValues(y[i]);
388                 s.ship[i] = Arrays.copyOf(xValue[i], data.DemandsSize);
389             }
390         }
391         s.status = masterProblem.getCplexStatus();
392         return s;
393     }
394 }
```

17.4.4 BendersDecomposition 类

BendersDecomposition.java

```
1  package BendersDecomposition_FCTP;
2
3  import java.util.Arrays;
4  import ilog.concert.*;
5  import ilog.cplex.*;
6
7  public class BendersDecomposition {
8      Data data;
9      protected IloCplex subProblem;
10     protected IloCplex masterProblem;
11     IloObjective subObj;            //记录子问题目标函数
12     IloLinearNumExpr subObj_expr;
13
14     //对偶变量
15     protected IloNumVar[] u;    // 资源约束的对偶变量
16     protected IloNumVar[] v;    // 需求约束的对偶变量
17     protected IloNumVar[][] w;    // x,y 约束的对偶变量
18     double[] uSource;    //子问题中目标函数里对偶变量 u 对应系数
19     double[] vDemand;    //子问题中目标函数里对偶变量 v 对应系数
20     double[][] wM;        //子问题中目标函数里对偶变量 w 对应系数
21
22     protected IloRange[][] subCon;    //子问题的约束方程
23     double[][] xValue;                //记录原问题的 x 值
24
25     protected IloNumVar subcost;    //子问题中的目标值，对应松弛的主问题模型中 q
26     protected IloNumVar[][] y;        //主问题中的变量
27     double UB;
28     double LB;
29     public static final double eps = 1.0e-7;
30
31     int[][] yInit;                        //子问题中变量 y 初始值
32
33     /**
34      * 构造函数
35      *
36      * @param d
37      */
38     public BendersDecomposition(Data d) {
39         this.data = d;
40     }
41
42     /**
43      * 置 1 函数：将数组中的数字全部设置为 1
44      */
45     void setOne(int[][] a) {
```

```
46          for (int i = 0; i < a.length; i++) {
47              for (int j = 0; j < a[i].length; j++) {
48                  a[i][j] = 1;
49              }
50          }
51      }
52
53      /**
54       * 建立主问题和子问题的模型
55       * @throws IloException
56       */
57      public void buildBendersModels() throws IloException {
58          subProblem = new IloCplex();        // 子问题
59          masterProblem = new IloCplex();        // 主问题
60          subProblem.setOut(null);            // 关闭子问题的求解日志
61          masterProblem.setOut(null);            // 关闭主问题的求解日志
62
63          //参数初始化
64          yInit = new int[data.SourcesSize][data.DemandsSize];
65
66          setOne(yInit);            // 初始化参数 y=[1]，主问题的目标值首先需要确定
67          u = new IloNumVar[data.SourcesSize];
68          v = new IloNumVar[data.DemandsSize];
69          w = new IloNumVar[data.SourcesSize][data.DemandsSize];
70          uSource = new double[data.SourcesSize];
71          vDemand = new double[data.DemandsSize];
72          wM = new double[data.SourcesSize][data.DemandsSize];
73          y = new IloNumVar[data.SourcesSize][data.DemandsSize];
74          subCon = new IloRange[data.SourcesSize][data.DemandsSize];
75          xValue = new double[data.SourcesSize][data.DemandsSize];
76
77
78          /*
79           * 创建决策变量
80           */
81          subcost = masterProblem.numVar(0.0, Double.MAX_VALUE, IloNumVarType.Float, "subcost");
82          for (int i = 0; i < data.SourcesSize; i++) {
83              u[i] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "u_" + i);
84          }
85          for (int i = 0; i < data.DemandsSize; i++) {
86              v[i] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "v_" + i);
87          }
88          for (int i = 0; i < data.SourcesSize; i++) {
89              for (int j = 0; j < data.DemandsSize; j++) {
90                  y[i][j] = masterProblem.numVar(0, 1, IloNumVarType.Int, "y_" + i + "_" + j);
91                  w[i][j] = subProblem.numVar(0.0, Double.MAX_VALUE,IloNumVarType.Float, "w_" + i + "_" + j);
92              }
93          }
94
95          /*
96           * 构建主问题的目标函数
97           */
98          IloNumExpr expr0 = masterProblem.numExpr();
99          for (int i = 0; i < data.SourcesSize; i++) {
100             for (int j = 0; j < data.DemandsSize; j++) {
101                 expr0 = masterProblem.sum(expr0, masterProblem.prod(data.fixed_c[i][j], y[i][j]));
102             }
103         }
104         masterProblem.addMinimize(masterProblem.sum(subcost, expr0), "TotalCost");
```

```
105
106         /*
107          * 构建子问题
108          */
109
110         // 子问题目标函数
111         subObj_expr = subProblem.linearNumExpr();
112         IloLinearNumExpr obj = subProblem.linearNumExpr();
113         for (int i = 0; i < data.SourcesSize; i++) {
114             uSource[i] = -data.supply[i];
115             obj.addTerm(uSource[i], u[i]);
116             subObj_expr.addTerm(uSource[i], u[i]);
117         }
118         for (int i = 0; i < data.DemandsSize; i++) {
119             vDemand[i] = data.demand[i];
120             obj.addTerm(vDemand[i], v[i]);
121             subObj_expr.addTerm(vDemand[i], v[i]);
122         }
123         for (int i = 0; i < data.SourcesSize; i++) {
124             for (int j = 0; j < data.DemandsSize; j++) {
125                 wM[i][j] = -data.big_M[i][j];
126                 obj.addTerm(wM[i][j] * yInit[i][j], w[i][j]);
127             }
128         }
129         subObj = subProblem.addMaximize(obj, "dualSubCost");
130
131         // 子问题约束方程
132         for (int i = 0; i < data.SourcesSize; i++) {
133             for (int j = 0; j < data.DemandsSize; j++) {
134                 IloNumExpr expr = subProblem.numExpr();
135                 IloNumExpr expr1 = subProblem.numExpr();
136                 expr = subProblem.sum(subProblem.prod(-1, u[i]), v[j]);
137                 expr1 = subProblem.sum(expr, subProblem.prod(-1, w[i][j]));
138                 subCon[i][j] = subProblem.addLe(expr1, data.c[i][j],"C" + i + "_" + j);
139             }
140         }
141         // turn off the presolver for the main model
142         subProblem.setParam(IloCplex.BooleanParam.PreInd, false);
143         subProblem.setParam(IloCplex.Param.RootAlgorithm, IloCplex.Algorithm.Primal);
144     }
145
146     /**
147      * Benders decomposition 算法的实现
148      *
149      * @throws IloException
150      */
151     public void bendersDecompositionSolve() throws IloException {
152
153         UB = Double.MAX_VALUE;        // 全局上界
154         LB = Double.MIN_VALUE;        // 全局下界
155
156         while (UB > LB + eps) {
157              //根据松弛的主问题中的 变量 y 值重置子问题目标函数
158              //subProblem.remove(subObj);
159              IloLinearNumExpr subObj_expr0 = subProblem.linearNumExpr();
160              for (int i = 0; i < data.SourcesSize; i++) {
161                     subObj_expr0.addTerm(uSource[i], u[i]);
162                 }
163                 for (int i = 0; i < data.DemandsSize; i++) {
```

```
                    subObj_expr0.addTerm(vDemand[i], v[i]);
                }
                for (int i = 0; i < data.SourcesSize; i++) {
                    for (int j = 0; j < data.DemandsSize; j++) {
                        subObj_expr0.addTerm(wM[i][j] * yInit[i][j], w[i][j]);
                    }
                }

            /*
            subObj_expr1 = (IloLinearNumExpr) subProblem.sum(subObj_expr,subObj_expr1);
            subObj = subProblem.addMaximize(subObj_expr0, "dualSubCost");
            */
            subObj.setExpr(subObj_expr0);//重置子问题目标函数
            subProblem.solve();

            //获取原问题中运载量变量 x 值
            for (int i = 0; i < data.SourcesSize; i++) {
                xValue[i] = subProblem.getDuals(subCon[i]);
            }
            IloCplex.Status status = subProblem.getStatus();

            //判断子问题的求解状态
            if (status == IloCplex.Status.Unbounded) {
                // 获得射线
                IloLinearNumExpr ray = subProblem.getRay();//获得子问题的一条极射线
                System.out.println("getRay returned " + ray.toString());
                //记录极射线的参数
                IloLinearNumExprIterator it = ray.linearIterator();
                double[] ucoef = new double[data.SourcesSize];    //极射线中 u 的系数
                double[] vcoef = new double[data.DemandsSize];    //极射线中 v 的系数
                double[][] wcoef = new double[data.SourcesSize][data.DemandsSize];//极射线中 w 的系数
                while (it.hasNext()) {
                    IloNumVar var = it.nextNumVar();
                    boolean varFound = false;
                    for (int i = 0; i < data.SourcesSize && !varFound; i++) {
                        if (var.equals(u[i])) {
                            ucoef[i] = it.getValue();
                            varFound = true;
                        }
                        for (int j = 0; j < data.DemandsSize && !varFound; j++) {
                            if (var.equals(w[i][j])) {
                                wcoef[i][j] = it.getValue() * wM[i][j];
                                varFound = true;
                            }
                        }
                    }
                    for (int i = 0; i < data.DemandsSize && !varFound; i++) {
                        if (var.equals(v[i])) {
                            vcoef[i] = it.getValue();
                            varFound = true;
                        }
                    }
                }

                //构造要添加到约束方程
                IloNumExpr expr1 = masterProblem.numExpr();
                double expr2 = 0;
                for (int i = 0; i < data.SourcesSize; i++) {
                    expr2 += ucoef[i] * uSource[i];
```

```
223                     expr1 = masterProblem.sum(expr1, masterProblem.scalProd(wcoef[i], y[i]));
224                 }
225                 for (int j = 0; j < data.DemandsSize; j++) {
226                     expr2 += vcoef[j] * vDemand[j];
227                 }
228 
229                 /*
230                  * addCut 方法返回一个 IloConstraint 类
231                  */
232                 //添加约束方程到主问题中
233                 IloConstraint r = masterProblem.addCut(masterProblem.le(masterProblem.sum(expr1, expr2), 0));
234                 // IloConstraint r = add(masterProblem.le(masterProblem.sum(expr1, expr2), 0));
235                 System.out.println("\n>>> Adding feasibility cut: " + r + "\n");
236 
237             } else if (status == IloCplex.Status.Optimal) {
238 
239                     //子问题有解，则最优解即一个极值点
240                     double[] ucoef = new double[data.SourcesSize];    //极点中 u 的系数
241                     double[] vcoef = new double[data.DemandsSize];    //极点中 v 的系数
242                     double[][] wcoef = new double[data.SourcesSize][data.DemandsSize];    //极点中 w 的系数
243                     ucoef = subProblem.getValues(u);
244                     vcoef = subProblem.getValues(v);
245                     for (int i = 0; i < data.SourcesSize; i++) {
246                         wcoef[i] = subProblem.getValues(w[i]);
247                     }
248 
249                     //构造要添加到约束方程
250                     double expr3 = 0;
251                     IloNumExpr expr4 = masterProblem.numExpr();
252                     for (int i = 0; i < data.SourcesSize; i++) {
253                         expr3 += ucoef[i] * uSource[i];
254                         for (int j = 0; j < data.DemandsSize; j++) {
255                             wcoef[i][j] = wcoef[i][j] * wM[i][j];
256                         }
257                     }
258                     for (int j = 0; j < data.DemandsSize; j++) {
259                         expr3 += vcoef[j] * vDemand[j];
260                     }
261                     for (int i = 0; i < data.SourcesSize; i++) {
262                         expr4 = masterProblem.sum(expr4, masterProblem.scalProd(wcoef[i], y[i]));
263                     }
264 
265                     //添加约束方程到主问题中
266                     IloConstraint r = masterProblem.addCut((IloRange) masterProblem.le(masterProblem.sum(expr3,
                    ↪  expr4), subcost));
267                     double expr5 =0;
268                     for (int i = 0; i < data.SourcesSize; i++) {
269                         for (int j = 0; j < data.DemandsSize; j++) {
270                             expr5 +=data.fixed_c[i][j]*yInit[i][j];
271                         }
272                     }
273                     UB = Math.min(UB, expr5+subProblem.getObjValue());
274                     // System.out.println(expr5+" "+subProblem.getObjValue());
275                     System.out.println("\n>>> Adding optimality cut: " + r + "\n");
276             } else {
277                 // unexpected status -- report but do nothing
278                 //出现不希望出现的状态，非法
279                 System.err.println("\n!!! Unexpected subproblem solution status: " + status + "\n");
280             }
```

```
281
282             // 求解主问题
283             masterProblem.solve();
284             LB = masterProblem.getObjValue();
285
286             //获得主问题中的变量 y 值
287             for (int i = 0; i < data.SourcesSize; i++) {
288                 for (int j = 0; j < data.DemandsSize; j++) {
289                     double aa = masterProblem.getValue(y[i][j]);
290                     if (aa > 0.5) {
291                         yInit[i][j] = 1;
292                     } else {
293                         yInit[i][j] = 0;
294                     }
295                 }
296             }
297         }
298
299     }
300
301
302     /**
303      * 调用 Bender Decomposition 算法求解模型
304      * @return
305      * @throws IloException
306      */
307     public final Solution solve() throws IloException {
308         Solution s = new Solution();
309         bendersDecompositionSolve();
310         //记录结果
311         if (masterProblem.getStatus()==IloCplex.Status.Optimal) {
312             s.cost = masterProblem.getObjValue();
313             s.ship = new double[data.SourcesSize][];
314             s.link_y = new double[data.SourcesSize][];
315             for (int i = 0; i < data.SourcesSize; i++) {
316                 s.link_y[i] = masterProblem.getValues(y[i]);
317                 s.ship[i] = Arrays.copyOf(xValue[i], data.DemandsSize);
318             }
319         }else{
320             System.out.println("!!!!!!!!!!!Unexpected masterProblem problem solution status");
321         }
322         s.status = masterProblem.getCplexStatus();
323         return s;
324     }
325 }
```

17.4.5 SolveMIPModel 类

SolveMIPModel.java

```
1  package BendersDecomposition_FCTP;
2
3  import ilog.concert.*;
4  import ilog.cplex.*;
5
6  /**
7   * 本代码为直接调用 CPLEX 求解 FCTP
8   */
9
```

```
10  public class SolveMIPModel {
11
12      private IloCplex cplex;
13      private IloNumVar[][] x;    // 决策变量 x[i][j]
14      private IloNumVar[][] y;    // 决策变量 y[i][j]
15
16      /**
17       * 建立 cplex 模型
18       *
19       * @param data
20       * @throws IloException
21       */
22      public void buildMIP(Data data) throws IloException {
23          cplex = new IloCplex();
24          // cplex.setOut(null);    // 关闭求解器的日志
25          x = new IloNumVar[data.SourcesSize][data.DemandsSize];
26          y = new IloNumVar[data.SourcesSize][data.DemandsSize];
27
28          // 定义 cplex 变量 x 和 y 的数据类型及取值范围
29          for (int i = 0; i < x.length; i++) {
30              for (int j = 0; j < x[i].length; j++) {
31                  x[i][j] = cplex.numVar(0, 1.0e5, IloNumVarType.Float, "x_" + i + "_" + j);
32                  y[i][j] = cplex.numVar(0, 1, IloNumVarType.Int, "y_" + i + "_" + j);
33              }
34          }
35
36          // 定义目标函数
37          IloNumExpr obj = cplex.numExpr();
38          for (int i = 0; i < data.SourcesSize; i++) {
39              for (int j = 0; j < data.DemandsSize; j++) {
40                  obj = cplex.sum(obj,
41                          cplex.sum(cplex.prod(data.fixed_c[i][j], y[i][j]), cplex.prod(data.c[i][j], x[i][j])));
42              }
43          }
44          cplex.addMinimize(obj, "total costs");
45
46          //添加约束
47          // supply
48          for (int i = 0; i < data.SourcesSize; i++) {
49              IloNumExpr expr1 = cplex.numExpr();
50              for (int j = 0; j < data.DemandsSize; j++) {
51                  expr1 = cplex.sum(expr1, x[i][j]);
52              }
53              cplex.addLe(expr1, data.supply[i], "Supply_" + i);
54          }
55
56          // demand
57          for (int i = 0; i < data.DemandsSize; i++) {
58              IloNumExpr expr2 = cplex.numExpr();
59              for (int j = 0; j < data.SourcesSize; j++) {
60                  expr2 = cplex.sum(expr2, x[j][i]);
61              }
62              cplex.addGe(expr2, data.demand[i], "Demand_" + i);
63          }
64          for (int i = 0; i < data.SourcesSize; i++) {
65              for (int j = 0; j < data.DemandsSize; j++) {
66                  cplex.addLe(x[i][j], cplex.prod(data.big_M[i][j], y[i][j]));
67              }
68          }
```

```
69          cplex.exportModel("MIP.lp");
70      }
71
72      // 求解 MIP 并获得结果
73      public Solution solve() throws IloException {
74          Solution s = new Solution();
75          if (cplex.solve()) {
76              s.cost = cplex.getObjValue();
77
78              // 注意这里获得二维数组解的方法，是一行一行的获取
79              s.ship = new double[x.length][];
80              s.link_y = new double[y.length][];
81              for (int i = 0; i < x.length; i++) {
82                  s.ship[i] = cplex.getValues(x[i]);
83                  s.link_y[i] = cplex.getValues(y[i]);
84              }
85          }
86
87          // 获得求解的状态
88          s.status = cplex.getCplexStatus();
89          return s;
90      }
91  }
```

17.4.6 run_this 类

该类用于算法的测试。我们分别使用直接调用 CPLEX 求解、调用基于 callback 实现的 Benders 分解、用户通过自己的方法实现的 Benders 分解算法对 FCTP 进行求解。

run_this.java

```
1   package BendersDecomposition_FCTP;
2
3   import ilog.concert.IloException;
4
5   /**
6    * 代码原作者：数据魔术师公众号
7    * 代码修改和注释：刘兴禄，清华大学
8    *
9    * 此代码分三部分：
10   * 1. 直接建立 cplex 模型求解。
11   * 2. 建立主问题和子问题的 cplex 模型，调用 cplex 的 LazyConstraintCallback 实现 benders decomposition。
12   * 3. 建立主问题和子问题的 cplex 模型，用户自行实现 benders decomposition。
13   *
14   * 这三部分分别对应 SolveMIPModel 类、BendersDecomposition_callback 类、BendersDecomposition 类。
15   */
16
17  public class run_this {
18          public static void main(String[] args) throws Exception {
19              Data data = new Data();
20
21              String path = "F:\\MyCode\\JavaCode\\MyAlgorithmLib\\src\\BendersDecomposition_FCTP\\test1.txt";
22
23              data.read_data(path);
24              long start;
25              long end;
26
27              /**
```

```
28              * 第一种方法：直接调用 CPLEX 进行求解 (SolveMIPModel)
29              */
30
31             try{
32                 start = System.currentTimeMillis();
33                 System.out.println("\n================================================="
34                         + "\n        Solving the entire MIP model          "
35                         + "\n=================================================");
36
37                 // 构建模型
38                 SolveMIPModel model = new SolveMIPModel();
39                 model.buildMIP(data);
40
41                 // 求解模型
42                 Solution s = model.solve();
43                 end = System.currentTimeMillis();
44                 System.out.println("\n***\nThe objValue of the MIP model: "
45                         + String.format("%10.2f", s.cost));
46                 System.out.println();
47                 s.print_ship();
48                 System.out.println("\n Elapsed time = "+ (end - start) + " ms\n");
49
50             } catch (IloException ex) {
51                 System.err.println("\n!!!Unable to solve the MIP model"
52                         + ex.getMessage() + "\n!!!");
53                 System.exit(1);
54             }
55
56
57             /**
58              * 第二种方法：
59              *     用 Benders Decomposition 算法来求解 (callback 实现的版本)
60              */
61
62             try{
63                 start = System.currentTimeMillis();
64                 System.out.println("\n================================================="
65                         + "\n     Solving via Benders decomposition (callback)    "
66                         + "\n=================================================");
67
68                 // 构建模型
69                 BendersDecomposition_callback model2 = new BendersDecomposition_callback(data);
70                 model2.buildBendersModels();
71
72                 // 使用 benders decomposition 算法进行求解
73                 Solution s = model2.solve();
74                 end = System.currentTimeMillis();
75                 System.out.println("\n***\nThe objValue of the problem (Benders via callback):"
76                         + String.format("%10.2f", s.cost));
77                 System.out.println();
78                 s.print_ship();
79                 System.out.println("\n Elapsed time = "+ (end - start) + " ms \n");
80             }catch (IloException ex) {
81                 System.err.println("\n!!!Unable to solve the problem via benders:\n"
82                         + ex.getMessage() + "\n!!!");
83                 System.exit(2);
84             }
85
86
```

```
87              /**
88               * 第三种方法:
89               *     用 Benders Decomposition 算法来求解 (自行实现的方法)
90               */
91
92              try{
93                  start = System.currentTimeMillis();
94                  System.out.println("\n================================================"
95                          + "\n   Solving via Benders decomposition (user designed)    "
96                          + "\n================================================");
97                  BendersDecomposition model3 = new BendersDecomposition(data);
98                  model3.buildBendersModels();
99                  Solution s = model3.solve();
100                 end = System.currentTimeMillis();
101                 System.out.println("\n***\nThe objValue of the problem (Benders via user designed): "
102                         + String.format("%10.2f", s.cost));
103                 System.out.println();
104                 s.print_ship();
105                 System.out.println("\n*** Elapsed time = "+ (end - start) + " ms ***\n");
106             }catch (IloException ex) {
107                 System.err.println("\n!!!Unable to solve the problem via benders:\n"
108                         + ex.getMessage() + "\n!!!");
109                 System.exit(2);
110             }
111
112         }
113
114 }
```

17.4.7 算例 1

算例 1

```
1  sources demands
2  4    3
3  supply
4  10    30    40    20
5  demand
6  20    50    30
7  variable cost
8  2    3    4
9  3    2    1
10 1    4    3
11 4    5    2
12 fixed cost
13 10    30    20
14 10    30    20
15 10    30    20
16 10    30    20
```

3 种方法的求解结果如下。

算例 1 在 3 种方法下的求解结果

```
1  ================================================
2          Solving the entire MIP model
3  ================================================
4
5  The objValue of the MIP model:     350.00
```

```
     0 -> 0: 0.000000    0 -> 1: 0.000000    0 -> 2: 10.000000
     1 -> 0: 0.000000    1 -> 1: 30.000000    1 -> 2: 0.000000
     2 -> 0: 20.000000    2 -> 1: 20.000000    2 -> 2: 0.000000
     3 -> 0: 0.000000    3 -> 1: 0.000000    3 -> 2: 20.000000

Elapsed time = 96 ms

================================================
     Solving via Benders decomposition (callback)
================================================
The benders model's solution has total cost  350.00000

     0 -> 0: -0.000000    0 -> 1: -0.000000    0 -> 2: 10.000000
     1 -> 0: -0.000000    1 -> 1: 30.000000    1 -> 2: -0.000000
     2 -> 0: 20.000000    2 -> 1: 20.000000    2 -> 2: -0.000000
     3 -> 0: -0.000000    3 -> 1: -0.000000    3 -> 2: 20.000000

Elapsed time = 30 ms

================================================
   Solving via Benders decomposition (user designed)
================================================

The benders model's solution has total cost  350.00000
     0 -> 0: 0.000000    0 -> 1: -0.000000    0 -> 2: 10.000000
     1 -> 0: -0.000000    1 -> 1: 30.000000    1 -> 2: 0.000000
     2 -> 0: 20.000000    2 -> 1: 20.000000    2 -> 2: -0.000000
     3 -> 0: -0.000000    3 -> 1: 0.000000    3 -> 2: 20.000000

Elapsed time = 388 ms
```

17.4.8 算例 2

算例 2

```
sources demands
8    7
supply
20 20 20 18 18 17 17 10
demand
20 19 19 18 17 16 16
variable cost
31    27    28    10     7    26    30
15    19    17     7    22    17    16
21    17    19    29    27    22    13
9     19     7    15    20    17    22
19     7    18    10    12    27    23
8     16    10    10    11    13    15
14    32    22    10    22    15    19
30    27    24    26    25    15    19
fixed cost
649   685   538   791   613   205   467
798   211   701   506   431   907   945
687   261   444   264   443   946   372
335   385   967   263   423   592   939
819   340   233   889   211   854   823
307   620   845   919   223   854   656
```

```
560   959   782   417   358   589   383
375   791   720   416   251   887   235
```

3 种方法的求解结果如下。

算例 2 在 3 种方法下的求解结果

```
================================================
        Solving the entire MIP model
================================================
The objValue of the MIP model:    4541.00

    0 -> 0:   0.0    0 -> 1:   0.0    0 -> 2:   0.0    0 -> 3:   0.0    0 -> 4:   0.0    0 -> 5:  16.0    0 -> 6:
    ↪  0.0
    1 -> 0:   0.0    1 -> 1:  19.0    1 -> 2:   0.0    1 -> 3:   0.0    1 -> 4:   0.0    1 -> 5:   0.0    1 -> 6:
    ↪  0.0
    2 -> 0:   0.0    2 -> 1:   0.0    2 -> 2:  19.0    2 -> 3:   0.0    2 -> 4:   0.0    2 -> 5:   0.0    2 -> 6:
    ↪  0.0
    3 -> 0:   0.0    3 -> 1:   0.0    3 -> 2:   0.0    3 -> 3:  18.0    3 -> 4:   0.0    3 -> 5:   0.0    3 -> 6:
    ↪  0.0
    4 -> 0:   0.0    4 -> 1:   0.0    4 -> 2:   0.0    4 -> 3:   0.0    4 -> 4:  17.0    4 -> 5:   0.0    4 -> 6:
    ↪  0.0
    5 -> 0:  17.0    5 -> 1:   0.0    5 -> 2:   0.0    5 -> 3:   0.0    5 -> 4:   0.0    5 -> 5:   0.0    5 -> 6:
    ↪  0.0
    6 -> 0:   0.0    6 -> 1:   0.0    6 -> 2:   0.0    6 -> 3:   0.0    6 -> 4:   0.0    6 -> 5:   0.0    6 -> 6:
    ↪  16.0
    7 -> 0:   3.0    7 -> 1:   0.0    7 -> 2:   0.0    7 -> 3:   0.0    7 -> 4:   0.0    7 -> 5:   0.0    7 -> 6:
    ↪  0.0

 Elapsed time = 121 ms

================================================
     Solving via Benders decomposition (callback)
================================================

The objValue of the problem (Benders via callback):   4541.00

    0 -> 0:  -0.0    0 -> 1:   0.0    0 -> 2:  -0.0    0 -> 3:  -0.0    0 -> 4:  -0.0    0 -> 5:  16.0    0 -> 6:
    ↪  -0.0
    1 -> 0:  -0.0    1 -> 1:  19.0    1 -> 2:  -0.0    1 -> 3:  -0.0    1 -> 4:  -0.0    1 -> 5:  -0.0    1 -> 6:
    ↪  -0.0
    2 -> 0:  -0.0    2 -> 1:  -0.0    2 -> 2:  19.0    2 -> 3:  -0.0    2 -> 4:   0.0    2 -> 5:  -0.0    2 -> 6:
    ↪  -0.0
    3 -> 0:  -0.0    3 -> 1:  -0.0    3 -> 2:  -0.0    3 -> 3:  18.0    3 -> 4:  -0.0    3 -> 5:  -0.0    3 -> 6:
    ↪  -0.0
    4 -> 0:  -0.0    4 -> 1:  -0.0    4 -> 2:  -0.0    4 -> 3:  -0.0    4 -> 4:  17.0    4 -> 5:  -0.0    4 -> 6:
    ↪  -0.0
    5 -> 0:  17.0    5 -> 1:  -0.0    5 -> 2:  -0.0    5 -> 3:  -0.0    5 -> 4:  -0.0    5 -> 5:  -0.0    5 -> 6:
    ↪  -0.0
    6 -> 0:  -0.0    6 -> 1:  -0.0    6 -> 2:  -0.0    6 -> 3:  -0.0    6 -> 4:   0.0    6 -> 5:  -0.0    6 -> 6:
    ↪  16.0
    7 -> 0:   3.0    7 -> 1:  -0.0    7 -> 2:  -0.0    7 -> 3:  -0.0    7 -> 4:  -0.0    7 -> 5:  -0.0    7 -> 6:
    ↪  -0.0

 Elapsed time = 165 ms

================================================
   Solving via Benders decomposition (user designed)
================================================
The objValue of the problem (Benders via user designed):    4541.00
```

```

    0 -> 0:  -0.0    0 -> 1:  -0.0    0 -> 2:  -0.0    0 -> 3:  -0.0    0 -> 4:  -0.0    0 -> 5:  16.0    0 -> 6:
    ↪  0.0
    1 -> 0:   0.0    1 -> 1:  19.0    1 -> 2:  -0.0    1 -> 3:  -0.0    1 -> 4:  -0.0    1 -> 5:  -0.0    1 -> 6:
    ↪  0.0
    2 -> 0:  -0.0    2 -> 1:  -0.0    2 -> 2:  19.0    2 -> 3:  -0.0    2 -> 4:  -0.0    2 -> 5:  -0.0    2 -> 6:
    ↪  -0.0
    3 -> 0:   3.0    3 -> 1:  -0.0    3 -> 2:  -0.0    3 -> 3:   7.0    3 -> 4:  -0.0    3 -> 5:  -0.0    3 -> 6:
    ↪  -0.0
    4 -> 0:   0.0    4 -> 1:  -0.0    4 -> 2:  -0.0    4 -> 3:  -0.0    4 -> 4:  17.0    4 -> 5:  -0.0    4 -> 6:
    ↪  -0.0
    5 -> 0:  17.0    5 -> 1:  -0.0    5 -> 2:  -0.0    5 -> 3:  -0.0    5 -> 4:  -0.0    5 -> 5:  -0.0    5 -> 6:
    ↪  -0.0
    6 -> 0:  -0.0    6 -> 1:  -0.0    6 -> 2:  -0.0    6 -> 3:  11.0    6 -> 4:  -0.0    6 -> 5:  -0.0    6 -> 6:
    ↪  6.0
    7 -> 0:  -0.0    7 -> 1:  -0.0    7 -> 2:  -0.0    7 -> 3:  -0.0    7 -> 4:  -0.0    7 -> 5:  -0.0    7 -> 6:
    ↪  10.0

Elapsed time = 1311 ms
```

17.5 Python 调用 Gurobi 实现 Benders 分解求解 FLP

本小节给出 Python 调用 Gurobi 实现 Benders 分解求解 FLP 的完整代码。本代码原作者为杉数科技算法工程师伍健[3]。

其中测试数据存在文件 `warehouse.dat` 中。其中：

- 第 1 行是候选仓库点的个数；
- 第 2 行是零售店的个数；
- 第 3 行是每个候选仓库点的供给量；
- 第 4 行是每个零售店的需求量；
- 第 5 行是每个候选仓库点的固定费用；
- 第 6 到 30 行是运输费用矩阵，行表示候选仓库点，列表示零售店。

在初始化 Benders Subproblem 时，我们直接将所有 y_i 固定为 1，即假设初始情况下，所有的仓库都开通。

本章提供的 Benders Decomposition 代码是使用 Gurobi 的 callback 实现的。

17.5.1 warehouse.py

warehouse.py

```
from __future__ import division, print_function

import gurobipy as GRBPY


# awkward restriction for 'callback'
def cbwarehouse(model, where):
    if where == GRBPY.GRB.Callback.MIPSOL:
        if model._iter >= 1:
            for i in range(model._nwarehouse):
                model._csupply[i].rhs = model.cbGetSolution(model._vmbuild[i]) * model._supply[i]

        model._sub.optimize()

        if model._sub.status == GRBPY.GRB.INFEASIBLE:
            print("Iteration: ", model._iter)
            print("Adding feasibility cut...\n")

            lazycut = GRBPY.quicksum(model._csupply[i].farkasdual * model._supply[i] * model._vmbuild[i] \
                                     for i in range(model._nwarehouse)) + \
                      sum(model._cdemand[i].farkasdual * model._demand[i] for i in range(model._nstore))

            model.cbLazy(lazycut >= 0)

            model._iter += 1
        elif model._sub.status == GRBPY.GRB.OPTIMAL:
            if model._sub.objval > model.cbGetSolution(model._maxshipcost) + 1e-6:
                print("Iteration: ", model._iter)
                print("Adding optimality cut...\n")

```

[3]Github 链接：https://github.com/wujianjack/optimizationmodels/tree/master/gurobi。

```
                lazycut = GRBPY.quicksum(model._csupply[i].pi * model._supply[i] * model._vmbuild[i] \
                                         for i in range(model._nwarehouse)) + \
                          sum(model._cdemand[i].pi * model._demand[i] for i in range(model._nstore))

                model.cbLazy(model._maxshipcost >= lazycut)

                model._iter += 1
        else:
            model.terminate()

class WareHouse:
    def __init__(self):
        # initialize data
        self.nwarehouse = 0
        self.nstore = 0
        self.supply = []
        self.demand = []
        self.fixcost = []
        self.varcost = []

        # initialize variables and constraints
        self.vmbuild = []
        self.vship = []
        self.csupply = []
        self.cdemand = []

    def read(self, filename):
        # input data
        with open(filename, "r") as data:
            self.nwarehouse = int(data.readline())
            self.nstore = int(data.readline())

            column = data.readline().split()
            for i in range(self.nwarehouse):
                self.supply.append(float(column[i]))

            column = data.readline().split()
            for i in range(self.nstore):
                self.demand.append(float(column[i]))

            column = data.readline().split()
            for i in range(self.nwarehouse):
                self.fixcost.append(float(column[i]))

            for i in range(self.nwarehouse):
                column = data.readline().split()
                lvarcost = []
                for j in range(self.nstore):
                    lvarcost.append(float(column[j]))
                self.varcost.append(lvarcost)

    def build(self):
        try:
            # define models for 'master' and 'sub'
            self.master = GRBPY.Model("master")
            self.sub = GRBPY.Model("sub")

            # disable log information
            self.master.setParam("OutputFlag", 0)
```

```
            self.sub.setParam("OutputFlag", 0)

            # use lazy constraints
            self.master.setParam("LazyConstraints", 1)

            # disable presolving in subproblem
            self.sub.setParam("Presolve", 0)

            # required to obtain farkas dual
            self.sub.setParam("InfUnbdInfo", 1)

            # use dual simplex
            self.sub.setParam("Method", 1)

            # construct master problem
            for i in range(self.nwarehouse):
                self.vmbuild.append(self.master.addVar(0.0, 1.0, 0.0, GRBPY.GRB.BINARY))

            self.maxshipcost = self.master.addVar(0.0, GRBPY.GRB.INFINITY, 0.0, GRBPY.GRB.CONTINUOUS)

            self.master.setObjective(GRBPY.quicksum(self.fixcost[i] * self.vmbuild[i] \
                                                    for i in range(self.nwarehouse)) + \
                                                    self.maxshipcost, GRBPY.GRB.MINIMIZE)

            # construct subproblem
            for i in range(self.nwarehouse):
                lvship = []
                for j in range(self.nstore):
                    lvship.append(self.sub.addVar(0.0, GRBPY.GRB.INFINITY, 0.0, GRBPY.GRB.CONTINUOUS))
                self.vship.append(lvship)

            for i in range(self.nwarehouse):
                self.csupply.append(self.sub.addConstr(GRBPY.quicksum(self.vship[i][j] \
                                                                      for j in range(self.nstore)) \
                                                                      <= self.supply[i] * 1.0))

            for j in range(self.nstore):
                self.cdemand.append(self.sub.addConstr(GRBPY.quicksum(self.vship[i][j] \
                                                                      for i in range(self.nwarehouse)) \
                                                                      == self.demand[j]))

            self.sub.setObjective(GRBPY.quicksum(self.varcost[i][j] * self.vship[i][j] \
                                                 for i in range(self.nwarehouse) \
                                                 for j in range(self.nstore)), GRBPY.GRB.MINIMIZE)
        except GRBPY.GurobiError as e:
            print('Error code' + str(e.errno) + ': ' + str(e))
        except AttributeError as e:
            print('Encountered an attribute error: ' + str(e))

    def solve(self):
        # build 'master' and 'sub'
        self.build()

        # register callback
        self.master._iter = 0
        self.master._nwarehouse = self.nwarehouse
        self.master._nstore = self.nstore
        self.master._supply = self.supply
        self.master._demand = self.demand
```

```
149
150         self.master._csupply = self.csupply
151         self.master._cdemand = self.cdemand
152         self.master._vmbuild = self.vmbuild
153         self.master._maxshipcost = self.maxshipcost
154
155         self.master._sub = self.sub
156
157         # optimize master problem
158         print("               *** Benders Decomposition Loop ***               ")
159         self.master.optimize(cbwarehouse)
160         print("                        *** End Loop ***                        ")
161
162         # it seems that 64-bit needs this extra work
163         for i in range(self.nwarehouse):
164             self.csupply[i].rhs = self.vmbuild[i].x * self.supply[i]
165
166         self.sub.optimize()
167
168     def report(self):
169         print("               *** Summary Report ***               ")
170         print("Objective: %.6f" % self.master.objval)
171         print("Variables:")
172         for i in range(self.nwarehouse):
173             if abs(self.vmbuild[i].x) > 1e-6:
174                 print("  Build[%d] = %.0f" % (i, self.vmbuild[i].x))
175
176         for i in range(self.nwarehouse):
177             for j in range(self.nstore):
178                 if abs(self.vship[i][j].x) > 1e-6:
179                     print("  Ship[%d][%d] = %.6f" % (i, j, self.vship[i][j].x))
180
181 if __name__ == "__main__":
182     warehouse = WareHouse()
183     warehouse.read("warehouse.dat")
184     warehouse.solve()
185     warehouse.report()
```

17.5.2 算例格式 multicommodity

算例格式 multicommodity

```
1  25
2  6
3  23070  18290  20010  15080  17540  21090  16650  18420  19160  18860  22690  19360  22330  15440  19330  17110
   ↪  15380  18690  20720  21220  16720  21540  16500  15310  18970
4  12000  12000  14000  13500  25000  29000
5  500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000 500000
   ↪  500000 500000 500000 500000 500000 500000 500000 500000 500000
6  73.78  14.76  86.82  91.19  51.03  76.49
7  60.28  20.92  76.43  83.99  58.84  68.86
8  58.18  21.64  69.84  72.39  61.64  58.39
9  50.37  21.74  61.49  65.72  60.48  56.68
10 42.73  35.19  44.11  58.08  65.76  55.51
11 44.62  39.21  44.44  48.32  76.12  51.17
12 49.31  51.72  36.27  42.96  84.52  49.61
13 50.79  59.25  22.53  33.22  94.30  49.66
14 51.93  72.13  21.66  29.39  93.52  49.63
15 65.90  13.07  79.59  86.07  46.83  69.55
```

```
 50.79   9.99  67.83  78.81  49.34  60.79
 47.51  12.95  59.57  67.71  51.13  54.65
 39.36  19.01  56.39  62.37  57.25  47.91
 33.55  30.16  40.66  48.50  60.83  42.51
 34.17  40.46  40.23  47.10  66.22  38.94
 41.68  53.03  22.56  30.89  77.22  35.88
 42.75  62.94  18.58  27.02  80.36  40.11
 46.46  71.17  17.17  21.16  91.65  41.56
 56.83   8.84  83.99  91.88  41.38  67.79
 46.21   2.92  68.94  76.86  38.89  60.38
 41.67  11.69  61.05  70.06  43.24  48.48
 25.57  17.59  54.93  57.07  44.93  43.97
 28.16  29.39  38.64  46.48  50.16  34.20
 26.97  41.62  29.72  40.61  59.56  31.21
 34.24  54.09  22.13  28.43  69.68  24.09
```

参考文献

[1] Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020.

[2] Claudia Archetti, Nicola Bianchessi, and Maria Grazia Speranza. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58(4):241–254, 2011.

[3] Roberto Baldacci and Aristide Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347, 2009.

[4] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[5] J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394, 1989.

[6] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

[7] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[8] JF Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[9] Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the Acm*, 1980.

[10] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Edmund K Burke, Graham Kendall, et al. *Search methodologies*. Springer, 2005.

[12] Paola Cappanera and Maria Paola Scaparra. Optimal allocation of protective resources in shortest-path networks. *Transportation Science*, 45(1):64–80, 2011. doi: 10.1287/trsc.1100.0340. URL https://doi.org/10.1287/trsc.1100.0340.

[13] Alain Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

[14] Boris V Cherkassky, Andrew V Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical programming*, 73(2): 129–174, 1996.

[15] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[16] Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985, 2019. doi: 10.1287/trsc.2018.0878. URL https://doi.org/10.1287/trsc.2018.0878.

[17] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959. doi: 10.1287/mnsc.6.1.80. URL https://doi.org/10.1287/mnsc.6.1.80.

[18] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101. URL https://doi.org/10.1287/opre.8.1.101.

[19] Guy Desaulniers, Jacques Desrosiers, Irina Ioachim, Marius M. Solomon, and Daniel Villeneuve. *A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems*. Springer US, 1998.

[20] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[21] Martin Desrochers. *La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes*. Université de Montréal, Centre de recherche sur les transports, 1986.

[22] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342. URL https://doi.org/10.1287/opre.40.2.342.

[23] Jacques Desrosiers, Paul Pelletier, and François Soumis. Plus court chemin avec contraintes d'horaires. *RAIRO-Operations Research*, 17(4):357–377, 1983.

[24] Jacques Desrosiers, François Soumis, and Martin Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[25] Robert Dial, Fred Glover, David Karney, and Darwin Klingman. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks*, 9(3):215–248, 1979.

[26] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[27] Moshe Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978, 1994. URL https://EconPapers.repec.org/RePEc:inm:oropre:v:42:y:1994:i:5:p:977-978.

[28] Yvan Dumas, Jacques Desrosiers, and Francois Soumis. The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22, 1991.

[29] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[30] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multicommodity flow problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 184–193. IEEE, 1975.

[31] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4or*, 8(4):407–424, 2010.

[32] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[33] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 300–309, 1998.

[34] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[35] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin Of the American Mathematical Society*, 64:275–278, 1958.

[36] Frederick S Hillier. *Introduction to operations research, tenth edition*. Tata McGraw-Hill Education, 2012.

[37] John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[38] David J. Jun. Houck, Jean Claude Picard, Maurice Queyranne, and R. R. Vemuganti. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch*, 17(2):93–109, 1980.

[39] Irina Ioachim, Sylvie Gelinas, Francois Soumis, and Jacques Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks: An International Journal*, 31(3):193–204, 1998.

[40] Erwin Kalvelagen. Benders decomposition with gams. 2002.

[41] Erwin Kalvelagen. Lagrangian relaxation with gams. 2002.

[42] Erwin Kalvelagen. Dantzig-wolfe decomposition with gams. *Amsterdam Optim Model Group LLC, Washington, DC, USA*, 2003.

[43] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35(2):266–273, 1987. doi: 10.1287/opre.35.2.266. URL https://doi.org/10.1287/opre.35.2.266.

[44] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[45] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the Acm*, 22(4):469–476, 1975.

[46] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[47] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989, 1963. doi: 10.1287/opre.11.6.972. URL https://doi.org/10.1287/opre.11.6.972.

[48] C. E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Acm*, 7(4):326–329, 1960.

[49] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79 – 102, 2016. ISSN 1572-5286. doi: https://doi.org/10.1016/j.disopt.2016.01.005. URL http://www.sciencedirect.com/science/article/pii/S1572528616000062.

[50] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[51] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[52] Robert S Russell and Bernard W Taylor-Iii. *Operations management along the supply chain*. John Wiley & Sons, 2008.

[53] Jeremy F Shapiro. Mathematical programing: structures and algorithms. Technical report, 1979.

[54] Hanif D Sherali and Amine Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global optimization*, 2(4):379–410, 1992.

[55] Lawrence V Snyder and Zuo-Jun Max Shen. *Fundamentals of supply chain theory*. Wiley Online Library, 2019.

[56] Solomon and M. Marius. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 1987.

[57] Zeki Caner Taşkin. *Benders Decomposition*. American Cancer Society, 2011. ISBN 9780470400531. doi: 10.1002/9780470400531.eorms0104. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470400531.eorms0104.

[58] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898718515. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.

[59] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[60] I-Lin Wang. Multicommodity network flows: A survey, part i: Applications and formulations. *International Journal of Operations Research*, 15(4):145–153, 2018.

[61] Wayne L Winston. *Operations research: applications and algorithms*, volume 3. 2004.

[62] Wayne L Winston and Jeffrey B Goldberg. *Operations research: applications and algorithms*, volume 3. Thomson/Brooks/Cole, 2004.

[63] Laurence A Wolsey. *Integer programming*, volume 52. John Wiley & Sons, 1998.

[64] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913, 2018.

[65] F Benjamin Zhan and Charles E Noon. Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.

[66]《运筹学》教材编写组. 运筹学. 2012.

[67] 胡运权，郭耀煌. 运筹学教程. 2012.

[68] 陈景良，陈向晖. 特殊矩阵. 清华大学出版社, 2010.