杭州电子科技大学

硕 士 学 位 论 文

题 目：堆场作业的两个优化模型与算法

研 究 生：李伟娟

专 业：运筹学与控制论

指导教师：陈光亭 教授

陈 永 副教授

完成日期：2018 年 3 月

杭州电子科技大学硕士学位论文

堆场作业的两个优化模型与算法

研 究 生： 李伟娟

指导教师： 陈光亭 教授

陈 永 副教授

2018 年 3 月

Dissertation Submitted to Hangzhou Dianzi University for the Degree of Master

TWO OPTIMIZATION MODELS AND ALGORITHMS IN YARD PLANNING

Candidate: Weijuan Li

Supervisor: Prof. Guangting Chen

Associate Prof. Yong Chen

March, 2018

杭州电子科技大学

学位论文原创性声明和使用授权说明

独创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：　　日期：　　年　　月　　日

学位论文使用授权说明

本人完全了解杭州电子科技大学关于保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属杭州电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为杭州电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。(保密论文在解密后遵守此规定)

论文作者签名：　　日期：　　年　　月　　日

指导教师签名：　　日期：　　年　　月　　日

摘 要

本文研究集装箱堆场作业中两个组合优化问题，其一是并行堆装载问题，其二是集装箱转运问题。重点讨论上述两个问题的计算复杂性（多项式可解或NP-困难性），整数规划模型的建立，近似算法的设计及算法近似比的理论分析，以及为满足实际需求而设计的启发式算法，并通过计算机编程加以实现。

全文共分为四章，每一章的具体内容如下。

第一章，首先给出了组合优化问题的有关定义，接着介绍计算复杂性与近似算法及算法性能比等概念，最后对本文所研究的堆场作业中的两个优化问题进行了详细描述。

第二章，主要研究并行堆装载问题，即给定一组带有优先级的到达顺序已定的集装箱，需要暂时贮存在若干平行堆中，已知集装箱i的优先级为P_i,集装箱j的优先级为P_j，不妨假设$i < j$，$P_i < P_j$，即集装箱i先于集装箱j到达。若在某种装载方案下，集装箱i和j恰好被分派到同一个堆栈中，且集装箱j刚好位于集装箱i的上方，则此时集装箱i和j产生一个堵塞，我们的目标是寻求一个堵塞数最少的装载方案。针对这个问题，我们首先给出了它的数学规划模型。当平行堆数$s = 2$且堆栈高度不限定时，设计了求解该问题的多项式时间最优算法。当平行堆数$s = 2$且堆栈高度限定时，利用划分问题进行规约，证明了该问题是NP-困难的。最后，通过计算机编程进行了数值实验以评估求解该问题所提出的启发式算法的性能。

第三章，主要研究了集装箱转运问题，货运列车上的集装箱需要通过起重机转运到卡车上，集装箱从货运列车到卡车有一定的转运时间，目标是寻求总转运时间最少的转运方案。该问题可转为为二部图中最小完美匹配子集权问题，针对子集个数为2的情形，我们首先建立了其整数规划模型接着利用奇偶划分问题进行规约，证明了该问题是NP-困难的，接着设计了最坏情况界为$\frac{3}{2}$的多项式时间最小权重优先算法，最后提出了最坏情况界为$\frac{4}{3}$的改进近似算法。

第四章，对前三章所完成的工作，进行简单概括并且展望未来研究方向。

关键词： 整数规划，完美匹配，近似算法，最坏情况界，计算复杂性

ABSTRACT

In this thesis, two combinatorial optimization problems in container yard operations are studied. One is the parallel stack loading problem, the other is the container transshipment problem.We mainly focus on the computational complexity of the above two problems (polynomial time solvable or NP-hard), the integer linear programming model, the design of approximation algorithms and its performance ratio analysis, as well as Heuristic algorithm which was proposed to meet the actual needs of application and was tested through computer programming.

The full thesis is divided into four chapters, the specific contents of each chapter are as follows.

The first chapter gives the definition of combinatorial optimization problems,then introduces the concept of computational complexity, approximation algorithm,worst-case ratio and so on,and finally provide detailed description for these two optimization problems in yard planning.

The second chapter studies parallel stack load problem, given a set of containers with arrival order which were temporarily stored in several parallel stacks,each container i has a priority p_i. Assume that $p_i < p_j$ and $i < j$,i.e. container i arrived before container j. In a loading scheme,if container i and j are assigned to the same stack and container j is just above on the container i, then there is a blockage between container i and j.Our goal is to find a loading scheme such that the total number of blockages is minimized. We first formulate the integer programming model.If the number of stacks $s = 2$ and height of stack is unbounded,polynomial time optimal algorithm is proposed. If the parallel number of $s = 2$, the stack height is bounded,the problem is proved to be NP-hard via a reduction from partition problem. Finally,Heuristic algorithm is provided and its performance is evaluated through computer programming.

The third chapter studies the container transshipment problem.In rail-road terminals,gantry cranes transship containers between trains and trucks. The positions of containers and parking slots are predefined and known.The problem is to assign each container to a parking slot so that the maximum sum of transportation time of the component will be minimal.This problem may be modeled in terms of the Min-max weighted perfect match-

ing problem in the bipartite graph. If the number of the component is equal to 2,we first formulate the integer programming and prove it is NP-hard via a reduction form odd-even partition problem, then minimum weight based approximation algorithm with worst-case ratio $\frac{3}{2}$ is proposed and improved approximation algorithm with worst-case ratio $\frac{4}{3}$ is obtained.

Chapter IV summarizes the main contents of the first three chapter and puts forward future study direction.

Keywords: Integer Programming,Perfect Matching, Approximation Algorithm, Worst Case Ratio, Computational Complexity

目 录

1 绪论

1.1 组合优化问题

组合优化问题属于运筹学中最重要的分支之一，包含了组合数学、线性规划以及算法理论的方法和技巧。组合优化（或称离散优化）是指研究决策变量只取某些离散数值（通常为有限多个）的优化问题的一门学科。它在过去的一段时期内获得了非常好的发展，因为它提供了从远距离通信到超大规模集成电路、从产品运销到航空公司的航班机组排班等领域的困难问题解决办法。组合优化中的著名代表主要有：设施选址问题、网络优化问题、排序问题、旅行售货商问题、可满足性问题、集合的覆盖问题、背包问题等[1, 2]。

本文主要在网络优化的基础上研究并行堆装载与货车转运的组合优化问题。下面给出网络优化的一些相关定义：

定义 1.1 [3] 一个无向图G可由一个有序二元组$< V,E >$表示,记作$G =< V,E >$,其中$V \neq \emptyset$，V中的元素称为结点；E是无序积$V\&V$的多重子集，E称作G的边集，E中的元素称为无向边或简称边。

定义 1.2 [4] 一个无向简单图记为$G =< V,E >$，假如任意两个结点之间都有边相连，则称G为完全图，具有n个结点的完全图记作K_n。

定义 1.3 [3, 4] 一个无向图记作$G =< V,E >$，假如能将点集V分成V_1和V_2（$V_1 \bigcup V_2 = V, V_1 \bigcap V_2 = \emptyset$）,使得$G$中的每条边的两个端点均是一个在$V_1$中,另一个在$V_2$中，则称$G$为二部图（或者称为二分图等），其中子集$V_1$和$V_2$称为互补结点子集，常将二部图$G$记为$< V_1, V_2, E >$,而$< V_1, V_2, E >$也称作为$G =< V,E >$的二部划分形式。又若$G$是简单二部图，$V_1$中任意一个结点均与$V_2$中全部结点相邻，则称$G$为完全二部图，记为$K_{r,s}$,其中$r = |V_1|$，$s = |V_2|$。

定义 1.4 [4, 5] 设H是带权图$G =< V,E,W >$的一个子图，H的每条边的权的和称为H的权。假如H是一条路P，则称它的权是路P的长。在赋权图中给定了结点u和v。如果结点u, v相连通，那么u到v可能有若干条路，这些路中一定有一条距离最短的路，称为从u到v的最短路。最短路的长也称作u 到v的距离，记为$d(u, v)$。

定义 1.5 [4, 5] 在一个非空无环图G中，$M \subseteq E(G), M \neq \emptyset$,如果$M$中任意两条边在$G$中均不相邻，那么把$M$记为图$G$的一个匹配。如果对图$G$的任何匹配$M'$,都有$|M'| < |M|$,那么称$M$是图$G$的最大匹配。$G$中的最大匹配中的边数称为匹配数，记为$\eta(G)$。

定义 1.6 [5] 设M是图G的匹配，G中与M中的边关联的结点称作M饱和点，不然称作非M饱和点；如果图G中的结点均为M饱和点，那么称M为G的完美匹配。

1.2 计算复杂性与近似算法

1.2.1 算法的时间复杂性

由于组合优化中的实际问题规模巨大，不存在具体的求解公式，故需要设计出合理的求解程序。我们把一个问题的解的程序称为算法[1],算法是解决问题的步骤，一个问题的解决方法可能有多个，那么不一样的算法之间从某一方面来对比就有了好坏之分。如何对算法进行比较呢? 算法可以针对不同的维度进行比较，比如易读性、健壮性、可维护性、可扩展性等，但这些均不是最重要的，很多时候，我们主要考量的是算法的执行效率。算法的灵魂是解决问题的速度[6]。

我们往往利用算法的时间复杂性函数去评估算法的运行效率，把问题例子输入长度的函数记为算法的时间复杂性函数。具体定义为：对于某一个问题π和任何输入长度，比如为n,把用给定算法求解π的所有大小为n的例子所花费的时间的最大值称为该算法在输入长度为n时的复杂性。目前，在计算机领域有这样的一个约定：仅当算法的时间复杂性函数伴随问题例子输入长度的增加而呈现多项式地增长时，我们才认为这个算法是实际有用的。根据这个观点，我们把算法分为两大类[6]。

在介绍两大类算法的定义之前，首先引入记号$O(\cdot)$来表示算法复杂性,在正整数集上分别定义两个正实值函数$f(n)$和$g(n)$,如果有两个正常数$c > c'$,使得当n 无限大时有$c'g(n) \leq f(n) \leq cg(n)$,那么记$f(n) = O(g(n))$。依据$O(\cdot)$可以把函数分为不同的类，有下述定义：

定义 1.7 [7] 若存在某个用输入长度n作为变量的多项式函数$p(n)$,使该函数的时间复杂性函数记为：$O(p(n))$的算法，则称为多项式时间算法（polynomial time algorithm）。比如，常见的有：复杂性为$O(n)$、$O(9^5n^2)$、$O(7n^5)$等的算法都是多项式时间算法。

定义 1.8 [7] 任意时间复杂性函数不会像上边用多项式函数去界定的算法，称为指数时间算法。典型的指数时间算法有：$O(2^n)$、$O(n!)$、$O(n^n)$、$O(2^{n^2})$等。

1.2.2 问题的计算复杂性

在问题是否有有效算法的分析中，如果把任意的问题都转化成只要求给出“是”或“否”的形式，将会带来非常多的便利，使得差别很大的各类问题进行对比成为可能[8]。我们称答案只有两个“是”和“否”，或者“Yes”和“No”的问题为判定问题。

定义 1.9 [9] P类是指所有多项式时间可解的问题组成的问题类。

一个判定问题是易解的当且仅当它属于P类。但大部分判定问题，现在不但没有找到多项式时间算法，而且没能证明难解的一类问题所对应的判定问题有什么样的难度。对于这些判定问题虽然我们一方面没有证明它们属于P类、另一方面也没有证明它们不属于P类，但是项式时间可验证的这一特点它们是共有的。

定义 1.10 [9] 设判定问题$\pi =< D, Y >$,若有两个输入变量的多项式时间算法A与多项式p,对任何一个实例$I \in D$,$I \in Y$当且仅当存在t,$|t| \leq p(|I|)$,且A对输入I和t输出“Yes”，那么称π为多项式时间可验证的，A是π的多项式时间验证算法，而当$I \in Y$时，称t是$I \in Y$的证据。由多项式时间可验证的判定问题构成的问题类称作NP类。

由上述定义1.9及定义1.10可知，P类问题总是有多项式时间的最优算法，NP类问题总能够在多项式时间内验证最优解。容易得到$P \subseteq NP$,但是多项式时间可验证性并不可以确保多项式时间可解性。

NP难问题不会比NP中的任意问题简单，而NP完全问题（NPC）是NP中最困难的问题，现在给出定义：

定义 1.11 [10] 如果对所有的$\pi' \in NP$,$\pi' \leq_p \pi$，则称π是NP难的。如果π是NP难的且$\pi \in NP$,那么称π是NP完全的。

根据定义1.11可得，若有NP完全问题$\pi \in P$,则NP类中的全部问题将都可以在多项式时间内可解，也就是有$P = NP$成立。

虽然“$P = NP$?”至今还没有解决，但研究人员普遍相信$P \neq NP$,因而NP完全性成为表明一个问题很可能是难解的（不属于P）有力证据。在$P \neq NP$的假定下，$P, NP, NPC, NP - hard$这四类问题之间的关系可用图1.1来表示。

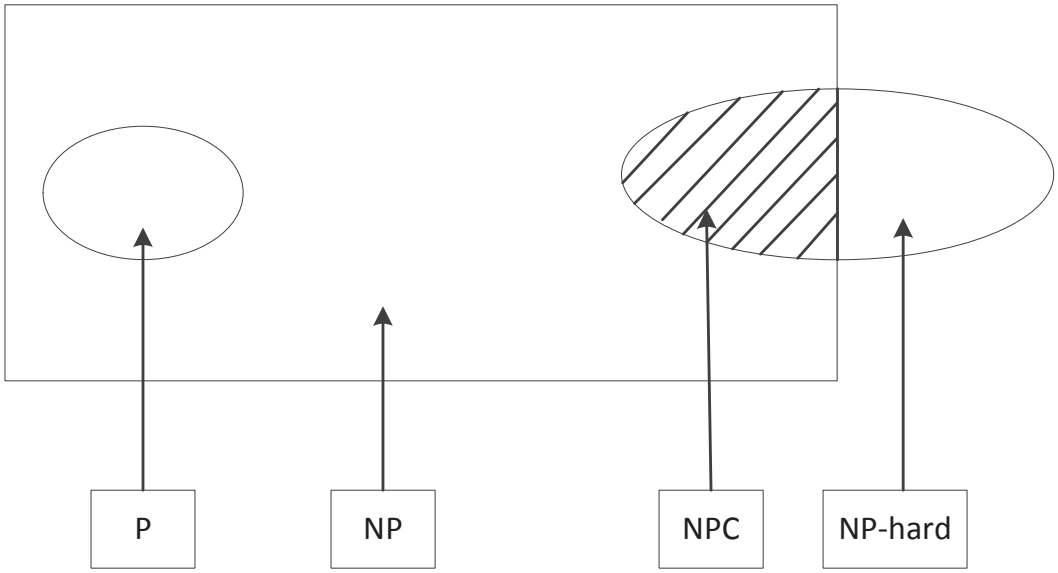

图 1.1 计算复杂性关系图

1.2.3 近似算法

因为*NP*完全问题具有难于求解性的特点，一种可行而有效的策略是在有限的时间内寻找问题尽可能好的近似最优解。所以近似算法或近似算法与某个精确算法的结合使用是求解*NPC*的关键方法。本节定义参考[7, 10]。

下面给出组合优化问题的正式描述：

一个组合优化问题π是极小化问题和极大化问题其中之一，它的三要素分别为：

1. 例子的一个集合D_π。
2. 对任意一个例子$I \in D_\pi$,有I的可行解得一个有限集合$S_\pi(I)$。
3. 有一个目标函数m_π,它对任意一个例子$I \in D_\pi$及任意一个可行解$\sigma \in S_\pi(I)$赋予一个正的有理数$m_\pi(I,\sigma)$, 记作解σ的目标值。

如果π为一个极小（大）化问题，那么例子$I \in D_\pi$的最优解是一个可行解$\sigma^* \in S_\pi(I)$,它满足：对所有的$\sigma \in S_\pi(I)$, 有$m_\pi(I,\sigma^*) \leq (\geq) m_\pi(I,\sigma)$。以下用$OPT(I)$来表示$I$的最优目标值，即$m_\pi(I,\sigma^*)$ [10]。

定义 1.12 [10, 11] 对于一个优化问题π,如果给定任何例子$I \in D_\pi$,它均可找到某一个可行解$\sigma \in S_\pi(I)$。A所找到的该可行解得目标值$m_\pi(I,\sigma)$ 简记为$A(I)$。如果对所有的$I \in D_\pi$,均有$A(I) = OPT(I)$,那么称A为π的一个最优算法。

近似算法理论一般研究的是对于一种给定的算法（通常为简单可行的），评估出由它产生的算法解与问题的最优解所对应的目标函数的值之间的差异度，从而在某种给定的情况下，它可以评估出几种给定的算法的优劣。往往评估算法优劣的两个标准是：一个标准是计算时间，另外一个标准是性能比。下面给出近似算法A性能比的概念[7]。

定义 1.13 [7] 如果π为一个极小（大）化问题，定义比率$R_A(I)$为：

$$R_A(I) = \frac{A(I)}{OPT(I)}(R_A(I) = \frac{OPT(I)}{A(I)})$$

当$OPT(I)(A(I)) = 0$时，上式应理解为$A(I) = R_A(I)OPT(I)(OPT(I) = R_A(I)A(I))$。若对一切$I \in D_\pi$有$R_A(I) \leq \delta$成立，则称$A$是近似比为$\delta$的算法。

定义 1.14 [7, 10] 近似算法A的绝对性能比R_A定义为：

$$R_A = inf\{r \geq 1 : R_A(I) \leq r, \forall I \epsilon D_\pi\}。$$

定义 1.15 [7, 11] A的渐近性能比R_A^∞则由下式定义：

$R_A^\infty = inf\{r \geq 1$：存在某个$N \in Z^+$,对所有满足$OPT(I) \geq N$的$I \in D_\pi$,有$R_A \leq r\}$。

显然总有$1 \leq R_A \leq \infty$和$1 \leq R_A^\infty \leq \infty$，且比率越接近于1，那么表明近似算法的性能越好。

1.3 码头堆场作业模型

1.3.1 并行堆装载问题的优化模型

并行堆装载问题的优化模型是建立在车辆调度与货物装载运输这一实际背景下的优化模型，对于该模型的研究有非常重要的理论和实际意义。该模型描述如下：

通常，给定到达的一连串物品，例如，集装箱、车辆，它们贮存在并行堆里，每个堆有一个给定的最大高度。在物品贮存之后，把它们再重新取回，每个物品j有给定的权值p_j,p_j表示物品j的取回优先值。例如，若物品i和j的取回优先值分别为$p_i = 3$,$p_j = 1$,则物品j在i之前应优先取回。若此时同一堆的物品i在物品j顶部贮存，那么取回物品时将导致一个堵塞（也指一个错误的覆盖物或货物倒装，参看[12]）也就是为了取回物品j,物品i必须先移动。并行堆装载问题的目标是在给定

堆数的平行堆贮存到达的物品，在不违反堆的高度情况下，使得堵塞数是极小化的。本文主要研究高度不限，堆数$S = 2$时的并行堆装载问题。

并行堆装载问题在许多物流应用里作为一个核心问题，例如有轨汽车站停放的货车见图1.2.

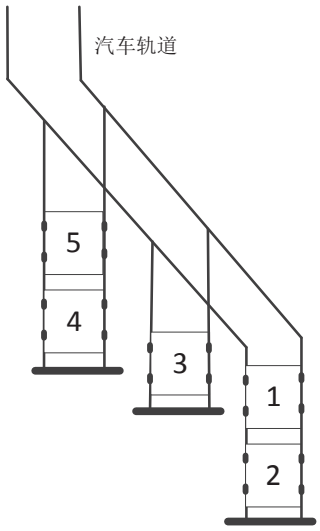

图 1.2 有轨汽车站货车停放图

并行堆装载在有轨货车上的应用，如果把物品当作轨道货车，在晚上它们相继到达头尾轨道（堆）组成的仓库进行停放，在第二天早上货车离开的顺序当作为取回优先值p_j, 停放的过程当作为在堆上贮存货车，使得在早上驱动它们按照正确顺序离开的精力极小化。

1.3.2 货车转运问题的优化模型

在铁路集装箱中心站，起重机在货运列车和卡车之间转运集装箱[13]。若干台起重机将集装箱从货运列车车厢转运到卡车停靠区域，目的是使得总转运时间尽可能少。总转运时间只考虑起重机运输集装箱的时间[14]。铁路集装箱中心站的布局如图1.3所示，该问题可以归结为如下的网络优化模型。集装箱的集合记为U,第i台起重机需要转运的集装箱集合记为U_i,卡车的停靠区域由多个停靠点构成，记停靠点的集合为V。给定赋权完全二部图$G = G(U, V, E)$,其中U和V是顶点集，E是边集。每条边$e = (u, v) \in E, u \in U, v \in V$, 边权记为$w(e)$。集合$U$被划分成$m$个不相交的子集$U_1, U_2, \cdots, U_m$，对于任何一个最大匹配$M$，定义子集$U_i$的权重为$w(U_i, M) = \sum_{u \in U_i, (u,v) \in M} w(u, v)$(简记$w(U_i)$), 匹配$M$ 的子

集为$w(M) = \max_{1\le i\le m} w(U_i, M)$。目标是寻求具有最小子集权的最大匹配[15]。本文研究$|U| = |V|$且$m = 2$的二部图中的完美匹配子集权的极小化问题。

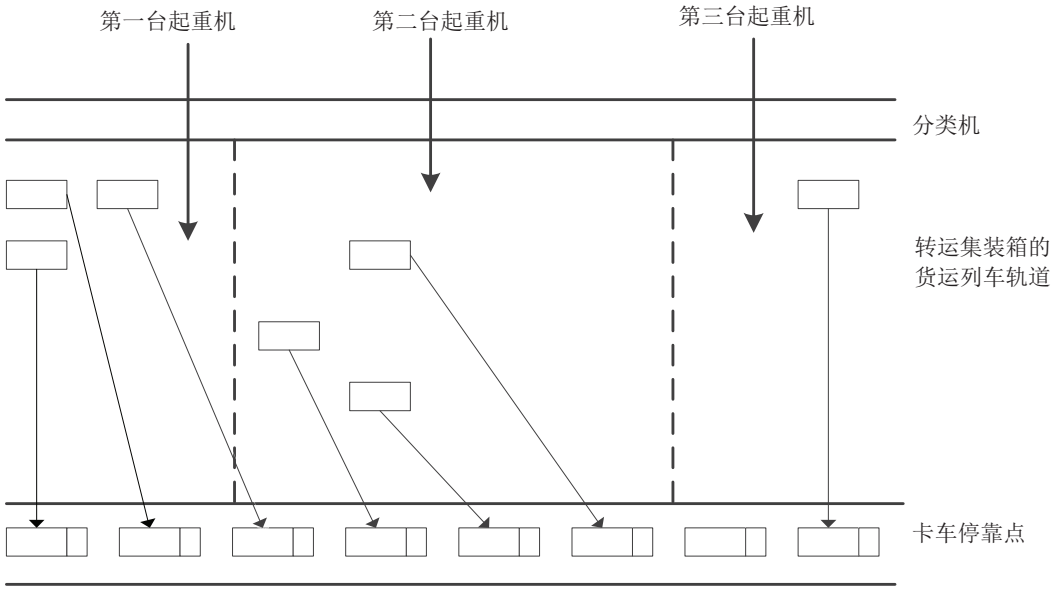

图 1.3 铁路集装箱中心站的布局

1.4 论文结构

本文总共分为四章。

第一章，绪论部分，首先介绍了组合优化问题的一些基本概念，图论与网络的相关定义，其次给出了算法的基本内容，最后具体描述了码头堆场作业模型。

第二章，本章主要研究了并行堆装载问题。关于此问题，我们给出了数学规划模型，设计了贪心算法。当堆数$s = 2$，高度不限定时，证明了问题是可解的，贪心算法是最优的。接着在$s = 2$,高度限定的情况下证明了问题的NP-难解性。最后，我们进行了数值实验评估算法的性能。

第三章，我们主要研究了货车转运问题，并把它转化为二部图中完美匹配子集权的网络问题来分析。对完美匹配两个子集权的极小化问题，设计了最小权重优先算法，且给出了算法的最坏情况界为$\frac{3}{2}$。在这个基础上设计了最坏情况界至多为$\frac{4}{3}$的改进算法。

第四章，对全文进行归纳与总结，并指出了相关的堆场作业问题的进一步有待研究的问题。

2 并行装载问题的优化模型与算法

2.1 问题描述及研究现状

2.1.1 并行堆装载问题

有n个集装箱按照到达顺序贮存，由集装箱集合J的排序方式定义达到顺序。不失一般性，假设集装箱按照一个接一个的顺序计数，这样，第一个贮存的集装箱记为$j=1$, 第二个贮存的集装箱记为$j=2$，以此类推直到最后一个集装箱$j=n$。贮存港口由S个堆组成（$s=1,\cdots,S$）且每个堆高度有T层（$t=1,\cdots,T$）,集装箱贮存在堆里。显然，有$T \geq n$,为了方便起见，假设最初所有堆都是空的。$j \in J$的每个集装箱给定一个确定的取回优先值p_j。p_j值越小，表明集装箱j越早取回。若一些集装箱的取回优先值为空，则假设这些集装箱的取回优先值是相同的，使得不贮存或没有优先值约束。注意不同的集装箱或许有同样的取回优先值。在现实里，这种情况可能出现，例如，两个集装箱由同一辆货车装载，这两个集装箱的装载顺序是无关紧要的。

由元组(j,t,s)组成的并行堆装载问题的贮存计划Ω，其中$j \in J$，$t \in \{1,\cdots,T\}$，且$s \in \{1,\cdots,S\}$，指集装箱j贮存在t层s堆。贮存计划Ω 的可行性说明：

- $j \in J$的每个集装箱由唯一一个$(j,t,s) \in \Omega$,也就是每个集装箱只能贮存在港口的唯一一个位置。
- 对于每对$(j,t,s) \in \Omega$和$(j',t',s') \in \Omega$其中$j \neq j'$，我们有$s \neq s'$且/或$t \neq t'$，即每个集装箱得到唯一贮存位置。
- 对于每个$(j,t,s) \in \Omega$,我们有$|\{(j',t',s') \in \Omega : j > j' \Lambda t > t' \Lambda s = s'\}| = t-1$,即集装箱不能漂浮，也就是，当$s$ 堆，$t-1$层的同一堆贮存优先值顺序靠前的集装箱时，集装箱j也只能贮存在s堆，t层。

堵塞的定义：对于任意一对集装箱$(j,t,s) \in \Omega$和$(j',t',s') \in \Omega$，集装箱j 在集装箱j'上面贮存，也就是$t = t'+1$且在同一堆$s = s'$。若有$p_j > p_{j'}$，则集装箱j必须比集装箱j'取回晚，那么记集装箱j堵塞集装箱j'。目标在于划分成S堆，每堆至多有T层，使得集装箱的堵塞数极小，简记为PSLP问题。我们研究堆数$S=2$，高度T 层不限，使得集装箱的堵塞数极小化。

2.1.2 相关研究现状

重新安置集装箱和重新取回集装箱的文献基本上分为两个部分，依据何时执行重新取回操作：一方面，一些问题允许重新安置和重新取回操作同时发生[16],称为BRP问题。但是，在另一方面，第二类问题分成两种情形，先重新安排集装箱，使堵塞极小化[17]，接下来，再执行重新取回集装箱的操作。为了避免复杂的装载情况下重新安置集装箱和重新取回集装箱同时发生，我们应用取代目标使得在取回集装箱过程中重新安置的精力极小化，取代目标为集装箱的堵塞数极小化。

我们的并行堆装载问题在有轨货车上的应用介绍见1.3.1所述。然而，现存的研究关注于堆场的释放问题且考虑同样类型的货车可以互换[18]，Winten等人处理了实际时间的调度[19]，或进一步使真实世界作为整体的研究可参考[20]和[21]，例如不同大小的货车和轨道，复杂的时间表约束和从两个方向获取的轨道。Boysen[22]等人提供关于排序过程的更详细描述。Lehnfeld[12]等人不仅总结处理装载的所有相关文献，而且提供了一个分类表，简单地定义不同的问题。他们所列出的所有问题清单，表明在文献研究里PSLP还未研究。Nils Boysen[17]等人证明了PSLP问题是强NP-hard，提出了整数规划并设计了启发式算法。此外其他有关研究看综述文献[15, 23–27]，有关问题研究参考相关文献[28–45]。

2.2 数学规划模型

在给出并行堆装载问题的数学规划模型之前，先介绍几个符号：

$x_{0,j}$　当集装箱j在某个堆的最底部，则$x_{0,j}=1$，否则为0;

$x_{j,n+1}$　当集装箱j在某个堆的最顶部，则$x_{j,n+1}=1$，否则为0;

$x_{j,k}$　在同一堆里集装箱k在集装箱j上边，则$x_{j,k}=1$，否则为0;

$b_{j,k}$　当集装箱的取回优先值$P_k > p_j$且$k > j$时，则$b_{j,k}=1$，否则为0。

该问题的数学规划模型如下：

$$MinimizeZ(X,t)=\sum_{j=1}^{n}\sum_{k=j+1}^{n} b_{j,k}x_{j,k}. \tag{2.1}$$

$Subject\ to$:

$$\sum_{k=0,\cdots,j-1} x_{k,j}=1, j=1,\cdots,n. \tag{2.2}$$

$$\sum_{j=1}^{n} x_{0,j} \leq 2. \tag{2.3}$$

$$\sum_{k=j+1}^{n+1} x_{j,k} = 1, j = 1, \cdots, n. \tag{2.4}$$

$$t_j + 1 - (n+1)(1 - x_{j,k}) \leq t_k, k = 1, \cdots, n, j = 0, \cdots, k-1. \tag{2.5}$$

$$1 \leq t_j \leq \frac{n}{2}, j = 1, 2, \cdots, n. \tag{2.6}$$

$$x_{j,k} \in \{0, 1\}, j = 1, \cdots, n+1, k = 0, \cdots, j. \tag{2.7}$$

$$x_{0,j} \in \{0, 1\}, j = 1, \cdots, n. \tag{2.8}$$

$$x_{j,n+1} \in \{0, 1\}, j = 1, \cdots, n. \tag{2.9}$$

目标函数(2.1)使堵塞数极小化，约束(2.2)确保每个集装箱安排1个贮存空间，约束(2.3)限制有效堆的数目。约束(2.2)、（2.4）、(2.5)确保每个集装箱有前继元和后继元且连续的集装箱在同一层是相邻的。约束（2.6）限制层数，约束(2.7)、(2.8)、(2.9)控制变量。利用Matlab将数学规划编程得到堆数$S = 2$，高度限定在$T = \frac{n}{2}$情况下所有情形的最小的堵塞数，主要代码见附录1。

2.3 贪心算法

1.用$\sigma = (p_n, p_{n-1}, \cdots, p_1)$记作集装箱的到达序列，其中$p_j(j\epsilon 1, \cdots, n)$中$j$记作集装箱到达顺序，$p_j$ 记作集装箱的取回优先值。为方便起见，令$p_0 = +\infty, p_{-1} = +\infty$分别放在$s = 1$堆和$s = 2$堆的最底部。

2.无论何时集装箱p_j到达，令p_a, p_b为两个堆此时最上边的集装箱，且使得$p_a < p_b$。若$p_a < p_j \leq p_b$,那么把p_j放在p_b上。否则，把p_j放在p_a上。

3.输出两个堆的堵塞数。

2.4 贪心算法的最优性证明（$S = 2, T = \infty$）

令An表示贪心算法处理n个集装箱的序列$\sigma_n = (p_n, p_{n-1}, \cdots, p_1)$产生的堵塞数，$OPT_n$表示最优算法处理$n$个集装箱产生的堵塞数。

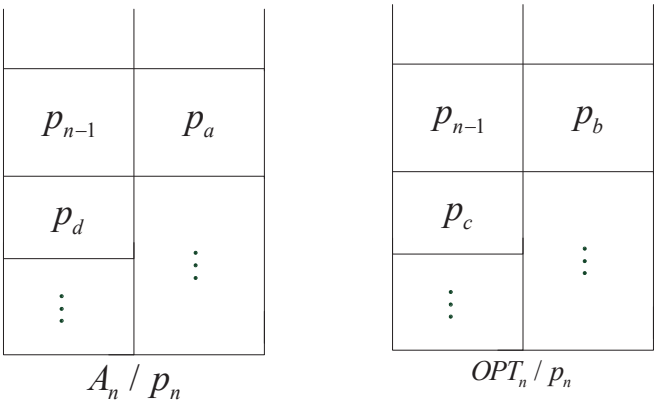

图 2.1 贪心算法解和最优解中去掉物品i_n

下证$A_n = OPT_n$。用反证法，记去掉集装箱p_n后的贪心算法解和最优解分别为A_{n-1}和OPT_{n-1}. 如图2.1所示。设σ_n为最小反例，即：$A_n > OPT_n$且$A_{n-1} = OPT_{n-1}$。所以p_n在A_n中必产生堵塞，而在OPT_n中不产生堵塞。由贪心算法的规则可知:

$$p_n > p_{n-1}. \tag{2.10}$$

$$p_n > p_a. \tag{2.11}$$

再由OPT_n可知：

$$p_n \leq p_b. \tag{2.12}$$

由（2.10）、(2.11)、（2.12）可知，有：

$$p_a < p_b. \tag{2.13}$$

根据贪心算法解以及最优解的不同取值分情况讨论如下：

情形1，若$p_a = p_{n-2}$,则$p_c = p_{n-2}$从而由（2.10）、(2.11)、（2.12）、(2.13)可知,有：

$$p_{n-2} < p_n \leq p_b. \tag{2.14}$$

情形1.1 如果$p_{n-1} > p_{n-2}$，因为在去掉集装箱p_n后$A_{n-1} = OPT_{n-1}$,若$p_{n-1} \leq p_d$时，此时OPT_n中p_{n-1}与$p_c = p_{n-2}$产生堵塞，而A_n中p_{n-1}与p_d不产生堵塞，若此时贪心算法和最优算法中都去掉集装箱p_{n-1},则$A_{n-2} > OPT_{n-2}$,产生更小反例，与最小反例矛盾，因此我们断言$p_{n-1} > p_d$。

再由贪心算法规则：$p_d \leq p_{n-2}$,即有$p_{n-1} > p_{n-2} \geq p_d$，也即：

$$p_b \geq p_n > p_{n-1} > p_{n-2} \geq p_d.$$

故在σ_n中删去p_{n-1}，记贪心算法解为A'_{n-1},最优解为OPT'_{n-1},则此时$A'_{n-1} = A_{n-1} > OPT_{n-1} - 1 \geq OPT'_{n-1}$，必定产生更小反例，与最小反例矛盾！

情形1.2 若$p_{n-1} \leq p_{n-2}$,则由贪心算法规则，必有：

$$p_{n-1} \leq p_d \leq p_{n-2}. \tag{2.15}$$

又由（2.14）、(2.15)可知：$p_{n-1} \leq p_d \leq p_{n-2} < p_n \leq p_b$ 若只去掉集装箱p_{n-1},此时A'_{n-1}的堵塞数与A_n的堵塞数相等，即有$A'_{n-1} = A_n > OPT_n \geq OPT'_{n-1}$，必定产生更小反例，矛盾！

情形2 若$p_a \neq p_{n-2}$,则p_{n-2}在p_{n-1}下面，依次找到p_{a+1}，如图2.2示。

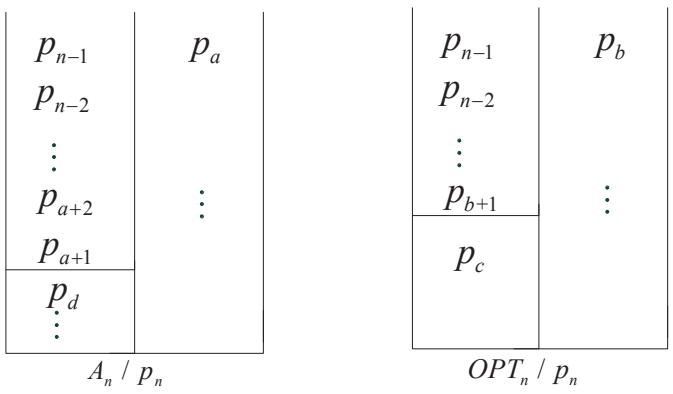

图 2.2

情形2.1 $b < a$.此时，由于$A_{n-1} = OPT_{n-1}$，若去掉$\{p_{a+1}, \cdots, p_n\}$的集装箱，记贪心算法解为A_{a+1},最优解为OPT_{a+1},从而$A_{a+1} = OPT_{a+1}$。若$p_{a+1} > p_a$时，如果此时$p_{a+1} \leq p_d$, 去掉$\{p_{a+1}, \cdots, p_n\}$的集装箱后，$A_a > OPT_a$,那么产生更小反例与最小反例矛盾。故我们断言，必有$p_{a+1} > p_d$。又由贪心算法规则，有$p_d \leq p_a$,此时去

掉$\{p_{a+1},\cdots,p_{n-1}\}$, 记贪心算法解为$A'_{a+1}$，最优解为$OPT'_{a+1}$,又因为$p_n > p_a \geq p_d$,$p_n \leq p_b$,所以$A'_{a+1} = A_{a+1} = OPT_{a+1} > OPT_a \geq OPT'_{a+1}$。可得更小反例，矛盾!

情形2.2 若$b \geq a$，如图2.3所示。

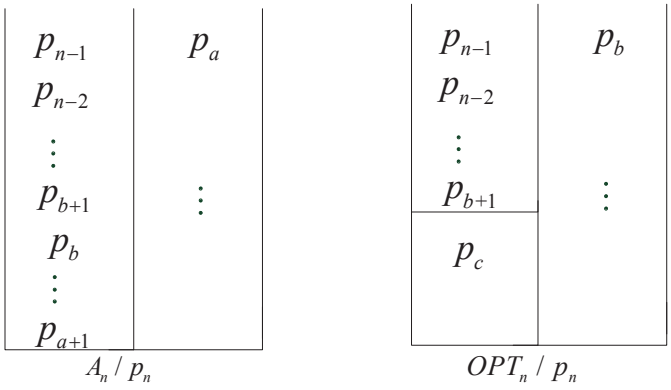

图 2.3

此时由于$p_a < p_b$,根据贪心算法规则，则必有$p_a < p_{n-1} < p_{n-2} < \cdots < p_{b+2} < p_{b+1} < p_b$。

记$X = \{p_{b+2},\cdots,p_{n-2}\}$,若$x \neq \Phi$,去掉集装箱集合$X$,记贪心算法解为$A'$, 最优解为$OPT'$,则$A' = A_n > OPT_n \geq OPT'$，得到更小反例，矛盾!

若$x = \Phi$,如图2.4所示。

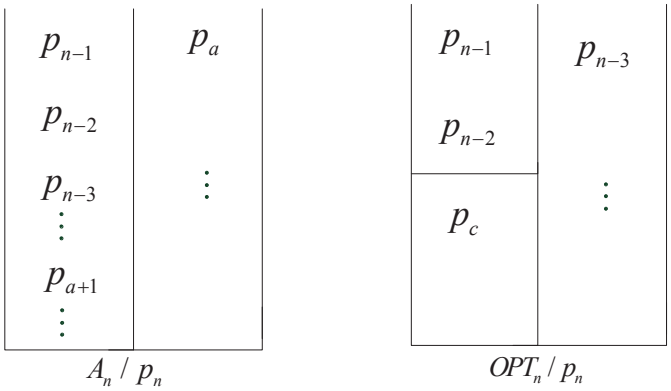

图 2.4

即：$p_{b+1} = p_{n-2}, p_b = p_{n-3}$，此时：$p_a < p_{n-1} < p_{n-2} < p_{n-3}$ 而$p_a < p_{n-1} < p_n \leq p_{n-3}$ 去掉集装箱p_n同时交换p_{n-1}与p_{n-2}的次序，因为$A' = A_n > OPT_n \geqslant OPT_{n-1} \geq OPT'$，所以得到更小反例，矛盾!

定理 2.1 $S = 2, T$固定时并行装载问题是NP-hard。

证明：等基数划分问题[46],给定n个正整数的集合$N = \{a_1, a_2, \cdots, a_n\}$及正整数$B$,且满足$\sum_{j=1}^{n} a_j = 2B$。问是否能将集合$N$划分成两个集合$N_1$和$N_2$,使得$|N_1| = |N_2| = \frac{n}{2}, N_1 \bigcup N_2 = N, N_1 \cap N_2 = \Phi$, 并且$\sum_{j \epsilon N_1} a_j = B, \sum_{j \epsilon N_2} a_j = B$。

等基数划分问题是NP-完全问题，对任意给定等基数划分实例，构造并行装载问题的实例如下：并行堆堆数$S = 2, T = B$，集装箱按照到达顺序的取回优先值为：$p_j = \{1, \cdots, 1, 2, \cdots, 2, \cdots, n, \cdots, n\}$。令$a_1 = \{1, \cdots, 1\}$，有$a_1$个1，$a_2 = \{2, \cdots, 2\}$,有$a_2$个2，……，$a_n = \{n, \cdots, n\}$, 有$a_n$个$n$。问并行装载问题是否存在堵塞总数小于等于$n-2$的可行装载方案?

如果等基数划分问题有解，即存在两个集合N_1和N_2，满足$|N_1| = |N_2| = \frac{n}{2}, N_1 \bigcup N_2 = N, N_1 \cap N_2 = \Phi$且$\sum_{j \epsilon N_1} a_j = B, \sum_{j \epsilon N_2} a_j = B$，则此时把$N_1$集合中相应的元素依次放入$s = 1$堆中，$N_2$集合中相应的元素依次放入$s = 2$堆中，即得到堵塞总数为$n-2$的一个可行装载方案。

如果等基数划分无解，则此时必有某个a_j相应的集装箱被分派到两个不同的堆中，于是其中一个堆中集装箱的堵塞数一定大于或等于$\frac{n}{2}$，即堵塞总数一定大于等于$n-1$，则此时不存在堵塞数小于等于$n-2$的可行装载方案。

下面给出等基数划分问题有解的一个实例，给定6个正整数集合$N = \{a_1, a_2, a_3, a_4, a_5, a_6\}$，其中$a_1 = 2, a_2 = 3, a_3 = 4, a_4 = 4, a_5 = 5, a_6 = 6$，将集合$N$分成$N_1, N_2$，使得$|N_1| = |N_2| = 3$,取$N_1 = \{a_6, a_4, a_1\} = \{6, 4, 2\}, N_2 = \{a_5, a_3, a_2\} = \{5, 4, 3\}$, 此时$\sum_{j \epsilon N_1} a_j = 12, \sum_{j \epsilon N_2} a_j = 12$。将该实例构造为并行装载问题实例，堆数$s = 2$,层数$T = 12$,集装箱到达顺序的优先值如图2.5所示。

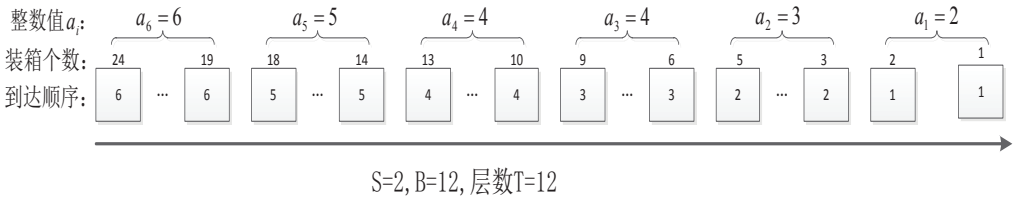

图 2.5 并行装载例子

等基数划分问题有解的同时并行装载问题的堵塞数极小化，如图2.6所示，堵塞数为$y = 6 - 2 = 4$。

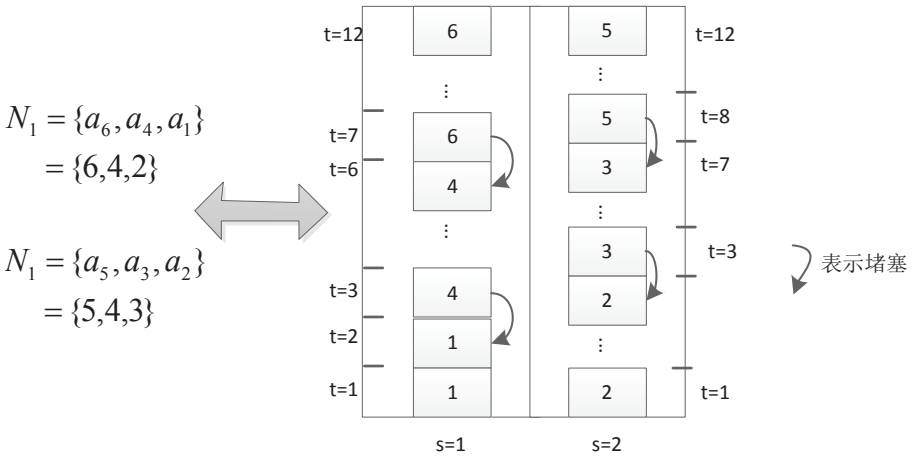

图 2.6 等基数划分与并行装载关系的例子

2.5 数值实验

为了评估我们设计的贪心算法*GA*的性能，根据算法的描述，我们通过Matlab软件工具从算法性能比的角度计算了算法产生的结果。因此，为了验证该算法的实际有效性，实验中，我们分别以2堆、3堆、5堆、10堆为例，高度限定为$\frac{n}{2}$，通过模拟不同的集装箱数及随机产生的取回优先值，给出了多组实验结果（见表2.1），具体实现的主要代码见附录2。其中，*Max*记作实验中产生的竞争比最大值，*Min*记作产生的竞争比最小值，*Avg*记作产生的竞争比平均值。

表 2.1 $S = 2$并行堆装载数值实验结果

集装箱数(n)	$\frac{C^{GA}}{C^*}$			实验次数(N)
	Max	Min	Avg	
10	3.000000	1.000000	1.219167	100
20	1.666667	1.000000	1.096298	100
50	2.000000	1.000000	1.048909	100

表 2.2 $S=3$并行堆装载数值实验结果

集装箱数(n)	$\frac{C^{GA}}{C^*}$			实验次数(N)
	Max	Min	Avg	
10	2.000000	1.000000	1.075000	100
20	1.000000	1.000000	1.000000	100
50	1.000000	1.000000	1.000000	100

表 2.3 $S=5$并行堆装载数值实验结果

集装箱数(n)	$\frac{C^{GA}}{C^*}$			实验次数(N)
	Max	Min	Avg	
10	1.000000	1.000000	1.000000	100
20	1.000000	1.000000	1.000000	100
50	1.000000	1.000000	1.000000	100

表 2.4 $S=10$并行堆装载数值实验结果

集装箱数(n)	$\frac{C^{GA}}{C^*}$			实验次数(N)
	Max	Min	Avg	
10	1.000000	1.000000	1.000000	100
20	1.000000	1.000000	1.000000	100
50	1.000000	1.000000	1.000000	100

数值实验中，对于贪心算法，我们随机生成100个实例，计算了算法所产生的最大误差界，最小误差界，平均误差界。表2.1、2.2、2.3、2.4分别为2堆、3堆、5堆、10堆时并行装载数值实验结果。

尽管我们已从理论上证明了，堆数为2，高度不限定时，贪心算法是最优的，但从以上4个表中，我们发现数值实验中，该算法实例所得两堆、三堆的情形，实际性能却比其他两种情况要差，问题可能出现在堆数越多堵塞数减少，由于实验中高度限定为$\frac{n}{2}$与$T=\infty$的情况类似，所以堆数越多，最坏情况界越接近于1。

2.6 本章小结

为服务于车辆调度与货物装载运输，本章主要研究了2堆，不限高的并行堆装载问题。我们的目标是极小化堵塞数。对此问题，我们不仅给出了问题的数学规划模型，设计了贪心算法。还给出了在堆数$S=2$,T固定时NP-hard证明。进一步，为了验证算法的实际性能，特地给出了数值实验的结果。

3 货车转运问题的优化模型与算法

3.1 问题描述及研究现状

在这一章中，我们主要研究$|U| = |V|$的二部图中的完美匹配子集权的极小化问题。对于铁路集装箱中心站里货车转运集装箱的结构描述同第一章。集装箱的集合记为U，在第i台起重机需要转运的集装箱集合记为U_i，卡车的停靠区域由多个停靠点构成，记停靠点的集合为V。对于$m = 2$的情形进行研究，目标仍然是使得总转运的时间尽可能少。

当$m = 1$时，该问题等价于经典指派问题[47]，可在$O(n^3)$时间内求解[48]。当$m = |U| = |V|$时，该问题等价于瓶颈型指派问题[49–52]，存在$O(n^2\sqrt{n/\log n})$时间的最优算法。当m为输入时，文献证明了该问题不存在最坏情况界小于2的近似算法。而在网络结构为非二部图的任意图时，即使$m = 1$的问题也是NP- 难的[53]，文献还给出了一个2-近似算法。本文研究$|U| = |V|$的二部图中的完美匹配子集权的极小化问题，对$m = 2$的情形设计了近似算法。

3.2 数学规划模型

在给出货车转运集装箱问题的数学规划模型之前，先介绍几个符号：

$x_{i,j}$　　当集装箱i由起重机转运到卡车j上，则$x_{ij} = 1$,否则为0;

$w(i, j)$　　表示集装箱i由起重机转运到卡车j上的时间;

Z　　表示目标函数的值;

该问题的数学规划模型如下：

$$Minimize Z. \tag{3.1}$$

$$\sum_{j\epsilon V} x_{i,j} = 1, \forall i, i\epsilon U. \tag{3.2}$$

$$\sum_{i\epsilon U} x_{i,j} = 1, \forall j, j\epsilon V. \tag{3.3}$$

$$\sum_{i\epsilon U_1, j\epsilon V} w(i,j)x_{i,j} \leq Z. \tag{3.4}$$

$$\sum_{i\epsilon U_2, j\epsilon V} w(i,j)x_{i,j} \leq Z. \tag{3.5}$$

...

$$\sum_{i\epsilon U_m, j\epsilon V} w(i,j)x_{i,j} \leq Z. \tag{3.6}$$

$$x_{i,j}\epsilon\{0,1\}, \forall i, i\epsilon U, j\epsilon V. \tag{3.7}$$

目标函数（3.1）是转运时间极小化，约束（3.2）和（3.3）确保每个集装箱被转运到一个卡车上，一个卡车上只能装载一个集装箱。约束（3.4）和（3.5）、（3.6）确保每台起重机转运完集装箱的时间不超过C,约束（3.7）控制变量。

3.3 二部图完美匹配子集权问题

在铁路集装箱中心站起重机从货车转运集装箱到卡车的问题可以归结为网络问题如下。给定一个赋权二部图$G = (U, V, E)$，U和V中均有$2n$个顶点，点集U被划分成不相交的$m = 2$个子集U_1和U_2且$|U_1| = |U_2| = n$。V中任意一个顶点均有两条边，一条在U_1中，另外一条在U_2 中，且同一个顶点$v_i(i = 1, \cdots, 2n)$的两条边权重相等，即$w(v_i, u_i) = w(v_i, u_{n+i}) = w(e_i), (i = 1, \cdots, n)$，$w(v_{n+i}, u_{n+i}) = w(v_{n+i}, u_i), (i = 1, \cdots, n)$。目标是寻求具有最小子集权的完美匹配$M^*$。

定理 3.1 二部图完美匹配子集权问题是NP-hard。

证明：奇偶划分问题[46]，给定$2n$个正整数的集合$N = \{a_1, a_2, a_3, a_4, a_5, a_6, \cdots, a_{2n-1}, a_{2n}\}$, 和正整数$B$，满足$\sum_{i=1}^{2n} a_i = 2B$。问是否能将集合$N$ 划分成两个集合N_1和N_2，使得任意一组$\{a_i, a_{i+1}\}$,要么$a_i \in N_1$,且$a_{i+1} \in N_2$;要么$a_i \in N_2$，且$a_{i+1} \in N_1$，且$N_1 \bigcup N_2 = N, N_1 \bigcap N_2 = \Phi$，。

奇偶划分问题是NPC问题，对任意给定奇偶划分实例，构造二部图完美匹配子集权问题的实例如下：给定一个赋权二部图$G = (U, V, E)$，U和V中均有$2n$个

顶点，$\sum_{i\in N_1} a_i = \sum_{i\in N_2} a_i = B$。点集$U$被划分成不相交的$m = 2$个子集$U_1$和$U_2$且$|U_1| = |U_2| = n$。$V$中任意一个顶点均有两条边，一条在$U_1$中，另外一条在$U_2$ 中，且同一个顶点$v_i(i = 1,\cdots,2n)$的两条边权重相等，即$w(v_i,u_i) = w(v_i,u_{n+i}) = w(e_{2i-1}) = a_{2i} - 1,(i = 1,\cdots,n)$; $w(v_{n+i},u_{n+i}) = w(v_{n+i},u_i) = w(e_{2i}) = a_{2i},(i = 1,\cdots,n)$。如图3.1所示，该二部图中是否存在一个完美匹配$M^*$，满足$max\{\sum_{i\in U_1} w(e_i)$fi $\sum_{i\in U_2} w(e_i)\} \le B$，

如果奇偶划分问题有解，则把N_1中的集合相应元素所对应的边也都选入集合U_1对应匹配$M_1^{'}$,N_2中的边选入另一个集合U_2对应的匹配$M_2^{'}$。$\sum_{i\in N_1} a_i = \sum_{i\in N_2} a_i = \sum w(M_1^{'}) = \sum w(M_2^{'}) = B$。

若该问题存在一个子集权重小于等于B的完美匹配，则U_1对应的匹配$M_1^{'}$所选的边对应的a_i 记为集合N_1的元素,U_2对应的匹配$M_2^{'}$所选的边所对应的a_i记为集合N_2的元素。由于$\sum w(M_1^{'}) \le B$,$\sum w(M_2^{'}) \le B$,而$\sum w(M_1^{'}) + \sum w(M_2^{'}) = 2B$,所以$\sum w(M_1^{'}) = \sum w(M_2^{'}) = B$。故综上所述，二部图完美匹配子集权问题是NP-hard。

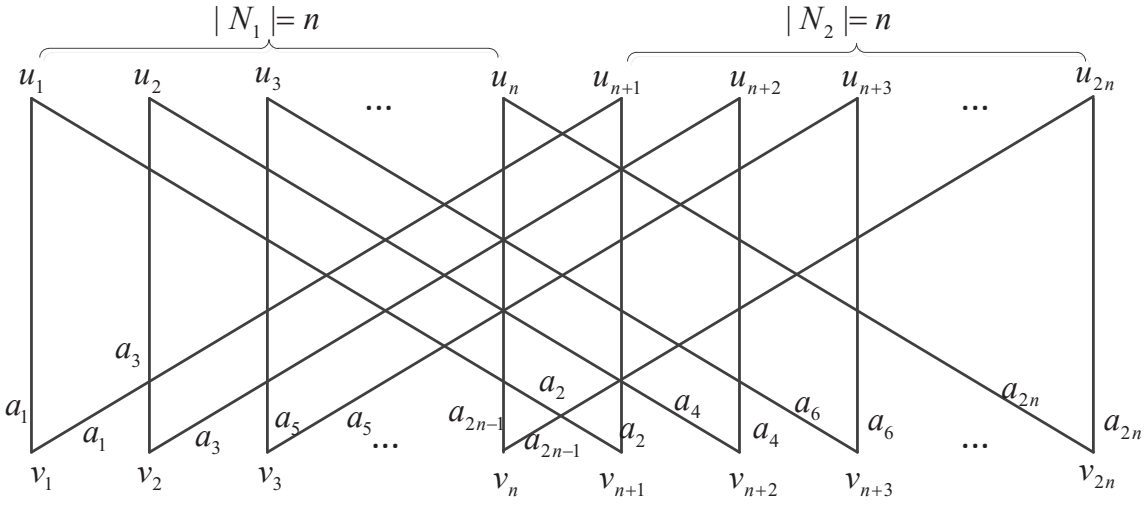

图 3.1 相关的奇偶划分示意图

3.4 算法设计与分析

本节设计$m = 2$情形的最小权重优先算法，并证明最坏情况界为$\frac{3}{2}$，给出紧例，并应用一一互换思想，设计了最坏情况界至多为$\frac{4}{3}$的改进算法。

最小权重优先算法（SWF算法）

步骤1：将权重按照从小到大的顺序$w(e_1) \le w(e_2) \le \cdots \le w(e_{2n})$;

步骤2：从权重最小的边开始，依次放入当前边权和最小的子集U_1或者U_2里;

引理 3.1 按照SWF算法，最终得到两个集合U_1和U_2里边的条数一定相等。

证明： 反证法。若集合U_1和U_2里边的条数不相等，一定存在一条边e_m，不妨设e_m在U_2，在e_m放入U_2之前，记$w(U_2) = s_m$，$w(U_1) = r$分别表示子集U_2，U_1的权重，则有：

$$s_m + w(e_m) < r. \tag{3.8}$$

令$d = r - s_m \leq w(e_{m-1})$，又因边权按照从小到大排序，即$w(e_{m-1}) \leq w(e_m)$，因而$d \leq w(e_m)$，故$s_m + w(e_m) \geq r$与(3.1)矛盾，所以引理得证。 □

定理 3.2 算法SWF的最坏情况界不超过$\frac{3}{2}$且是紧的。

证明： 记M_1为算法SWF得到的完美匹配，设边权$w(e_{2n})$放入子集U_2时，U_2中边权和记为s_{2n}，则有:

$$\begin{aligned} w(M_1) = s_{2n} + w(e_{2n}) &\leq \frac{\sum_{j=1}^{2n-1} w(e_j)}{2} + w(e_{2n}) \\ &= \frac{\sum_{j=1}^{2n} w(e_j)}{2} - \frac{w(e_{2n})}{2} + w(e_{2n}) \leq \frac{\sum_{j=1}^{2n} w(e_j)}{2} + \frac{w(e_{2n})}{2}. \end{aligned}$$

又由于$w(M^*) \geq \frac{\sum_{j=1}^{2n} w(e_j)}{2}$，$w(M^*) \geq w(e_{2n})$,所以：

$$w(M_1) \leq w(M^*) + \frac{w(M^*)}{2} = \frac{3w(M^*)}{2}.$$

□

下面的实例说明$\frac{3}{2}$的界是紧的：$|U_i| = 2(i = 1, 2)$，U_1有两个点u_1, u_2。U_2有两个u_3, u_4。V中有4个点分别记为v_1, v_2, v_3, v_4。各边的权重分别为：

$$\begin{aligned} &w(e_1) = w(v_1, u_1) = w(v_1, u_3) = \varepsilon, w(e_2) = w(v_2, u_2) = w(v_2, u_4) = 1, \\ &w(e_3) = w(v_3, u_3) = w(v_3, u_1) = 1, w(e_4) = w(v_4, u_4) = w(v_4, u_2) = 2. \end{aligned}$$

最优解如图3.3所示，$w(M^*) = \varepsilon + 2$，而算法解如图3.4所示，$w(M_1) = 3$。所以当$\varepsilon \longrightarrow 0$时, $\frac{w(M_1)}{w(M^*)} = \frac{3}{\varepsilon+2} \longrightarrow \frac{3}{2}$。

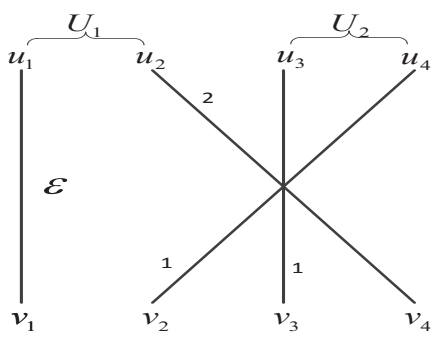

图 3.2 最优解

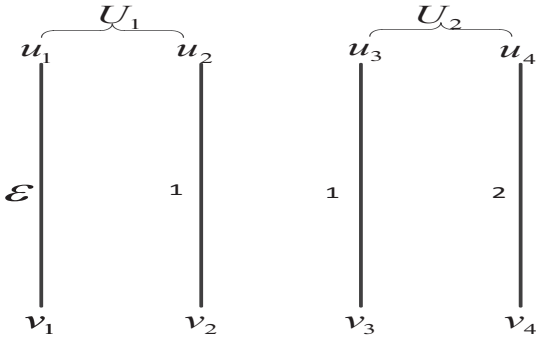

图 3.3 算法解

下面我们给出改进的一一互换算法。令U_1, U_2是$\{e_1, e_2, \cdots, e_{2n}\}$的一个划分且$|U_i| = n(i = 1, 2)$，不妨假设$w(U_q) < w(U_p)$，$p \neq q$。一一互换定义如下：设$e_r \in U_p$，$e_s \in U_q$，当$0 < w(e_r) - w(e_s) < w(U_p) - w(U_q)$时，令$U_p = U_p \bigcup \{e_s\} \backslash \{e_r\}$，$U_q = U_q \bigcup \{e_r\} \backslash \{e_s\}$，称为一次一一互换，易知，互换后的匹配最大权只可能减少。

互换算法

步骤1：将U任意划分成边数相等的两个子集$U_i(i = 1, 2)$；

步骤2：不断进行上述的一一互换过程，直至不能进行互换为止。

定理 3.3 互换算法的最坏情况界至多是$\frac{4}{3}$。

证明： 记M_2为互换算法得到的完美匹配，假定$w(M^*) = 1$($w(e_i)$除以$w(M^*)$)。因此对于划分U_1，U_2有：

$$w(U_1) + w(U_2) = \sum_{i=1}^{2n} w(e_i) \leq 2. \tag{3.9}$$

假设$w(U_1) = 1+\alpha$且$w(U_1)$是子集权和最大者，$w(U_2)$是子集中权和较小者。记$\Delta_{12} = w(U_1) - w(U_2)$，$\delta_{12} = min\{w(e_r) - w(e_s) | w(e_r) - w(e_s) > 0, e_r \in U_1, e_s \in U_2\}$。若$U_1, U_2$已不能进行一一互换，那么$\Delta_{12} < \delta_{12}$，根据（3.2）我们可得：

$$\Delta_{12} = w(U_1) - w(U_2) = 2w(U_1) - (w(U_1) + w(U_2)) \geq 2(1 + \alpha) - 2 = 2\alpha.$$

所以$\Delta_{12} > 2\alpha$，令$U_1 = \{e_1, \cdots, e_n\}$且$w(e_1) \geq w(e_2) \geq \cdots \geq w(e_n)$；$U_2 = \{e'_1, \cdots, e'_n\}$且$w(e'_1) \geq w(e'_2) \geq \cdots \geq w(e'_n)$，因为$w(U_1) > w(U_2)$，所以$w(e_1) > w(e'_2)$且$w(e_1) > \delta_{12}$。因为$w(e_2) \geq \delta_{12}$，所以$w(U_1) = 1+\alpha \geq w(e_1)+w(e_2) \geq 2\delta_{12} \geq 4\alpha$，即$\alpha \leq \frac{1}{3}$，$\frac{w(U_1)}{w(M^*)} = \frac{1+\alpha}{1} \leq \frac{4}{3}$，定理3.3得证。 □

3.5 本章小结

本章研究了在铁路中心站起重机从货运列车转运集装箱到卡车的问题，目标是转运集装箱的时间极小化，本章研究$m = 2$的情形，并设计最小权重优先算法，证明该算法的最坏情况界为$\frac{3}{2}$且是紧的。其次，利用一一互换思想，设计了最坏情况界至多为$\frac{4}{3}$的改进算法。针对后续的研究，可以设计近似算法来实现m更大时的问题求解。

4 结 论

本文中，主要研究了堆场作业的两个优化问题。一个是并行装载问题，一个是起重机转运集装箱问题。

在第一个问题中，堆数$s = 2$，高度不限情况下，我们设计了多项式时间的最优算法。另外，高度限定时，证明了此问题是NP-hard，设计了求解此问题的贪心算法，并通过计算机编程模拟以评价该算法的实际性能。

在第二个问题中，我们把问题归结为网络模型来处理，研究了$m = 2$时的二部图完美匹配子集权问题。我们证明了最小权重优先算法的最坏情况界为$\frac{3}{2}$，并用一一互换思想，设计了最坏情况界至多为$\frac{4}{3}$的改进算法。

至此，我们仍有很多问题值得去研究:前者问题中堆数s作为输入时，高度限定情况下，设计近似算法。后者问题中$m > 2$时，二部图中完美匹配子集权问题的近似算法设计与分析。

致　谢

光阴似箭，日月如梭，紧张而充实的研究生学习生活就要结束了，而复试通过时的情景还历历在目，初来杭州时的场景记忆犹新。想起这两年来的一幕幕，心情比较复杂，既有黑夜里的挑灯苦读，也有和师兄妹的嬉戏打闹。值得无悔的是两年多的时间在汗水和奋斗中度过，受益匪浅；庆幸的是我报考了杭电，遇到了一生的好老师，给了我很多的指引和帮助。使我的人生历程更加精彩。

首先，衷心感谢我尊敬的导师陈光亭教授。不管是在学习上还是在生活上，陈老师就像一位慈祥而又严厉的父亲教育我们，让我们不断成长。告诉我们对待学习切忌浮躁，要多多思考。以他的丰富阅历教导我们，鼓励我们。在今后的岁月里，陈老师的教育依然会影响着我。

其次，在此对陈永老师和张安老师在我的科研道路上给予的巨大帮助表示真心的感谢。陈永老师在出国访问期间，依然不辞辛苦，不顾时差，带我讨论问题，鼓励我继续努力。在此期间我在研究中遇到问题或有新的想法，也会第一时间找张安老师讨论，他不论多忙、多累都会放下手中的工作耐心听我的一些想法并给予指导。曾两次带我参加学术会议，使我有幸一睹国内外教授老师的风采，拓宽了视野，感受了浓厚的学术氛围。

同时，感谢我的师兄张文帅、师姐陈蕾，是你们让我更快融入这个大家庭。感谢同门杨帆、李丹和我一起学习，探讨问题，互帮互助。感谢师妹王翼展，师弟张雷、宣鑫乐陪我度过了一段难忘而又愉快的研究生生活。另外，还有感谢所有15级的同学们，因为有你们的陪伴，使我的研究生生活多姿多彩。

感谢我的家人与朋友们，谢谢你们对我的默默付出。

感谢对论文进行评审并提出宝贵意见的各位专家。

最后，感谢杭州电子科技大学给我提供了非常好的学习氛围。

参考文献

[1] 越民义, 李荣珩. 组合优化导论[M]. 科学出版社, 2014.

[2] 张学良，史永堂.组合优化[M].高等教育出版社，2011.

[3] 张清华.图论及其应用[M].清华大学出版社，2013.

[4] 张先迪，李正良.图论及其应用[M].高等教育出版社，2005.

[5] 王朝瑞.图论（第三版）[M].北京理工大学出版社，2007.

[6] 王红梅，胡明.算法设计与分析（第二版）[M].清华大学出版社，2013.

[7] 陈志平, 徐宗本. 计算机数学: 计算复杂性理论与 NPC. NP-难问题的求解[M]. 科学出版社, 2001.

[8] 姚恩瑜，何勇，陈仕平.数学规划与组合优化[M].浙江大学出版社，2001.

[9] 屈婉玲，刘田.算法设计与分析（第二版）[M].清华大学出版社，2016.

[10] M. H. Alsuwaiyel著.吴伟昶，方世昌等译.算法设计与分析[M].电子工业出版社，2010.

[11] 张立昂.可计算性与算法复杂性导引（第三版）[M].北京大学出版社，2011.

[12] J. Lehnfeld, S. Knust. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification[J]. European Journal of Operational Research, 2014, 239(2): 297-312.

[13] M.Barketau,E.Pesch,Y.Shafransky. Minimizing maximum weight of subsets of a maximum matching in a bipartite graph[J]. Discrete Applied Mathematics, 2015, 196: 4-19.

[14] N.Boysen,M.Fliedner. Determining crane areas in intermodal transshipment yards: The yard partition problem[J]. European Journal of Operational Research, 2010, 204(2): 336-342.

[15] N.Boysen,M.Fliedner,F.Jaehn, et al. A survey on container processing in railway yards[J]. Transportation Science, 2013, 47(3): 312-329.

[16] M.Caserta,S.Schwarze,S.Voß. A mathematical formulation and complexity considerations for the blocks relocation problem[J]. European Journal of Operational Research, 2012, 219(1): 96-104.

[17] N.Boysen,S.Emde. The parallel stack loading problem to minimize blockages[J]. European Journal of Operational Research, 2016, 249(2): 618-627.

[18] U.Blasum,M.R.Bussieck,W.Hochstättler, et al. Scheduling trams in the morning[J]. Mathematical Methods of Operations Research, 1999, 49(1): 137-148.

[19] T.Winter,U.T.Zimmermann. Real-time dispatch of trams in storage yards[J]. Annals of Operations Research, 2000, 96(1): 287-315.

[20] R.Freling,R.M.Lentink,L.G.Kroon, et al. Shunting of passenger train units in a railway station[J]. Transportation Science, 2005, 39(2): 261-272.

[21] P.M.Jacobsen,D.Pisinger. Train shunting at a workshop area[J]. Flexible services and manufacturing journal, 2011, 23(2): 156-180.

[22] N.Boysen,M.Fliedner,F.Jaehn, et al. Shunting yard operations: Theoretical aspects and applications[J]. European Journal of Operational Research, 2012, 220(1): 1-14.

[23] N.Boysen,A.Scholl,N.Wopperer. Resequencing of mixed-model assembly lines: Survey and research agenda[J]. European Journal of Operational Research, 2012, 216(3): 594-604.

[24] N.Boysen,M.Zenker. A decomposition approach for the car resequencing problem with selectivity banks[J]. Computers & Operations Research, 2013, 40(1): 98-108.

[25] M.Demange,T.Ekim,D.de Werra. A tutorial on the use of graph coloring for some problems in robotics[J]. European Journal of Operational Research, 2009, 192(1): 41-55.

[26] D.E.Knuth. The art of computer programming, 3rd edn. seminumerical algorithms, vol. 2[J]. 1997.

[27] M.Bóna. A survey of stack-sorting disciplines[J]. the electronic journal of combinatorics, 2003, 9(2): A1.

[28] D.Di Stefano,S. Krause, Lübbecke M E, et al. On minimum k-modal partitions of permutations[J]. Journal of Discrete Algorithms, 2008, 6(3): 381-392.

[29] G.Zäpfel,M.Wasner. Warehouse sequencing in the steel supply chain as a generalized job shop model[J]. International journal of production economics, 2006, 104(2): 482-501.

[30] L.Tang,R.Zhao,J.Liu. Models and algorithms for shuffling problems in steel plants[J]. Naval Research Logistics (NRL), 2012, 59(7): 502-524.

[31] T.Nishi,M.Konishi. An optimisation model and its effective beam search heuristics for floor-storage warehousing systems[J]. International Journal of Production Research, 2010, 48(7): 1947-1966.

[32] M.Caserta,S.Schwarze,S.Voß. Container rehandling at maritime container terminals[J]. Handbook of terminal planning, 2011: 247-269.

[33] A.Bortfeldt,F.Forster. A tree search procedure for the container pre-marshalling problem[J]. European Journal of Operational Research, 2012, 217(3): 531-540.

[34] M.Caserta,S.Voß,M.Sniedovich. Applying the corridor method to a blocks relocation problem[J]. OR spectrum, 2011, 33(4): 915-929.

[35] K.H.Kim,G.P.Hong. A heuristic rule for relocating blocks[J]. Computers & Operations Research, 2006, 33(4): 940-954.

[36] R.Dekker,P.Voogd,E.van Asperen. Advanced methods for container stacking[J]. OR spectrum, 2006, 28(4): 563-586.

[37] B.Borgman,E.van Asperen,R.Dekker. Online rules for container stacking[J]. OR spectrum, 2010, 32(3): 687-716.

[38] K.H.Kim,Y.M.Park,K.R.Ryu. Deriving decision rules to locate export containers in container yards[J]. European Journal of Operational Research, 2000, 124(1): 89-101.

[39] J.Kang,K.R.Ryu,K.H.Kim. Deriving stacking strategies for export containers with uncertain weight information[J]. Journal of Intelligent Manufacturing, 2006, 17(4): 399-410.

[40] A.H.Gharehgozli,Y.Yu,R.de Koster, et al. A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal[J]. International Journal of Production Research, 2014, 52(9): 2592-2611.

[41] Y.H.Han,C.Zhou. Dynamic sequencing of jobs on conveyor systems for minimizing changeovers[J]. The International Journal of Advanced Manufacturing Technology, 2010, 49(9-12): 1251-1259.

[42] B.I.Kim,J.Koo,H.P.Sambhajirao. A simplified steel plate stacking problem[J]. International Journal of Production Research, 2011, 49(17): 5133-5151.

[43] Y.Lee,S.L.Chao. A neighborhood search heuristic for pre-marshalling export containers[J]. European Journal of Operational Research, 2009, 196(2): 468-475.

[44] D.Steenken,S.Voß,R.Stahlbock. Container terminal operation and operations research-a classification and literature review[J]. OR spectrum, 2004, 26(1): 3-49.

[45] C.Zhang,W.Chen,L.Shi, et al. A note on deriving decision rules to locate export containers in container yards[J]. European Journal of Operational Research, 2010, 205(2): 483-485.

[46] M.R.Garey,D.S.Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness[M].Las Vegas:Freeman,1979

[47] R.E.Burkard,M.Dell'Amico,S.Martello. Assignment Problems, Revised Reprint[M]. Siam,2009.

[48] C.H.Papadimitriou,K.Steiglitz. Combinatorial optimization: algorithms and complexity[M]. Courier Corporation, 1982.

[49] G.Carpaneto,P.Toth. Algorithm for the solution of the bottleneck assignment problem[J]. Computing, 1981, 27(2): 179-187.

[50] U.Derigs,U.Zimmermann. An augmenting path method for solving linear bottleneck assignment problems[J]. Computing, 1978, 19(4): 285-295.

[51] R.S.Garfinkel.Technical note -an improved algorithm for the bottleneck assignment problem[J].Operations Research,1971,19(7):1747-1751.

[52] O.Gross. The bottleneck assignment problem[R]. RAND CORP SANTA MONICA CALIF, 1959.

[53] T.Fujito,H.Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem[J]. Discrete Applied Mathematics, 2002, 118(3): 199-207.

附录1–数学规划主要程序代码

```
function [ret] = IP(a, n, S)
    b = zeros(n, n);
    for j = 1:n
        for k = j+1:n
            if a(k) > a(j)
                b(j,k) = 1;
            end
        end
    end
    f = zeros(1, (n + 2)^2 + n);
    for j = 1:n
        for k = j+1:n
            f(1, j*(n+2)+(k+1)) = b(j,k);
        end
    end
    A = zeros(n*(n+1)/2+1, (n + 2)^2 + n);
    B = zeros(n*(n+1)/2+1, 1);
    idx = 1;
    for j = 1:n
        A(1, j + 1) = 1;
    end
    B(1, 1) = S;
    for k = 1:n
        for j = 0:k-1
            idx = idx + 1;
            A(idx, (n+2)^2 + j) = 1;
            A(idx, (n+2)^2 + k) = -1;
            A(idx, j*(n+2)+(k+1)) = 1;
            B(idx, 1) = n;
```

```
        end
    end
    aeq = zeros(2*n, (n + 2)^2 + n);
    for j = 1:n
        for k = j+1:n+1
            aeq(j, j*(n+2)+(k+1)) = 1;
        end
    end
    for j = 1:n
        for k = 0:j-1
            aeq(j+n, k*(n+2)+(j+1)) = 1;
        end
    end
    beq = ones(2*n, 1);
    lb = zeros((n+2)^2+n, 1);
    ub = ones((n+2)^2+n, 1);
    for j = 1:n
        lb((n+2)^2+j, 1) = 1;
        ub((n+2)^2+j, 1) = n / 2;
    end
    intcon = zeros(1, (n+2)^2 + n);
    for j = 0:n+1
        for k = 0:n+1
            intcon(j*(n+2)+(k+1)) = j*(n+2)+(k+1);
        end
    end
    for j = 1:n
        intcon((n+2)^2+j) = (n+2)^2+j;
    end
    [~, ret,~,~] = intlinprog(f, intcon, A, B, aeq, beq, lb, ub);
end
```

附录2–贪心算法主要程序代码

```
function [ret] = Greedy(a, n, S)
    ret = 0;up = n / 2;
    stk = (n + 1) * ones(1, S);
    dep = zeros(1, S);
    for i = 1:n
        now = n + 1;
        idx_now = 0;
        nxt = n + 1;
        idx_nxt = 0;
        for j = 1:S
            if (stk(j) > a(i))&&(stk(j) <= now)&&(dep(j) < up)
                now = stk(j);
                idx_now = j;
            end
            if (dep(j) < up)&&(stk(j) <= nxt)
                nxt = stk(j);
                idx_nxt = j;
            end
        end
        if idx_now ~= 0
            stk(idx_now) = a(i);
            dep(idx_now) = dep(idx_now) + 1;
        else
            ret = ret + 1;
            stk(idx_nxt) = a(i);
            dep(idx_nxt) = dep(idx_nxt) + 1;
        end
    end
end
```

附录

作者在读期间发表的学术论文及参加的科研项目

发表的学术论文：

[1] 李伟娟，陈永，陈光亭，张安.二部图中的完美匹配子集权的极小化问题[J].杭州电子科技大学学报，2017，37(5)：97-99.

参加的科研项目：

[1] 陈光亭.若干极小网络及其集成组合优化问题的算法研究，国家自然科学基金项目.课题编号:11571252.

[2] 张安. 集装箱港口作业驱动的排序模型与算法，国家自然科学基金项目.课题编号:11771114.

[3] 陈永.带运输和外包服务的供应链排序问题算法研究，国家自然科学基金项目.课题编号:11401149.

[4] 张安.若干新型调度模型的minsum目标问题，浙江省自然科学基金项目.课题编号:LY16A010015.