集装箱堆场智能优化研究

重庆大学硕士学位论文

学生姓名：李保顺

指导老师：易正俊　教　授

专　　业：应用数学

学科门类：理　学

重庆大学数学与统计学院

二〇一〇年十一月

Intelligent Optimization of Container Yard

A Thesis Submitted to Chongqing University

in Partial Fulfillment of the Requirement for the

Degree of Master of Science

By

Li Baoshun

Supervised by Prof. Yi Zhengjun

Major: Applied Mathematics

College of Mathematics and Statistics of

Chongqing University, Chongqing, China

November 2010

摘　　要

集装箱提箱效率是影响港口堆场工作效率，服务质量的主要因素，而提箱时的倒箱落箱位的选择直接影响着堆场发箱速度，甚至严重影响着装船系统的整体流畅性。

本文对一般性的集装箱取箱中的倒箱问题和出口集装箱的倒箱问题进行了深入研究，对于一般性的倒箱问题本文建立了一个博弈模型，把客户和码头看成博弈过程中的局中人，客户取走集装箱的时间作为局中人的策略，这些策略通过电子交换信息传递到码头，码头得到客户局中人的策略，从而得到码头集装箱被取走的优先级，通过建立码头局中人的效用函数，在未来时段倒箱次数最少和计算时间较少转换成寻求效用函数的最大，并通过启发式算法实现，提出了一种博弈启发式优化算法。实验结果表明该种算法的优化效果与目前国内外的文献的同类优化算法相比较均有明显的改善。

对于出口箱的倒箱问题本文以最小化倒箱总次数，尽量均衡每次发箱对应的倒箱次数、最大程度地保证装船系统的流畅性为约束条件，在深入分析倒箱操作实质以及提箱装船时堆存贝内倒箱问题的优化模型的基础上，实现了两种算法，首先利用预期倒箱数数的概念建立了一个出口箱控制优化算法，其次在corridor method（简称C-M）的基础上提出了一种基于改进的C-M的启发式优化算法，并通过实验对相关算法进行了比较，实验表明所提出的算法均在尽可能的均衡单次提箱倒箱次数的同时很大程度的优化了总倒箱数。

关键词：集装箱码头，堆场，博弈模型，C-M 模型，倒箱

ABSTRACT

Container yard suitcase is a major factor to the port efficiency and the service quality. And the choice of the down container location will affects the speed of container sending, and even affects the overall fluency loading system seriously.

In this paper, we had an in-depth study to the general problem of the container relocation and the export container. To the general problem of the containers relocation the paper presented a Game model for the relocation containers in the yard. We saw the customers and the terminal as the insider of the Game process, the time when the customers coming to pick up the containers as the insider's strategy. Their strategy passed to the terminal though the electronic exchange of information, when the terminal got the information, they also got the propriety of every container, then they proposed Game heuristic algorithm though establishing the utility function, which is achieved by converting the least relocation and the heuristic algorithm to the largest utility function and the heuristic algorithm. The result shows that the performance of the algorithm is better than other similar algorithms at home and abroad.

To the problem of export containers, the paper based on in-depth analysis of inverted containers and optimization model of the containers loading when operating in real terms, it took the minimize total number of inverted containers, tried best to balance the inverted number of each sending, the maximum smooth shipment of the loading system as the constraints, and proposed a heuristic optimization algorithm based on corridor method (short for C-M). The experiments showed that the algorithm balanced the down containers number of single suitcase, and at the same time it optimized the total number of inverted containers at a large degree.

Keywords： Container terminals, Yards, Game model, C-M model, Relocating

目　　录

1 绪　论

1.1 问题的提出背景与研究意义

改革开放以来，我国的海运和河运交通有了很大的发展，特别是随着我国经济快速持续发展，港口货物吞吐量以及集装箱吞吐量已连续几年位居世界第一。到 2010 年中国货运海运量将达到四十亿吨，占世界总量的的百分比将达到百分之五十七。随着水运货运量的不断增加，也给港口码头的泊位建设，机械调度、堆场管理等各方面带来了新的问题，为了满足货运量的需求，港口除了应对码头进行规划扩建，更要有效的整合现有资源，从整体上提高码头的作业效率，满足客户的需要。

集装箱堆场是港口的重要组成部分，堆场的主要任务就是集、散货物以及办理集装箱的装、拆等业务[1]。随着集装箱业务的不断发展，现在装箱堆场业务已由传统的货物装、拆箱扩展到集装箱空箱的堆存管理以及集装箱的公路中转运输等各个方面，这样就增加了集装箱堆场各环节管理的复杂度。为保证集装箱堆场的合理使用，以及高效率的装、卸、存、取的操作，应研究出一些切实有效地集装箱优化算法，以期尽可能避免在作业过程中产生不必要的人力、物力、空间等资源的浪费，提高堆场的作业效率，进而提高港口的吞吐量。因此在整个集装箱运输中，集装箱堆场的布局和操作方面的优化占据着极为重要的地位。

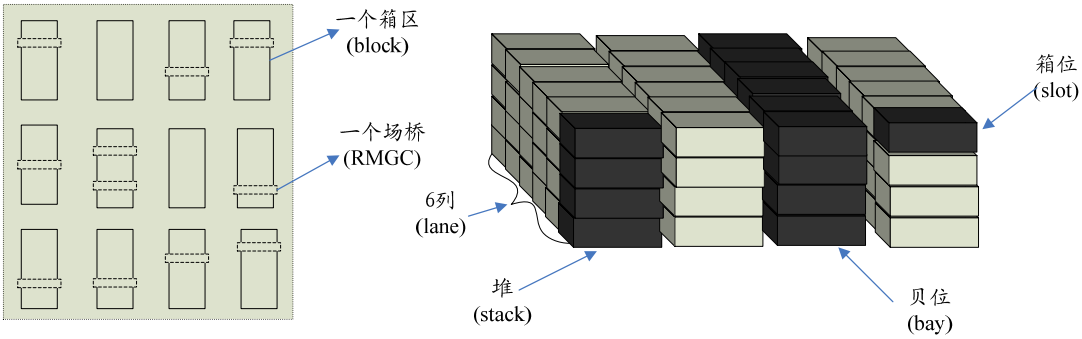

图 1.1 堆场结构[2]

Fig.1.1 Yard structure

在堆场操作中，客户取箱和集装箱装船操作占据重要地位。客户提取集装箱或集装箱装船时，取箱作业的效率是与集装箱堆存方式有着直接的关系的[3]。若将集装箱按理想的堆存顺序堆放在堆场——先提走的箱子堆放在上面，再配合有序

的机械运作，将是最合理的作业方式。但由于客户提取集装箱的顺序的随机性，这种方式很难实现，因此必须研究相关的倒箱算法以期尽可能地降低倒箱作业次数。

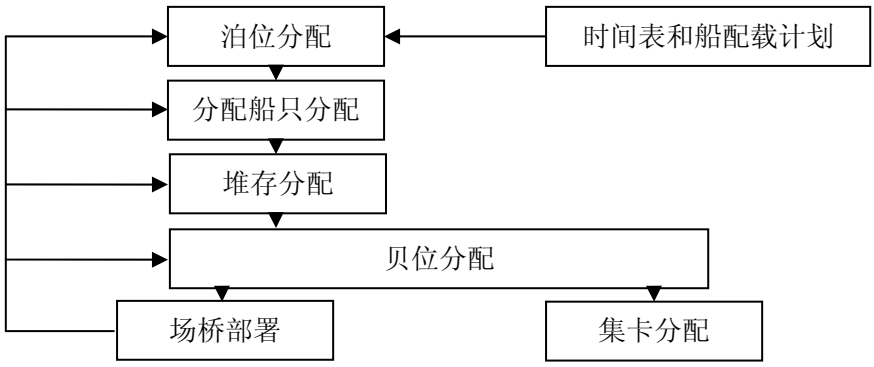

图 1.2 操作的决策层次

Fig.1.2 Decision hierarchy in operations

由于集装箱在堆场中是分层堆放的，由于集装箱到来时间的随机性以及在收箱时集装箱被提走的顺序的不确定性，致使提箱时不可避免地会出现一定数量的倒箱，如何有效避免或减少倒箱已成为提高堆场作业效率的关键问题。在集装箱堆场中，垂直堆放的一列集装箱称为一个栈，并排堆放的若干个栈组成一个贝。如果提取的集装箱不处于所在栈的最上层，就必须将堆放在其上的所有集装箱翻倒到其它栈，这一过程称为倒箱。随着堆场堆垛高度的增加，取箱时的倒箱概率会相应增加。当堆场面积有限，集装箱存放密度过高时，由于堆场箱位空间非常有限，不同的重量级、目的港、甚至不同船号的集装箱不得不混放在一个堆存贝内，集装箱的堆存状态通常不满足发箱顺序的要求；另外集装箱重量信息不准，航次或目的港临时改变，抽检或熏蒸后的集装箱不再放回原来箱位等随机因素的存在均会导致集装箱堆存的位置与理想的发箱顺序不同，倒箱操作是无可避免的。

对于出口箱的倒箱[4]，可以分为装船前的预倒箱整理和装船取箱时的倒箱，预倒箱是利用堆场机械的空闲时间在装船前对出口箱进行整理，最大程度地保障整理后的存放状态符合集装箱的计划发箱顺序，从而达到减少或避免装船时需要倒箱的可能，最大限度的提高装船效率，减少船舶停靠等待时间。并非所有集装箱码头都有足够的能力对所有的出口箱进行装船前预倒箱整理，装船时必须倒箱的情况是普遍存在的，另一方面，考虑到堆场作业中岸桥和场桥的大车移动需花费较长的时间，而且费用较高，必须尽可能的减少相互等待时间。现实工作中，确定岸桥和场桥的指派配比时一般仅考虑岸桥和场桥的平均作业能力，然而，在既

定的岸桥——场桥配置情况下，只有将各种因素对每次作业的影响均控制在平均水平，最大程度地减少整个装船过程中发箱或装船的速度波动性，才能保证每次作业的岸桥、场桥、集卡等机械配比和调度方案的一致性，提高各种机械的协调性，减少相互等待时间，保持装船过程的流畅性。为此，研究出口箱装船的堆场取箱问题时，将单次发箱对应的倒箱次数控制在平均水平，对减少各次发箱耗时的波动性至关重要，应作为控制整个装船过程总倒箱次数之外的另一个需解决的核心内容。

1.2 本文的主要研究内容

本文主要研究集装箱堆场优化问题，主要讨论了取箱操作时的倒箱问题，为堆场优化，提高码头工作效率做算法支持。主要内容可分为如下部分：

根据集装箱倒箱问题的实际情况，结合博弈模型的特点，提出了集箱倒箱博弈模型及启发式优化算法，通过实验表明该算法确实优化了倒箱次数。

根据出口箱装船的特殊情况，本文在预期倒箱数的概念基础上提出了一种自适应算法，该算法在保证机械运作的流畅性、调度的合理性基础上优化了总倒箱数。

进一步针对出口箱装船问题，在前人提出的C-M模型的基础上通过改进通道设置，把该算法应用到出口集装箱的倒箱优化问题上在保证机械运作的流畅性、调度的合理性基础上进一步优化了总倒箱数。

1.3 本文的结构安排

第一章介绍了选题背景与研究意义，简单阐述了集装箱运输的国内外现状，强调了本课题的重要性，并对本文的主要研究内容和结构安排做了说明。

第二章主要从码头系统的研究、船舶规划的研究、岸桥分配的研究、仿真系统的研究、集装箱堆码的研究几个方面介绍了相关课题的研究现状。

第三章提出了基于博弈理论的集装箱倒箱启发式算法，首先介绍了相关博弈理论基础，其次建立了数学模型，最后通过仿真实验论证了算法的优越性。

第四章根据出口箱取箱操作的特点，提出了考虑作业流畅性的基于预期倒箱数的出口箱取箱优化算法，本章首先建立了出口箱取箱作业模型，而后定义了取箱规则，进而实现了算法，最后通过实验验证了该算法的有效性。

第五章继续讨论出口箱的倒箱问题，介绍了 C-M 理论，利用 C-M 理论建立了相关优化算法，最后分析了实验结果。

第六章为本文的总结和展望，总结了论文的创新之处，对进一步的研究工作进行了展望。

2 相关研究

集装箱堆场规划是对集装箱在堆场内进行装、卸、运、存、等操作的安排，是为更能使堆场发挥最大的经济价值和有计划的进行集装箱装卸工作而制定的。绝大部分的集装箱作业都与堆场有关，因此，关于堆场优化的研究是相关学者的研究重点之一。国外很多学者已经致力于集装箱堆场管理的研究并提出了一些管理方案，相关学者研究了一些算法和费用模型以寻找不同条件下的堆放策略。

对集装箱堆场的相关研究有以下几方面：

2.1 集装箱码头系统的研究

一些研究人员对整个终端系统进行了研究。这些研究大多是描述性的决策或模拟。Choi et al.[5] 提出了 ERP 集装箱码头的整合系统，使得设施效率的提高靠资源的整合得以实现。Hartmann[6]提出了一个方案发生器，用于生成用于测试优化模型的正确数据。Bielli et al.[7]以提高港口的作业效率为目的给出了一个码头的仿真模型。Murty et al.[8]论述了一个决策支持系统的开发，以实现对码头日常事物的决策，并介绍了相关数学模型和算法。

2.2 船舶规划的研究

由于泊位是船只的装卸地点，泊位同一时间服务的船的数目以及大小决定于到来船只的大小、泊位的长度。泊位分配问题确定集装箱船靠泊时间和位置， Kim 和 Mon[9]利用模拟退火技术规划集装箱船的停靠位置。Imai et al.[10]以最大化泊位利用率为目标建立了混合整数规划模型，并实现了启发式算法。Imai et al.[11]通过指定服务的优先级来处理集装箱船的停靠问题，这里优先次序是由船只的服务时间所决定的。Imai et al.[12] 所讨论泊位规划问题和前人所讨论的离散泊位不同，提出了连续泊位分配问题，结果显示此方法高于以往算法所能实现的港口效率；装载规划是讨论集装箱在船上存放位置的问题，主要是按路线和船舶的稳定性加以讨论，目的是最大限度地利用船舶空间和尽量减少船舶上的集装箱的移箱。Wilson 和 Roach[13]使用禁忌搜索启发式算法处理该问题，在该文中禁忌搜索启发式算法被用到目标函数上以减少搜寻时间。Ambrosino et al.[14]讨论了主贝规划问题，建立了二进制线性规划模型。并提出了一种启发式算法以及一些规则。Imai et al.[15]对装载规划问题利用一个多目标整数规划模型加以解决，并用加权处理方法获得近似解。

2.3 岸桥分配的研究

Daganzo[16]建立了一个混合整数模型讨论了在没有船只到来时的岸桥静态分配问题。Peterkofsky 和 Daganzo[17]提出了利用分支定界法减少停泊船只的延误成本的方法。Park 和 Kim[18]对岸桥分配问题提出了一个两阶段解决方案，在第一阶段通过次梯度优化得到问题的近似解，这个结果被用于第二阶段岸桥具体的分配。Kim 和 Park[19]提出了一种混合整数规划模型并利用分支定界法加以解决，然后针对该混合整数规划模型提出了一种启发式算法——贪婪随机自适应搜索算法。

2.4 仿真系统的研究

Sgouris et al.[20] 利用电脑模拟来评估一个中型的进口集装箱码头装卸，目的是对短期规划和过程进行仿真使得决策得以优化。Yang et al.[21] 讨论了在一个自动化集装箱码头进行模拟评估的问题，并了建立相应模型。

2.5 集装箱堆码的研究

当集装箱到达码头，被存储到堆场等候船只或车辆来取，存储时间取决于集装箱是进口箱、出口箱还是中转箱，时间为几天到几周不等。在此期间，集装箱被存储在某个箱区的某个栈内。在存放时的可能操作可能包括：储存、拆箱、倒箱、熏蒸、搬迁等。Chung et al.[22]通过建立一个仿真系统模拟并分析了缓冲区面积对集装箱倒箱次数的影响。Taleb et al.[23] 在假定船只抵达的模式和工作负荷已知的情况下，利用缓冲空间和预倒箱来减少空间的浪费，较全面的讨论了出口集装箱的空间分配和存储问题。Kim[24] 对堆场集装箱预期倒箱数概念进行了分析估计。利用倒箱数估计吞吐率，通过假设取箱的随机性，在考虑贝层、列的基础上进行回归分析，计算和预测倒箱数目，通过实验显示了给出的近似公式的精确度相比以前的估计有明显提高。

Kim[25]讨论了出口集装箱的预倒箱问题，此操作是在 Kim[24] 定义的集装箱清场概念基础上讨论的，进一步讨论了清场问题。该文在假设堆场最初布局已知的条件下给出了预倒箱问题的解决方案，并把问题分为三个子问题加以解决——首先是贝匹配问题，是由动态规划解决的其次是集卡的移动规划问题，这是典型的运输问题，最后是一个旅行商问题。Kim et al.[26] 利用推导的方法在一个贝位中定位了出口集装箱的落箱位置，该文通过建立动态规划优化模型，以集装箱的重量级为基础，为到来的集装箱在贝内找到确切的落箱位置，以期尽量减少出口集装箱的预期倒箱数，为了实际的需要作者进一步利用决策树和分支定界法相结合的启发式算法对于规模较大的实例进行了分析。通过实例说明了启发式算法所得到的解是可以接受的。

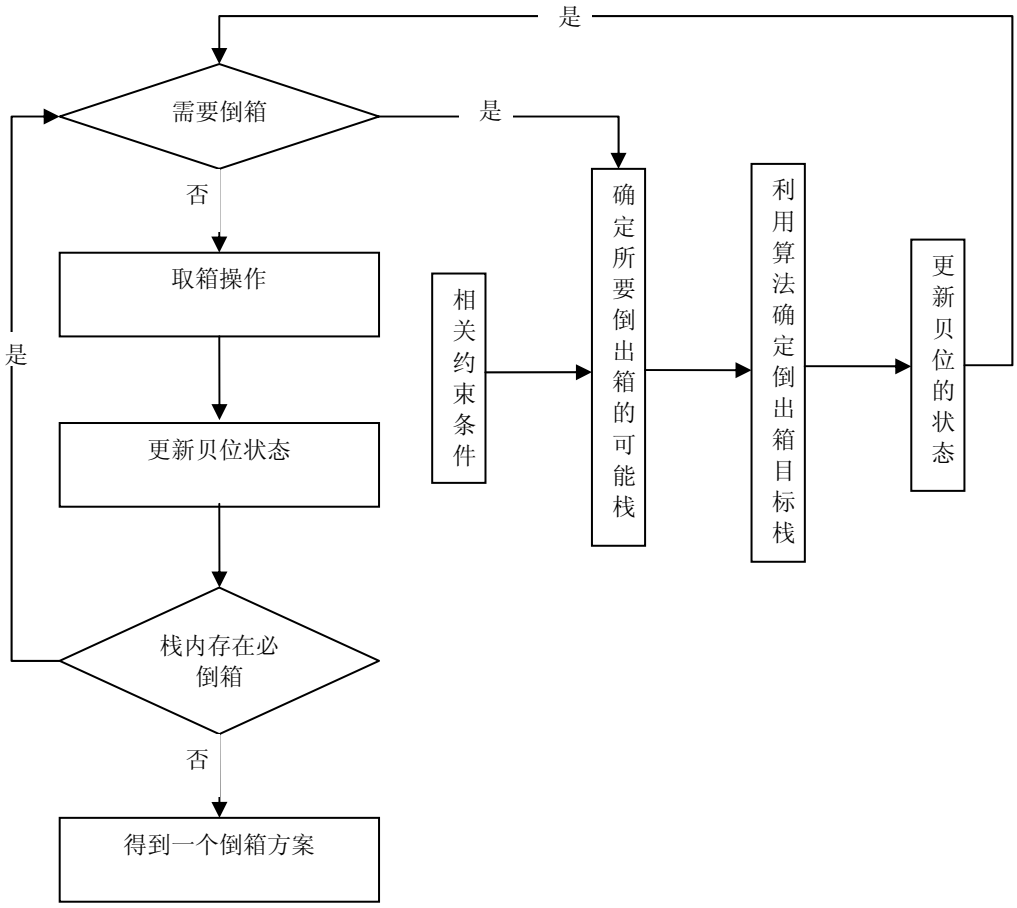

图 2.1 取箱操作模型框架

Fig.2.1 Model framework containers relocation

Kim et al.[27] 所提的问题是本文研究的出发点，在取箱过程中倒箱不可避免，如何选取被翻倒箱的落箱位直接影响着装船和装车效率，作者建立了倒箱优化模型，通过分支定界法给出了问题的最优解，为了保证计算的实时性作者给出了基于预期倒箱数的启发式算法。白治江[28]、徐亚等[29]对翻倒箱落箱位置的确定问题进行了进一步的研究，提出了一种启发式算法 H 及其改进算法 IH。

3 集装箱堆场倒箱博弈启发式优化算法

3.1 倒箱中的博弈

3.1.1 博弈论简介

博弈论[30]是研究在竞争条件下进行决策分析的科学，是应用数学的一个分支，也是运筹学的一个非常重要的学科。博弈论研究的典型问题是若干个利益冲突者在同一环境中进行决策以使自己的利益得到满足的策略问题，数学家们将具体的问题抽象化，通过建立自完备的理论框架和体系以研究其规律及变化。目前博弈论在经济学、国际关系学、军事战略、最优化理论和其他很多学科都有广泛的应用。在博弈论中，具有竞争或对抗性质的行为称为博弈行为。在博弈行为中，参加斗争或竞争的各方均具有自己的目标或利益。为了达到各自的目标和利益，各方必须考虑对手的各种可能的行动方案，力图选取对自己最为有利或最为合理的方案[30]。

博弈问题的要素[30]：

①局中人：所谓局中人是指在一场博弈中，为自己的利益进行决策的每一个有决策权的参与者。只有两个局中人的博弈现象称为“两人博弈”，而多于两个局中人的博弈称为“多人博弈”。

②可行方案集：可行方案集是局中人可以采取的行动方案的全体。在博弈问题中，局中人在其可行方案集上的选择就是决策分析。

③决策先后顺序：决策先后顺序是实际问题动态性质的反映，任何一种博弈的规则都要明确各个局中人进行决策分析的时间先后。

④收益函数（效用函数）：博弈最后结果中各个局中人利益的表示。在决策分析中，局中人把收益当作目标函数，力求使其最大化。

⑤信息：是指局中人在决策中对其条件的认知。

⑥策略博弈：称三元组

$$G =< N,(X_i),(P_i) >$$

为策略博弈，或正规博弈，其中 $N=\{1,2,\cdots,n\}$ 为局中人集，共 n 个局中人，$(X_i)=(X_1,X_2,\cdots,X_n),X_i$ 是局中人 i 的可行方案集，$i=1,2,\cdots,n$，$(P_i)=(P_1,P_2,\cdots,P_n),P_i$ 是局中人 i 的效用函数，比较具体地 $P_i(x_1,x_2,\cdots,x_n)$ 是在行为组合 $(x_1,x_2,\cdots,x_n)$ 之下局中人 i 的效用值，$x_j \in X_j, j=1,2,\cdots,n$。而局中人 i 在博弈中理性化的表现是使 P_i 极大化。又称行为组合 $(x_1,x_2,\cdots x_n)$ 为对策局势，且记局中人集 N 的人数为 $|N|$，即令 $|N|=n$。

3.1.2 倒箱的博弈模型

堆场集装箱取箱操作中，可以把客户和码头看成博弈过程中的局中人，把客户的取箱时间先后顺序作为客户局中人的策略，码头局中人的倒出箱的每一个落箱位置作为码头局中人的策略，建立码头局中人的效用函数，把倒箱优化问题转成效用函数的最大值问题。

在确定倒出集装箱的落箱位置时，应尽量避免较早提走的集装箱被晚提走的集装箱压在下面，即尽量使堆放在上层的集装箱先于堆放在其下的集装箱被提走。然而在很多情况下，集装箱到来的顺序是随机的，在收箱时尚不能确定其被提走的顺序，即便客户采用电子交换信息方式确定提箱顺序，往往受场内外各种因素的制约，实际提箱顺序与计划提箱顺序有很大的偏差，在取走其集装箱时使码头产生大量的翻箱作业，影响了堆场的作业效率和增加了作业成本。由于这些现实因素的存在，不可避免地会出现一定数量的倒箱。为此需要在提箱过程中对倒出的集装箱的落箱位置进行优化，尽可能减少二次倒箱，降低总倒箱率。这一过程实际上是一个动态的博弈过程，把集装箱客户和码头看成局中人，客户选择提走集装箱的时间作为局中人的策略，待客户局中人选定策略后通过电子交换信息传到堆场，码头局中人就获得了提箱顺序，把堆场的集装箱分为不同的优先级，越早提走的集装箱，其优先级就越高。寻求倒出箱的落箱最优位置转化成为计算博弈过程中的效用函数的最大效用值。效用函数的最大值应该体现出未来倒箱次数最少。

出于作业便利和安全等方面的考虑，倒箱一般只在同一个贝内进行。集装箱在堆场的堆存高度不能超过其最高堆放高度，在进行倒箱时，可选择的落箱位只能是那些还没有达到最高存放高度的栈（称为候选栈），这些栈就是堆场局中人的侯选策略栈，把它们分为两类，一类是侯选栈所堆放的集装箱的优先级均低于倒出集装箱的优先级；另一类是侯选栈所堆放的集装箱至少有一个优先级高于倒出集装箱的优先级。如图 3.1，为提取栈 1 中的集装箱 1，需把集装箱 3 翻倒到其它的栈，候选栈 3 和 4 为集装箱 3 的第一类候选栈，栈 2 为第二类候选栈。倒箱首先应该在第一类侯选策略栈寻找最优落箱位置，因为在这类侯选栈中倒出的箱不会第二次倒出，第二类侯选策略栈不可避免地要进行二次倒箱。由于客户局中人的策略已经给定，只需要设置码头局中人的效用函数 $u(s)$，效用函数应该与倒箱次数和计算时间有关，如果把集装箱倒在某个侯选栈引起未来的倒箱次数越少和计算时间越短，这个侯选栈策略对应的效用函数值就应该越大。每个策略代入效用函数中就对应一个效用值，效用值最大所对应的箱位就是倒箱的目标落箱位置。

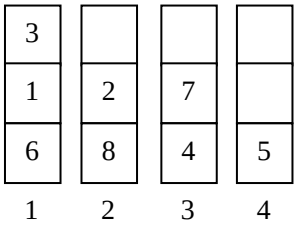

图 3.1 三层贝实例

Fig.3.1 The instance of three-storey bay

① 第一类候选策略栈的效用函数设置

为构造第一类候选栈的效用函数引入预期倒箱数[31] (ENAR）的概念，一个栈的预期倒箱数是指未来的倒箱中同一贝位其它栈可能的倒出箱落在该栈的空箱位使该栈增加的倒箱数。若栈 l 中集装箱的最高优先级为 n，空箱位为 e_l 则栈 l 的预期倒箱数可表示为 $E(e_l,n)$。在提箱过程中，下一个被倒在该栈的集装箱 i 可能是堆放在其它栈中的任何一个，若其优先级高于 n，则该箱不需要二次翻倒，该栈的预期倒箱数变为 $E(e_l-1,i)$；若其优先级低于 n，则该箱需要二次翻倒，该栈的预期倒箱数变为 $E(e_l-1,n)+1$。综上则有：

$$E(e_l,n)=\sum_{z=1}^{n-1}(p_{zl}\times E(e_l-1,z))+\sum_{z=n+1}^{N}(p_{zl}\times(E(e_l-1,n)+1)) \tag{3.1}$$

其中： p_{zl} 表示 z 为下一个要翻倒到栈 l 中的概率， N 表示初始贝中集装箱的数量。

若为提取优先级为 k 的集装箱需倒出优先级为 τ 的集装箱，对 τ 的第一类侯选栈的集合 $D_{1k\tau}(d)$（ d 表示当 前的贝位状态），如果存在 p 个第一类侯选栈，则堆场局中人相应地有 p 个纯策略 $\{s_{1k\tau1},s_{1k\tau2,}\cdots s_{1k\tau p}\}$。设栈 $l\in D_{1k\tau}(d)$，若记其剩余空箱位数为 $e_{lk\tau}$，最高优先级为 $n_{lk\tau}$， $E(e_{lk\tau},n_{lk\tau})$ 表示未来可能发生的倒箱而造成的预期倒箱数。则有：

$$E(e_{lk\tau},n_{lk\tau})=\sum_{i=1}^{n_{lk\tau}-1}(p_{il}\times E(e_{lk\tau}-1,i))+\sum_{i=n_{lk\tau}+1}^{N}(p_{il}\times(E(e_{lk\tau}-1,n_{lk\tau})+1)) \tag{3.2}$$

上式中 p_{il} 表示集装箱 i 要倒在栈 l 中的概率， N， h 分别表示初始贝中集装箱的数量和栈的额定堆存高度。有 $N-(h-e_{lk\tau})$ 个集装箱堆放在其它栈中，其中集装箱 $1\sim n_{lk\tau}-1$ 优先级高于 $n_{lk\tau}$。假设原来贝位中其它栈中集装箱以相同的概率成为下一个倒在 l 栈的集装箱，则 l 栈的预期倒箱数 $E(e_{lk\tau},n_{lk\tau})$ 可用下式估计[29][31]：

若 $e_{lk\tau}=1$，则：

$$E(e_{lk\tau},n_{lk\tau})=\frac{N-h-n_{lk\tau}+2}{N-h+1} \tag{3.3}$$

若 $e_{lk\tau}>1$，则：

$$E(e_{lk\tau},n_{lk\tau})=\frac{\sum_{i=1}^{n_{lk\tau}-1}E(e_{lk\tau}-1,i)}{N-h+e_{lk\tau}}+\frac{N-n_{lk\tau}-h+e_{lk\tau}+1}{N-h+e_{lk\tau}}\times(E(e_{lk\tau}-1,n_{lk\tau})+1) \quad (3.4)$$

如果把τ倒放到栈l，则栈l的预期倒箱数变为$E(e_{lk\tau}-1,\tau)$，其估计式为：

若$e_{lk\tau}=1$，则：

$$E(e_{lk\tau}-1,\tau)=0 \quad (3.5)$$

若$e_{lk\tau}>1$,则：

$$E(e_{lk\tau}-1,\tau)=\frac{\sum_{i=1}^{\tau-1}E(e_{lk\tau}-2,i)}{N-h+e_{lk\tau}-1}+\frac{N-\tau-h+e_{lk\tau}}{N-h+e_{lk\tau}-1}\times(E(e_{lk\tau}-2,\tau)+1) \quad (3.6)$$

用$\Delta E_{lk\tau}(\Delta E_{lk\tau}=E(e_{lk\psi}-1,\tau)-E(e_{lk\tau},n_{lk\tau}))$表示把$\tau$放到候选栈$l$所引起的预期倒箱数的增加量，该值越小表明未来倒箱的可能次数越少。因为把集装箱τ翻倒到栈l的操作发生状态变化的只有τ原来所在的栈以及栈l，故该值也表示对整个贝位引起的未来预期倒箱数的增加量，因此第一类候选栈的l策略栈的效用值可定义为：

$$u(l)=c-\Delta E_{lk\tau} \quad (3.7)$$

其中c是一个较大的常数。

若：

$$u(s_{1k\pi l})=\max_{i\le i\le p}(u(s_{1k\pi i})) \quad (3.8)$$

则栈$s_{1k\pi l}$为目标候选栈。

②第二类候选策略栈的效用函数设置

为提取优先级为k的集装箱需倒出优先级为τ的集装箱，若τ的落箱位置不存在第一类侯选栈，则只有在第二类侯选栈寻找最优落箱位置。设τ的第二类侯选栈的集合$D_{2k\tau}(d)$，第二类候选栈分成两种情形$D_{21k\tau}(d)$和$D_{22k\tau}(d)$。$D_{21k\tau}(d)$表示倒出集装箱的优先级比侯选栈所堆放的集装箱的最上层集装箱的优先级低的候选栈；$D_{22k\tau}(d)$的元素是满足倒出的集装箱的优先级比侯选栈所堆放的集装箱的最上层集装箱的优先级高的第二类候选栈。如图 3.2 为提取集装箱 1 需要倒出集装箱 7，这时候选栈为栈 2，栈 3 和栈 4，而这三个候选栈都属于第二类候选栈，这时考虑栈 2，栈 3 和栈 4 中的最上层的集装箱 6，4 和 9，有 6<7，4<7，9>7，所以，栈 2 和栈 3 均属于第二类候选栈中的第一类,而栈 4 为第二类候选栈中的第二类候选栈。

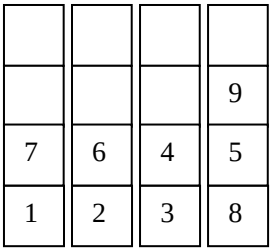

图 3.2 第二类候选栈实例

Fig.3.2 The instance of the second candidate stack

以图 3.3 中的第一个图为贝位初始布局说明第二类侯选栈的第一种情形效用函数的设置，为提取集装箱 1，需要把集装箱 7 倒出，此时的侯选栈只有栈 2 和 3（均为第二类侯选栈的第一种候选栈），其最上层集装箱的优先级分别为 6 和 4，如果集装箱 7 放在栈 2（6 的上层），集装箱 1 提走后，提取集装箱 2，需要倒出 7 和 6，由于有第一类侯选栈 1 的存在，把 7 和 6 倒出落在栈 1 不压箱；在集装箱 2 提走以后，提取集装箱 3，需要倒出集装箱 4，放在栈 2；整个提箱过程需要倒箱 4 次。如果集装箱 7 放在侯选栈 3（4 的上层），无论采用什么提箱规则，要把贝中的的集装箱提完至少需要 5 次倒箱（如图 3.4 所示）。因此倒出的集装箱应放到 $\min(7-6,7-4)=7-6$ 对应的栈 2 的集装箱 6 的上层，码头局中人的效用函数最大。一般地，设 $D_{21k\tau}(d)$ 有 m 个候选栈，记为：

$$D_{21k\tau}(d)=\{s_{21k\psi 1},s_{21k\tau 2,}\cdots s_{21k\varpi m}\} \tag{3.9}$$

$s_{21k\eth j}(1\le j\le m)$ 栈的最上层集装箱的优先级为 $a_{21k\eth j}$，倒出集装箱的优先级为 τ，侯选栈 $s_{21k\eth j}$ 的效用值定义为：

$$u(s_{21k\eth j})=c-(\tau-a_{21k\eth j}). \tag{3.10}$$

其中 c 是一个较大的常数。

若：

$$u(s_{21k\eth j})=\max_{1\le j\le m}(u(s_{21k\eth j})) \tag{3.11}$$

则栈 $s_{21k\eth j}$ 为目标候选栈。

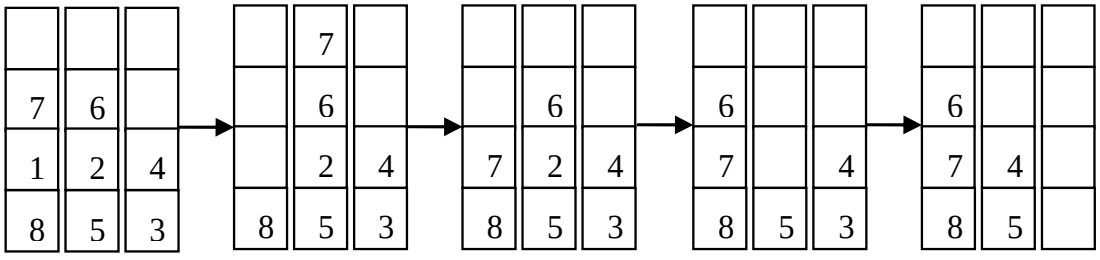

图 3.3 一般倒箱策略

Fig.3.3 Relocated strategy in general

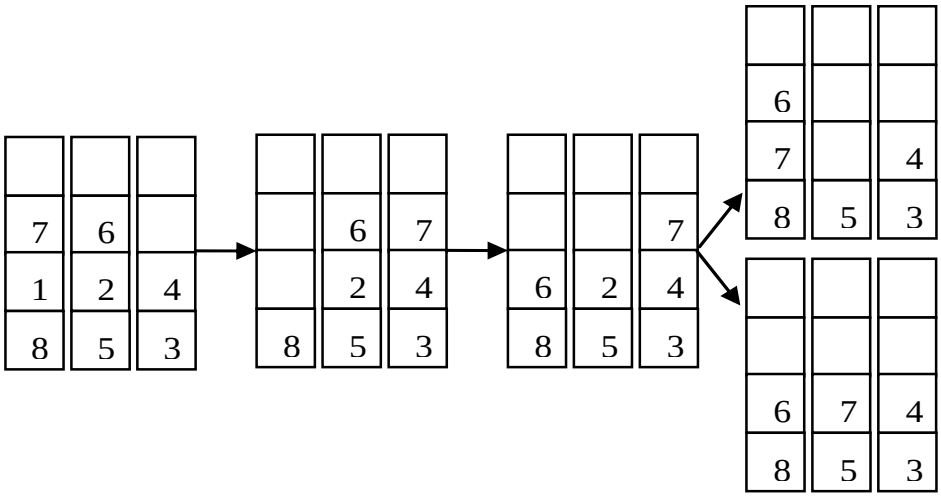

图 3.4 基于本文的倒箱策略

Fig.3.4 Relocated strategy in this paper

同理可得第二类侯选栈的第二种情形效用函数的表达式。假设 $D_{22k\tau}(d)$ 有 n 个候选，记为 $D_{22k\tau}(d)=\{s_{22k\tau 1},s_{22k\tau 2},\cdots s_{22k\tau m}\}$, $s_{22k\eta j}(1\le j\le n)$ 栈的最上层集装箱的优先级为 $a_{22k\eta j}$，倒出集装箱的优先级为 τ，侯选栈 $s_{22k\eta j}$ 的效用值定义为：

$$u(s_{22k\eta j})=c-(a_{22k\eta j}-\tau) \tag{3.12}$$

其中 c 是一个较大的常数。

若：

$$u(s_{22k\eta j})=\max_{1\le j\le m}(u(s_{22k\eta j})) \tag{3.13}$$

则栈 $s_{22k\eta j}$ 为目标候选栈。

在选择倒出的集装箱的落箱位置时，首先在第一类候选栈选取，若不存在第一类候选栈，就在第二类候选栈的第一种情形寻找，最后才在第二类侯选栈的第二种情形寻找。

3.2 算法步骤

t：提取目标箱 k 所需翻倒的集装箱所组成的列表；

τ：为当前状态 d 下当 $t\neq\Phi$ 时目标箱所在栈最上层的集装箱（当前需要翻倒的集装箱）；

$d'=T(d,x)$：在 d 状态下把 τ 放到栈 x 后产生新的状态 d'，其中 T 称为状态变换函数。

具体算法步骤如下：

步骤 1　为提取集装箱 $k=1$ 得到初始状态 d，转步骤 2。

步骤 2　判断 t 是否为空集，若 $t=\Phi$ 提走目标箱 k，转步骤 3，否则转步骤 4。

步骤 3　判断 k 的大小，若 $k < N$，令 $k = k + 1$，更新 d，转步骤 2，否则计算结束退出。

步骤 4　判断 $D_{lk\tau}(d)$ 是否为空集，若 $D_{lk\tau}(d) \neq \Phi$ 转步骤 5，否则转步骤 6。

步骤 5　把集装箱 τ 放入栈 $s_{1k\pi}$，$s_{1k\pi} \in D_{1k\pi}(d)$ 并且 $u(s_{1k\pi}) = \max_{s_{1k\pi} \in D_{1k\tau}(d)} (u(s_{1k\pi}))$，$d' = T(d, s_{1k\pi})$，$d = d'$ 转步骤 2。

步骤 6　判断 $D_{21k\tau}(d)$ 是否为空集，若 $D_{21k\tau}(d) \neq \Phi$，转步骤 7，否则转步骤 8。

步骤 7　把集装箱 τ 放入栈 $s_{21k\pi q}$，$s_{21k\pi q} \in D_{21k\tau}(d)$ 并且 $u(s_{21k\pi q}) = \max_{s_{21k\pi j} \in D_{21k\tau}} (u(s_{21k\pi j}))$，$d' = T(d, s_{21k\pi q})$，$d = d'$ 转步骤 2。

步骤 8　把集装箱 τ 放入栈 $s_{22k\tau y}$，$s_{22k\tau y} \in D_{22k\tau}(d)$ 并且 $u(s_{21k\tau y}) = \max_{s_{22k\pi h} \in D_{22k\tau}} (u(s_{22k\pi h}))$，$d' = T(d, s_{22k\tau y})$，$d = d'$ 转步骤 2。

表 3.1 两种算法求解实例所得到的平均倒箱数和平均计算时间

Table 3.1 The average number of containers relocation and average computing time of two algorithms

H	r	N	算法*IH* 平均倒箱数	算法HI平均计算时间/s	本章算法 平均倒箱数	本章算法平均计算时间/s
5	6	24	16.17	0.02	15.30	0.02
5	7	28	18.13	0.03	17.33	0.03
6	6	30	27.53	0.04	26.00	0.04
6	7	34	28.58	0.04	28.00	0.05
7	6	36	38.93	0.08	37.40	0.08
7	7	42	42.35	0.08	41.10	0.09

3.3 实验仿真

为了说明本章所提算法的有效性和实时性，仿真实验采用 C++编写程序并在 Visual C++6.0 内存平台上测得，实验结果均在 1.9GHz CPU，2G 内存上测得。 对六种实际应用中常见的贝位规模各自随机产生 40 个实例进行模拟优化，并与 *IH* 算法[29]在优化效果和计算时间上作比较。结果如表 3.1 所示。5×6、6×6、7×6（层数 *h*×栈数 *r*）三种规模的贝状态，每种 40 个实例的 *IH* 算法与本文算法贝位倒箱数（贝位倒箱数：贝中集装箱全部提走所发生的总的倒箱次数）的差值分布图分别如图 3.5，图 3.6 和图 3.7 所示。从表 3.1 和图 3.5、图 3.6 和图 3.7 可以看出对于四种不同规模的贝位，本文所提出的算法的求解效果优于 *IH* 算法。两种算法在相同规模下的平均计算时间相差很小，不大于 0.02s，满足应用中的实时性(平均不大

于 0.1s)。

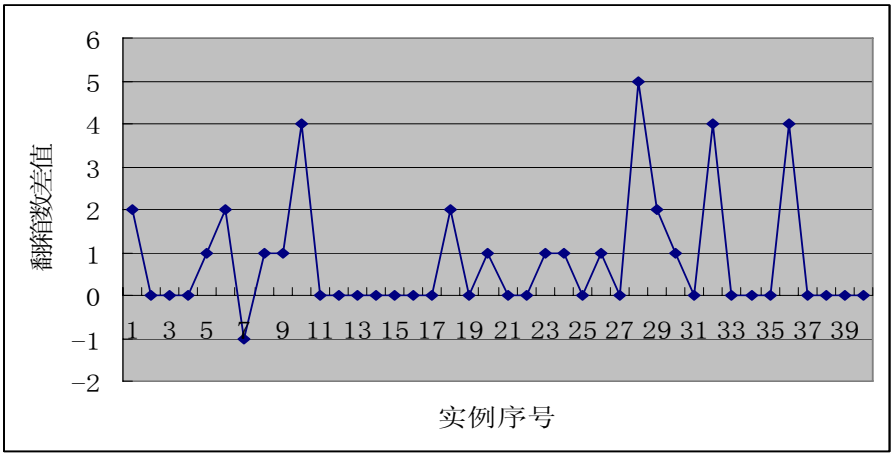

图 3.5 在 5×6 规模下 40 个例子中算法 IH 与本文算法贝位倒箱数的差值分布

Fig.3.5 In 40 instances of 5×6, the different distribution between IH algorithm and the paper algorithm

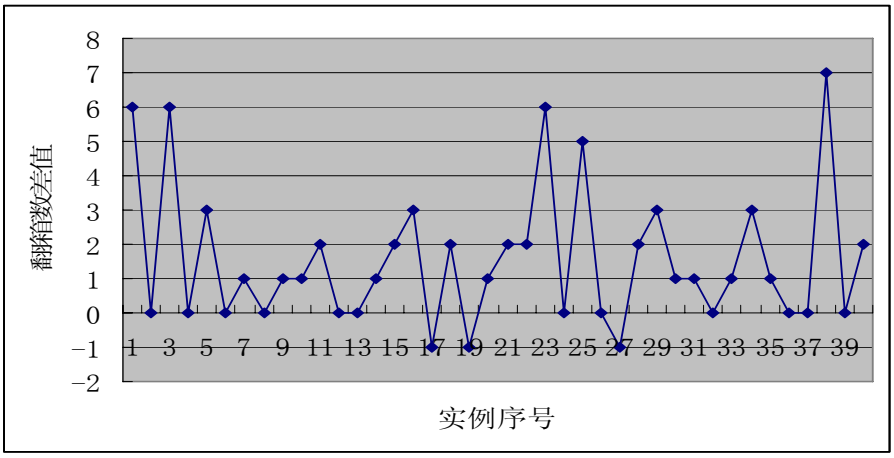

图 3.6 在 6×6 规模下 40 个例子中算法 IH 与本文算法贝位倒箱数的差值分布

Fig.3.6 In 40 instances of 6×6, the different distribution between IH algorithm and the paper algorithm

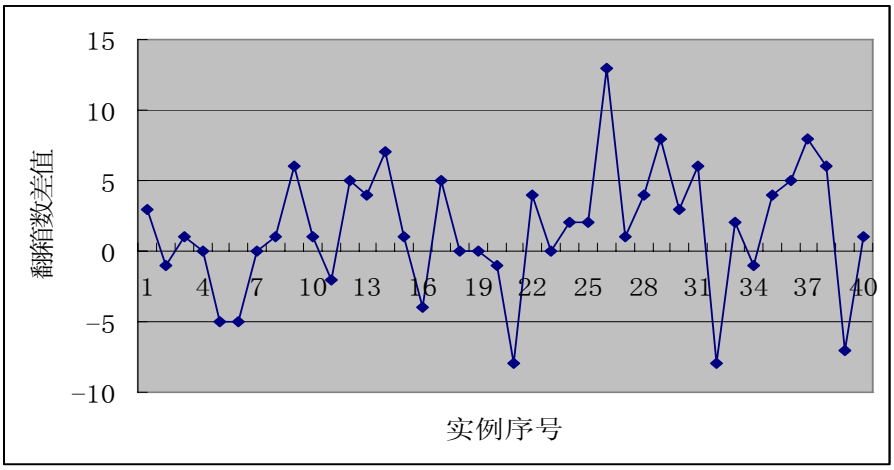

图 3.7 在 7×6 规模下 40 个例子中算法 IH 与本文算法贝位倒箱数的差值

Fig.3.7 In 40 instances of 7×6, the different distribution between IH algorithm and the paper algorithm

为进一步说明本文算法的有效性，对实际集装箱堆放中有可能出现的5×5，5×6，5×7，6×6，6×7，6×8，6×9，6×10规模的贝位随机产生 50 个例子进行计算，给出本文算法相对算法 *H* 和算法 *IH* 的平均改善程度。记 *H* 算法平均倒箱数为 a，本文算法平均倒箱数为 b，则本文算法相对于 *H* 算法的平均改善程度 *AIH* 为 $(a-b)/a$；同理可得出本文算法相对于 *IH* 算法的平均改善程度 *AIIH*。计算结果如表 2 所示。从表 2 中可以看出在各种规模的实验中本文算法得到的效果均有一定的提高。

表 3.2 不同规模下本文算法和算法 *H*、算法 *IH* 比较的结果

Table 3.2 The comparison between the paper algorithm, H algorithm and IH algorithm under different scales

h	*r*	*N*	算法 *H*	算法 *IH*	本文算法	*AIH*	*AIIH*
5	5	20	14.83	14.20	14.01	5.5%	1.3%
5	6	24	15.98	16.17	15.30	4.3%	5.4%
5	7	28	18.65	18.13	17.33	7.1%	4.4%
6	6	30	26.93	27.53	26.00	3.5%	5.6%
6	7	35	29.48	28.58	28.10	4.7%	1.4%
6	8	40	32.95	32.08	31.53	4.3%	1.7%
6	9	45	37.10	35.23	33.60	9.4%	4.6%
6	10	50	39.00	38.50	36.95	5.3%	4.0%

3.4 小结

本章为集装箱堆场建立了一个博弈模型，通过合理设置码头局中人的效用函数，把集装箱倒箱优化转化成寻求效用函数的最大值，有效地减少集装箱取箱过程中的倒箱数，提高堆场的作业效率和节约大量的成本。实验结果表明本文提出的优化算法的结果较徐亚[29]的算法 *H* 和算法 *IH* 的优化结果有明显的改善，为码头集装箱的智能化管理提供了坚实的理论基础。

4 基于预期倒箱数的出口箱取箱优化算法

4.1 前言

对于出口箱的取箱操作（这里出口箱的取箱操作也包括中转箱的装船操作）由于要考虑机械设备调度的协调性，减少大型设备相互等待时间，必须考虑单次取箱所倒箱次数的均衡性。张艳伟等[33]对于考虑单次提箱倒箱约束的提箱优化问题作了详细的论述，并建立了相关模型，然而至今还没有一种通用的算法，以保证在较短的时间内得到问题的最优解或较优解，以快速得到在任意堆存贝位装船过程最优或较优的倒箱方案，以达到控制场桥按优化结果执行发箱的目的。本章将在深入研究该问题的基础上提出一个基于预期倒箱数和最小差值思想相结合的启发式优化算法。

4.2 出口箱取箱作业的理论基础

模型的理论基础依赖于堆存箱位为多层多排的直接堆垛方式、位内只允许堆存相同尺寸的集装箱、装船前堆存位初始状态和各发箱顺序为已知、位内有足够的空间供倒箱以及提箱过程中不允许新的集装箱堆放到该贝的 5 个假设。在倒箱时为控制总倒箱次数和单次发箱对应的倒箱次数，需要引入目标箱、必倒箱，阻塞箱和取箱代价几个概念[33]。目标箱是在某一堆存状态下最先需要发的集装箱；必倒箱是目标箱上层的集装箱；阻塞箱是连续堆存在某集装箱上方的必倒箱；取箱代价是为取出某个集装箱所需的最少倒箱次数[33]。

在初始堆存状态确定的情况下，集装箱单次发箱对应的倒箱次数直接受倒出箱的落箱位选取的影响，因此需要合理的落箱位的选取，即可实现对整个装船过程单次倒箱次数的控制。

设 S_k 表示 k 号集装箱发箱后堆存贝位的最新状态，S_{k-1}^{i} 表示 $k-1$ 号箱已取箱，在取 k 号目标箱时上方有 i 个阻塞箱已倒后得到的堆存最新状态，NF_k 表示在状态 S_{k-1} 下，k 号箱上方阻塞箱的总个数，也即取走集装箱 k 所用的最小代价，AF_k 表示 k 号目标箱上方各阻塞箱的倒箱作业构成的倒箱序列，N 表示初始堆存状态下，贝位内集装箱的总个数。则装船位内倒箱的优化方案定义为[33]：

$$Solution_{optimal} = \arg \min_{AF_1,\cdots AF_L} \sum_{j=1}^{L} NF_j, L \le N \tag{4.1}$$

其中，$\sum_{j=1}^{L} NF_j$ 表示为提取 L 个集装箱实际取箱代价的总和；目标箱 L 发出以后不再有必倒箱，此时就得到一个倒箱方案。若目标箱上方有阻塞箱，一次发箱涉及

目标箱上方各阻塞箱的倒箱操作和目标箱取箱操作两部分，否则，目标箱可直接取箱。一系列的倒箱和取箱操作致使堆存状态由一个状态转化为另一个状态，直到位内不存在必倒箱就形成一个倒箱方案。其中，倒箱操作依次将相关阻塞箱直接放入按照一定规则选定的箱位，而取箱操作只需将目标箱从所在的箱位内直接移出堆存位。由于目标箱所在的位置由初始堆存状态或前序倒箱决定，为此，在既定初始堆存状态下，倒箱目标箱位的选取直接决定了堆存状态的改变。由于候选栈往往不止一个，最终可形成不同的倒箱方案组合，其中倒箱次数最小的为最优方案。若出现初始堆存状态不理想，无法找到可行倒箱方案的情况，可通过放松对候选栈的选取约束，找到实际生产中可以接受的倒箱方案。

4.3 堆存箱位的选取规则

由于倒箱落箱位的选取会影响后继取箱操作的倒箱次数和提箱作业的流畅性，每次倒箱的候选栈的选取是十分关键的。如图 4.1 所示为提取集装箱 1 需要倒箱 7，如果把箱 7 放到栈 2 的空箱位中即放到箱 5 的上面，则下一步提取集装箱 2 时就要翻倒三次集装箱，这样不利于提箱作业的流畅性。如图 4.2 为提取栈 1 中的集装箱 1，若把集装箱 3 翻倒到栈 3，栈 4 不会对现有造成阻塞，翻到栈 2 不可避免地引起二次倒箱。因此堆存箱位选取原则应该兼顾提箱作业的流畅性和翻箱次数尽可能的少。

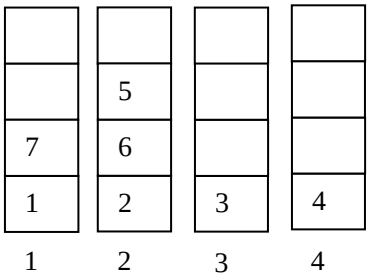

图 4.1 实例 1

Fig.4.1 Example 1

一般地，在既定的初始堆存状态，集装箱单次发箱对应的翻箱次数直接受翻出箱堆存箱位的选取的影响，只有通过合理地堆存箱位的选取原则，对各阶段进行合理的倒箱决策，才能实现对整个装船过程单次取箱对应倒箱次数的控制。如果目标箱上方无阻塞箱，不涉及倒箱决策。当目标箱上方有阻塞箱时，目标箱的取箱代价不会小于阻塞箱的个数，且阻塞箱的个数最少为 1，此时需要进行倒箱操作。选取其它栈没有达到最高堆放高度的栈作为倒出阻塞箱的堆存初始侯选栈；若初始侯选栈中现有各箱均比预移入的阻塞箱发箱晚，则该栈优先作为需翻倒集

装箱的候选栈，阻塞箱移入后不会对栈内现有各箱造成阻塞，现有各箱的最小取箱代价保持不变。本章把这类栈称为针对出口箱的第一类候选栈；若这类候选栈不存在，则放松约束条件寻找栈中最早发箱的集装箱上方阻塞箱最少的栈作为候选栈，本文把这类栈称为针对出口箱的第二类候选栈。在文中为了较好的讨论单次提箱的流畅性以及总倒箱数之间的均衡性，第二类候选栈的选取取阻塞箱较少的栈加以讨论。

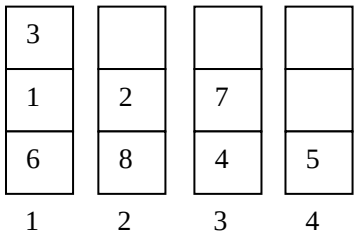

图 4.2 实例 2

Fig.4.2 Example 2

4.3.1 针对出口箱的第一类侯选栈中落箱栈的选取

在针对出口箱的第一类侯选栈中寻找一个理想的落箱栈需要和前一章相同的引入预期倒箱数[27]的概念，具体符号说明及讨论可参照第三章的相关内容。

若为提取优先级为 k 的集装箱需倒出优先级为 τ 的集装箱，对 τ 的针对出口箱的第一类侯选栈的集合 $D'_{1k\tau}(d)$，设栈 $l \in D'_{1k\tau}(d)$，若记其剩余空箱位数为 $e_{lk\tau}$，最高优先级为 $n_{lk\tau}$，用 $E(e_{lk\tau}, n_{lk\tau})$ 表示未来可能发生的倒箱而造成的预期倒箱数。则有：

$$E(e_{lk\tau}, n_{lk\tau}) = \sum_{i=1}^{n_{lk\tau}-1}(p_{il} \times E(e_{lk\tau}-1, i)) + \sum_{i=n_{lk\tau}+1}^{N}(p_{il} \times (1 + E(e_{lk\tau}-1, n_{lk\tau}))). \quad (4.2)$$

用 ΔE_{lkx} $(\Delta E_{lkx} = E(e_{lkx}-1, \tau) - E(e_{lkx}, n_{lkx}))$表示把 τ 放到候选栈 l 所引起的预期倒箱数的增加量, 该值越小表明未来倒箱的可能次数越少。因为把集装箱 τ 翻倒到栈 l 的操作发生状态变化的只有 τ 原来所在的栈以及栈 l，故该值也表示对整个贝位引起的未来预期倒箱数的增加量, 因此 若

$$i = \arg \min_{l \in D'_{1k\tau}(d)} \Delta E_{lk\tau}. \quad (4.3)$$

则栈 i 为目标落箱栈。

4.3.2 针对出口箱的第二类侯选栈的落箱栈的选取

对于只有针对出口箱的第二类候选栈的情况，本算法基于减少两次或更多次可能的倒箱为目的，比较所翻倒的集装箱的提箱优先级和每个针对出口箱的第二类候选栈中最上层的集装箱的提箱优先级的关系，把针对出口箱的第二类候选栈再一次分为两种情形，优先考虑最上层的集装箱的提箱优先级高于所翻倒的集装

箱的优先级的一类，把集装箱翻到这一类候选栈中的最上层的集装箱的提箱优先级最低的栈中；如果没有前一类，就考虑把集装箱翻倒到最上层的集装箱的提箱优先级最高的针对出口箱的第二类候选栈中。

为提取优先级为 k 的集装箱需倒出优先级为 τ 的集装箱，若 τ 的落箱位置不存在针对出口箱的第一类侯选栈，则只有在针对出口箱的第二类侯选栈寻找最优落箱位置。设 τ 的针对出口箱的第二类侯选栈的集合为 $D'_{2kx}(d)$，针对出口箱的第二类候选栈分成两种情形 $D'_{21kx}(d)$ 和 $D'_{22kx}(d)$。 $D'_{21kx}(d)$ 表示倒出集装箱的优先级比侯选栈所堆放的集装箱的最上层集装箱的优先级低的针对出口箱的第二类候选栈；$D'_{22kx}(d)$ 表示倒出的集装箱的优先级比侯选栈所堆放的集装箱的最上层集装箱的优先级高的针对出口箱的第二类候选栈。设 a_{21kx} 为 $D'_{21kx}(d)$ 中候选栈 j 的最上层集装箱的取箱优先级，设 $a_{22k\tau w}$ 为 $D'_{22kx}(d)$ 中候选栈 w 的最上层集装箱的取箱优先级，则：

若

$$v = \arg \min_{j \in D'_{21k\tau}(d)} (\tau - a_{21k\tau j}) \tag{4.4}$$

则栈 v 为 τ 在 $D'_{21kx}(d)$ 中的目标落箱栈栈.。

若

$$y = \arg \min_{w \in D'_{22k\tau}(d)} (a_{22k\tau w} - \tau) \tag{4.5}$$

则栈 y 为 τ 在 $D_{22kx}(d)$ 中的目标落箱栈。

如图 4.3，为提取集装箱 1 需要翻倒集装箱 14，容易判断针对出口箱的第一类候选栈不存在， 由于栈 2、栈 3、栈 5、栈 6 中栈 2、栈 3 的最早发箱的集装箱上方阻塞箱最少故第二类候选栈为栈 2 和栈 3，这时考虑栈 2 和栈 3 中的最上层的集装箱 13 和 2， 13 小于 14 并且 2 小于 14，即栈 2 和栈 3 均属于针对出口箱的第二类候选栈中的第一种情形栈，根据式(4.5)可知应把 14 放入栈 2。

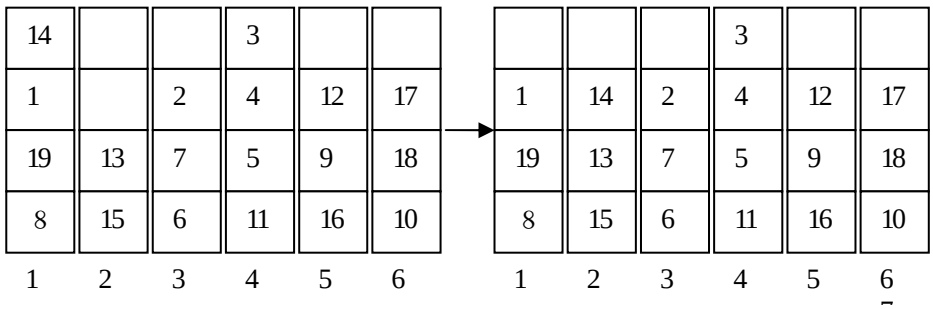

图 4.3 策略分析实例

Fig 4.3 The instance of strategy analysis

在图 4.4 中，为提取集装箱 1 需要翻倒集装箱 14，容易判断不存在针对出口箱的第一类候选栈，栈 2、栈 3、栈 5、栈 6 中栈 2、栈 5 最早发箱的集装箱上方阻塞箱最少所以得到第二类候选栈为栈 2 和栈 5，栈 2 属于针对出口箱的第二类候选栈中的第二种情形栈，根据式(4.6)可知应把 14 放入栈 2。

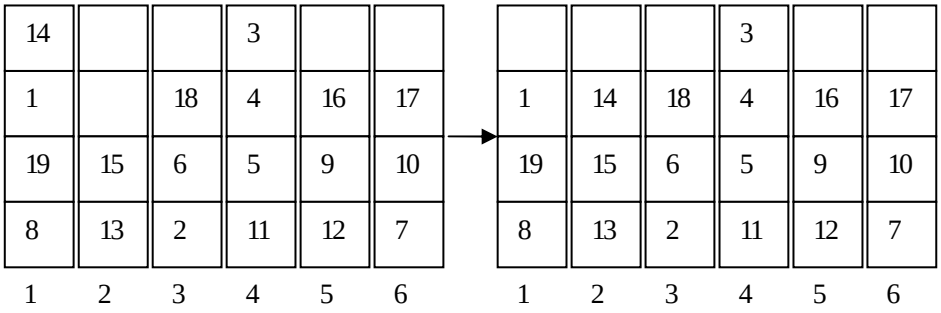

图 4.4 策略分析实例

Fig.4.4 The instance of strategy analysis

4.4 算法描述

4.4.1 符号说明

$a_{lk\tau}$：为提取集装箱 k 需翻倒集装箱 τ 时栈 l 的最顶层集装箱的优先级，若栈 l 为空栈，则取 $a_{lk\tau}=N+1$；

$n_{lk\tau}$：为提取集装箱 k 需翻倒集装箱 τ 时栈 l 中所堆放集装箱的最高优先级；

$e_{lk\tau}$：在为提取集装箱 k 翻倒集装箱 τ 时栈 l 的空箱位数；

t：提取目标箱 k 所需要翻倒的集装箱所组成的列表，即提取 k 时的必倒箱的集合；

$D'_{1k\tau}(s)$：为提取集装箱 k，翻倒集装箱 τ 的针对出口箱的第一类候选栈的集合，$D'_{1k\tau}(s)=\{l \mid l\in D'(s), n_{lk}>\tau\}$；

$D'_{m2k\tau}(s)$：为提取集装箱 k，翻倒集装箱 τ 的针对出口箱的第二类候选栈的集合，即：除了第一类候选栈以外的没有达到最高高度限制的并且栈中最高优先级的箱子的阻塞箱少于 m 的栈的集合；

$D'_{m21k\tau}(s)$：为翻倒集装箱 τ 的针对出口箱的第二类候选栈中满足 $a_{lk\tau}<\tau$ 的栈的集合，$D'_{21k\tau}(s)=\{l \mid l\in D'_{m2k\tau}(s), a_{lk\tau}<\tau\}$；

$D'_{m22k\tau}(s)$：为翻倒集装箱 τ 的针对出口箱的第二类候选栈中满足 $a_{lk\tau}>\tau$ 的栈的集合，$D'_{m22k\tau}(s)=D'_{m2k\tau}(s)-D'_{m21k\tau}(s)$。

4.4.2 算法步骤

步骤 1　为提取集装箱 $k=1$ 得到初始状态变量 s，转步骤 2。

步骤 2　判断 t 是否为空集，若 $t=\Phi$ 提走目标箱 k，转步骤 3，否则转步骤 4。

步骤 3　判断 k 的大小，若 $k < N$，令 $k = k + 1$，更新 s 转步骤 2，否则计算结束，退出。

步骤 4　判断 $D'_{1k\tau}(s)$ 是否为空集，若 $D'_{1k\tau}(s) \neq \Phi$ 转步骤 5，否则转步骤 6。

步骤 5　把集装箱 τ 放入栈 x，$x \in D_{1k\tau}(s)$ 且 $\Delta E_{xk\tau} = \min_{l \in D'_{1k\tau}} \Delta E_{lk\tau}$ $s' = T(s, x)$，$s = s'$ 转步骤 2。

步骤 6　令 $m = 1$（或者大于 1 的一个值，实验中将具体分析），判断 $D'_{m2k\tau}$ 是否为空集，若 $D'_{m2k\tau} \neq \Phi$ 转步骤 7，否则 $m = m + 1$ 转步骤 6。

步骤 7　判断 $D'_{m21k\tau}(s)$ 是否为空集，若 $D'_{m21k\tau}(s) \neq \Phi$，转步骤 8，否则转步骤 9。

步骤 8　把集装箱 τ 放入栈 v，$v \in D'_{m21k\tau}(s)$ 且 $(\tau - a_{vk\tau}) = \min_{l \in D'_{m21k\tau}} (\tau - a_{lk\tau})$，$s' = T(s, x)$，$s = s'$ 转步骤 2。

步骤 9　把集装箱 τ 放入栈 w，$w \in D'_{m22k\tau}(s)$ 且 $(a_{wk\tau} - \tau) = \min_{l \in D'_{m22k\tau}} (a_{lk\tau} - \tau)$，$s' = T(s, x)$，$s = s'$ 转步骤 2。

4.5 实验分析

在本实验中对堆放中经常出现的 4×6，4×7，5×6，5×7，6×6，6×7 规模贝位分别随机产生 40 个例子进行实验。为了说明本文算法的优越性，把本文算法步骤 6 中的 m 分别取初始值 1 和 2，并和算法 IH 在优化效果上作比较，结果如表 1 所示。从表 1 可以看出对于这几种规模的贝位，本文所提出的算法对于贝位平均翻箱数的平均求解效果均优于算法 IH，并且 $m = 1$ 时本文算法的优化效果最好。

表 4.1 算法比较

Tab.4.1 The comparison of the algorithms

贝位层	贝位栈	最高高度	算法 IH 平均倒箱数	本文算法 L=1 时平均倒箱数	本文算法 L=2 时平均倒箱数
4	6	5	15.55	15.000	15.475
4	7	5	18.11	18.000	18.012
5	6	6	26.87	26.400	26.550
5	7	6	28.58	28.036	28.075
6	6	7	38.93	38.075	38.000
6	7	7	42.38	42.000	41.880

为了进一步分析算法在对单次提箱耗时波动性控制方面的优越性，在表 2、表 3、表 4 中分别对 4×6、5×6、6×6 的 40 个实验作统计分析，列出了对于不同取

箱代价的集装箱的百分比，并列出了不同算法的不同取箱代价的集装箱百分比的样本方差，用来度量单次提箱耗时波动性。从表中可以看出在各种规模的实验本文算法得到的效果在控制波动性方面均有一定的提高。

表 4.2 4×6 型贝位实例比较

Tab.4.2 The comparison of 4×6 model

	倒箱数为 0	倒箱数为 1	倒箱数为 2	倒箱数为 3	倒箱数为 4	方差
IH	64.2%	17.3%	9.3%	8.1%	1.2%	0.0642
m=1	59.3%	23.6%	10.4%	6.6%	0.1%	0.0556
m=2	62.5%	17.3%	13.5%	6.6%	0.1%	0.0608

表 4.3 5×6 型贝位实例比较

Tab.4.3 The comparison of 5×6 model

	倒箱数为 0	倒箱数为 1	倒箱数为 2	倒箱数为 3	倒箱数为 4	倒箱数为 5	方差
IH	58.8%	16.4%	10.7%	6.6%	5.4%	2.1%	0.0451
m=1	55.6%	21.0%	11.6%	6.4%	5.2%	0.3%	0.0413
m=2	58.1%	15.0%	15.2%	6.4%	5.1%	0.3%	0.0446

表 4.4 6×6 型贝位实例比较

Tab.4.4 The comparison of 6×6 model

	倒箱数为 0	倒箱数为 1	倒箱数为 2	倒箱数为 3	倒箱数为 4	倒箱数为 5	倒箱数为 6	方差
IH	58.0%	13.7%	9.1%	7.8%	5.7%	4.2%	1.6%	0.0386
m=1	50.9%	21.2%	11.5%	7.8%	4.8%	3.5%	0.2%	0.0307
m=2	54.4%	14.4%	14.5%	7.9%	5.0%	3.7%	0.2%	0.0341

4.6 小结

出口集装箱装船时取箱优化属于多阶段决策过程的组合最优化，在既定的初始堆存状态下，以最小化倒箱总次数和确保装船系统的流畅性为优化目标。本章对装船时多层堆场箱区堆存位内的倒箱问题进行了描述，设计的优化搜索算法可

以保证在较短的时间内得到问题的最优解或较优解。通过一种基于预期倒箱数和最小区别思想相结合的启发式优化算法合理选取候选栈既保证了单次提箱的流畅性又较好的限制了贝位提箱总倒箱率。

5 改进的 C-M 模型在出口箱取箱优化中的应用

5.1 C-M 模型简介

2006 年 Moshe Sniedovich 和 Stefan Voß[34]提出了一种新的用于求解动态规划的启发式算法 C-M(corridor method)模型：设问题：

$$P(X)：\ z^* = \underset{x\in X}{opt}\ f(X) \tag{5.1}$$

C-M 思想是：当问题 $P(X)$ 的搜索空间 X 太大（即出现所谓的维数灾难）以至于利用成熟的优化算法 M 不能实时的解决时，根据优化方法 M 的搜索特点构造一种迭代局部搜索方法使得问题 $P(X)$ 在新的搜索空间内能够实时的应用方法 M 得到最优解或近似解。

C-M 框架如下：

假设：问题 $P(X)$，方法 M .

条件：N_M（基于 M 的邻域函数）、基于 M 的终止条件、Π_M（Π_M 是 X 到本身的一个映射，$x' = \Pi_M(x)$ 是 x 的扰动）.

开始：令 $j=1$、选择 $x^{(1)} \in X$

迭代：重复以下过程直到满足基于 M 的终止条件

$$x^* = \arg\ \underset{y\in N(x^{(j)})}{opt_M}\ f(y)$$

如果 $f(x^*) = f(x^{(j)})$，令 $x^{(j+1)} = \Pi_{\mathrm{M}}(x^{(j)})$

否则，令 $x^{(j+1)} = x^*$.

令 $j = j+1$.

集装箱翻倒箱操作本质上是动态规划问题，对于这类问题可以应用现已存在的传统的启发式算法，比如禁忌搜索、模拟退火、遗传算法和迭代局部搜索方法求解，但是由于随着问题规模的增大，利用这些方法求解该问题的计算时间以及内存占用空间都会成指数形式增加，不利于问题求解的实时要求。针对集装箱翻倒箱问题 Marco Caserta, Stefan Voß，Moshe Sniedovich[35]通过建立翻倒集装箱的数学模型，

$$f(s) = \min_{x\in D(s)}\{1 + f(T(s,x))\}, k = n-1,\cdots,1 \tag{5.2}$$

其中：$f(n,i,\Phi,C)=1$，$s=(k,i,t,C)$：状态变量，这里为 $k\in\{1,\cdots,n\}$ 需要提走的集装箱，$i\in\{1,\cdots,m\}$ 为当前目标箱所在的栈，t 为 k 的上面所有集装箱的集合，即当前需要翻倒的集装箱的集合，C 为栈 i 以外的栈的布局；$D(s)$：当前需翻倒的集装箱的候选栈组成的集合；$s'=(k',i',t',C')$：利用决策 $x\in D(s)$ 所得到的新的状态变量，表示为 $s'=T(s,x)$。

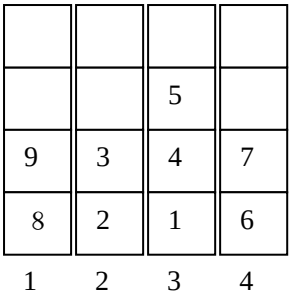

图 5.1 相关概念说明实例

Fig.5.1 The instance of related concepts

以图 5.1 为例说明上述符号：假设需提走所有的集装箱，则$n=9$，当前需提走的集装箱为箱 1，即$k=1$这时$i=2,t=\{5,4\},C=\{\{9,8\},\{3,2\}\{7,6\}\},D(s)=\{1,3,4\}$。如果把集装箱 5 倒到栈 1，则$x=1$，$s'=T(s,x)$，$s'=(1,2,\{4\},\{\{5,9,8\},\{3,2\},\{76\}\})$。

M. Caserta et al[35]利用上述递归公式对翻倒箱问题进行求解，然而由于随着贝位规模的增大，计算时间不能控制在一个可接受的范围内。作者进一步利用 C-M 的思想，通过对当前目标箱构造 “2 维”通道（two-dimensional corridor）约束：

$$D(s,\delta,\lambda)=\{x\in\{1,\cdots,m\}\setminus\{i\}\mid i-\delta\le x\le i+\delta,|c_x|<\lambda\}. \tag{5.3}$$

δ 为对栈个数的约束；λ 为对栈层数的约束。

这样从 s 可以产生的领域为：

$$N(s)=\{s'\mid s'=T(s,x),\forall x\in D(s,\delta,\lambda)\} \tag{5.4}$$

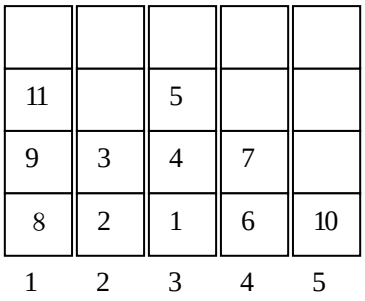

图 5.2 相关概念说明实例

Fig.5.2 The instance of related concepts

如图 5.2，$k=1$， $i=3,t=\{5,4\},C=\{\{11,9,8\},\{3,2\}\{7,6\}\{10\}\}$, 若取$\delta=1$ $\lambda=3$ 则$D(s,\delta,\lambda)=\{2,4\}$。这样通过约束领域的大小。减少了算法的计算时间，实验表明该方法对翻倒箱的数量控制达到了较好的结果。

5.2 针对出口箱取箱操作的 C-M 改进模型

M. Caserta et al.[35]在利用 C-M 解决倒箱问题时并没有考虑堆场对出口集装箱装船时的时间波动性的特殊要求，没有考虑单次装船的时间约束。在这里基于最大程度的降低翻倒箱数，以及减小出口箱取箱时的时间波动性两方面的考虑，把上述递归模型作以下改动如下：

设集装箱 k 被取走时为提取箱 $k+1$ 所必须翻倒的集装箱个数为 $|t_{k+1}|$，记 $T=(\mathrm{t}_1,\cdots \mathrm{t}_n)$，$D(T)$ 表示对 T 求方差，c 为一个正数。

$$f(s)=\min_{x\in D(s)}\{1+f(T(s,x))\},k=n-1,\cdots,1 \tag{5.5}$$

并且：$D(T)\le c$.

通过添加约束式 $D(T)\le c$ 使得每次提箱的翻倒箱数在一定程度上趋于平均值。这样对于出口箱的提箱过程既保证了总的作业量控制在较少的水平，又控制了作业流程的顺畅性。

由于[45]并没有考虑作业流程顺畅性这一条件，故利用其所给出的 C-M 启发式求解方法并不能较好求解上述模型。现改进 C-M 启发式求解方法，这里把“‘2 维’通道”（“two-dimensional”corridor）约束作如下改动：

定义 $J=\{j\mid j\in\{1,\cdots,m\}\setminus i\}$，即除了当前目标箱所在栈之外的栈的集合，$a_j(s)$ 表示当前状态下栈 j 中优先级最高的集装箱的优先级，$b(a_j)$ 表示 $a_j(s)$ 上面集装箱的数目，

$$B(s,g)=\{j\in J\mid (b(a_j)<g)\vee(a_j(s)>\tau)\} \tag{5.6}$$

g 为一个正整数，τ 为当前需翻倒的集装箱

$$D'(s,g,\lambda)=\{x\in J\mid x\in B(s)\wedge|c_j|<\lambda\} \tag{5.7}$$

$|c_j|$ 表示栈 j 中集装箱的个数，即当前的层数.

这样有：

$$N(s)=\{s':s'=T(s,x),\forall x\in D'(s,g,\lambda)\} \tag{5.8}$$

14			3		
1		18	4	16	17
19	15	6	5	9	10
8	13	2	11	12	7
1	2	3	4	5	6

图 5.3 图示说明

Fig.5.3 Icon description

以图 3 为例说明如下：$k=1,\tau=14$，$J=\{2,3,4,5,6\}$，$a_2(s)=13,\ a_3(s)=2,\ a_4(s)=3,\ a_5(s)=9,\ a_6(s)=7$ 相应的 $b(13)=1,\ b(2)=2,\ b(3)=0,\ b(9)=1$，$b(7)=2$，设 $g=2$，$\lambda=4$，则 $B(s,g)=\{2,4,5\}$，$D'(s,g,\lambda)=\{2,5\}$。

关于 g、λ 的选取，对于问题本身来说，g 越小越能保证搜索空间的减小，然而必然会使问题的解远离最优解，另一方面，g 越小越能使得 $D(Y)$ 的值减小；另一方面如果 g 的值太小会使 $D'(s)$ 的元素个数小于 $|t_k|$ 的值，这样就是的算法无法进行下去。综上讨论，在这里提每个 k 时 g 先取一个初值 $g=g_0$（为一正整数）分别判断 $D'(s)$ 的元素个数和 $|t_k|$ 的值，若 $D'(s)$ 的元素个数小于 $|t_k|$ 的值就取 $g=g+1$，直至 $D'(s)$ 的元素个数不小于 $|t_k|$ 的值；λ 的选取可根据操作机械的特点以及集装箱实际受压排放高度要求来取，在这里根据码头堆场的实际要求取得。

5.3 实验分析

在本实验中对堆放中经常出现的 4×6，4×7，5×6，5×7，6×6，6×7 规模贝位分别随机产生 40 个例子进行实验。为了说明本文算法的优越性，把 g 的初值 g_0 分别取初始值 1 和 2，并和 M. Caserta et al[35]的算法优化效果上作比较，结果如表 1 所示。从表 5.1 可以看出对于这几种规模的贝位，本文所提出的算法对于贝位平均翻箱数的平均求解效果均优于 M. Caserta et al[35]的算法 C-M，并且 $m=1$ 时本文算法的优化效果最好。

表 5.1 算法比较

Table 5.1 The comparison of the algorithms

贝位层	贝位栈	最高高度	算法 C-M 平均倒箱数	本章算法取 g_0=1 平均倒箱数	本章算法取 g_0=2 平均倒箱数
4	6	5	15.50	15.500	15.480
4	7	5	18.11	18.000	18.010
5	6	6	26.85	26.400	26.560
5	7	6	28.54	28.040	28.080
6	6	7	38.95	38.070	38.000
6	7	7	42.38	42.000	41.800

通过比较可知本章所提的基于改进的 C-M 模型的算法在平均倒箱数上较之第四章所提算法有较好的优化结果，在倒箱均衡性方面效果差别不大。在运算时间上看，由于本章算法利用的基本框架是搜索方法，故运算效率不及第四章所提的基于启发式的算法。在表 5.2 中给出了第四章所提算法当 L=2 时和本章所提算法

g0=2 时的常见贝位状态的 40 个实例的平均计算时间比较数据。

表 5.2 第四章所提算法和本章算法在 L=2，g_0=2 时计算时间比较

Table 5.2 Algorithm of the previous chapter and algorithm of this comparison of computation time

贝位层	贝位栈	最高高度	前章算法 L=2 时平均计算时间（s）	本章算法 g_0=2 时平均计算时间（s）
4	6	5	0.04	0.05
4	7	5	0.04	0.05
5	6	6	0.06	0.07
5	7	6	0.07	0.09
6	6	7	0.08	0.10
6	7	7	0.08	0.12

为了进一步分析算法在对单次提箱耗时波动性控制方面的优越性，在表 5.3、表 5.4 中分别对 4×6 、5×6 的 50 个实验作统计分析，列出了对于不同取箱代价的集装箱的百分比，并列出了不同算法的不同取箱代价的集装箱百分比的样本方差 D(Y)，用来度量单次提箱耗时波动性。从表中可以看出在各种规模的实验本文算法得到的效果在控制波动性方面均有一定的提高。

表 5.3 4×6 型贝位实例比较

Table 5.3 The comparison of 4×6 model

	倒箱数为 0	倒箱数为 1	倒箱数为 2	倒箱数为 3	倒箱数为 4	D(Y)
C-M	65.2%	17.0%	9.4%	8.0%	1.2%	0.064
g_0=1	60.4%	24.1%	10.5%	6.7%	0.1%	0.056
g_0=2	63.0%	18.0%	13.6%	6.7%	0.1%	0.061

表 5.4 5×6 型贝位实例比较

Table 5.4 The comparison of 5×6 model

	倒箱数为 0	倒箱数为 1	倒箱数为 2	倒箱数为 3	倒箱数为 4	倒箱数为 5	D(Y)
C-M	58.9%	16.5%	10.8%	6.7%	5.3%	1.8%	0.045
g_0=1	55.5%	21.1%	11.5%	6.5%	5.2%	0.3%	0.041
g_0=2	58.2%	15.1%	15.3%	6.5%	5.1%	0.2%	0.043

5.4 小结

进一步讨论了第四章所提出的装船时贝内取箱倒箱优化最优化问题——在既定的初始堆存状态下，以最小化倒箱总次数和确保作业机械装船系统的流畅性为优化目标。本章对贝内的倒箱问题进行了进一步的研究，设计了优化搜索算法可以保证在较短的时间内得到问题的较优解。通过建立改进的 C-M 模型以及 C-M 启发式优化算法合理选取候选栈既保证了单次提箱的流畅性又较好的限制了贝位提箱总倒箱率，较之第四章所提算法优化效果较好，并弥补了第四章所提算法的自适应性较低的问题，并通过实验验证了本章所提算法的优越性，进一步为码头集装箱的智能化管理提供了坚实的理论基础。

6 结论与展望

在集装箱码头堆场中，集装箱取箱时的倒箱操作直接影响码头的作业效率，甚至成为了影响港口码头吞吐量的重要因素。对于翻倒箱落箱位置的确定，由于不确定因素多，实时性要求高，在实际作业中，有的依靠码头规划调度系统中提供的简单规则，有的则完全依靠现场操作人员的经验，这些简单的规划，不可避免的产生不必要的误判和作业效率的低下。本文提出的算法优化结果较优、实时性较高、切实可行，所需相关数据可从现有的码头作业规划调度系统中得到。

在作业过程中，往往不可避免的由于某些不确定因素造成取箱顺序临时改变，或者场桥司机没有按照指定的位置落箱，这时就需要根据现场变化实时更新相应的数据，重新计算翻倒箱的落箱位置。将最新方案传输到移动终端。本文算法实时性较好，完全可以满足系统的实时性要求。

应用本文提出的算法对取箱过程中所有翻倒箱的落箱位置进行计算，将当前翻倒箱的最佳落箱位置以可视化的方式显示在场桥的移动终端上，规范取箱时的倒箱操作，从而最大限度地减少倒箱作业次数，避免不合理的倒箱所造成的无效作业、提高堆场的作业效率。

本文算法的创新之处在于：

一、根据集装箱倒箱问题的实际情况，结合博弈模型的特点，提出了集装箱倒箱博弈模型及启发式优化算法，实验数据表明该算法确实优化了倒箱次数，为码头节约成本；

二、根据出口箱装船的特殊情况，本文在预期倒箱数的概念基础上提出了一种自适应算法，该算法在保证机械运作的流畅性、调度的合理性基础上优化了总倒箱数；

三、进一步针对出口箱装船问题，在前人提出的C-M模型的基础上通过改进通道设置，把该算法应用到出口集装箱的倒箱优化问题上在保证机械运作的流畅性、调度的合理性基础上进一步优化了总倒箱数；

进一步的工作：

一、对于一些集装箱码头可以根据自身的条件进行预翻箱问题，可以进一步研究机械在空余时间对集装堆场进行预翻箱的算法，结合本文提到的算法做成切实可行的倒箱系统；

二、对于自身有条件的码头，可以在堆场设置运输临时中转站，确保集装箱发箱的顺序性，对于这种问题可以考虑改进列车车厢调度算法等相关算法使之适合集装箱发箱问题；

三、由于龙门吊等机械移动不灵活，移动成本高，可以进一步在考虑龙门吊运动成本的基础上改进本文算法，使之确实能在实际运用中在节约时间的基础上节约总成本；

四、考虑跨贝位的三维空间集装箱倒箱问题，在考虑机械调度的基础上在整个箱区综合考虑倒箱问题；

五、本文只针对倒箱问题进行了研究，对于堆场的其他优化问题没有做深入的讨论，今后可在堆场整体布局、龙门吊合理分配、岸边船只调度，岸桥合理分配等方面做进一步的研究，并在此基础上综合考虑集装箱堆场的优化问题，使该多阶段优化问题得到切实可行的深入研究。

总之，集装箱堆场优化研究工作还在进一步深入研究阶段，各水运大国都投入了大量资金进行研究，但由于本课题是一个多阶段多目标的优化问题，很多成果在实际运用中表现的并不理想，大量的工作还有待于今后进一步研究和完善。

致　　谢

时光飞逝，转眼研究生生活就要过去，值此论文完成之际，向曾经关心、帮助、支持过我的所有老师、同学、朋友、家人表示最诚挚的谢意！首先，衷心感谢我的导师易正俊老师的悉心教导，无论是做人，还是做学问，易老师都言传身教。易老师谦虚严谨的科学作风也使我深受启发，从易老师身上学到了很多做人的道理和处事方法，这些都将帮助我更好的面对未来。

感谢我的父亲、母亲、哥哥、朋友，如果没有他们的精神上和物质上给我的支持和帮助我不可能有机会读研究生，更不可能完成此论文，是他们的鼓励与支使持使我具有不怕困难精神和迎难而上的恒心。我衷心的祝愿他们健康长寿，生活快乐！

在此，向曾经关心、帮助、支持过我的所有老师、同学、朋友、家人表示最诚挚的谢意！

李保顺

二O一O年十一月 于重庆

参考文献

[1] 袁福昌等.集装箱装卸搬运机械[M]. 武汉: 港口装卸杂志社，1988.

[2] 白治江.动态负载下堆场资源规划的在线决策[J]. 上海海事大学学报, 2010, 31(3), 52-57.

[3] 荣朝等主编.集装箱多式联运与综合物流[M]. 北京: 中国铁道出版社，2001.

[4] Y. W. Zhang, W. J. Mi, et al. An optimization model for intra-bay relocation of outbound container on container yards [J]. International Conference on Automation and Logistics, 2007, 8, 18-21.

[5] H. R. Choi, H. S. Kim, B. J. Park, N. K. Park, S. W. Lee. An ERP Approach for Container Terminal Operating Systems [J]. Maritime Policy & Management, 2003, 30(3), 197-211.

[6] S. Hartmann. Generating Scenarios for Simulation and Optimization of Container Terminal Logistics [J]. OR Spectrum, 2004, 26(2), 171-193.

[7] M. Bielli, A. Boulmakoul, M. Rida. Object Oriented Model for Container Terminal Distributed Simulation [J]. European Journal of Operational Research. 2005, 15(3), 72-83.

[8] K. G. Murty, J. Liu, Y. Wan, R. Linn. A Decision Support System for Operations in a Container Terminal Decision Support Systems [J]. 2005, 39(3), 309-332.

[9] K. H. Kim, K. C. Moon, Berth Scheduling by Simulated Annealing [J]. Transportation Research Part B: Methodological, 2003, 37(6), 541-560.

[10] A. Imai, E. Nishimura, S. Papadimitriou. The Dynamic Berth Allocation Problem For a Container Port [J]. Transportation Research-B, 2001, 35(4), 401–417.

[11] A. Imai, E. Nishimura, S. Papadimitriou. Berth Allocation with Service Priority [J]. Transportation Research Part B: Methodological, 2003, 37(5), 437- 457.

[12] A. Imai, X. Sun, E. Nishimura, S. Papadimitriou [J]. Berth Allocation in a Container Port: Using a Continuous Location Space Approach [J]. Transportation Research Part B: Methodological, 2005, 39(3), 199-221.

[13] D. Wilson, P. A. Roach, Principles of Combinatorial Optimization Applied to Container-Ship Stowage Planning [J]. Journal of Heuristics, 1999, 5(4), 403-418.

[14] D. Ambrosino, A. Sciomachen, E. Tanfani [J]. Stowing a Containership: The Master Bay Plan Problem [J]. Transportation Research Part A: Policy and Practice, 38(2), 81-99, February 2004.

[15] A. Imai, K. Sasaki, E. Nishimura, S. Papadimitriou. Multi-Objective Simultaneous Stowage and Load Planning for a Container Ship with Container Rehandle in Yard Stacks [J]. European Journal of Operational Research, 2004, 15(2), 3-18.

[16] C. F. Daganzo. The Crane Scheduling Problem [J]. Transportation Research-B, 1989, 23(3),

159–175.

[17] R. I. Peterkofsky, C. F. Daganzo. A Branch and Bound Solution Method for the Crane Scheduling Problem [J]. Transportation Research Part B: Methodological, 1990, 24(3), 159-172.

[18] Y. M. Park, K. H. Kim. A Scheduling Method for Berth and Quay Cranes [J]. OR Spectrum, 2003, 25(1), 1-23.

[19] K. H. Kim, Y. M. Park. A Crane Scheduling Method for Port Container Terminals [J]. European Journal of Operational Research, 2004, 156(3), 752-768.

[20] S. P. Sgouridis, D. Makris, D. C. Angelides. Simulation Analysis for Midterm Yard Planning in Container Terminal [J]. Journal of Waterway, Port, Coastal & Ocean Engineering, 2003, 129(4), 178-188.

[21] H. Yang, Y. S. Choi, T. Y. Ha. Simulation-Based Performance Evaluation of Transport Vehicles at Automated Container Terminals [J]. OR Spectrum, 2004, 26(2), 149-170.

[22] Y. G. Chung, S.U. Randhawa, E. D. McDowell. A Simulation Analysis for a Transtainer-Based Container Handling Facility [J]. Computers & Industrial Engineering, 1998, 14(2), 113-125.

[23] M. T. Ibrahimi, B. De Castilho, C. F. Daganzo. Storage Space vs. Handling Work in Container Terminals [J]. Transportation Research B, 1993, 27, 13-32.

[24] K. H. Kim. Evaluation of the number of re-handles in container yards [J]. Computers and Industrial Engineering, 1997，32 (4)：701–711.

[25] K. H. Kim, J. W. Bae. Re-Marshaling Export Containers in Port Container Terminals [J]. Computers & Industrial Engineering, 1998, 35(3)-(4), 655-658.

[26] 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, 89-101.

[27] K. H. Kim, G. P. Hong. A Heuristic Rule for Relocating Blocks [J]. Computers & Operations Research, 2006, 33, 940-954.

[28] 白治江,王晓峰.集装箱翻箱优化方案设计[J]. 水运工程，2008, 4:57-61.

[29] 徐亚等.集装箱倒箱问题的启发式算法研究[J]. 系统仿真学报，2008, 20(14): 3666-3674.

[30] 侯定丕. 博弈论导论[M]. 合肥: 中国科学技术大学出版社，2003

[31] 董琳,刘庆敏,王超,王晓,吕长虹.集装箱翻箱问题的模型分析及算法[J].经济数学，2006, 23(2):181-186.

[32] 张维英等，出口集装箱堆场取箱作业优化模型研究[J]. 武汉理工大学学报(交通科学与工程版), 2006, 30(2):314-317.

[33] 张艳伟，石来德，宓为建. 出口集装箱堆场翻箱问题优化模型研究[C]. 2007 机械工程全国博士生学术论坛，武汉, 2007: 438-446.

[34] M. Sniedovich, S. Voß. The corridor method: a dynamic programming inspired Metaheuristic[J], Control and Cybernetics, 2006, 35(3), 551-578.

[35] M. Caserta et al. Applying the corridor method to a blocks relocation problem [J], OR Spectrum, Published online: 18 June 2009.

[36] N. Y. Hsu. An optimal network model for pre-marshalling in the container yard [D]. Master Thesis, Department of civil engineering, National Cheng Kung University, 2002.

[37] C. H. Sung. A heuristic of the pre-marshalling problem in the container yards [D]. Master Thesis, Department of civil engineering, National Cheng Kung University, 2004.

[38] J. Kang, M. S. Oh, E. Y. Ahn, K. R. Ryu K.-H. Kim. Planning for intra-block remarshalling in a container terminal [J]. IEA/AIE, LNAI 4031, 2006, 1121-1220.

[39] I. Watanabe. Characteristics and analysis method of efficiencies of container terminal-an approach to the optimal loading/unloading method [J]. Container Age, 1991, March: 36–47.

[40] M. Avriel, M. Penn, N. Shpirer, S. Witteboon. Stowage planning for container ships to reduce the number of shifts [J]. Annals of Operations Research 1998, 76: 55–71.

[41] Y. Hirashima, N. Ishikawa, K. Takeda. A new reinforcement learning for group-based marshaling plan considering desired layout of containers in port terminals [J]. Sensing and Control, 2006, 5, 670-675.

[42] Y. Lee, N. Y. Hsu. An optimization model for the container pre-marshalling problem [J]. Computer & Operations Research, 2007, 34, 3295-3313.

[44] Y. Hirashima, N. Ishikawa, and K. Takeda. A new reinforcement learning for group-based marshaling plan considering desired layout of containers in port terminals [J]. Sensing and Control, 2006, 670-675.

[45] Y. Lee and N.-Y. Hsu. An optimization model for the container pre-marshalling problem[J]. Computer & Operations Research, 2007, 34, 3295-3313.

[46] K. H. Kim, H. B. Kim. Segregating space allocation models for container inventories in port container terminals [J]. International Journal of Production Economics, 1999, 59(1): 415-423.

附　　录

A. 作者在攻读学位期间发表的论文目录

[1] 易正俊,李保顺,李新强.集装箱堆场倒箱博弈启发式优化算法.上海海事大学学报.2010.31(3):47-51.

B. 作者在攻读学位期间参加的科研项目

[1] 军工项目：车辆预先维修及事故预防系统，项目编号：JW20*2008042.

[2] 重庆市科技攻关项目：大型设备智能预防性维护与自动控制系统，项目编号：CSTC2009AC3037.