天津大学

硕士学位论文

集装箱堆场分配问题的启发式方法研究

姓名：王维圳

申请学位级别：硕士

专业：管理科学与工程

指导教师：李波

20080501

中文摘要

随着经济一体化、全球化趋势的发展和市场经济的不断完善，我国国民经济和对外贸易迅速增加，集装箱运输量取得迅猛的发展，快速的集装箱运输需求使我国现有的不少港口出现了能力不足的现象。集装箱堆场是港口中用于堆存集装箱的专设区域，其作为港口主要的组成部分，在集装箱港口作业物流流程中起着很重要的作用，所以堆场空间分配的优劣将直接影响港口整体的效益。

本论文以集装箱港口堆场空间分配为研究内容，对于一系列的空间请求，运用新算法，来最小化其堆场空间占用。

主要内容包括：

第一部分，首先阐述了我国港口现状以及集装箱堆场的概念、功能和其运作程序。介绍了一些用于解决堆场空间分配问题的算法，如模拟退火、遗传算法、禁忌算法等。

第二部分，基于递归算法，使用一种“下落”（DROP）方法来分配堆场空间。该方法对于给定了优先序列的一组集装箱空间存放请求，通过递归思想来分配空间，以达到空间占用的最小化。通过仿真试验，提出解决集装箱堆场空间分配问题的关键是找到空间请求的最优序列的结论。

第三部分，针对如何找到最优序列，提出一种关键请求局部邻近搜索方法。该方法先使用一种基于时间排序的方法来产生初始序列，然后通过寻找关键请求进行局部邻近搜索，改进优先序列的质量，最终找到最优序列。并通过仿真试验来验证算法的有效性。

第四部分，作为研究的拓展，分析了堆场调度信息系统的相关功能和架构问题，及如何采用 OpenGL 技术来实现其中显示问题。

关键词：堆场空间分配;；关键请求；局部搜索；调度信息系统

ABSTRACT

In the past few years we have seen the improvement of free market and the globalization of the trade. The development makes the logistics and transportation more and more important, particularly in the marine transportation systems. As the part of the container terminal, terminal yard is a zone specially for storing containers, which plays a very important role in the whole container terminal operational logistics.

This thesis, using the heuristic method, studies on the container terminal yard allocation problem.

The main works are:

Firstly, the paper expounded China's port status and container yard concept, function, operation procedures. Introduced a number of algorithms to solve the problem of yard space allocation, such as simulated annealing, genetic algorithms, tabu algorithms.

Secondly, based on recursive algorithm, a yard approach to the yard space allocation, named DROP. The method for a group of container storage space request that have been given the priority sequence, recurrent allocation of space, to achieve the minimum use of space. Through the simulation experiments, the key to solve container yard space allocation problem is to find the optimal sequence space request.

Thirdly, for how to find the optimal sequence, giving a critical request local neighborhood search method. The method uses a sort of time-based method to produce the initial sequence, and then searches for critical request, improves the quality of priority sequence by neighborhood local search method, eventually finds the optimal sequence. And the simulation experiments to verify the effectiveness of the approach.

Finally, as search and development, analyzing the functions and structure of the yard scheduling information system, and how to use OpenGL technology.

KEY WORDS: Yard Space Allocation; Critical Request; Local Search; Scheduling Information System

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：王维钿　　签字日期：2008年 6 月 6 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：王维钿　　导师签名：李波

签字日期：2008年 6 月 6 日　　签字日期：2008年 6 月 6 日

第一章 绪论

1.1 论文背景及意义

集装箱运输是社会生产大发展的产物。它是一种先进的运输方式，不仅促进了水、陆、空各种运输工具之间的联运，而且解决了复杂而零星的小包件货物的运输问题，同时也吸引了大量的整车适箱货物。近年来，集装箱运输量取得了迅猛的发展，快速增加的集装箱运输需求使我国现有的不少集装箱港口出现了能力不足的现象。这就不仅要求港口要不断增大容量来适应快速增加的货源需求，而且要求港口管理者按照现代物流的要求，合理配置港口内部的物流资源，优化物流资源的流通渠道，完善物流信息系统，以不断提高港口内部物流网络运作效率，降低作业成本。而在港口中，集装箱堆场空间资源的优化利用正是达成此目的的关键。

1.1.1 现代港口的组成

由于科学技术不断进步和经济不断发展，以运输、装卸、包装、仓储的合理化和系统化、加工配送一体化及信息管理网络化为主要内容的现代物流，不仅成为了企业竞争的主要因素，还成为一个国家综合国力的重要标志。现代物流的发展要求货物能够在不同运输方式、各货运站之间实现无缝连接，以最小的费用、最短的时间，准确、安全地实现时间和空间的转移。

港口作为物流系统中的重要节点，具有运输、工业和商业等多种功能，是一个国家和地区的重要经济资源。一般由水域和陆域两大部分组成。水域是提供船舶进出港，以及在港内运转、锚泊和装卸作业使用，因此要求它有足够的水深和面积，水面基本平静，流速和缓，以便于船舶的安全操作。陆域提供旅客上下船以及货物的装卸、堆存和转运使用，因此陆域必须有适当的高程、岸线长度和纵深，以便在这里安置装卸设备、仓库、堆场、铁路、公路以及各种必要的生产、生活设施。

1．港口水域。水域是港口最主要的组成部分，港口的水域主要包括港池、锚地和航道。港池一般指码头附近的水域，需要有足够的深度和宽广的水域，供船舶靠离操作。对于开敞海岸港口，为了阻挡海浪或泥沙的影响，保持港内水面平静与水深，必须修筑防波堤。锚地是提供船舶抛锚候潮、避风、办理进出口手

续、接受船舶检查或过驳装卸等停泊的水域。

2．港口陆域。凡是在港口范围的陆地面积统称为陆域，陆域由以下几个部分组成。

（1）码头与泊位。码头主要供船舶停靠，以便旅客上下、货物装卸，码头前沿线即为港口的生产线，也是港口水域和陆域的交接线。泊位是提供船舶停泊的位置，一个泊位即可供一艘船舶停泊，泊位的长度依船型的大小有所差异。

（2）仓库和堆场。它们提供货物装船前和卸船后短期存放使用的。

（3）铁路及道路。货物在港口的集散还需要利用陆路交通，因此铁路和公路是港口陆域上的重要设施。当有大量货物需要用铁路运输时，需设置专门的港口车站，公路对于港口货物的集散也起着重要的作用，对于有集装箱运输的港口，道路系统尤为重要。

（4）起重运输机械。现代港口装卸工作基本是由各种机械来完成的，用来起吊货物的机械称为起重机械，用于搬运货物的机械成为运输机械。它们在港口可对船舶、火车和汽车进行装卸工作，在船舱内进行各种搬运、堆码和拆垛工作，在货场进行起重、搬运、堆码、拆垛等工作。

（5）辅助生产设施。为维护港口的正常生产秩序，保证各项工作得以顺利进行，港口还需要在陆域上配备一些辅助设施，如给排水系统、输电、配电系统、燃料供应站、船舶修理站、公众办公用房等。

1.1.2 中国集装箱港口的发展

中国（有关数字未包括中国香港特别行政区和中国台北）的集装箱货运始于20世纪70年代后半期。自此之后，中国的集装箱海上运输迅猛发展。港口集装箱码头也在沿海和内河纷纷设立，现今具有装卸集装箱能力的大小港口已达 57个，其中沿海港口38个，内河港口19个。在这些港口中，共有集装箱泊位258个，其中万吨级以上的泊位83个，千吨级以及上至万吨级以下的泊位175个。伴随着我国经济贸易的高速发展，港口的集装箱吞吐量迅猛增加，表1-1为2007年全国国际标准集装箱吞吐量前10名排序表。

表 1-1 全国国际标准集装箱吞吐量前 10 名排序表

名次	港名	2007 年 12 月（万 TEU）	2007 年 1-12 月（万 TEU）	同比增幅%
1	上海港	229.72	2615.00	20.45
2	深圳港	229.37	2109.91	14.24
3	青岛港	81.66	946.20	22.85
4	宁波-舟山港	76.48	936.00	32.42
5	广州港	81.23	920.00	39.39
6	天津港	61.81	710.29	19.37
7	厦门港	42.90	462.70	15.29
8	大连港	34.38	381.30	18.71
9	连云港港	22.95	200.06	53.62
10	营口港	10.74	137.13	35.70

结合当今我国集装箱港口现状，从集装箱吞吐量变化趋势看，未来中国集装箱港口呈现 5 个发展趋势：

1．集装箱港口国际化。随着中国国际贸易地位的提高和对外贸易的扩大，港口集装箱码头业务也开始向国际化发展。特别是在欧盟成为我国最大的贸易伙伴后，中国的贸易格局有了新的变化。为了促进经济贸易发展，改善经济结构和运输结构，扩大对外开放，近年来我国集装箱港口基础设施有了显著改善，国际化进程明显加快，企业规模不断扩大，服务质量日益提高。

2．集装箱港口规模化：随着集装箱船舶的超大型发展，使码头基础设施建设规模发生了根本的变化。航运船舶大型化，这就要求世界上主要的集装箱枢纽港建设向规模化、大型化发展。我国相关部门正在积极建立高效、统一的管理和协调工作机制，逐步改善宏观政策法制环境，进一步规范集装箱运输市场秩序，改善口岸服务设施，合理规划建设集装箱运输基础设施，加强集装箱运输支持保障系统的建设，调整港口功能，鼓励港口走规模化和集装箱化发展模式，发挥主枢纽港的整体功能。

3．集装箱港口集约化：集装箱运输的发展离不开集装箱港口的发展。经过几十年的发展，我国集装箱港口布局和分工已日趋合理，初步形成了大连港、天津港和青岛港为主的北部枢纽港群，以上海港和广州港为主的南部枢纽港群。随着沿海北、东、南三大集装箱主枢纽港群以显雏形，未来各个港口群内各港的定位更加明确，具有中转功能的核心港口逐步凸现，各港群与周边国家港口的竞争

能力将明显增强。使集装箱港口能更加集中合理地运用现代管理与技术，充分发挥人力资源的积极效应。

4. 集装箱港口自动化、信息化。注重加强装卸设备和装卸技术的先进性，集装箱装卸设备逐年更新换代，装卸工艺自动化程序逐年提高，设备种类和系统配套逐步完善。同时，港口信息化管理水平也不断提高，各大港集装箱码头装卸公司与航运公司、口岸、海关等各单位实施集装箱电子数据交换，实现了无纸化操作，简化了手续，提高了作业效率。

5. 集装箱港口物流化。随着集装箱运输全球物流系统的逐渐完善，港口在物流链中的纽带作用正向多元化发展，即以港口为平台和载体，建设临港出口加工区、集装箱保税区、集装箱物流中心、集装箱中转站等，多方面开拓具有积疏运功能的新领域和新市场，扩大港口综合管理业务，提高运营服务水平。

1.1.3 信息技术在港口中的应用

以通讯技术、网络技术、感测技术、控制技术等为代表的现代信息技术的发展以及全球信息网络的兴起，使港口管理从传统的工业时代进入了一个崭新的信息时代，为集装箱港口进行快速、高效生产提供了可能。

目前，信息技术已经在港口航运企业得到了广泛的应用。例如，我国沿海一些先进的大型集装箱港口用无线电通讯代替了高频对讲机，使港口内部物流信息和作业信息传递更为快速、准确。在新加坡港和香港，EDI（电子数据交换）技术的使用和普及，实现了通关过程电子化，使通关可在半个小时左右完成，大大提高了通关效率。完整的 GIS（地理信息系统）分析软件集成了车辆路线模型、最短路径模型、网络模型、分配模型和定位模型，为最优路径选择和最佳运输任务分配提供了决策支持。除此之外，条码技术、射频技术（RF）和全球定位系统（GPS）等技术的广泛应用都为集装箱港口内物流高效运作提供了保障。

1.1.4 本论文研究意义

集装箱港口是资本密集型的经济实体，要使港口作业高效而经济地运行，集装箱港口作业资源的优化配置关系到港口的运作效率及其竞争力。而集装箱港口堆场作为港口中用于堆存集装箱的专设区域，在集装箱港口作业物流流程中起着很重要的作用，堆场空间利用率的大小将直接影响到港口整体的效率和收益。合理的集装箱堆场空间分配不仅能够提高堆场的利用率，而且能够提高堆场机械的作业效率，减少桥吊的等待时间，缩短船舶的停港时间。

本文将以集装箱港口堆场空间利用率为研究对象，考虑每个请求在一段时间内的空间需求，尽可能使堆场空间占用最小化，以提高堆场空间利用率，节省成

本。本论文的分析与研究将为我国集装箱港口堆场管理的利用和运作提供一定的指导和决策依据。

1.2 集装箱堆场空间利用理论与方法的研究现状

1.2.1 国外研究发展现状

国外对集装箱港口堆场空间利用问题研究起步比较早，Sculli 和 Hui[1]通过仿真的方法研究了堆场堆高、堆场利用率和翻箱率之间的关系。E.mcdowill et.al[2]在文中讨论了集装箱港口的作业流程，在考虑了各作业流程所发生成本基础上，建立了一个成本模型来解决堆场堆存问题。在这种成本模型下，虽然可以降低总的作业成本，但是没有考虑到效率的问题。Taleb-Ibrahimi et al.[3]也采用仿真的方法讨论了出口箱区不同安排原则下堆高、堆场利用率和翻箱率之间的关系。De Castihno and Dagano[4]一文中研究了两种策略下的进口箱堆存空间分配问题。Peter Preston，Erhan Kozan[5]在文中讨论了集装箱堆场的最优堆存方案，并用遗传算法求解。Jean-Francois Cordeau[6]研究了堆场集装箱配置问题，并建立了整数规划模型，目标函数是使集装箱从堆场箱区到船舶的运输距离最小，并用 TABU 搜索方法和遗传算法求解。Ping Chen et al[7]在文中讨论了堆场集装箱空间分配问题，文中提出用多种算法解决这个问题，包括 TABU 搜索算法，模拟退火算法，遗传算法以及 SWO 优化算法，最后结合 TABU 搜索方法和 SWO 优化算法的特点，提出了一种混合方法来求解。Chuqian Zhang[8]通过建立一个线性整数规划模型来解决堆场集装箱的空间配置问题，模型可得到分配到各箱区作业的集装箱数量。EbruK.Bish[9]在文中对将进口箱堆场空间配置问题同桥吊调度问题整体考虑。并把这一问题分成两个子问题并分别进行考虑：（1）为每个卸载的集装箱确定在堆场的存贮位置；（2）为每个集装箱分配集卡；（3）对每个桥吊进行装卸作业优化以实现集装箱运输船的周转时间最短。最后，作业采用一个 TLS（Transshipment Problem Based List Scheduling Heuristic）启发式规则对大型问题进行求解并进行了最坏情形的分析。

1.2.2 国内研究发展现状

近年来，国内也有一部分学者对堆场问题进行了研究。徐剑华[10]提出用择箱指数方法寻求集装箱最佳堆存高度和最佳布置，以使堆场的面积利用率和堆场的取箱率获得综合优化的结果。宋天威[11]以上海港军工路集装箱码头为例对两种堆码方式进行了探讨。郝聚民，纪卓尚等[12]介绍了出口集装箱堆场优化的重要意义，

在图搜索技术和模式识别理论的基础上建立了随机条件下的混合顺序作业堆场优化模型。杨淑琴，张运杰等[13]根据出口集装箱先到港、尽量轻箱在下重箱在上的原则，采用启发式方法合理安排出口集装箱堆场箱位安排。

1.3 论文的主要工作和结构

1.3.1 论文的主要工作

本文的研究方向为如何最大化的利用集装箱堆场空间，为此将从集装箱空间存放请求的优先序列入手，使用一种关键请求局部邻近搜索的启发式方法来解决优先序列的问题，拟采用计算机仿真技术进行算法的模拟，来对结果进行检验分析。

本文要解决的主要问题包括：

（1）集装箱堆场空间利用的相关理论研究，为解决问题提供理论基础。主要包括国内外研究此问题用到的遗传算法、模拟退火等启发式算法。

（2）研究解决此问题的启发式算法。在理论学习的基础上，改造已有的启发式算法，提出一种新的方法来产生初始序列，并用一种关键请求局部邻近搜索的方法来寻找一系列集装箱堆场空间存放请求的优先序列，以便能够使生成的序列结果质量更高，速度更快。

（3）运用新的算法进行仿真试验，检验新算法是否达到设计预期，并对算法的结果进行分析比较和进行评价，寻找是否还有改进的可能。

（4）研究如何使集装箱堆场空间调度的系统设计问题，包括如何分别对集装箱前方堆场和后方堆场的调度系统进行分析设计。

1.3.2 论文的结构

本文共分为五章，分别对国内外现状，基本理论知识，基本模型和启发式算法，集装箱堆场空间调度系统设计等几个部分进行了分析研究，具体包括：

第一章是绪论，主要介绍了目前该方向的国内外研究概况以及论文的研究目的、意义及要主要工作等内容。

第二章主要介绍了集装箱堆场概念，特点和功能；并针对本论文将要采用的相关理论，如 NP 问题和贪婪法的基本理论知识进行了阐述。

第三章阐述了集装箱堆场空间分配问题的基本概念，并说明其属于动态空间分配问题，建立空间分配的基本数学模型，使用一种名为“下落”（DROP）的空间分配方法，递归使用此方法以达到最小化空间占用的目的。

第四章介绍了一种启发式算法来解决一系列空间请求的排序问题，提出了先使用基于时间排序的方法来产生初始序列，然后利用交换（Swap）和移动（Move）方法进行局部邻近搜索，最后用改变关键请求优先级的方法来摆脱局部最优，求得最优序列。

第五章阐述了集装箱堆场调度系统的分析与设计，包括集装箱前方堆场调度系统的分析与设计，以及后方堆场调度系统的分析与设计，将本文中算法应用到堆场调度系统中去，并应用 OpenGL 技术解决系统的三维模拟实现问题。

论文的内容和结构见图 1-1：

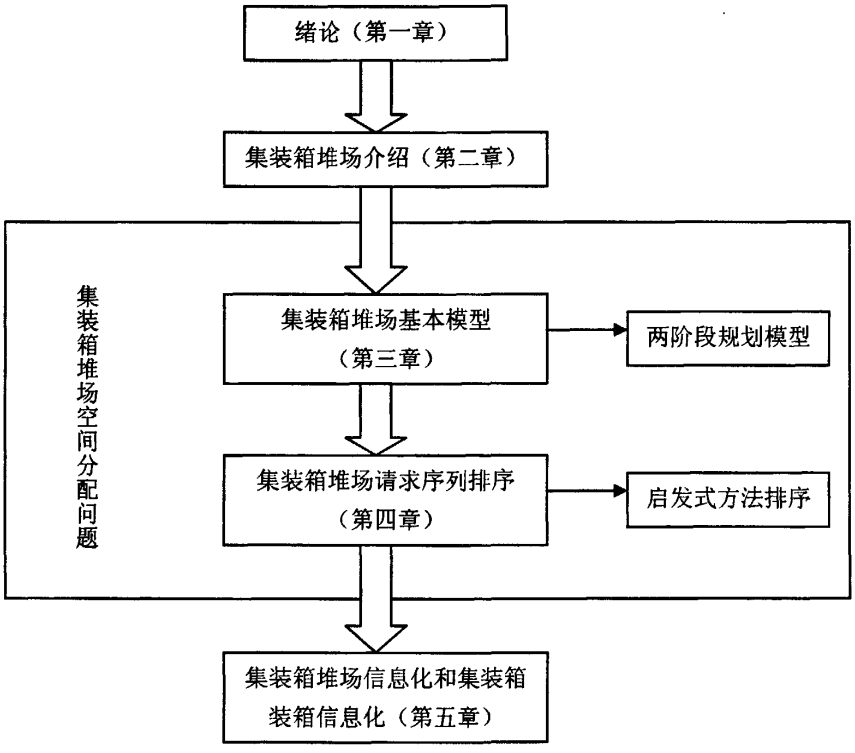


图 1-1 论文内容和结构

第二章 集装箱堆场空间分配问题及算法的相关理论基础

集装箱堆场空间分配问题在国内外都已经有了一定的研究，在这一章我们会首先对集装箱堆场的基本概念作介绍，然后会重点介绍解决堆场空间分配这类问题已有的理论及其算法。

2.1 集装箱堆场概念

2.1.1 集装箱堆场的基本概念

集装箱运输除了海上运输之外，更多的是在陆地上的运输。其中集装箱堆场在集装箱运输中就是一个不可缺少的组成部分。集装箱堆场，有些地方也叫场站，作用是把所有出口客户的集装箱在某处先集合起来（不论通关与否），到了出港时间之后，再统一上船。也就是说，堆场是集装箱通关上船前的统一集合地，等待通关的集装箱货物存放其中，这样便于船公司、海关等进行管理。

集装箱堆场一般分为前方堆场和后方堆场。前方堆场是指在集装箱码头前方，为了加速船舶装卸作业，暂时堆放集装箱的场地。其作用是当集装箱船到港前，有计划有次序的按积载要求将出口集装箱整齐的集中堆放，卸船时将进口集装箱暂时堆放在码头前方，以加速船舶装卸作业。

后方堆场是集装箱重箱或空箱进行交接、保管和堆存的场所。有些国家对集装箱堆场并不分前方堆场或后方堆场，统称为堆场。集装箱后方堆场是集装箱装卸区的组成部分，是集装箱运输“场到场”交接方式的整箱货进行办理交接的场所。

2.1.2 集装箱堆场的发展趋势

集装箱码头的运作环境在不断的变化，经济全球化和货物的集装箱化给世界集装箱运输带来了飞速的增长。全球集装箱吞吐量 1980 年仅为 3600 万 TEU，而 2004 年达到 3.6 亿 TEU，德鲁里航运咨询公司预测 2008 年世界集装箱吞吐量将达 5.7 亿 TEU。集装箱吞吐量的快速增长，给集装箱码头带来了前所未有的发展机遇，但是船舶的大型化对深水泊位及码头设施的要求，多式联运的发展使港口优势腹地不复存在，港口的发展带来的环境问题，货主对门到门运输和及时运

输(JIT) 的强烈要求等等，这些问题都将使集装箱码头面临着更大挑战。

为提升竞争优势并降低运营成本，发展大型化与高速化的集装箱船已成为航运市场趋势。随着大型化船舶下水运营，单船在港口的集装箱装卸量已大幅增加，造成码头堆场空间的需求上升。特别是在西欧和亚洲的部分集装箱港口，港口土地资源严重缺乏，集装箱吞吐量已超过港口的通过能力，港口拥堵现象十分严重，集装箱码头堆场已越来越成为制约港口发展的主要因素。

为了解决这一问题，很多港口都在码头外寻找堆场。釜山港出口集装箱多运往码头外部堆场，深圳港赤湾集装箱码头建设了码头外堆场，对于这些国际性的繁忙港口，堆场空间的紧张和其他港口的竞争，都促使港口管理者尽可能的提高堆场空间利用率。

2.2 集装箱堆场功能和容量

2.2.1 堆场功能

集装箱码头堆场在码头上扮演着缓冲区的角色，是供装卸船舶堆放集装箱的场所，同时也是临时保管和向货主交接集装箱的地方，用以暂时将集装箱集中堆存，以利各项连续作业进行。它的主要功能包括：

1．集装箱交接（Interchange）功能。在集装箱运输中，集装箱的交接工作是一个重要的环节，其中集装箱堆场与提/还箱人间的交接往往是划分集装箱经营人（包括集装箱所有人、承运人等）与用箱人（指集装箱货物的托运人、收货人、以及他们委托的货运代理人）责任的基础。在集装箱出口时，船公司应向货主提供适载（Seaworthy）和适货（Cargoworthy）的集装箱，即通常所说的“完好（Sound）的集装箱”；托运人或其代理人在船公司堆场提取空箱时，应该对集装箱进行检查，并在相应的《设备交接单》(Equipment Interchange Receipt，简称 EIR）上与堆场经营人一起签字确认；堆场的集装箱交接功能也因此伴随着其业务的开展而产生。

2．集装箱检验和修理（Survey and Repair）功能。集装箱的使用寿命一般在 10 年左右。在这段时间之内，无论是由于集装箱的自然消耗和损坏，还是人为损坏，除了失去修理价值，在集装箱投入正常的营运之前，这些损坏的箱子，都必须修理；一般集装箱的检验和修理都要在陆上堆场完成。

3．集装箱堆存（Storage）功能。集装箱从新箱出厂到旧箱彻底报废的整个服务过程，都伴随着集装箱堆存的问题。集装箱公共堆场可能同时堆存数家、甚至数十家经营人的集装箱。又由于各个公司的具体要求不完全一致，加上集装箱

种类多样、状态也经常会发生变化，这都要求集装箱堆场有较高的管理水平，有较为完善的集装箱管理体系。

所有这些活动都对集装箱堆场提出了相当高的要求。在集装箱堆场，货主众多，各种货物进进出出，货物保管和运输的条件各有不同。堆场要保证这些货物安全、准确、快捷和经济的运输，必须要有完善的货运保障机制。这在国内外也有许多成功的经验和模式。

需要进一步说明的是，集装箱堆场有许多形式，各个堆场功能的侧重面也有不同。随着港口集装箱装卸量迅速上升，集装箱港内堆场的场地面积越来越难以满足要求，所以在一些大的集装箱口岸，为了更好的取得港口集装箱装卸的规模效益，集装箱港内堆场的功能主要是集装箱由船舶到陆地或陆地到船舶的集疏运。集装箱堆场的其他各种功能开始向港外转移，于是往靠近码头的区域出现了许多港外集装箱堆场。与集装箱堆场相比，港外堆场也有其一定的优势，它是一个纯粹的集装箱堆场，具有更强的专业性，能够提供更完善的服务。尤其在集装箱的检验和修理方面，港外堆场可以发挥更好的作用。再则，由于场地费用比港区内要低，所以港外集装箱堆场的各项费用也比港内堆场便宜，这也是集装箱经营人要考虑的一个因素。

2.2.2 增大堆场容量的方法

堆场容量与集装箱堆场的可取性一直是码头运营者在堆场管理上所必须取舍的问题。理论上，较低的堆存方式在置入或取出集装箱时，具有较高的存取可及性，例如早期美国海陆公司等土地资源充裕港口采用底盘车系统，直接将集装箱放在拖车上，需移动时，再以牵引车连接将集装箱拖入或拖离堆场。此堆场作业方式虽可获得高度的集装箱存取可及性，但由于必须将集装箱和底盘一同储存，无法堆叠集装箱，大幅度降低了堆场土地所能提供的集装箱储存容量。反之，若利用龙门起重机搭配排列紧密化，堆叠高层化的方式，将集装箱直接放置在地面上，并以垂直堆叠的方式堆储，可大幅提升堆场的储存容量，适合土地资源有限的码头堆场采用。例如新加坡最先进的巴西班让集装箱码头，将集装箱垂直堆叠八至九层。但相对的，集装箱存取可及性则大幅降低，必须进行妥善的事前规划，降低不具生产力的集装箱搬移动作。一般来讲，采用不同的装卸工艺系统时，如底盘车装卸系统、跨运车装卸系统、轮胎式起重机装卸系统、轨道式起重机装卸系统时，集装箱码头堆场容量与集装箱的可取性大致对应的关系如图 2-1。

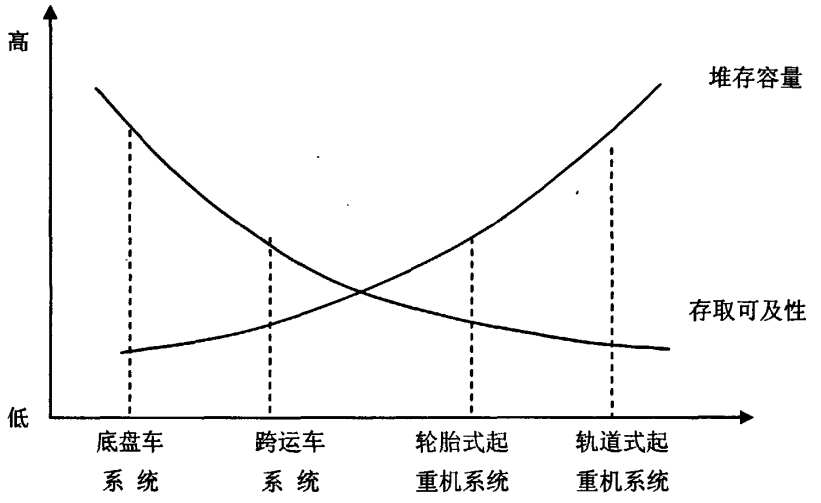


图 2-1 集装箱码头不同装卸系统下堆场容量和存取可及性图

为了减小集装箱码头堆场使用面积，增大集装箱堆场容量，提高集装箱码头的通过能力，增强港口的竞争力，可以采用以下方式：

1．集装箱堆垛高度增加。北美部分港口由于采用底盘车装卸系统，集装箱仅能堆垛一层，堆场利用率很低。在采用龙门起重机装卸系统的码头中，一般集装箱堆垛 3~5 层，有些港口堆 8~9 层，而在空间严重缺乏的港口，堆垛 12 层。部分港口已考虑用龙门起重机替代底盘车或跨运车来提高集装箱堆垛高度。显然，增加集装箱的堆垛高度会提高集装箱堆场利用率。

2．集装箱码头全天候运作。集卡、列车和集装箱船全天候的运作，能增加码头的效率，减少集装箱在码头的堆存时间，提高集装箱堆场的利用率。但由于某些地区当地法律和工会的影响，集装箱码头的全天候运作不能得以实现。同时集装箱码头的全天候运作会造成电力成本的剧增，码头运营者需要全面权衡。

3．减少集装箱在码头的堆存时间。由于各种因素造成集装箱货物到港的时间不确定，如货主提货通知被延迟，一般集装箱码头会让到港的集装箱货物免费堆存一段时间，在国内一般为 7 天，国外某些港口为 3~4 天。或者由于货主认为码头堆场堆存费用较便宜或便利，也会延长货物在码头的堆存时间。集装箱在码头堆存时间越短，对码头堆场面积需求越小，码头堆场的利用率越高。

4．码头智能化运输系统的运用。集装箱码头智能化的运用，能使集装箱码头各个运输装卸环节的配合更顺畅，提高港口的运作效率，减少集装箱在港时间，提高集装箱堆场利用率。码头的自动化是港口发展的趋势，自动化的实现将使集装箱码头提高资源利用率，减少劳动力费用. 提高码头运作效率，将极大解决船

舶大型化带来的单船集装箱装卸的效率问题和对码头堆场空间的需求。荷兰鹿特丹 Delta Terminal 码头、德国汉堡 CAT 码头已经实现了集装箱码头的自动化。但由于技术原因，自动化装卸系统缺乏灵活性和可靠性是这个系统目前为止最突出的内在缺点。

5．码头前沿铁路的运作。码头前沿铁路运输能使集装箱直接从暂存的前沿堆场装卸列车，消除了码头对后方堆场的需求。由于铁路道路口在码头前沿的交叉带来运输的复杂性，加上铁路占去码头前沿大量的土地，目前能实现铁路直达码头前沿的集装箱码头很少。面对港口环境和港口土地资源缺乏的压力，以及港口公路集疏运拥堵的问题，美国洛杉矶港正在规划建设港口码头前沿铁路直达。

本文考虑的情况主要是在集装箱前场如何最大化利用空间资源问题，所以对于堆场的机械装备和信息化水平等不进行相关考虑，只是计算对于一系列的集装箱存放的空间请求，如何让占用空间最小，目标是提高空间利用率。

2.3NP 问题理论

在国内外关于集装箱堆场空间分配问题的研究中，已经被证明该问题属于 NP 问题。

NP 问题是非确定性多项式时间问题的意思（Non-deterministic Polynomial），这类问题随着问题规模的增长，求解的复杂度呈指数增长；若直接求解，即使是最快的计算机也没法承受。

一些典型的 NP 问题有以下几类：

1．推销员问题。一个推销员要在 n 座城市中完成他的推销任务，从某个城市出发，城市之间的距离是不一样的，需要确定一条最优路径，依路径从某城出发再回到该城，所经历的路程最短。

2．包装问题。有 n 个重量小于 1 公斤的物品和足够可以装 1 公斤东西的盒子，现在将物品装于盒子之中，多个物品可以装于一盒，但任何一盒不得重于 1 公斤，试求最小的盒子数。

3．舞伴问题。有 n 个男孩和 n 个女孩参加舞会，每个男孩和女孩都交给主持人一份名单，上面写上他（她）中意的舞伴，舞伴可以不止是一个人，问主持人在收到所有的名单后，是否可以把男孩和女孩分成 n 对，使每人均可以得到他（她）中意的舞伴。

这类 NP 问题在经历了很长时间以后，也无人能够找出时间复杂度为 $O(n^k)$ 的可以快速解决这类问题的算法。

解决此类问题有以下几种典型解法：

1．穷举法。以推销员问题为例，要解决该问题最简单的方法应该是穷举法，只要列出所有可能的路径，总是能够找到一条最短路径，但是这个寻找过程却远非想象中的简单。计算速度与信号穿过硅晶体管的时间有关，这个时间大致可由以下公式给出：$t=d/s, d$ 为信号传过的距离，s 为信号穿越的速度，以当前的技术，d 已经可以达到$0.18\mu m$，$s=10^{11}\mu m/\sec$，$t\approx 2\times 10^{-12}$秒，这个时间只是执行一次原操作的时间，计算机要完成一次排列，需要经过很多次原操作，不妨假设计算机可以在一个原操作时间内完成一次排列，推销员要在 n 个城市中完成推销任务，所有可能路径为 $n!$，如果 $n=23$，计算机要在 23 个城市中找到一条最短的路经，需要 1640 年的时间，如果要将计算速度提高至每10^{-15}秒完成一次原操作，以这个速度来找出 23 个城市的最短路线，需要 299 天，计算速度提高了，计算时间的确大幅度减少了，但是，若城市的数目增加 1 个，找出最短路线的时间就会骤然升至 20 年。看来，计算速度的提高对于本问题来说几乎于事无补，因为只要令城市的数目再增加 1，那由于计算速度提高而缩短的计算时间将瞬间被成指数级增长的计算量抵消。所以，用穷举法求解 NP 问题，当 n 很大时是行不通的。

2．并行计算。采用并行计算，同时使用多台计算机处理一个问题，确实可以在一定程度上提高计算速度。如果 $n=61$，同时启用2^{200}个处理器，在计算速度为10^{-15}秒的情况下，也要 10 年的时间才能计算出所有排列。如果 $n=62$，在同样的情况下，则需要 624 年的实践。如果 n 无限大时，所有并行计算对该问题也显得无能为力，并且处理器的数目不可能无限增加。

3．贪婪算法：当采用提高计算速度的方法求解已经行不通时，就必须转而寻求算法上的解决方法，只要能够找到一种方法，可以在不断完善路径的同时不断逼近最优解，那么就能找到一个时间复杂度为 n^2 或 n 的算法，求出与最佳解答大致相近的解答。有一种算法，其思想就是在不断地寻找目前推销员所在的城市距离最近的城市，寻找的顺序就是所求路径，这种算法被形象地称为贪婪算法。但是贪婪算法的结果并不一定是最优的，早在 1977 年，Rosenkrantz 等人已经证明这不是很理想的解法。他们证明了如果城市间的距离满足三角不等式，则用贪婪算法所走的总路径为 L_0 与最短路径 L_1 之间的关系为：$L_1\le\frac{1}{2}(\log_2 n+1)L_0$，如果各城之间的距离不满足三角不等式，Sahni 和 Gonzalez 在 1976 年证明了若 $P\neq NP$，则不可能存在一个有限的 m 和一个 $O(n^k)$ 计算量的走法，可以使总路径 L_1 在 m 为任何值的时候满足 $L_1\le mL_0$，也就是说，m 一定是无穷大，或者所有 $O(n^k)$ 计算量的走法都不理想。

很多 NP 问题的近似解受问题本身的影响而各不相同，就是把一个 NP 问题

分解成很多部分，每一部分求解出最优的解答，也没有一个解法使之可以通用于所有 NP 问题。随着未来人们对世界的认识继续向前发展，对于 NP 问题的解法也在不断发展，从七十年代开始对 NP 问题的研究主要是以许多不同的计算模型来分析难解问题的本质。这些新的计算模型包括了平行计算模型、概率计算模型、布尔线路、判断树、平均复杂性、交互证明系统以及程式长度复杂性等等。对这些新的计算模型的研究一方面使我们对难解问题有了更深一层的认识，另一方面也产生了一些预想不到的应用。最显著的一个例子就是计算密码学的革命性突破，即基于 NP 问题的公钥密码体系；另一个有名的例子是线性规划的多项式时间解的发现。

到了八十年代中，对 NP 完全问题的研究有了纵向的突破，在许多表面看来并不相关的计算模型之间发现了新的关系。这些关系不但解决了几个令人困扰多年的未解问题，同时也刺激了其它相关领域的发展。其中之一是对线路复杂性的研究发现了一些问题在某种有限制的线路模型中必有指数下界。这些结果使用了组合数学与概率方法等新的数学工具，并且解决了一个有名的有关多项式分层的未解问题。

现今对于 NP-Hard 问题的解决方法主要是使用遗传算法（Genetic Algorithm，GA），模拟退火（Simulated Annealing，SA），禁忌算法（Tabu Search，TS）等算法。

2.4 贪婪算法理论和混合算法理论

贪婪法是一种典型的邻域搜索技术，算法从一个随机选取的可能解出发，对该解的邻域进行搜索，如果找到更好的解，则将新解作为当前解继续以上过程，直到所有邻域操作都不能改变解的质量。贪婪法总是做出在当前看来最好的选择，具有很强的局部寻优能力，但并不能对所有的问题都能得到全局最优解。

例如平时购物找钱时，为使找回零钱的硬币数最少，不考虑找零钱的所有各种可能性而是从最大面值的币种开始，按递减的顺序考虑各币种，先尽量用大面值的币种，当不足大面值币种的金额时才去考虑下一种较小面值的币种，这就是在使用贪婪法。这种方法在这里总是最优，是因为银行对其发行的硬币种类和硬币面值的巧妙安排。如只有面值分别为 1、5 和 11 单位的硬币，而希望找回总额为 15 单位的硬币。按贪婪算法，应找 1 个 11 单位面值的硬币和 4 个 1 单位面值的硬币，共找回 5 个硬币。但最优的解应是 3 个 5 单位面值的硬币。

一个典型的用贪婪法解决的问题是装箱问题，设有编号为 0、1、…、$n-1$ 的 n 种物品，体积分别为 v_0、v_1、…v_{n-1}。将这 n 种物品装到容量都为 V 的若干箱子里。

约定这 n 种物品的体制均不会超过 V，即对于 $0 \le i \le n$，有 $0 \le v_i \le V$。不同的装箱方案所需要的箱子数目可能不同。装箱问题要求使装尽这 n 种物品的箱子数要尽量少。

若考察将 n 中物品的集合划分为 n 个或者小于 n 个物品的所有子集，最优解可以找到。但所有可能划分的总数太大。对适当大的 n，找出所有可能的划分要花费的时间是无法承受的。为此，对装箱问题采用非常简单的近似算法，即贪婪法。该算法一次将物品放到它第一个能放进去的箱子里，该算法虽然不能保证找到最优解，但还是能找到非常好的解。不失一般性，设 n 件物品的体积是从大到小排好序的，即有 $v_0 \ge v_1 \ge \cdots \ge v_{n-1}$。如不满足上述要求，只要先对这 n 件物品按它们的体积从大到小排序，然后按排序结果对物品重新编号即可。

装箱算法实现的方法可以描述如下：

```
{
    输入箱子的容积;
    输入物品种数 n;
    按体积从大到小顺序，输入各物品的体积;
    预置已用箱子链为空;
    预置已用箱子计数器 box_count 为 0;
     for (i=0;i<n;i++)
     {   从已用的第一只箱子开始顺序寻找能放入物品 i 的箱子 j;
         if （已用箱子都不能再放物品 i）
         {   另用一个箱子 j，并将物品 i 放入该箱子;
             box_count++;
         }
         else
            将物品 i 放入箱子 j;
     }
}
```

上述算法能求出需要的箱子数 box_count，并能求出各箱子所装物品。下面的例子说明该算法不一定能找到最优解，设有 6 种物品，它们的体积分别为 60、45、35、20、20 和 20 单位体积，箱子的容积为 100 个单位体积。按上述算法计算，需三只箱子，各箱子所装物品分别为：第一只箱子装物品 1、3；第二只箱子装物品 2、4、5；第三只箱子装物品 6。而最优解为两只箱子，分别装物品 1、4、5 和 2、3、6。

在实际的计算中，算法的混合策略是弥补各种算法自身缺陷的一种有效方

法，其宗旨是将不同算法在优化机制、进程、搜索行为、操作上进行有机结合，使其相互取长补短，以产生更好的优化效率。

例如贪婪-遗传算法，遗传算法的内在并行性使得算法可在较大的解空间内迅速寻优，能以较大概率求得全局最优解，具有很好的全局寻优性能。但是遗传算法不能很好地利用局部信息，常常造成搜索速度慢、早熟等现象。因此把传统的邻域局部搜索积疏于遗传算法相结合，可增强算法的局部搜索能力，提高算法的收敛速度和优化性能。采用遗传算法作为主框架，在进化过程中，对每代经过遗传操作后的新种群实施贪婪搜索。但贪婪操作本身需要耗费一定的时间，如果对每代种群中所有个体进行贪婪操作将花费大量的时间，有可能使算法的收敛速度减慢。因此，需要对贪婪操作进行适当的控制，比如设定贪婪操作概率，对少数较优个体实施贪婪操作等等，以保证算法的整体性得到提升。

又如模拟退火（SA）-遗传算法（GA），模拟退火算法使用串行优化算法，整个优化过程中仅有一个当前解，并且优化操作对退温过程有很强的依赖性，要使算法收敛需要较长的优化时间，而遗传算法采用种群并行搜索，将 SA 嵌入 GA 框架中，能够使 SA 成为并行 SA 算法，提高 SA 的优化能力，同时 SA 作为一种自适应变概率的变异操作，能够增强 GA 的进化能力。另外，SA 和 GA 对算法参数设置具有很强的依赖性，参数选择不当往往严重影响算法的优化性能，将两种算法相结合使得算法各方面的搜索能力都有所提高，因此对算法的参数选择不必过分严格。

还有禁忌搜索-遗传算法，将遗传算法作为主体框架，把禁忌搜索作为算子嵌入到遗传算法中，每代种群在实施遗传操作后，对其每个个体进行禁忌搜索。这种混合策略可以融合遗传算法并行的大范围搜索能力和禁忌搜索较好的局部搜索能力，并且禁忌搜索较好的“爬山”能力可以弥补遗传算法“爬山”能力较弱的缺陷，能够改善算法的全局收敛性能，避免陷入局部最优。此外，还有模拟退火-遗传算法等等。

本文将使用一种启发式算法与贪婪法相结合的算法，利用启发式算法来确定一系列集装箱存放空间请求的优先序列，以能够求出最优结果。然后用贪婪法等来计算出所用空间的大小。在这个过程中，一个高质量的优先序列是十分关键的，对于后面计算出空间利用大小结果的最优性具有决定作用。

2.5 常用的几种算法

2.5.1 模拟退火

模拟退火算法是一种试探性的邻域搜索算法，它根据对固体退火过程的模拟而得名。固体退火整个过程是这样的：将固体加温至充分高，再让其慢慢冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。用固体退火模拟组合优化问题，是将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复"产生新解→计算目标函数差→接受或舍弃"的迭代，并逐步衰减 t 值，算法终止时的当前解即为所求得近似最优解。关键是可以避免像爬山算法那样陷入局部最优。

模拟退火算法与初始值无关，算法求得的解与初始解状态(算法迭代的起点)无关；其具有渐近收敛性，已在理论上被证明是一种以概率 1 收敛于全局最优解的全局优化算法；并且模拟退火算法也具有并行性。

采用下面的模拟退火算法来解决集装箱堆场空间分配问题：

1）选择一个初始温度 T_0 和一个随机的初始状态 θ_0，令 $T=T_0$，定义目标函数为 $En()$，冷却进度表为 σ。

2）在当前状态 θ 的邻近范围内，提出一个新的参数空间状态值 θ'，令 $\theta'=\theta+\phi$，ϕ 为一些随机向量。

3）令 $\delta=En(\theta')-En(\theta)$。当满足下面条件时，该变为 θ'

$$\alpha(\theta,\theta')=\begin{cases}1, & if\ \delta<0\\ \exp(-\dfrac{\delta}{T}), & 其他\end{cases}$$

4）重复 2 和 3 步骤 K 次，直到认为达到平衡。

5）降低温度，令 T=T×σ，重复步骤 2-4 直到达到制定的标准。

由于 T 的对数衰减，我们令 T_0=1000，能量方程定义为堆场所需要的大小，概率 exp(-δ/T) 为 Boltzmann 常数。循环次数 K 根据输入数据大小来定。

2.5.2SWO 优化算法

SWO 优化算法（Squeaky Wheel Optimization）是一种求得近似最优解的方法。其核心是构造、分析、确定优先级，如图 2-2 所示。先通过贪婪法，通过一个已知的优先序列下构造出一个初始结果，根据得到的结果对优先序列进行分

析，找出问题并重新进行序列排序，再根据新序列构造新的结果，不断循环直到求得近似最优值。

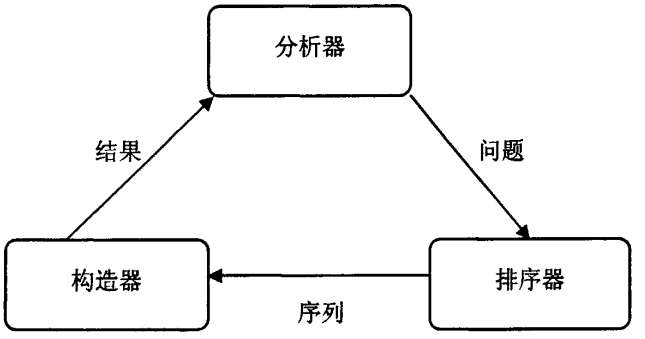


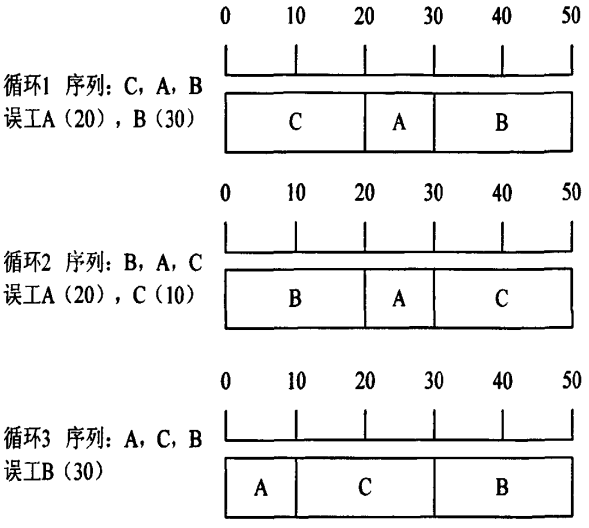
图 2-2 SWO 算法结构

图 2-3 给出了一个简单的例子来说明 SWO 算法的执行过程。假设我们有一个生产线，有三个任务在计划表中，分别是任务 A，B，C。在一个时段只能有一个任务在生产线上进行生产。开始时间为$t=0$，每个任务的持续时间和截止时间都已在图 2-3 中给出，目标函数时最小化误工时间。当只有一个任务产生误工时我们就任务达到了最优值。

假设初始的优先级序列为$\langle C,A,B\rangle$，通过构造器生成结果，结果有两个误工的任务，分别为任务 B 和 C。在分析过程中，分析器会找出每个任务的误工时间。

在这个例子中，任务 A，B，C 在第一循环中的误工时间分别为 20，30 和 0。在下一个循环中，排序器必须根据前一个序列和分析器中得到的结果重新产生一个新的优先序列。可以根据每个任务的误工时间来制定一个排序的规则，这样产生的一个新的优先序列就为$\langle B,A,C\rangle$。在第二个循环后，任务 A 和 C 会产生误工，误工时间分别为 20 和 10。所以产生的新的优先序列为$\langle A,C.B\rangle$。

在第三次循环后产生的结果中只有一个任务 B 产生误工，误工时间为 30。根据这一点我们可以认为这个结果为最优值。通过这个简单的例子，我们可以看出在处理有关优先序列问题上，SWO 算法具有一定的优势。



任务	持续时间	截止时间
A	10	10
B	20	20
C	20	40

图 2-3 一个简单的 SWO 优化例子

用 SWO 算法来解决集装箱堆场空间利用问题，首先会根据一个初始的优先序列，使用贪婪法来得到一个结果，初始的序列可以通过随机得出。通过分析器进行分析，改进优先序列以便能改进目标函数值。产生的新序列将通过构造器生成新的结果。这种 C-A-P 的循环将会一直持续下去，直到达到规定的极限次数，或者得到一个可以接受的结果。也可以认为 SWO 算法把这一个问题分为了两个搜索过程，一个是找出最优优先序列，一个是根据优先序列得到结果，从中可以看出优先序列问题是集装箱堆场空间利用问题的关键。

2.5.3 禁忌搜索

禁忌搜索方法是由 Glover 所提出的一种算法，适用于处理各种最优化或排列组合的问题，其主要发展的概念是原自登山演算法（Hill-Climbing）的演算过程，以搜索邻近解中之最优解为基础，不断反复寻找最优解，并且改善登山演算法易于陷入局部最优解（Local optimal）的缺点，加入适度记忆（adaptive memory）得架构，利用禁忌名单（Tabu list），记录搜寻过的点，避免搜索仅限于部分区域中。流程构架参考图 2-4。

禁忌搜索方法的记忆策略分为短期记忆（short-term memory）与长期记忆（long-term memory），短期记忆主要是记录搜索过的点，以避免重复的轨迹，而长期记忆是透过短期记忆，以寻找合适的方向找寻最优解。

一般禁忌搜索方法的架构分为以下几项：起始解（initial solution）、移步（move）、禁忌名单（tabu list）、候选名单（candidate list）、破禁原则（aspiration

criterion)、停止条件（stopping criterion)。

（1）起始解（initial solution)：

起始解为开始搜索的点，也可说是禁忌搜索方法的输入值，一般的做法是采用随机产生，也有部分研究使用启发式演算法来求出初始值。

（2）移动（move)：

移动也就是产生邻近解（neighbor solution）的方式，一般常见的有交换（Swap)、插入（Insert)、加入/销去（Add/Drop）和增加/减少（Increase/Decrease）等方式。

（3）禁忌名单（tabu list)：

此为用来记录搜索过程中的轨迹，名单的大小有各种的研究，名单长虽然可记录的点比较多，但也相对会拖慢搜索的过程，一般采用 Glover 提出的魔术数字（magic number）7 作为禁忌名单大小。

（4）候选名单（candidate list)：

所谓的候选名单指的是，扣除禁忌名单与破禁原则后的邻近解，由此名单中选出该移步的点。

（5）破禁原则（aspiration criterion)：

此为解除禁忌名单的条件，在某些问题中，若达到这些条件，释放部分禁忌名单中的点，可改善求解的品质。

（6）停止条件（stopping criterion)：

指的就是演算法的循环数，一般分为两种：最大循环数（max iteration)、最大解无改善循环数（max iteration with no improvement)，前者指的是搜索的最多次数，后者指在此次数下，若解仍无改善，则停止。

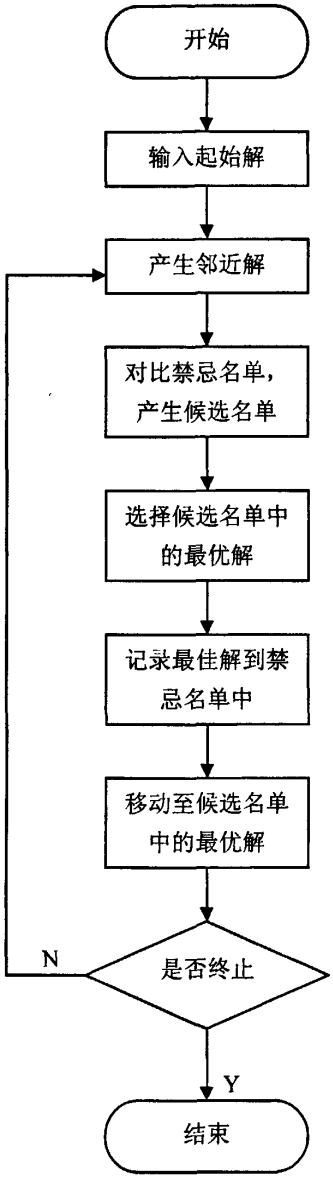


图 2-4 禁忌搜索方法流程图

结合上面提到的 SWO 优化方法，我们可以找出一种禁忌搜索算法与 SWO 优化算法相结合的方法。具体的流程如图 2-5。

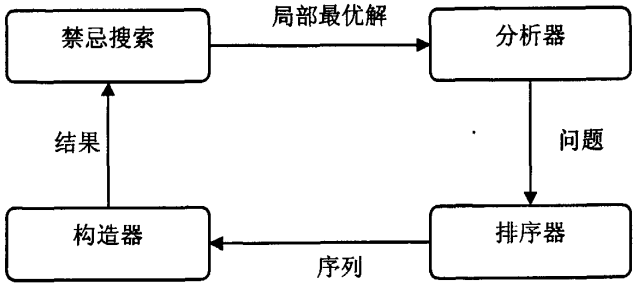


图 2-5 禁忌算法与 SWO 优化算法结合流程图

2.5.4 遗传算法

遗传算法（Genetic Algorithm，GA）由 Holland 于 1975 年提出，主要概念源于达尔文的进化论，利用一个群体（population）进行演化，群体中包含多个染色体，利用复制（reproduction）、交配（crossover）、突变（mutation）等等运算逐渐接近最优解，已被应用于最优化、工程、排序、电脑等多个领域上。

遗传算法以群体为架构，每个群体中包含多个染色体，而每个染色体包含多个基因，每个基因代表一种属性，架构可参考图 2-6，对应至最优化问题中则群体代表一组解，染色体代表一个解，基因代表解上的某一个变数，利用此方式可解最优化问题。

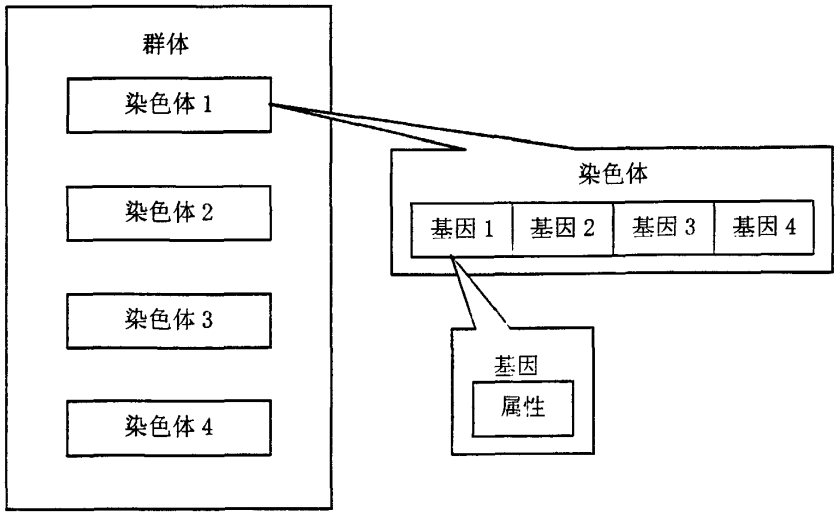


图 2-6 染色体架构图

遗传算法不同于一般的算法在于，它并非采用单点搜索，而是多点搜索，当解空间很大时，仅在起始解附近开始搜寻最佳解，效率较差。遗传算法的结构，主要包括下列几个部分：起始解、编码、适合度、复制、交配、突变、停止条件，其流程图见图 2-7。

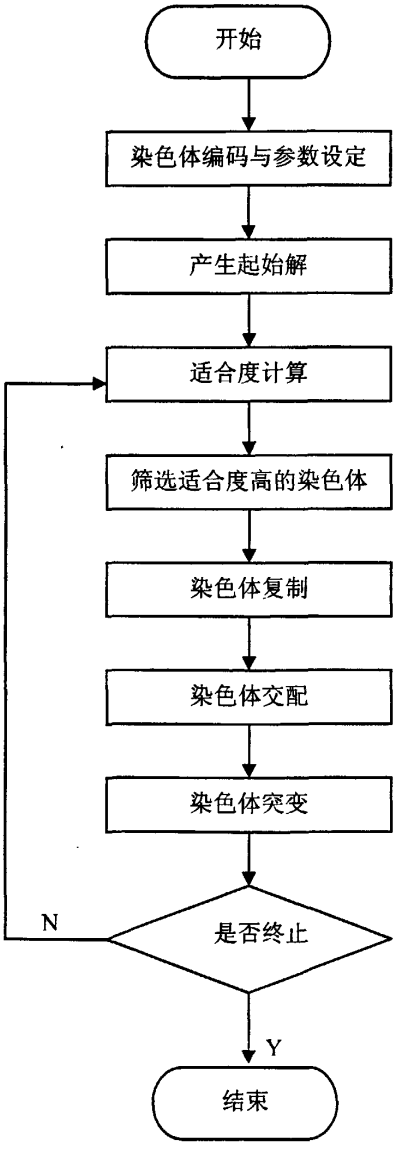


图 2-7 遗传算法流程图

（1）起始解：

产生初始解时，会先设定群体的大小，群体的大小应向求解的速度，大的群体虽然涵盖的范围较广，但是所消耗的时间也多，这是需要考虑的地方。

（2）编码：

染色体的编码，必须依照问题的解的类型来决定，一般常见的编码有二元数值（Binary）、实数（Real）、整数（Integer）、及符号（Symbol），必须依照问题特性来选择。

（3）适合度：

适合度是遗传算法中用来衡量染色体优劣的方式，而染色体的优劣则会应向其存活的概率，适合度越高者，存活几率越高，而适合度函数的设计，则需要依照问题特性决定。

（4）复制：

复制指的就是从群体中挑选适合度高的染色体保留至下一代，一般常见的方法有轮盘法（Roulette Wheel Selection），就是在一轮盘上，面积代表被选中的几率，因此越高的适合度有越大的面积，也表示越容易被选中至下一代。

（5）交配：

当完成复制阶段后，群体种的染色体则有机会进行交配，但并非每条染色体都会进行交配，而是会根据交配率（crossover rate），选择偶数条染色体进行交配，而交配的方式有单点交配、双点交配、均匀交配、算术型交配、及矩阵型交配法，我们简单介绍由 Vignaux 等提出的矩阵型交配法：

假设染色体 X 为一矩阵型编码染色体，而 X 展开后为

$$X=\left[\begin{array}{c|cccc} x_{11} & x_{12} & x_{13} & \Lambda & x_{1n} \\ x_{21} & x_{22} & & & M \\ x_{31} & & \mathrm{O} & & M \\ M & & & \mathrm{O} & M \\ x_{m1} & \Lambda & \Lambda & \Lambda & x_{mn} \end{array}\right] \qquad X_{mn}：\text{基因值}$$

步骤 1：染色体交配

若想交配的两个染色体分别为 $X1$ 和 $X2$，则在交配过程中会产生两矩阵 $y1$，$y2$

$$y1=\mathrm{int}(\frac{x1+x2}{2})$$

$$y2=(x1+x2)\bmod 2$$

步骤 2：产生新的染色体

将 $y2$ 随机拆成两个矩阵，并使得 $y2=y21+y22$，则新产生的染色体 $x1'$ 和 $x2'$ 为：

$x1' = y1 + y21$

$x2' = y2 + y22$

（6）突变：

若交配时仅有交配运算，则易陷入局部最优解，因此为了避免这种情形，加入了突变运算，以增加搜索的广度。演化过程中，依照突变率决定是否突变，通常突变率小于 0.1，而不同的编码采用不同的突变法，一般常见的突变有交换式（interchange）突变、取代式（displacement）突变法。交换式突变的做法是任选两个基因，然后对其进行交换；取代式突变法的做法则是任选一个基因，然后取一个合理数值进行取代。

（7）停止条件：

常见的停止条件有下列几种：

A、设定演化的世代数；

B、设定电脑运算的时间；

C、若经过了 x 世代，而最佳的适合改善不到 $y\%$；

2.6 几种算法的比较

在以往人们对于排序问题的求解过程中，遗传算法、模拟退火算法和禁忌搜索算法都是经常被使用的，比较它们各自的特点，如表 2-1 和表 2-2 所示。

表 2-1 启发式算法比较表

启发式算法	分类
遗传算法（genetic algorithm）	A/S/P
模拟退火算法（simulated annealing）	A/S/1
禁忌搜索算法（Tabu search）	M/N/1

表 2-2 名词解释表

代号	特征代号	特征说明
x	M	使用适度记忆（adaptive memory），记录过程中搜寻过的解。
	A	无记忆功能。
y	N	有系统化的方式搜寻邻近解。

	S	随机产生邻近解。
z	1	以单一解进行搜寻。
	P	以群体解进行搜寻。

算法特性说明：

遗传算法：在搜寻过程中，不记录过程中搜寻过的解，而产生邻近解的方式是以运算子随机产生，搜寻过程则是以群体为单位产生邻近解。

模拟退火：在搜寻过程中，不记录过程中搜寻过的解，而产生邻近解的方式是以运算子随机产生，不过在搜寻过程中是以单一个体为单位产生邻近解。

禁忌搜索：在搜寻过程中，会记录过程中搜寻过的解，其产生邻近解的方式由特殊的运算子产生，在搜寻过程中是以单一个体为单位产生邻近解。

2.7 本章小结

在本章中我们首先介绍了集装箱堆场的概念，接着说明了堆场的功能和容量。在基础理论方面，介绍了贪婪算法理论和混合算法理论，以及NP问题的基本原理。

然后重点介绍了几种经常用来解决集装箱堆场问题的算法，对于后续章节中要用到的理论基础进行了详细的介绍。

第三章 集装箱堆场空间分配问题的模型建立

在第二章中主要介绍了堆场的概念功能，以及一些算法理论基础。在这一章中，将会对空间分配问题的概念进行介绍，并且建立集装箱堆场空间分配的基本数学模型，以及介绍一种“下落”（DROP）递归空间分配方法。

3.1 空间分配问题

3.1.1 空间分配问题概念

空间分配问题也可以称作空间布局问题，是指企业或个人根据自己的经营目标和目的，在已确定的空间场所内，按照一定的规则，对空间作适当的分配和有效的组合，以便获得最大的经济效益。

好的空间分配方法在存储问题上可以提高空间利用率，在选址布局问题上可以提高运输和生产效率。如在存储方面，好的空间分配方案既有利于节约空间也可便于存取；又如工厂布局在实施规划设计中占有重要的地位，它的好坏直接影响整个系统的物流、信息流、生产能力、生产效率、生产成本以及生产安全，优劣不同的工厂布局在施工费用上可能相差无几，但对于生产运营的影响会有很大不同。

3.1.2 空间分配问题类型

空间分配问题按时间分类可以分为静态空间分配和动态空间分配：

（1）静态空间分配：静态空间分布与时间无关，它的已知条件如空间大小和分配条件等都是确定的，只需要做出一次性的分配。属于静态空间分配的例子有物流配送中心的选取和工厂布局等。这类问题的分配方案都无需考虑时间因素。

（2）动态空间分配：与静态空间分配相反，动态空间分配中时间是分配方案中考虑的一个重要因素，在一个时间段内，空间需求是不断变化的，这就需要随着时间的变化，相应的分配方案也要变化。如仓储空间利用，码头堆场空间利用等。如果选择的时间段比较短，那么就意味着较少的计算量和较差的预测性；相反，选择的时间段比较长，会使计算量增大，预测的不确定性也将会增大，如

图 3-1。

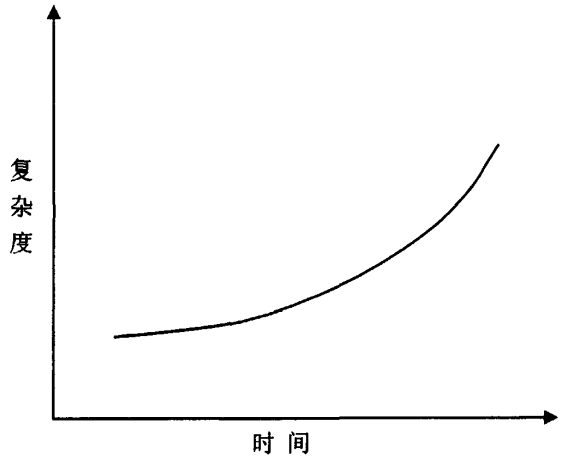


图 3-1 时间与复杂度关系图

3.1.3 空间分配问题的应用

空间分配问题在现实中有很多实际应用，具体可以分为运输装载问题，仓储问题，工厂设施布置设计等。

（1）运输装载问题：集装箱装载问题是企业在产品运输中，尤其是国际运输中必须考虑的一个问题。生产出来的产品被放入某种规格的包装容器内，由于产品本身的特性，存放产品的包装容器在集装箱内的存放要求不一。在这个前提下，包装容器要紧密地放入集装箱内,以提高集装箱的空间利用率，降低运输成本。同时，由于人力、物力及产品本身的要求，可能包装容器只能一次性地装入，而不能反复装卸,否则会造成产品的磨损，这就极大地限制了集装箱利用率的提高。目前将包装容器装入集装箱的工作大部分依靠手工操作来进行。调度员在现场凭借经验估计被装入的几种包装容器的数量及所需的集装箱数，并开出装箱单据,然后装箱工人依照单据将包装产品的容器放入。这种操作程序存在两方面的问题：一是虽然调度员经验丰富，但仅仅凭借直观估计很难保证高的装箱率；二是装箱工人装箱过程中的随意性，同样可能造成集装箱的低利用率。显然，这种手工装箱的方式在货物运输环节中增加了企业的成本。利用计算机模拟技术求出一个合理的集装箱布局及装载方案，以提高集装箱空间利用率和装载效率就属于一个典型的空间分配问题。

（2）仓储问题：许多商品由于生产或需求原因，其库存水平呈现出季节性

波动，如粮食作物、发电用煤、空调等。当企业所生产的产品或所使用的原材料的库存具有上述特征，即属于季节性库存时，该企业对于仓储空间的需求也将具有季节性。在季节性库存的条件下，企业的仓储空间需求会随着月份变化出现仓储空间需求的高潮和低潮，如果不好好的利用空间资源，机会造成资源的浪费增加企业成本。

（3）工厂设施布置设计：考虑工厂各个组成部分，包括生产车间、辅助生产车间、仓库等等，同时要考虑物料的流向和流程、厂内外运输方式。车间布置设计要解决各生产工部、工段、服务辅助部门、储存设施等作业单位，同时也要解决物料搬运的流程及方式。

3.2 集装箱堆场空间分配问题研究

3.2.1 集装箱堆场空间分配问题

船舶到港后，为了保证船舶能够快速周转，港口能保持高水平的作业物流，控制作业成本，除了为其配置相应的泊位、桥吊、水平运输设备资源之外，更为关键的是要优化堆场空间资源的配置。

堆场一般是由集装箱箱区组成的。每个箱区有大量的集装箱，箱区的列数和高度由使用的龙门吊的型号、跨度决定。一般的轮胎式龙门吊工作箱区为六列集装箱宽，四、五层集装箱高，每一列有大约 20 个集装箱。图 3-2 中给出了一个典型的堆场作业图，集装箱从船上卸载后或者准备装船前都要在堆场区堆存。

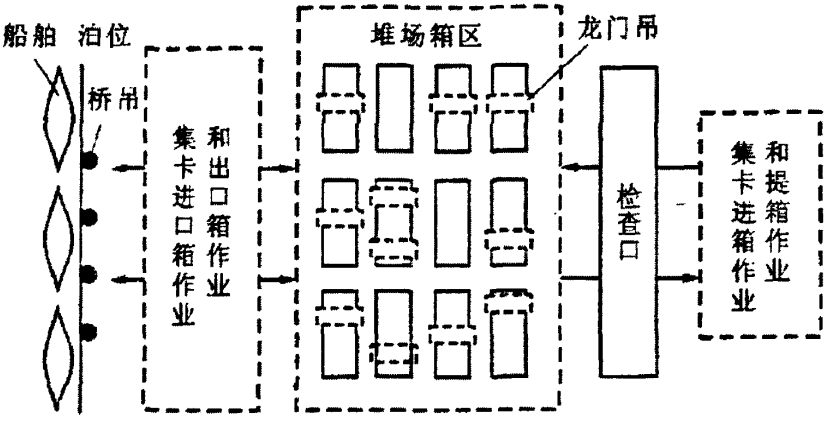


图 3-2 堆场作业图

3.2.2 集装箱堆场分配的动态特性

集装箱码头一年365天连续不断地运作，港口中的作业箱流是由到达的船舶引起的，集装箱到港后，会运送到堆场进行暂时的保存，并等待下一步的运作。集装箱在堆场存放的时间长短也是由客户要求所确定，所以可以看出整个集装箱堆场的空间分配问题都是和时间因素紧密相关，属于一个动态空间分配问题。

船舶到达港口的时间一般都是固定的，集装箱量也是可以预知的，所以这就给了提前制定分配计划的可能性，也是解决这一动态空间分配问题的基础。

3.2.3 对集装箱堆场空间分配的其他影响因素

在实际的堆场操作中，还存在很多其他的影响因素。如由于天气原因造成船舶不能及时进出港，使集装箱积压港口，或者客户的货物没有及时的从堆场中取走，都有可能导致堆场空间分配出现问题。此外还要考虑堆场内的运输方面、进出口管理等方面，增大了码头的不确定性，使问题复杂程度增大。

在本论文中，去掉了天气等复杂原因，只考虑了在动态时间情况下的堆场空间利用问题。

3.3 集装箱堆场空间分配问题模型的建立

3.3.1 问题描述

集装箱堆场在一个时间段内会收到很多集装箱存储请求，为了充分利用空间同时又满足客户需求，需要我们建立相关的数学模型对其进行研究。

集装箱堆场问题已经被证明是NP-Hard问题[22]，然而此类问题可以转化为一个二步决策问题。在第一步中，对于有限个的空间存放请求，对其进行排序，优先级越高的请求会越早分配；在第二步中，应用一个多项式时间贪婪法来对于给定的序列建立分配方案。

由于此问题要涉及到堆场大小和请求时间段的问题，所以在建立数学模型之前要做出下列一些假设和简化：

（1）每一个存放请求都有一个单独的时间段和对应在这个时间段中的一系列堆场空间占用大小。

（2）集装箱存放请求分配到的任何堆场空间直到请求时间结束后才能释放．请求占用的空间大小只能不变或者增大。

（3）假设E为一个无限大的集装箱堆场，R为n个空间请求的集合。

3.3.2 模型参数和决策变量

根据以上的问题说明和假设条件，相关的模型参数说明如下：

R_i：第i个请求，$R_i \in R$，$i=1,2,...,n$；

TS_i：第i个请求的开始时间，$i=1,2,...,n$；

TE_i：第i个请求的结束时间，$i=1,2,...,n$；

T_i：请求R_i的时间跨度，$T_i=\{TS_i, TS_i+1, \cdots, TE_i\}$；

$T_{\max}$：需要考虑的最大时间范围长度，$T_{\max}=\max_{i\in R_i}\{TE_i\}$；

$Y_{i,t}$：第i个请求，t时间的空间请求，$i=1,2,...,n$，$t\in T_i$；

$L_{i,t}$：第i个请求，t时间请求的空间大小，$i=1,2,...,n$，$t\in T_i$；

F：集装箱堆场分配图；

$F(Y_{i,t})$：分配图F对每一个$Y_{i,t}$分派的开始配置，$F(Y_{i,t})\in E, i\in R, t\in T_i$；

为了请求分配空间，一个分配图 F 需要对每一个$Y_{i,j}$选择分派一个开始位置$F(Y_{i,j})$。因为一个请求分配到的空间要一直占用到请求结束，所以对于每一个请求$R_i \in R, t\in T_i, t>TS_i$，分配图 F 必须满足这个条件：

$$F(Y_{i,t})\le F(Y_{i,t-1}) \quad 和 \quad F(Y_{i,t})+L_{i,t}\ge F(Y_{i,t-1})+L_{i,t-1} \quad ;$$

同一个空间不能够同时分配给不同的请求，满足下面约束。对于两个不同的请求：

$$F(Y_{i,t})+L_{i,t}\le F(Y_{j,t}) \quad 或 \quad F(Y_{j,t})+L_{j,t}\le F(Y_{i,t}) \quad ;$$

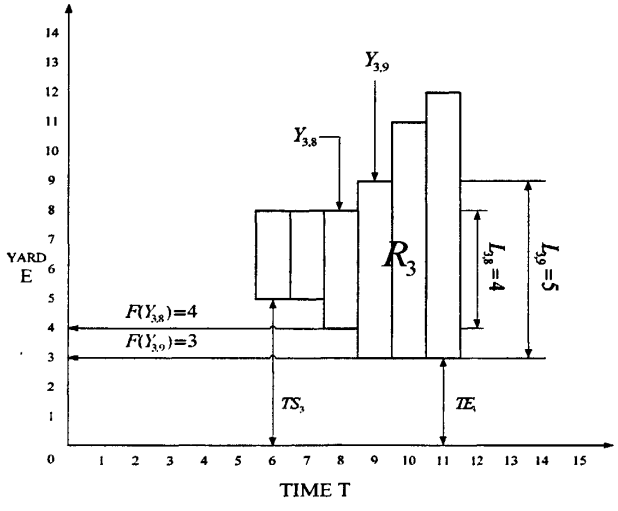


图 3-3 一个有效的空间请求

例如，图 3-3 中显示了一个有效空间请求的布局，表示了请求在对应时间内的空间大小要求，横坐标表示时间，纵坐标表示堆场空间。请求为R_3，时间跨

度是从 $TS_3=6$ 到 $TE_3=11$。它的空间请求 $Y_{3,8}$ 和 $Y_{3,9}$ 的开始位置分别为 $F(Y_{3,8})=4$ 和 $F(Y_{3,9})=3$。对于请求 R_3 每个时间段的开始位置分别为（5，5，4，3，3，3），满足条件约束，如 $F(Y_{3,8})\geq F(Y_{3,9})$ 和 $F(Y_{3,8})+L_{3,8}\leq F(Y_{3,9})+L_{3,9}$。

3.3.3 模型建立

根据上述对模型背景的描述以及相关参数和假设的说明，可以得到以下的数学模型：

$$\min_{F}\max_{i\in R,t\in T_i}\left(F\left(Y_{i,t}\right)+L_{i,t}\right) \tag{3-1}$$

满足：

$$F\left(Y_{i,t}\right)\leq F\left(Y_{i,t-1}\right) \qquad R_i\in R,t\in T_i,t>TS_i \tag{3-2}$$

$$F\left(Y_{i,t}\right)+L_{i,t}\geq F\left(Y_{i,t-1}\right)+L_{i,t-1} \qquad R_i\in R,t\in T_i,t>TS_i \tag{3-3}$$

$$F\left(Y_{i,t}\right)+L_{i,t}\leq F\left(Y_{j,t}\right)$$

$$\text{OR}\quad F\left(Y_{j,t}\right)+L_{j,t}\leq F\left(Y_{i,t}\right) \qquad R_i,R_j\in R,t\in T_i\cap T_j \tag{3-4}$$

这里目标函数（3-1）保证了占用空间的最小化，而约束中的条件保证了使每个请求为有效请求。约束（3-2）保证了每个请求要求的空间大小在其时间段内是不会减少的，只能增加或者不变；约束（3-3）保证在每个请求的时间段内，分配的空间要占用到整个请求的结束；约束（3-4）保证了不同请求在同一时间占用的空间不会重叠。

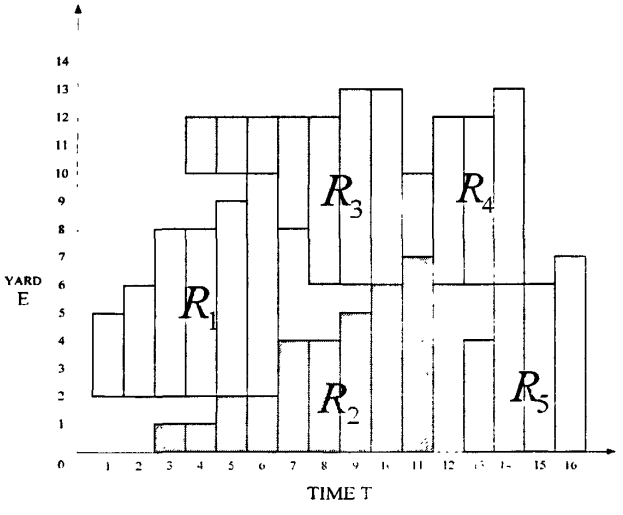


图 3-4 一个优化后的堆场空间分配图

在图 3-4 中给出的是一个有 5 个有效空间请求进行优化后的配置图，从 R_1 到

R_5，这 5 个请求的空间大小在其时间段内不能减少，只能不变或者增加，分配到的空间占用到整个请求的结束，而且相互之间的空间不互相交叠。满足约束（3-2）、（3-3）、（3-4），都属于有效的空间请求，根据目标函数（3-1）可以得出空间需求 $F_{3,10}+L_{3,10}=13$。

3.4 集装箱堆场存放请求空间分配的一种递归算法

在上节的问题描述中，空间请求分配问题被描述为一个二步决策问题。在有了一系列集装箱堆场空间分配请求优先序列后，为了能够得到优化后集装箱分配图，将按照此优先序列顺序，按优先级高到低进行分配。该分配过程使用一种递归的算法让空间请求分配到合适的位置，这种递归算法是通过贪婪法来寻求最优近似值，同时要求结果满足上一节的约束（3-2）、（3-3）和（3-4），是一个有效空间请求。因为此方法主要通过递归“下落”每个空间请求，所以这里把这种方法称为“下落“（DROP）递归算法，下面将给出 DROP 递归算法的执行过程。

算法：“下落”（DROP）递归算法

$DROP(R_i,t,h)$（h 是在 TS_i 到 t 时间段内 R_i 的每个空间请求能“下落的”最小值，t 初始值为 TE_i）

1）H=整个空间请求 R_i 能够“下落”的最低位置，t'=最低位置时的时间段；

2）如果 $H<1$；

3）$H=1$；

4）循环让整个空间请求“下落”到最低位置 H；

5）$DROP(R_i,t'-1,h)$；

通过“下落”（DROP）递归算法将每个时间段内的空间请求降到合适的起始位置。下面将通过一个例子来对这个算法进行说明。

如图 3-5 中所示为一个待分配的有效空间请求 R_i，时间段从 $TS_i=5$ 到 $TE_i=11$。

第一步：将请求中的每个时刻所占用空间长度尽可能的下调，空间请求的每个时刻能够下调的最低位置分别为（2，2，0，5，2，3，0），从中取最小值，所以 h=0。而整个空间请求 R_i 能够下调的最低位置为 H=3，t'=10，所以整个空间请求 R_i 下调 1 个空间单位，得到的结果如图 3-6。

第二步：循环进入下一个 DROP 方法，优化的时间段变为从 5 到 9，这个时间段内能够下调的最低位置为（2，2，0，5，2），取最小值 h=0,而 5 到 9 时间段内请求能够下调的最低位置为 H=5，所以整个空间请求可以下调 2 个空间单位。

第三步：时间段将从 5 到 7，重复刚才的过程，最终得到 R_i 空间分配图,如图 3-7。

需要注意的是，要时刻注意前文提到的对于有效空间请求约束。

以上通过“下落”（DROP）递归算法得到空间分配图的过程是在空间请求序列已知的前提下，而如何得到此优先序列就变成需要求解问题的关键，高质量的优先序列能够高效准确的得到目标值。不幸的是得到高质量的优先序列是一个NP-Hard 问题[22]，所以本文在第四章将使用一种关键请求局部邻近搜索方法来求解一系列空间请求下的最优序列问题。

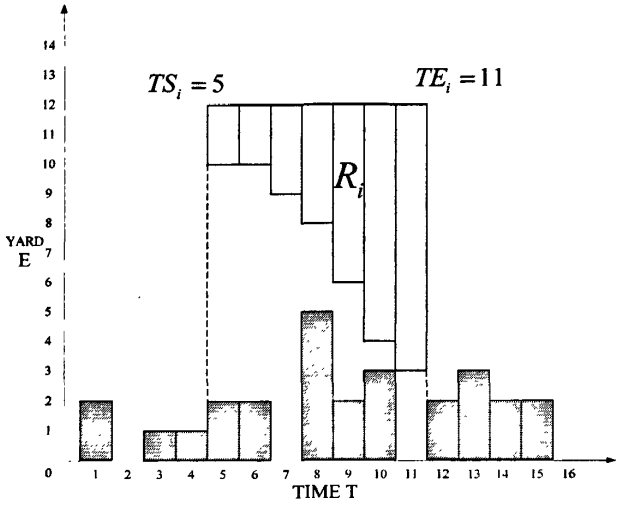


图 3-5 优化前的有效空间请求

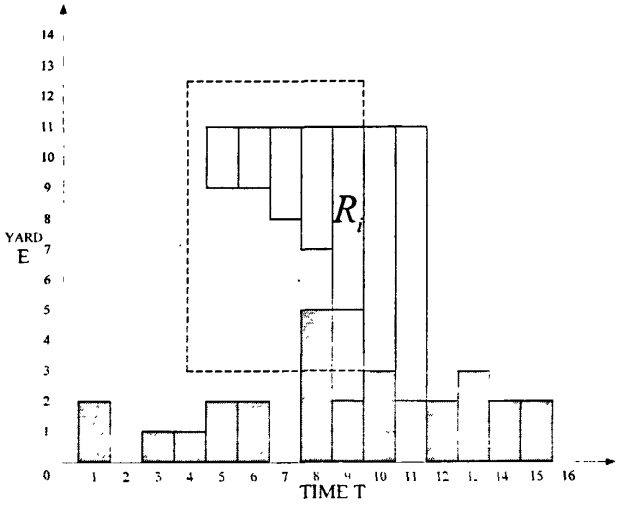


图 3-6 循环一次后的分配图

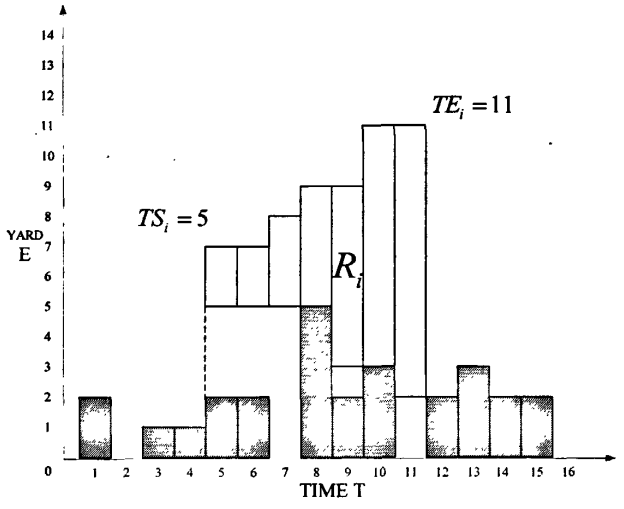


图 3-7 优化完成后分配图

3.5 仿真试验

这里假设有 6 个空间请求具体见表 3-1，表中 T 代表时间，L 代表所需空间（单位 TEU）。

图 3-1 空间请求数据表

请求	具体数据	
R_1	T	1，2，3，4，5，6
	L	2，2，4，4，6，6
R_2	T	5
	L	5
R_3	T	6，7，8，9，10，11，12
	L	2，2，2，4，6，7，7
R_4	T	7，8
	L	5，5，
R_5	T	9，10
	L	2，2
R_6	T	12，13，14，15
	L	5，5，6，7

对于这 6 个请求，如果随机产生优先序列，可以有 720 种可能。在这里我们先随机产生 20 组优先序列，然后用 DROP 的递归算法分别求空间占用最小值。具体试验结果如表 3-2：

表 3-2 仿真结果表

序列	最小值
$R_1, R_2, R_3, R_4, R_5, R_6$	13
$R_6, R_5, R_4, R_2, R_1, R_3$	13
$R_2, R_1, R_4, R_5, R_6, R_3$	13
$R_5, R_6, R_4, R_3, R_2, R_1$	13
$R_6, R_2, R_5, R_1, R_4, R_3$	13
$R_3, R_5, R_6, R_4, R_1, R_2$	13
$R_1, R_5, R_3, R_2, R_4, R_6$	13
$R_3, R_5, R_1, R_4, R_2, R_6$	13
$R_5, R_2, R_3, R_6, R_1, R_4$	13
$R_1, R_4, R_3, R_2, R_6, R_5$	13
$R_5, R_1, R_6, R_2, R_4, R_3$	12
$R_3, R_6, R_4, R_2, R_5, R_1$	13
$R_2, R_1, R_5, R_4, R_3, R_6$	18
$R_2, R_5, R_4, R_6, R_3, R_1$	13
$R_6, R_3, R_4, R_1, R_2, R_5$	18
$R_1, R_3, R_4, R_6, R_2, R_5$	13
$R_5, R_4, R_3, R_2, R_1, R_6$	14
$R_3, R_4, R_2, R_5, R_1, R_6$	13
$R_6, R_3, R_2, R_4, R_1, R_5$	13
$R_2, R_1, R_3, R_4, R_5, R_6$	18

从表中可以发现，不同的优先序列，可以产生不同的最小值。在表 3-2 中，可以求出这 6 个请求空间占用最小值为 12，其优先序列为 $R_5, R_1, R_6, R_2, R_4, R_3$。图 3-8、3-9、3-10 为按照此优先序列进行空间分配，前 5 个请求依次落入堆场 E 中，最后的请求 R_3 的分配过程如下：

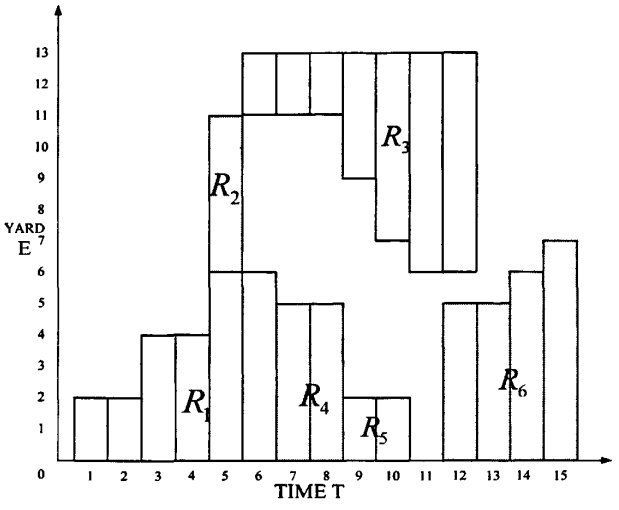


图 3-8 R_3 分配图（1）

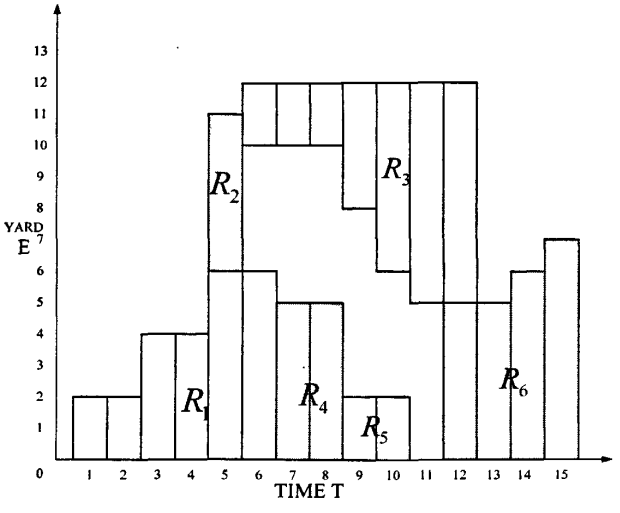


图 3-9 R_3 分配图（2）

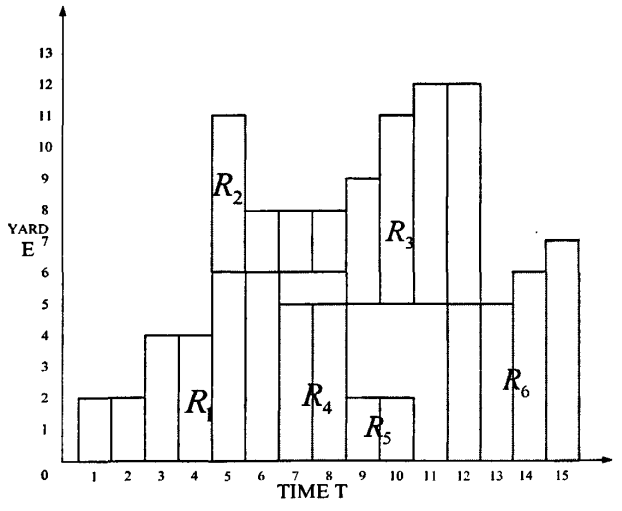


图 3-10 分配图（3）

通过对不同序列得到的最小值结果进行比较，可以得出结论，优先序列的优劣决定了分配后的空间利用最小值是否为最优结果。

3.6 本章小结

本章讨论了空间分配问题的概念、类型和应用，并在该问题研究框架下，分析了集装箱堆场空间分配的特性和影响因素。通过简化问题背景，建立了集装箱堆场空间分配问题的约束优化模型。为了求解该问题模型，把集装箱堆场空间分配问题转化为一个二步决策问题，第一步先求出空间请求的优先序列，第二步则是用一种 DROP 递归算法来最小化空间占用。而这一问题的关键就是求得最优序列，在下一章中将会进一步介绍一种求最优分配序列的启发式算法。

第四章 堆场空间存放请求排序研究

在第三章中已经建立了堆场空间分配的基本数学模型及其分配方法，知道影响空间分配大小的关键因素是空间请求的优先级序列，所以在第四章中就重点研究对集装箱空间存放请求进行排序的问题。首先提出了一种基于时间排序的方法来产生初始序列，然后通过运用关键请求局部邻近搜索的方法来求出最优序列，并通过仿真试验来验证方法的效果。

4.1 空间请求排序的实现步骤

实现堆场空间占用最小化的关键是要有一个高质量的优先序列，这样在分配空间时才能保证使用 DROP 方法最小化空间占用。

在对 n 个已知集装箱空间存放请求进行排序时，关键局部邻近搜索方法会从艺个初始序列 σ_0 开始，这个初始序列可以由随机产生也可以用一些规则来产生，下节中将介绍一种基于时间排序的方法来生成初始序列。

已知初始序列 σ_0，空间请求排序的实现将按照下面的步骤不断进行迭代改进，来求得最终的空间请求排序结果。首先，分析空间请求序列 σ 的构成，来选择一个包含 k 个关键请求的邻近范围集合 S。通过改变集合 S 中关键请求的优先级，来获得改善优先序列的机会。对于如何改变关键请求的优先级，本文将采用交换（Swap）和移动（Move）的方法，这将在下一节中阐述。这种通过改变关键请求优先级来改善序列的方法与 SWO 优化方法相似，但是并不像 SWO 优化算法那样提高关键请求的优先级，而是采用随机的方法，这种方法使结果更好的收敛于较好的优先序列。因为当一个优先序列已经具有了较高的优先级时，如果再提高其优先级，其结果也不会得到改善。

算法 1：空间请求排序的实现步骤

1）对 n 个空间请求排序生成初始序列 σ_0；

2）令 σ_0 赋予 σ，同时将 σ 赋予 σ_{opt}；

3）令 N 为需要循环的次数，$N = N_{\max}$；

4）当 $N > 0$ 时；

5）从当前序列 σ 中选出 k 个关键请求，组成集合 S；

6）改变关键请求的优先级以产生新的序列 σ'；

7）用局部邻近搜索方法改进序列σ'为σ''；

8）如果σ''优于σ_{opt}，

9）则将σ''赋予σ_{opt}，σ''赋予σ，$N_{\max}$赋予 N；

10）否则如果σ''不比σ差；

11）则将σ''赋予σ；

12）$N=N-1$；

13）结束循环；

为了更清楚的描述，可参考图 4-1。在这个实现步骤中有几个关键点，：初始序列σ_0的生成；关键请求的优先级改变；局部邻近搜索方法；这些都会在下面小节中一一介绍。而循环次数$N_{\max}$的大小要视具体的空间请求个数来确定，一般情况下，$N_{\max}$与空间请求个数一致。

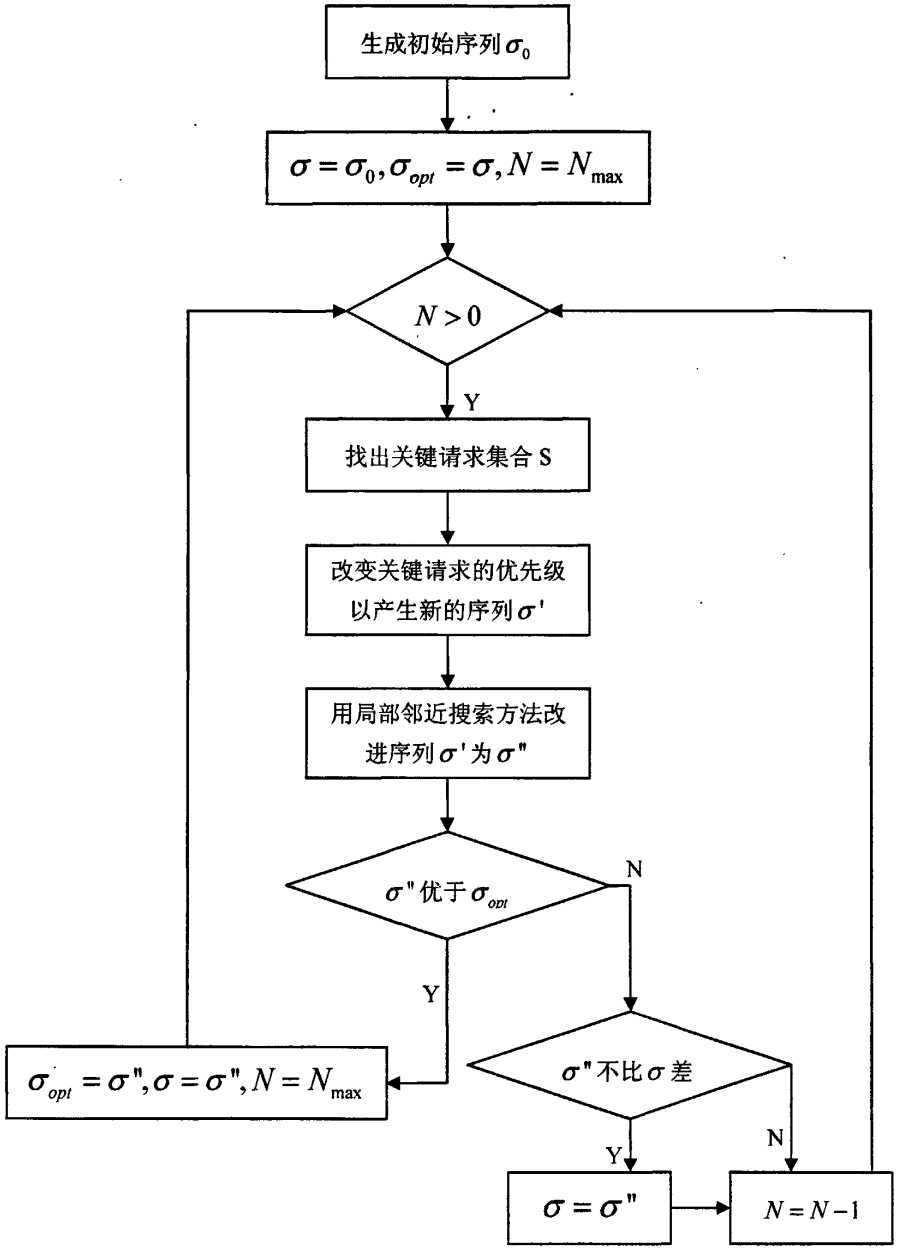


图 4-1 空间请求排序的实现流程图

4.2 初始序列的产生和排序的基本方法

4.2.1 基于时间排序的方法生成初始序列σ_0

从上节集装箱在堆场空间存放请求排序的实现步骤可以看出，生成初始序列σ_0是整个步骤的关键一步。本节将进一步分析研究初始序列产生和排序的方法。

对于已知的n个空间请求，首先需要生成一个初始序列σ_0，以便后续对其进行改进。当然，最简单的就是采用随机产生的方法来生成初始序列σ_0，但是经过试验证明随机产生的初始序列与最优序列相比有很大的差距，所以这里提出一种基于时间排序的改进方法来生成初始序列σ_0。

这个基于时间排序方法的基本思想是：

1. 已知n个空间请求，先从空间请求的集合中挑选出一个时间段最长的请求R_i，它的优先级别为最高，值为n；
2. 然后，在剩余的请求中寻找是否存在请求R_j，其$TS_j \geq TS_i$且$TE_j \leq TE_i$，若存在，则其优先级为$n-1$；
3. 若R_j分配完或者不存在则在时间段T，$0 \leq T \leq TS_i$寻找距离请求R_i时间最近的空间请求R_j'，分配下一优先级；
4. 接着寻找是否有请求R_k的时间段t，满足$TS_j' \leq t \leq TE_i$，若存在则继续分配优先级，否则向时间0的方向继续排序，直到时间段T，$0 \leq T \leq TS_i$内的请求都已分配完优先级，然后再在时间段T'，$T' \geq TE_i$内继续分配，直到所有请求都已被分配了优先级。

需要注意的是在当两个请求的时间段长度一样时，开始时间早的请求优先。如图4-2所示。

使用这种基于时间排序方法来产生初始序列可以将时间段最长的请求优先存放，减少其对最后结果的影响。而且这种排序可以将有时间交叠的请求优先级相连，这样在进行后续改进时有利于最优值的收敛。

如图4-3，有6个空间请求，他们分别是$R_1.R_2.R_3.R_4.R_5.R_6$，按照上面给出的方法产生初始序列σ_0。首先找出时间段最长的两个请求分别为$R_1.R_3$，但是由于R_1的开始时间靠前，所以把R_1的有优先级为6；因为R_2的时间段在R_1的时间段内，所以R_2的优先级为5；由于R_1左边的时间段内没有空间请求了，所以向右边寻找，分配R_4的优先级为4，类推R_5和R_6的优先级分别为3和2；因为R_3的时间段在TS_1和TE_6之间，所以R_3的优先级为1。整个序列的顺列位$\sigma=(6,5,1,4,3,2)$，按照此序列进行空间分配，则得到的最小空间占用为12，符合我们前文中空间分配的基本数学模型算法得出的结果.所以此序列可为最优序

列。

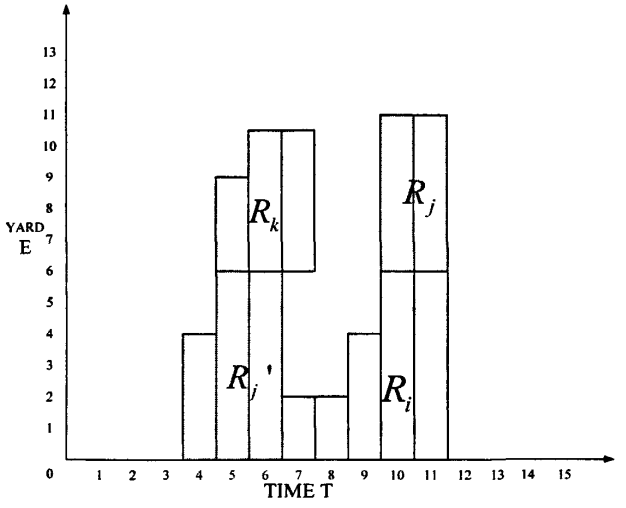


图 4-2 初始序列产生示意图

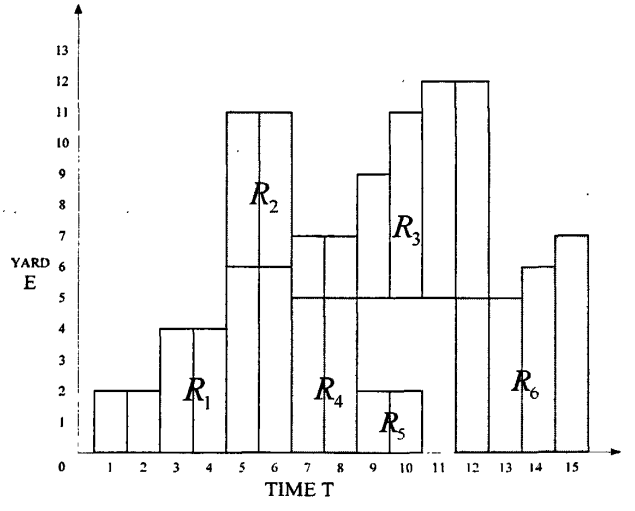


图 4-3　6 个空间请求分配图

4.2.2 局部邻近搜索方法

对于已经给出了优先序列σ'的 n 个空间请求，这里将采用一种局部邻近搜索方法来改进该优先序列的质量，以便能够得到高效准确的结果。

首先，定义以下两个优先序列改进的方法：

方法 1：交换 $Swap(\sigma', i, j)$

其中σ'表示优先序列，i和j分别对应着序列中不同的请求。对于请求R_i和R_j，交换$\sigma'(i)$和$\sigma'(j)$。例如 6 个空间请求的优先序列$\sigma'=(6,1,2,5,4,3)$，按照优先级从高到低排列为R_1,R_4,R_5,R_6,R_3,R_2，进行$Swap(\sigma',1,6)$，交换空间请求R_1和R_6的优先级，得到改进后的优先序列为$\sigma''=(3,1,2,5,4,6)$，空间请求的排列为R_6,R_4,R_5,R_1,R_3,R_2。

方法 2：移动$Move(\sigma',i,k)$

其中σ'表示优先序列，i表示对应的请求，而k表示要移动到的优先级。在序列中把R_i移动到$(n-k+1)$，使$\sigma'(i)$变成 k 和其他请求优先级相应变化。

同样，以前面的优先序列$\sigma'=(6,1,2,5,4,3)$为例，进行$Move(\sigma',3,6)$，将R_3的优先级变为 6，其余空间请求的优先级依次变化。因为R_3占据了优先级 6，因此原先优先级为 6 的请求R_1的优先级就要变为 5，也就是说在这 6 个请求中，以前移动前比R_3优先级高的请求的优先级都要下调 1，而比R_3优先级低的请求的优先级则不变。按照这一过程，R_1的优先级变为 5，R_4的优先级变为 4，R_5的优先级变为 3，R_6的优先级变为 2，而R_2的优先级移动前就比R_3低，所以仍然为 1。得到改进后的优先序列为$\sigma''=(5,1,6,4,3,2)$，空间请求排列为R_3,R_1,R_4,R_5,R_6,R_2。

基于以上这两种操作方法，可以采用穷举的方法来不断进行一个空间请求排序的交换和移动，从而达到改进序列的质量目标。但是显然在序列中随机的挑选一个或两个空间请求进行这两种操作需要比较长的计算时间，计算效率很低。因此，为简化计算量，这里提出采用一种局部邻近搜索的方法。

下面将进一步研究考虑在优先序列σ'内定义一个邻近的范围。

首先我们先要介绍空间请求的时间相交叠概念。对于每个空间请求R_i，$R_i\in R$。定义N_i为一个空间请求的集合，满足集合中的空间请求与R_i的时间跨度相交叠。例如一个请求R_1，它的时间跨度为$TS_1=2,TE_1=5$；另一个空间请求R_2，它的时间跨度为$TS_2=3,TE_2=6$。这样的两个空间请求就认为发生了时间相交叠。如图 4-2 中，请求R_2就与R_1的时间相交叠。

因此，在进行 Swap 和 Move 操作时，操作的对象将会在N_i的集合中选出，来实现操作$Swap(\sigma',i,j),j\in N_i$和$Move(\sigma',i,k),k=\sigma'(j),j\in N_i$。显然，这种满足空间请求的时间相交叠概念的局部邻近搜索范围要比一般的局部搜索要小很多。一般局部搜索的寻优为$O(n^2)$，而局部邻近搜索的寻优将能够减少到$O(\delta n)$，δ是对于$R_i\in R$的N_i最大范围。在多数情况下比n小。

下面给出描述局部邻近搜索过程的算法步骤：

算法 2：使用局部邻近搜索方法改进优先序列σ'

1）定义σ''为从优先序列σ'改进后的结果；

2）先将σ'赋予σ''；

3）如果不是最优序列，进行5）中的操作直到找到改进结果；

4)在Swap和Move中选择一个操作进行序列质量改进，$Swap(\sigma',i,j),j\in N_i$和$Move(\sigma',i,k),k=\sigma'(j),j\in N_i$，从中找出能够最小化空间占用的优先序列，定义为序列x；

5）如果序列x优于σ''，那么把x赋给σ''；

6）返回最优序列σ''；

为了更清楚的描述算法2，可见流程图4-4。

这种局部邻近搜索的方法，在序列中空间请求少的时候看不出其相对于局部搜索的优势，但是一旦空间请求数量增加，就可以看出局部邻近搜索在时间上的绝对优势。但是，局部邻近搜索所得出的解只是局部最优解，而在一些时候局部最优解并不等于全局最优解，这就需要有一定的方法来跳出局部最优。局部邻近搜索得出的结果可能会落入局部最优，所以我们要进行一些改进，来改变局部最优的情况。

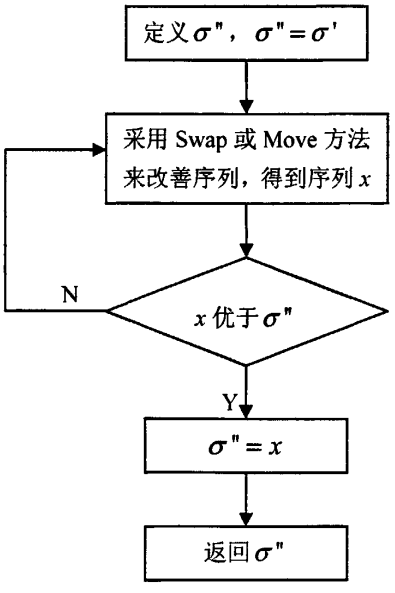


图4-4　局部邻近搜索方法流程图

4.3 跳出局部最优解的方法

在模拟退火算法中，为了跳出局部最优，一般采取的方法是接受一个较差结

果，然后再重新循环求解。而在 SWO 优化算法中，通过对解的分析找出关键请求，通过提高关键请求的优先级来重新求解。

因此对于由于使用局部邻近搜索造成的序列局部最优的情况，我们可以借鉴模拟退火算法或者 SWO 优化算法，通过改变序列中关键请求的优先级，来脱离局部最优。

4.3.1 关键空间请求的选择方法

要想脱离局部最优，首先要选择关键空间请求。在 SWO 优化算法中，关键请求总是选择那些超出规定最多的和最影响结果的，然后增加关键请求的优先级以便能够得到更好的序列。在大多数情况下，单纯的增加关键请求的优先级是行不通的，有些时候减少关键请求的优先级对于求得最优序列将有效的多。

例如在图 4-5 中，改进分配的最有效的方法是把 R_3 的优先级降为 1，执行操作 $Move(\sigma,3,1)$，这样就能达到图 4-1 中的效果。相反的，如果增加 R_3 的优先级效果会更差，会更加偏离最优结果。

因此，为了能够更好的确定哪个请求为关键请求，需要进一步做出分析判断。为了确定标准，定义一个函数 $Z(R_i)$，表示一个空间请求的关键度，关键度越高，表示其影响力越大。

在 R_i 的时间段内，定义 $Z(R_i)=\sum Z(Y_{i,t})$，$t\in T_i$。如果在一个时间空间请求有交叠，那么关键度 $Z(Y_{i,t})$ 的值就为 1；相反的值就为 0。

例如图 4-5 中所示，可以求出 $Z(Y_{3,7})=Z(Y_{3,8})=Z(Y_{3,9})=Z(Y_{3,10})=Z(Y_{3,12})=1$，$Z(Y_{3,11})=0$，所以 $Z(R_3)=5$；同理，$Z(R_1)=Z(R_2)=Z(R_4)=Z(R_5)=2$，$Z(R_6)=1$。因此，就像我们所预期的一样，$R_3$ 是关键请求。

根据每个空间请求的关键度函数 $Z(R_i)$，就可以通过下面的方法来选择出一个由 k 个关键请求组成的集合 S。假设 $Z_{\max}$ 是 $Z(R_i)$ 函数中的最大值，ρ 为人为确定的一个比例，则在 $Z(R_i)$ 值大于 $\rho Z_{\max}$ 的空间请求中，随机的挑选 k 个请求组成关键请求集合。ρ 和 k 的值的选取都会影响到产生结果的速度和质量，所以要随需求的变化而调整。

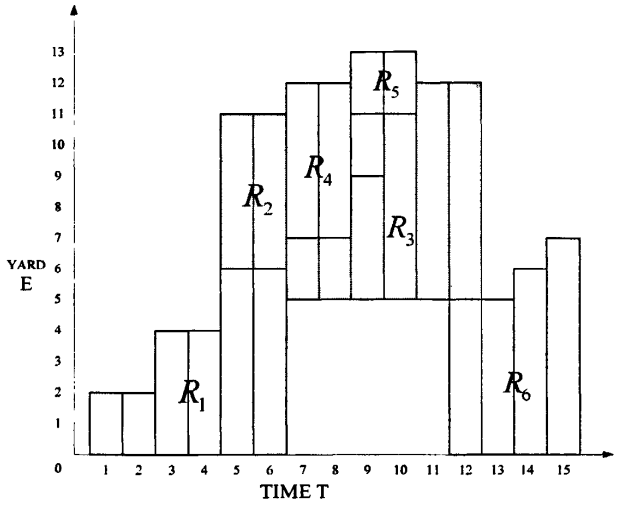


图 4-5 一个未达到最优的分配图

4.3.2 关键空间请求的移动

采用上文方法把关键请求选择出来后，就需要对其优先级进行改变以期望获得更好的空间排序结果。正如上文中所说不能只增加优先级或者减少优先级，特别当一些关键请求已经有了很高或者很低的优先级。

所以为了能确保有一个较优的序列，本文采用随机移动的方法来改进序列。在关键请求的序列中，随机的选择一个请求，进行 Move 方法，移动的目标是与之时间相交叠的空间请求的优先级。如图 4-5 中，如果要移动关键请求 R_3 的优先级，那么移动的对象为与之相交叠的 R_4, R_5 或者 R_6。经过改善后，得到优先序列。

4.4 仿真试验

4.4.1 数据结构

为了能够通过计算机模拟出堆场空间分配，我们使用一个二维的矩阵来表示集装箱堆场，矩阵横向表示时间，竖向表示空间，矩阵大小可以根据请求个数的多少来预先设定。此矩阵初始状态下，矩阵所有数值均为 0，表示空间还未分配，当空间被分配时，矩阵对应中的值变化为响应请求的符号。以第三章中仿真试验的 6 个空间请求为例，初始状态矩阵如图 4-6，优化后矩阵如图 4-7 所示。

$$
\left[\begin{array}{c|cccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

图 4-6 初始状态矩阵

$$
\left[\begin{array}{c|cccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 4 & 4 & 5 & 5 & 0 & 6 & 6 & 6 & 6 \\
1 & 1 & 1 & 1 & 1 & 1 & 4 & 4 & 5 & 5 & 0 & 6 & 6 & 6 & 6 \\
0 & 0 & 1 & 1 & 1 & 1 & 4 & 4 & 0 & 0 & 0 & 6 & 6 & 6 & 6 \\
0 & 0 & 1 & 1 & 1 & 1 & 4 & 4 & 0 & 0 & 0 & 6 & 6 & 6 & 6 \\
0 & 0 & 0 & 0 & 1 & 1 & 4 & 4 & 0 & 0 & 0 & 6 & 6 & 6 & 6 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 3 & 3 & 3 & 3 & 0 & 6 & 6 \\
0 & 0 & 0 & 0 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 0 & 0 & 6 \\
0 & 0 & 0 & 0 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

图 4-7 优化后矩阵

本试验在 VisualStudio 环境下，使用 C#语言进行编码。在产生初始序列的编码中则是通过比较时间得出，然后通过关键请求局部搜索来改进。在这个过程中要不断应用 For 循环来进行序列的调整。

4.4.2 仿真结果及分析

为了检验该算法的有效性，本章中提出了一个 170 个请求的空间分配问题，具体请求数据见附录。

在编码过程中，将所有的空间请求存入一个一维数组中，而求最优序列的过程就是不断变化请求在数组中位置的过程。进行 Swap 操作就是将数组中的两个请求互换位置，而在进行 Move 操作时，就是将数组中的一个请求取出，插到相应的位置，插入位置后的请求在数组中相应的后移。

首先我们分别求出 30，60，100，130，170 个请求的最小值，结果见表 4-3。

表 4-3 仿真试验结果

请求个数	30	60	100	130	170
最小值	12	31	47	60	77

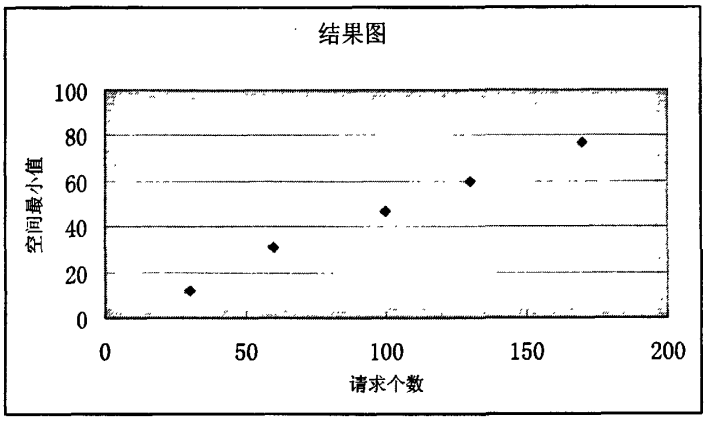


图 4-6 结果图

接下来为了验证基于时间排序产生初始序列的优势，在这里我们采用两种方法生成初始序列，分别是 4.3.1 中介绍的基于时间排序的方法和随机产生初始序列，然后分别以这两个初始序列进行排序，分别对每 10 个请求求得的空间占用最小值，对其求出的结果进行对比。

表 4-3 仿真试验结果对比

请求个数	最小值（按时间产生初始排序）	最小值（随机产生初始排序）
10	11	11
20	12	12
30	12	16
40	24	26
50	26	35
60	31	33
70	34	38
80	36	45
90	41	53
100	47	54
110	54	61
120	54	65
130	60	70
140	60	69
150	68	81
160	72	80
170	77	90

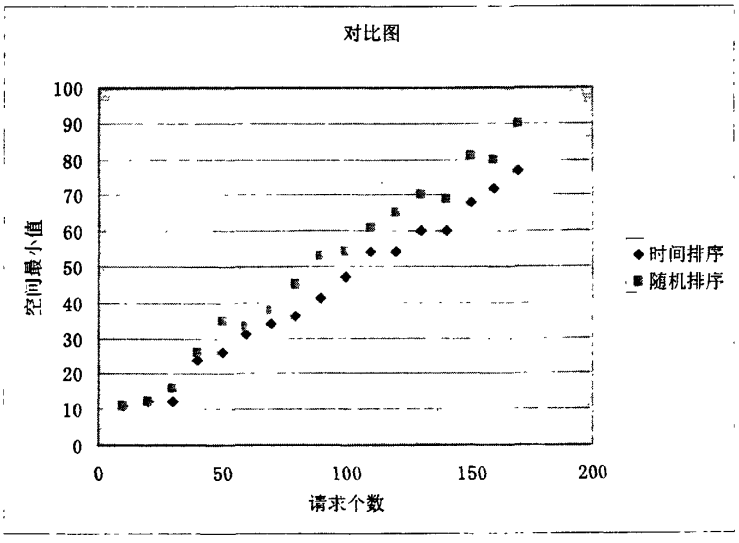


图 4-7 对比图

从上表中，我们可以发现用基于时间排序的方法生成初始序列相对于随机生成的方法来说，更容易收敛于最优解，根据其所生成的最优序列也质量更高。在图 4-7 中，我们更可以清楚地发现当请求个数较少时，两者的最优解不会有太大的差距，但是随着请求个数的不断增大，其差距也不断加大，这都是由随机方法的不确定性引起的，当数量少时不确定性影响较小，可以忽略，但是当数量大时，不确定性得影响就会显露出来。所以当请求个数大于 50 个时，我们应该避免用随机的方法来生成初始序列，以避免误差的增大。而应该应用根据时间排序的方法产生初始序列，以追求准确的最优解。

第五章 集装箱堆场调度系统分析与设计

在前面各章中已经就集装箱堆场空间分配问题进行了详细的理论研究，在这一章中将会研究集装箱堆场调度系统的结构及其实现。

5.1 集装箱堆场空间调度系统分析

5.1.1 前方堆场空间调度系统分析

前方堆场位于集装箱码头前方，为加速船舶装卸作业，是暂时堆放集装箱的场地。其作用是：当集装箱船到港前，有计划有次序地按积载要求将出口集装箱整齐地集中堆放；卸船时将进口集装箱暂时堆放在码头前方，以加速船舶装卸作业。

由前面各章的分析知道，前方堆场的空间调度系统主要功能为安排集装箱进出港，并分配其存放位置。这就需要系统能够及时的根据管理员所输入的集装箱到港计划，分析得出优化后的堆场分配图，并及时地反映出堆场的剩余容量以做出调整，避免出现船舶停港时间过长，导致压港现象。

5.1.2 后方堆场空间调度系统分析-

集装箱后方堆场是指储存和保管空、重箱的场地，包括中转箱堆场、进口重箱堆场、空箱堆场等。它的一个重要功能是对集装箱进行装箱和拆箱。而这里的装箱问题就涉及到一个集装箱内部空间的分配问题。

而对于装箱方法的选择，目前有以下三种方法：

（1）顺序法：顺序法是指在集装箱的某边方向上始终顺序摆放包装件的某边.最后剩余空间能装就装.不能装就放弃。顺序法通常只需优化集装箱的3个面，在每个面所对应的垂直方向上不需要进行优化，自由摆放，摆到不能再摆的高度就放弃。顺序法的优点是顺序摆放，集装方便，集装实体部分无体积损失，缺点是3 个方向边缘剩余空间利用率不高，降低了集装率。

（2）垂直法：垂直法是指在单方向上垂直摆层，并使此方向上摆层后剩余空间最小，但在每一层内进行平面集装优化。3个单方向上摆层，相当于3 次装满集装箱，最后取3 次结果的最优值。每个方向上的摆层方法有3 种.那么3个方

向就有9 种摆法。以其中一个方向为例，设此方向上摆放包装件的3个面的最佳个数分别为$n1,n2,n3$，在此方向上所放包装件3个面分别所对应高的个数为$N1,N2,N3$，显然集装箱集装包装件个数为$V1=n1\times N1+n2\times N2+n3\times N3$。同理，也可以求出其它两个方向上的集装包装件个数$V2$和$V3$。最后取3 次结果的最大值：$VB=\max[V1,V2,V3]$。这就是垂直法的优化结果，垂直法的结果有时接近最优。从摆层方案上看，垂直法是单方向上摆层，摆层方案最简单。

（3）改进法：改进法就是在充分利用顺序装法实体部分无体积损失的优点基础上，充分挖掘边缘剩余子空间的一种集装优化方法。通过装载优化，可以使得大大提高集装箱空间的利用率。但往往装载方法比较复杂。

5.2 集装箱堆场空间调度系统设计

集装箱前方堆场进行空间分配的前提是需要知道集装箱到港计划，也就是需要有堆场计划作为实施分配的前提。因此整个空间分配都是基于堆场计划进行的。

5.2.1 堆场计划调度过程

堆场计划就是为在港口码头进出口的集装箱合理分配堆场位置，保障最大程度地利用堆场的空间和设备，最低程度上减少倒箱次数，以达到提高港口效率和集装箱吞吐量的目的。所谓的堆场信息化，很大程度上就要利用计算机技术对堆场计划进行模拟仿真，信息化。堆场计划主要分成三个过程：

（1）宏观堆场计划：在堆场上为装载或卸载到船上的进出口集装箱预先分配适当的区域。

（2）微观堆场计划：根据集装箱的进出港口（Port）、尺码（Size）、种类（Category）和重量（Weight，简称 PSCW）在堆场中为它们分配位置。

（3）传送堆场计划：基于在宏观和微观堆场计划阶段对集装箱在堆场中位置的，堆场计划人员把为这些堆场位置的报告作为参数发送到港口操作系统。

图 5-1 描述了堆场计划工作过程图，具体的堆场计划工作过程如下：

在进行堆场计划活动之前，集装箱码头堆场的外部形态假设已经被详细的描述清楚了。这些信息包括各自的操作码头的堆场街区情况。堆场计划人员也传递同样的堆场外部形态信息给港口操作系统，使堆场计划系统和港口操作系统两个系统之间的信息达到一致。

首先，从泊位计划系统得到一个在有效计划时间内到港的船舶的信息列表。堆场计划人员可以从船舶信息列表选择任意一艘船来进行宏观堆场计划。根据期

望的集装箱体积和船舶在港口停靠的位置，堆场计划人员可以分配进口/出口集装箱堆场区域。堆场区域通常在船到泊位前一个星期分配给它。

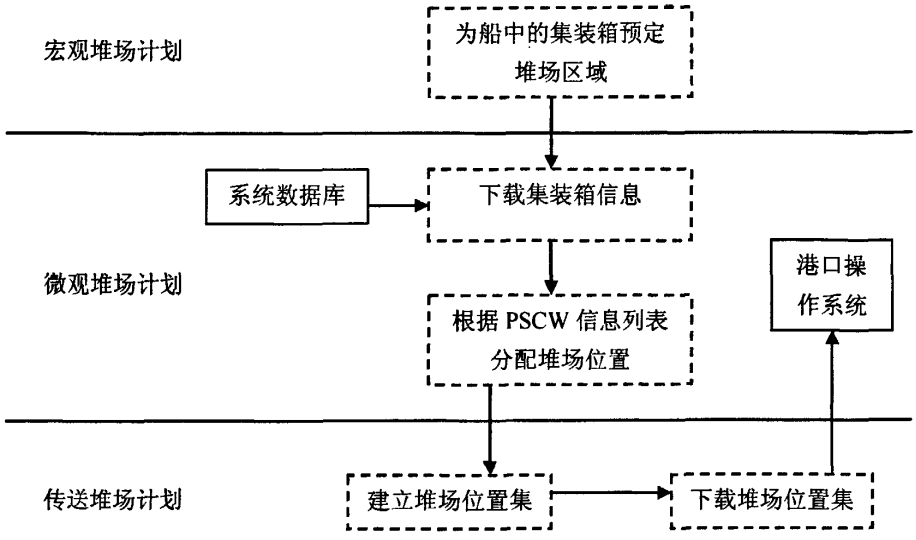


图 5-1 堆场计划工作过程图

然后，堆场计划人员可以开始为船上的集装箱制定微观堆场计划。这个阶段的计划通常在船到达泊位前八到十二小时制定。进口集装箱信息写到细目分类表之后被下载到堆场计划系统中。堆场位置将根据集装箱 PSCW 属性和先前已经被保留的区域的参数进行分配，然后建立相应的进口/出口细目分类表。如果泊位计划系统已经委托了起重机，本地进出口细目表可以根据起重机的数目进行进一步的计划。在这一步，堆场计划人员可以根据需要扩张、收缩、插入或删除现有的堆场区域，来解决细目分类表、集装箱和堆场之间的冲突。

当计划的堆场位置确定以后，堆场计划就要传递给其他系统开始准备实施了。堆场计划人员打印堆场区域报表并确定所有的集装箱已经被分配了合适的堆场区域。以上确定无误后，堆场计划人员上传计划分配堆场区域给港口操作系统。

当集装箱从船上卸载到堆场或从大门里拖到堆场时，操作系统根据这些堆场区域分配真实的堆场位置给集装箱。然后港口操作系统更新堆场计划系统中关于这些被移动的集装的数据，直到集装箱离开堆场。

5.2.2 堆场计划调度的功能

根据上面所描述的堆场计划，可以把整个管理信息系统分成三个主要的部

分，每个部分都具有自己的功能模块，如图 5-2。

1．宏观堆场计划功能

在宏观堆场计划中，历史数据分析主要是完成堆场计划人员对历史数据的分析，从而根据分析得到的信息制定更多的决策。历史数据分析主要有服务序列管理、服务序列的堆场预分配图管理、堆场区域形状管理和报表管理四个功能。

（1）服务序列管理：堆场计划人员根据历史数据按照船名、航次、航向的属性对船进行分类，形成服务序列作为堆场区域划分的数据基础。包括建立服务序列，删除服务序列，编辑服务序列，复制服务序列，重命名服务序列，显示服务序信息等功能。

（2）服务序列的堆场预分配图管理：堆场计划人员需要为该服务序列的分配堆场区域的空间图进行管理。包括建立服务序列的堆场预分配图和显示服务序列的堆场预分配图信息等功能。

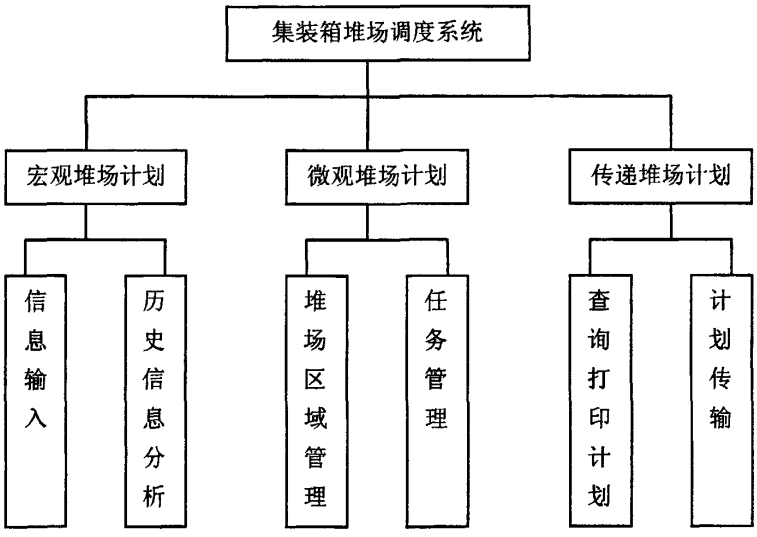


图 5-2 功能模块图

（3）堆场区域形状管理：当在建立服务序列的堆场预分配图的时候，服务序列的堆场预分配图由一系列的堆场区域形状组成，该用例对这些堆场区域进行管理，包括建立堆场区域形状、编辑堆场区域形状、删除堆场区域形状等功能。

（4）报表管理：在建立完服务序列后，服务序列的堆场预分配图和堆场区域形状之后，打印输出查看他们的信息。

2．微观堆场计划功能

在微观堆场计划中，主要是一个动态编制计划的过程，用来完成在堆场中划

分出来堆场区域以存放进出口集装箱，主要包括堆场区域管理和任务管理。

（1）堆场区域管理：在堆场中为集装箱分配合理有效的堆场区域，包括创建堆场区域，删除堆场区域，移动堆场区域和检测堆场区域冲突等功能。

（2）任务管理：当检测出堆场区域有冲突的时候，就会自动生成任务，等待堆场计划人员来解决。

3．传递堆场计划功能

在传递堆场计划中，主要是把制定好的堆场分配计划传递给操作系统，保证计划的执行。包括查询打印计划和计划传输。

5.3 集装箱调度过程的三维实现设计

与堆场空间利用相类似的一个问题是集装箱装载空间利用问题。集装箱装载问题在企业的产品运输中是一个非常重要的环节。生产出来的产品被放入某种规格的长方体纸箱内,由于产品本身的特性,存放产品的包装物在集装箱内的放置形状和大小都不是统一的，需要考虑的因素很多，情况也很复杂。在这个前提下,包装物要紧密地放入集装箱内,以提高集装箱的空间利用率,降低运输成本。同时,由于人力、物力及产品本身的要求,包装物最好一次性装入,而不能反复装卸,否则会极大地限制集装箱调度效率。目前将包装物装入集装箱的工作大多依靠人工操作来进行。调度员在现场凭借经验估计被装入的几种包装物的数量及所需集装箱数,并开出装箱单据,然后装箱工人依照单据将包装物放入。

这种操作程序存在两方面的问题:调度员虽然经验丰富,但作直观估计很难保证高的装箱率;装箱工人的随意性同样造成集装箱的低利用率。无形中,这种人工装箱的方式在货物运输环节中增加了企业成本。如何摆脱人工操作的落后局面,利用计算机技术给出一个合理的集装箱布局及装载方案,以提高集装箱空间利用率和装载。

几十年来，集装箱运输在世界范围内获得了迅速发展，但是它在我国全面应用时间还不长，利用率也不高，一个 20ft 集装箱，容积是 $33m^3$，一般装 $25m^3$，利用率约为 75%；一个 40ft 集装箱，容积是 $67m^3$，一般装 $55m^3$，利用率约为 82%。如果包装箱本身装载不饱满，那么，集装箱的实际利用率会更低。针对集装箱装载效率低的现状,可以结合计算机仿真技术，对集装箱的空间布局寻求最优解。

5.3.1 框架设计

系统由公共界面、优化方法检查与计算、三维模拟计算、图形仿真显示、后置处理等模块组成.总体结构框如图5-3所示。优化方法检查与计算模块会根据用

户在公共界面上输入的参数,自动地生成装载数据,该数据将在公共界面的下侧窗口上显示出来。同时,也可读取磁盘上已有的文件,建立和保存新的文件.三维模拟计算模块对程序数据自动查错并解释成OpenGL 代码，形成三维显示文件。OpenGL是一套图形标准,并且能够在多种平台上应用，不仅能绘制三维模型，并且可以进行三维交互、动作模拟等，非常适合集装箱视景模拟系统。图形仿真显示模块用于显示立方体包装件在装载过程中的不同时刻的位置参数和更新数据。

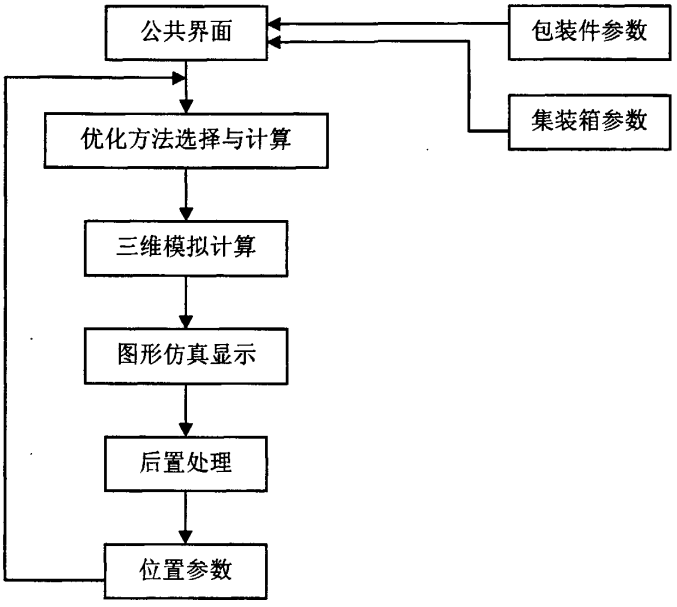


图5-3 系统总体框架设计

三种优化方法的计算结果利用OpenGL 创建模型后,需要把这些模型放在三维空间中的适当位置,并选择最佳角度进行观看。视景转换包括视点、模型、投影和视口转换。同时可以对模型平移、旋转和缩放等,利用模型转换可以确定物体模型在场景中的位置和方向。从视点发出的所有通过对象的射线和投影平面的交点形成了对象的透视投影。在投影转换中要定义视景体。视景体有两个作用:一是确定物体投影到屏幕上是透视投影，还是正视投影；二是确定了场景中有哪些物体要显示在最终的屏幕区域中。为了绘制场景，OpenGL 必须对场景中的每个顶点按照指定的顺序进行转换。最后， OpenGL 执行视口转换,这个转换确定了顶点映像到屏幕上的方式。

立方体必须在视景体中进行坐标变换，由OpenGL 中从三维场景到屏幕图形要经历的变换过程如下: 模型坐标—模型变换—世界坐标—观察变换—观察坐标—投影变换—投影坐标—设备变换设备坐标。OpenGL 坐标系分为:世界坐标

系和当前绘图坐标系。世界坐标是OpenGL 中用来描述场景的坐标，Z 轴垂直屏幕向外，X 轴从左到右，Y轴从下到上，是右手笛卡尔坐标系统。用这个坐标系来描述物体及光源的位置。将物体放到场景中也就是将物体平移到特定位置，旋转一定角度,这些操作就是坐标变换。利用OpenGL的矩阵运算命令,就可实现任意复杂的坐标变换。

5.3.2 实现的技术 OpenGL 介绍

OpenGL最初是SGI公司为其图形工作站开发的、可以独立于操作系统和硬件环境的图形开发系统。目前OpenGL已经成为高性能图形和交互式图像处理的工业标准，OpenGL已被多家大公司采用作为图形标准，并能够在多种平台上应用。

OpenGL实际是一个3D的API(Application Programming Interface)，它独立于硬件设备和操作系统，以它为基础开发的应用程序可以十分方便地在各种平台间移植。从程序员的角度来看，OpenGL是一组绘图命令和函数的集合。在微机版本中，OpenGL提供了三个函数库，它们是基本库、实用库和辅助库。利用这些命令或函数能够对二维和三维几何形体进行数学描述，并控制这些形体以某种方式进行绘制。

OpenGL不仅能够绘制整个三维模型，而且可以进行三维交互、动作模拟等。具体功能主要有：模型绘制、模型观察、颜色模式的指定、光照应用、图像效果增强、位图和图像处理、纹理映射、实时动画。

它严格按照计算机图形学原理设计而成，符合光学和视觉原理，非常适合集装箱视景模拟系统。主要因为：

1、在OpenGL中允许视景对象用图形方式表达，如由物体表面顶点集合构成的几何模型，这类图形数据含有丰富的几何信息，得到的模拟图像能充分表达出其形体特征；而且在OpenGL中有针对三维坐标表示的顶点的几何变换，通过该变换可使顶点在三维空间内进行平移和旋转，对于由顶点集合表达的物体则可以实现其在空间的各种运动。

2、OpenGL通过光照处理表达出物体的三维特征，其光照模型是整体光照模型，它把顶点到光源的距离、顶点到光源的方向向量以及顶点到视点的方向向量等参数代入该模型，计算顶点颜色。因此，视景模拟图像的颜色体现着物体与视点以及光源之间的空间位置关系，具有很强的三维效果。

另外，为了弥补图形方法难于生成复杂自然背景的不足，OpenGL提供了对图像数据的使用方法，即直接对图像数据读、写、拷贝或者把图像数据定义为纹理与图形方法结合在一起生成视景图像以增强效果。为增强计算机系统三维图形的运算能力，有关厂家已研制出了专门对OpenGL进行加速的三维图像加速卡，

其效果可以与工作站相媲美。由此可见，OpenGL非常适合微机环境下模拟集装箱视景系统的开发。

5.3.3 基于 OpenGL 技术的模拟系统开发分析

利用计算机仿真技术对集装箱空间布局寻求最优解，其基本的处理原则是考虑单个包装产品容器的长、宽和高在空间三个方向上的利用率，但是商品一旦摆入空间，与之对应的三块分支空间也就确定了；即使当前包装产品容器的空间利用率达到最优化，也可能由于对分支空间的划分不合理而造成整体的布局不当。因此，应该以包装产品容器的组合空间利用率作为包装产品容器模型的选择机制。通过对空间的不断整合，尽可能排除对包装产品容器摆放产生不利影响的因素，这样就可以实现空间利用率的最大优化。

使用OpenGL实现装载系统的模拟一般步骤包括：视景生成、图像生成和动画表现。

1、视景生成：基于OpenGL标准开发集装箱装载视景系统，应为OpenGL创建适当的图形操作描述表并设置正确的点格式。OpenGL基本库提供了绘制基本图元的函数，如绘制点、线、多边形等，调用这些函数即可构造并显示集装箱空间视景。对于复杂物体的物体装载，这样做显得过于繁琐，浪费大量的时间和计算机资源，可利用其他转化软件和常用的动画软件简化复杂物体的建模，可得到相应复杂问题的顶点坐标值及对应点的纹理坐标值等数据，且内含该物体的OpenGL显示列表，直接用于系统调用生成模拟图像。

2、图像生成：计算机图形生成的途径主要有两类：第一类方法是用计算机图形学的三角形面为基元，配以纹理填充和光照模型，最后以透视投影到视见区产生可视画面，利用OpenGL中的纹理映射技术、融合技术及深度测试可以实现具有真实感的自然景物背景，其缺点是速度受硬件的限制；第二类方法是图像序列法，在外存（光盘或实时磁盘）上存储空间连续的大量图像，可记录相当范围的实际景物，随后根据操纵者的运动选取所需场景予以动态显示。这类方法达到实时要求的关键是高速海量外存，其特点是算法的运算量固定，不随景物复杂度和逼真度的增加而改变，并且用户可以根据需要更换模拟内容，但不利于表达运动物体，通过比较，可以看出图像序列法逼真度高，运算量固定，是一种很有前途的方法，但为获得连续景物显示，需要克服价格昂贵的大容量、高速外存设备等不利因素。

3.动画表现：模拟对象可视化信息的动画表现是一个关键问题。目前，Auto CAD、3D studio、Animator都具有一定的动画功能，但都是传统的计算机动画，即关键帧动画，帧动画是由若干幅连续的画面组成的图像或图形序列，即物体的运动路径需人为指定；而基于物理的计算机动画（也称过程动画、算法动画、造型动画）

可将环境及特征因素考虑在内，运用运动学、动力学等学科规律，建立模型的运动方程，作为动画模型的运动路径。造型动画需对每一个活动对象分别进行设计，赋予每个对象一些特征（如形状、尺寸、颜色等），然后用这些对象拼成完整的画面，这些对象在设计要求下实时变换，最终形成连续的动画过程。由于硬件速度和其他主客观条件的限制，目前只能采用关键帧动画，但三维实体造型动画一直是动画开发人员及用户追求的最终目标。双缓存（double-buffering）显示法是实现实时动画的理想方法，又称实时动画法、换页面动画法。采用双缓存器显示法时，尽量减少在绘图页面上的作图时间是增强动画效果的关键。OpenGL提供了双缓存技术的一系列函数，该技术提供了生成动画效果图形所需要的机制，使得所生成的图形能够像电影一样平滑运动。双缓存即前台缓存和后台缓存，任何时刻只显示前台位面，其绘图历程正常情况下至更新后台位面。为了获得平滑的动画效果，将一个已完全画好的图像在前台位面显示一段时间（如1/30s）。与此同时，清空后台位面并绘好下一幅图像，然后调用SwapBuffers()函数交换前后台位面，把已经生成的图像从内存拷贝到屏幕上显示，这样就很方便的实现了动画效果。

5.3.4 仿真试验分析

系统采用 C/S 架构，应用 C#语言和 SqlServer 数据库构建。主要分为两大功能模块，分别为前方堆场调度模块和后方堆场调度模块，如图 5-4。

在前方堆场界面中主要实现的功能包括入场管理、空间管理和出场管理。

（1）入场管理主要是录入进入前方堆场的集装箱信息，包括货物信息、在堆场存放时间、需求空间大小以及产生的费用，并按时提醒工作人员进行管理。这里输入的信息要详细，以集装箱箱号作为主键存入数据库，这样可以方便管理者对集装箱数据进行统计整理，以便作出合理决策。

（2）空间管理主要是通过本文中所提到的算法对入场集装性存放位置进行管理，以使空间得到最大利用。得到的结果以图形方式直观的表现出来。在特殊情况下可以通过人为调整来达到管理目的。

（3）出场管理是要是提醒工作人员对准备离开前方堆场的集装箱进行操作，包括相应的费用结算，以及将出场集装箱所占用的空间进行释放，更新堆场空间信息，

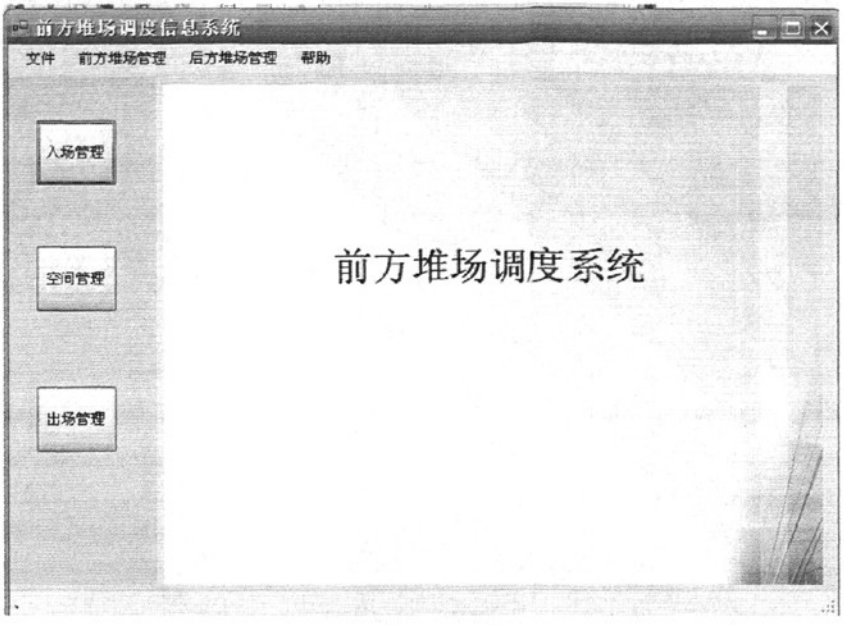


图 5-4 前方堆场调度系统主界面

后方堆场主要是为了集装箱的维修以及集装箱的装箱和拆箱，所以后方堆场调度系统功能主要包括集装箱维护以及装箱功能，如图 5-5。

（1）集装箱维护功能：堆场管理者将后方堆场集装箱维护申请输入系统，系统在适当时间安排维修人员前往。

（2）集装箱装箱功能：这是后方堆场的主要功能，堆场管理者将需要装箱的集装箱尺寸信息以及货物尺寸信息输入系统，系统经过后台计算得出合理的装箱方案，此方案可以用前文提到的 OpenGL 技术，在三维模拟环境下显示出来，从而给装箱操作者一个直观的感觉，便于实际操作。此功能可以加快装箱速度，加少重复操作的次数，缩短集装箱在占用堆场空间时间。

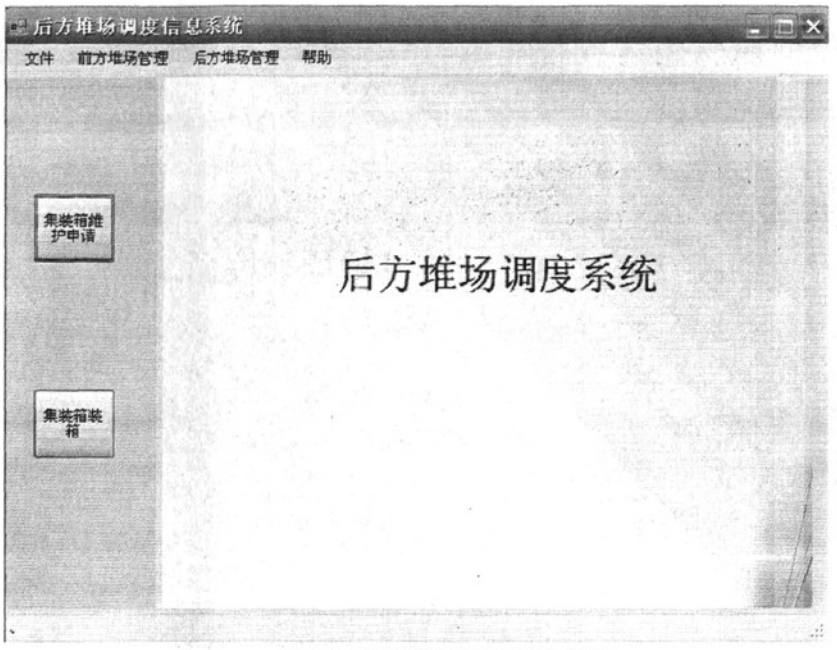


图 5-5 后方堆场调度系统主界面

在集装箱装箱系统中，具体包括如下功能：

（1）数据导入功能：该功能模块读取药装载货物的基本信息，包括货物的名称、编码、数量、长、宽、高、可否旋转、是否中转等，

（2）最优化自动装载：最优化自动装载是本系统的核心功能模块，在计算的过程中，要综合考虑的约束条件有：预先指定可使用的集装箱或可利用的集装箱；考虑装载工具的大小、最大装载重量、允许超载的重量以及超出的长、宽、高的范围；在最大重量范围内达到最大容积装载率。

（3）数据导出功能：对于经过自动计算过的装载结果，经确认后，可以打印输出。

简单的界面演示如下：

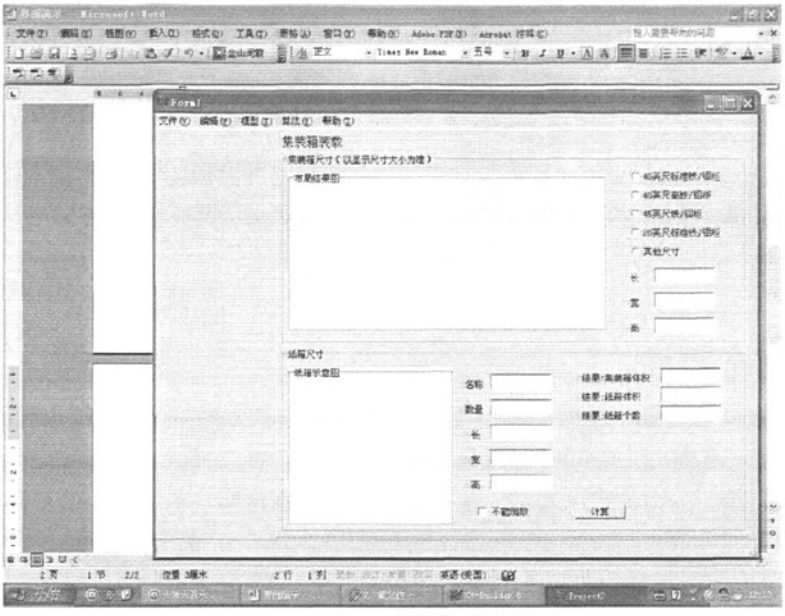


图 5-6 界面数据操作图

输入集装箱及纸箱的信息，用户可以直接点击窗体内已经给定的集装箱的规格，或手动输入集装箱的规格，在左上方的黄色空白框内会自动显示一个按所给规格，程序自动生成的集装箱立体图，相同的在界面右下角，输入纸箱的相关信息，左下方的黄色空白框内会显示之乡的立体图形，点击计算，就会进入三维模拟的界面。

举一个简单的例子：

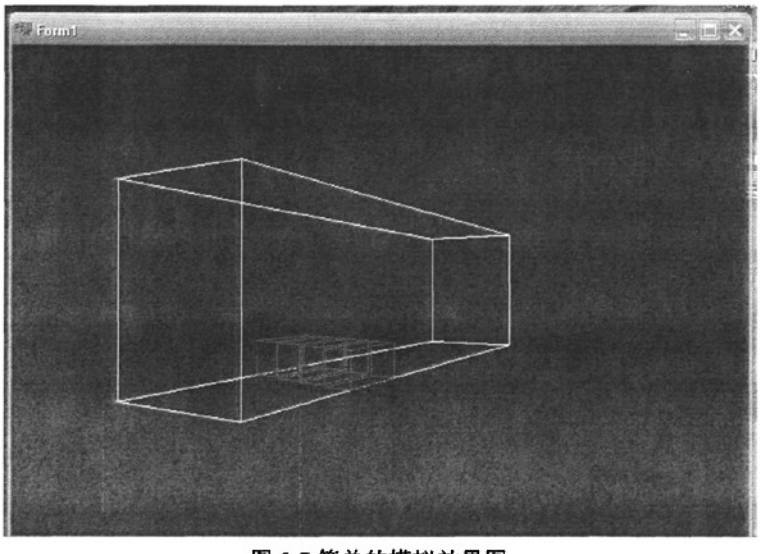


图 5-7 简单的模拟效果图

首先确定系统中的原点，确定三维坐标，以原点为出发点，确定观察点，例如，原点为（0.0，0.0，0.0），可以确定观察点为（0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0,1.0, 0.0）,先以原点为中心，根据程序设定的集装箱的基本信息，将集装箱绘出来，然后在三维坐标的一个维度上摆放纸箱，例如：首先在 x 轴上摆放纸箱，要求纸箱的长度之和不大于集装箱在 x 上的长度，排完后，再选择新的坐标轴进行下一次排箱，三维模拟装箱的图形显示如图 5-7。

5.4 本章小结

在本章中介绍了如何将前几章的堆场空间分配方法应用到堆场调度系统中，以达到在实际中使用的目的。主要描述了堆场调度系统的基本功能结构，探讨了如何将调度过程图形化模拟实现，介绍了 OpenGL 技术，并提出了应用 OpenGL 技术到集装箱装箱过程中，以提高整体调度系统的图形化。

附 录

170 个请求的具体数据：第一行为时间，第二行为空间（单位 TEU）

请求	数据	请求	数据
1	1, 2, 3, 4 4, 6, 8, 8	86	1, 2, 3, 4 4, 5, 7, 10
2	1, 2 3, 5	87	1, 2 3, 3
3	3 1	88	3 2
4	6, 7, 8, 9, 10 4, 4, 4, 4, 7	89	5, 6, 7, 8 3, 4, 6, 6
5	6 3	90	9, 10, 11, 12 3, 3, 5, 7
6	7 2	91	13, 14, 15 2, 2, 4
7	7 3	92	17, 18, 19, 20, 21, 22 3, 4, 7, 10, 11, 11
8	8, 9 4, 7	93	24, 25, 26, 27, 28 1, 3, 6, 7, 9
9	11, 12, 13 2, 3, 3	94	23, 24, 25 1, 4, 7
10	14, 15, 16 1, 3, 6	95	29, 30, 31, 32, 33, 34 4, 7, 9, 12, 12, 15
11	14 1	96	26, 27, 28, 29 4, 5, 5, 8
12	17 1	97	42, 43 3, 5
13	18, 19, 20 4, 5, 7	98	44, 45, 46, 47, 48 3, 6, 9, 10, 10

14	23, 24, 25, 26, 27 4, 5, 8, 9, 11	99	49, 50, 51, 52 2, 2, 2, 2
15	28, 29, 30, 31, 32 3, 6, 8, 11, 12	100	39, 40, 41, 42, 43, 44 1, 3, 6, 7, 10, 12
16	33, 34, 35 3, 3, 5	101	49, 50 3, 3
17	37 3	102	49, 50 3, 3
18	38, 39, 40 4, 5, 5	103	46, 47, 48, 49 2, 4, 4, 5
19	41, 42, 43 3, 4, 7	104	39, 40 2, 6
20	44, 45, 46, 47, 48, 49 4, 4, 6, 8, 9, 13	105	35, 36, 37, 38, 39 4, 5, 7, 7, 9
21	50, 51, 52, 53, 54, 55 1, 2, 3, 5, 7, 8	106	40, 41, 42 3, 6, 6
22	11, 12, 13, 14 4, 5, 8, 8	107	1, 2, 3 3, 4, 6
23	17, 18, 19 2, 4, 5	108	10, 11, 12, 13, 14 2, 2, 3, 4, 6
24	21, 22, 23 3, 6, 9	109	2 4
25	24 4	110	5, 6, 7, 8, 9, 10 2, 3, 3, 5, 8, 11
26	33, 34 4, 5	111	11, 12 3, 6
27	36, 37 1, 4	112	13, 14, 15 4, 6, 8
28	35, 36 4, 6	113	16, 17, 18, 19 4, 4, 5, 6
29	42 3	114	20, 21 2, 2
30	40, 41, 42, 43, 44, 45 2, 2, 2, 2, 5, 6	115	20, 21, 22, 23, 24 4, 4, 5, 6, 7

31	1, 2, 3 2, 5, 6	116	26, 27, 28, 29, 30, 31 4, 4, 7, 9, 10, 12
32	4, 5, 6 3, 3, 6	117	31, 32, 33 1, 4, 5
33	4, 5 1, 1	118	35, 36, 37 4, 5, 6
34	6 3	119	40, 41 3, 5
35	7, 8, 9, 10 4, 5, 6, 6	120	44, 45 1, 1
36	11, 12 4, 5	121	42, 43, 44, 45, 46 1, 2, 3, 3, 3
37	13, 14, 15, 16 3, 3, 5, 7	122	48, 49, 50 3. 4. 6
38	17, 18, 19, 20, 21 3, 6, 8, 11, 14	123	2, 3 3, 5
39	25, 26, 27, 28, 29, 30 3, 6, 7, 11, 13, 16	124	4, 5, 6, 7 2, 3, 3, 5
40	34 1	125	2, 3 3, 3
41	31, 32, 33, 34, 35 3, 4, 4, 7, 9	126	4, 5, 6, 7, 8 3, 5, 7, 10, 13
42	36, 37, 38, 39, 40 4, 6, 7, 10, 10	127	12 2
43	41, 42, 43, 44 3, 5, 5, 5	128	16, 17, 18 1, 1, 1
44	43, 44, 45 3, 4, 7	129	13, 14, 15, 16, 17, 18 1, 4, 7, 8, 8, 9
45	47, 48, 49, 50 2, 5, 5, 8	130	9, 10, 11, 12, 13, 14 3, 4, 4, 5, 5, 6
46	1, 2, 3, 4, 5 2, 3, 4, 8, 11	131	19, 20, 21, 22 2, 5, 8, 10
47	7, 8 1, 1	132	23, 24, 25, 26, 27 3, 3, 4, 4, 7

48	7, 8, 9, 10, 11, 12 4, 7, 9, 9, 12, 12	133	23, 24, 25, 26 2, 4, 7, 7
49	14, 15 2, 3	134	27, 28 2, 5
50	13, 14 2, 2	135	29, 30, 31, 32 3, 6, 7, 7
51	13, 14, 15, 16, 17 4, 7, 10, 10, 11	136	34, 35, 36 2, 3, 3
52	20 3	137	36, 37 2, 2, 4
53	23, 24, 25 1, 4, 7	138	38, 39, 40 4, 4, 5
54	23 4	139	41, 42, 43, 44, 45, 46 2, 4, 5, 5, 7, 10
55	26, 27, 28 2, 4, 7	140	47, 48, 49, 50 2, 2, 3, 3
56	29 4	141	47, 48, 49, 50 4, 7, 9, 9
57	31, 32, 33, 34 1, 1, 3, 4	142	1, 2, 3, 4 3, 6, 7, 8
58	31, 32, 33, 34, 35, 36 2, 2, 4, 5, 6, 7	143	8, 9, 10, 11 1, 4, 6, 6
59	39, 40, 41 2, 2, 3	144	17, 18, 19, 20 3, 6, 6, 8
60	37, 38, 39, 40, 41, 42 3, 4, 6, 9, 10, 13	145	25 4
61	44, 45, 46, 47 3, 3, 6, 6	146	26, 27, 28, 29 3, 5, 7, 8
62	48 2	147	31, 32, 33, 34, 35, 36 2, 3, 6, 8, 9, 11
63	44, 45, 46, 47 4, 5, 5, 5	148	37, 38 3, 6
64	1, 2, 3, 4, 5, 6 3, 6, 6, 8, 11, 14	149	39, 40, 41, 42, 43, 44 3, 5, 5, 8, 9, 9

65	7 2	150	37 4
66	7 3	151	37, 38, 39, 40, 41 3, 3, 5, 6, 7
67	9, 10, 11, 12, 13 1, 4, 5, 8, 11	152	45, 46, 47, 48 1, 2, 3, 6
68	8, 9 2, 3	153	43, 44, 45, 46, 47 1, 4, 5, 5, 8
69	7, 8, 9, 10 3, 4, 4, 4	154	48, 49 1, 4
70	19, 20, 21, 22, 23, 24 4, 6, 6, 6, 8, 11	155	1, 2, 3, 4, 5, 6 3, 4, 4, 5, 5, 5
71	13, 14, 15, 16, 17, 18 2, 5, 6, 9, 9, 11	156	4, 5, 6 2, 2
72	19, 20, 21, 22, 23 3, 3, 4, 6, 9	157	7, 8, 9, 10, 11 3, 6, 7, 9, 9
73	20 2	158	17, 18, 19, 20, 21, 22 1, 2, 4, 5, 5, 6
74	26, 27, 28 1, 2, 5	159	13, 14, 15, 16, 17 3, 3, 3, 3, 6
75	24, 25, 26, 27 4, 5, 6, 6	160	6, 7, 8 1, 1, 2
76	29, 30, 31, 32, 33 1, 4, 7, 9, 9	161	12, 13, 14, 15 4, 7, 8, 8
77	36 3	162	16, 17, 18, 19 4, 7, 8, 10
78	37, 38 1, 1	163	21, 22, 23, 24, 25, 26 4, 7, 9, 10, 11, 12
79	35, 36, 37, 38 3, 6, 8, 11	164	27, 28 1, 3
80	40, 41 2, 3	165	28, 29, 30, 31, 32, 33 1, 4, 7, 8, 11, 12
81	43, 44 1, 4	166	34, 35, 36, 37, 38, 39 4, 7, 7, 7, 10

82	44, 45, 46 3, 4, 4	167	42, 43, 44, 45, 46 1, 3, 3, 3, 5
83	47, 48, 49, 50 3, 3, 6, 9	168	48, 49, 50 3, 5, 6
84	47, 48 1, 3	169	50, 51, 52, 53 2, 4, 4, 7, 7
85	49, 50, 51, 52 1, 4, 7, 10	170	43, 44 4, 5

参考文献

[1]Sculli, D.,and Hui, C.F.(1988),Three dimensional stacking of containers, OMEGA 16, 585-594

[2]E.mcdowill, g martin d.cho,west.A study of maritime container handling Oregon state university.sea grant college program ads Corvallis, Oregon 97:1-10,1985

[3]Taleb-Ibrahimi, M, De Castilho, B., Daganzo, C.F.(1993),Storage space vs hangling work in container terminals,Transportation Research B 27,13-32

[4]De Castilho, B.and Daganzo, C.F.(1993),Handling strategies for import container at marine terminals,Transportation Research B,27B:151-166

[5]Peter Preston,Erhan Kozan 2001, An approach to determine storage locations of containers at seaport terminals Computers&Operations Research 28(2001), 983-995

[6]Jean-Francois Cordeau et al Solving Berth Scheduling and Yard Management Problems at the Gioia Tauro Maritime Terminal , 2001

[7]Ping Chen, Zhaohui Fu, Andrew Lim 2002, the yard allocation problem Eighteenth national conference on artificial intelligence. p56-65

[8]Chuqian Zhang a, Jiyin Liu a, Yat-wah Wan, 2001 Storage space allocation in container terminals Transportation Research Part B 37(2003):883-903

[9]Ebru.K.,Bish(2003), A multiple-crane-constrained scheduling problem in a container terminal,European Jounal of Operational Research, 144(1), 83-107

[10]徐剑华 用择箱指数法优化集装箱货场的利用效率和取箱效率 港口装卸 1991.72:46-51

[11]宋天威 集装箱码头堆场堆码方式的研讨.2000.集装箱港站 8-10.

[12]郝聚民，纪卓尚，林焰(2000)，混合顺序作业 BAY 优化模型，大连理工大学学报 40(1)，102-105

[13]杨淑芹，张运杰，王志强(2002)，集装箱堆场问题的一个数学模型及其算法，大连海事大学学报 28,115-117

[14]林祖乙，国际集装箱运输，北京：人民交通出版社，1997

[15]齐二石，方庆馆，物流工程，北京：机械工业出版社，2006.5

[16]郑全成，运输与包装，北京：清华大学出版社，2007.3

[17]武德春，集装箱运输实务，北京：机械工业出版社，2003

[18]陆见成，李正宏，集装箱运输对港口发展趋势的影响，中国港口，2003（3）：24-25

[19]洪承礼，港口规划与布置，北京：人民交通出版社，1999 年：64-67

[20]戈闻怡，集装箱港口作业物流资源配置的研究，上海海运学院硕士学位论文，2003 年：38-40

[21]Andrew Lim, Zhou Xu, A critical-shaking neighborhood search for the yard allocation problem, European Journal of Operation Research, 174(2006),1247-1259

[22]P. Chen, Z. Fu, A. Lim, The yard allocation problem, Proceedings of the Eighteenth American Association for AI National Conference(AAAI), Edmonton, Alberta, Canada, 2002,3-8

[23]P. Chen, Z. Fu, A. Lim, Port yard optimization problem, IEEE Transaction on Robotics and Automation, 1(2004), 26-37

[24]P. Chen, Z. Fu, A. Lim, B. Rodrigues, The general yard allocation problem, Proceedings of Genetic and Evolutionary Computation Conference 2003, Chicago, Illinois, USA, 2003, pp. 1986-1997

[25]B. Hajek, Cooling schedules for optimal annealing, Mathematics of Operations Research 13(1998), 311-329

[26]D.E. Joslin, D.P. Clements, Squeaky wheel optimization, Journal of Artificial Intelligence Research, 10(1999), 353-373

[27]S. Kirpatric, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science, 220(1983), 671-680

[28]A. Lim, On the ship berthing problem. Operations Research Letters, 22(1998), 105-110

[29]邢文训，谢金星，现代优化计算方法，北京：清华大学出版社，1999 年：247-293

[30]N. Bostel and P. Dejax, Models and algorithms for container allocation problems on trains in a rapid transshipment shunting yard, Transport Sci, 32(4)(1998), 370-379

[31]L. Gambardella, A. Rizzoli, and M. Zaffalon, Simulation and planning of an intermodal container terminal. Simulation, vol.71, pp.107-116,1998

[32]W.H. Atkins, Modern marine terminal operations and management, San Francisco, The Compage Company, 1983

[33]E. Edmond, Operation capacity of container berths for scheduled service by queuing theory, Dock Harbor Authority 56(1975), pp.230-234

[34]F. Sabria, C. Daganzo, Queuing systems with scheduled arrivals and established service order, Transportation Research B 23(1989), 159-175

[35]R. Otten, L. Van Ginneken, The Annealing Algorithm, Kluwer Academc Publishers, Boston, 1989

[36]蔡临宁，物流系统规划—建模与仿真实例分析，北京：机械工业出版社，2003

[37]张晓琴，黄玉清，基于禁忌搜索的启发式求解背包问题算法，电子科技大学学报，2005，34(3)：359-362

[38]李大卫，王莉，王梦光，遗传算法与禁忌搜索的混合算法，系统工程学报，1998，13(3)：28-34

[39]Wang. Y, Using genetic algorithm methods to solve course scheduling problems, Expert Systems with Applications, 2003, 25(1): 39-50

[40]Glover. F, Kelly. J, Laguna. M, Genetic algorithms and tabu search: hybrids for optimization. Computers and Operations Research, 1995, 22(1): 111-134

[41]Chelouah. R, Siarry. P, Tabu search applied to global optimization, European Journal of Operational Research, 2000, 123(2): 256-270

[42]李建忠，集装箱港口堆场资源配置研究，大连海事大学学报，2005，(2): 65-69

[43]郭媚，集装箱码头堆场优化管理研究，大连海事大学，2006

[44]Chun Jin, Y. Ren, Optimum Planning on Material Handling Operation in Container Terminal, Transactions of the Japan Society of Mechanical Engineers, 1999, 65(640): 4931-4938

[45]彭传圣，汉堡港的自动化集装箱码头，集装箱化，2005(2)：21-23

[46]罗先渝，集装箱堆场计算机管理系统，集装箱化，2000(9)：24-25

[47]曹锦方，信息系统分析与设计，北京：北京航空航天大学出版社

[48]陈其，刘国良，OpenGL 三维图形系统开发与实用技术，重庆：重庆大学出版社，2003. 8

发表论文和科研情况说明

参与的科研项目：

1、陕西省安康市供电局管理信息系统

2、基于智能表示的动态空间调度管理理论与方法研究及在物流管理中的应用

致　谢

本论文的工作是在我的导师李波教授的悉心指导下完成的，李波教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢两年来李波教授对我的关心和指导。

在进行科研工作及撰写论文期间，室友刘波、王冬、程春光等同学对于我论文的研究工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢家人，他们的理解和支持使我能够在学校专心完成我的学业。