第48卷第5期
2008年9月

大连理工大学学报
Journal of Dalian University of Technology

Vol. 48, No. 5
Sept. 2008

船舶、土木工程

文章编号：1000-8608(2008)05-0673-06

集装箱船全航线预配优化模型与算法研究

张维英*[1,2]，林　焰[1]，纪卓尚[1]，孙文志[2]，余报楚[2]

（1.大连理工大学 船舶CAD工程中心，辽宁 大连　116024；
2.大连水产学院 海洋工程学院，辽宁 大连　116023）

摘要：集装箱船全航线配载问题属于NP-hard问题.为降低问题求解难度，提出了解决全航线配载问题的分解算法，即将配载问题分解为Bay位选择和Bay位中集装箱排序两个子问题.将Bay位选择看成是“装箱问题”，以不同属性集装箱作为待装“物品”，以船舶上的Bay位为箱子，以最优装箱（即使用箱子的数量最少）及集装箱在每个港口的倒箱数量最少为目标进行总布置配载；Bay位中集装箱排序是将Bay位选择阶段分配到不同Bay位的集装箱按某些规则进行排序，确定其在Bay位中的具体箱位.主要研究了Bay位选择阶段的模型及算法.实例模拟结果表明该方法可行，为集装箱船全航线配载优化提供了一个实用的模型.

关键词：集装箱船；预配；装箱算法；二叉搜索树；全航线

中图分类号：U695.22　　**文献标志码**：A

0　引　言

集装箱配载问题是学术界和集装箱运输业最关注的问题之一，是具有复杂约束的NP-hard组合优化问题[1~4]，问题的规模取决于船舶载箱量、挂靠港口数量和在每一个港口装卸集装箱数量.即使对装箱量很小的集装箱船配载问题由于考虑的因素太多（如船舶强度、稳性与浮态、危险货物隔离、货物类型、箱子尺寸与挂靠多个港口等）也是大规模的复杂优化问题.

对于NP-hard问题，随着问题规模的增加，解空间成指数倍扩大呈现“爆炸”现象.例如，对于容量为100 TEU的集装箱船，在空船装载时，可行的装载位置约是9.3×10^{157}个；如果集装箱船的容量为1 000 TEU，则在空船装载时可行的装载位置是$4\times10^{2\,567}$个.随着载箱量的增加，装载位置的增加呈指数倍增长.这是容易理解的，如果m表示船上箱位数量，n表示集装箱数量，则n个集装箱在船上位置确定是一个排列问题.

对于NP-hard问题，目前还未能找到多项式时间算法[5].为了减少配载问题的求解难度，将配载问题分解为两个子问题，即集装箱Bay位选择（也称为预配）及Bay位排箱两个子问题，本文给出问题的分解模型，并根据模型特点提出相应的优化算法.

1　配载问题描述及分解求解策略

1.1　全航线配载问题的分解

将集装箱船全航线配载问题分解为2个子问题[6].

子问题1：Bay位选择（或称为预配）.以船上Bay为装载单位，将集装箱按类别（如尺寸、目的港、不同种类箱等）组成同类箱组，以同类箱组为装载单元，采用装箱算法确定同类箱组在船上纵向布置，完成配载的总布置图，即预配图.

子问题2：Bay位中集装箱排序.根据Bay位中集装箱的属性（目的港），依据一定的配载规则，如满足船舶稳性、倒箱数量最少等要求，确定集装箱在Bay位中的具体位置.

收稿日期：2006-11-15；　**修回日期**：2008-08-11.
基金项目：辽宁省教育厅高等学校科研计划基金资助项目（05L091）.
作者简介：张维英*（1963-），女，博士，教授；林　焰（1963-），男，教授，博士生导师；纪卓尚（1938-），男，教授，博士生导师.

配载问题的分解算法结构如图 1 所示.

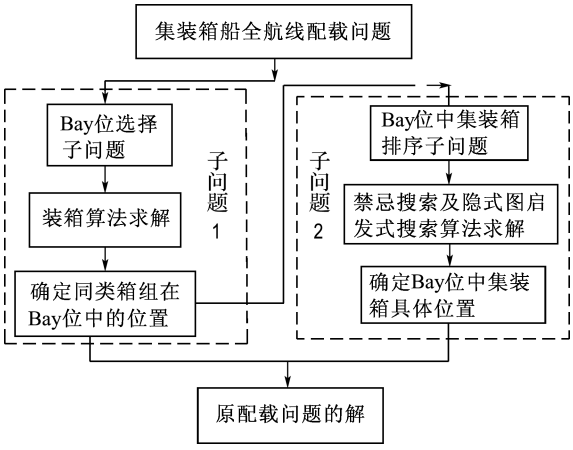

图 1 集装箱船全航线配载问题分解求解策略

Fig. 1 Strategy for containership stowage in full route

1.2 集装箱船全航线预配求解策略

预配问题求解策略:将预配问题看成是“装箱”问题,采用“装箱”算法求解. 在预配时,将 Bay 位看成是容量为 c_j($j=1,2,\cdots$)的箱子(c_j 为 Bay 位中箱位数量),在某个港口装载的同类箱组(相同属性、相同尺寸、相同目的港的集装箱)看成是待装入箱子的物品,(在港口 p)同类箱组中集装箱数量看成是物体所占单元个数(即物品的尺寸). 求对当前港使用最少 Bay 位数量的装载为最优装载,并且满足倒箱量最少的要求.

考虑到集装箱配载时,同一目的港的集装箱数量可能大于船上 Bay 位中箱位数量,所以要考虑物品尺寸超过最大箱子容量时的装载问题,此时必须拆分超尺寸物品才能装箱,在限定拆分次数的条件下物品可以按任意比例拆分. 集装箱配载问题应使用超尺寸物品装箱算法求解.

Bay 位中集装箱排序:将已分配到各个 Bay 中的集装箱以一定的规则指派到特定的箱位上去,优化目标是横倾力矩最小、重心高度最优和倒箱数量最少,根据 Bay 位中集装箱的属性不同,求解的基本方法是禁忌搜索算法和隐式图搜索方法. 本文主要研究子问题 1 的模型及算法.

2 超尺寸物品装箱问题描述及算法

2.1 超尺寸装箱问题描述

2.1.1 经典一维装箱问题(bin packing problem,简称 BPP 问题) BPP 问题是要求把一些物品放入具有固定容量的箱子中,严格的数学定义为

给定 n 个物品序列 $A=(a_1,a_2,\cdots,a_n)$,物品 $a_i(1\leqslant i\leqslant n)$ 的大小为 $s(a_i)\in(0,1)$,要求将这些物品装入容量为 1 的箱子序列 $b_1,b_2,\cdots,b_m$,使得每个箱子中物品大小之和不超过 1,并使所用箱子数量 m 最小.

经典的一维装箱问题,处理对象是 n 个物品序列和一个无限多的等容量的箱子序列,目标是把所有物品放入箱子中并最小化所使用的箱子数目.

2.1.2 可变尺寸装箱问题(variable size bin packing problem,简称 VSBP) 给定一系列物品及不同容量的箱子,每种箱子都有无穷多个,物品尺寸均不超过最大箱子容量,要求将所有物品装入箱子中,使得所用箱子的总容量最小. 可变尺寸物品装箱问题的数学模型可描述如下:

设有限种物品集 $A=\{a_1,a_2,\cdots,a_n\}$,每个物品需占用 $s(a_i)$ 个单元($i=1,2,\cdots,n,n\in\mathbf{Z}^+$,其中 $\mathbf{Z}^+$ 为正整数集). 有若干个箱子 $b_{in}=\{l_1,l_2,\cdots,l_m\}$ 组成的集合,$c_j(j=1,2,\cdots,m)$ 是箱子 l_j 的容量. 令 $B_{in}=\{b_i,b_2,\cdots,b_k\}$ 表示某个实例中使用箱子的数量,目标函数是极小化下列函数:

$$\min f(b_1,b_2,\cdots,b_k)=\sum_{j=1}^{k}c_j \tag{1}$$

2.1.3 超尺寸物品装箱问题(bin packing problem with over-sized items,简称 BPOS) 如果物品尺寸超过最大箱子的容量(称这样的物品为超尺寸物品),需要拆分才能装箱,如何将所有物品装入箱子中,使得所用箱子的总容量最小,称之为超尺寸物品装箱问题[7,8],超尺寸物品装箱问题可通过物品拆分变成可变尺寸物品装箱问题,此类问题属于 NP-hard 问题.

对于此类问题有许多近似算法,目前最为流行的有 4 种:最先匹配法(FF)、最优匹配法(BF)、最先匹配递减法(FFD)及最优匹配递减法(BFD). 这 4 种算法都不能保证最优装载[9].

2.2 基于 BFD 箱子装载问题的算法[10]

Step 1 初始化箱子容量、箱子标号、物品数量及各物品所占空间.

Step 2 以箱子容量为关键值构造带有重复

值的二叉搜索树，并为每个节点标号.

Step 3　依次装载每个物品，如果物品的尺寸大于容量最大的箱子，则将物品拆分，先将超尺寸物品装入当前容量最大的箱子中. 如果拆分后物品的残余部分还大于箱子的容量，则继续拆分，否则转 Step 4，将其与其余的物品按下述方法装载.

Step 4　搜索. 寻找最优匹配的箱子，假设 a_i 物品所占的空间为 $s(a_i)$，寻找大于等于 $s(a_i)$ 的关键值：首先从根开始，如果根为空，那么搜索树不包含任何关键值，查找失败. 否则，将 $s(a_i)$ 与根的关键值相比较，如果 $s(a_i)$ 小于根节点的关键值，那么就不必搜索右子树中的元素，只要在左子树中搜索即可；如果 $s(a_i)$ 的值大于根节点的关键值，则正好相反，只需在右子树中搜索即可；如果 $s(a_i)$ 等于根节点的关键值，则查找成功，搜索终止. 在子树中的查找与此类似.

装载物品后，如果箱子已满，转 Step 6；如果箱子未满，转 Step 5.

Step 5　插入. 如果箱子未被装满，则以箱子的剩余空间更新箱子容量，假设箱子的剩余空间为 e，则首先通过搜索二叉树来确定要插入值 e 的位置，如果搜索不成功，新元素插入到搜索的中断点，转 Step 7.

Step 6　删除. 如果箱子已被装满，则从树中删除最优箱子，分 3 种情况：

Case 1　要删除的节点 t 是树叶：丢弃树叶节点.

Case 2　要删除的节点 t 只有一个非空子树：如果 t 没有父节点，则将 t 丢弃，t 的惟一子树的节点成为新的搜索树的根节点；如果 t 有父节点 $node-f$，则修改 $node-f$ 的指针，使得 $node-f$ 指向 $node$ 的惟一孩子，然后删除节点 $node$.

Case 3　要删除一个左右子树都不为空的节点中的元素，将该元素替换为它的左子树中的最大元素或右子树中的最小元素.

Step 7　所有物品都被装载了吗，是，转 Step 8；否则转 Step 3.

Step 8　输出所用箱子个数、总容量及装箱结果，结束.

3　集装箱船预配优化模型与算法

3.1　同类箱组及装箱矩阵的定义

3.1.1　同类箱组　由满足以下 3 个原则的集装箱构成的集合称为同类箱组[6].

(1)集装箱内装载货物类型相同；

(2)集装箱尺度相同；

(3)货源港及目的港相同.

同类箱组中集装箱的重量可以不同.

3.1.2　装箱矩阵定义

$$\boldsymbol{T}=\begin{pmatrix} T_{12} & T_{13} & \cdots & T_{1N} \\ T_{22} & T_{23} & \cdots & T_{2N} \\ \vdots & \vdots & & \vdots \\ T_{N2} & T_{N3} & \cdots & T_{N-1,N} \end{pmatrix}= \begin{pmatrix} T_{12} & T_{13} & \cdots & T_{1N} \\ 0 & T_{23} & \cdots & T_{2N} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & T_{N-1,N} \end{pmatrix}$$

表示装箱矩阵. 式中：元素 T_{ij} 表示货源港为 i、目的港为 j 的集装箱个数，因为当 $i \geqslant j$ 时，$T_{ij}=0$，所以 $\boldsymbol{T}$ 是一个上三角矩阵.

3.2　集装箱船预配优化数学模型

3.2.1　优化总体目标及假设条件　对于预配主要考虑以下两个目标[6]：

(1)每个目的港的集装箱占用的 Bay 位数量越小越好，这样可以保证同类箱组尽量集中放置，可以减少装卸时岸边装卸桥移动的距离，提高装卸效率；

(2)倒箱数量最少(如果不同目的港的集装箱放在同一 Bay，则应保证目的港远的箱子在目的港近的箱子的下面).

优化模型的假设条件有以下几个方面：

(1)集装箱船物理结构已知，船上 Bay 位数量、每个 Bay 位中列数及层数即船上箱位信息已知；

(2)各个港口的集装箱信息已知，即各港口装卸集装箱数量、集装箱尺寸、货物类型及箱子重量等已知；

(3)船舶挂靠 P 个港口，第一个港口为初始港，在第一个港口所有箱位为空，在中间挂靠港口卸下本港口集装箱，并装载后续港口集装箱，第 P 个港口(最后一个港口)所有的集装箱均被卸下；

(4)假设所有集装箱在一个航次中都能被装

载,即不考虑装箱的选择问题;

(5)假设舱盖与 Bay 位是一一对应关系;

(6)集装箱尺寸均为同一尺度,不考虑危险品箱或冷藏箱.

3.2.2 参数及变量

$BayID=\{1,2,\cdots,n\}$,由船上 Bay 位标号组成的集合,n 为 Bay 位的数量;

$BayC=\{c_1,c_2,\cdots,c_n\}$,$c_i$ 为 $BayID(i)$ 容量,即 Bay 位中箱位的数量;

$BayX=\{X_1,X_2,\cdots,X_n\}$,$X_i$ 表示船上 $BayID(i)$ 的 X 坐标;

$$BayState=\begin{cases}p; & \text{Bay 位已装载目的港为 } p \text{ 的集装箱}\\ 1; & \text{Bay 位已装满}\\ 0; & \text{Bay 位为空}\end{cases} \tag{2}$$

表示 Bay 位装载状态;

$$x_{pij}=\begin{cases}1; & \text{在港口 } p,\text{同类箱组 } j \text{ 放在 } i\text{Bay}\\ 0; & \text{否则}\end{cases} \tag{3}$$

$$z_i=\begin{cases}0; & \text{Case 1}'\\ y_{pij}; & \text{Case 2}'\end{cases} \tag{4}$$

式中:y_{pij} 为在港口 p 第 iBay 中存放 j 组集装箱数量;z_i 表示当 s_{pj} 箱组放在 $BayID(i)$ 产生倒箱的个数,有两种情况:

Case 1′ 如果 $BayID(i)$ 中只装载一个目的港集装箱或如果不同目的港集装箱混装时,上面集装箱的目的港比下面集装箱的目的港近;

Case 2′ 当 $BayID(i)$ 中的集装箱是多个目的港混装时,目的港远的集装箱压住目的港近的集装箱的数量为 z_i.

3.2.3 目标函数

$$\min\begin{cases}\sum\limits_{p=1}^{P}\sum\limits_{i=1}^{n}\sum\limits_{j=p+1}^{P}x_{pij} & (5)\\ \sum\limits_{p=1}^{P}\sum\limits_{i=1}^{n}\sum\limits_{j=p+1}^{P}x_{pij}z_i & (6)\end{cases}$$

3.2.4 约束条件

$$\text{s.t.}\quad \sum_{j=1}^{P-1}\sum_{p=1}^{j}T_{pj}-\sum_{p=1}^{P-1}\sum_{j=p+1}^{P}T_{pj}\geqslant 0 \tag{7}$$

$$\sum b_m\leqslant\sum_{i=1}^{n}BayID(i) \tag{8}$$

$$\frac{1}{2}\left(l\cdot\Delta_0+\sum_{i=1}^{n}W_{pi}X_i-\Delta\cdot C\cdot L\right)\leqslant[M] \tag{9}$$

$$[t]\leqslant\frac{\sum\limits_{i=1}^{n}12W_{pi}b_i}{B\cdot L^2}\leqslant 0 \tag{10}$$

$$W_{pi}\leqslant[W_i] \tag{11}$$

$$i=1,2,\cdots,n;\quad p=1,2,\cdots,P$$

式(5)表示同一目的港的集装箱所占的 Bay 位数量越少越好,这个目标由最优匹配法箱子装载问题的算法保证.式(6)表示倒箱数越少越好,是优化的主要目标.式(7)表示船舶在满载情况下在某个港口卸下箱子的总数大于等于装载量,可以保证所有的箱子均被装载.式(8)表示调用箱子装载程序时,所用箱子数量(即 Bay 个数)不大于船舶上 Bay 的总和.式(9)表示船舶纵向强度约束,式中 l 表示船体前半体和后半体重量对船中剖面力矩和的相当力臂,$l=0.274\ 3L$,m;Δ_0 表示空船排水量,t;W_{pi} 表示第 iBay 在 p 港装卸集装箱的重量,t;X_i 表示 iBay 位重心到船中距的绝对值,m;Δ 表示船舶在计算状态下的排水量,$\Delta=\Delta_0+\sum\limits_{i=1}^{n}W_i$,t;$L$ 表示船长,m;C 为船体前半体和后半体浮力对船中剖面的力矩和的相当力臂系数,与船舶的方形系数有关[11].式(10)表示船舶首尾吃水差约束,原始定义为首吃水与尾吃水的差,此简化公式来源于文献[12],式中 B 表示船宽,m;b_i 表示 iBay 到船舶浮心的纵向距离,m;$[t]$表示许用吃水差,m.式(11)表示 Bay 位中集装箱总重量约束,$[W_i]$表示 iBay 位中总重量的许用值,由舱内和甲板上最大层叠重量与 Bay 位列数确定.

3.3 基于“超尺寸物品装箱算法”的集装箱船配载启发式算法

Step 1 初始化货物空间与装箱矩阵.

Step 2 对第一个港口的集装箱进行配载:

(1)将集装箱按目的港分成 P 个同类箱组子集,$p=1,2,\cdots,P$,P 为船舶挂靠港口数量;

(2)对这 P 个子集按目的港进行排序,将所有集装箱按目的港递减排序,即目的港最远的箱子排在第一,次远的排在第二,依次类推;将目的港最远的同类箱组看成是第一个待装物品,次远的看成是第二个,依次类推;

(3)调用超尺寸装箱算法,检查约束条件,对当前港口的集装箱进行装载.

Step 3　对于非第一个装载港，以当前港口卸载后的箱位状态为初始装载状态，更新 Bay 位容量、数量及状态参数，分析 $BayState$.

如果 $BayState=0$ 表示 Bay 位中没有集装箱，则同样将同类集装箱组按目的港远近递减排序，调用装箱算法进行装载，转 Step 4；

如果 $BayState=1$ 表示 Bay 位中已装满，无法进行装载，转 Step 4；

如果 $BayState=p$ 表示 Bay 位中有目的港为 p 的集装箱，以式(6)作为评价函数进行评价. 如果式(6)不为零，则记录目标函数的值，并继续寻找其他 Bay，比较不同 Bay 位装载时的目标函数值，使式(6)值最小的位置是最优装载位置.

Step 4　所有当前港口集装箱都装载完毕，转 Step 5；否则转 Step 3.

Step 5　输出配载结果，结束.

4　实例模拟与结果分析

假设某船舶在某航线上挂靠 6 个港口，船上有 8 个 Bay 位，每个 Bay 位中的箱位数量见表 1，模拟算例中没有考虑集装箱的重量约束. 如下产生满足要求的装箱矩阵：

$$E_1 = \sum_{i=1}^{n} c_i - \sum_{p=2}^{P} T_{1p} \tag{12}$$

$$E_p = E_{p-1} + \sum_{k=1}^{P-1} T_{kp} - \sum_{p=1}^{P} T_p;$$

$$p = 2,3,\cdots,P-1 \tag{13}$$

$$E_p \geqslant 0 \tag{14}$$

$$\boldsymbol{T} = \begin{pmatrix} T_{12} & T_{13} & T_{14} & T_{15} & T_{16} \\ & T_{23} & T_{24} & T_{25} & T_{26} \\ & & T_{34} & T_{35} & T_{36} \\ & & & T_{45} & T_{46} \\ & & & & T_{56} \end{pmatrix} = \begin{pmatrix} 116 & 89 & 38 & 61 & 32 \\ 0 & 59 & 41 & 34 & 57 \\ 0 & 0 & 62 & 35 & 38 \\ 0 & 0 & 0 & 48 & 55 \\ 0 & 0 & 0 & 0 & 129 \end{pmatrix}$$

表 1　“超尺寸物品装箱算法”获得的近似最优总布置配载结果

Tab. 1　Result of pre-stowage for containership based on over-sized items bin packing algorithm

Port	Bay 1 36	Bay 2 40	Bay 3 44	Bay 4 50	Bay 5 66	Bay 6 66	Bay 7 80	Bay 8 90
Port 1 装载后	32(1,6)	38(1,4)	40(1,2)+4(1,3)			61(1,5)	76(1,2)	85(1,3)
Port 2 卸载后	32(1,6)	38(1,4)	4(1,3)			61(1,5)		85(1,3)
Port 2 装载后	32(1,6)	38(1,4)	4(1,3)+34(2,5)	41(2,4)	57(2,6)	61(1,5)	59(2,3)	85(1,3)
Port 3 卸载后	32(1,6)	38(1,4)	34(2,5)	41(2,4)	57(2,6)	61(1,5)		
Port 3 装载后	32(1,6)	38(1,4)	34(2,5)	41(2,4)	57(2,6)	61(1,5)	38(3,6)+35(3,5)	62(3,4)
Port 4 卸载后	32(1,6)		34(2,5)		57(2,6)	61(1,5)	38(3,6)+35(3,5)	
Port 4 装载后	32(1,6)	40(4,5)	34(2,5)	48(4,5)	57(2,6)	61(1,5)	38(3,6)+35(3,5)	55(4,6)
Port 5 卸载后	32(1,6)				57(2,6)		38(3,6)	55(4,6)
Port 5 装载后	32(1,6)	14(5,6)		50(5,6)	57(2,6)	65(5,6)	38(3,6)	55(4,6)

注：表中 $A(B,C)$ 表示货源港为 B、目的港为 C 的集装箱数量为 A；$A_1(B_1,C_1)+A_2(B_2,C_2)$，表示此 Bay 位装载多个不同货源港与目的港的集装箱，$A_1(B_1,C_1)$ 先装(在下面)，$A_2(B_2,C_2)$ 后装(在上面)

装载模拟结果见表 1. 由模拟结果可知，此次航行挂靠 6 个港口，总装箱量为 894 个，有 2 个 Bay 位 5 个 Bay 次出现不同目的港集装箱混装的情形，混装 Bay 位率为 12.5%，混装 Bay 位均能满足后到港箱在下、先到港箱在上的要求，全航线倒箱数量为 0.

4　结　语

本文提出了一种集装箱船全航线配载问题分解求解策略，通过将问题分解可以降低问题求解的难度. 本文主要研究了预配子问题的优化模型与算法，预配时以最优装载及在各个港口装卸过程中倒箱数最少为目标，将相同尺寸、相同类型、相同目的港的集装箱组分配到船舶不同 Bay 位上，实现集装箱在船上的总布置配载. 装箱算法有比较成熟的启发算法，可以得到近似最优解. 实际模拟结果表明此方法可行，为集装箱全航线优化配载问题提供了一个实用的模型.

参考文献：

[1] AVRIEL M, PENN M, SHPIRER N, *et al*. Stowage planning for container ships to reduce the number of shifts [J]. **Annals of Operations Research**, 1998, **76**:55-71

[2] WILSON I, ROACH P. Principles combinatorial optimization applied to container-ship stowage planning [J]. **Journal of Heuristics**, 1999(5):403-418

[3] AVRIEL M, PENN M, SHPIRER N. Container ship stowage problem: complexity and connection to the coloring of circle graphs [J]. **Discrete Applied Mathematics**, 2000, **103**:271-279

[4] AVRIEL M, PENN M. Exact and approximate solutions of the container ship stowage problem [J]. **Computers and Industry Engineering**, 1993, **25**(1-4): 271-274

[5] 谢金星,邢文训. 现代优化算法[M]. 北京:清华大学出版社, 2000

[6] 张维英. 集装箱船全航线配载智能优化研究[D]. 大连:大连理工大学,2006

[7] 邢文训,陈 峰. 超尺寸物品装箱问题及其算法[J]. 应用数学学报, 2002, **25**(1):8-14

[8] XING Wen-xun. A bin packing problem with over-sized items [J]. **Operations Research Letters**, 2000, **30**:83-88

[9] SARTAJ S. **Data Structures, Algorithms, and Applications in C ++** [M]. Beijing: China Machine Press, 1999

[10] 侯识忠. 数据结构算法:Visual C ++ 程序集[M]. 北京:中国水利水电出版社,2005

[11] 李锡蔚. 集装箱船舶积载[M]. 北京:人民交通出版社, 1997

[12] AKIO I, KAZUYA S, ETSUKO N, *et al*. Multi objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks [J]. **European Journal of Operational Research**, 2006, **171**(2):373-389

Optimum model and algorithm of containership′s pre-stowage planning in full routes

ZHANG Wei-ying[*1,2], LIN Yan[1], JI Zhuo-shang[1], SUN Wen-zhi[2], YU Bao-chu[2]

(1. Ship CAD Engineering Centre, Dalian University of Technology, Dalian 116024, China;
2. College of Marine Engineering, Dalian Fisheries University, Dalian 116023, China)

Abstract: Containership stowage plan problem is an NP-hard problem. The problem is decomposed into two sub-problems, namely, pre-stowage plan and arranging containers of bays in order to reduce the computational complexity. Firstly, pre-stowage problem is regarded as packing problem, ship-bays on the board of vessel are regarded as bins, and the number of cells at each bay is taken as capacities of bins, containers with different characteristics (homogeneous containers group) are treated as items packed. At this stage, there are two objective functions: one is minimizing the number of bays packed by containers and the other is minimizing the number of rehandles. Secondly, containers assigned to each bays at first stage are allocated to special slot, and the objective functions are to minimize the metacentric height, heel and rehandles. The taboo search heuristics algorithm is used to solve the sub-problem. The main focus is on the first sub-problem. A practical case certifies the feasibility of the model and the algorithm.

Key words: containership; pre-stowage; bin packing problem; binary search tree; full routes