Modeling and solution for the ship stowage planning problem of coils in the steel industry

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Citation: TANG, L. ...et al., 2015. Modeling and solution for the ship stowage planning problem of coils in the steel industry. Naval Research Logistics, 62(7), pp. 564-581.

Additional Information:

• This is the peer reviewed version of the following article: TANG, L. ...et al., 2015. Modeling and solution for the ship stowage planning problem of coils in the steel industry. Naval Research Logistics, 62(7), pp. 564-581., which has been published in final form at http://dx.doi.org/10.1002/nav.21664. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Self-Archiving.

Metadata Record: https://dspace.lboro.ac.uk/2134/21674

Version: Accepted for publication

Publisher: © Wiley

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
Modeling and solution for the ship stowage planning problem of coils in steel industry

Lixin Tang\textsuperscript{1*}, Jiyin Liu\textsuperscript{2}, Fei Yang\textsuperscript{1}, Feng Li\textsuperscript{1}, Kun Li\textsuperscript{1}

\textsuperscript{1}The Institute of Industrial Engineering and Logistics Optimization, Liaoning Key Laboratory of Manufacturing System and Logistics, Northeastern University, Shenyang, China

\textsuperscript{2}School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, UK

This paper has been published in \textit{Naval Research Logistics}. It can be cited as

\*Corresponding author. Tel: +86-24-83680169; Fax: +86-24-83680169; E-mail: qhjytlx@mail.neu.edu.cn
Abstract:

We consider a ship stowage planning problem (SSPP) where steel coils with known destination ports are to be loaded onto a ship. The coils are to be stowed on the ship in rows. Due to their heavy weight and cylindrical shape, coils can be stowed in at most two levels. Different from stowage problems in previous studies, in this problem there are no fixed positions on the ship for the coils due to their different sizes. At a destination port, if a coil to be unloaded is not at a top position, those blocking it need to be shuffled. In addition, the stability of ship has to be maintained after unloading at each destination port. The objective for the stowage planning problem is to minimize a combination of ship instability throughout the entire voyage, the shuffles needed for unloading at the destination ports, and the dispersion of coils to be unloaded at the same destination port. We formulate the problem as a novel mixed integer linear programming model. Several valid inequalities are derived to help reducing solution time. A tabu search (TS) algorithm is developed for the problem with the initial solution generated using a construction heuristic. To evaluate the proposed TS algorithm, numerical experiments are carried out on problem instances of three different scales by comparing it with a model-based decomposition heuristic, the classic tabu search algorithm, the particle swarm optimization algorithm, and the manual method used in practice. The results show that for small problems, the proposed algorithm can generate optimal solutions. For medium and large practical problems, the proposed algorithm outperforms other methods.

Key Words: stowage planning, steel coils, integer programming, tabu search.
1. Introduction

Steel products are widely used in construction and in producing other products such as automobiles. Due to the large volume and heavy tonnage, steel products are often transported by ship wherever possible. From one of the steel plants of Baosteel where this study is conducted, over three million tons of steel coils are transported annually to customers at different destination ports, while seventy percent by ship. Figure 1 shows a ship being loaded with steel coils. At a destination, if a coil to be unloaded is not on top, those above them must be shuffled. Moreover, the stability of the ship has to be maintained throughout the whole voyage. To keep ship stability and to avoid unnecessary shuffling, effective ship stowage planning is critical.

![Steel coils are being loaded onto a ship](image)

Figure 1  Steel coils are being loaded onto a ship

For each arriving ship, the steel company needs to make a loading plan. The plan is made in two phases. The first phase, called the ship consolidation planning problem (SCPP), is to select steel coils to be loaded onto the ship considering the destinations of the ship, the quantities and due dates of customer orders, and the positions of the coils in the storage yard. The second phase is to plan the stowage of the coils on the ship, called the ship stowage
planning problem (SSPP). In this paper, we study the second-phase problem, i.e., the ship stowage planning problem. The study is conducted for a steel factory which owns a port terminal to conduct product delivery. The loading capacities of the smallest and the biggest ships are about 200 tons and 2000 tons, respectively. The average weight of a coil is about 16 tons. Therefore, the number of coils that one ship can carry ranges from 12 to 120. Meanwhile, the most common ships have loading capacities that range from 600 tons to 1200 tons.

Given a set of steel coils to be loaded onto a ship, the problem is to determine the location of each coil on the ship, considering the sizes of the coils, stability of the ship and convenience of unloading at the destination ports. In current practice, a rough stowage plan is made manually based on the planner’s experiences. Due to the large problem size in practice, such a manual plan often results in the need for a large number of shuffles at destination ports. A more effective stowage planning method is needed.

There has been little research on the ship stowage planning problem for steel coils. Our literature survey only found one previous study on the problem. Umeda et al. [19] studied a ship stowage planning problem of steel products including coils. The problem was considered as a three-dimensional allocation problem and solved using simulated annealing. In determining the stowing-positions of the products, the weight balance of the ship, the loading ratio and the work efficiency of stowing were considered. The ship considered in the study had fixed slots for stowing the products. In practice, the sizes of coils are different and there are no fixed positions on the ship. In our problem, the loading positions on the ship are unfixed and the coordinates of coils on the ship need to be determined.

The stacking method in coil yards and warehouses are similar to that for coil stowage on ships. Tang et al. [18] studied a coil shuffling problem (CSP) in a warehouse served by a crane. Considering the practical stacking and shuffling features of CSP, a linear integer programming model was formulated. Minimizing the logistics cost for shuffling operations is taken as the objective. Jang and Kim [9] studied a pyramid stacking system with multiple groups of cylindrical units in the warehouse. They established a mathematical model to optimize the re-handling cost and the space cost of the system in three different cases. In their instances, the maximum number of tiers exceeds 2. However, these problems do not
have ship stability issue, since the coils are stacked on the ground.

Most other papers focus on stowage planning on container ship and stacking problems in container terminal yards. For example, Avriel et al. [2] studied a stowage planning problem to minimize the number of container shifts needed for unloading them at destination port. Some constraints such as stability of the ship were not considered. Ambrosino et al. [1] addressed a stowage planning problem considering constraints such as container dimensions, weights and destinations as well as the balance of the ship. The objective was to minimize the total stowage time according to the positions of stowed containers on board. The problem was formulated as a binary integer programming model and solved using a three stage approach. Through preprocessing and prestowage in the first two stages, the search space was reduced. By further relaxing some constraints the reduced model was then solved in the final stage. Sciomachen and Tanfani [14] related the stowage planning problem with the three-dimensional bin packing problem. They considered the container stowage and quay crane assignment together and proposed a heuristic solution method to optimize crane productivity. Imai et al. [8] determined the container stowage and loading plans of a ship considering stacking configurations of containers in the yard to maximize ship stability and minimize the number of container shuffles during the loading process. A genetic algorithm was proposed to solve the problem and to obtain a set of non-dominated solutions. Álvarez [3] proposed an approach using tabu search and multi-start techniques to generate vessel loading plans in reach-stacker based terminals, considering the vessel stowage requirements and the container stacking information in the yard as inputs. Tang et al. [15] investigated the container stacking and reshuffling issues in a container terminal. Five effective heuristics were proposed and a discrete event simulation model is developed to animate the stacking, retrieving, and reshuffling operations in both static and dynamic environments.

There are significant differences between the stowage planning of steel coils and that of containers. Containers are of standard sizes and the containers of the same size are stacked one on top of another on the ship. Steel coils, on the other hand, have different weights, diameters and widths. Due to the cylindrical shape, the coils cannot be stowed in the same way as containers. In addition, ship stability in the container stowage planning studies was considered only in terms of the total weights in different parts of the ship rather than the more
precise moments. For steel coils, due to the heavy weight and the different sizes, only considering the total weights is far from being accurate. Only considering the weight balance of the ship may cause the situation where some area of the ship is full of coils of the largest weight and another area is full of coils of the smallest weight. Although the ship may not sink during the voyage, the service life of the ship will be reduced substantially.

More recently, Hvattum et al. [7] studied a tank allocation problem for bulk cargo shipping. The decisions in the problem included ship routing as well as allocation of loads to tanks on the ship with the constraint that each tank could only hold one load at any time. Due to the nature of the cargo considered, there were no stacking and shuffling issues in their problem. They considered ship stability and strength in a comprehensive way, including balances in fore-stern, left-right as well as diagonal dimensions. Because the positions of the tanks were fixed on the ship, it was convenient to express the moment contributions of the loads in the tanks in each dimension. Øvstebø et al. [13] studied a stowage problem for RoRo ships. The problem was to decide which cargoes to carry, how many vehicles of each cargo to carry, and how to stow vehicles on board a RoRo ship with a given voyage route. The objective was to maximize the revenue from optional cargoes minus penalty costs incurred when having to shuffle cargoes. The loading and unloading method of the RoRo ship determined that the shuffling was in horizontal direction rather than vertical direction. As the vehicles were stowed in lanes, the stowage shared some similarities with container stowage, which was different from the coils stowage. They also considered ship stability constraints using moments, though the constraints in the fore-stern direction was not explicitly listed. However, while the width of each lane of the vehicles on board was a variable, the distance from each lane to the centre of the deck was estimated using a width of equal lanes and a potential number of lanes.

The rest of this paper is organized as follows. In the next section, the stowage planning problem is described in detail and formulated as an integer programming model. Due to the complex loading constraints, it takes too long to solve the model for practical problems. Therefore, we derive a number of valid inequalities for the model, and propose a decomposition heuristic to solve the model approximately in Section 3. To further effectively solve the problem, a tabu search (TS) algorithm is developed in Section 4 to get an
efficient and superior stowage plan, with the initial solution obtained by a constructed heuristic. Computational results are reported in Section 5 to demonstrate the effectiveness of the proposed TS algorithm. Section 6 concludes the paper.

2. The ship stowage planning problem and model

2.1 Problem description

A given set of steel coils with known destination ports are to be loaded to a ship. The diameter, width, and weight of each coil are known. Considering the practicality of stacking, the coils are to be stored in the ship in the way shown in Figure 2. Due to the heavy weight of the coils and for stability, at most two layers of coils can be stored on the ship. Clearly a coil cannot be loaded to an upper-layer position unless the two lower-layer positions underneath it are occupied by other coils. As a result, the maximum number of coils that can be loaded to the upper layer of a row is one fewer than that to the lower layer of the same row. Due to the different sizes of the coils, there are no fixed locations on the ship for the coils. The ship stowage planning problem is to determine the coordinates of loading locations of the coils on the ship.

![Figure 2 Measurements and stacking of coils](image)

In practice, steel coils can be stowed on a ship in the stowage pattern described below. Coils are placed in rows across the length of the ship as shown in Figure 3. For convenience in description, we refer to the direction along the length of the ship as the horizontal direction, and the direction along the width of the ship as the vertical direction. Accordingly, the way of stowing coils shown in Figure 3 is called the horizontal stowage pattern. Coils at the lower layer in each row are placed from the stern to the fore, and no space between adjacent coils is allowed in the initial stowage. When the lower layer is full
of coils, additional coils can be loaded to the upper layer. To ensure stability and avoid
damage, both the width and the diameter (and hence the weight as well) of a coil at the upper
layer cannot exceed those of any of the two coils underneath it. The sum of the diameters of
the coils at the lower layer of a row cannot exceed the length of the stowage area of the ship.
The width of a row is defined as the maximum width of all the coils in the row. In the
vertical direction, there must be a safety gap between any two adjacent rows and between a
side row and the edge of the stowage area. The sum of the widths of all rows and these gaps
must not exceed the width of the stowage area.

![Figure 3](image)

Figure 3  Horizontal stowage pattern on a ship

The loaded ship is to visit a number of destination ports in a given sequence delivering
the coils. Each coil on board is to be unloaded at a specified destination port. Nothing is
loaded onto the ship at these destination ports. At a destination port, if a coil to be unloaded
is at the upper layer or at the lower layer without any coil above it, then the coil can be
unloaded directly. When a coil to be unloaded is at the lower layer and there is any coil with
a later destination above it, then the coil above needs to be shuffled in order to unload the
target coil. The shuffled coil will be placed back to the lower layer position left by the
unloaded coil. It can be seen from Figure 2 that there may be two coils blocking the target
coil. In this case both blocking coils need to be shuffled. After the target coil is unloaded,
the larger blocking coil (both in terms of width and in terms of diameter) will be put in the
position left by the unloaded coils and the other will be placed back to its original position.

Stability of the ship needs to be considered for the entire voyage. Horizontal and
vertical central lines of the stowage area divide the area into four parts. In most previous
studies, ship stability was considered by balancing the weights on the two sides of each
central line. However even with perfect weight balance among the four parts, placing the
coils differently within a part may affect the stability of the ship. A better way to consider ship stability is to balance the moments in both horizontal and vertical directions. There has been only one recent study that tried to calculate moments but using estimated distances. In this paper we introduce a new method for calculating the moments of the coils with precise distances. We take horizontal direction as an example to describe the calculation. The horizontal moment contribution of a coil is defined as its weight times its distance to the vertical central line. The distance can be computed by subtracting the half length of the stowage area from the horizontal coordinate of the coil. Note that the distances and thus the moments can be positive or negative. If the sum of the horizontal moment contributions of all coils equals zero, the ship is in perfect balance in horizontal direction. Otherwise the absolute value of this sum is defined as horizontal moment imbalance which cannot exceed certain limit to ensure safety. Similarly, vertical moment imbalance can be calculated and should be restricted within certain limit.

Considering the balance of the ship, once the stowage plan is decided, the order for physically loading them onto the ship is determined according to their positions in the stowage plan and the standard loading procedures. Therefore, we do not need to consider the physical loading operations in this paper.

Proper stowage planning of coils helps guaranteeing safety during the voyage as well as enhancing the loading efficiency. Considering the stowage requirement and characteristics of the problem, the objective to be minimized includes the following elements.

1) the moment imbalance of the ship

The ship has to keep balance during the voyage. Therefore, the moment imbalances of the ship in *horizontal* and *vertical* directions should be minimized for the loaded ship at the original port as well as for the ship after the unloading at each destination port. As described before, after unloading a coil, any blocking coil is either placed back to its original position or relocated to the position left by the unloaded coils. The horizontal coordinate of the blocking coil may change at most half of a coil diameter while its vertical coordinate will keep unchanged. We ignore such small change, and therefore the moment contributions of any coil will not be changed in the voyage until it is unloaded.

2) the number of shuffles
As described earlier, when unloading a lower-layer coil at a destination port, if there is any coil with a later destination above it, then the coil above needs to be shuffled. The number of coils that need to be shuffled will be called the number of shuffles for short. Minimizing the number of shuffles will enhance the efficiency of the unloading operations and reduce damages to the coils.

3) the dispersion of coils for the same destination

Placing the coils with the same destination close to each other on the ship can help to enhance the efficiency of unloading.

The ship stowage problem is to determine the locations of a given set of steel coils on a given ship considering the coil information and the required stowage pattern of the ship, and to minimize the objective function consisting of the above elements.

2.2 Notation

We define the following notation for modeling our problem.

Known parameters:

- $C$ the set of coils to be loaded onto the ship, $C = \{c_1, c_2, ..., c_n\}$, where $n$ is the number of the coils. We will also refer to coil $c_i$ as coil $i$ when it does not cause confusion.
- $Q^h_0$ the maximum allowed moment imbalance in horizontal direction.
- $Q^v_0$ the maximum allowed moment imbalance in vertical direction.
- $S_{iw}$ Parameters of coil $c_i$, $v = \{1, 2, 3\}$. $S_{i1}$, $S_{i2}$, and $S_{i3}$ denote the width, diameter and weight of coil $c_i$, respectively.
- $p_i$ the destination number of coil $c_i$, $p_i = \{1, ..., d, ..., P\}$. A smaller number means that the ship visits the destination earlier.
- $P$ the number of the destinations of the voyage.
- $D$ the minimum space between adjacent rows to allow the quay crane loading and unloading coils.
- $W$ the width of the storage area.
- $L$ the length of the storage area.
- $M$ a very large positive real number.
- $\alpha_1$ the coefficient of moment imbalance cost.
\( \alpha_2 \) the coefficient of unit shuffle cost.

\( \alpha_3 \) the coefficient of unit dispersion cost.

Because the stowage locations on the ship are not fixed, we want the model to decide the number of rows that the coils should be stored in and the number of coils stored at the lower layer of each row. To build the model we need to identify an upper bound for the number of rows, \( m \), and an upper bound for the number of coils at the lower layer of a row, \( g \). These upper bounds can be calculated as follows.

From \( (m+1)D + \max \{S_i \mid i = 1, \ldots, n\} + (m-1)\min \{S_i \mid i = 1, \ldots, n\} \leq W \), we can obtain

\[
m = \left\lfloor \frac{W - \max \{S_i \mid i = 1, \ldots, n\} + \min \{S_i \mid i = 1, \ldots, n\} - D}{D + \min \{S_i \mid i = 1, \ldots, n\}} \right\rfloor. \tag{1}\]

Re-order the coils such that \( S_{i|2} \leq S_{j|2} \leq \cdots S_{n|2} \), then \( g \) can be determined by

\[
\sum_{i=1}^{g} S_{i|2} \leq L, \quad \sum_{i=1}^{g+1} S_{i|2} > L, \quad \text{for each row } k = 1, \ldots, m. \tag{2}\]

Because of the different sizes of the coils, the actual coordinates of the coils need to be calculated accordingly in order to calculate their moment contributions. For this purpose, we define the left-stern corner as the origin for the coordinates of the locations of coils. For a coil \( c_i \) in the \( j \)th position at the lower layer of a row, as the coils are placed from stern to fore without gaps, the horizontal coordinate \( c_{x_i} \) of this coil can be expressed as the sum of the diameters of the coils placed in positions 1 to \( j-1 \) at the lower layer in the same row plus half of the diameter of this coil itself. Similarly, for a coil in the \( j \)th position at the upper layer of a row, because it is above the middle point between two coils in positions \( j \) and \( j+1 \) at the lower layer, its horizontal coordinate can be considered as the sum of the diameters of the coils in positions 1 to \( j \) at the lower layer of the same row. Therefore, the distances \( dx_i (dy_i) \) for calculating horizontal (vertical) moment contribution of coil \( c_i \) can be obtained by subtracting \( L/2 \) \( (W/2) \) from the horizontal (vertical) coordinate \( c_{x_i} (c_{y_i}) \) as shown in Figure 4.
Decision variables:

\[ x_{ijkl} = \begin{cases} 
1 & \text{if coil } i \text{ is located to location } j \text{ of layer } l \text{ in row } k \text{ of the ship} \\
0 & \text{otherwise} 
\end{cases} \]

Note that the rows on the ship are numbered from left to right and the locations are numbered from stern to fore as shown in Figure 3. Therefore, the coil \( c_i \) is placed at location 10 of row 3. The lower layer is defined as layer 0 and the upper layer is defined as layer 1. In addition, we also use \( jk0 \) and \( jk1 \) to denote the lower-layer and upper-layer position \( j \) in the row \( k \), respectively.

\( S_{\text{max}1,k} \) = the maximum width of coils in row \( k \), \( k=1, \ldots, m \).

\( y_k \) = the vertical coordinate of row \( k \), \( k=1, \ldots, m \).

For modeling convenience, we define the following constants: \( x_{i1,m+1,0} = 0 \), \( y_0 = 0 \), \( y_{m+1} = W - D \), \( S_{\text{max}1,0} = 0 \), \( S_{\text{max}1,m+1} = 0 \).

\[ u_{jkhd} = \begin{cases} 
1 & \text{if the upper coil at location } j \text{ of row } k \text{ is relocated to location } j \text{ at the lower-layer of row } k \text{ after reaching destination } d \text{ while } h=0, \\
\text{or the upper coil at location } j \text{ of row } k \text{ is relocated to location } j+1 \text{ at the lower-layer of row } k \text{ after reaching destination } d \text{ while } h=1 \\
0 & \text{otherwise} 
\end{cases} \]

\[ u_{z_{jk0}} = \begin{cases} 
1 & \text{if there is a coil in location } j \text{ at the upper-layer of row } k \text{ and it is still placed at the same position after reaching destination } d \\
0 & \text{otherwise} 
\end{cases} \]

\[ z_{jk0} = \begin{cases} 
1 & \text{if the upper-layer coil in location } j \text{ of row } k \text{ has a destination number larger than that of the lower-layer coil in location } j \text{ of row } k \text{ while } h=0, \\
\text{or the upper-layer coil in location } j \text{ of row } k \text{ has a destination number larger than that of the lower-layer coil in location } j+1 \text{ of row } k \text{ while } h=1 \\
0 & \text{otherwise} 
\end{cases} \]
If the upper-layer coil in location $j$ of row $k$ has a destination number larger than that of any coil underneath it:

$$z_{jk}^1 = \begin{cases} 1 & \text{if } j_{jk1} \text{ is unloaded, the neighbouring upper-layer coil is first shuffled to this lower-layer position and when that coil is unloaded, the coil at location $jk1$ is then shuffled to this position} \\ 0 & \text{otherwise} \end{cases}$$

If after the two coils below the coil at location $jk1$ are unloaded at different ports, the two neighbouring upper-layer coils are shuffled to these lower-layer positions respectively while the coil at location $jk1$ is still placed at its own position:

$$z_{jk}^3 = \begin{cases} 1 & \text{if the upper-layer coil at location } j \text{ of row } k \text{ has a destination different from that of the lower-layer coil at location } j \text{ of row } h \text{ while } h=0, \text{ or the upper-layer coil at location } j \text{ of row } k \text{ has a destination different from that of the lower-layer coil at location } j+1 \text{ of row } h \text{ while } h=1 \\ 0 & \text{otherwise} \end{cases}$$

If a coil needs to be shuffled during the voyage, $z_{jk}^1$ indicates the first shuffle needed. $z_{jk}^2$ and $z_{jk}^3$ indicate the second and the third shuffles needed respectively, if there is any.

If the lower-layer coil at location $j$ of row $k$ has a destination number different from that of the coil at location $j+1$ of the same layer in row $k$:

$$z_{jk}^4 = \begin{cases} 1 & \text{if } \text{the contribution of coil } i \text{ to the moment in horizontal direction of the ship just before reaching destination } d \\ 0 & \text{otherwise} \end{cases}$$

$$H_{idh}^1 = \text{the contribution of coil } i \text{ to the moment in vertical direction of the ship just before reaching destination } d$$

2.3 Optimization model

Using the above notation, the model of the problem can be presented.

Minimize
\[ \alpha \left( \sum_{d=1}^{P} Q_d^1 + \sum_{d=1}^{P} Q_d^2 \right) + \alpha_2 \left( \sum_{k=1}^{m} \sum_{l=1}^{n} z_{l,k}^1 + \sum_{k=1}^{m} \sum_{l=1}^{n} z_{l,k}^2 + \sum_{k=1}^{m} \sum_{l=1}^{n} z_{l,k}^3 \right) + \alpha_3 \left( \sum_{k=1}^{m} \sum_{l=1}^{n} z_{l,k}^4 + \sum_{k=0}^{m} \sum_{l=1}^{n} z_{l,k}^5 \right) \]  

Subject to

\[
\sum_{i=0}^{1} \sum_{k=1}^{m} \sum_{l=1}^{n} x_{ijkl} = 1 \quad i = 1, \ldots, n
\]  

\[
\sum_{j=1}^{n} x_{ijkl} \leq 1, \quad k=1, \ldots, m, \quad j = 1, \ldots, g - l, \quad l=0,1
\]  

\[
\sum_{j=1}^{n} x_{ijkl} \geq \sum_{j=1}^{n} x_{ijkl}, \quad k=1, \ldots, m, \quad j = 1, \ldots, g - 1
\]  

\[
\sum_{j=1}^{n} \sum_{i=1}^{n} x_{ijkl} S_{ij} \leq L, \quad k=1, \ldots, m
\]  

\[
\sum_{j=1}^{n} \sum_{i=1}^{n} x_{ijkl} S_{ijn} \leq \sum_{j=1}^{n} x_{ijkl} S_{ijn}, \quad k=1, \ldots, m, \quad j = 1, \ldots, g - 1, \quad l=0,1, \quad v=1,2
\]  

\[
\sum_{i=1}^{n} S_{ijn} x_{ijkl} \leq S_{\text{max},ik}, \quad k=1, \ldots, m, \quad j = 1, \ldots, g
\]  

\[
Y_{i-1} + S_{\text{max},ik} / 2 + D \sum_{i=1}^{n} x_{i,ik,0} + S_{\text{max},ik} / 2 \leq y_i, \quad k=1, \ldots, m + 1
\]  

\[
H_d' \geq S_{i,j} \left[ \left( 1 - l \right) \left( \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijkl} S_{ij} + S_{ij} - \frac{L}{2} \right) + \left( 1 - l \right) \left( \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijkl} S_{ijn} + S_{ijn} - \frac{L}{2} \right) \right] + M \left( x_{ijkl} - 1 \right),
\]  

\[
i = 1, \ldots, n, \quad k=1, \ldots, m, \quad l=0,1, \quad j = 1, \ldots, g - l, \quad d=1,\ldots,P, \quad \text{if } p_i \geq d
\]  

\[
H_d' \leq S_{i,j} \left[ \left( 1 - l \right) \left( \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijkl} S_{ij} + S_{ij} - \frac{L}{2} \right) + \left( 1 - l \right) \left( \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijkl} S_{ijn} + S_{ijn} - \frac{L}{2} \right) \right] + M \left( 1 - x_{ijkl} \right),
\]  

\[
i = 1, \ldots, n, \quad k=1, \ldots, m, \quad l=0,1, \quad j = 1, \ldots, g - l, \quad d=1,\ldots,P, \quad \text{if } p_i \geq d
\]  

\[
H_d' = 0, \quad i = 1, \ldots, n, \quad d=1,\ldots,P, \quad \text{if } p_i \geq d
\]  

\[
H_d' \geq S_{i,j} \left( y_j - \frac{W}{2} \right) + M \left( \sum_{i=0}^{1} \sum_{j=1}^{n} x_{ijkl} - 1 \right),
\]
\[ i = 1, \ldots, n, \ k = 1, \ldots, m, \ d = 1, \ldots, P, \ \text{if } p_i \geq d \]

\[ H_{ad}^2 \leq S_{hi} \left( y_k - \frac{W}{2} \right) + M (1 - \sum_{j=0}^{1} \sum_{i=1}^{x_{ijkl}}), \]

\[ i = 1, \ldots, n, \ k = 1, \ldots, m, \ d = 1, \ldots, P, \ \text{if } p_i \geq d \]

\[ H_{ad}^2 = 0, \]

\[ Q_d^i \geq \sum_{i=1}^{k} H_{ad}^i, \]

\[ d = 1, \ldots, P \]

\[ Q_d^i \geq -\sum_{i=1}^{k} H_{ad}^i, \]

\[ d = 1, \ldots, P \]

\[ Q_d^i \geq \sum_{i=1}^{k} H_{ad}^i, \]

\[ d = 1, \ldots, P \]

\[ Q_d^i \geq -\sum_{i=1}^{k} H_{ad}^i, \]

\[ d = 1, \ldots, P \]

\[ Q_d^i \leq Q_0^i, \]

\[ d = 1, \ldots, P \]

\[ z_{hjk}^0 \geq \left( \sum_{i=1}^{n} p_i x_{ijk} \right) / P, \]

\[ k = 1, \ldots, m, \ h = 0,1, \ j = 1, \ldots, g - h - 1 \]

\[ z_{hjk}^0 \leq z_{jk}^i, \]

\[ k = 1, \ldots, m, \ h = 0,1, \ j = 1, \ldots, g - h - 1 \]

\[ \left( d + 1 - \sum_{i=1}^{n} p_i x_{jk1} \right) / (P + 1) \leq 1 - u_{hijd}, \]

\[ k = 1, \ldots, m, \ j = 1, \ldots, g - 1, \ h = 0,1,2, \ d = 1, \ldots, P \]

\[ \sum_{i=1}^{n} p_i x_{ijk0} - d \right) / P \leq 1 - u_{hijd}, \]

\[ k = 1, \ldots, m, \ j = 1, \ldots, g, \ d = 1, \ldots, P \]

\[ \sum_{i=1}^{n} p_i x_{ijkl0} - d \right) / P \leq 1 - u_{ijkl}, \]

\[ k = 1, \ldots, m, \ j = 1, \ldots, g - 1, \ d = 1, \ldots, P \]

\[ u_{hijd} + u_{ijkd} + u_{zjkl} \leq 1, \]

\[ k = 1, \ldots, m, \ j = 1, \ldots, g - 1, \ d = 1, \ldots, P \]
\begin{align*}
u_{0,jkd} + u_{i,j-1,kd} &\leq 1, \quad k = 1, \ldots, m, \quad j = 1, \ldots, g - 1, \quad d = 1, \ldots, P \quad (29) \\
\left(\sum_{i=1}^{\alpha} p_i x_{jk1} - d\right)/P &\leq u_{0,jkd} + u_{1,jkd} + u_{2,jkd}, \\
\quad k = 1, \ldots, m, \quad j = 1, \ldots, g - 1, \quad d = 1, \ldots, P \quad (30) \\
\left(\sum_{i=1}^{\alpha} p_i x_{jk0} - d - 1\right)/(P+1) + 1 + u_{i,j-1,kd} + u_{0,jkd} &\geq u_{2,j-1,kd}, \\
\quad k = 1, \ldots, m, \quad j = 1, \ldots, g, \quad d = 1, \ldots, P \quad (31) \\
\left(\sum_{i=1}^{\alpha} p_i x_{jk0} - d - 1\right)/(P+1) + 1 + u_{i,j-1,kd} + u_{0,jkd} &\geq u_{2,jkd}, \\
\quad k = 1, \ldots, m, \quad j = 1, \ldots, g, \quad d = 1, \ldots, P \quad (32) \\

u_{1,0,kd} = 0, \quad u_{2,0,kd} = 0, \quad u_{0,jkd} = 0, \quad u_{2,jkd} = 0, \quad k = 1, \ldots, m, \quad d = 1, \ldots, P \quad (34) \\

M \left(u_{2,j-1,kd} + u_{0,jkd} - 2\right) + \sum_{i=1}^{\alpha} S^v_i x_{i,j-1,k1} &\leq \sum_{i=1}^{\alpha} S^v_i x_{i,j,k1}, \\
\quad k = 1, \ldots, m, \quad j = 1, \ldots, P, \quad v = 1, 2 \quad (35) \\

M \left(u_{2,jkd} + u_{1,j-1,kd} - 2\right) + \sum_{i=1}^{\alpha} S^v_i x_{i,j,k1} &\leq \sum_{i=1}^{\alpha} S^v_i x_{i,j-1,k1}, \\
\quad k = 1, \ldots, m, \quad j = 1, \ldots, P, \quad v = 1, 2 \quad (36) \\
u_{i,j-1,kd} + u_{0,jkd, v+1} - 1 &\leq z_{jk}^2, \quad k = 1, \ldots, m, \quad j = 2, \ldots, g - 1, \quad d = 1, \ldots, P - 1 \quad (37) \\
u_{0,j+1,kd} + u_{1,jkd, v+1} - 1 &\leq z_{jk}^2, \quad k = 1, \ldots, m, \quad j = 1, \ldots, g - 2, \quad d = 1, \ldots, P - 1 \quad (38)
\end{align*}
\[
\left( \sum_{i=1}^{n} p_i x_{ijk0} - \sum_{i=1}^{n} p_i x_{i,j+1,k0} \right)/P + u_{0,j+1,kd} + u_{1,j-1,kd} + u_{2,jkd} - 3 \leq z_{jk}^3, \\
k = 1, ..., m, \ j = 2, ..., g - 2, \ d = 1, ..., P
\]

(39)

\[
\left( \sum_{i=1}^{n} p_i x_{i,j+1,k0} - \sum_{i=1}^{n} p_i x_{ijk0} \right)/P + u_{0,j+1,kd} + u_{1,j-1,kd} + u_{2,jkd} - 3 \leq z_{jk}^3, \\
k = 1, ..., m, \ j = 2, ..., g - 2, \ d = 1, ..., P
\]

(40)

\[
M \left( 1 - \sum_{i=1}^{n} x_{i,j+1,k,0} \right) + z_{jk}^4 \geq \left( \sum_{i=1}^{n} x_{i,j+h,k,0} \times p_i - \sum_{i=1}^{n} x_{i,j+h,k,0} \times p_i \right)/P, \\
k = 1, ..., m, \ j = 1, ..., g - 1, \ h = 0, 1
\]

(41)

\[
M \left( 1 - \sum_{i=1}^{n} x_{ijk,1} \right) + z_{jk}^5 \geq \left( \sum_{i=1}^{n} x_{ijk,1} \times p_i - \sum_{i=1}^{n} x_{ijk,1} \times p_i \right)/P, \\
k = 1, ..., m, \ j = 1, ..., g - 1, \ h = 0, 1
\]

(42)

\[
M \left( 1 - \sum_{i=1}^{n} x_{ijk,1} \right) + z_{jk}^6 \geq \left( \sum_{i=1}^{n} x_{ijk,1} \times p_i - \sum_{i=1}^{n} x_{ijk,1} \times p_i \right)/P, \\
k = 1, ..., m, \ j = 1, ..., g - 1, \ h = 0, 1
\]

(43)

\[
z_{jk}^0, z_{jk}^1, z_{jk}^2, z_{jk}^3, z_{jk}^4, u_{0,jk}, u_{2,jk} \in \{0, 1\}, \ h = 0, 1, \ k = 1, ..., m, \ j = 1, ..., g - 1
\]

(44)

\[
x_{ijk} \in \{0, 1\}, \quad i = 1, ..., n, \ j = 1, ..., g, \ k = 1, ..., m, \ l = 0, 1
\]

(45)

The objective of the model is to minimize the weighted sum of three elements. The three terms in the objective function (3) represent the total cost of ship imbalance during the voyage, the total cost of shuffles, and the total cost of the dispersion of coils of each destination. Note that \( z_{jk} \) counts the first shuffle needed for the coil at the upper-level location \( jk1 \). \( z_{jk}^2 \) and \( z_{jk}^3 \) count additional shuffles needed for the coil after that.

Constraints (4) and (5) guarantee that each coil can be placed on only one location and each location can be occupied by no more than one coil. Constraints (6) require that coils at the lower layer in each row are placed from the stern to the fore with no space between adjacent coils. Constraints (7) ensure that the sum of diameters of coils at the lower layer in each row cannot exceed the length of the ship. According to the practical operation
requirements, constraints (8) ensure that the width and diameter of a coil at the upper layer cannot exceed those of any of the two coils underneath it. The vertical coordinates of the rows can be expressed using constraints (9) and (10). Constraints (9) ensure that the width of any coil in a row is not greater than the maximum width of the row. Constraints (10) set restrictions on the vertical coordinates of the rows so that the gap between any adjacent rows of coils is at least the required minimum \( D \). When \( k=1 \) and \( k=m+1 \), Constraints (10) ensure that the coils in the first and last rows keep at least the minimum distance \( D \) from the left and right sides of the ship, respectively.

Constraints (11)-(13) calculate the horizontal moment contribution of each coil \( i \) before each destination \( d \), \( H^i_d \). If coil \( i \) is placed in position \( j \) at layer \( l \) of row \( k \) (\( x_{ijkl}=1 \)) and if it is still on the ship during the voyage lag before destination \( d \) (i.e., it is to be unloaded at destination \( d \) or later \( (p_i \geq d) \), then its horizontal moment is calculated by constraints (11) and (12). If it is not placed at that position, then these two constraints are redundant. If the coil has been unloaded before this part of the voyage \( (p_i < d) \), then constraints (13) set \( H^i_d \) to be zero. In a similar way, constraints (14)-(16) calculate the moment contributions of coils in vertical direction. Constraints (17)-(20) calculate the absolute value of the total contribution of all coils on board to the moment in horizontal and vertical directions of the ship in the voyage lag before destination \( d \). To keep stability, the sums of the moment contributions in the two directions should be close to zero, and have to be within \([-Q^1_0, Q^1_0]\) and \([-Q^2_0, Q^2_0]\), respectively, expressed by constraints (21) and (22).

Constraints (23) mean that for the upper-layer coil in location \( j \) of row \( k \), if any of the two coils underneath is unloaded at an earlier destination than its own, then the upper-layer coil has to be shuffling and so \( z_{ijk}^0 = 1 \). Constraints (24) count the first shuffles of the coils. Constraints (25) ensure that coil \( jk1 \) (the coil allocated to location \( j \) of the upper-layer in row \( k \)) will not occupy any position on the ship after it is unloaded at its destination \( p_i \). Constraints (26) and (27) require that coil \( jk1 \) cannot take the position of any of the two coils below it before that coil is unloaded. Constraints (28) require that coil \( jk1 \) may take at most one of the three positions, its original position and the two positions below it. Constraints (29) guarantee that when the coil at a lower position is unloaded, at most one of the coils above it
can be relocated to this position. Constraints (30) ensure that coil $jk1$ must take at least one of the three positions, its original position and the two positions below it, before it is unloaded. Constraints (31) state that once coil $jk1$ is shuffled to a lower position, it will stay there until it is unloaded. Constraints (32) and (33) state that when a coil at a lower position is unloaded, its position must be taken by one of the coils above it, if there is one. If there are two coils above it, then one takes its position and the position of the other is not changed. Constraints (34) reflect the fact that there is no coil at location 0 or $g$ at the upper layer. This also guarantees that no upper-layer coil will float in the air after a lower-layer coils is unloaded. Safety rules require that the width and diameter of a lower coil must be larger than those of any coil above it. Constraints (35) and (36) ensure that such rules are still satisfied after unloading and shuffling at each destination. Constraints (37) to (40) calculate additional shuffles needed for each upper-layer coil during the voyage besides the shuffle counted by $z^{4}_{jk}$.

Constraints (41)-(43) calculate the dispersion of coils with the same destination over the stowage area. Constraints (41) require that if a coil is placed at location $j+1$ of the lower layer in row $k$ and its destination is different from that of the coil at position $j$ of the same layer in the same row, then the dispersion of coil $i$ is $z^{4}_{jk} = 1$. Otherwise the constraint is redundant. Constraints (42) and (43) require that if there is a coil located at location $j$ of the upper layer in row $k$ and its destination is different from any of the coils underneath it, the dispersion of coil $i$ is $z^{5}_{jk} = 1$. Otherwise this constraint is redundant. Constraints (44) and (45) specify the decision variables.

3. **Valid inequalities and heuristic solution for the model**

The above MILP model can be solved using a standard software package such as CPLEX. However, as the problem size increases, the computation time for solving the model increases rapidly so that for problem instances with over 20 coils the model cannot be solved within a reasonable time. To help reducing solution time, we derive the some valid inequalities for the model, and propose a decomposition heuristic to solve the model approximately.
3.1 Valid inequalities

1) Because the number of coils at the lower level of a row has an upper bound of \( g \), the maximum number of coils that may be stowed in a row is \( 2g-1 \). Based on this, a lower bound for the number of rows can be calculated: \( \left\lceil \frac{n}{2g-1} \right\rceil \). In the model the number of stowed rows can be obtained by checking whether there is a coil stowed in the first position of each row. This analysis gives the following valid inequality:

\[
\sum_{k=1}^{m} \sum_{i=1}^{n} x_{i,k,0} \geq \left\lceil \frac{n}{2g-1} \right\rceil 
\]

(46)

2) Suppose that the coils are loaded in \( m^0 \) rows in the ship. Because in each of these rows the number of coils on upper layer is at least one fewer than that on the lower layer, the total number of coils on the upper layer can at most be \( \left\lfloor \frac{n-m^0}{2} \right\rfloor \). We have already known that \( m^0 \geq \left\lceil \frac{n}{2g-1} \right\rceil \) from 1) above. So we have the following valid inequality:

\[
\sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ijk} \leq \left\lfloor \frac{n-m^0}{2} \right\rfloor 
\]

(47)

3) In the situation where the sizes of the coils are all different, i.e., \( S_i \neq S_j \) for any \( i \neq t \), \( v = 1, 2 \). Because each upper-layer coils needs to be supported by two coils at the lower layer, and the width and diameter of an upper-layer coil cannot exceed of those of any of its supporting coils, the coils with the maximum and second maximum widths (or diameters) cannot be placed on the upper layer. Similarly, the coils with the third and fourth widths (or diameters) cannot be both placed on the upper layer. Denote coil \( i' \) (\( i'' \)) as the coils with the \( i \)th maximum width (diameter), we have the following valid inequalities:

\[
\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ijk} \leq 0, \quad i = 1', 2', 1'', 2''
\]

(48)
\[
\sum_{k=1}^{m-1} \sum_{j=1}^{x_{jk,1}} + \sum_{k=1}^{m-1} \sum_{j=1}^{x_{jk,1}} = 1 \tag{49}
\]

\[
\sum_{k=1}^{m-1} \sum_{j=1}^{x_{jk,1}} + \sum_{k=1}^{m-1} \sum_{j=1}^{x_{jk,1}} = 1 \tag{50}
\]

4) At the initial port, if the upper-layer coil in location \( j \) of row \( k \) has a destination number equal to or smaller than that of the lower-layer coils both in location \( j \) and \( j+1 \) of row \( k \), this upper-layer coil will not be shuffled during the voyage. Similarly, if there is no coil allocated in position \( jk1 \) after a port, the corresponding variable \( u_{2jkd} \) of the subsequent ports must be 0. Therefore, we have the following valid inequalities:

\[
z_{jk}^2 \leq z_{jk}^1, \quad j = 1, \ldots, g - 1, \quad k = 1, \ldots, m \tag{51}
\]

\[
z_{jk}^3 \leq z_{jk}^1, \quad j = 1, \ldots, g - 1, \quad k = 1, \ldots, m \tag{52}
\]

\[
\sum_{j=1}^{n} x_{jk1} \geq u_{2jk1}, \quad j = 1, \ldots, g - 1, \quad k = 1, \ldots, m \tag{53}
\]

\[
u_{2jkd} \geq u_{2jk,d+1}, \quad j = 1, \ldots, g - 1, \quad k = 1, \ldots, m, \quad d = 1, \ldots, P - 1 \tag{54}
\]

3.2 A model-based decomposition heuristic

To reduce solution time, we propose a model-based decomposition heuristic of two steps. In the first step the original model is simplified by ignoring the vertical moment balance requirement. Hopefully the simplified model can be solved in much shorter time than the original model. The solution of the simplified model in this step will provide the stacking plan in each row. We refer to each row of coils in this solution as a “line” of coils. However, the vertical moment balance may be poor or even does not satisfy the balancing requirement. Therefore in the second step we reassign the lines of coils to the rows on the ship and optimize the vertical coordinates of the rows to minimize the vertical moment imbalance. Note that the decisions of the second step are for each line of coils as a whole and so will not affect the coil stacking within the lines. Therefore the objective function value of the first step will not be changed, i.e., the ship imbalance in horizontal direction during the voyage, the number of shuffles, and the dispersion of coils of each destination will
not be affected by the second step because the horizontal position and the layer for each coil are fixed.

The second step problem can be solved easily using a small integer programming model. We define the following parameters which can be calculated from the first step results.

- $m'$: the number of coil lines.
- $S_{\text{max},i}$: the largest width of the coils in line $i$, $i = 1, \ldots, m'$.
- $S_{\text{total},i,d}$: the total weight of coils in line $i$ before destination $d$, $i = 1, \ldots, m'$, $d = 1, \ldots, P$.

We redefine variables $H_{d}^{i}$ and $y_{k}$ and define new assignment variables as follows.

- $H_{d}^{i}$: the contribution of coil line $i$ to the moment in vertical direction before destination $d$.
- $y_{k}$: the vertical coordinate of row $k$, $k = 1, \ldots, m'$.
- $\beta_{ik} = 1$ if line $i$ of coils is assigned to row $k$.
- $\beta_{ik} = 0$ otherwise.

For modeling convenience, we define the following constants: $y_{0} = 0$, $y_{m'+1} = W$, $S_{\text{max},0} = 0$, $S_{\text{max},m'+1} = 0$, $\beta_{i0} = \beta_{i,m'+1} = 0$, $i = 1, \ldots, m'$.

Minimize

$$\alpha \sum_{d=1}^{P} Q_{d}$$

Subject to

$$\sum_{k=1}^{m'} \beta_{ik} = 1, \quad i = 1, \ldots, m'$$

$$\sum_{i=1}^{m'} \beta_{ik} = 1, \quad k = 1, \ldots, m'$$

$$y_{i-1} + \sum_{i=1}^{m'} (S_{\text{max},i} \beta_{ik} / 2) + D + \sum_{i=1}^{m'} (S_{\text{max},i} \beta_{ik} / 2) \leq y_{i}, \quad i = 1, \ldots, m'+1$$

$$H_{d}^{i} \geq \sum_{k=1}^{m'} (S_{\text{total},i,d} (y_{k} - W / 2) + M (\beta_{ik} - 1)), \quad i = 1, \ldots, m', \quad k = 1, \ldots, m', \quad d = 1, \ldots, P$$

$$H_{d}^{i} \leq \sum_{k=1}^{m'} (S_{\text{total},i,d} (y_{k} - W / 2) + M (1 - \beta_{ik})), \quad i = 1, \ldots, m', \quad k = 1, \ldots, m', \quad d = 1, \ldots, P$$

$$Q_{d} \geq \sum_{i=1}^{m'} H_{d}^{i}, \quad d = 1, \ldots, P$$
The overall procedure of the heuristic is presented below.

**Heuristic H1:**

Step 1. Solve the simplified model:

Minimize \( \alpha_1 \sum_{d=1}^{P} Q_d^1 + \alpha_2 \left( \sum_{i=1}^{m} \sum_{j=1}^{z_{i,j}^1} + \sum_{i=1}^{m} \sum_{j=1}^{z_{i,j}^2} + \sum_{i=1}^{m} \sum_{j=1}^{z_{i,j}^3} \right) \) + \( \alpha_3 \left( \sum_{i=1}^{m} \sum_{j=1}^{z_{i,j}^4} + \sum_{i=1}^{m} \sum_{j=1}^{z_{i,j}^5} \right) \)

Subject to all constraints in the original model except constraints (14), (15), (16), (19), (20) and (22).

Step 2. Based on the results of step 1, calculate parameters \( m', S_{max,i} \) and \( S_{total,i,j} \), then formulate and solve the model for vertical balance as presented above.

From the description of the heuristic, we can see clearly that the first step model is a relaxation of the original model and its objective function is part of the original objective function. Therefore we have the following property.

**Property 1:** The objective value of the first step model in H1 is a lower bound for the objective value of the whole problem. If the second step model is feasible, the result of the second step provides a heuristic solution for the original problem, with an objective value being the sum of the objective values of the two steps. In addition, if the objective value of the second step model is 0, then the solution obtained by this algorithm is optimal.

Another possible way to decompose the model is to ignore the horizontal balance requirement first, and then in step 2 the sequence of coils in the same row can be adjusted to reduce the horizontal imbalance. This may speed up the solution process to some extent, but the solution quality would be affected. In H1, though the vertical balance requirement is ignored in step 1, it can still be balanced perfectly in step 2 because not only the lines (rows) can be rearranged, their exact positions (and so the vertical moment) can be adjusted continuously subject to the requirement of the gap in between. Therefore, it gives better solutions. In addition, since the objective of step 1 only ignores vertical imbalance which
will be close to zero, it provides a tight lower bound which may be used to evaluate the other heuristics.

The model of step 2 is small and can be solved very quickly. The solution also has an excellent vertical balance. Therefore, there is little room for improving the solution through iteratively executing the two steps of H1. On the other hand, step 1 itself takes long to solve. It would be possible to further decompose the problem of this step and solve it iteratively. For example, we may first divide all coils into $m$ subsets, each for a row, and solve the sub-problem for each row; and then based on the results for different rows we may re-divide the coils and start a new iteration. This becomes a search process for a good division of coils to the rows, and as the results of the rows may not provide a good guide on the re-division of coils, the search may not be as effective as metaheuristics such as tabu search. More importantly, the current step 1 model provides a lower bound for the problem and we use it to evaluate heuristic solutions for small and some medium problems. An iterative heuristic solution of step 1 cannot give a lower bound anymore. Therefore, we do not further decompose the problem in step 1 of H1.

The computation time needed for H1 is still too long if the number of coils to be loaded is large. This motivates us to develop faster heuristic algorithms for SSPP.

4. The tabu search algorithm

In this section we present an algorithm based on the Tabu Search (TS) technique proposed by Glover [5]. There are many algorithms for various optimization problems in steel production, e.g., Tang et al. [16] and [17]. However, for sequence-based scheduling problems, TS has been shown to be one of the most effective local search techniques able to avoid being trapped at a local optimum and to find near-optimal solutions, e.g., Nowicki and Smutnicki [12], Barbarosoglu and Ozgur [4], Watsona et al. [20], Norman [11], and Grabowski and Wodecki [6]. The SSPP is to determine the arrangement of coils on the ship and the problem shares some features with scheduling problems, which motivates us to adopt TS to solve our problems.

The main components of TS include initial solution, move, neighborhood, tabu list and stop criteria. According to the features of SSPP, a heuristic is proposed to generate an initial
solution and a speed-up strategy is adopted to accelerate the search process of the algorithm. A tabu list with variable length is proposed for our problems. Each of the components is described as follows.

4.1 Initial solution

In this section, we provide an item stowage heuristic (IS) to generate an initial solution for TS. The IS heuristic has three stages: estimating the upper bound for the number of rows, \( m \); sequencing the coils; and allocating the coils on the ship.

Stage 1. Calculate the upper bound for the number of rows, \( m \), and the upper bound for the number of coils at the lower layer of a row, \( g \), using formulae (1) and (2).

Stage 2. Sequence the coils in reverse order of their destinations. For coils with the same destination, priority is given to those with higher value of \( \lambda_1 S_{i1} + \lambda_2 S_{i2} \), where \( \lambda_1 \) and \( \lambda_2 \) are adjusting parameters. Further ties are broken arbitrarily.

Stage 3. Determine the stowage position of the coils in the following three steps.

1. Assign the coils, one by one, in the sequence obtained in stage 1 into the \( m \) rows in the following way: row 1, row 2, ..., row \( m \); row \( m \), row \( m-1 \), ..., row 1; row 1, row 2, ..., and so on. Figure 5 illustrates this method using an example with 3 rows, where \( i' \) is the \( i^{th} \) coil in the sequence obtained in Stage 1.

2. Calculate the total weights of the coils assigned to the rows. Rearrange the rows on the ship in decreasing order of their weights. If \( m \) is an odd number, place the first row in the middle position, place the rest rows alternately on the left and the right sides of the stowage area as expressed in Figure 6(a). If \( m \) is an even number, the odd numbered and even numbered rows are allocated symmetrically as...
Step 3 In each row, arrange the stowage of coils, one by one in their order in the sequence obtained in stage 1, from the middle position to two sides alternately. When the lower-layer of a row is full of coils, the rest coils are assigned on the upper-layer also starting from the middle position. Figure 7(a) and 7(b) show situations where $g$ is odd and even, respectively. Final, shift the position of whole row backward until there is no gap between the rear coil and the back edge of the stowage area.

After determining the positions of coils, the objective function value of the solution can be calculated in the same way as in the model.

4.2 Neighborhood structure

Considering the characteristics of the problem, in this algorithm, we generate a neighbor of a given solution by feasibly swapping a coil with another coil or with an empty position under the size and operation constraints.

1. A feasible swap of two coils means that after the swap each of the two coils satisfies the width and diameter relationships with coils around its new position and that the
new lengths and widths of related rows satisfy the size constraints of the storage area.

2. A feasible swap between a coil (in the upper layer) and an empty position within the same row means that the width and diameter of the coil must not exceed, respectively, those of any coil underneath it after the swap. Because the initial solution provides a good vertical balance of the ship, if an upper coil swaps with an empty position of a different row, the vertical balance is more likely to become worse. That is why swapping between a coil and an empty position is limited within the same row. On the other hand, such swapping may generate more promising solution with better horizontal balance. Due to the stowage rules given in the problem description, we know that there must be no empty position in the lower layer and so a coil in the lower layer cannot be swapped with an empty position.

We define the neighborhood of the current solution as a set consisting of $N$ randomly generated neighbors of the solution.

4.3 Tabu list

In the proposed TS, we construct a Tabu list to directly record the solutions of most recent moves to avoid the repetitive searching during the iteration process. In the neighborhood of the current solution, among those not in the tabu list, the one with the minimum objective value is accepted as the current solution for next iteration, and added to the tabu list. When the number of solutions in the tabu list is over a certain length, the oldest solution in the list will be dropped.

In this paper, the length of the tabu list ($LT$) is dynamically changed according to the difference between the best objective value found so far, $f_{best}$, and the best objective value obtained in the last $nr$ iterations, $f_{ar}$. Within the last $nr$ consecutive iterations, if the objective value $f_{best}$ is not improved and the above mentioned difference is above a threshold value $\delta_u$, the tabu list length will be decreased by 1. On the contrary, if $f_{best}$ has not been improved and the difference is below a threshold value $\delta_l$, the tabu list length will be increased by 1. The initial length is set to $\left\lfloor \sqrt{n} \right\rfloor$. 

27
4.4 Stopping criteria

Two common stopping criteria are used in the algorithm: 1) the maximum number of iterations \((T)\) has been reached; 2) the maximum number of consecutive iterations without improvement on the objective value \((TW)\) has been reached.

4.5 Calculation of objective function

For any given solution, its objective function value can be calculated using the expression in the model. In the tabu search process, each trial solution is obtained by a swap on the current solution. In order to accelerate the search process, we calculate the objective value of a trial solution based on the objective value of the current solution and the objective value change caused by the swap.

4.6 Further improvement

During the searching process of TS, the vertical coordinates of the rows are fixed. Therefore the balance in vertical direction may be further improved by adjusting these coordinates. After the search finishes, we use step 2 of H1 to minimize the vertical imbalance based on the solution obtained by the TS iteration.

4.7 The procedure of proposed TS

Combining the components introduced above, the whole procedure of proposed TS can be described as follows.

Step 1. Generate an initial solution \(s_0\) using the method presented in Section 4.1 and calculate the corresponding objective function value \(f_0\). Set \(s_{cur} = s_{best} = s_0\), and \(f_{cur} = f_{best} = f_0\).

Initialize \(N, LT, T, TW\), and \(nr\). Set iteration count \(t=0\) and iteration count without improvement \(tw=0\).

Step 2. Iteration of TS

Step 2.1. Create a neighborhood of \(s_{cur}\) and calculate the corresponding objective function values.

Step 2.2. Find the best solution in the neighborhood but not in the tabu list, set \(s_{cur}\) and \(f_{cur}\) as this solution and its corresponding objective value, update the tabu list, and set \(t = t + 1\).

Step 2.3. If \(f_{cur} < f_{best}\), set \(f_{best} = f_{cur}\), \(s_{best} = s_{cur}\), and \(tw = 0\), go to Step 2.5. Otherwise, set \(tw = tw + 1\) and go to Step 2.4.

28
Step 2.4. Update $LT$ and the tabu list according to Section 4.3.

Step 2.5. If $t<T$ and $tw<TW$, go to Step 2.1. Otherwise, go to Step 3.

Step 3. Further improve the solution by step 2 of H1. Stop.

5. Computational experiments

In this section, we evaluate the performance of the solution methods mentioned in this paper for the stowage planning problem through computational experiments. We generate a test instances according to the real data obtained from Baosteel. These data include ship information and coil information in the stowage plans. The solutions compared in the experiments include those obtained by the original model, the model with valid inequalities, the model-based decomposition heuristic H1, the proposed TS algorithm ($p$TS), the classic TS algorithm ($c$TS), the classic Particle Swarm Optimization ($PSO$), and the manual method. $PSO$ is a population-based meta-heuristic proposed by Kennedy and Eberhart [10], which searches the solution space based on particles’ velocity and position update mechanisms without any prior knowledge of the problem and performs well in solving many other problems. All the methods are implemented in Visual Studio C# 2012 and the MILP models are solved using CPLEX 12.5. The experiments run on a personal computer with an Intel 2.83GHz CPU, 4 GB memory and Windows 7 operating system.

5.1 Experimental data

Observing the practical data collected, we found that the diameters and the widths of most coils are in the ranges of [1.68m, 1.78m] and [0.95m, 1.25m], respectively. Moreover, based on the data of 150 real coils, we have established the following relation function between the weights and the sizes of coils by binary regression method.

$$S_3 \approx 15.7 \times S_{11} + 16.8 \times S_{12} - 29.6 \quad (65)$$

For all test instances, the values of $S_{11}$ (width) and $S_{12}$ (diameter) of each coil $c_i$ are generated randomly from the ranges given above, and corresponding $S_3$ (weight) value is then calculated using (65). According to the number of coils in the problem, the test instances we generate can be classified into three groups, i.e., the small, the medium, and the large problems. For the medium and the large problems, the sizes of ships are similar to those of the real ships as described in section 1. To test our mathematical model, small problem
instances are generated by scaling down the ship size according to the number of coils selected. Each test group contains 6 instances with different numbers of destinations which range from 1 to 4. The corresponding values of \( n, L, W, \) and \( P \) are summarized in the Table 1.

Table 1. The setting of experimental data in different problem groups

<table>
<thead>
<tr>
<th>Scale</th>
<th>Index</th>
<th>( n )</th>
<th>( L(m) )</th>
<th>( W(m) )</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>15</td>
<td>11</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>15</td>
<td>11</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Medium</td>
<td>1</td>
<td>30</td>
<td>15</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>30</td>
<td>15</td>
<td>5.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>40</td>
<td>20</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>40</td>
<td>20</td>
<td>5.5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>50</td>
<td>24</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>50</td>
<td>24</td>
<td>5.5</td>
<td>3</td>
</tr>
<tr>
<td>Large</td>
<td>1</td>
<td>80</td>
<td>26</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>80</td>
<td>26</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>100</td>
<td>30</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>100</td>
<td>30</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>120</td>
<td>35</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>120</td>
<td>35</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

5.2 Manual method

The stowage plan used to be made manually based on the planner’s experiences and some simple rules. This manual solution method consists of two steps as described below.

Step 1. The given coils are first grouped by their destinations. The group of coils with a larger destination number will be loaded onto the ship earlier.

Step 2. The coils in the same group are loaded onto the ship in descending order of their size and weight. The coils are stowed onto the ship in rows one by one, and are adjusted only considering the balance of the weight of coils and the stowage requirement that the width, diameter or weight of an upper-layer coil cannot exceed those of the coils underneath it.
The planners only consider the balance of the weight of coils in horizontal and vertical directions rather than the balance of the moment of coils at each destination considered in our problem. Following the above description, a manual-based heuristic (MAN) is implemented to generate manual solutions for comparison with pTS on all test instances.

5.3 Optimal and lower bounds

To obtain the optimal solution of the problem, we use CPLEX to solve the original mathematical model and the model with the proposed valid inequalities. These two methods are denoted by CPLEX and CPLEX_V, respectively.

Furthermore, to evaluate the performance of the algorithms for larger problems, we construct two lower bounds, i.e., LB1 and LB2. LB1 is obtained by solving step 1 of H1. LB2 is obtained by solving the original model three times, each time with one objective component and the constraints related to it, and then summing up the results to get the whole objective value.

5.4 Parameter setting

To implement the solution methods, we need to set the values of coefficients $\alpha_1$, $\alpha_2$, and $\alpha_3$ in the objective function. In practice, the balance of the ship during the whole voyage is the uppermost important objective of the stowage planning problem, because it directly affects the safety of the ship and cargos. The number of shuffles affects the efficiency of unloading operation at the destinations as well as the surface of the shuffled coils, while the dispersion of the coils to be unloaded at the same destination also affects the efficiency of unloading operation. Considering the relative importance of the three components discussed above, the coefficients $\alpha_1$, $\alpha_2$, and $\alpha_3$ are set to 0.5, 0.3, and 0.2, respectively. Note that these coefficients may be set differently for different companies according to their needs. Our testing results show that the values of these coefficients do not have big influence on the performance of the algorithms.

The parameters used in the proposed algorithm, $\lambda_1$, $\lambda_2$, $N$, $LT$, $T$, $TW$, and $nr$ are set to 0.5, 0.5, $n$, $\left\lceil \sqrt{n} \right\rceil$, 10000, 1000, and 50, respectively. The cTS uses the same parameter settings on $\lambda_1$, $\lambda_2$, $N$, $T$, and $TW$. Besides that, it adopts a tabu list with a fixed length $\left\lceil \sqrt{n} \right\rceil$, ends the search without the improvement by step 2 of H1, and employs the swapping of upper coils with empty positions without limitation proposed in pTS. It represents a TS not taking full
advantage of the problem structure. The mean value and the standard deviation of the computational results obtained by \( pTS \), and \( cTS \) over 10 independent runs are employed to evaluate these meta-heuristics performance. For \( PSO \), we carried out test runs to choose best values of the parameters.

5.5 Comparison results

We evaluate the proposed TS algorithm in two ways. The first is to compare it with \( CPLEX \), \( CPLEX_V \), \( LB1 \), \( LB2 \), and H1 to see how close the heuristic results are to optimal. The objective values (Obj) and the computation times (CPU) taken by the above methods are listed in Table 2. The computation times are in the format of \( hh:mm:ss \). The maximum computation time for each instance is set to 48 hours. The model, the bounds and H1 cannot generate results for the large problems and some medium problems.

Based on the results presented in Table 2, we can calculate the relative deviation of the objective value of a heuristic solution from the optimal objective value, \( Dev \), and the relative deviation of objective value of a heuristic solution from the lower bound, \( Dev_L \).

\[
Dev = \frac{Obj - Obj_{opt}}{Obj_{opt}} \times 100\% \tag{66}
\]

\[
Dev_L = \frac{Obj - LB}{LB} \times 100\% \tag{67}
\]

where \( LB \) is the best know lower bound, i.e., \( LB = \max\{LB1, LB2\} \).

From the results in Table 2, we can observe that

(1) For small sized instances, both the original model and the model with valid inequalities can be solved by \( CPLEX \) but the latter takes less time to get a solution. The lower bounds are equal to the optimal solution. Comparing with the optimal solution, we can see that heuristic H1 and \( pTS \) can obtain the optimal solutions for these instances, while \( pTS \) is more efficient. Consequently, the relative deviation \( Dev \) and \( Dev_L \) of H1 and \( pTS \) are 0% for all small test instances.

(2) For medium scale problems with practical ship width and length, \( CPLEX \) takes too long to solve the model even with the valid inequalities. H1 can only solve the instances with 30 coils and takes a long time. Lower bound can only be obtained for problems with up to 40 coils. The relative deviations of the heuristic solutions from the best lower bound
for the medium cases are calculated and listed in Table 3. For the problems that H1 can solve, H1 performs better than pTS, but it takes hours to solve a problem instance. The solutions given by pTS are not far from the lower bound and it takes only a few second to solve an instance.

Table 2. Results comparing pTS with CPLEX, CPLEX_V, LB1, LB2, and H1

<table>
<thead>
<tr>
<th>Scale</th>
<th>Index</th>
<th>CPLEX</th>
<th>CPLEX_V</th>
<th>LB1</th>
<th>LB2</th>
<th>H1</th>
<th>pTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
<td>Obj</td>
<td>CPU</td>
</tr>
<tr>
<td>Small</td>
<td>1</td>
<td>14.213</td>
<td>0:00:01</td>
<td>14.213</td>
<td>0:00:01</td>
<td>14.213</td>
<td>0:00:01</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12.522</td>
<td>0:00:01</td>
<td>12.522</td>
<td>0:00:01</td>
<td>12.522</td>
<td>0:00:01</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>12.967</td>
<td>0:01:45</td>
<td>12.967</td>
<td>0:01:29</td>
<td>12.967</td>
<td>0:00:53</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12.241</td>
<td>0:02:19</td>
<td>12.241</td>
<td>0:02:07</td>
<td>12.241</td>
<td>0:01:11</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2.403</td>
<td>1:24:06</td>
<td>2.403</td>
<td>1:14:12</td>
<td>2.403</td>
<td>1:09:22</td>
</tr>
<tr>
<td>Medium</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.737</td>
<td>10:23:06</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2.436</td>
<td>9:09:11</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.078</td>
<td>37:21:54</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.836</td>
<td>43:28:09</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Large</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3. The relative deviation of heuristic solutions from the best lower bound for medium problems

<table>
<thead>
<tr>
<th>Index</th>
<th>Dev_L(H1)</th>
<th>Dev_L(pTS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00%</td>
<td>8%</td>
</tr>
<tr>
<td>2</td>
<td>0.00%</td>
<td>9.03%</td>
</tr>
<tr>
<td>3</td>
<td>0.00%</td>
<td>10.2%</td>
</tr>
<tr>
<td>4</td>
<td>0.00%</td>
<td>7.29%</td>
</tr>
</tbody>
</table>

Since we cannot get optimal solutions or lower bounds for larger problems, we compare the proposed pTS with cTS, PSO, and MAN and observe how much improvements the meta-heuristics can make over the manual solutions. The experiment results are summarized in Table 4. Column Mean and Std indicate the mean value and the standard deviation of the
computational results obtained over 10 runs of each algorithm. These computation times are not presented in the table because these algorithms all take short time to solve the test instances.

Table 4. Results comparing pTS, cTS, PSO with MAN

<table>
<thead>
<tr>
<th>Scale</th>
<th>Index</th>
<th>pTS Mean</th>
<th>pTS Std</th>
<th>cTS Mean</th>
<th>cTS Std</th>
<th>PSO Mean</th>
<th>PSO Std</th>
<th>MAN Mean</th>
<th>MAN Std</th>
<th>Obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>14.213</td>
<td>0.000</td>
<td>14.213</td>
<td>0.000</td>
<td>14.213</td>
<td>0.000</td>
<td>15.115</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12.522</td>
<td>0.000</td>
<td>12.522</td>
<td>0.000</td>
<td>12.522</td>
<td>0.000</td>
<td>13.314</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>12.967</td>
<td>0.000</td>
<td>12.967</td>
<td>0.000</td>
<td>12.967</td>
<td>0.000</td>
<td>14.359</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12.241</td>
<td>0.000</td>
<td>12.241</td>
<td>0.000</td>
<td>12.460</td>
<td>0.154</td>
<td>13.536</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2.403</td>
<td>0.000</td>
<td>2.403</td>
<td>0.000</td>
<td>2.891</td>
<td>0.288</td>
<td>3.471</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1.894</td>
<td>0.000</td>
<td>1.955</td>
<td>0.103</td>
<td>2.741</td>
<td>0.489</td>
<td>3.403</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>1</td>
<td>1.876</td>
<td>0.123</td>
<td>2.534</td>
<td>0.003</td>
<td>3.201</td>
<td>0.034</td>
<td>4.335</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.656</td>
<td>0.312</td>
<td>3.974</td>
<td>0.297</td>
<td>4.765</td>
<td>0.331</td>
<td>5.528</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.392</td>
<td>0.401</td>
<td>4.120</td>
<td>0.492</td>
<td>4.134</td>
<td>0.930</td>
<td>5.956</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.897</td>
<td>0.173</td>
<td>2.573</td>
<td>0.198</td>
<td>3.100</td>
<td>0.576</td>
<td>3.061</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>15.004</td>
<td>0.022</td>
<td>16.727</td>
<td>0.192</td>
<td>17.082</td>
<td>0.612</td>
<td>18.606</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>1</td>
<td>22.879</td>
<td>3.313</td>
<td>23.571</td>
<td>0.149</td>
<td>24.483</td>
<td>0.597</td>
<td>26.519</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.732</td>
<td>3.021</td>
<td>32.214</td>
<td>3.601</td>
<td>32.228</td>
<td>3.967</td>
<td>34.709</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>29.263</td>
<td>0.026</td>
<td>29.792</td>
<td>4.413</td>
<td>31.286</td>
<td>4.811</td>
<td>34.556</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>40.041</td>
<td>7.445</td>
<td>40.876</td>
<td>8.097</td>
<td>40.910</td>
<td>8.534</td>
<td>45.908</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>42.234</td>
<td>2.957</td>
<td>42.675</td>
<td>3.115</td>
<td>43.558</td>
<td>3.156</td>
<td>43.809</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to the results listed in Table 4, we can get the conclusions that pTS, cTS, and PSO outperform MAN in all cases. For our SSPP, the neighbourhood-based meta-heuristic TS searching from an initial solution can obtain more promising solutions. The improvement of the objective value of a heuristic solution over that of the manual solution can be calculated as follows.

\[
\text{Imp} = \frac{\text{Obj}_{\text{MAN}} - \text{Obj}}{\text{Obj}_{\text{MAN}}} \times 100\% \quad (68)
\]

Calculating the improvement for each instance and then taking average, we can obtain that the average improvement of pTS, cTS and PSO over MAN are 13.80%, 9.97 and 7.30%, respectively. Note that some of the problem instances have small objective values, and so a
small improvement in the objective value will appear to be a high relative improvement. If only those instances with higher objective values are considered, the above percentage improvement will be lower. The results in the table give more detailed information. Nevertheless, the results clearly show that the proposed pTS algorithm significantly improves the manual solution and performs better than the standard heuristics tested.

6. Conclusion

In this paper, we studied a ship stowage planning problem of coils arising in the practical logistics system of the Baosteel Company. A mixed integer linear programming model was proposed and several valid inequalities were constructed to accelerate the process of solving the model. A property of optimal solutions of the problems was derived. To solve the problem more efficiently, a heuristic was introduced to generate the initial solution in order to start the searching of a modified TS algorithm. Experiment results showed that the proposed TS approach can obtain the optimal solutions for small problems and good solutions in medium and large scale scenarios. Compared with the manual solutions and other methods, the proposed TS showed great improvement.

Acknowledgement

This research is partly supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 71321001) and State Key Program of National Natural Science Foundation of China (Grant No. 71032004).

References


