日本船舶海洋工学会講演会論文集 Vol.14, pp. 431--434 (2012)

スタック構造を有する造船大組ブロックストックヤードのスケジューリング最適化

正会員　木 村 元*　　非会員　酒 井 栄 輔**

Optimization of Scheduling in a Ship-Building Stockyard with Stack Structure

by Hajime Kimura, *Member*　　Eisuke Sakai, *Nonmember*

Key Words: *Block Stockyard, Block Placement System, scheduling optimization, Stacks*

1. 緒　　言

造船所では，建造量を増やすためにドッグにおける建造期間の短縮に努めており，そのためにはブロック総組およびドックでのブロック搭載を迅速に行うことが重要である．一方で，生産の効率化の見地からは工場の設備や人員が一定であることが理想的であるため，ブロックの建造能力もほぼ一定となることから，総組およびブロック搭載日にジャスト・イン・タイムにブロックを建造・供給することは不可能である．そこで，大組ブロックをストックヤードにストックしておく必要が生じる．しかし、船舶の建造ブロックは巨大であるため一般的な機械部品のように倉庫のラックにストックしたり，コンテナのように積み重ねることが不可能である上，品数も多いため広大なストックヤード用の敷地が必要となる。国土が狭く地価が高い日本の造船所では，十分な広さのストックヤードの確保は困難である．ブロックを台車で運搬する場合、ストックヤードの形態としては，必要なブロックをすぐに取り出せるようブロック置場の全てが運搬用の通路に面していることが理想的だが，ストックヤード面積を節約し，より多くのブロックを置けるようにするために，通路に面した置場の奥の方に，通路に面さない別の置場を配し，手前の置場を通ってブロックを出し入れする「スタック」と呼ばれる先入れ後出し(First In Last Out: FILO)の構造にせざるを得ない．しかし，スタックによる蔵置では，全ブロックの出入庫スケジュールに合わせて巧みに配置場所を決めてやらない限り，奥のブロックを取り出す際に手前のブロックをどかすという無駄がほぼ必然的に発生するという問題がある．造船所の規模や扱うブロック数によっては，ブロック蔵置場所の決定を現場任せにしておくと，ストックヤード内を四六時中台車が右往左往するような事態を招いてしまう．本研究の対象としている造船所では，大部分のスタックの深さが 2～4 段であり，また最大で 6 段ものスタックが存在している．そのためスタック構造まで考慮に入れ，無駄の生じないブロックの蔵置場所を自動的に計画するシステムが求められる．著者らは，スタック構造を持つストックヤードへのブロック蔵置問題を組み合わせ最適問題へ帰着し，分枝限定法を用いて解くシステムを提案[1)]した．本研究では最適化アルゴリズムを改良し，さらにブロック形状ごとに置場を区別したり，あらかじめ置場にブロックが置かれている状況からのスケジュールが行えるような、より実用的なシステムを構築したので報告する．実際の造船所のデータを元にシミュレーションを行い，提案手法の有用性を確認する．

* 九州大学　大学院工学研究院
** トヨタ自動車株式会社

原稿受付　(学会にて記入します)
春季講演会において講演　(学会にて記入します)

2. 問　題　設　定

2.1 スタック構造とは

スタック構造とは，情報工学におけるデータの保持方法としてデータを後に入れたものを先に出す構造を表す．

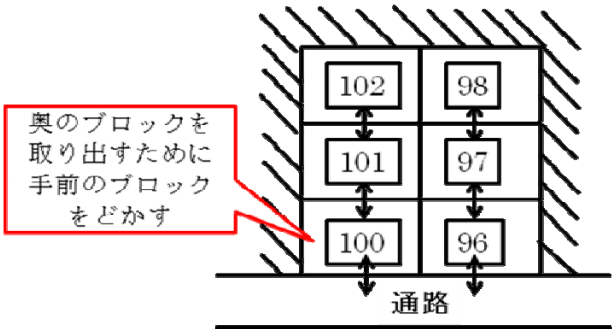

Fig. 1 A stack structure in a stockyard.

例えば Fig.1 のようにブロック置場にブロックが置かれていると仮定する．このとき，102 番のブロックを取り出すためには，それよりも通路側に置かれている 100 番と 101 番のブロックを一旦通路へ出して別の置場へ仮置きしなければならない．このとき，ブロックを支えている脚あるいは架台および台車の都合により，たとえ隣の 96, 97, 98 番のブロックが置かれていないとしても 101 番のブロックを右側にずらしてから通路へ運び出すことはできない. よって Fig.1 の 100, 101, 102 番ブロックは, 102, 101, 100 番の順にスタックへ入り，出るときは逆に 100, 101,102 番の順でなければならない．これが先入れ後出し（FILO）のスタック構造である．

蔵置される全ブロックは互いに区別され，それぞれ蔵置開始日と蔵置終了日が予め決められている．蔵置期間が最短の場合，蔵置開始日の翌日が蔵置終了日となる．

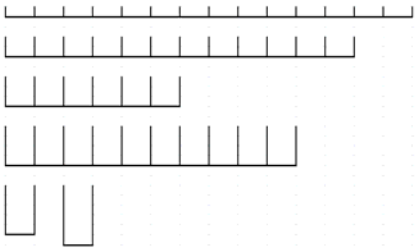

Fig. 2 A model of a real stockyard.

本研究の対象としている造船所では，Fig.2 に示すように深さ 1 段のスタック（普通スタックとは呼ばないが）が 14 組，深さ 2 段のスタックが 12 組，深さ 3 段のスタックが 6 組，深さ 4 段のスタックが 10 組，深さ 5 段と 6 段の深さのスタックがそれぞれ 1 つずつ，合計で 44 組のスタック，107 個分のブロック置場となっている（先行研究[1]とは若干異なる）．ここで解くべき問題は，蔵置期間が決まっている全ブロックをその期間中にどのスタックへ置くかを割り当てることである．ストックヤード内でのブロックの無駄な移動を避けるためには，ブロックをその蔵置期間中ずっと同一スタックで蔵置しておくことが理想的である．

2. 2 無駄なくスタックへ蔵置するためのルール

スタックではブロックの出し入れは FILO でなければならないが，これをスタックへ蔵置されるブロック同士の蔵置期間についての制約条件として定式化する．任意の 1 つのスタックについて，ブロックの蔵置がスケジュールされていると仮定し，ある時刻 t（本研究の場合 1 日単位）においてスタックの深さ d（1 が最も深く，数字が増加するほど浅い置場とする）に置かれているブロックの蔵置期間に注目する．その蔵置開始時刻を bs(t,d)と表し，蔵置終了時刻を be(t,d)と表す．ここで，時刻 t と深さ d で指定された深さと時刻に何も置かれていない場合は bs(t,d)=be(t,d) = t とする．このとき，ブロックの出し入れが FILO となるための条件は、全期間における任意の t において bs(t,d1) ≦ bs(t,d2) かつ be(t,d1) ≧ be(t,d2) ただし深さ d1≦d2 を満たすことであり，この条件を**「理想スタック蔵置ルール（ideal stacking rule）」**と呼ぶ．

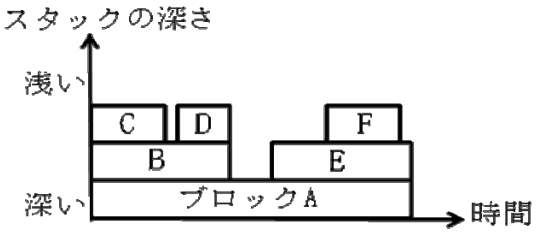

Fig.3 An example of an effective schedule following the ‘ideal stacking rule’ in a stack.

Fig.3 は上記の理想スタック蔵置ルールを満たすような蔵置期間のブロックを選んで同じスタックへ置いた例を示す．図の縦軸方向はスタックの深さ d を示すが，視覚的に理解するために上方向ほど d が大きく浅いことを示す．横軸はスケジュールの時刻を表し，図中のA～Gの矩形は各ブロックの蔵置期間を表す．つまり，各ブロックを蔵置期間中にスタックのどの深さに置くのかを示す．この Fig.3 のような置き方をすれば，奥に置かれたブロックを取り出すために手前のブロックを動かす必要は無い．

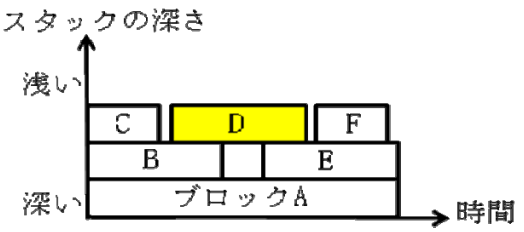

Fig.4 An example of a bad schedule which does not follow the ‘ideal stacking rule’ in the stack.

一方，Fig.4 は理想スタック蔵置ルールを満たさない置き方の例を示している．Fig.4 のブロック B をスタックから取り出す場合にはブロック D を一時的にどかす必要が生じてしまう．よってこのスタックに蔵置期間がブロック D のようにブロック B と E の期間をまたぐものをこれらより浅い位置へ置くことは不適当である．

3. ブロック割当アルゴリズムの提案

先行研究[1]では，理想スタック蔵置ルールを満たすよう置き場所を決めるためのブロックの検討順序として，まず蔵置開始日の昇順にソートし，さらに蔵置開始日が同じブロック間においては蔵置期間の降順にソートすることを提案した．この順に分枝限定法でスケジュールを作成すると，基本的に蔵置開始日の早いブロックから隙間なくきっちりとスタックを埋めていくが，蔵置期間の極端に長いブロックが一番段数の多いスタックの最深部に置かれなかったり，最悪の場合どのスタックにも蔵置期間を分割することなく置くことができないケースが観察され，スケジュールの効率低下を招いていた．本研究では，蔵置期間の長いもの順にソートを行い、これを段数が多いスタックから優先的に蔵置していくという単純なヒューリスティクスによるアルゴリズムを提案する．

3. 1 前準備：蔵置予定ブロックデータのソート

一般にスケジューリングにおいても設計作業においても，全体に対して大きな割合を占める重要（あるいは厄介）な要素から順に決定していくというのが定石である．例えば配管設計では太いパイプから，機器配置設計では大きな機器から順に配置していくことで，良好な設計案が得られることが知られている．本問題においては，蔵置期間の長いブロックほど扱いが厄介である．そこで，蔵置期間の長いブロック順にソートを行い、この順序でブロックの置き場所を検討することにより，効率良くスケジュールすることが期待できる．

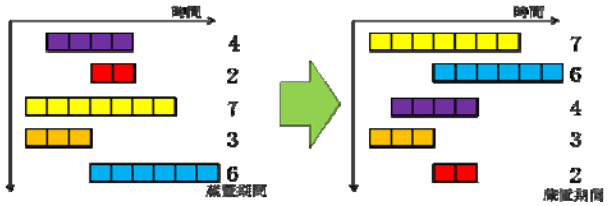

Fig. 5 An example of the sorted stock periods of blocks.

Fig.5 は 5 個分のブロックの蔵置期間について上記のソート方法を用いてソートした例を示す．Fig.5 の縦軸は各ブロックの区別を表し，横軸は時間を表す．メッシュの数（数字）は，蔵置期間を表している．このような蔵置期間順にソートした後で，Fig.5 の上のブロックから順に置くべきスタックの割当を検討していけば良い．

3. 2 ブロックの割当アルゴリズム

【深いスタックから優先して置くヒューリスティクス】

理想スタック蔵置ルールを遵守したブロックの置き方を考える場合，深いスタックほど扱いが厄介であることは直感的に自明である．そこで，3.1 節でソートされたブロック順にまず最も深いスタックに理想スタック蔵置ルールを遵守して置けるブロックを選択していき，スタックへ置くことが決まったブロックは先のソートされたブ

ロック順列から削除していく．こうして最も深いスタックに置けるブロックが無くなったら，次に深いスタックを選択し，再び残ったソート順列ブロック順に置けるブロックをソート順に選択していくという操作を繰り返す．すると，浅いスタックにあまりブロックが置かれずスカスカになったり，理想スタック蔵置ルールを満たせず置場の決まらないブロックが残るが，深さが浅いスタックは扱い易く，また残ったブロックは後述のように蔵置期間を分割して空いたスタックへ蔵置するので修正は容易である．このアルゴリズムは，配置対象のスタックに対し，3.1 節でソートされたブロック順列中のどのブロックを選択するのかについては，置けるものは必ず選択し，置けないものは選択しないという極めて単純であることが大きな特徴である．

Fig.6 は上記の方法により 5 つのブロックを 2 つのスタックへ配置した例を示す．ここで Fig.6 の左側は各ブロックが上から蔵置期間の長い順にソートされている様子を示し，図の右側は 2 つのスタックへ各ブロックが振り分けられる様子を表す． まず蔵置期間が最も長い 7 のブロックが最も段数の大きい Fig.6 右上のスタックの最も深い置場に置かれる．次いで蔵置期間の長い 6 のブロックは最も段数に多い Fig.6 右上のスタックには理想スタック蔵置ルールに従って置けないので， 2 番目に段数の多い右下のスタックに置かれる．次いで蔵置期間の長い 4 のブロックは，最も段数の大きい Fig.6 右上のスタックの 2 番目の深さの置場に置かれる．同様に蔵置期間 3 のブロックは右上のスタックに置けないので右下のスタックに置かれ，蔵置期間 2 のブロックは右上スタックの 3 番目の深さの置場に置かれる．

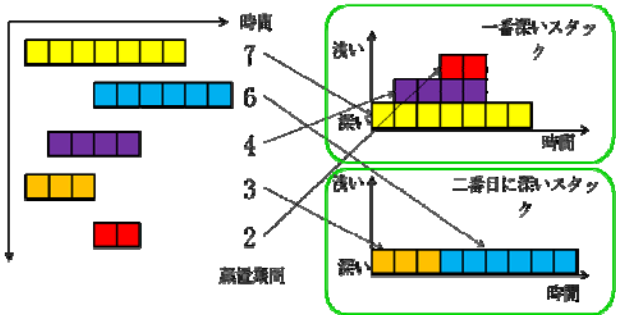

Fig. 6 An example of an arrangement of five blocks into two stack.

3.3 理想スタック蔵置ルールを遵守した場合に配置できなかったブロックの分割蔵置

ブロック数に対して置場やスタック数の余裕がないと，理想スタック蔵置ルールに従った蔵置ができないブロックが発生する．これは好ましいことではないが，このようなブロックはその蔵置期間中にストックヤード内を移動することになる．Fig.8 はあるスタックにおいて理想スタック蔵置ルールを遵守するようブロックをスケジュール後，スタックの最も浅い置場が空いている期間に，配置できなかった別のブロックを蔵置させた場合の例である．Fig.8 の赤い矩形で示すとおり，蔵置期間を分割してストックヤード内での移動が 1 回発生するたびに必要な置場が 1 つ増加してしまうことが分かる．これは不可避ではあるが，このような必要な置場の増加を抑制するため，蔵置できていないブロックの蔵置期間の分割パターンを最適化する必要がある．残念ながら本研究ではそこまでのシステムを構築できていない．今のところ暫定的に，各スタックの空き期間をサーチして，蔵置期間をなるべく分割しないで済み，かつ置き切れなかったブロックの蔵置期間と上記の空き期間とが重なる期間が最大になるブロックを選択して蔵置期間を分割する．

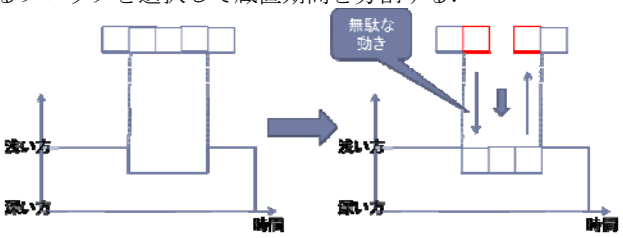

Fig. 7 Splitting the stock period of an unassigned block.

3.4 再スケジューリング手法の提案

実際のスケジューリングでは工程の遅れなどによりブロックの入庫日や出庫日が変わり，計画の中途変更をせざるを得ないことが多々ある．このとき変更された日程データを用いて再スケジューリングを行うことになるが，最も扱いが厄介なのは，すでにストックヤードに置かれているブロックの日程が変更された場合である．そこで本研究では，3.1 節および 3.2 節で説明したブロック割当処理に先立ち，まずスケジューリング開始日においてストックヤードに置かれている全ブロックが理想スタック蔵置ルールを満たしているかどうかをチェックし，満たしていないブロックに関して 3.3 節で示したブロックの蔵置期間分割を行い，全スタックにあらかじめ置かれているブロックが理想スタック蔵置ルールを満たすよう前処理を行う．これにより再スケジューリングに対応可能となる．

3.5 先行研究の分枝限定法 [1]との併用

ブロックの蔵置期間順にソートし，この順に深いスタックから埋めていくという提案手法（3.1～3.2 節）は，スケジュール全体の中でも特に蔵置期間の長いブロックの割当てに関しては効果的であるが，逆に蔵置期間の短いブロックの割当てに関しては，単に置ける置場に置くだけなので性能は期待できない．つまり，提案手法は蔵置期間の長いブロックの扱いに対して効果的だが，そうでないブロックに対しては先行研究の分枝限定法 [1]のほうが効果的である．よって，提案手法によって蔵置期間の長いブロックを割当後，先行研究の分枝限定法で残りのブロックの割当を行うのが効果的である．予備実験の結果，造船所の実データに対して，ブロックの蔵置期間順にソートしたブロックの上位 5%（のべ蔵置期間にして全体の約 16%を占める）を本手法で割り当てた場合に最も効率の良いスケジュールを生成した．

4. シミュレーション実験

本実験では 265 日間にストックヤードへ蔵置される実際のブロックの日程計画データを用いて，Fig.2 で示されるストックヤードに蔵置するスケジューリングを行った．ストックヤードに蔵置されるブロック数はのべ 3079 個になる．またこの造船所では置場の能力限界よりもブロック搭載日程の遵守のほうが優先事項であったため，日によってはストックヤードの最大蔵置能力（107 個）を 20 個程度上回る場合もある．

Table 1 Results of the simulation.

	B&B 300 branches[1)]	Proposed method 100%	Proposed method 5% + B&B 300 branches[1)]
Before splitting stock periods of unassigned blocks			
Number of unassigned blocks	174	386	163
Total number of unassigned block space	1312	1223	1050
After splitting stock periods of unassigned blocks			
Number of unassigned blocks	254	361	198
Total number of unassigned block space	869	990	796

Table 1 は各手法を用いてスケジュールを行った結果得られた解の質を，割当てられずに残ったブロック数（Number of unassigned blocks）および割り当てられずに残ったブロックの蔵置期間の合計（total number of unassigned block space）で評価した様子を示す．これらの値はゼロに近いことが望ましいが，前述のとおりヤードの最大蔵置能力をオーバーする数のブロックがヤード内に存在することがあるため，このデータにおけるブロックの蔵置期間の合計の理論的下限値は 412 である．ここでは先行研究の枝数 300 の分枝限定法(B&B 300 branches)[1)]，全ブロックを 3.2 節の提案手法で配置する方法(Proposed method 100%)およびソートされたブロックの上位 5%を 3.2 節の提案手法で配置，残りを先行研究の枝数 300 の分枝限定法で配置する方法(Proposed method 5% + B&B 300 branches)の 3 種類のアルゴリズムの比較を示す．Table 1 の上半分は 3.3 節で説明した置ききれないブロックの蔵置期間を分割する処理を適用する前のスケジュールの評価で，この値が小さいほどストックヤード内での無駄なブロック移動が少ないことを意味する．また Table.1 の下半分は 3.3 節の分割処理適用後に生成されたスケジュールのトータルな評価を表す．

提案手法は，極めて単純な処理にもかかわらず，ブロック蔵置期間分割処理を行う前のスケジュール結果は先行研究の分枝限定法よりも優れる．しかし 3.5 節で述べたとおり提案手法と分枝限定法を併用した場合が最も優れたスケジューリング結果となり，従来手法に比べ，のべ未蔵置ブロック期間期間（total number of unassigned block space）が 8.4%減少した．この下限の理論値は 412 なので，これとの差を 100%とするなら 14%減少したことになる．

Fig.8 はスケジュール期間内においてヤードに蔵置されるブロックの個数の変化，および本研究で得られた最良スケジュール（Table.1 の右下）のスケジュールに従って蔵置した場合にヤードに置ききれずにあふれるブロックの個数を示す．ブロックがストックヤードからあふれる日は置場に蔵置される個数が上限（107）より多い日とほぼ一致し，先行研究[1)]の結果よりも改善されている．

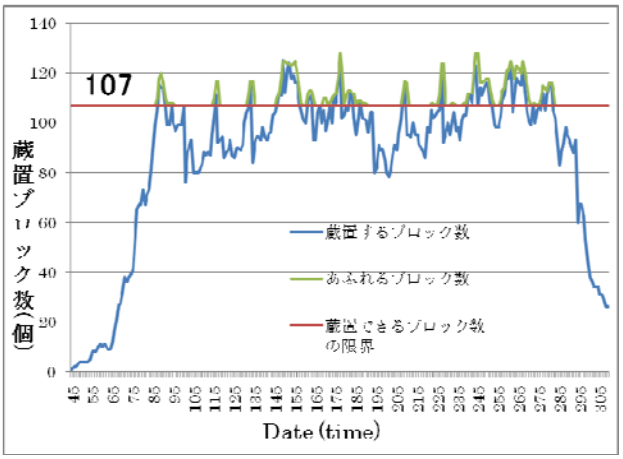

Fig.8 The number of blocks and the number of unassigned blocks in the best solution.

実際のブロックの蔵置期間データは 1 例しかないため，このデータから 1 日あたりにヤードに入るブロック個数のヒストグラムと，ブロックの蔵置期間のヒストグラムを作成し，これらのヒストグラムからサンプリングを行うことで仮想的な日程計画データを数例作成し，各手法を用いてスケジュールを行い比較を行った．すると，全てのケースで 3.2 節の提案手法を全く用いない先行研究の手法のほうが良い結果を得た．実際の蔵置期間データとヒストグラムから生成した蔵置期間データとの違いについて分析したところ，蔵置期間が同じ複数のブロックの間において、実データでは蔵置開始日と終了日が完全に同じブロックの割合が非常に高いのに対し，ヒストグラムから生成したデータでは蔵置期間が長いブロックほど蔵置開始日と終了日の一致が見られなくなる．よって 3.2 節の提案手法は蔵置開始日と終了日が完全に同じブロックが多い場合において有効であると考えられる．

5. おわりに

本論文では，スタック構造を有するストックヤードにおいて無駄な移動を生じないブロックの配置計画を生成する新しいアルゴリズムを提案した．本手法は蔵置期間の長いブロック順にソートして深いスタックから置けるブロックを置いていく極めて単純なものだが，蔵置開始日と終了日が一致するブロックが多い場合に極めて効果的であることを実験により確認した．また実用に供するために 1）予め置場にブロックが置かれた状態から再スケジュールを行う機能，2）並行部ブロック置場と曲がり部ブロック置場の区別，などの機能を盛り込んだ．

本システム実用化への課題として，実際の置場の状況を把握してシステムへ簡単に入力する方法の開発がある．

謝　　辞

本研究を遂行するにあたりまして，株式会社大島造船所の人位康弘様より貴重なデータおよびコメントをご提供いただきました．ここに厚く御礼申し上げます．

参 考 文 献

1) 木村 元：造船所内ブロックストックヤードのスケジューリングにおけるブロック配置の最適化，日本船舶海洋工学会講演会論文集，第 13 号, 2011, pp.91-94..