特別研究報告書

コンテナ積載計画問題における重量バランスを考慮した解のZDDによる列挙

指導教員：川原 純 准教授

京都大学工学部情報学科

古渡 健太

2024/01/30

コンテナ積載計画問題における重量バランスを考慮した解のZDDによる列挙

古渡 健太

内容梗概

☒ 内容梗概 (和文) ☒

english title

KOWATARI Kenta

Abstract

☒ 内容梗概 (英文) ☒

コンテナ積載計画問題における重量バランスを考慮した解のZDDによる列挙

目次

1　はじめに

コンテナ積載計画問題 (CSPP; Container Stowage Planning Problem) とは，各種の制約条件を満たしながら，コンテナ船へのコンテナの最適な積載配置を決定する問題である．最適化の目的は，荷役コストを最小化しながらコンテナ船の利用効率を最大化することであるが，この目的に従ってコンテナの配置を決定することは，非常に幅広く複雑な制約条件と目的を併せ持つ最適化問題である．

この複雑性を具体的に述べると，まずコンテナには次のような性質の幅広さがある．すなわち長さ (主に 20ft, 40ft, 45ft)，高さ (主に 8.6ft, 9.6ft)，種類 (DRY; 通常のコンテナ, REEFER; 冷蔵・冷凍機能を備えたコンテナ, OOG; 上部や側部に天井や壁が存在しないコンテナ, TANK; タンク形状のコンテナ)，内容物の重量，内容物の種類 (IMDG Codes; The International Maritime Dangerous Goods Codes に定められたコンテナ内容物の区別) [1] などの性質が存在し，その性質に伴う船との間の制約条件や，コンテナ同士の制約条件が存在する．また，コンテナが船から下される予定の港と，コンテナ船がある港を出港してからが寄港する港の順番との兼ね合いによって生まれる，コンテナの積載順番の制約なども存在する．これら以外にも，コンテナ船の構造の違い，港によって異なる荷役機器の種類など，現実の CSPP には考慮しなければならない制約条件が非常に多く存在する．

また，このような制約条件複雑の他に．目的の複雑さも存在する．具体的には，コンテナの積載にかかる時間の最小化，重心の最適化，荷役機器の移動距離の最適化，シフト作業 (あるコンテナを取り出すために，その上に置かれているコンテナを先んじて別の場所に移動させる作業) 回数の最小化，あるいはこれらの目的に対する多目的最適化などが存在する．

こういった複雑な CSPP を解く研究では，様々なアプローチと問題の単純化の手法が試されている．

問題の単純化の手法としては，REEFER の存在を考慮しないもの，OOG の存在を考慮しないもの，IMDG による内容物の違いを考慮しないもの，などが存在する．また，CSPP を解くアプローチとしては，厳密アルゴリズム，貪欲法，ヒューリスティクス，決定木ベースのアルゴリズム，機械学習など，様々なアプローチが検討されている．中でも，決定木ベースのアプローチの 1 つとして，

R. M. Jensen らによる，決定木の一種である BDD (Binary Decision Diagram) を用いた CSPP の解列挙の研究があり，制約条件を満たすコンテナ配置の解を表現する BDD の構築を行なっている．

本研究では，BDD の派生系である ZDD (Zero-suppressed BDD) を用いた CSPP の解列挙とそれを用いた最適化の手法を提案する．重量を考慮しない問題設定において，同じ解集合を表現する BDD・ZDD について，ほとんど全ての場合で ZDD の節点数が BDD の節点数よりも少なくなることを確認した．また，ZDD を用いた線型重み最適化の手法によって，コンテナ重量を考慮して重心が最適な解の取得を行なった．

2 準備

2.1 BDD

BDD は論理関数を DAG (Directed Acyclic Graph; 有効非巡回グラフ) で表現する手法である．$F(a,b,c) = a\bar{b}c \vee \bar{a}b\bar{c}$ を二分決定木・BDD で表現した例を図 1 に示す．BDD は，論理関数の値を全ての変数について場合分けした結果を表す二分決定木に対して，簡約化処理を可能な限り加えることによって得られる．これによって，論理関数をコンパクトかつ一意に表せることが知られている．まず，二分決定木においては，各節点の 1-枝と 0-枝は，その節点が表現する変数に 1 を割り当てるかどうかの場合分けを表し，値の 0 である葉 (0-終端節点) と 1 である葉 (1-終端節点) はそれぞれその葉に対応する変数割り当てが偽となるか真となるかに対応する．この二分決定木に対して，場合分けする変数の順序を固定し，(1) 冗長な節点を削除する，(2) 等価な節点を共有する，という簡約化規則を可能な限り加えることで，BDD を得ることができる．

また，BDD を用いて組合せ集合を表現することもできる．組合せ集合は，特性関数と呼ばれる論理関数に対応付けることができるが，この特性関数を BDD で表すことにより，その組合せ集合を非明示的に表現できる．

2.2 ZDD

ZDD は，組合せ集合データを表現・操作するのに特化した BDD の派生系である．等価な節点を共有する規則は BDD と同様であるが，冗長な節点を削除する規則が異なる．ZDD の場合は，1-枝が 0-終端節点を指している場合に，こ

れを取り除くという規則を用いる．この規則を用いた場合でも，表現の一意性は失われない．

一般に，組合せ集合に類似する部分組合せが多数含まれる場合，それを表現する BDD には等価なサブグラフが多く出現する．これらが互いに共有されることによって，コンパクトに圧縮された表現が可能となる，特に疎な組合せ集合の場合，ZDD ではその簡約化規則から，組合せ集合に登場しない要素に関する節点が削除されるため，同じ組合せ集合を表現する BDD よりも効率よく組合せ集合を表現・操作することができる．BDD と ZDD のデータ圧縮率の違いは，組合せ集合の各要素に含まれるアイテムの出現頻度に依るが，例えばアイテムの出現頻度が 1%である場合，ZDD は BDD よりも 100 倍コンパクトになる可能性がある．

2.3 BDD・ZDD の演算

論理関数を BDD・ZDD で表現したとき，この論理関数の AND/OR 演算は，同じ論理関数を特性関数とした組合せ集合の交わり (intersection)/結び (union) の演算にそのまま対応する．従って，多数の要素を含む集合同士の演算を，BDD 処理系で一挙に実行することが可能である．

ZDD の場合も同様であり，ZDD は ZDD 同士の様々な演算を，圧縮されたデータ量にほぼ比例する計算時間で実行できるという利点も持つ．すなわち，圧縮されたデータを，元のデータ量に戻すことなく，圧縮されたままの状態で高速に演算処理が可能である．

2.4 BDD・ZDD による⇒の表現

論理関数 f と g が与えられたとき，$f \Rightarrow g$ は $\bar{f} \vee g$ と等価である．また，論理関数は BDD で表現することができ，$\wedge$ や $\vee$ の論理演算を BDD のまま計算するアルゴリズムが知られている．従って，f と g を表現する BDD が与えられたときに，$\bar{f} \vee g$ を計算することで，$f \Rightarrow g$ を表現する BDD が構築できる．

2.5 ZDD による線型重み最小化

ZDD を用いることで，線型重み最適化が可能である．ZDD の各変数に対して重みを設定することで，ZDD が表現する組合せ集合の中から，重みの総和が最小・最大の組合せを取得したり，上位複数個の解を表現する ZDD を取得し

たり，重みの総和が指定した値以上・以下・未満・より大きい組合せを表現する ZDD を取得したりすることができる．

本研究では, SBDD_helper ライブラリに用意された getMinimum 関数, weightGE 関数，そして getKLightest 関数を使用する．以下に，それぞれの関数のアルゴリズムを説明する．

2.5.1 各種関数

getMinimum 関数

ZDD に対して用い，重みベクトルと空の std::set を引数とし，重み最小の集合とその重みを計算して返す関数である．

Algorithm 1 $getMinimum$

1: something

weightGE 関数

Algorithm 2 $weightGE$

1: something

getKLightest 関数

Algorithm 3 $getKLightest$

1: something

3 既存手法

3.1 CSPP の定式化

CSPP の解列挙の問題を，Jensen に基づき定式化する．セル (コンテナが置かれる可能性がある位置) の配置については簡単のため 2 次元とし，水平方向 T, 垂直方向 V の $T \times V = M$ 箇所のセルとする．セルの位置を表現する変数として $c_i (i \in \{1, 2, \ldots, M\})$ を用意する．図に示す通り，添え字は下左端セルを 1 として水平方向に増えていき，T 個ずつ垂直方向に重なるように並ぶ.

コンテナは1つずつ個別にではなく, コンテナ複数を含むグループ単位で区別さ

れるものとし, グループの番号を $1, 2, \ldots, G$ とする．また, 値 0 を ”コンテナが置かれていない” を表現する特別なグループ番号とする．$c_i = g(g \in \{0, 1, \ldots, G\})$ によってセル c_i にグループ g のコンテナが置かれていることを表現する．グループごとのコンテナの個数については，インスタンスごとに定められているものとし，グループ 0 も含めたグループ毎のコンテナの数の合計がセルの総数 M になるものとする．

以上のように表現されるセルへのコンテナの配置に対して，様々な制約条件を考え，制約条件を満たすコンテナの配置を全て求める問題が，CSPP の解列挙の問題である．

先行研究において考慮される制約条件としては，次のようなものが存在する．

1. 1 つのセルに 20ft と 40ft のコンテナを同時に置くことはできない．
2. 20ft コンテナは 40ft コンテナの上に置くことはできない．
3. あるセルに 20ft コンテナが置かれるならば，同じセルにもう 1 つの 20ft コンテナが必ず置かれる．
4. コンテナは宙に浮かない (高さ 2 以上の位置のセルにコンテナが置かれたとき，その下には必ず別のコンテナが置かれている)．
5. 1 つのコンテナは 1 つのセルにのみ置かれる．
6. 1 つの列 (水平座標が等しいセルの集合) に置かれるコンテナの総重量は，重量上限を超えない．
7. 荷下ろしされる港の番号を，寄港する順番に 1, 2, …とすると，水平座標が等しいセル集合の中で，垂直座標が小さい方から順に，コンテナの荷下ろし港の番号は降順に並ぶ．

本研究では簡単のため，コンテナのサイズの 20ft と 40ft を区別しないこと，コンテナを個別にではなくグループで区別すること，列ごとの重量制限を考慮しないこと，コンテナの輸送先港を区別しないこととする．従って，上記 4. の制約条件のみを採用する．

3.2 解集合を表現する BDD の構築方法

以上の問題設定の上で，解集合を表現する BDD を構築するための方法を述べる．

3.2.1 バイナリエンコード

セルを表現する変数 c_i に割り当てるグループ番号は 2 値よりも多い場合がある．一方，BDD と ZDD はともに 0-1 論理変数のみを扱うことができる．取りうる値が 2 値よりも多い変数を BDD・ZDD で扱うことは，次のように変数のバイナリエンコードを行うことによって実現できる．

1. 変数 x_i が取りうる値の数を N とした時，1 変数あたり $\log\lfloor N \rfloor + 1$ 個の 0-1 変数を用意して，$x_i^{\log\lfloor N \rfloor}, x_i^{\log\lfloor N \rfloor - 1}, \ldots, x_i^0$ とする．
2. $x_i^{\log\lfloor N \rfloor}, x_i^{\log\lfloor N \rfloor - 1}, \ldots, x_i^0$ とした時，これを x_i がとる値の 2 進数表示であるとする．

以上より，取りうる値が 0 から G である本問題設定では，1 つのセルあたり $\log\lfloor G \rfloor + 1$ 個の 0-1 論理変数でバイナリエンコードを行う．

3.2.2 制約条件の表現

次の 2 つの制約条件を BDD で表現する．

(1) コンテナは宙に浮かない

(2) あるグループのコンテナの数を k_g とした時，値 k を取る変数 c_i の数は k_g 個である．

まず，(1) の制約条件条件については，コンテナは下から床または別のコンテナによる支えがない限り，置くことはできないということを意味するものである．この制約条件は，次のように言い換えることができる．すなわち，あるセルにコンテナが置かれないならば，その上のセルにもコンテナは置かれない，ということである．さらにこれを言い換えると，あるセル変数にグループ番号 0 が割り当てられたならば，垂直方向でそのセルの 1 つ上のセルを表す変数にも，グループ番号 0 が割り当てられなければならない，と言い換えることができる．これを論理式で表現すると．

$c_i = 0 \Rightarrow c_{i+T} = 0, \forall i$

と表現できる．この論理式に対応する BDD を構築する．

続いて，(2) の制約条件については，インスタンス毎に定められたグループ毎のコンテナの数に対し，セル変数に割り当てられるグループ番号との間に，誤差が生まれないようにする，というものである．これは，次のように表現できる．

まず，n 個の論理変数のうち，k 個の論理変数が 1 を取る時に真，それ以外のとき偽となる論理関数を考える．図のように，論理変数に対応する節点を規則的に配置し．また，0-枝と 1-枝は，0-終端節点と 1-終端節点を配置するよう

にする．この時，終端節点を左端から 0, 1, . . . , k, . . . , n 番目の終端節点とすると，k 番目の終端節点のみを 1-終端節点，それ以外を 0-終端節点とする．こうして構築された DAG に対して BDD の簡約化規則を用いて得られた BDD は目的の論理関数を表現している．これを，nCk を表現する BDD であることから nCkBDD 呼ぶこととする．

次に, バイナリエンコードされている場合を考える．バイナリエンコードする前の変数をメタ変数, バイナリエンコードした変数をバイナリ変数と呼ぶこととする．n 個のメタ変数のうち, k 個のメタ変数が値 v を取るとき真, それ以外のとき偽となる論理関数を考える．例として 1 つのメタ変数あたり，3 個のバイナリ変数によってバイナリエンコードがされているとき, 例えば図のような DAG は,「メタ変数が 5 すなわち 2 進数表示で 101 を取る時に真，それ以外のとき偽となる」という論理関数を表現する．これを, バイナリエンコードする前の, nCkBDD の簡約化前の DAG の節点の位置にそれぞれ配置することで, 図のような DAG を得る．この DAG に対して BDD の簡約化規則を用いて得られた BDD は，目的の論理関数を表現している．任意の $n(n \leq 1), k(k \leq n), b(1 \leq b), v(0 \leq v \leq 2^b - 1)$ に対して,「1 個のメタ変数あたり b 個のバイナリ変数でバイナリエンコードされた n 個のメタ変数のうち, k 個のメタ変数が値 v を取るとき真, それ以外のとき偽となる」という論理関数を nCk(b,v) とし, これを表現する BDD を nCk(b,v)BDD と呼ぶこととする．

3.2.3 解集合を表現する BDD の構築

解集合を表現する BDD は，次の疑似コードで示すように．解が満たすべき制約条件を表す BDD 全ての intersection を取ることで得られる．

Algorithm 4 解集合を表現する BDD の構築

1: {コンテナは宙に浮かない を表現する BDD の構築}
2: $BDDNoFloating \leftarrow BDD(1)$
3: **for** $i = 1; i \leq M - T; i++$ **do**
4: $NoFloating \leftarrow NoFloating$ **and** ($c_i = 0 \Rightarrow c_{i+T} = 0$ を表現する BDD)
5: **end for**
6: { 解集合を表現する BDD の構築 }
7: $BDDtargetBDDleftarrowNoFloating$
8: **for** $g = 0; g \leq G; g++$ **do**
9: $targetBDD \leftarrow targetBDD$**and**$(nCk(b, g)$ を表現する BDD)
10: **end for**

4 提案手法

既存手法の場合と同様の問題設定に対して，ZDD によって解集合を表現する手法と，コンテナに対して重量を導入し，線型重み最適化を行う手法を提案する．

4.1 解集合を表現する ZDD の構築方法

既存手法の場合と同様の問題設定に対して，ZDD によって解集合を表現する手法を提案する．

4.1.1 制約条件の表現

BDD の場合と同様，以下 2 つの制約条件を考える．(1) コンテナは宙に浮かない

(2) あるグループのコンテナの数を k_g とした時，値 k を取る変数 c_i の数は k_g 個である．

まず，(1) については，$c_i = 0 \Rightarrow c_{i+T} = 0, \forall i$ に論理式に対応する ZDD を構築する．続いて，(2) については，nCkBDD を元に考える点は BDD の場合と同じであるが，バイナリエンコードされている場合の ZDD の構築方法が異なる．例として 1 つのメタ変数あたり，3 個のバイナリ変数によってバイナリエンコードされているとき，例えば図のような DAG は，「メタ変数が 5 すなわち 2 進数表示で 101 を取る時に真，それ以外のとき偽となる」という論理関数を表現す

る．これを，バイナリエンコードする前の，nCkBDD の簡約化前の DAG の節点の位置にそれぞれ配置することで，図のような DAG を得る．この DAG に対して ZDD の簡約化規則を用いて得られた ZDD は，目的の論理関数，すなわち nCk(b,v) を表現している．nCk(b,v) を表現する ZDD を nCk(b,v)ZDD と呼ぶこととする．

4.1.2 解集合を表現する ZDD の構築

解集合を表現する ZDD は，次の疑似コードで示すように．解が満たすべき制約条件を表す ZDD 全ての intersection を取ることで得られる．

Algorithm 5 解集合を表現する ZDD の構築

1: {コンテナは宙に浮かない を表現する ZDD の構築}
2: $ZDDNoFloating \leftarrow getPowerSet(M \times b)$ {初期 ZDD として，}
3: **for** $i = 1; i \leq M - T; i++$ **do**
4: $NoFloating \leftarrow NoFloating$ **and** ($c_i = 0 \Rightarrow c_{i+T} = 0$ を表現する BDD)
5: **end for**
6: { 解集合を表現する ZDD の構築 }
7: $BDDtargetBDD \leftarrow NoFloating$
8: **for** $g = 0; g \leq G; g++$ **do**
9: $targetBDD \leftarrow targetBDD$**and**$(nCk(b, g)$ を表現する BDD)
10: **end for**

4.2 コンテナ重量の表現と重心の最適化

ここまでの問題設定では，グループ毎の値は，「コンテナを置かない」を表現する 0 という値を除いて，グループ間の区別以上の情報を持たないものだった．そこで，グループ毎の値に何らかの意味を付与することによって，より複雑な問題設定を検討する．ここでは，グループ毎の値がコンテナの重量に対応した値であるという設定を導入する．コンテナの重量は，コンテナが持つ情報の中でも，船の安定性や構造上の制約を考える上で重要な性質である．これを考慮することで，現実に近い問題を考えることができる．

本研究では，グループ番号 g のコンテナは $5 \cdot g$ トンであるとする．

コンテナの重量を導入した際に，制約条件と目的のそれぞれで新たに考慮すべき点が生まれる．

まず，新たな制約条件として，

(3) あるコンテナの上には，そのコンテナより重いコンテナをおいてはならない

という制約条件を考える．これは，コンテナの構造上の制約条件や積まれたコンテナ集合の安定性などを考慮する上で用いられる制約条件である．この制約条件は，あるセル変数にグループ番号 $j(j \in \{1, 2, \ldots, G\})$ が割り当てられたならば，垂直方向でそのセルの 1 つ上のセルを表す変数には，グループ番号 0,1,. . . ,j のいずれかが割り当てられなければならない，と言い換えることができる．これを論理式で表現すると，

$c_i = j \Rightarrow \bigvee_{k=0,\ldots,j} c_{i+T} = k, i \in \{1, 2, \ldots, M - T\}, j \in \{1, 2, \ldots, G\}$

と表現できる．この論理式に対応する ZDD を構築する．

続いて，重心の位置が最適になるようにするという目的を考える．重心の位置は船の航行の安全上重要な指標の一つである．垂直方向と水平方向 2 つの重心を考えることとし，それぞれ次のように定義する．i 番目のセルに置かれたコンテナの重量を w_i とし，i 番目のセルの垂直方向の座標を $v_i(0 \leq v_i \leq V, \forall i)$, 水平方向の座標を t_i(t_i は整数, T が偶数のとき $(-T/2 \leq t_i < 0, 0 < t_i \leq T/2)$, T が奇数のとき $(-T - 1/2 \leq t_i \leq T - 1/2)$, $\forall i$) とし，

$$\text{垂直重心} = \frac{\sum_{i=1,\ldots,M} w_i \cdot v_i}{\sum_{i=1,\ldots,M} w_i} \tag{1}$$

$$\text{水平重心} = \frac{\sum_{i=1,\ldots,M} w_i \cdot t_i}{\sum_{i=1,\ldots,M} w_i} \tag{2}$$

とする．また，本問題設定においては，インスタンス毎のコンテナの総重量は，コンテナの積載の仕方によらず定数であるため，最適化のための計算の際には，

$$\text{垂直モーメント} = \sum_{i=1,\ldots,M} w_i \cdot v_i \tag{3}$$

$$\text{水平モーメント} = \sum_{i=1,\ldots,M} w_i \cdot t_i \tag{4}$$

を用いる．

4.2.1 コンテナ重量の表現

b 個のバイナリ変数によってバイナリエンコードされたセル変数 c_i の $l(1 \leq l \leq b)$ 桁めが $5 \cdot 2^{l-1}$ トン分の重量を表現しているものとし，これをバイナリ重量と呼ぶことにする．グループ番号 g のコンテナは $5 \cdot g$ トンであるとしている設定と，次の等式によって結びつく．

$$i \text{番目のセルに置かれたコンテナの重量} = \sum_{l=,1,2,\ldots,b} c_i^l \cdot (5 \cdot 2^{l-1}), \forall i \qquad (5)$$

すると，このバイナリ重量と垂直方向の座標および水平方向の座標の積によって，垂直方向と水平方向それぞれにおける，最適化のためのバイナリ変数毎の重みを次のように設定することができる．すなわち，

$$\text{垂直重み} = 5 \cdot 2^{l-1} \cdot v_i, \forall i, \forall l \qquad (6)$$

$$\text{水平重み} = 5 \cdot 2^{l-1} \cdot t_i, \forall i, \forall l \qquad (7)$$

とする．

4.2.2 重心の最適化

ZDD を用いると，getMinimum 関数を利用することで線型重み和が最小の解を得られるだけでなく，weightGE 関数や getKLightest 関数を用いることで，線型重み和が一定の値以上の複数の解を表現する ZDD を得たり，線型重み和最小化の複数の上位解を表現する ZDD を得ることができる．実用上，多様な解を選択肢として提示することができることは，解の決定を行う上で有用であると考えられる．

次の方法で，解集合を表現する ZDD の中から，垂直モーメントが最小の解を 1 つおよび複数，水平モーメントが最小の解を 1 つおよび複数取得する．

1. 垂直重み和最小の解 1 つの取得は，次の方法で行う．
   (a) 解集合を表現する ZDD を構築する．これを targetZDD とする．
   (b) 式 6 で定められるバイナリ変数毎の重みを引数とし，getMinimum 関数を targetZDD に適用する．
2. 垂直重み和最小化の複数の上位解の取得は，次の方法で行う．
   (a) 解集合を表現する ZDD を構築する．これを targetZDD とする．

(b) 式 6 で定められるバイナリ変数毎の重みを引数とし，K=1, strict=1 として getKLightest 関数を targetZDD に適用する．

3. 水平重み和最小の解 1 つの取得は，次の方法で行う．
   (a) 解集合を表現する ZDD を構築する．これを targetZDD とする．
   (b) 式 7 で定められるバイナリ変数毎の重みを引数とし, bound を 0 として weightGE 関数を targetZDD に適用する．これによって得られる水平重み和が 0 以上の解集合を表現する ZDD を GE0_targetZDD とする．
   (c) 式 7 で定められるバイナリ変数毎の重みを引数とし，getMinimum 関数を GE0_targetZDD に適用する．
4. 水平重み和最小化の複数の上位解の取得は，次の方法で行う．
   (a) 解集合を表現する ZDD を構築する．これを targetZDD とする．
   (b) 式 7 で定められるバイナリ変数毎の重みを引数とし, bound を 0 として weightGE 関数を targetZDD に適用する．これによって得られる水平重み和が 0 以上の解集合を表現する ZDD を GE0_targetZDD とする．
   (c) 式 7 で定められるバイナリ変数毎の重みを引数とし，K=1, strict=1 として getKLightest 関数を GE0_targetZDD に適用する．

5 実験結果

5.1 BDD と ZDD による解集合の表現

5.2 ZDD を用いた重心最適化

6 おわりに

本研究では，CSPP の解列挙の問題に ZDD を用いて，BDD を用いた場合と比較して多くの場合でノード数が少なくなることを確認した．また，適切に重みを設定し，ZDD を用いた線形重み最適化の手法を用いることで，水平・垂直方向での重心が最適である解や複数の上位解を取得できることを確認した．

今後の課題としては，本研究で考慮できなかった制約条件の考慮が挙げられる．具体的には，3 次元座標の導入のほか，位置ごとの重量上限の考慮，コンテナのサイズ (20ft, 40ft, 45ft) の区別，コンテナの種類 (DRY, REEFER, OOG, etc.) の区別，コンテナの内容物の IMDG に基づく区別と，それらの区別に基づく制約条件，などが挙げられる．

これらの条件を考慮する拡張を行えば，現実のコンテナ積載計画を考える業

務の支援を行うシステムの構築が検討できる．

謝辞

本論文の執筆にあたりご指導頂きました指導教員の川原純先生に深く感謝いたします．また，研究を進める上でご助言を頂いた湊真一先生をはじめ，研究会を通して様々な意見を下さり，議論を共にさせていただきました湊研究室の皆様にも心よりお礼申し上げます．

参考文献

[1] International Maritime Organization: IMDG code : International maritime dangerous goods code, Vol. 41-22.