整数計画法によるコンテナ積み付け最適化

米沢 隆

Integer Programming Optimization for Container Loading

Takashi Yonezawa

筆者は物流をはじめとする種々の最適化ソリューションを提供しているが, ある製造業のお客様向けに新たなソリューションとして海上輸送用のコンテナへの荷物の積み付け最適化システムを構築することとなった. このシステムは, 従来経験に基づいて人手で行われていた積み付け計画を, 最適化技術を適用することによって自動立案を行い納期回答プロセスを月次から週次へ短縮を狙うものである. これを実現するためには現場の制約を取り込みながら多数のコンテナの計画を高速に立案する必要があり, 筆者はこのコンテナへの積み付けの最適化を整数計画法を2段階に分けて適用することにより実現した. 本論文ではこの最適化アルゴリズムと現実の制約の取り込む方法に関して紹介する.

The author has been providing various optimization solutions including those for logistics. This paper discusses the establishment of an optimal container loading system for maritime transportation, requested by a client in the manufacturing industry. This system aims to shorten delivery timing process from monthly to weekly basis by automating the planning work using optimization technology, instead of planning manually based on experience, as was formerly used in the industry. In order to realize this system, there is a need for a high speed planning of many containers by taking into consideration the restrictions on the scene. The author implemented such optimal container loading system by applying integer programming in two stages. This paper introduces this optimization algorism and means to incorporate on-the-scene restrictions.

Key Words & Phrases：物流最適化, コンテナ積み付け, 組み合わせ最適化問題, 最適化技術, 整数計画法
logistics optimization, container loading, combinatorial optimization, optimization technology, integer programming

1. はじめに

製造業の多くにおいて物流改革は大きな課題であり, コストの削減だけではなく, リードタイムの短縮が求められており, さらに物流まで含めた納期回答プロセスのリードタイムの短縮も求められている.

しかしながらこれ等の物流計画の立案にはいまだに人手に頼らざるを得ない部分が多く, 計画の最適性に限界があるとともに計画立案に時間を要してリードタイム短縮の妨げとなっている.

このため, これらの物流計画の立案に関してシステム化, 自動化が求められており, 筆者は例えば配送経路の最適化においてはVRP（Vehicle Routing Planner）[1]といった最適化技術を用いることにより計画立案の最適化とリードタイムの短縮を実現してきた.

今回, ある製造業のお客様において海外輸出用の海上コンテナの積み付け計画の立案を自動化し, 納期回答プロセスのリードタイム短縮を目指すシステムの構築を担当することとなった.

海外輸出の納期回答プロセスは一般的に以下のような流れで行われる.

1. オーダー受注
2. 生産・引き当て計画作成
3. コンテナ計画作成
4. コンテナ船計画作成
5. 納期回答

この最終的な納期回答を行うには3のどのコンテナ船に積載するのかの計画を決定する必要があり, そのためには製品がどのようにコンテナに積み付け可能かを計画する必要がある.

提出日：2004年9月6日　再提出日：2005年12月26日

従来は, このコンテナへの積み付け計画の立案は海貨業者に人手での計画の立案を委託する場合も多く, 荷量によっては1週間といった計画立案時間が必要であり, 納期回答の短縮を困難にしている.

そこで, 本プロジェクトではコンテナへの積み付け計画に最適化技術を適用し, 手作業で作成した計画と同等以上の積み付け計画を立案することにより, このプロセスを1時間以内という非常に短時間に完了させることを目指した.

本論文では, この海上コンテナへの積み付けを現実のさまざまな制約を含め整数計画法にモデル化し, 短い計算時間で高い充填率を実現したのでその手法を紹介する.

2. コンテナ積み付けの要件

一般的にコンテナへの積載は, 木や段ボールの箱か, パレットと呼ばれる板に製品を一塊にまとめ(以降ケースと表す), それをフォークリフトでコンテナへ積み付ける. これらの積み付けには段積み制約などさまざまな現実的な制約があり, これを満たさなければ計画上は積み付け可能となったものが現場では積み付け不可能と判断され計画の手修正が必要となったり, 納期回答を訂正する必要が出てくることになる. 従ってコンテナへの積み付け計画は現場の制約を満たした実行可能なものであることが必須要件となる. 特に, システムの目的が自動化によるリードタイム短縮の場合には, 人の介在を最小限にとどめることが重要となり, 一部の制約が未実装となると, 人手による修正が必要となりシステムの効果が大きく損なわれる.

以下にこれらの要件の主なものを紹介する.

2.1. フォーク刺し口要件

ケースにはフォークの刺し口が4方向のものと2方向のものがあり, 4方向の場合には縦横自由に回転して配置できるが, 2方向の場合には回転ができず指定された方向にしか配置できない.

2.2. コンテナ内部の突起・鴨居要件

コンテナの内部は完全な直方体ではなく, クレーンでつり下げる部位に立方体状の突起が奥の隅にあり, これに干渉するケースは**図1**のように手前に配置するか, 干渉しないケースを配置する必要がある. また, コンテナの入り口には鴨居状の突起があり, この鴨居に干渉するケースの段積みはコンテナ内部で行う必要がある. そのためコンテナの入り口の直近には鴨居に干渉するケースは配置できない.

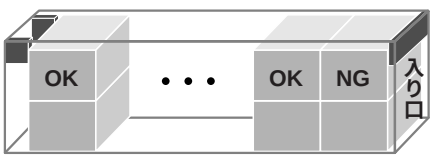

図1. コンテナの突起と鴨居による制約

ただし, このような突起や鴨居に干渉しないようにケースが設計されているものが大半で, 一部の例外的なケースに対して考慮が必要となる.

2.3. その他の要件

ケースにはその素材や重量, サイズにより段積み可能か否かの制約があり, 荷崩れ防止のためにすき間を指定値以下にしなければならないといった制約も存在する.

3. コンテナ積み付けの従来手法

3次元積み付けの研究は古く[1][2]からあり, さまざまな手法[3][4][5]が提案されている. 近年では, 遺伝的アルゴリズム, 疑似焼き鈍し法, タブーサーチといったメタヒューリスティックス手法[4][6]を用いた手法が主流となっており, 市販のパッケージ・ソフトの多くはこれらの手法を用いている.

これらの手法では, コンテナへ積み付けるケース一つ一つに対してどのように配置するのかを探索する近似解法で, 膨大な組み合わせの中から効率よく探索する工夫が施されているが, ケースを個々に処理するためにケース数が多くなった場合に計算時間が増大し, 大規模な計画への適用を困難にしている.

また, 先に挙げたコンテナ内の突起や鴨居の制約を考慮したコンテナ積み付けパッケージ・ソフトは見当たらず, 現実業務の細かな制約まで考慮した計画の立案は困難で人手による修正を必要としている.

従って, これらの手法は, システムに組み込まれ人手によらず実行されなければならず, さらに多数のコンテナの積み付け計画を立案する場合には適さないといえる.

4. 提案手法(2段階最適化)

本プロジェクトでは整数計画法を2段階に適用するという他のパッケージ・ソフトとは異なる手法を採用した. この手法を説明するにあたり, まず整数計画法[7][8]に関して簡単に説明し, その後に2段階に分けた整数計画法のモデルについて説明する.

4.1. 整数計画法

目的関数や制約条件を決定変数の1次式で表し問題を最適化する手法を線形計画法と呼ぶが, 整数計画法はその決定変数に整数制約を付加したもので,

その整数の性質を用いて分割不可能な資源を表現したり, 決定変数の0と1で採用, 不採用を表現するといった論理表現が可能となるなど多彩なモデル化が可能な非常に一般的で強力な手法である.

この整数計画法の問題を実際に解くソフトウェアである最適化ソルバーもILOG社[9]のCPLEX[10]等多数あり, またIBMが立ち上げたCOINプロジェクト[11]にもCBCというソルバーがある. 利用者は現実の問題を整数計画法のモデルに定式化できれば後はこれらのソルバーが実際の求解を行ってくれるため高度な最適化のアルゴリズムを知らなくても問題を解くことができる. ただし, 整数変数を利用して多様な制約を表現する定石に関する知識や, 定式化によって計算時間がどの程度かかるのかといった経験は重要といえる.

4.2. 最適化の概要

3次元の積み付け問題はNP困難な問題[1]であり, 何らかのヒューリスティックスに基づいた近似的な解法が必要である. 既存のアルゴリズムとしては一般的な“コンテナ”に対する積み付けを扱うものが多いが, ここでは海上コンテナ特有の性質に基づいた手法を提案する. 海上コンテナへの積み付けは複数のケースを積み重ねてフォークリフトで積み込むために複数のケースにまたがった形の段積みは行われない. また**図2**のようにその形状はL寸方向に極端に長く, ケースの荷姿も比較的大きいために, W寸方向, H寸方向への積み付けはある程度列挙可能である.

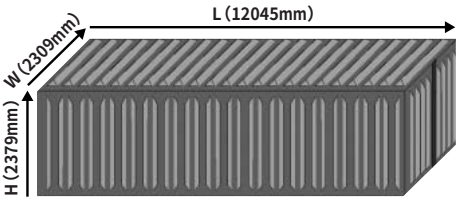

図2. 40ft海上コンテナのサイズ

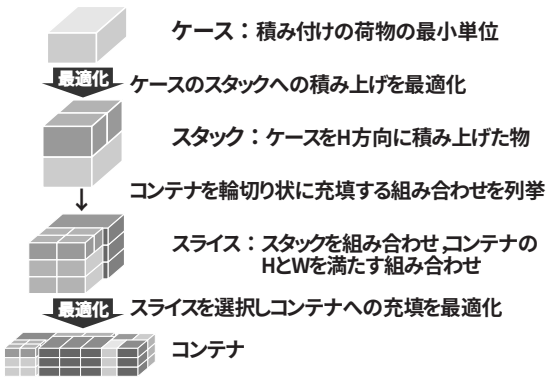

図3. 最適化のプロセスとデータ項目の概要

これらの知見に基づき**図3**のように, まずケースを段積みしたスタックの生成を第1段階目の最適化, そのスタックを組み合わせたスライスによるコンテナのL寸方向の充填(じゅうてん)を第2段階目の最適化として, 整数計画法の厳密解法が適用可能な2段階に分けて最適化する方法を採用した.

4.3. 第1段階の最適化

第1段階目の最適化として, スタックの生成を集合被覆問題[2]として定式化し, その床面積が最小となるよう最適化を行う.

4.3.1. スタックひな形の生成

与えられたケースをさまざまなL寸, W寸, H寸の形状および, 回転可能か否かの属性により分類する. この分類をケース種別と呼び, このケース種別を元にその積み上げ方であるスタックひな形を列挙する. この際に現実世界の段積みの制約, 例えば, 上段のケースは下段のケースの−10cm以内でなければならないといった制約を考慮してスタックひな形を生成する. この種類としては**図4**のように1〜4種類のケース種別を積み上げたものとし, さらに最上段のケース種別では横に並べて複数のケースを配置するタイプも列挙する. この場合にケース種別ごとの実ケースの数も考慮し, 無駄なひな形の生成を行わないようにする. 例えば1個しかないケース種別を2段積むようなひな形は生成しない.

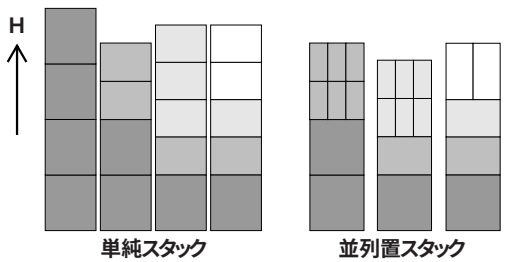

図4. スタックひな形のパターンの列挙方法

4.3.2. スタックひな形の選択

次に**図5**のように作成されたスタックひな形のいずれを幾つ選べば効率よくケース種別の数をカバーできるのかを最適化する.

このようにあらかじめ実行可能なパターンを多数列挙しておき, その中から最適なパターンを選択するという定式化は, 1次式では表せないような複雑な制約もパターンの生成時に考慮すればよいので一般的によく使われる定石の一つである.

1 問題規模が大きくなるに従って計算時間が指数関数的に増加し厳密解を求めることが現実的にはできない問題.

2 集合の要素を部分集合でカバーする組み合わせを最適化する問題.

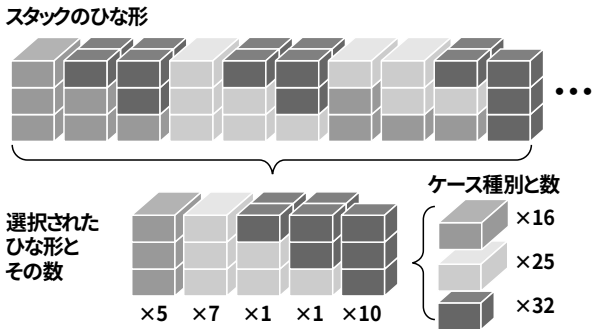

図5. スタックひな形の選択によるケース種別の被覆

4.3.3. 最適化の定式化と実行

この最適なスタックひな形の選択を整数計画法で定式化すると次のようになる. 目的関数はスタックひな形の底面積の総和として, これを最小化する.

決定変数：	
x_i	スタックひな形iを選択する数(整数変数)
定数値：	
n_{ij}	スタックひな形iがケース種別jを含む数
N_j	ケース種別jの総数
s_i	スタックひな形iの底面積
制約条件：	
$\sum_i n_{ij}x_i \geqq N_j$	ケース種別の被覆制約
目的関数：	
$z = \sum_i s_i x_i$	総床面積最小化

これらの定式化に基づき整数計画法ソルバーで実際の最適化を行う.

4.3.4. ケースの引き当てとスタックの生成

最適化の結果得られたスタックひな形の種類とその数により実ケースの引き当てを行い実際のスタックを生成する.

4.4. 第2段階の最適化

第2段階目の最適化はナップサック問題[3]として定式化し, スライスがカバーするスタックを最大化するように最適なスライスを選択するという最適化を行う.

4.4.1. スライスひな形の生成

第1段階目の最適化で生成されたスタックをL寸, W寸および回転可能か否かで分類する. これをスタック種別と呼び, このスタック種別をさまざまに組み合わせ, コンテナを輪切り状に充填するスライスひな形を**図6**のように列挙する. この列挙においても無駄なひな形を削減することは最適化の実行時間短縮に非常に重要である. 今回は大量に存在するタイプ3に関しては最大数を500個, タイプ4,5,6に関しては100個と制限した. この選択においてはスライスが被覆するスタックの面積をスライスのLで割りL寸当たりの充填率を計算しさらにこれを再度Lで割った値を選択の評価値とした. この理由は, L寸が大きいスライスほど充填率が高くなりやすい傾向を補正するためと, ナップサック問題として充填する場合にはL寸の大きいものは有効に働きにくいためである.

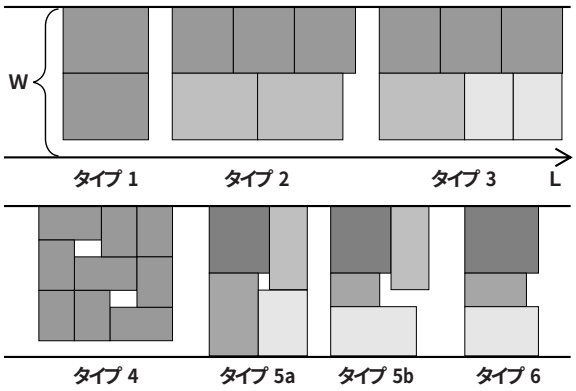

図6. スライスひな形のパターンの列挙方法

4.4.2. スライスひな形の選択

次に列挙されたスライスひな形により, **図7**のようにコンテナのL寸に収まる範囲内でより多くのスタックをカバーするように, どのスライスひな形を幾つ選択すればよいかを最適化する.

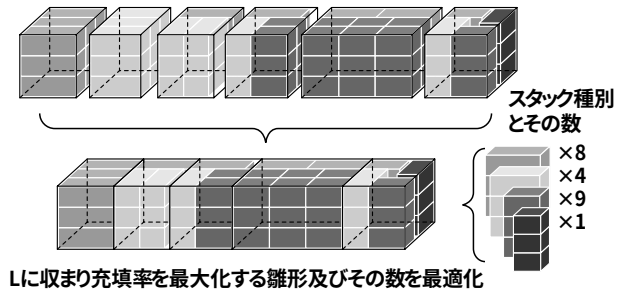

図7. スライスひな形の選択によるスタック種別の被覆

4.4.3. 最適化の定式化

この最適なスライスひな形の選択を整数計画法で定式化すると次のようになる. これは容器に物を詰めるナップサック問題のバリエーションとなり, 目的関数として, 各スライスがカバーするスタックの体積を最大化する問題となる.

決定変数：	
x_i	スライスひな形iを選択する数(整数変数)
y_i	スライスひな形によって過剰に積載された

3 一定容量のナップサックに価値の高い荷物をいかに詰め込むかを最適化する問題.

スタック種別jの数

定数値：

L	コンテナのL寸
l_i	スライスひな形iの長さ
n_{ij}	スライスひな形iがスタック種別jを含む数
N_j	スタック種別jの総数
v_j	スタック種別jの平均の体積
u_i	スライスひな形iの平均の体積 $=\sum_j n_{ij}v_j$
k	二次的な目的関数を示す係数

制約条件：

$$\sum_i l_i x_i \leqq L \quad (1)$$

コンテナL寸制約

$$\sum_i n_{ij}x_i - y_j \leqq N_j \quad (2)$$

スタック種別の過剰積載を補償する制約

目的関数（最大化）：

$$z = \sum_i u_i x_i - \sum_j v_j y_j - k\sum_i l_i x_i \quad (3)$$

積載体積最大化および積載L寸最小化

スライスひな形iをコンテナに選択する数を決定変数とすることにより，コンテナに積載されるL寸の条件は制約条件（1）として表現され，積載されるスタックの体積は$\sum_i u_i x_i$と表されるのでこの体積を最大化する．

ただし，スライスがカバーするスタックには限りがあるために，スライスが過剰にスタックをカバーすることを防ぐ必要があり，その過剰にカバーされたスタックの数をy_jという決定変数で表現する．これにより目的関数から過剰にカバーされたスタックの体積である$\sum_j v_j y_j$を引くことにより正味にカバーされた体積を最大化することができる．このy_jが過剰にカバーされたスタック種別の数を表すようにするために（2）の制約を追加する．$\sum_i n_{ij}x_i$がスタック種別jの数N_jを上回った場合にはy_jが正の値をとることにより，積載されたスタック種別がN_jとなるように補償する．

ここで，制約式（2）が不等式になっているのは，スタックが多数ありカバーしきれない場合を表すためである．

また，目的関数の第3項はスタックをカバーしない不必要なスライスひな形が採用されないようにするとともに，なるべく奥に効率よく詰めて積載することを目的関数に反映するためのものである．

4.4.4. コンテナの突起および鴨居の定式化

次に，先に述べたコンテナの奥の突起と入り口の鴨居を考慮する定式化について述べる．

定数値：

f_i	スライスひな形iの入り口への配置可能性
b_i	スライスひな形iの最奥への配置可能性
ML	スタック種別の最大の長さ
PL	コンテナ奥隅の突起物の長さ

制約条件：

$$\sum_i (l_i - ML f_i)x_i \leqq L - ML \quad (4)$$

入り口への配置制約

$$\sum_i (l_i - PL b_i)x_i \leqq L - PL \quad (5)$$

最奥への配置制約

$$\sum_i (l_i - (ML+PL)(f_i \mid b_i))x_i \leqq L - (ML+PL) \quad (6)$$

いずれにも配置不可の場合の制約

それぞれのスライスひな形に対して奥の突起に干渉することなく奥に配置できるか，手前に配置することができるかを属性として検査しておき，f_i, b_iとして表すものとする．

選択されたx_iにf_iが1すなわち入り口に配置可能なものがn個あったとすると（4）は

$$\sum_i l_i x_i \leqq L + (n-1)ML \quad (4')$$

となって，nが0の場合には

$$\sum_i l_i x_i \leqq L - ML \quad (4'')$$

となり，左辺の合計の長さが$L-ML$以下という式となり，nが1以上の場合には（1）よりL以下となることが分かる．これにより選択されたすべてのスライスひな形が鴨居に干渉する場合にはコンテナの全長をML分だけ短く制約することができる．ここで本来MLは積載されたスタックに応じて決定されるべきものではあるが安全をみてすべてのスタックの最大の長さを用いている．

同様にコンテナ奥の突起に関しても（5）で突起に干渉するスライスひな形のみが選択された場合にはコンテナの全長をPL分だけ短く制約することができ，（6）により突起にも鴨居にも干渉するスライスひな形しかない場合に$ML+PL$分だけ短く制約することができる．

4.4.5. スタックの引き当てとスライスの生成

最適化の結果得られたスライスひな形の数により実スタックの引き当てを行い実際のスライスを生成し，それを順次コンテナに割り当てる．

また，ここで得られたコンテナに含まれるスタックが残りのスタックからそっくり同等のスタックが引き当てられる場合には繰り返して引き当てを行い複数のコンテナを生成する．ケースの数が多い場合ほどこの同じコンテナの複数の引き当てが起こる可能性が高く高速な処理が可能となる．

4.4.6. 複数コンテナへの繰り返し

上記の最適化でコンテナに積載できなかった残りのスタックに対して，スライスひな形の生成から再度繰り返し最適化を実行し，積み残しのスタックがなく

なるまで繰り返す.

4.5. 最適化全体の流れ

以上, 全体の流れを図に表すと**図8**のようになる.

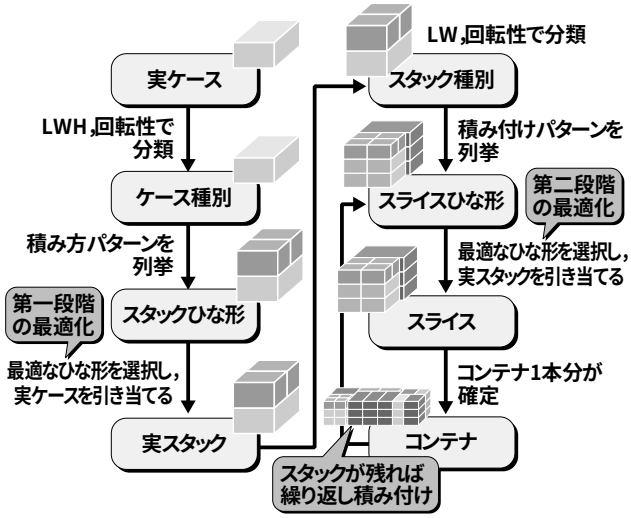

図8. 最適化全体の流れ

5. 提案手法の適用評価

以上の基本のコンテナ積み付け最適化エンジンに加え, お客様固有の機能を拡張したエンジンで実際のデータを最適化した結果が**表1**である. 計算単位は, 輸出先などによって分けられた単位で, この単位内で最適な積み付けを行う. また, ここで端数とはこの計算単位内の最後のコンテナを表しこの部分では充填率は低くならざるを得ないが, それ以外の部分では最適化エンジン本来の機能が端的に表れる. この端数以外においては体積充填率で81.3%, L寸の充填率においては98.3%と非常に高い充填率を実現することができた. 計算時間も13.6分で1時間以内という要件を十分満たすものである.

表1. 最適化の実行結果

計算単位数	193
ケース数	20667
コンテナ数	479
体積充填率	71.1%
L寸充填率	86.3%
体積充填率(端数外)	81.6%
L寸充填率(端数外)	98.4%
計算時間(分)	13.6
実行環境 CPU Xeon 2.8GHz 2way Memory 1GB, OS Windows® XP COIN CBC(2005/06/09)	

また, 本手法の特長を確認するために, 実験的に積み付けるケースを2倍に増やしたデータを作成し計算を行ったところ計算時間は19.6分であった. これは元の1.44倍の計算時間でしかなく, 本手法が個々のケースを扱うのではなく, その種別やひな形によって最適化を行っているためにケース数の増加に計算時間が直接影響しないという特長が示されたものといえる.

最適化の結果得られた積み付けの様子を**図9**に示す. これらの例でさまざまなスタック, スライスによりL寸, W寸, H寸のそれぞれの方向を最大限に生かし充填率を高めていることが分かる.

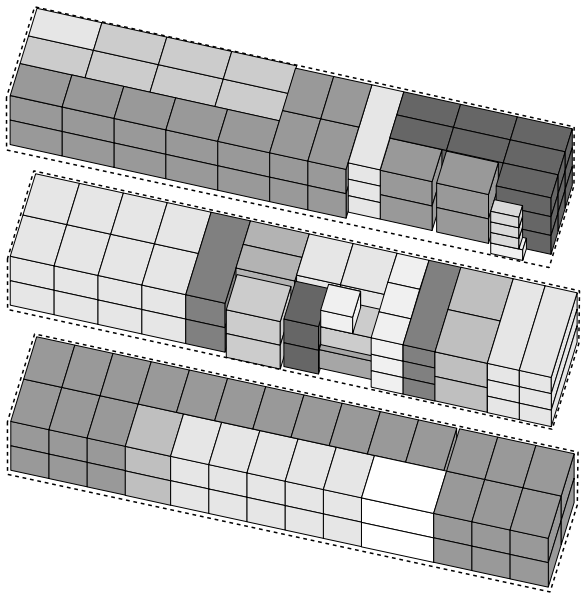

図9. 積み付け結果の実例

6. おわりに

以上のように, 現実に行われている業務から高さ方向の最適化と幅・長さ方向の最適化に分けて2段階に最適化を行うというモデルを作成し, それぞれを整数計画法を用いて最適化することによって積み付け計画を短時間に立案することができた.

この中で, 通常の3次元の積み付けのみならずコンテナ内の突起や鴨居に関する制約や安定性に関する制約もモデル化することができ, 現場で使える積み付け結果を作成することができた.

また, 市販のパッケージ・ソフトのようにコンテナ積み付け最適化専用のエンジンを一から作りこむ場合と比べて最適化アルゴリズム自体を作成する必要がないために非常に短期間に開発することができ, かつ整数計画法の実績のある最適化アルゴリズムを使用することができ十分高速な実行速度を得ることができた.

今回のプロジェクトで最大のボトルネック工程と考えられた積み付け計画部分を自動化し納期回答プロセスの中に組み込み, その後の人手による計画全体の確認処理を含めても数日間で完了できるシステムを構築でき, プロセスのリードタイム短縮を実現できた.

オンデマンドの時代にビジネスの変革が求められる中で, 今後, 最適化技術がさまざまなビジネス・プロセスの中に組み込まれ「最適化サービス」という形で使用されていくと考えられる. これによって今まで人の経験に頼っていた処理がごく短時間に行うことができるようになり, 月次から週次, 週次から日次といったプロセス短縮や, センス&レスポンド型のシステムのように数分や数秒で意思決定を行い即時の応答が可能となっていくものと考える.

筆者は最適化技術をもって, このような現実のビジネスの変革に貢献して行きたいと考えている.

謝辞

本プロジェクトの機会を与えていただいたお客様, プロジェクトに参加されたすべての方々に感謝いたします.

参考文献

[1] VRP, http://www.ibm.com/jp/software/gis/new_vrp.html
[2] J.A. George and D.F. Robinson, "A heuristic for packing boxes into a container", *Computer and Operations Research*, 7：147-156, 1980.
[3] N. Ivancic, K. Mathur und B. B. Mohanty, "An integer programming based heuristic approach to the three-dimensional packing problem", *Journal of Manufacturing and Operations Management*, 2（4）：268-298, 1989.
[4] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem", *Operations Research*, 48（2000）pp. 256--267.
[5] D. Pisinger, "Heuristics for the container loading problem", *European Journal of Operations Research*, 141, 2002, pp. 382-392.
[6] T. Osogami, "Approaches To 3D Free-Form Cutting And Packing Problems And Their Applications：a Survey", IBM Research Report, 1998
[7] 今野 浩 他, "整数計画法と組合せ最適化", 日科技連出版社, 1982
[8] 久保 幹雄 他, "応用数理計画ハンドブック", 朝倉書店, 2002
[9] ILOG, http：//www.ilog.com/
[10] CPLEX, http：//www.ilog.com/products/cplex/
[11] COIN Project, http：//www.coin-or.org/

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
アーキテクト

米沢 隆 Takashi Yonezawa

[プロフィール]
1989年, 日本IBM入社. 1999年より配送経路最適化プログラムVRPの開発およびサービスに従事し, その後さまざまな最適化プロジェクトを実施.
2001年より4年間, 湘南工科大学の非常勤講師としてオペレーションズ・リサーチおよび数理計画法を担当.
日本オペレーションズ・リサーチ学会会員.
yonezat@jp.ibm.com