解存在不可能性の証明法を用いたコンテナプリマーシャリング問題の効率的な計算

小池　英勝[1]

アブストラクト：本論文は，組み合わせ最適化問題の一つであるコンテナプリマーシャリング問題（CPMP）の最適解を効率的に得るための手法を提案する．この手法は，CPMP の計算中に現れる特定の状態に対して，ある特定の解集合の存在不可能性を判定する．この判定結果を用いることで特定の問題の計算効率を改善できることを示す．

キーワード：組み合わせ最適化，コンテナプリマーシャリング問題，最適化，不可能性証明，ロジスティクス．

1　はじめに

本研究の目的は，組み合わせ最適化問題の一つであるコンテナプリマーシャリング問題（Container Pre-Marshalling Problem, CPMP）の最適解を計算機で高速に解くことである．CPMP は，問題の規模が大きくなると，その計算量は大幅に増えるため，これまでに知られている解法では，実用的なサイズの問題に対して現実的な計算時間で最適解を求めることができるケースは限られている．既存の解法の多くは，ヒューリスティックスや近似アルゴリズムを用いて計算量を抑えながら（最適解ではなく）近似解を求める．最適解を得ようとすると，計算量の削減が難しくなり，現状の計算機で（最適なもの以外を含めて）解を得ることが難しくなる．最適解を求める研究が少ないことは，この問題の難しさを表していると考えられる．実用規模で問題の最適解を得る試みは著者が知る限りほとんどない．著者はこれまで，CPMP の最適解をできるだけ高速に求めるための手法とそれに基づくプログラムを開発してきた（小池，2017; Koike, 2017）．本研究の長期的な目的は，流通上のコンテナ混雑問題を改善すると同時に，実用的な規模の組み合わせ最適化問題に対する新しいアプローチを提案することである．

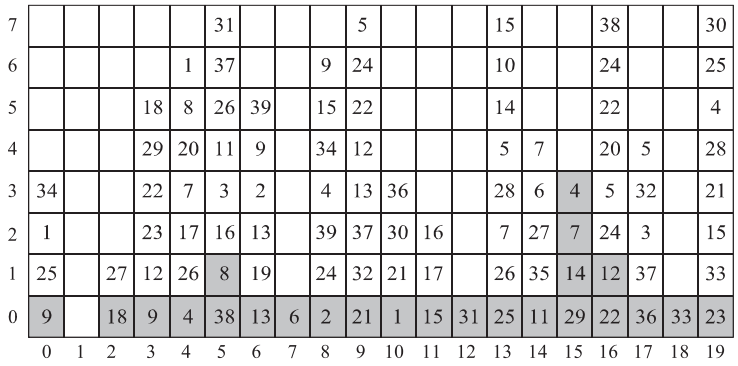
以降，本論文は以下の様に構成される．２節で，コンテナプリマーシャリング問題を概説し，本研究の特徴である最適性の保証について説明する．３節で，本論文で提案する証明方法の動機とその詳細を説明するために，CPMP とその関連項目を定義する．４節で，本研究の動機と提案する方法の必要性について述べる．５節で，具体的な問題の解存在不可能性を証明する．６節で，証明方法を一般化し，自動化するための証明手続きを定義する．７節で，本論文が提案する方法を実装した実験プログラムの性能について述べる．８節でまとめる．

2　コンテナプリマーシャリング問題

2.1　CPMP の概要

コンテナプリマーシャリング問題（CPMP）とは，コンテナ埠頭などで起こる流通混雑問題解決のための，コンテナ並べ替え問題の一つである．コンテナが船に搬入され輸送される前に，コンテナヤードに一時集積される．コンテナの移動に用いるクレーン，スペース，そして，安全上等の制約から，コンテナは Bay と呼ばれる幅と高さが決められた 2 次元空間でのみ移動可能である．集積されたコンテナは，船に搬入される前に，搬入の優先度が決まる．搬入時は，その優先度に従って搬入順序が決定される．もし，高い優先度を持つコンテナが低い優先度を持つコンテナの下にある場合，上にあるコンテナを同じ Bay の空いている場所に退避した後に目的のコンテナを取り出さなくてはならない．船積み時間をできるだけ短くするために，コンテナの優先度が決まってから船が到着するまでの時間を使って，低い優先度のコンテナが高い優先

[1] 札幌学院大学 経済学部；koike@sgu.ac.jp.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7						31				5				15			38			30
6					1	37			9	24				10			24			25
5				18	8	26	39		15	22				14			22			4
4				29	20	11	9		34	12				5	7		20	5		28
3	34			22	7	3	2		4	13	36			28	6	4	5	32		21
2	1			23	17	16	13		39	37	30	16		7	27	7	24	3		15
1	25		27	12	26	8	19		24	32	21	17		26	35	14	12	37		33
0	9		18	9	4	38	13	6	2	21	1	15	31	25	11	29	22	36	33	23

図１　BF28-17の初期状態　色のないコンテナは必ず移動する

(10, 1) (5, 12) (10, 12) (19, 12) (17, 7) (9, 7) (17, 18) (17, 15) (0, 1) (4, 15) (0, 15)
(2, 12) (3, 2) (0, 12) (19, 12) (8, 0) (9, 12) (4, 0) (14, 0) (14, 0) (19, 7) (11, 2) (8, 2)
(8, 1) (13, 2) (8, 7) (3, 18) (19, 18) (14, 18) (3, 18) (9, 18) (10, 18) (9, 2) (13, 2) (3, 1)
(19, 1) (4, 1) (11, 1) (19, 11) (13, 11) (9, 11) (3, 11) (13, 0) (4, 3) (4, 1) **(10, 15)** (6, 10)
(8, 10) (16, 10) (5, 10) (9, 10) (17, 10) (14, 17) (6, 2) (19, 17) (9, 17) (6, 7) (13, 17)
(13, 3) (4, 17) (6, 9) (6, 19) (5, 10) (13, 10) (5, 14) (8, 13) (16, 13) (5, 4) (16, 13)
(16, 13) (16, 0) (16, 17) (5, 19)

図２　BF28-17の最適解の一例

度のコンテナの上に一つもない状態(クリーンな状態)に並べ替えておけば，船積みの時間を最短にできる．CPMP とは，優先度が考慮されずに集積されたコンテナの初期の状態から，クリーンな状態に並べ替えるための最短の手順を発見する問題である．

図１は，Bay の初期状態の一例を表す．枠の中にある数字はその数字が示す優先度のコンテナがあることを表す（０が最優先）．初期状態では，船積みのときの順番が考慮されていない状態でコンテナが Bay に集積されている．図中の影のついたコンテナは偶然優先度順に積まれていることを表し，並べ替えなくてもよい可能性がある．影がついていないコンテナは，必ず並べ替える必要がある．

図３は，図１のコンテナを図２で示す最適解で並べ替えた結果である．図３は優先度の高い順に上に位置しており，クリーンな状態である．各図の詳細は，３節以降で述べる．

2.2　CPMP の特性と既存の研究

これまでの CPMP の研究は，CPMP が NP 困難に属する問題特有の性質を持つことから，近似解を求めるアプローチがほとんどであった．その特性は以下の通りである．

・一つの解の候補の生成を低コストで行える．
・一つの解の候補の評価を低コストで行える．
・解の候補の数が問題によっては，現状の計算機では扱えないほど多くなることがある．

以上の特性から，何らかの方法で解の候補を現実的な数に減らしてそれらを評価して解を得る，という方針は妥当であるように感じられる．しかし，最適解を得ようとすると，解の候補を減らすことが難しくなる．

著者が知る限り，最適解を扱う文献は少なく，その少数の文献も，実用規模と比較して小規模な問題しか扱えないか，ヒューリスティックアプローチとの比較に用いられるものだった．2015年には，最適解アプローチを扱った文献（Zhang, *et al.*, 2015）が現れ，文献中で扱える問題の規模が実用規模であると主張されていたが，ヒューリスティックアプローチと比較すると，扱える規模はまだ小さいといえる．

ヒューリスティックアプローチでは，その性質上最適解を得られる保証がなくなり，特定の問題では解が得られないことがあり，このような制限は実用的にも問題になりうる．ヒューリスティックアプローチを採用すると計算効率の改善と，解の精度にトレードオフの関係がおこる．始めに，精度は低いが解が簡単に得られる状態から，工夫して解の精度を最適に近づけようとする．著者が問題だと考えていることは，多くのヒューリスティックアプローチが，その手続きと実験

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7		17	9								26		24			1			21	
6	5	17	10								26		25			1		24	22	
5	5	20	12					2			37		25			1		26	22	
4	6	21	15					4			37	12	27	20		3		28	27	
3	7	23	15					4			37	13	30	22		4		32	28	
2	8	34	16	7				5			38	14	30	24		7		33	29	16
1	9	34	18	7	3	8		5		13	39	15	31	24	11	14	12	35	32	19
0	9	36	18	9	4	38	13	6	2	21	39	15	31	25	11	29	22	36	33	23

図 3　最適解によって得られるクリーン（fixed）な Bay

結果・性能比較について議論されるが，高速化の代償として失われる最適性について，ほとんど考察されていないことである．

本研究では，解の候補の数を減らす手法を採用するときに，最適性を損なわないもののみを採用するという条件を厳守する．これによって，解の最適性を保証しながらも，効率的に計算できることを示した（小池, 2017; Koike, 2017）．本論文では，解の最適性を保証しながら，さらなる効率化のための新しい方法を導入する．

3　CPMP の定義

本節では，CPMP を本研究で厳密に議論するための定義を行う．この定義は（小池英勝，2017）で定義したものに本研究に必要な定義を加え拡張したものである．また，本研究に必要ない部分は省略する．

3.1　前提条件

CPMP はコンテナの並べ替え問題であるが，類似の問題, 例えば, コンテナリロケーション問題（Container Relocation Problem, CRP; Kim & Hong, 2006）などと明確に区別するためにコンテナに関する前提条件を以下に明示する．

① コンテナは，同一の Bay でのみ移動可能である．並べ替え中にコンテナの総数は変わらない．
② 全てのコンテナは，同じサイズである．
③ 全てのコンテナには，前もって優先順位を示す番号が与えられている．
④ 異なるコンテナが同じ優先順位を持つことが許される．
⑤ コンテナのスタック間の移動（すなわち，横の移動）のコストは無視できる．

前提条件⑤は，クレーンによるコンテナの上下の移動が，横の移動よりもコストが大きいことから仮定される（Bortfeldt & Forster, 2012）．前提条件⑤によって，Bay をスタックのマルチセットとみなすことが可能になり，本研究ではこの概念によって，計算量を削減することに成功した．マルチセットを用いた計算量削減手法の詳細は文献（Koike, 2017）で述べた．

3.2　要素の定義

上の前提条件を基にして，CPMP の構成要素を定義し，更にそれを用いて，問題を定義する．また，本論文の動機の説明と本論文で導入する証明手順の定義にも用いる．

（**Bay の定義**） コンテナの移動について議論する場合，Bay は，スタックの列である．
Bay を構成するスタックの数を S で表す．

（**スタックの定義**） スタックは, スロットの列であり，LIFO（Last In First Out）でコンテナを出し入れする．同一の Bay を構成するスタックは，全て同じ長さを持ち，それを H で表す．H を Bay の高さと呼ぶ．Bay 中のスタックを 0 から始まる整数で識別することがある．0 は Bay 中の最左のスタックを表す．スタックを識別する整数をスタック id と呼ぶ．

（**スロットの定義**） スロットは，コンテナを高々 1 個格納できる．スロットがコンテナを持たないとき，スロットは空であるという．スタックの長さ H は，スタックが持つスロットの個数である．

（**コンテナの位置**） Bay は，S×H の 2 次元配列ともみなせる．Bay 中のコンテナの位置を（s，h）で表

す．ここで，sはスタックidであり，hは，コンテナの縦の位置を表す．hは，0から始まる整数で，0はスタックの底に位置することを表す．$0 \leq s < S$，$0 \leq h < H$である．スロットの位置も，コンテナと同様の表現で表す．コンテナがスタックの最も上に位置するとき，そのコンテナをトップコンテナであるという．

（**コンテナ下のスロットの制約**） コンテナが（s，h_t）に位置するとき，その下（s，h_b），$0 \leq h_b < h_t$にある全てのスロットは空ではない．

（**コンテナの優先度とグループ**） 全てのコンテナには，優先度を表す整数pが与えられる．最低の優先度のコンテナにPが与えられるとき，$0 \leq p \leq P$である．コンテナが優先度0を持つとき，最高の優先度を持つ．コンテナにpが与えられているとき，コンテナはグループpに属するという．異なるコンテナが同じグループに属することがある．優先度pを持つコンテナをc(p）と表すことがある．

（**well-placed**） コンテナcが以下の条件のいずれかを満たすときcはwell-placedであるという．

・cがスタックの底に位置する．

・cがp_1に属し，かつ，p_2に属するwell-placedなコンテナの直上に位置し，かつ，$p_1 \leq p_2$である．

スタックが空の場合，または，スタックが格納する全てのコンテナがwell-placedのとき，そのスタックはwell-placedであるという．

あるスタック中のwell-placedなコンテナの中で最も上に位置するものを「well-placedなトップコンテナ」という．あるスタック中のwell-placedなコンテナの数をN_wで表す．

（**badly-placed**） コンテナがwell-placedでないときbadly-placedであるという．スタックがwell-placedでないとき，そのスタックはbadly-placedであるという．

（**fixed**） コンテナcが以下の条件のいずれかを満たすときcはfixedであるという．

・cが最も低い優先度を持ち，かつ，well-placedである．

・cがwell-placedで，かつ，cより優先度の低いコンテナが全てwell-placedである．

・cがwell-palcedで，以降のcの移動が冗長であると明らかになったとき．

コンテナがfixedである場合，そのコンテナは決して移動しない．スタックが格納するコンテナが全てfixedであるとき，そのスタックはfixedであるという．Bay中の全てのコンテナがfixedのとき，そのBayはfixed，または，クリーンであるという．

（**コンテナの移動**） コンテナの移動とは，スタックの最上位にあるコンテナを別のスタックの最上位に移動することである．スタックS_bにあるコンテナがスタックS_aに移動したとき，この移動をスタックidのペア（S_b，S_a）で表す．$S_b \neq S_a$である．m_i（$0 \leq i \leq N$）を移動とするとき，m_0，...，m_Nを移動の列と呼ぶ．移動の列をBayに適用するとは，m_0から順にm_Nまでその内容に従ってBay中のコンテナを移動することである．

3.3　CPMPの定義

（**CPMP**） コンテナプリマーシャリング問題（CPMP）とは，与えられたBayをfixedにするための最短のコンテナの移動の列を求めることである．この最短の移動の列を最適解と呼ぶ．

（**CPMPの解**） 与えられたBayをfixedにするコンテナの移動の列をBayの解と呼ぶ．解がN個の移動で構成されるとき，その解を長さNの解という．

（**下界**） あるBayの下界Lは，そのBayの解の長さが少なくともLであることを表す．

3.4　コンテナの移動のクラスとBG解の定義

文献（Bortfeldt & Forster, 2012）では，コンテナの移動をBG，GG，BB，GBという4つのクラスに分類している．本論文でも，コンテナの移動に対してこのクラス分けを用いる．BGに属する移動はbadly-placed（B）なコンテナをwell-placed（G）の状態に移動する．他のクラスも同様にして2つの頭文字でその意味が表されている．BGのクラスに属するコンテナの移動をBG移動と呼ぶことがある．他の3つのクラスに関しても同様である．

（**BG解**） 解がBG移動のみで構成されるとき，その解をBG解と呼ぶ．

4　本研究の動機と方針

著者はこれまでに，文献（Bortfeldt & Forster, 2012）に掲載された，大規模な問題に対して最適解を得る取り組みを行ってきた．本論文では，この問題のテストケース BF28の20題のうちの一つである cpmp_20_8_96_39_72_17.bay について考察する．この問題は20題あるテストケース BF28の17番目の問題なので，以降，BF28-17と呼ぶ．図 1 は，BF28-17の Bay の初期状態を表す．この Bay はコンテナを最大 8 個格納できるスタックを20個持つ．図中の枠はスロットを表す．枠の中に数字を書くことで，そのスロットに数字が表す優先度を持つコンテナが配置されていることを表す．空白の枠は，コンテナが配置されておらずスロットが空であることを表す．Bay の左側の数字は，スロットの縦の位置を表すのに用いる． 0 はスロットがスタックの底に位置することを表し， 7 は最上位に位置することを表す．Bay の下側の数字はスタック id で， 0 が最左のスタックを表し，19が最右のスタックを表す．この Bay は，コンテナの総数が96個で，そのうち72個が badly-placed である．灰色のコンテナは，well-placed である．著者の開発したシステムでは，この Bay の解として，長さ73の解を得ることができる．その解の一例を図 2 に示す．この解は72回の BG 移動と， 1 回の GG 移動で構成される移動の列である．図 2 では GG 移動を太字で表した．図 2 の解を図 1 の Bay に適用して得られた結果を図 3 に示す．解を得るまでの時間は現状のシステムで 1 秒未満である．この問題の難しさは，得られた解の最適性の証明である．著者のシステムでは，この問題の初期状態の下界を72と算出するので，上記の解の最適性を証明するためには，如何なる72回のコンテナの移動列も与えられた Bay を Fixed にできないことを示す必要がある．しかし，文献（小池，2017; Koike, 2017）で述べた方法だけでは，計算に時間がかかり最適性を証明することはできない．

BF28-17は72個の Badly-placed のコンテナを持つ．これを72回の移動で fixed にするためには，全ての移動が BG である必要がある．これによって， 4 種類ある移動のうち 3 種類を排除できるが，それでも，可能な移動を計算し尽くすには時間がかかりすぎる．よって，可能な全ての移動の列をひとつずつ生成しコンテナを動かしながら評価するのではなく，できるだけ Bay の状態のみで，72回の BG 移動では fixed に移動できないことを証明することを考える．

5　問題 BF28-17に関する解の最適性の証明

本節では問題 BF28-17の最適解が73個の移動で構成されることを証明する．

5.1　問題の定義

問題の条件とその仮定のもと，証明しようとしている命題をまとめる．

（pc1） CPMP として BF28-17が与えられている．
（pc2） BF28-17の初期状態の下限が72と計算されている．
（pc3） Badly-placed なコンテナの個数は72個である．
（pc4） 長さ73の解が既に得られている．

上記 4 つの条件で，
命題 0「BF28-17の最適解は長さ73である」
が真であることを証明する．

上で導入した BG 解の概念を用いて命題 0 に論理等価な命題 1 を定義する．
命題 1「BF28-17に関する長さ72の BG 解は存在しない」

ただし，命題 1 も条件（pc2），（pc3），（pc4）が与えられているとする．

5.2　証明

条件（pc2）より，71個以下の如何なる移動列にも解は存在しない．条件（pc2）と（pc3）より，長さ72の解が存在するならば，それは BG 移動のみで構成される．このことから，BG 移動のみでは Bay を fixed にできないことを証明すれば，命題 1 が真であることを証明できる．

Bay に長さ72の BG 解が存在しないことを証明するには，BG 移動できないコンテナを一つ発見できればよい．BG 移動できないことを証明する前に，コンテナが BG 移動できる条件を考える．全ての badly-placed なコンテナは BG 移動する必要があるので，以下に示す条件（bc1）が必要である．以降，可能な BG 移動には，以下の条件（bc1）が満たされていることを前提とする．

（bc1） あるコンテナの BG 移動が，他の badly-placed なコンテナの BG 移動を不可能にしない．ここで，あるコンテナの BG 移動が

不可能とは，そのコンテナの可能な BG 移動が一つもないことである．

コンテナがトップコンテナである場合の条件を考える．優先度 p を持ち，スタック s の bad-placed なトップコンテナ c(p) について，以下の条件のいずれかが成り立てば，c(p) は BG 移動可能である．

(bc2)　空のスタックがある．

(bc3)　s 以外の well-placed なスタックがあり，そのスタックの空のスロットが 1 個以上で，トップコンテナの優先度は p 以上である．

(bc4)　s 以外の badly-placed なスタックがあり，その well-placed なトップコンテナの優先度は p 以上であり，全ての badly-placed なコンテナは BG 移動可能である．

次に，優先度 p のコンテナ c(p) がトップコンテナでない場合の，BG 移動可能な条件を考える．

(bc5)　条件 (bc2)，(bc3)，(bc4) のいずれかを満たし，かつ，c(p) の上に位置する全てのコンテナが BG 移動可能である．

上の条件から，優先度 p のコンテナ c(p) が BG 移動不可能な条件を考える．c(p) は以下の条件 (bc6) または (bc7) を満たせば BG 移動不可能である．

(bc6)　条件 (bc2)，(bc3)，(bc4) をいずれも満たさない．

(bc7)　条件 (bc2)，(bc3)，(bc4) のいずれかを満たし，かつ，c(p) の上に位置する 1 個以上のコンテナが BG 移動不可能である．

上記の条件を使って命題 1 を証明する．

まず，BF28-17の最も優先度が低いコンテナに注目する．図 1 より39が最も優先度が低い（0 が最優先）．優先度39のコンテナ c(39) はスタック 6 とスタック 8 にある．スタック 6 中の c(39) はトップコンテナであり，スタック 1 が空なので条件 (bc2) が成り立ち，BG 移動可能である．次に，スタック 8 中の c(39) について考える．このコンテナには，その上に 4 つのコンテナがあるので，それらの BG 移動可能性を考える．

c(39) 上の 4 つのコンテナのうち最も優先度が低いのは c(34) だから，c(34) の BG 移動可能性について考える．c(34) はその上に 2 つのコンテナを持つので，条件 (bc7) を考える．条件 (bc2) について，スタック 1 が空であるが，このスタックは，c(34) の下に位置する c(39) に予約されている．c(34) をスタック 1 に移動すると c(39) の BG 移動が不可能になるので，条件 (bc1) が満たされず，c(34) を BG 移動するためにその移動先をスタック 1 にできない．よって，c(34) の移動先候補はスタック 1 以外の条件 (bc2)，(bc3)，(bc4) のいずれかを満たすスタックになる．スタック17が唯一の移動先の候補である（条件 (bc4) を満たす）．スタック17は badly-placed なので，同スタック中の badly-placed なコンテナについて考える．スタック17中の badly-placed なコンテナで，最も優先度が低いのは c(37) である．よって，c(37) の BG 移動可能性について考える．c(37) の BG 移動での移動先の候補は，スタック 1 のみだが，スタック 8 の c(39) に予約されて，条件 (bc1) により移動できない．よって，スタック17中の c(37) の BG 移動不可能性が証明された．BG 移動不可能なコンテナが見つかったので，BG 以外の移動が少なくとも一つ必要である．よって，長さ72の BG 解を構成することは不可能であるので，命題 1 が証明された．

6　証明の自動化

本節では，5 節の証明を一般的な問題に適用できるように一般化し，自動化する．以降，この証明を「BG 解の存在不可能性証明」と呼ぶことがある．

6.1　コンテナによるスタックの予約

5 節の証明過程で現れた，コンテナによるスタックの予約の概念について考える．図 4 は，この概念を表している．図中の矢印が，あるコンテナが BG 移動するときのもとの位置と行き先を示している．矢印についている丸で囲まれた数字は証明過程で予約が起こった順番を示している．図 4 が BF28-17の BG 移動について示していることは以下の通りである．

(m1)　スタック 8 中の c(39) が BG 移動するには，スタック 1 に移動するしかない．

(m2)　スタック 8 中の c(39) を BG 移動するには，それより先に，スタック 8 中の c(34) を BG 移動させる必要がある．

(m3)　スタック 8 中の c(34) を BG 移動するには，スタック17に BG 移動させるしかない．

(m4)　スタック 8 中の c(34) を BG 移動するには，それより先に，スタック17中の c(37) を BG 移動させる必要がある．

(m5)　スタック17中の c(37) を BG 移動するには，スタック 1 に移動するしかない．

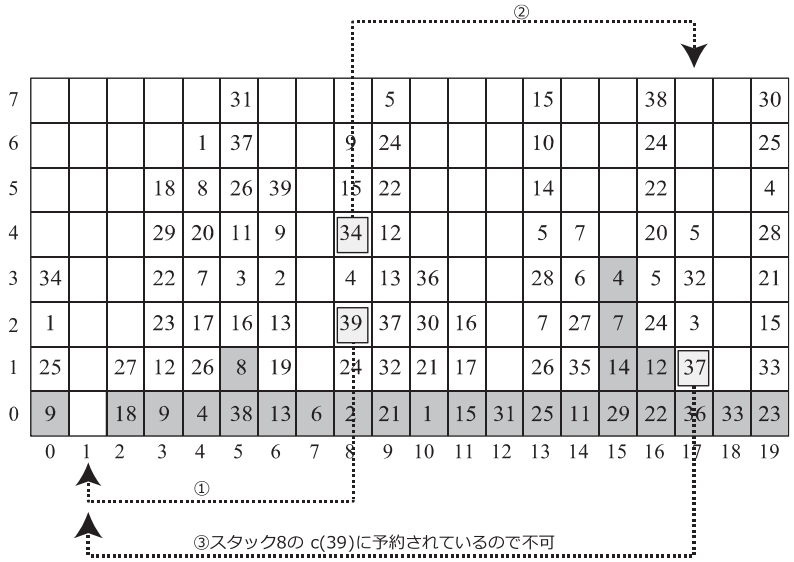


図 4　BF28-17の BG 移動不可能性の証明手順とスタックの予約の概念

図中の矢印①は上の（m1）を表す．図中の矢印②は（m2）と（m3）を表す．図中の矢印③は（m4）と（m5）を表す．

①と②矢印の始まりにある各コンテナは次の矢印のコンテナが移動した後にのみ移動できる．具体的には，c(39) は c(34) の移動の後，c(34) は c(37) の移動の後に移動可能になる．

上記から，BG 移動の不可能性を証明するときのスタックの予約とは，あるコンテナが BG 移動 m1＝(x1, s1) によって移動するときの移動先のスタックを特定し，移動 m1を実行する前に実行しなくてはならない別の移動 m2＝(x2, s2) に対して，s1≠s2となるように移動先を限定することである．

5 節の条件（bc1）は以下のように形式的に表現できる．

(bc1') Bay の全てのスタック id の集合を S，スタックの予約 SR を SR⊆S とする．証明過程のコンテナの BG 移動 m＝(s1, s2) について，s2∉SR である．また，m が採用された場合 SR は SR∪{s2} に更新される．証明過程の始め SR は空集合である．

6.2　BG 解の不可能性証明の手続き

本節では，証明を自動化するための手続きを定義する．

(pr1)　Bay 中の最も優先度が低い badly-placed なコンテナ c(P_c) を特定しその個数 N_{BPC} を求める．c(P_c) を持つ各スタックに関して以下の手続きを実行する．

(pr2)　c(P_c) に関して条件（bc2），（bc3），（bc4）のいずれかを満たすスタックを特定する．各条件を満たすスタックの集合をそれぞれ S_{bc2}, S_{bc3}, S_{bc4} とする．

(pr3)　上で特定したスタックの集合 $S_{bc2} \cup S_{bc3} \cup S_{bc4}$ から，BG 移動で潜在的に格納可能なスロットの数 N_{PDS} を求める．N_{PDS} は，Bay の高さ H と（スタック s の）well-placed なコンテナの数 N_{w_s} の差 H- N_{w_s} の合計であり次の式で表せる．$N_{PDS} = \sum_{s \in S_{bc2} \cup S_{bc3} \cup S_{bc4}} (H - N_{w\_s})$．

(pr4)　$N_{BPC} > N_{PDS}$ であれば，BG 解不可能と結論して証明を終了する．

(pr5)　c(P_c) がトップコンテナでない場合，コンテナ c(P_c) 以外で，かつ，コンテナ c(P_c) より上に位置する最も優先度が低いコンテナ c(P'_c) を特定する．c(P'_c) に関して，この証明手続きを再帰的に行い，BG 解不可能という結論の場合これを証明の結論として終了する．

(pr6)　もし $S_{bc2} \cup S_{bc3}$ が空集合で，かつ，S_{bc4} が空集合でないならば，S_{bc4} の各スタックに関して以下の手続きを実行する．

①　コンテナ c(P_c) 以外で，かつ，最も優先度が低い badly-placed なコンテナ c(P''_c) を特定する．

②　c(P''_c) に関して，この証明手続きを再帰

的に行い，BG 解不可能という結論の場合これを証明の結論として終了する．

（pr7） BG 解不可能性は証明できなかったとして，終了する．

（pr1）で決定されたスタックの集合の各要素に対して（pr2）以下の手続きが実行されることに注意されたい．

7　BG 解の不可能性証明による計算効率の改善

本研究では CPMP に対して，解の最適性を保証した効率化で高速な計算の実現を目指す．本研究で開発，または，採用した効率化を整理すると，履歴，下界，マルチスレッド，その他最適性を保証した候補の除外の４つに分類できる．本論文で導入した方法は，下界とその他最適性を保証した候補の除外のどちらにも属するといえる．

これまでの CPMP に関する論文で公開された例題の中でも，著者が知る限り，最大のものが文献（Bortfeldt & Forster, 2012）で示されている．本論文では，この例題のテストケース BF28の20題のうち17番目の例題（本論文では BF28-17と呼んでいる）の解の最適性を証明する方法を導入した．実験に用いた計算機は Intel Xeon E5-2687Wv2を２個搭載し32論理並列実行可能で，128GB のメモリを搭載する．OS は Windows10Pro である．実験プログラムは C＋＋で記述されている．

この方法を導入する前は，文献（小池英勝，2017）で BF28-17と同じテストケースの問題18題について，どれも１秒未満で最適解を出力し，その最適性も証明できていた．その後のプログラムの最適化によって，BF28-17以外の19題について0.4秒未満で最適解を出力し，その最適性も証明できていた．BF28-17についても，解は約0.2秒で出力できていたが，その最適性の証明は１日かけても計算が終了しないため中断していた．本法を導入した後は，約1.9秒で解の発見とその最適性の証明を終了するようになった．

8　まとめ

本研究は，CPMP に対して，最適解を求めようとするものである．効率化の手法は，全て解の最適性を保存するものだけを採用する．上述の実験結果が示すように，このアプローチで解の最適性，処理効率両面で良好な結果が得られることが示された．

本論文で示した証明手続き開発の最初の動機は，文献（Bortfeldt & Forster, 2012）で示された CPMP の例題の一つである BF28-17の73個の移動で構成される解の最適性を示すことであった．既存の解の下界の計算方法と初期状態の badly-placed なコンテナの個数から，この問題を長さ72の BG 解が存在しないことを証明する問題としてとらえた．そして，BG 解が存在しないことを証明するための証明手続きを一般化して定義し，他の問題にも適用可能にした．

本研究で提案した方法は，BF28-17では大きな効果が確認できたが，全ての問題で効率が改善するとは限らない．この方法は，比較的コストの高い処理であるため，この方法を効率的に適用するための条件について今後研究する必要がある．

謝辞　本研究は，2016年度札幌学院大学研究奨励金（C）SGU-AS16-08の助成を受けた．

参考文献

[1] Bortfeldt, A. and Forster, F. (2012). A tree search procedure for the container pre-marshalling problem. European Journal of Operational Research, 217(3), 531-540.

[2] Kim, K. H. and Hong, G. P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research, 33, 940-954.

[3] 小池英勝（2017）. コンテナプリマーシャリング問題における探索空間の削減, 札幌学院大学 総合研究所紀要, 4 , 9-15.

[4] Koike, H. (2017). Search Space Reduction with Multiset for Effectively Solving the Container Pre-Marshalling Problem, 2017 International Conference on Mathematical Methods & Computational Techniques in Science & Engineering (AIP Conference Proceedings), 1872 (1), 020026 .

[5] Zhang, R., Jiang, Z. and Yun, W. Y. (2015). Stack pre-marshalling problem: A heuristic-guided branch-and-bound algorithm, International Journal of Industrial Engineering, 22(5), 509-523.

The Proceedings of the Research Institute of Sapporo Gakuin University Vol.5, 7–15 (2018)

Efficient Computation of the Container Pre-Marshalling Problem with a Proof Method of the Impossibility of Solutions

Hidekatsu KOIKE[1]

Abstract: This paper proposes an efficient method for optimal solutions of the container pre-marshalling problem (CPMP). The method determines the possibility of a certain class of solutions to a certain class of CPMPs. This paper shows that the class of CPMPs can be efficiently solved by using the proposed method.

Keywords: Combinational Optimization, Container Pre-Marshalling Problem, Optimization, Impossibility Proof Method, Logistics.

[1] Department of Economics, Sapporo Gakuin University; koike@sgu.ac.jp.