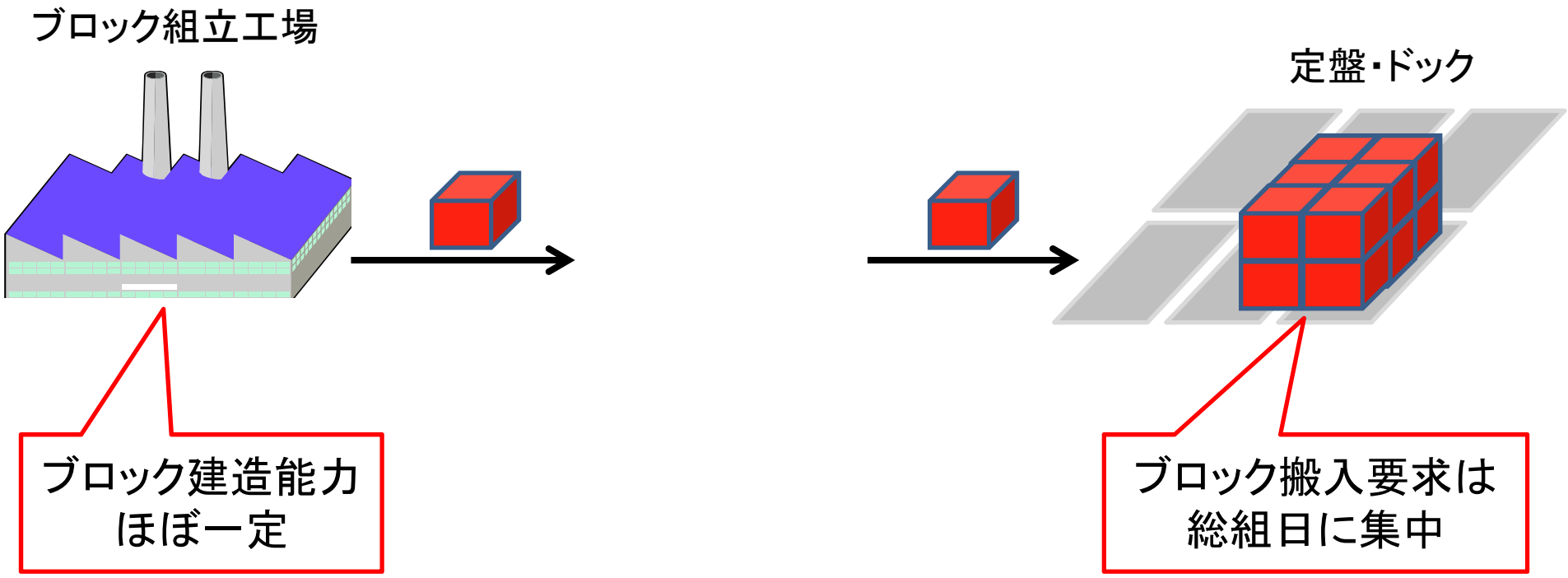
造船所内ブロックストックヤードの
スケジューリングにおけるブロック配置の最適化

九州大学　木村　元

日本船舶海洋工学会西部支部秋季講演会
2011年11月1日～2日＠下関

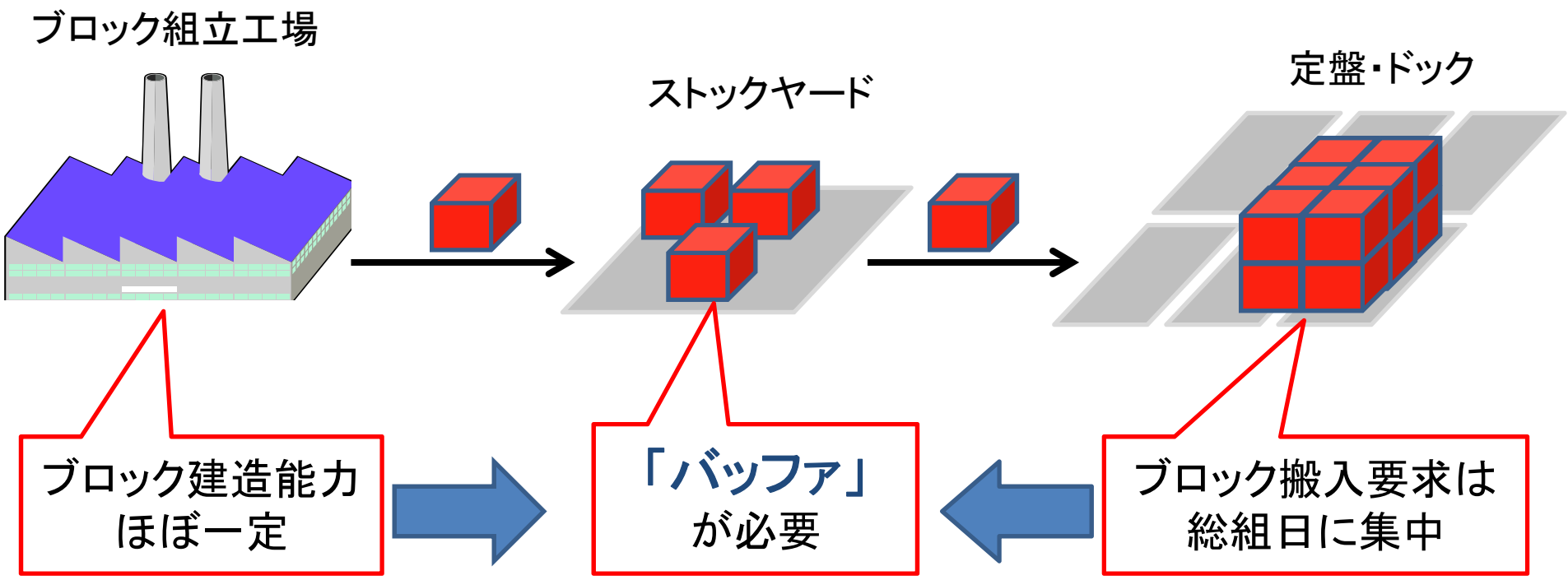
背景(1)

- ストックヤードの必要性



背景(1)

- ストックヤードの必要性



背景(2)

- 造船所のブロックストックヤードの特徴

ストックヤードに蔵置される船舶建造ブロックは...

1） 巨大・重量物である

2） (自動車部品のように)ラックにストックできない

3） (コンテナのように)積み重ねたりできない

広大な敷地が必要

背景(2)

- 造船所のブロックストックヤードの特徴

ストックヤードに蔵置される船舶建造ブロックは...

1） 巨大・重量物である

2） (自動車部品のように)ラックにストックできない

3） (コンテナのように)積み重ねたりできない

広大な敷地が必要

背景(2)

- 造船所のブロックストックヤードの特徴

ストックヤードに蔵置される船舶建造ブロックは...

1） 巨大・重量物である

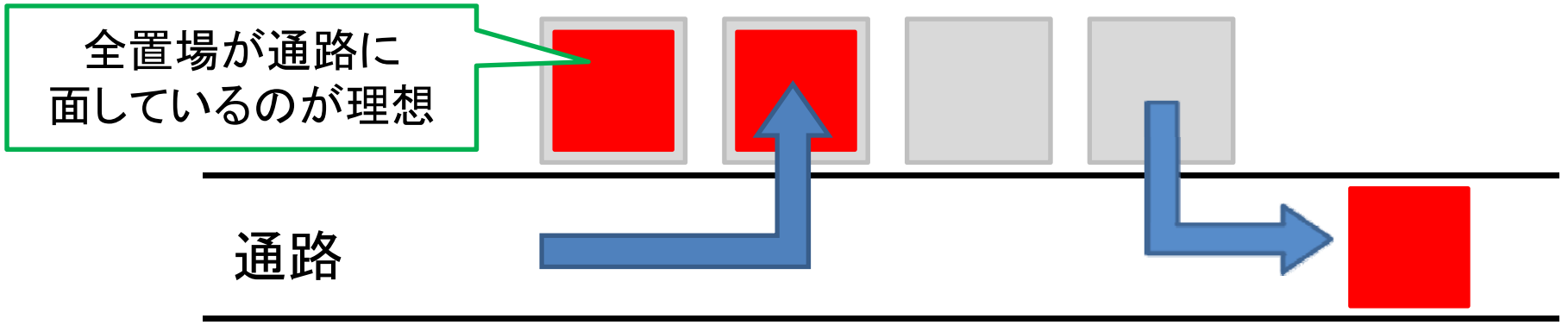
2） (自動車部品のように)ラックにストックできない

3） (コンテナのように)積み重ねたりできない

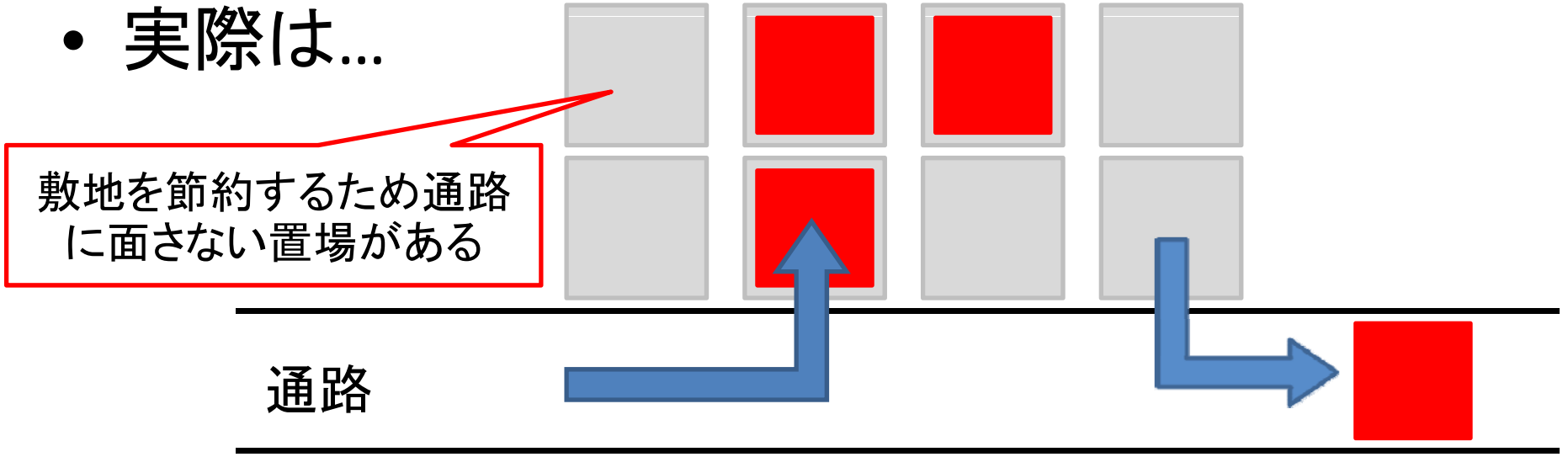
広大な敷地が必要

背景(3)

- 理想的なブロックストックヤード

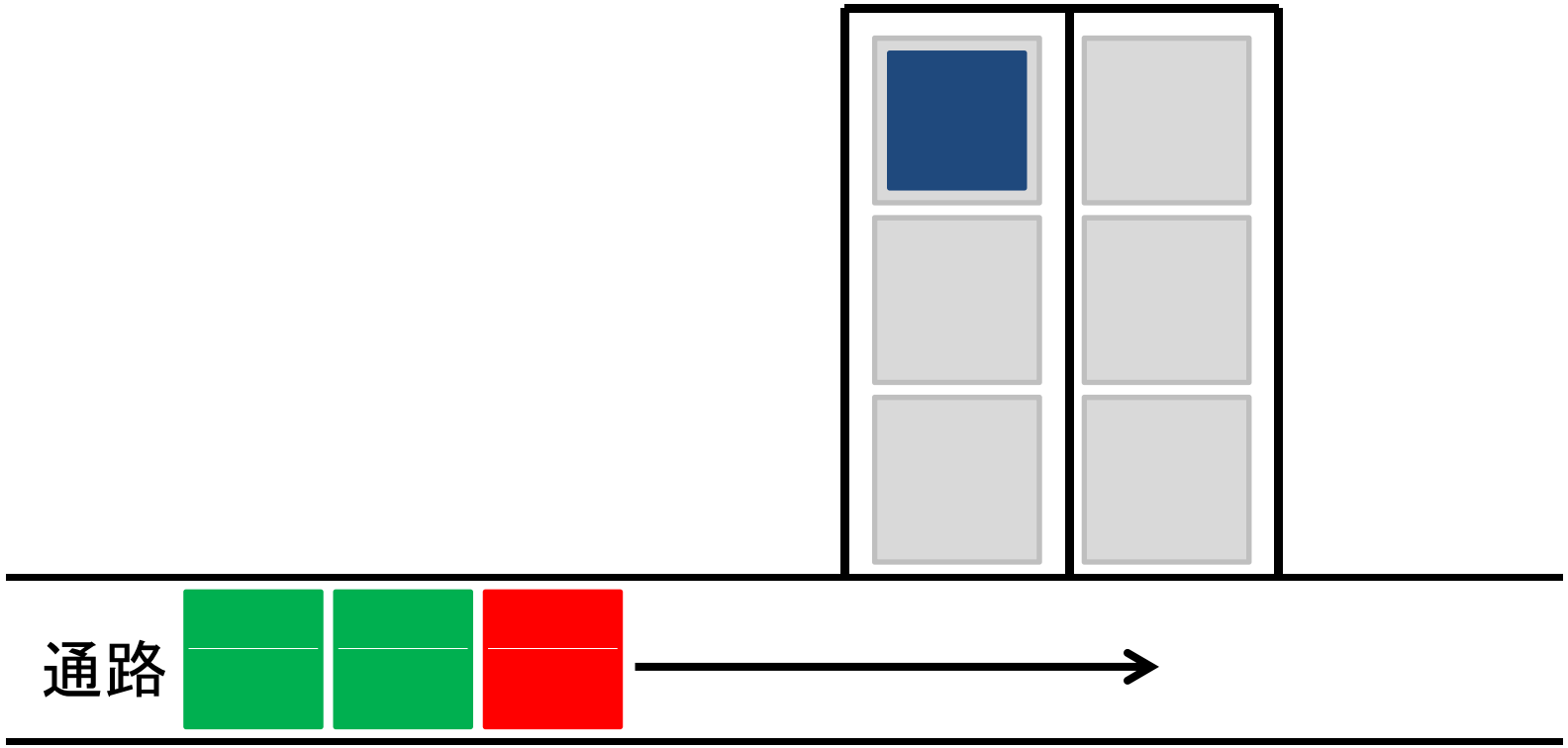


- 実際は…



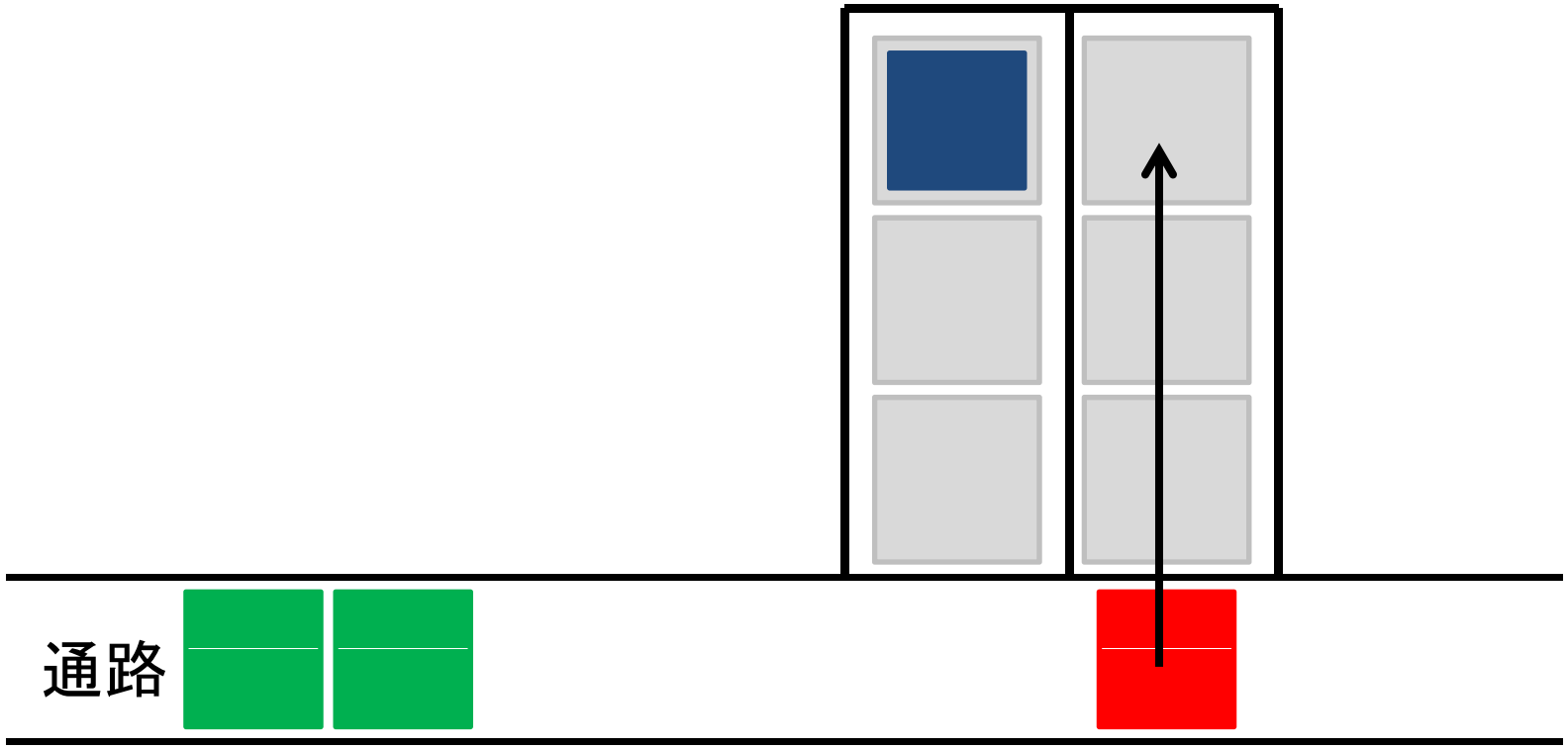
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



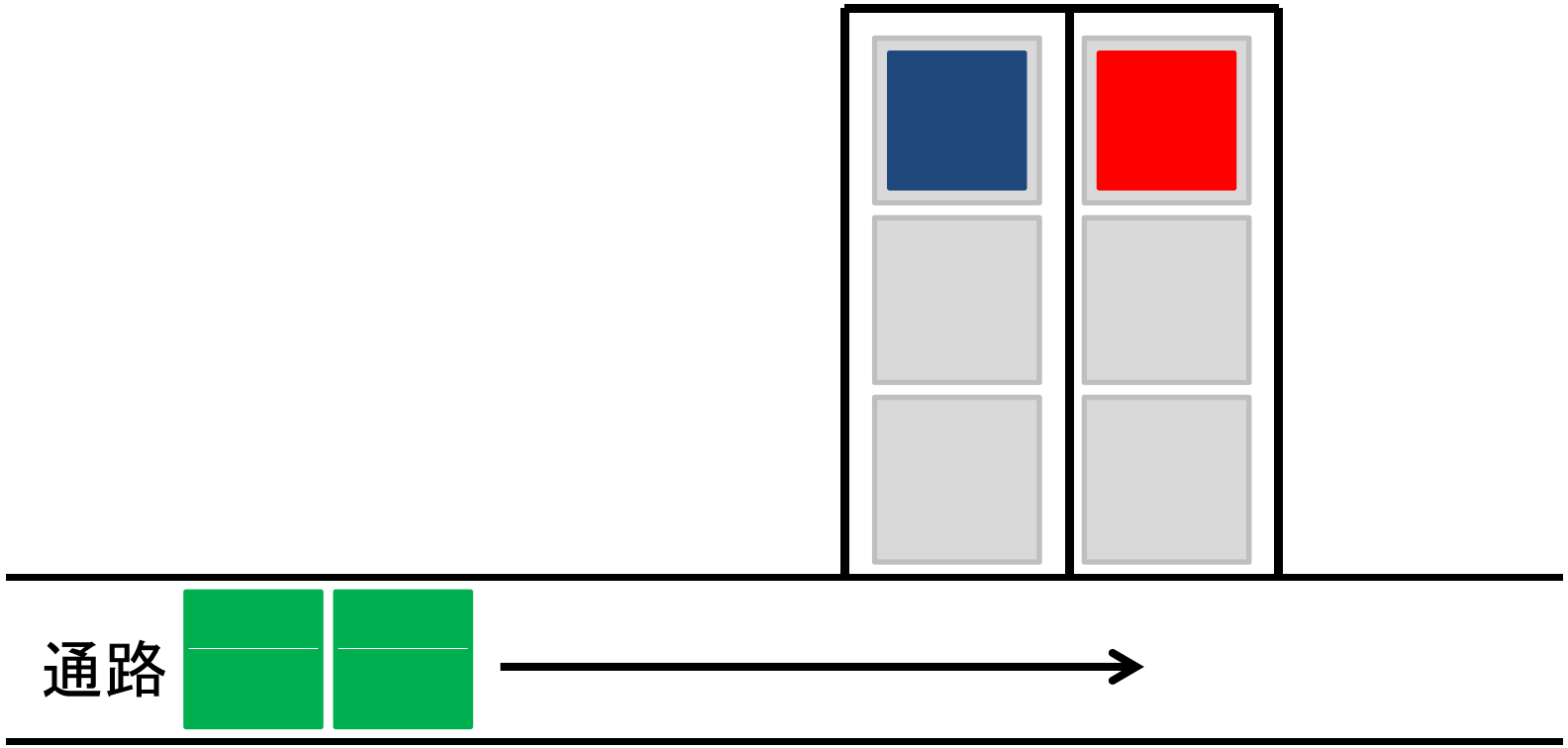
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



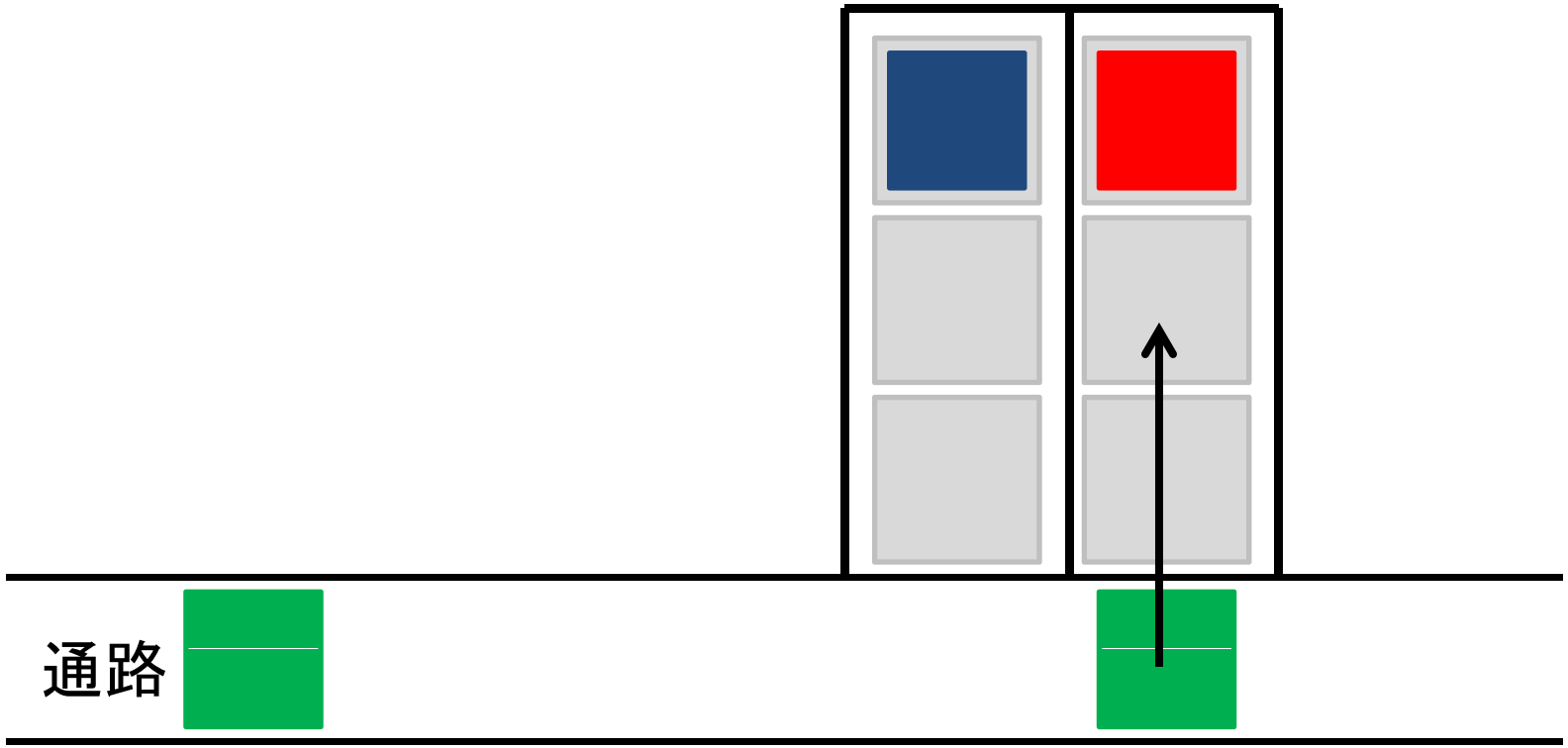
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



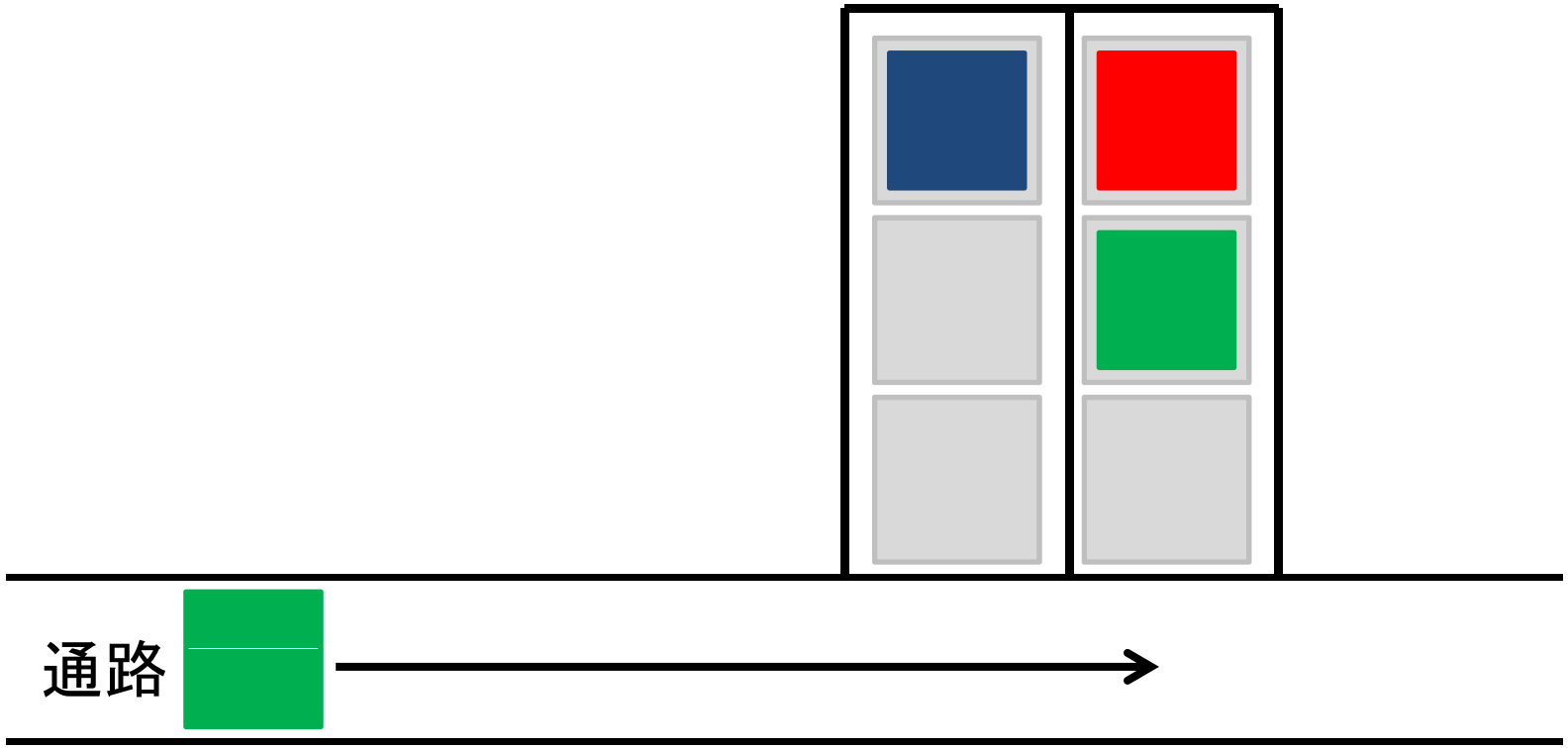
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



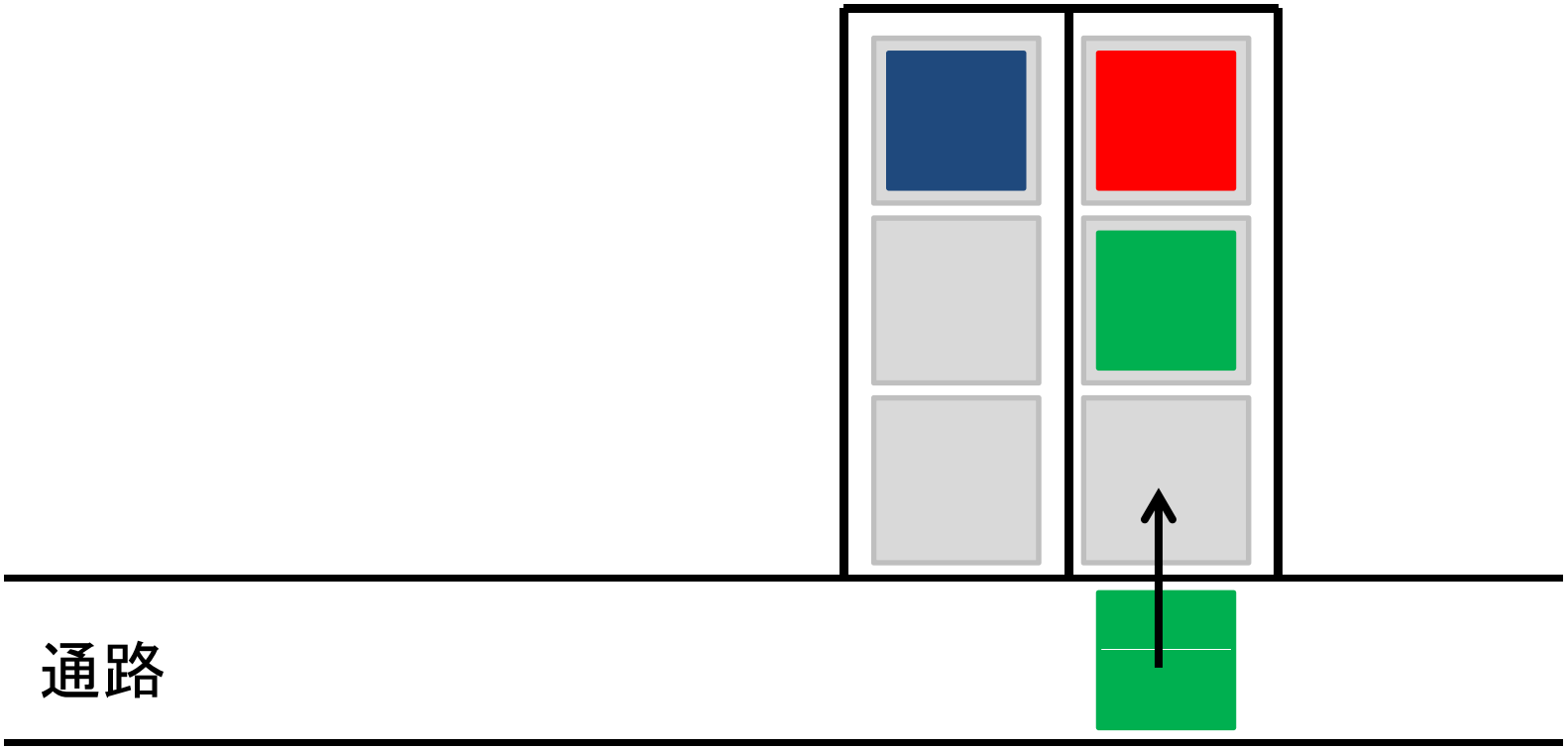
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



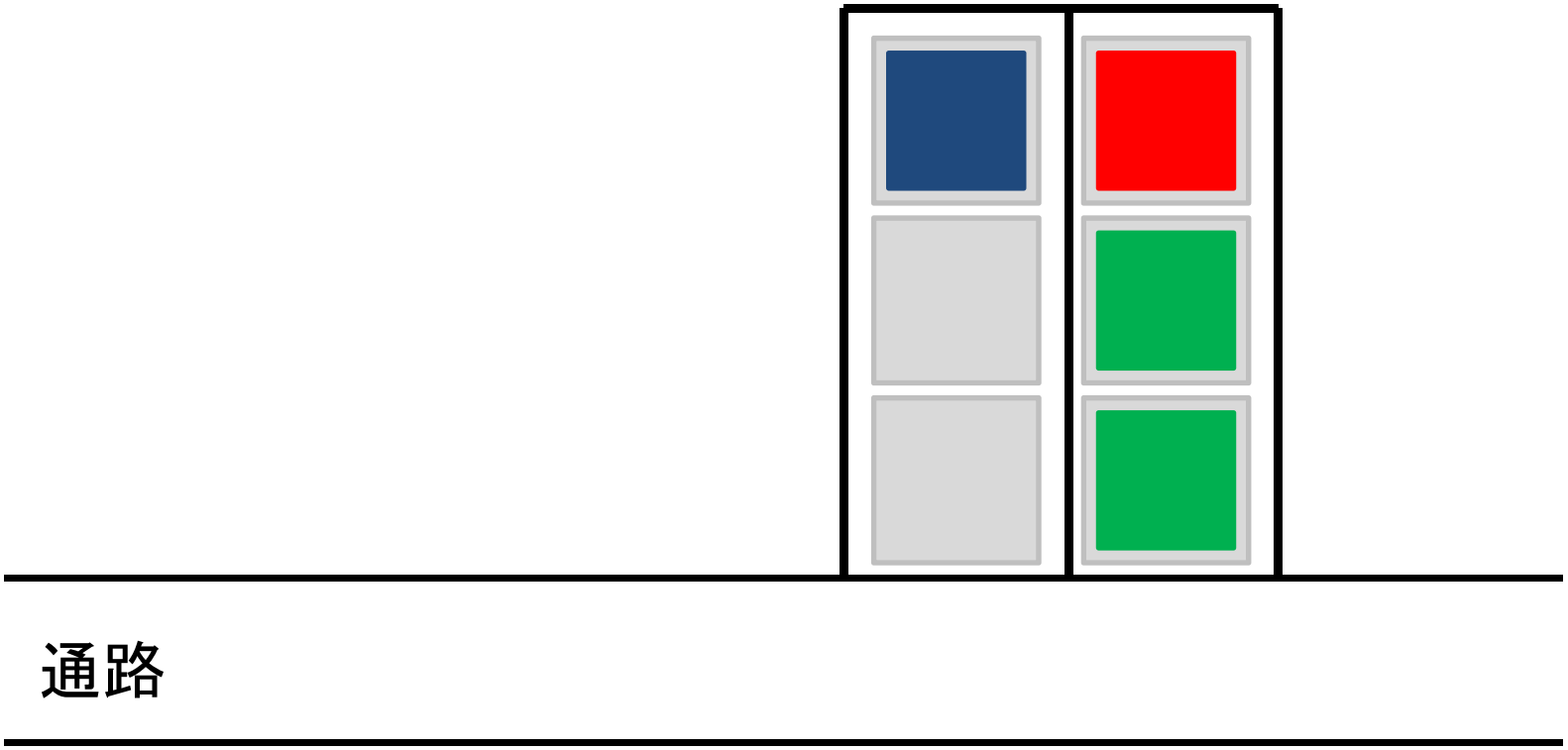
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



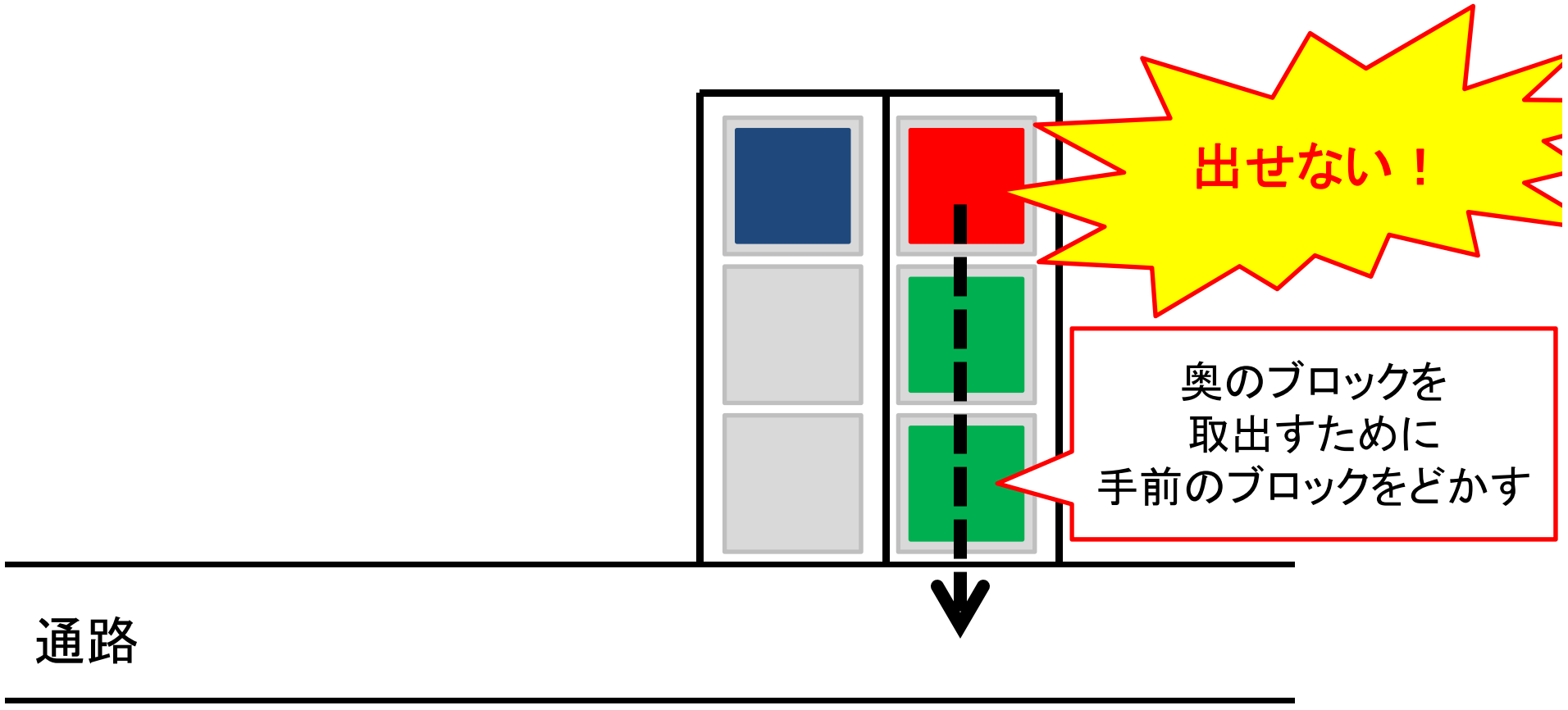
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



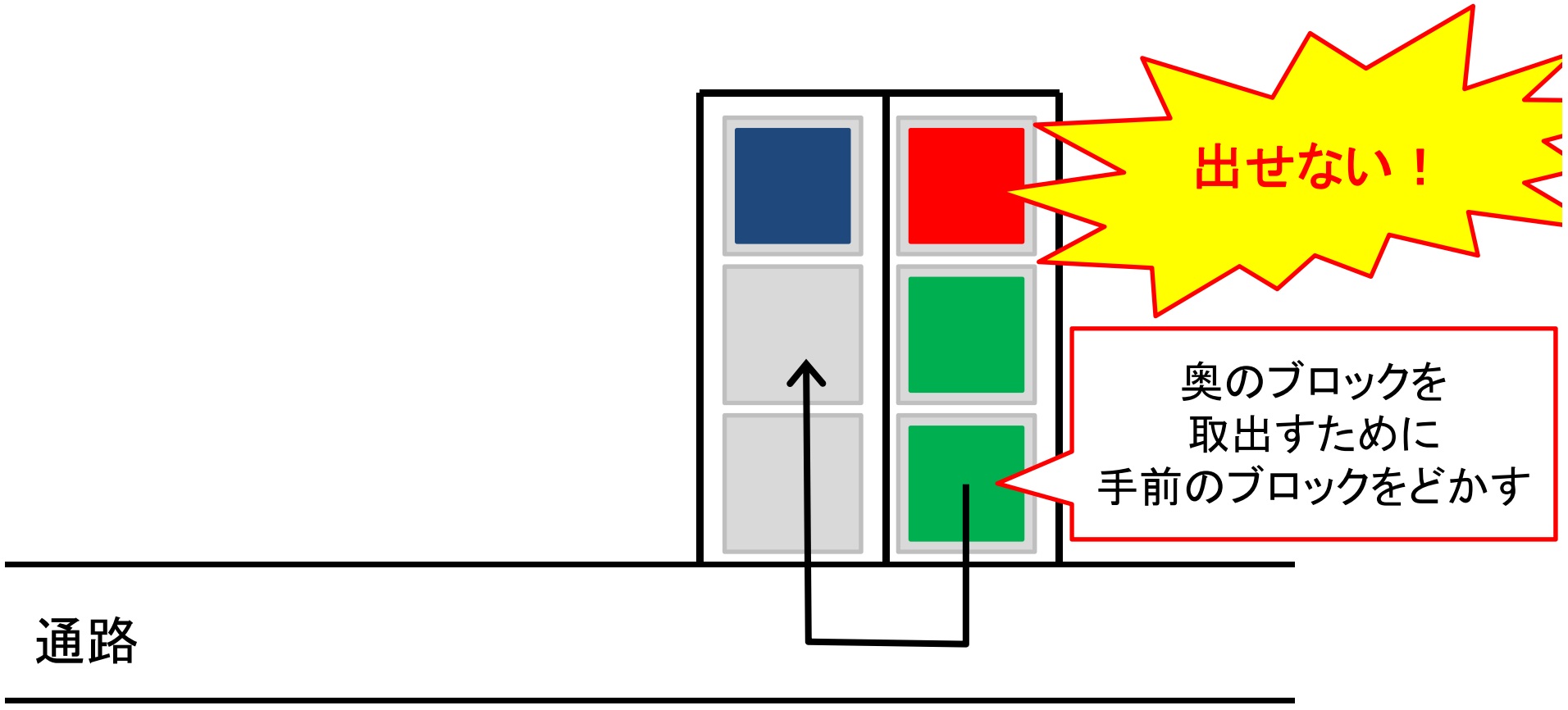
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



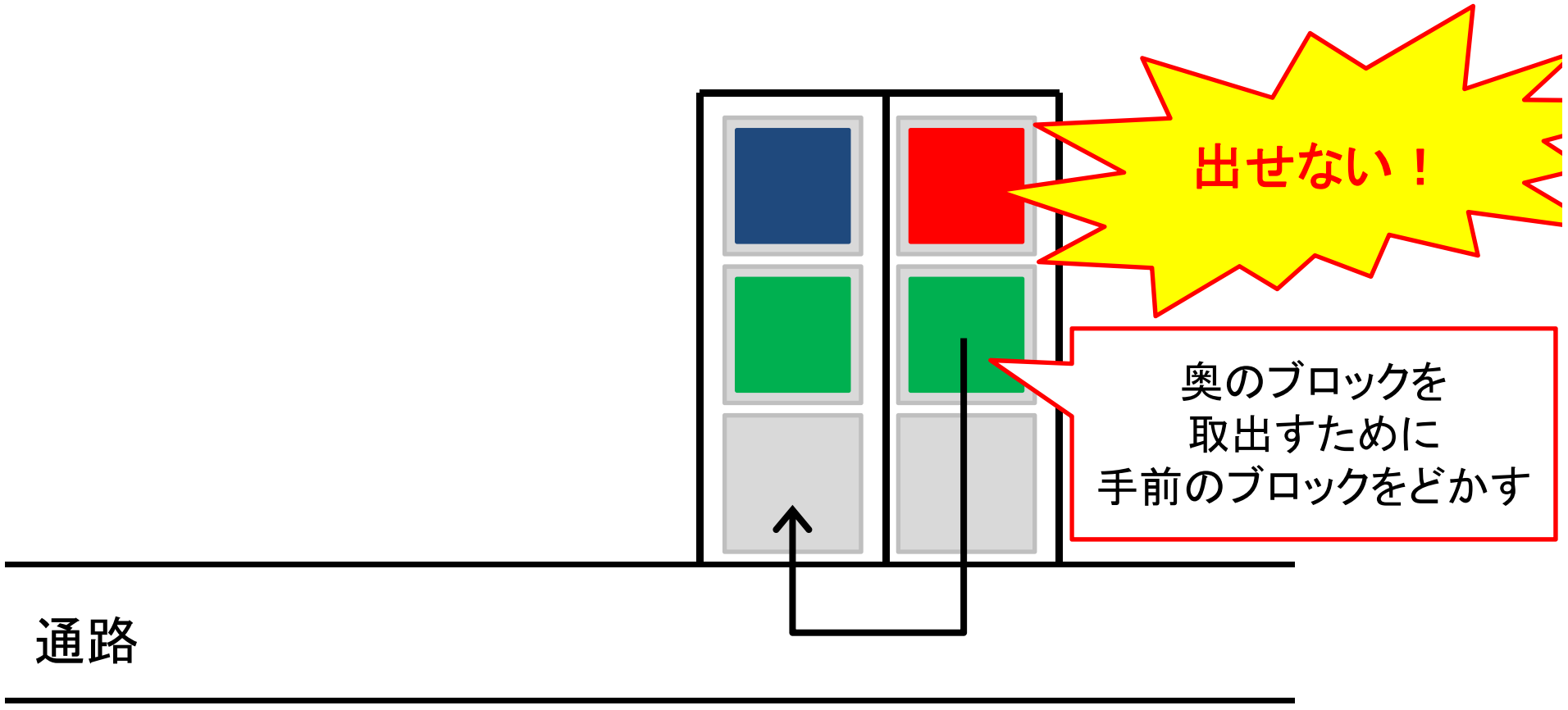
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



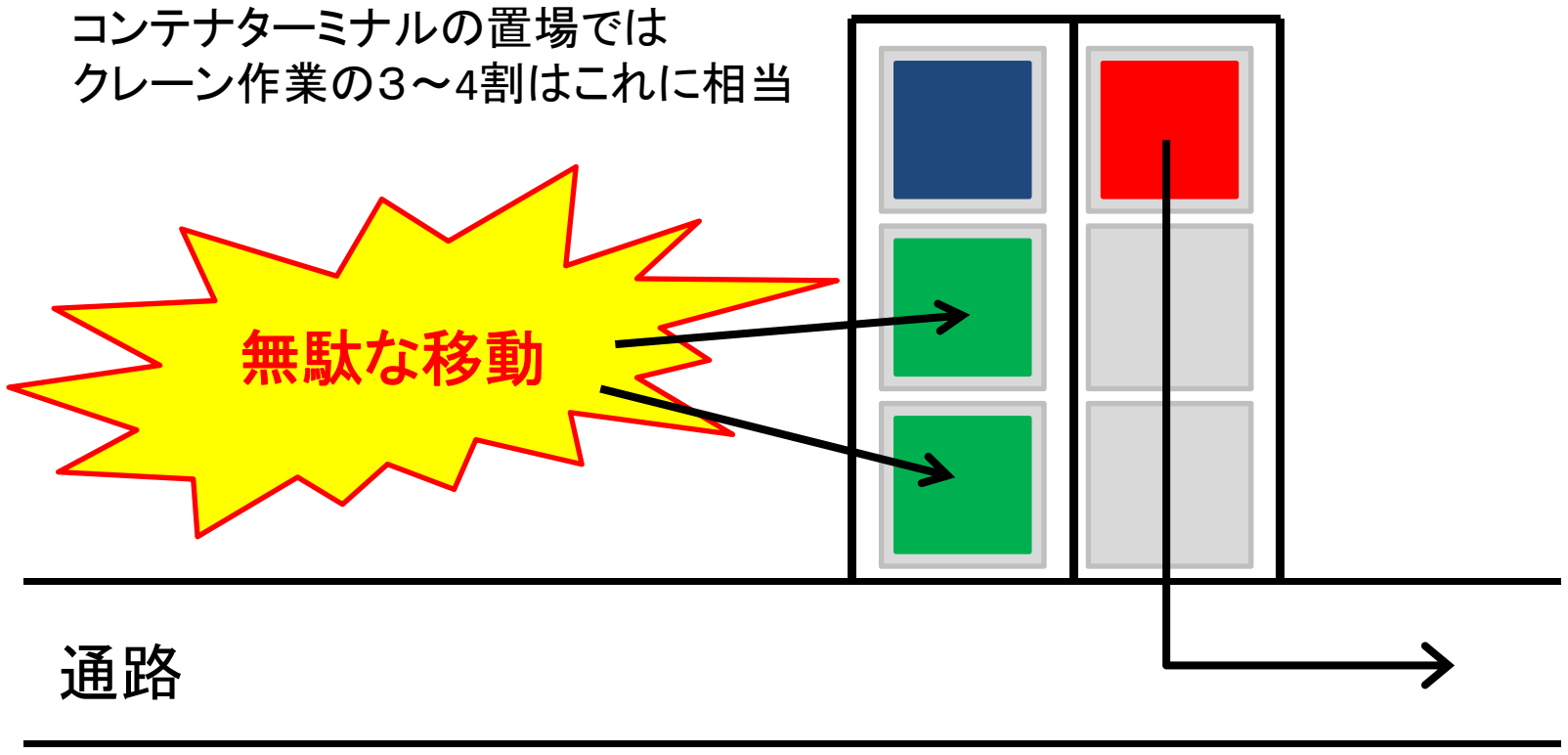
背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



背景(4)

- 先入れ後出し(First In Last Out: FILO)の**スタック構造**



各ブロックはストックヤードへの搬入日と搬出日が決まっている

置場の指示を適切に行えば無駄な移動を減らせる

研究の目的

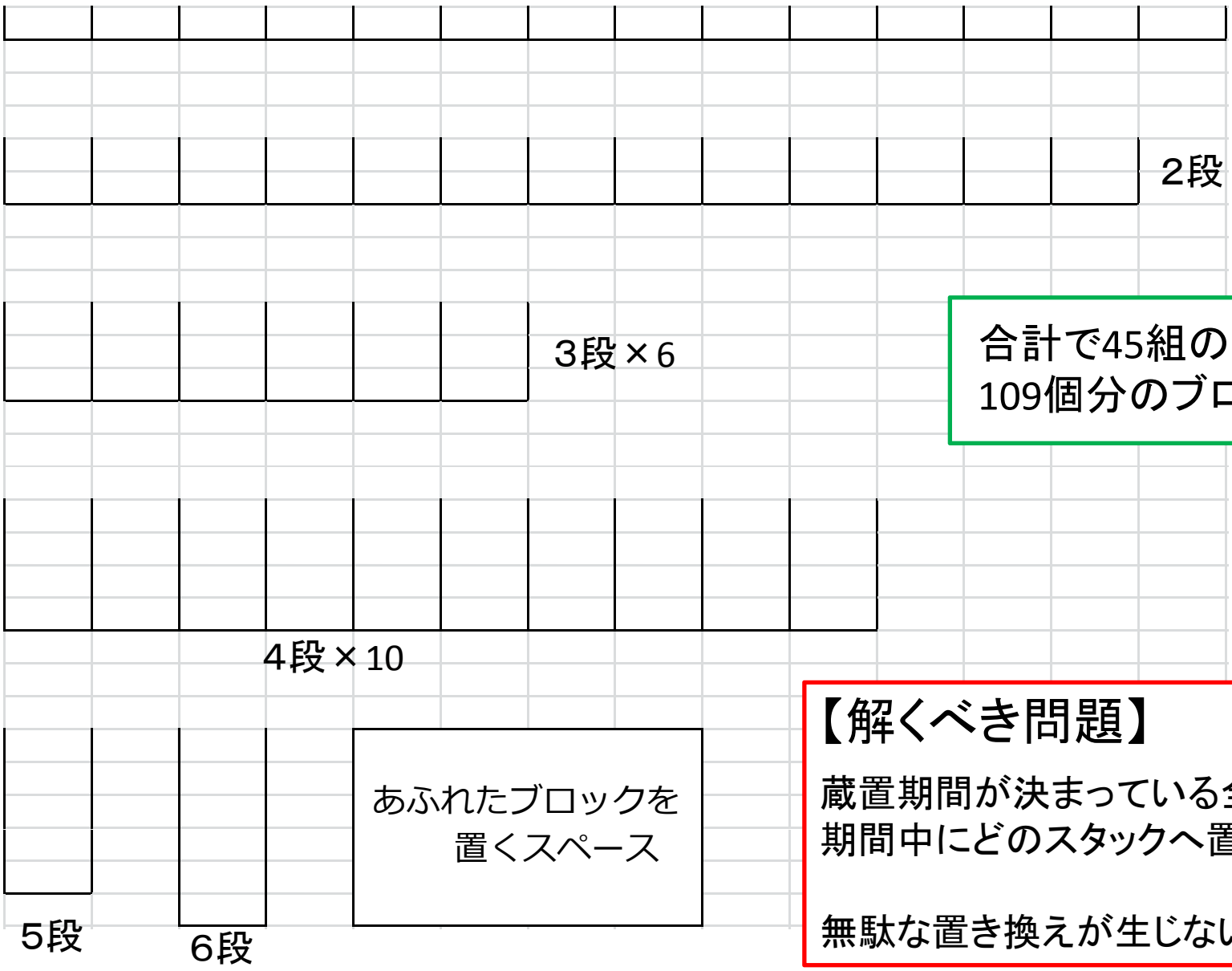
・スタック構造を有するブロックストックヤードにおいて、
　ブロックが無駄な動きをしない適切な置場の指示方法
・自動化システムの構築

【アプローチ方法】

（1）特殊な**組み合わせ最適化問題**へ帰着

（2）組み合わせ最適化問題の解法：　**分枝限定法**

問題のモデル化



合計で45組のスタック
109個分のブロック置場

【解くべき問題】

蔵置期間が決まっている全ブロックを
期間中にどのスタックへ置くか？

無駄な置き換えが生じない置き方で

無駄な置き換えの生じない蔵置法：理想スタック蔵置ルール

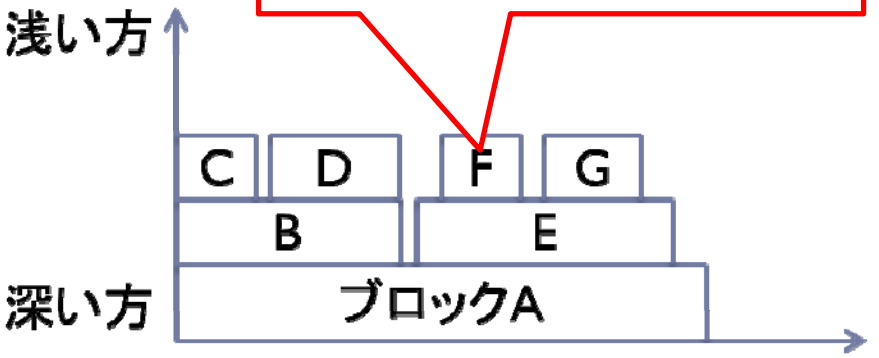
ある時刻tにおいてスタックの深さdに置かれているブロックの蔵置を考える
（スタックの深さdは、1が最も深く，数字が増加するほど浅い置場とする）
蔵置開始時刻 bs(t,d)
蔵置終了時刻 be(t,d)
時刻tと深さdで指定された深さと時刻に何も置かれていない場合は bs(t,d)=be(t,d) = t

ブロックの出し入れがFILOとなる条件：

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

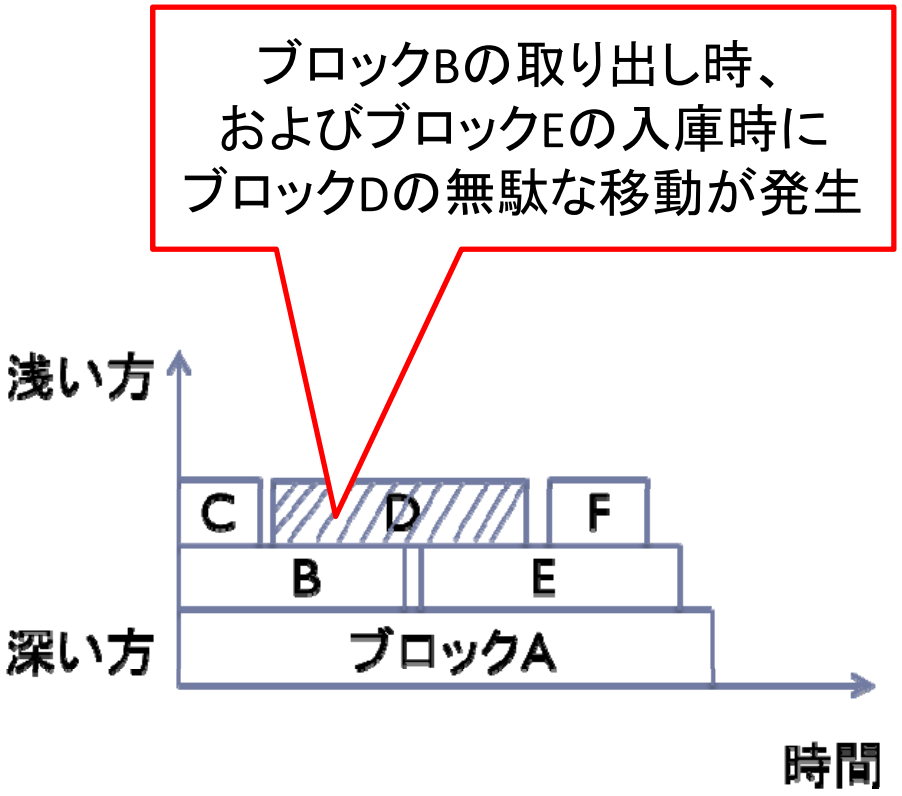
理想スタック蔵置ルール

同一スタックにおいて、
浅い場所に置かれるブロックの
蔵置期間は、奥に置かれた物
の蔵置期間の範囲内にある

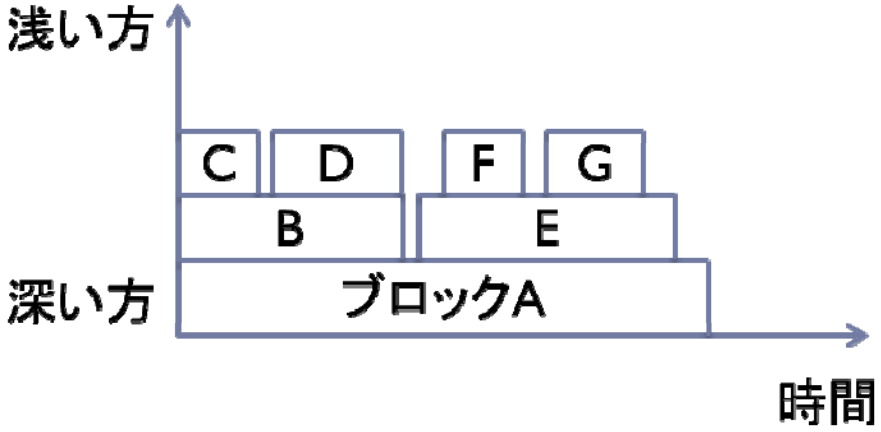


【理想スタック蔵置ルールに従った置き方】

無駄な置き換えの生じない蔵置法：理想スタック蔵置ルール



【理想スタック蔵置ルールに従わない置き方】



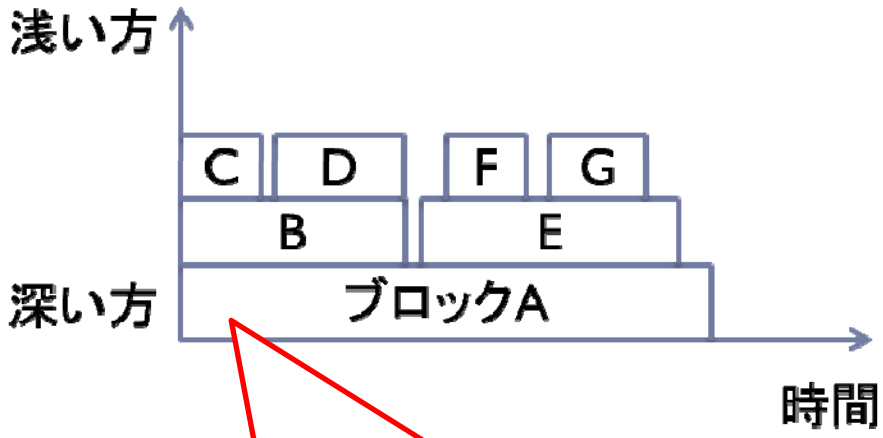
【理想スタック蔵置ルールに従った置き方】

ブロック割り当てアルゴリズムの提案

【1】 蔵置開始日と期間によるブロックデータのソート

ブロックNo.	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
1000	■	■	■	■													
1001	■	■	■	■													
1002	■	■	■														
1003	■	■	■														
1004	■	■	■														
1005	■	■	■														
1006	■	■	■														
1007		■	■	■	■	■	■	■	■	■							
1008		■	■	■	■												
1009		■	■	■	■												
1010		■	■	■													
1011		■	■	■													
1012		■	■	■													
1013		■	■	■													
1014		■	■														
1015		■	■														
1016			■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
1017			■	■	■	■	■	■	■	■	■	■	■	■	■	■	
1018			■	■	■	■											
1019			■	■	■	■											
1020			■	■	■												
1021			■	■	■												
1022			■	■	■												
1023			■	■	■												
1024			■	■	■												
1025			■	■													
1026				■	■	■	■	■	■	■	■	■	■				
1027				■	■	■	■										
1028				■	■	■	■										
1029				■	■	■	■										
1030				■	■	■											
1031				■	■	■											
1032				■	■	■											
1033				■	■	■											
1034					■	■	■	■	■	■	■	■	■				
1035					■	■	■	■	■	■	■	■					
1036					■	■	■	■	■	■	■	■					
1037					■	■	■										
1038					■	■	■										
1039					■	■	■										
1040					■	■	■										
1041					■	■	■										
1042					■	■											
1043					■	■											

【理想スタック蔵置ルールに従った置き方】



蔵置開始日が同じブロックは、
蔵置期間が長いものが先に
スタックの奥へ置かれるべき

ブロック割り当てアルゴリズムの提案

【2】 ソートされたブロックの割り当てアルゴリズム1

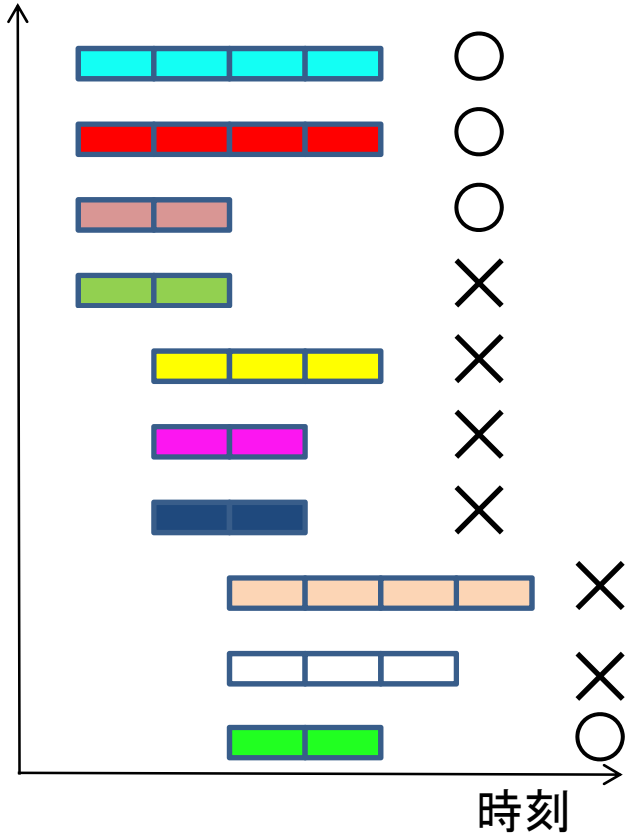
数段の多いスタックほど扱いが厄介

段数の多いスタックから隙間なくブロックを詰め込む
（最適解を得る保障が無いヒューリスティクス）

1） 1つのスタックに注目し、置場の決まっていないブロックを選択してソートした順に割り当てていく

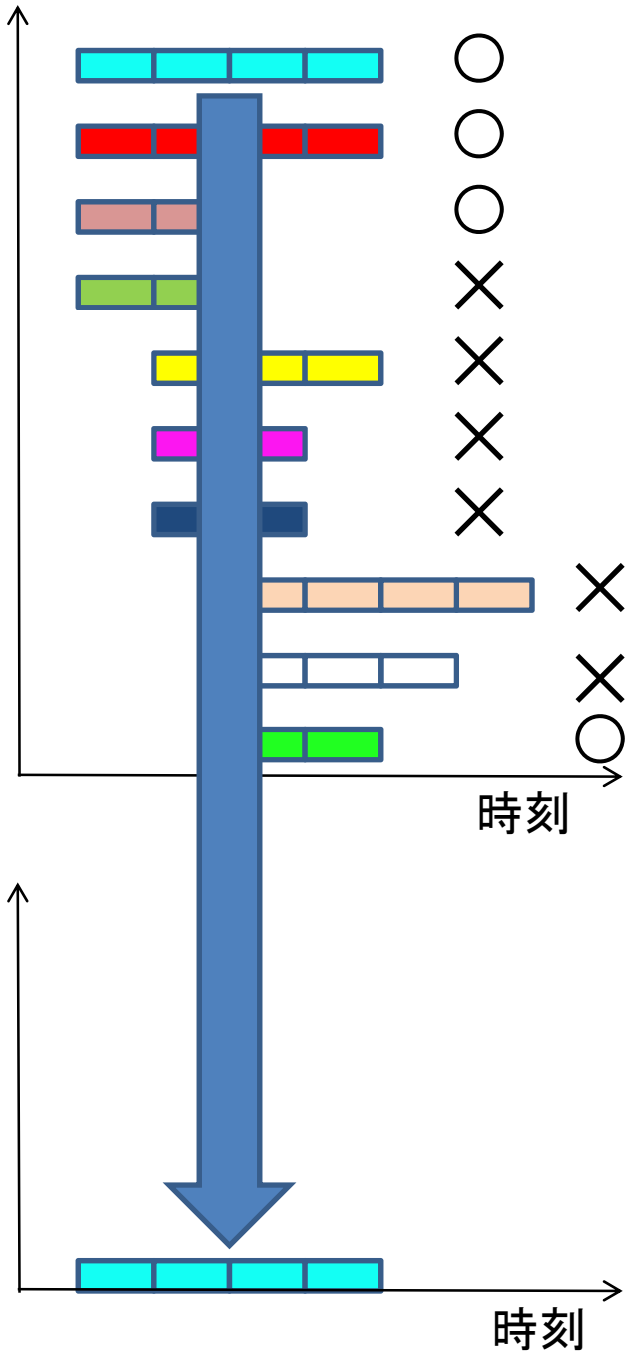
2） 理想スタック蔵置ルールを順守

3） スタックの利用効率が高くなるようブロック選択



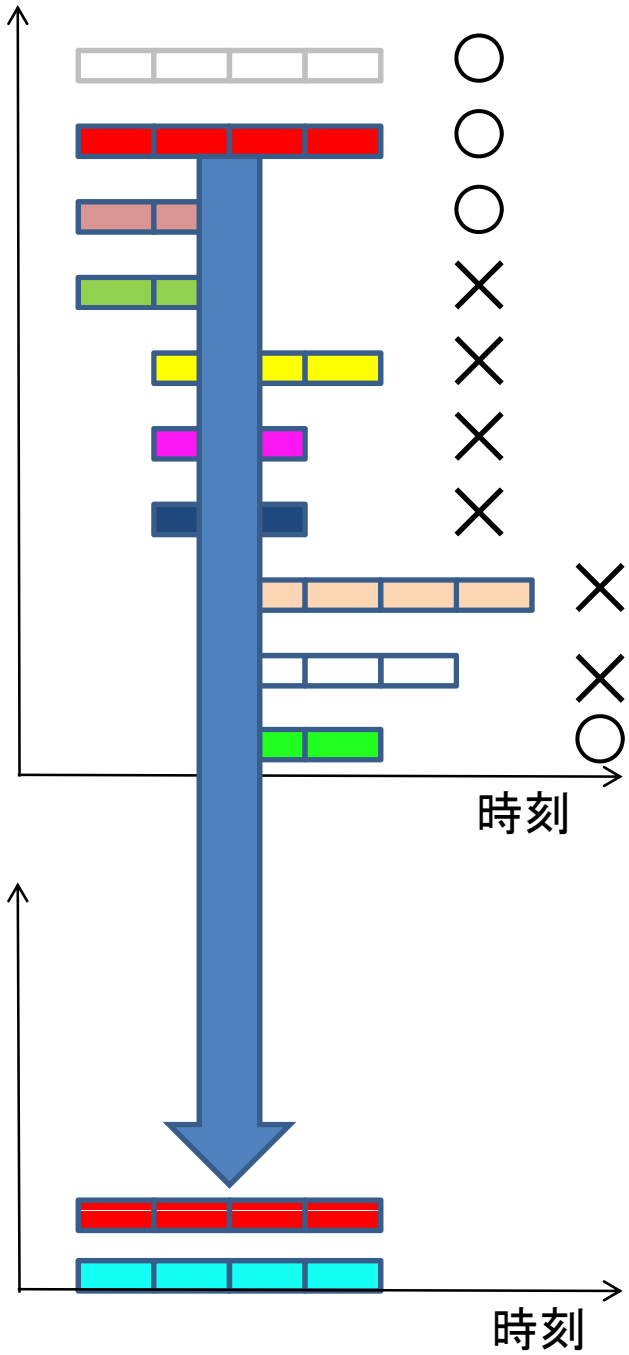
蔵置開始日と期間でソートされたブロック

3段のスタックへの割り当て例



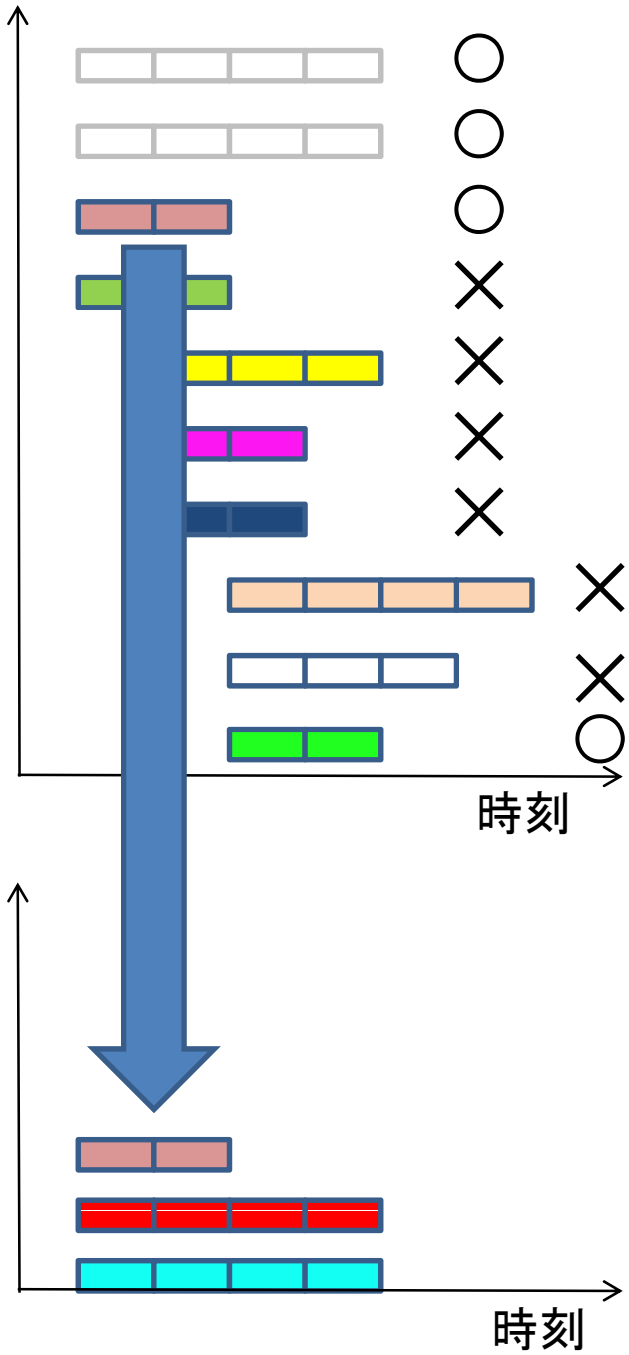
蔵置開始日と期間でソートされたブロック

3段のスタックへの割り当て例



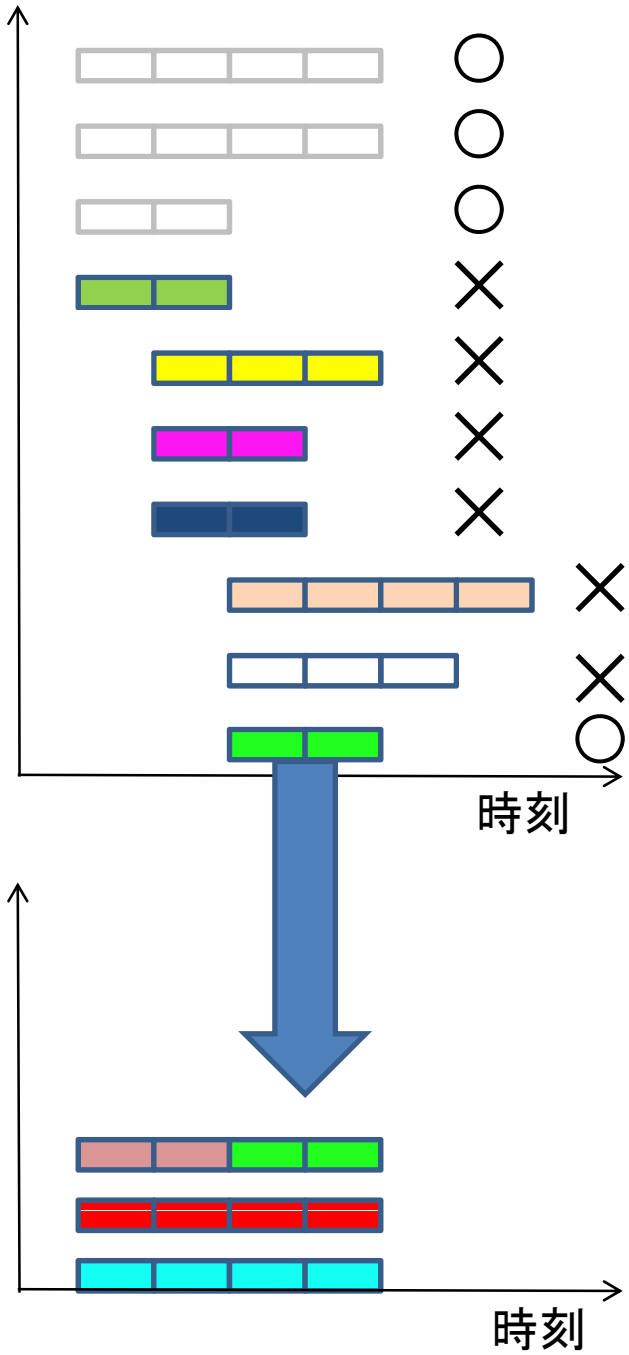
蔵置開始日と期間でソートされたブロック

3段のスタックへの割り当て例



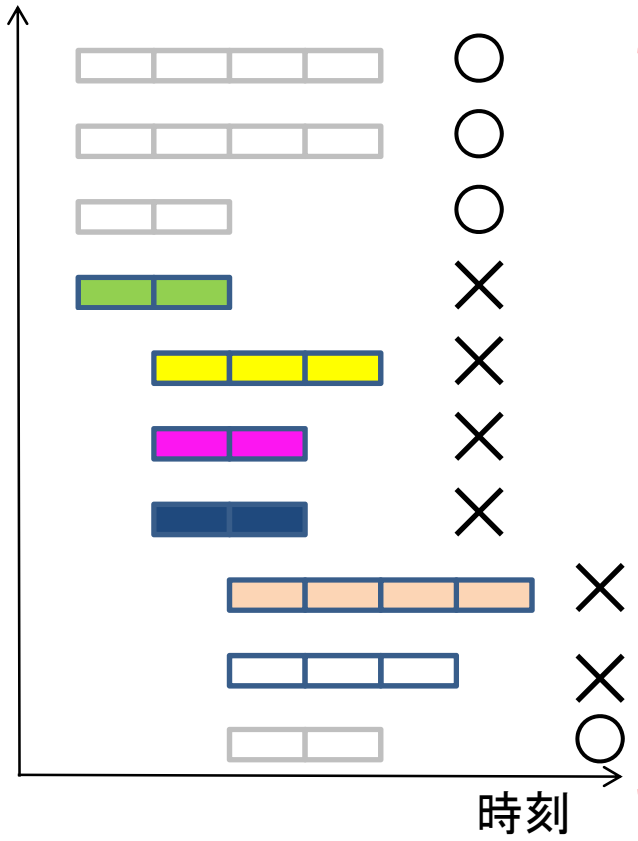
蔵置開始日と期間でソートされたブロック

3段のスタックへの割り当て例



蔵置開始日と期間でソートされたブロック

3段のスタックへの割り当て例



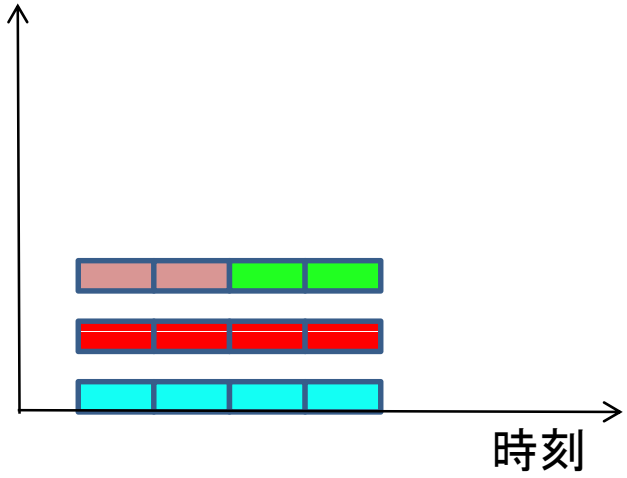
蔵置開始日と期間でソートされたブロック

問題の定式化

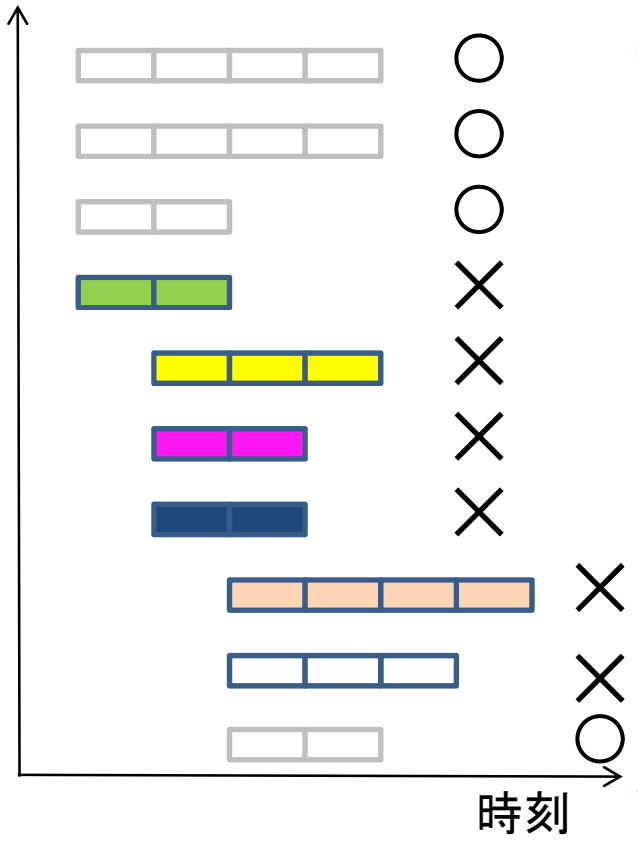
注目するスタックに
未蔵置ブロックを割り当てるかどうか

2の(ブロックの個数)乗の
組み合わせ最適化

分枝限定法で解く



3段のスタックへの割り当て例



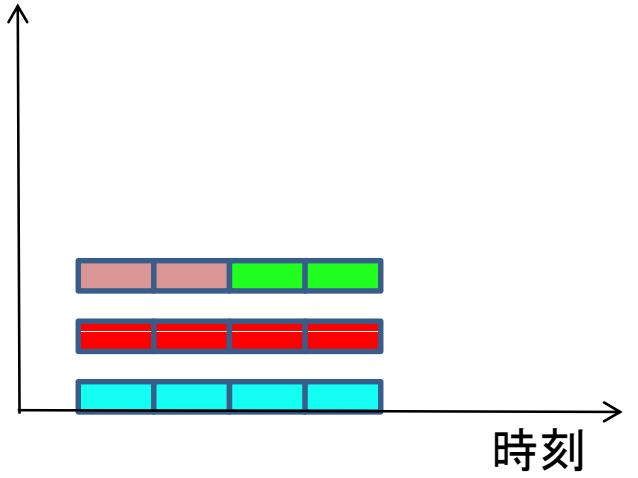
蔵置開始日と期間でソートされたブロック

問題の定式化

注目するスタックに
未蔵置ブロックを割り当てるかどうか

2の(ブロックの個数)乗の
組み合わせ最適化 2^{3061}

分枝限定法で解く

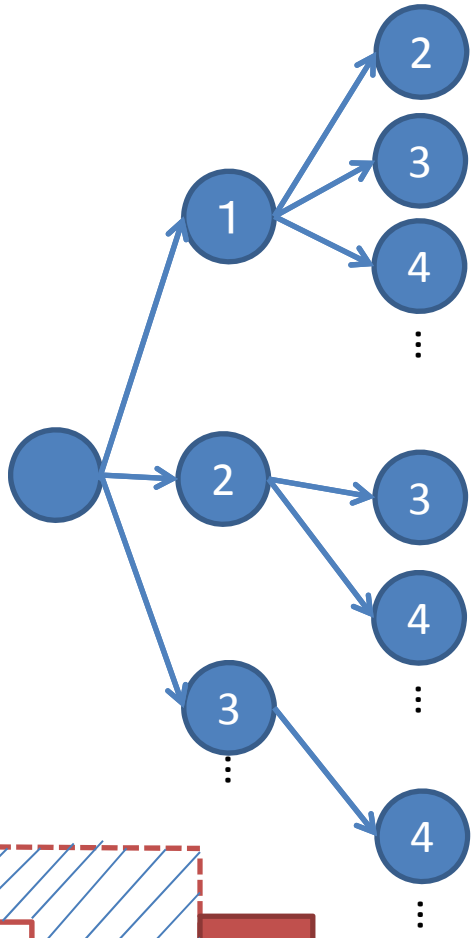


3段のスタックへの割り当て例

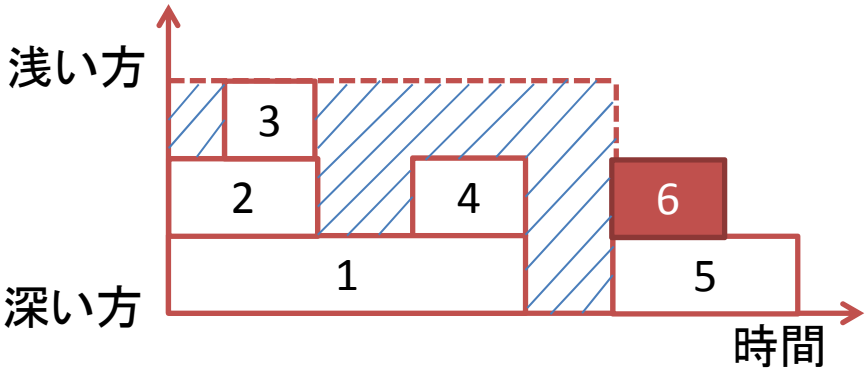
深いスタックから割り当てる分枝限定法1

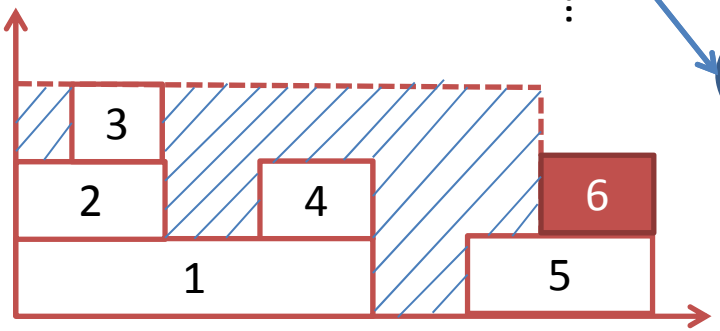
- スタックに置くブロックのパターンは、右図のように決定木で表すことができる。
- コストを定義し、コストの低い順に決められた数の枝を残し、あとは刈り取る。

ソートされたブロック



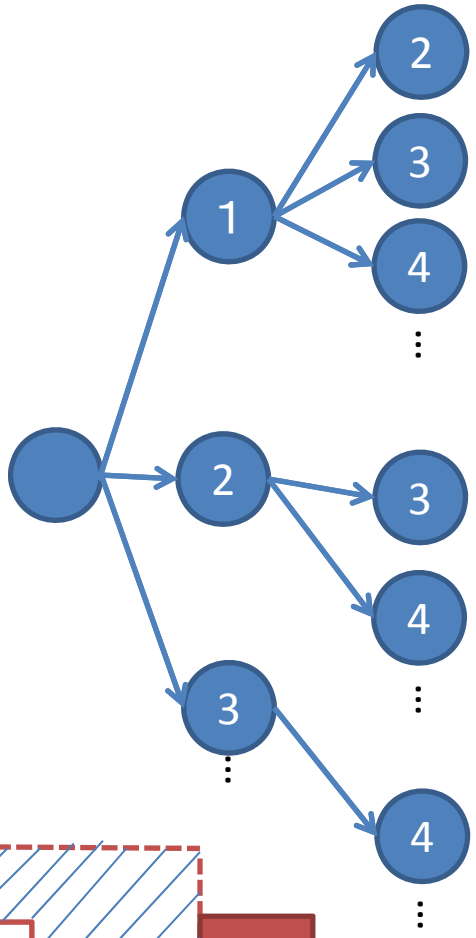
コストの計算（3段のスタックの例）





深いスタックから割り当てる分枝限定法1

- スタックに置くブロックのパターンは、右図のように決定木で表すことができる。
- コストを定義し、コストの低い順に決められた数の枝を残し、あとは刈り取る。



ブロック割り当てアルゴリズムの提案

【2‘】 ソートされたブロックの割り当てアルゴリズム2

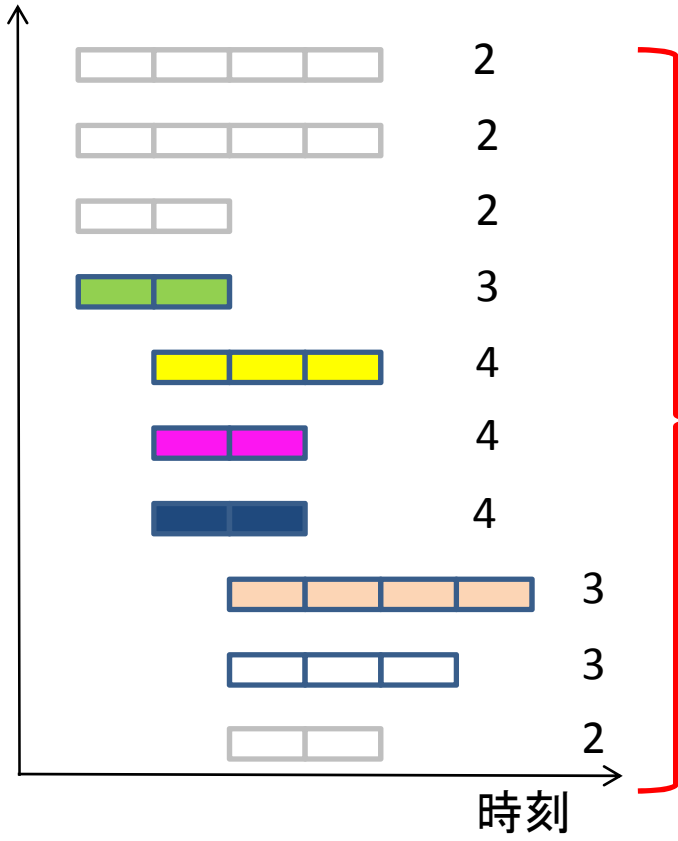
全スタックを考慮に入れ、計算機の力まかせにブロックを詰め込む
（分枝限定法2）

1） ソートされたブロック順に、置けるスタックがある限り
ブロックをいずれかのスタックへ割り当てていく

2） 理想スタック蔵置ルールを順守

3） スタックの利用効率が高くなるようブロック選択

蔵置開始日と期間でソートされたブロック



割当アルゴリズム２‘における問題の定式化

ソートされた未蔵置ブロックを
どのスタックへ割り当てるか

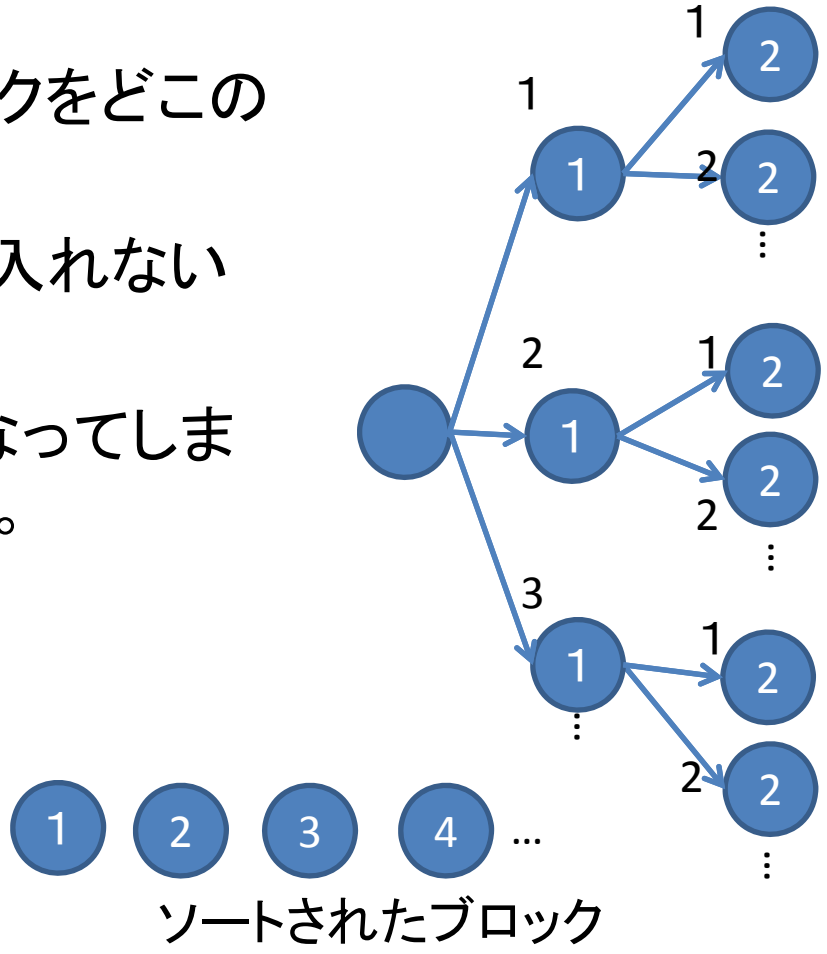
（スタック個数）の（ブロックの個数）乗の
組み合わせ最適化

45^{3061}

分枝限定法で解く

全てのスタックを探索する分枝限定法2

- 分枝限定法1と違い、ブロックをどこのスタックに入れるかを探索。
- ブロックがあったらどこかに入れないといけないというスタンス。
- スタックがどこにも置けなくなってしまえば、そのブロックはとばす。
- コスト計算は1と同じ。



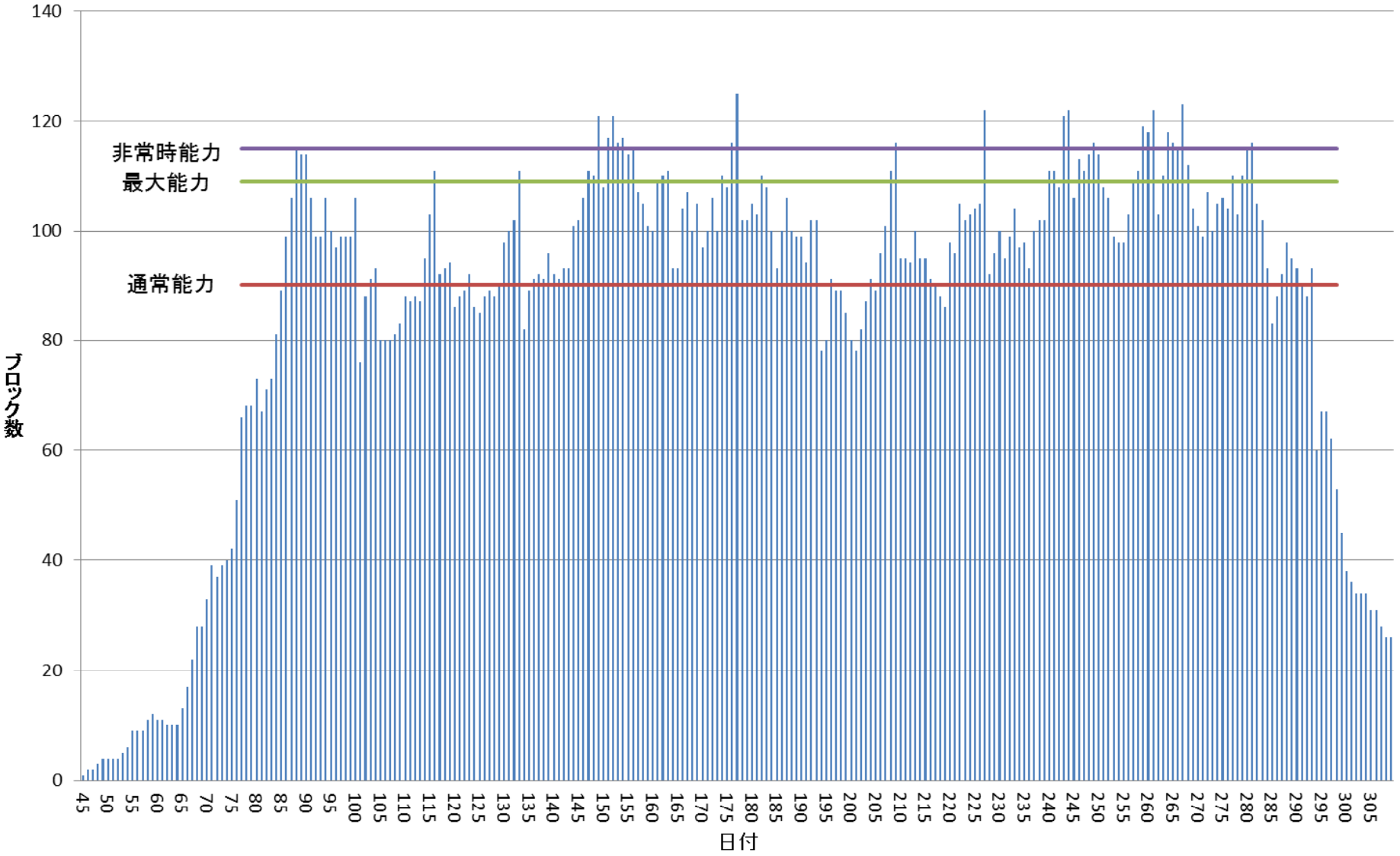
計算機シミュレーション実験

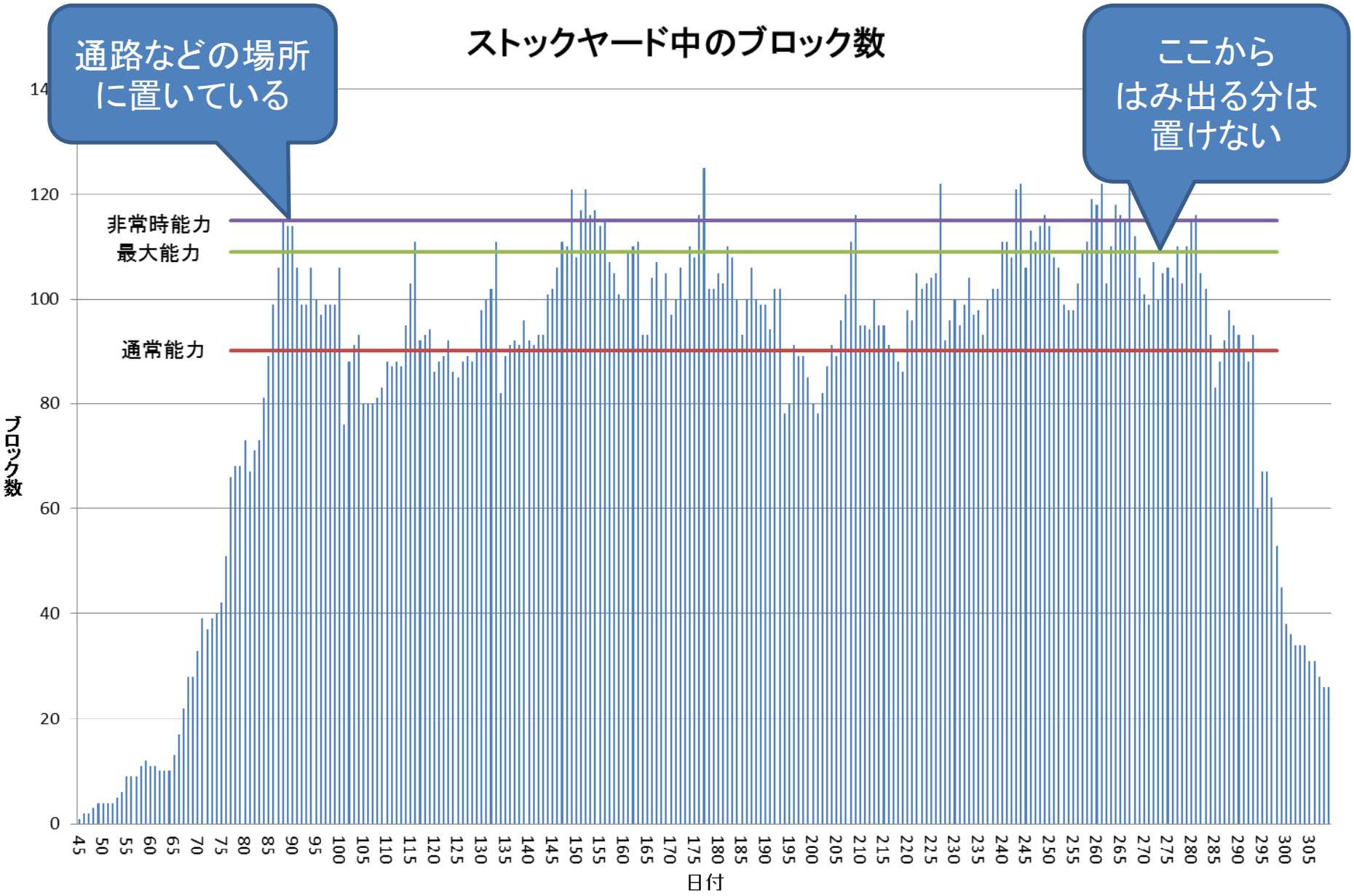
合計で45組のスタック
109個分のブロック置場

スケジュール期間： 264日

扱うブロック数： 3061個

ストックヤード中のブロック数




ストックヤード中のブロック数
通路などの場所
に置いている
ここから
はみ出る分は
置けない
非常時能力
最大能力
通常能力
ブロック数
日付
0
20
40
60
80
100
120
14
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305

シミュレーション結果：
完全な理想スタック蔵置ルール

	蔵置されなかったブロック数	蔵置されなかったのべブロック数	計算時間
分枝限定法1（枝数300）	139	1030	3～4時間
分枝限定法1（枝数600）	149	1058	3～4時間
分枝限定法2（枝数500）	213	1126	6～10時間
分枝限定法2（枝数1000）	213	1118	6～10時間

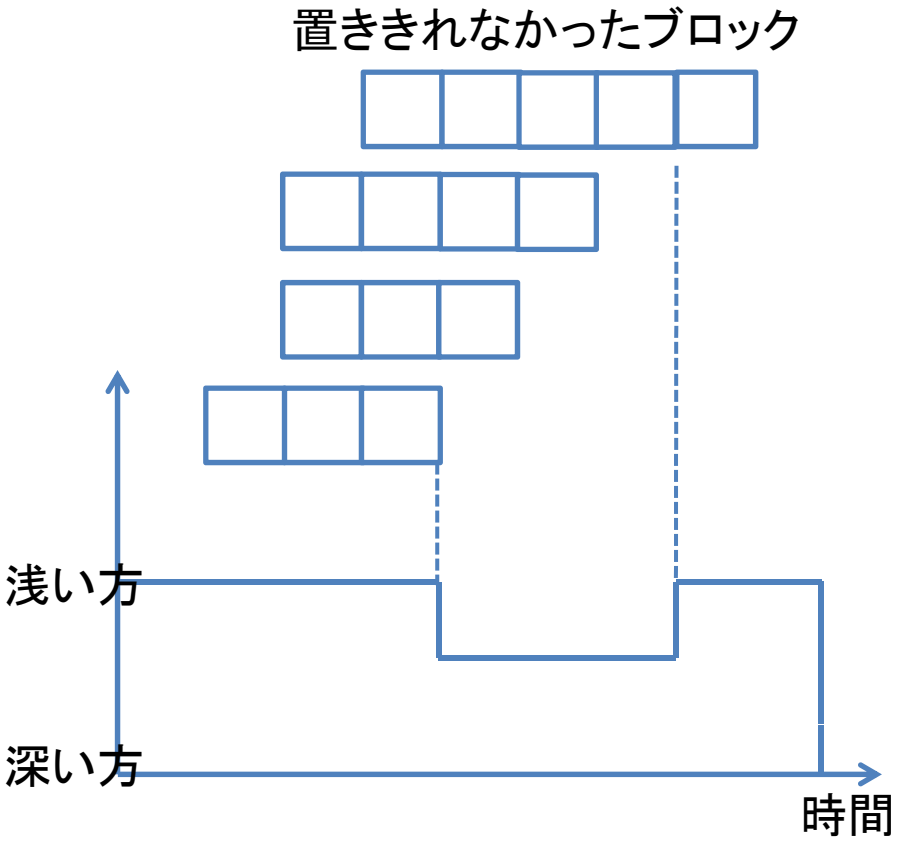
- 分枝限定法2について、PCのメモリ容量の制約により枝の数1000が限界。結果は分枝限定法1より劣る

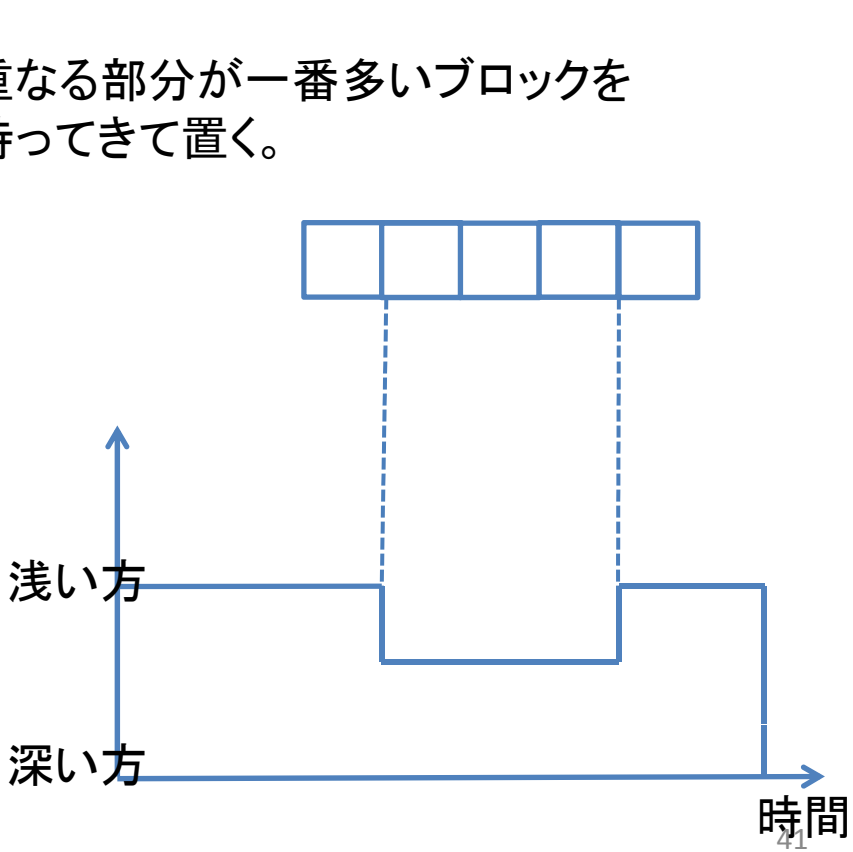
  → コンピュータの性能がよければもっといい結果が出る可能性

- 深いスタックからスケジュールするのは意外と強力なヒューリスティクス

理想スタック蔵置ルールを守った場合に置ききれなかったブロックの分割蔵置

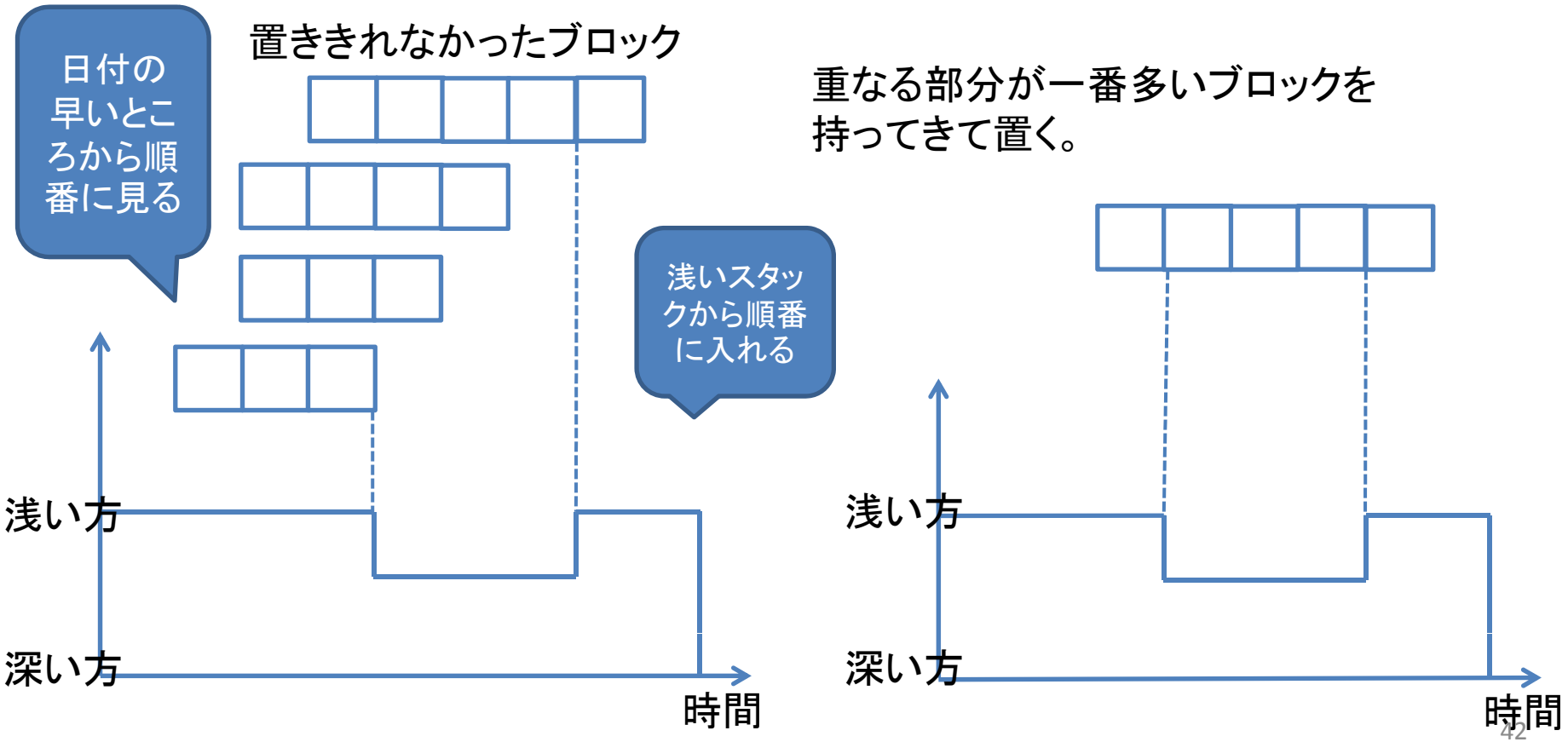
- 一番いい結果と思われる分枝限定法1（枝の数300）の結果を元に、置ききれなかったブロックの期間を分割し、空いているスペースに置いた。





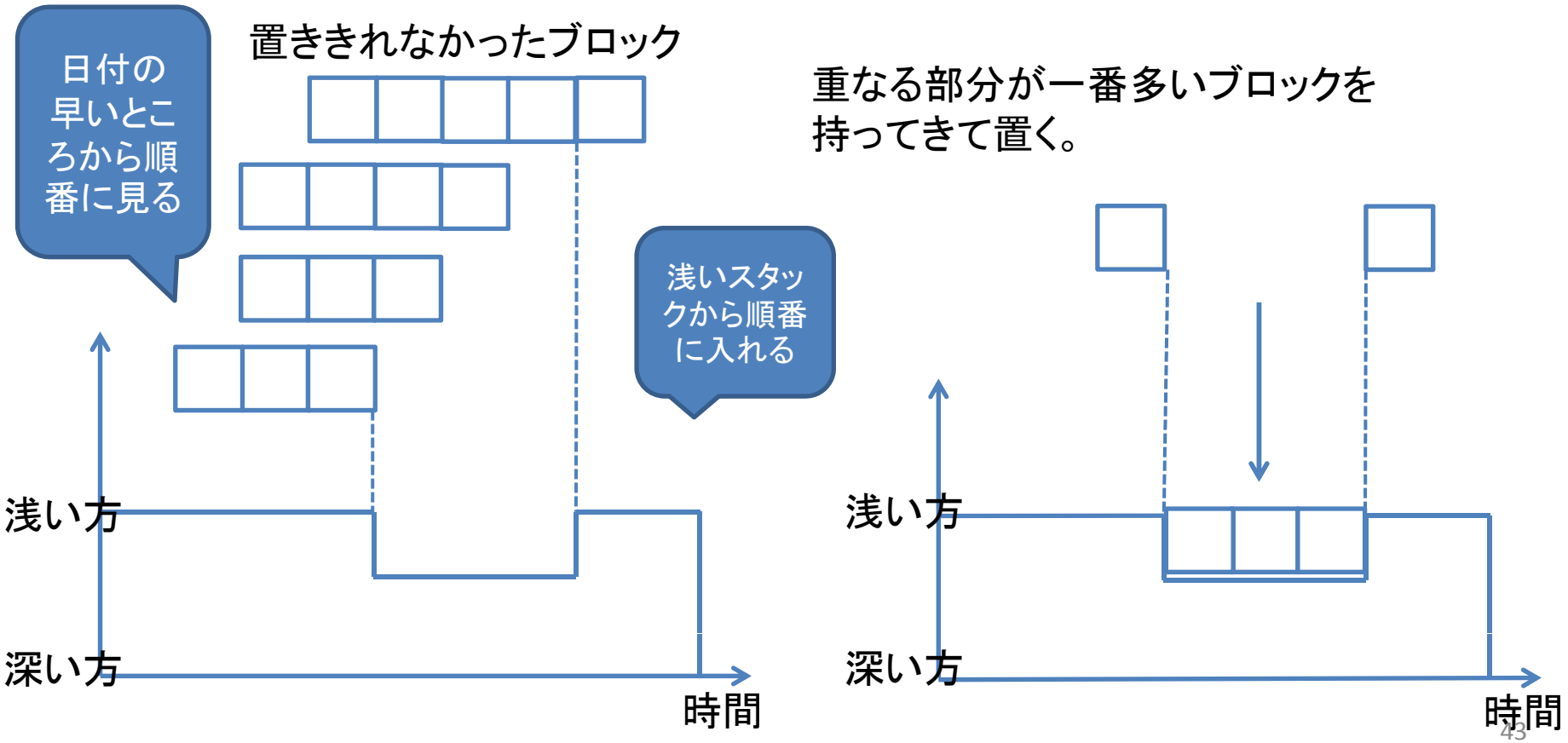
理想スタック蔵置ルールを守った場合に置ききれなかったブロックの分割蔵置

- 一番いい結果と思われる分枝限定法1（枝の数300）の結果を元に、置ききれなかったブロックの期間を分割し、空いているスペースに置いた。



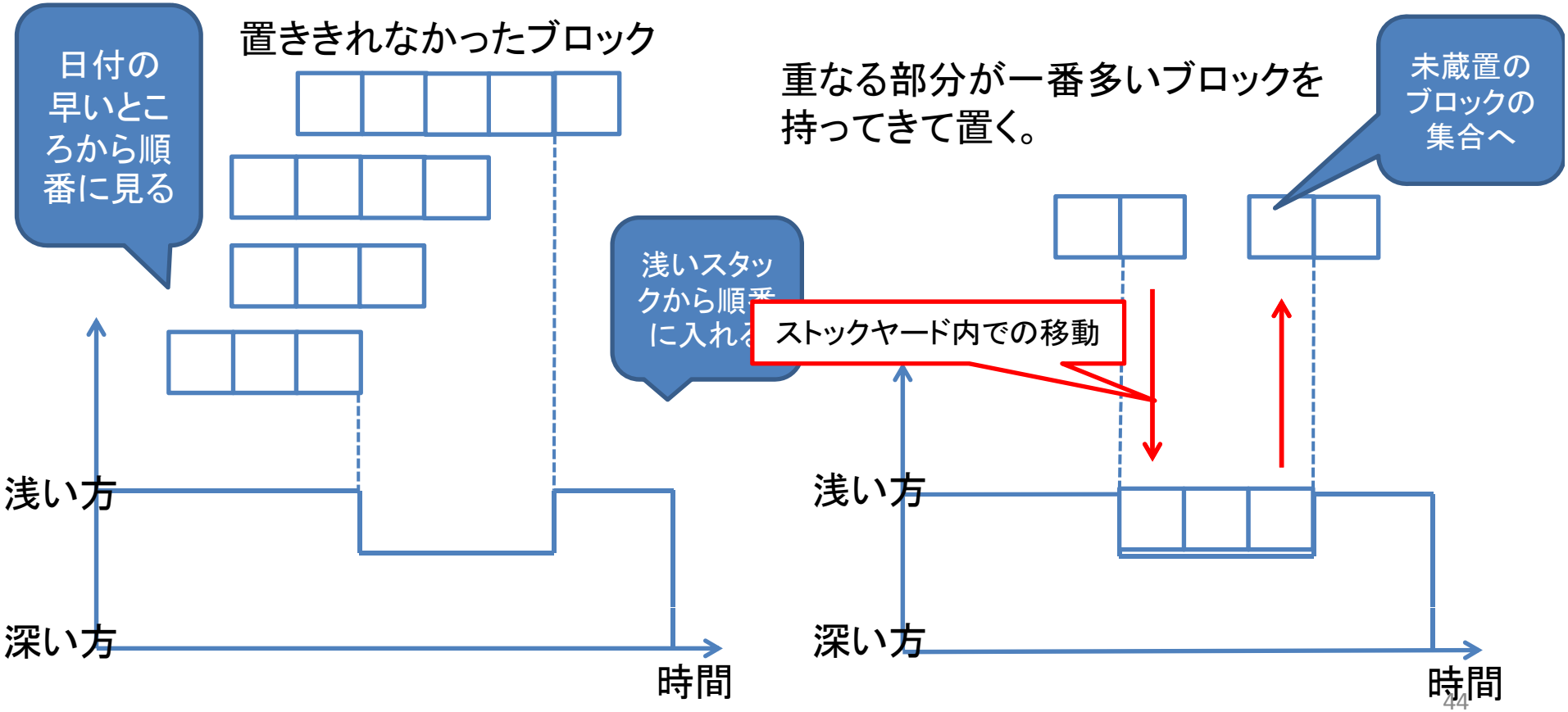
理想スタック蔵置ルールを守った場合に置ききれなかったブロックの分割蔵置

- 一番いい結果と思われる分枝限定法1（枝の数300）の結果を元に、置ききれなかったブロックの期間を分割し、空いているスペースに置いた。



理想スタック蔵置ルールを守った場合に置ききれなかったブロックの分割蔵置

- 一番いい結果と思われる分枝限定法1（枝の数300）の結果を元に、置ききれなかったブロックの期間を分割し、空いているスペースに置いた。

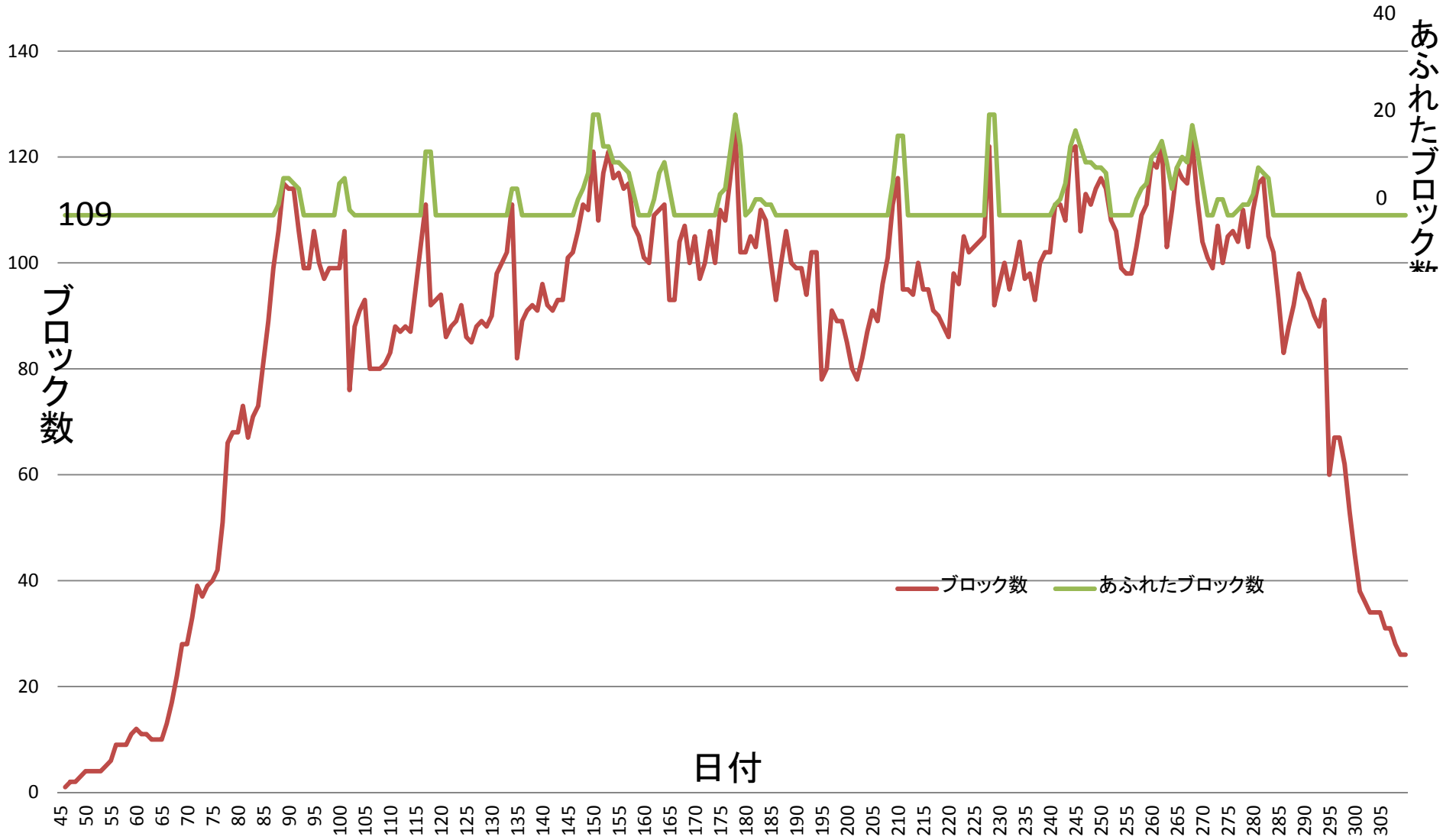


置き切れなかったブロックの分割蔵置を追加した場合のシミュレーション結果：

	蔵置されなかったブロック数	蔵置されなかったのべブロック数	計算時間
分枝限定法1（枝数300）	139	1030	3～4時間
分枝限定法1（枝数300）＋ブロック蔵置期間分割	192	629	3～4時間

- ブロック蔵置期間の分割は、ストックヤード内での移動を伴い、好ましくない。
- 未蔵置 のべブロック数がほぼ半減
- 持ってくるブロックの選び方は最適ではない。適当に置いている。
  → 何らかの最適化を駆使するなど、改善の余地あり。
- 手動で大変だったスケジューリングが自動化された

ストックヤード内のブロック数と、
スケジューリングにより置ききれなかったブロック数の比較



関連研究①

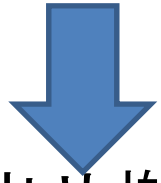
- 本研究の問題は、蔵置期間が長いものを下に置き、短いものを上に置くといった点で埠頭のコンテナヤード問題に似ている。

- 樋口、阿部、伊藤らは、ファジイ理論を用いて埠頭のコンテナヤード問題に取り組んでいる。

関連研究①

- 本研究の問題は、蔵置期間が長いものを下に置き、短いものを上に置くといった点で埠頭のコンテナヤード問題に似ている。

- 樋口、阿部、伊藤らは、ファジイ理論を用いて埠頭のコンテナヤード問題に取り組んでいる。



- しかし、コンテナの搬出や搬入の時期が本研究の様に確定的ではなく、かなり余裕のある話。本研究では使えない。

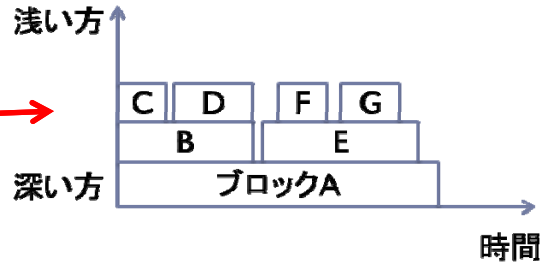
関連研究②

- また、田房、井手らは、「造船組立におけるブロック配置システムの開発」において、ブロックの配置の効率化をWebアプリケーションによって支援するシステム開発を行っている。

- 自動配置と称している部分があったが、スタックへの配置を最適化したものではない。

まとめ

・スタック構造を有するブロックストックヤードにおいて、
　ブロックが無駄な動きをしない適切な置場の指示方法



→　理想スタック蔵置ルール
→　搬入日と蔵置期間でソート
→　ソートしたブロックに対してスケジューリング
　（1）　段数の深いスタックから分枝限定法で割り当て
　（2）　全スタックを考慮して分枝限定法で割り当て
→　置き切れなかったブロックの期間を分割して割当て

・自動化システムを構築・シミュレーション
（1）の方法＋分割蔵置が最も効率良くスケジュール

今後の課題

- あふれたブロックの**期間分割蔵置では、最適化をしていない**
  → 何らかの最適化が必要

- **ストックに余裕がある期間は、ギリギリに詰める必要がない**
  → 別の評価項目（ブロックの移動距離など）を定めてスケジュール

- コンピュータの性能があれば分枝限定法2でより良い結果が出る可能性

- 途中で計画が変わった場合などに、そこから**再スケジュール**する機能

- ブロック置場は、**並行部ブロック置場**と**曲がり部ブロック置場**に分かれており、
  曲がり部ブロックは並行部ブロック置場に置けるが、逆はできない等の制約
  これらを考慮に入れたスケジューリング