

Optimizing Item Loading and Retrieval

Shunji Tanaka

Okayama University

shunji.tanaka@okayama-u.ac.jp

October 21, 2026 @ Scheduling Seminar

Table of Contents

Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary

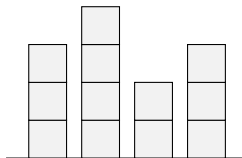
- 1 Introduction
- 2 Related Problems
- 3 Block Relocation Problem
- 4 Parallel Stack Loading Problem
- 5 Summary

Item Stacking and Retrieval

Items (boxes, containers, and plates, etc.)

- Arranged in stacks within storage facilities (warehouses, storage yards, etc.)
- Only topmost items are accessible (last-in, first-out)
- Retrieving a non-topmost item requires relocating all items above it

How to reduce unnecessary relocations by clever loading and/or retrieval?

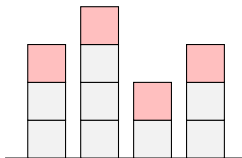


Item Stacking and Retrieval

Items (boxes, containers, and plates, etc.)

- Arranged in stacks within storage facilities (warehouses, storage yards, etc.)
- Only **topmost items** are accessible (last-in, first-out)
- Retrieving a non-topmost item requires relocating all items above it

How to reduce unnecessary relocations by clever loading and/or retrieval?

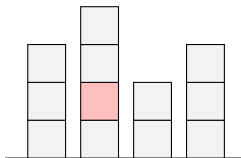


Item Stacking and Retrieval

Items (boxes, containers, and plates, etc.)

- Arranged in stacks within storage facilities (warehouses, storage yards, etc.)
- Only topmost items are accessible (last-in, first-out)
- Retrieving **a non-topmost item** requires relocating all items above it

How to reduce unnecessary relocations by clever loading and/or retrieval?

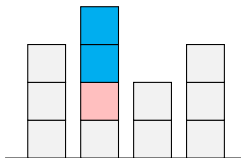


Item Stacking and Retrieval

Items (boxes, containers, and plates, etc.)

- Arranged in stacks within storage facilities (warehouses, storage yards, etc.)
- Only topmost items are accessible (last-in, first-out)
- Retrieving a **non-topmost item** requires relocating **all items above it**

How to reduce unnecessary relocations by clever loading and/or retrieval?



Introduction

Item Stacking and
Retrieval

Storage Yard Example

Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

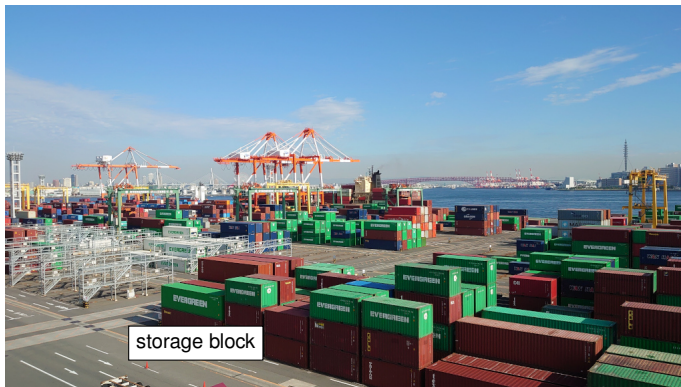
Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary

Typical Example: Container Storage Yard



Container terminal at seaports

- Connects maritime and land transport
- Offers temporary storage for containers

Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

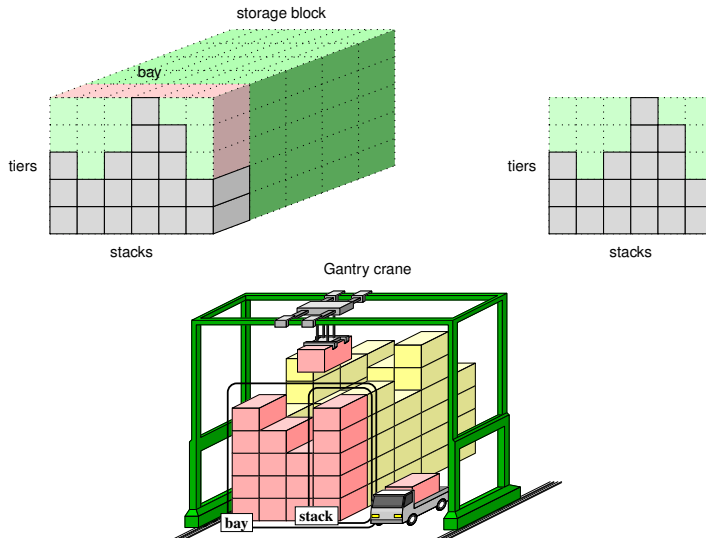
Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary

Bays and Stacks



Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

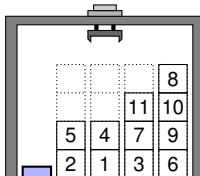
Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary

Container Retrieval



number: retrieval order

Container retrieval order

- Determined by ship stowage plans (outbound) or truck appointments (inbound).
 - $\neq$ (Arrival order)
-
- Only topmost containers are accessible
⇒ Those above the target container must be relocated beforehand.
 - Container movement between different bays is avoided due to safety guidelines
⇒ Relocations within a bay

Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem
Summary

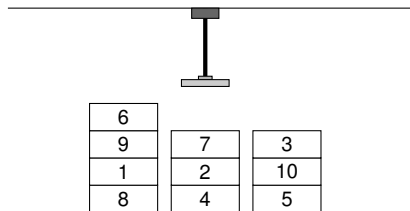
Slab Storage Yard

Slab

- Steel plates in steel production
- Temporarily stacked in slab storage yards (similar to containers)



Laurentiu Iordache/Alamy



Three Types of Storage Yard Problems

Loading/stacking

- Stack incoming items
- No outgoing items
- Minimize efforts required for future retrieval

Pre-marshalling and re-marshalling

- No incoming or outgoing items
- Rearrange items with the minimum efforts so that items can be retrieved without any relocations
(pre-marshalling: within a bay, re-marshalling: across bays)

Retrieval

- No incoming items
- Retrieve items in a specified order with the minimum efforts

Three Types of Storage Yard Problems

Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary

Loading/stacking: container loading problem

- Stack incoming items
- No outgoing items
- Minimize efforts required for future retrieval

Pre-marshalling and re-marshalling

- No incoming or outgoing items
- Rearrange items with the minimum efforts so that items can be retrieved without any relocations
(pre-marshalling: within a bay, re-marshalling: across bays)

Retrieval

- No incoming items
- Retrieve items in a specified order with the minimum efforts

Example of Container Loading Problem

Introduction

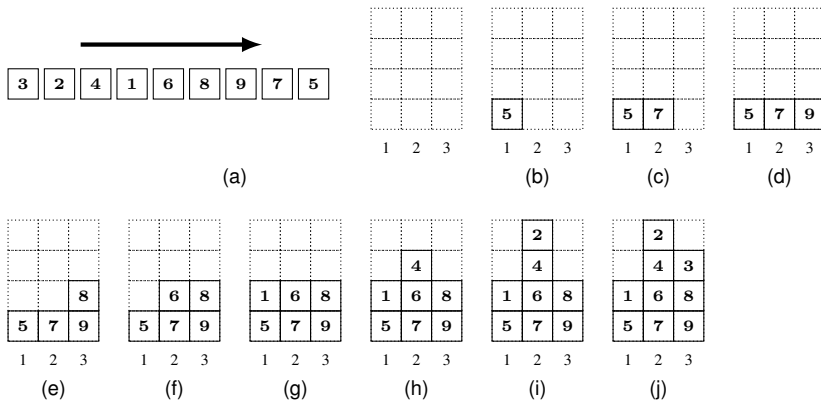
Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary



Example of Container Loading Problem

Introduction

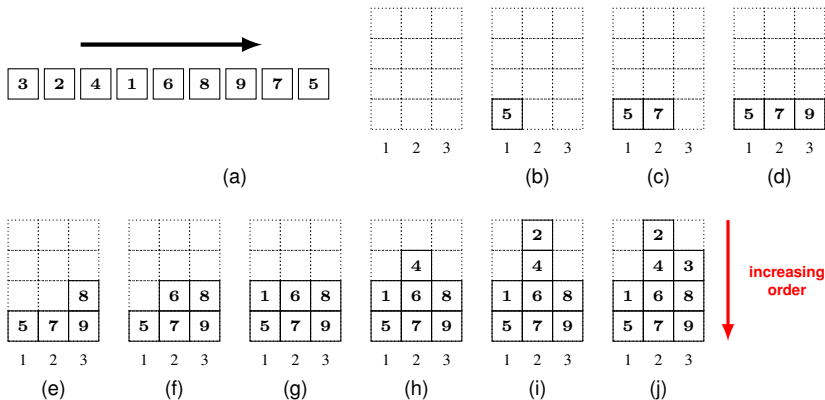
Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Summary



Three Types of Storage Yard Problems

Loading/stacking

- Stack incoming items
- No outgoing items
- Minimize efforts required for future retrieval

Pre-marshalling and re-marshalling: container pre-marshalling problem

- No incoming or outgoing items
- Rearrange items with the minimum efforts so that items can be retrieved without any relocations
(pre-marshalling: within a bay, re-marshalling: across bays)

Retrieval

- No incoming items
- Retrieve items in a specified order with the minimum efforts

Example of Container Pre-marshalling Problem

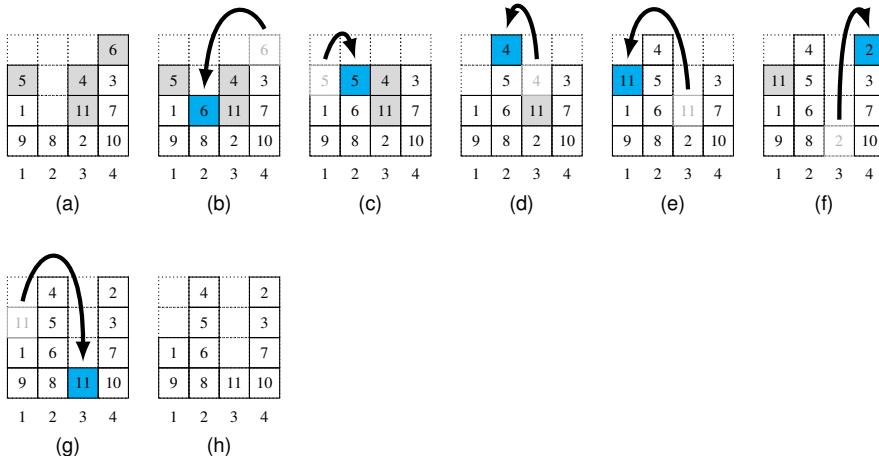
Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem
Summary



(minimum number of relocations)= 6

Example of Container Pre-marshalling Problem

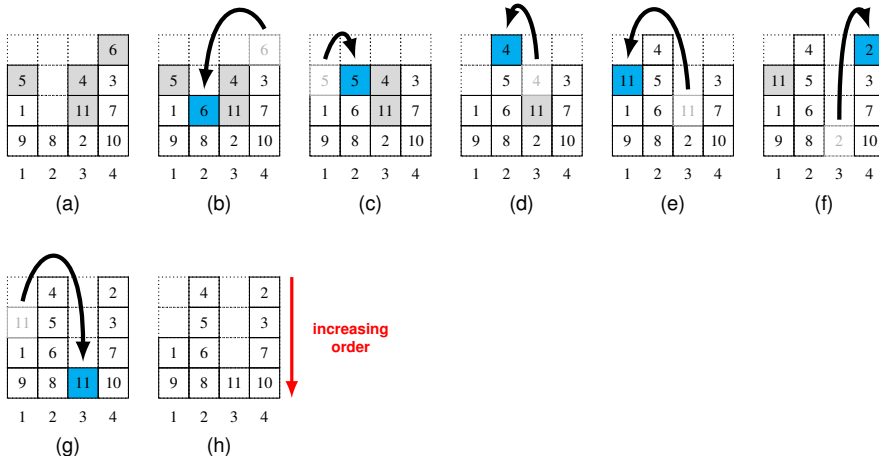
Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem
Summary



(minimum number of relocations)= 6

Three Types of Storage Yard Problems

Loading/stacking

- Stack incoming items
- No outgoing items
- Minimize efforts required for future retrieval

Pre-marshalling and re-marshalling

- No incoming or outgoing items
- Rearrange items with the minimum efforts so that items can be retrieved without any relocations
(pre-marshalling: within a bay, re-marshalling: across bays)

Retrieval: block (container) relocation problem

- No incoming items
- Retrieve items in a specified order with the minimum efforts

Example of Block Relocation Problem

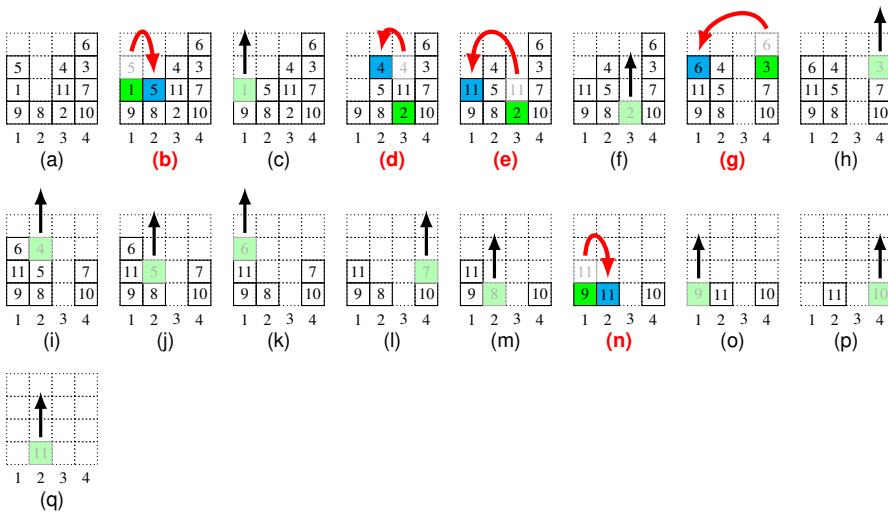
Introduction

Item Stacking and
Retrieval
Storage Yard Example
Bays and Stacks
Container Retrieval
Slab Storage Yard
Storage Yard Problems
CLP Example
Storage Yard Problems
CPMP Example
Storage Yard Problems
BRP Example

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem
Summary



(minimum number of relocations)= 5

Table of Contents

- 1 Introduction
- 2 Related Problems
- 3 Block Relocation Problem
- 4 Parallel Stack Loading Problem
- 5 Summary

Blocks World

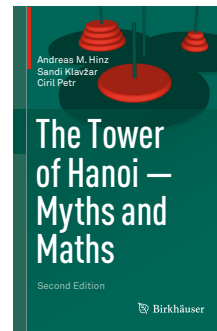
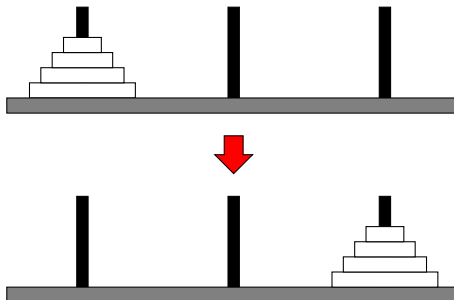
- Studied mainly in the AI community 20+ yrs ago (cf. Slaney and S. Thiébaux, 2001)
- Rearrange blocks into a desired configuration
- Similar to the pre-marshalling problem
- The standard problem does not restrict the number of available stacks



J. Slaney and S. Thiébaux: Blocks World revisited, Artificial Intelligence 125, 119/153, 2001.

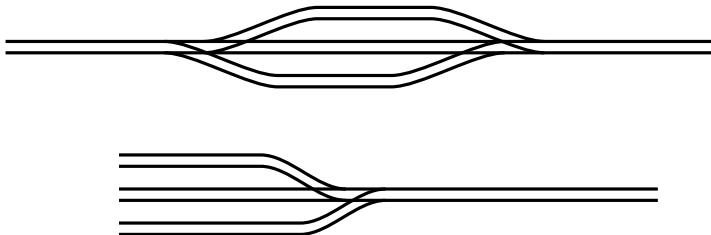
Tower of Hanoi

- Three pegs in general
- Move discs from the leftmost peg to the rightmost peg
- Similar to the pre-marshalling problem
- A larger disc cannot be stacked on a smaller disc



Train Shunting (Marshalling) Problem

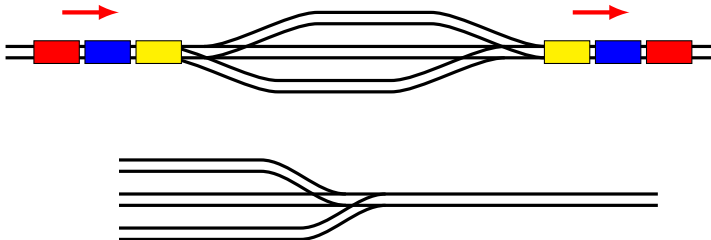
- Rearrange train cars in a shunting yard
- Both arrival and departure are considered



N. Boysen et al.: Shunting yard operations: Theoretical aspects and applications, European Journal of Operational Research 220, 1/14 (2020).

Train Shunting (Marshalling) Problem

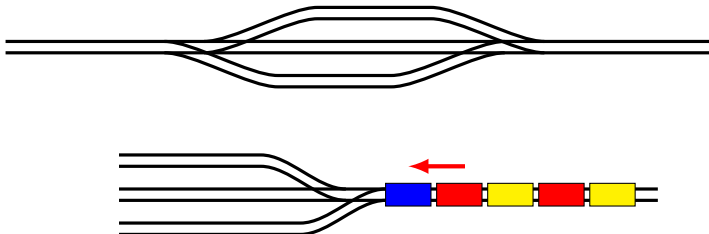
- Rearrange train cars in a shunting yard
- Both arrival and departure are considered



N. Boysen et al.: Shunting yard operations: Theoretical aspects and applications, European Journal of Operational Research 220, 1/14 (2020).

Train Shunting (Marshalling) Problem

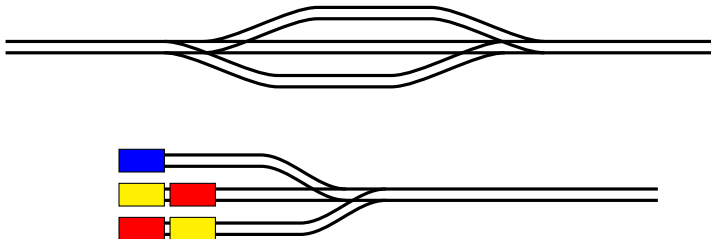
- Rearrange train cars in a shunting yard
- Both arrival and departure are considered



N. Boysen et al.: Shunting yard operations: Theoretical aspects and applications, European Journal of Operational Research 220, 1/14 (2020).

Train Shunting (Marshalling) Problem

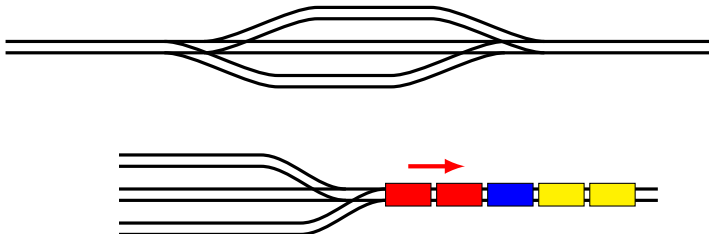
- Rearrange train cars in a shunting yard
- Both arrival and departure are considered



N. Boysen et al.: Shunting yard operations: Theoretical aspects and applications, European Journal of Operational Research 220, 1/14 (2020).

Train Shunting (Marshalling) Problem

- Rearrange train cars in a shunting yard
- Both arrival and departure are considered



N. Boysen et al.: Shunting yard operations: Theoretical aspects and applications, European Journal of Operational Research 220, 1/14 (2020).

Stack Sorting

Stack Sorting

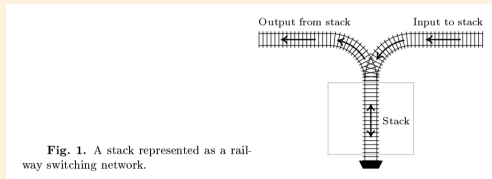


Fig. 1. A stack represented as a railway switching network.

(D. Knuth: The Art of Computer Programming Volume 1, 3rd Ed., 1997)

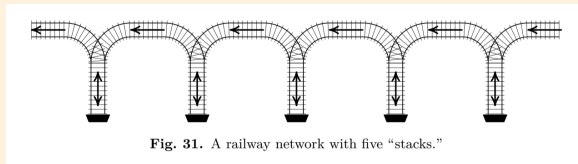


Fig. 31. A railway network with five "stacks."

(D. Knuth: The Art of Computer Programming Volume 3, 2nd Ed., 1998)

S. Even and A. Itai: Queues, stacks and graphs, Proceedings of an International Symposium on the Theory of Machine and Computations, 71/86 (1971).

R. Tarjan: Sorting using networks of queues and stacks, Journal of the Association for Computing Machinery 19, 341/346 (1972).

Today's Topics

Loading/stacking: **parallel stack loading problem**

- Stack incoming items
- No outgoing items
- Minimize efforts required for future retrieval

Pre-marshalling and re-marshalling

- No incoming or outgoing items
- Rearrange items with the minimum efforts so that items can be retrieved without any relocations
(pre-marshalling: within a bay, re-marshalling: across bays)

Retrieval: **block relocation problem**

- No incoming items
- Retrieve items in a specified order with the minimum efforts

Table of Contents

- 1 Introduction
- 2 Related Problems
- 3 Block Relocation Problem**
- 4 Parallel Stack Loading Problem
- 5 Summary

Definition of BRP

$(H =)$

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4(= S)

S : number of stacks
 H : stack height (capacity)
 number: priority

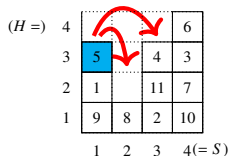
Block (Container) Relocation Problem

Retrieve N blocks from the bay in ascending order of their priority:
 block 1, block 2, ..., block N .

- Relocation
a block on top of a stack is moved to the top of another stack
- Retrieval
a block with the highest priority is removed if it is on top of a stack

Objective: minimization of the **total number of relocations**

Definition of BRP



S : number of stacks
 H : stack height (capacity)
 number: priority

Block (Container) Relocation Problem

Retrieve N blocks from the bay in ascending order of their priority:
 block 1, block 2, $\dots$, block N .

- Relocation
 a block on top of a stack is moved to the top of another stack
- Retrieval
 a block with the highest priority is removed if it is on top of a stack

Objective: minimization of the **total number of relocations**

Definition of BRP

$(H =)$

4				6
3			4	3
2	1	5	11	7
1	9	8	2	10
	1	2	3	4(=S)

A red arrow points upwards from the cell containing block 1 (row 2, column 1) to the top of the stack.

S : number of stacks
 H : stack height (capacity)
 number: priority

Block (Container) Relocation Problem

Retrieve N blocks from the bay in ascending order of their priority:
 block 1, block 2, ..., block N .

- Relocation
a block on top of a stack is moved to the top of another stack
- Retrieval
a block with the highest priority is removed if it is on top of a stack

Objective: minimization of the **total number of relocations**

Definition of BRP

$(H =)$

4				6
3			4	3
2		5	11	7
1	9	8	2	10
	1	2	3	4(= S)

S : number of stacks
 H : stack height (capacity)
 number: priority

Block (Container) Relocation Problem

Retrieve N blocks from the bay in ascending order of their priority:
 block 1, block 2, ..., block N .

- Relocation
a block on top of a stack is moved to the top of another stack
- Retrieval
a block with the highest priority is removed if it is on top of a stack

Objective: minimization of the **total number of relocations**

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. Group Priorities
- Unrestricted vs. Restricted
 - Unrestricted
All the topmost blocks are relocatable.
 - Restricted
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. Group Priorities
- Unrestricted vs. Restricted
 - Unrestricted
All the topmost blocks are relocatable.
 - Restricted
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. **Group** Priorities
- Unrestricted vs. Restricted
 - Unrestricted
All the topmost blocks are relocatable.
 - Restricted
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. Group Priorities
- **Unrestricted** vs. Restricted
 - **Unrestricted**
All the topmost blocks are relocatable.
 - **Restricted**
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- **Distinct** vs. Group Priorities
- Unrestricted vs. **Restricted**
 - Unrestricted
All the topmost blocks are relocatable.
 - **Restricted**
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. **Group** Priorities
- Unrestricted vs. **Restricted**
 - Unrestricted
All the topmost blocks are relocatable.
 - **Restricted**
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. **Group** Priorities
- Unrestricted vs. Restricted
 - Unrestricted
All the topmost blocks are relocatable.
 - **Restricted**
Only the topmost block of the stack having the next retrieved block is relocatable.

Classification of BRP

4				6
3	5		4	3
2	1		11	7
1	9	8	2	10
	1	2	3	4

Distinct Priorities

4				3
3	3		2	1
2	1		6	3
1	5	4	1	5
	1	2	3	4

Group Priorities

Classification of BRP

- Distinct vs. **Group** Priorities
- Unrestricted vs. Restricted
 - Unrestricted
All the topmost blocks are relocatable.
 - **Restricted**
Only the topmost block of the stack having the next retrieved block is relocatable.

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark
Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

5 8 must be relocated before retrieving 1

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

5 8 must be relocated before retrieving 1

9 must be relocated before retrieving 2

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

5 8 must be relocated before retrieving 1

9 must be relocated before retrieving 2

10 must be relocated before retrieving 3

Badly-placed and Well-placed Blocks

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

5			10
8			3
1	7	9	12
6	11	2	4

5 8 must be relocated before retrieving 1

9 must be relocated before retrieving 2

10 must be relocated before retrieving 3

12 must be relocated before retrieving 4

Badly-placed and Well-placed Blocks

5			10
8			3
1	7	9	12
6	11	2	4

5 8 must be relocated before retrieving 1

9 must be relocated before retrieving 2

10 must be relocated before retrieving 3

12 must be relocated before retrieving 4

Badly-placed (Blocking) Blocks 5 8 9 10 12

Badly-placed: a block beneath which is a block to be retrieved earlier

Well-placed: 1 2 3 4 6 7 11

Intuitive lower bound on the number of relocations [Kim and Hong, 2006]

- (number of badly-placed blocks) = 5
- Easy to compute

Computational Complexity of BRP

Computational Complexity of BRP

$N \leq S$:

- Polynomially solvable
- Each badly-placed block can be relocated to an empty stack and is never relocated again

$N > S$ and $6 \leq H < \infty$:

- NP-hard (restricted and unrestricted) [Caserta et al. 2012]
- Reduction from MES (Mutual Exclusion Scheduling) on permutation graphs

MES on permutation graphs (NP-complete when $m \geq 6$ [Jansen, 2003])

Given a permutation graph $G = (V, E)$ and positive integers m and t , is there a partition of V into t independent sets $V_1, V_2, \dots, V_t$ with $|V_i| \leq m$?

Reduction from MES on Permutation Graphs

6			
3			
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23

- Badly-placed **6 3 7 8 2 5** must be relocated before retrieving **1**
- The total number of relocations is 6, if they become well placed after relocation

Reduction from MES on Permutation Graphs

6			
3			
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23



- Badly-placed **6** **3** **7** **8** **2** **5** must be relocated before retrieving **1**
- The total number of relocations is 6, if they become well placed after relocation
- **7** becomes badly placed, if it is relocated above **6**

Reduction from MES on Permutation Graphs

6			
3			
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23



- Badly-placed **6** **3** **7** **8** **2** **5** must be relocated before retrieving **1**
- The total number of relocations is 6, if they become well placed after relocation
- **7** becomes badly placed, if it is relocated above **6** $\Rightarrow$ ⑥ — ⑦

Reduction from MES on Permutation Graphs

6			
3			
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23



- Badly-placed **6** **3** **7** **8** **2** **5** must be relocated before retrieving **1**
- The total number of relocations is 6, if they become well placed after relocation
- **7** becomes badly placed, if it is relocated above **6** $\Rightarrow$ **6** — **7**

Reduction from MES on Permutation Graphs

Introduction

Related Problems

Block Relocation Problem

Definition of BRP

Classification of BRP

Badly-placed and

Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

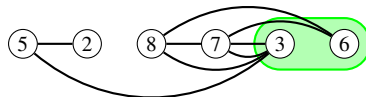
Results for Benchmark

Instances

Parallel Stack Loading Problem

Summary

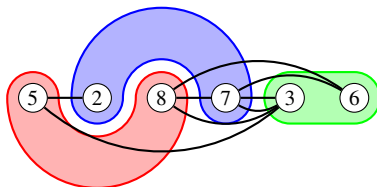
6	3		
3	6		
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23



- Badly-placed **6** **3** **7** **8** **2** **5** must be relocated before retrieving **1**
- The total number of relocations is 6, if they become well placed after relocation
- **7** becomes badly placed, if it is relocated above **6** $\Rightarrow$ **6** — **7**
- Well-placed blocks in a stack $\Leftrightarrow$ An independent set

Reduction from MES on Permutation Graphs

6	3	2	5
3	6	7	8
7	9	10	11
8	12	13	14
2	15	16	17
5	18	19	20
1	21	22	23



- Badly-placed $\boxed{6} \boxed{3} \boxed{7} \boxed{8} \boxed{2} \boxed{5}$ must be relocated before retrieving $\boxed{1}$
- The total number of relocations is 6, if they become well placed after relocation
- $\boxed{7}$ becomes badly placed, if it is relocated above $\boxed{6} \Rightarrow \textcircled{6} - \textcircled{7}$
- Well-placed blocks in a stack $\Leftrightarrow$ An independent set
- $(\# \text{relocations}) = 6 \Leftrightarrow$ Answer to MES is yes

Exact Approaches to the BRP

Article	Method	Restricted		Unrestricted	
		Dist.	Group	Dist.	Group
Kim and Hong (2006)	B&B	✓	✓		
Wan (2009)	IP	✓			
Ünlüyurt and Aydın (2012)	B&B	✓			
Caserta et al. (2012)	IP	✓		✓	
Zhu et al. (2012)	IDA*	✓		✓	
Petering and Hussein (2013)	MIP			✓	
Zehendner and Feillet (2014)	B&P	✓			
Expósito-Izquierdo et al. (2014)	A*, IP	✓		✓	
Expósito-Izquierdo et al. (2015)	B&B	✓			
Zehendner et al. (2015)	IP	✓			
Eskandari and Azari (2015)	IP	✓			
Tanaka and Takii (2016)	IDB&B	✓	✓		
Ku and Arthanari (2016)	B&B	✓			
Tricoire et al. (2018)	B&B			✓	
de Melo da Silva et al. (2018)	B&B		✓ [†]		
Galle et al. (2018)	IP	✓			
Tanaka and Mizuno (2018)	IDB&B	✓		✓	
Quispe et al. (2018)	B&B	✓			
Lu et al. (2020)	IP			✓	
Bacci et al. (2020)	B&C	✓			
Tanaka and Voß (2021)	IP-based	✓			
Liu et al. (2022)	IP			✓	
Jin and Tanaka (2023)	IDB&B			✓	✓

ID: Iterative Deepening

[†] BRP with only two priority values

- Most approaches are based on branch and bound
- State-of-the-art IP-based algorithm for the restricted BRP [Tanaka and Voß, 2021]

Exact Approaches to the BRP

Article	Method	Restricted		Unrestricted	
		Dist.	Group	Dist.	Group
Kim and Hong (2006)	B&B	✓	✓		
Wan (2009)	IP	✓			
Ünlüyurt and Aydın (2012)	B&B	✓			
Caserta et al. (2012)	IP	✓		✓	
Zhu et al. (2012)	IDA*	✓		✓	
Petering and Hussein (2013)	MIP			✓	
Zehendner and Feillet (2014)	B&P	✓			
Expósito-Izquierdo et al. (2014)	A*, IP	✓		✓	
Expósito-Izquierdo et al. (2015)	B&B	✓			
Zehendner et al. (2015)	IP	✓			
Eskandari and Azari (2015)	IP	✓			
Tanaka and Takii (2016)	IDB&B	✓	✓		
Ku and Arthanari (2016)	B&B	✓			
Tricoire et al. (2018)	B&B			✓	
de Melo da Silva et al. (2018)	B&B		✓ [†]		
Galle et al. (2018)	IP	✓			
Tanaka and Mizuno (2018)	IDB&B	✓		✓	
Quispe et al. (2018)	B&B	✓			
Lu et al. (2020)	IP			✓	
Bacci et al. (2020)	B&C	✓			
Tanaka and Voß (2021)	IP-based	✓			
Liu et al. (2022)	IP			✓	
Jin and Tanaka (2023)	IDB&B			✓	✓

ID: Iterative Deepening

[†] BRP with only two priority values

- Most approaches are based on branch and bound
- State-of-the-art IP-based algorithm for the restricted BRP [Tanaka and Voß, 2021]

IP Formulation: Basic Idea

			6
5		4	3
1		11	7
9	8	2	10

(a)

			6
5		4	3
1	5	11	7
9	8	2	10

(b) 5 (1, 1, 2)

			6
	4	4	3
	5	11	7
9	8	2	10

(c) 4 (2, 3, 2)

			6
	4		3
11	5	11	7
9	8	2	10

(d) 11 (2, 3, 1)

			6
6	4		3
11	5		7
9	8		10

(e) 6 (3, 4, 1)

11			
9	11		10

(f) 11 (9, 1, 2)

Relocation sequences

- Only badly-placed blocks are relocated
- A relocation is denoted by (target block, src stack, dst stack)

5: (1, 1, 2)

11: (2, 3, 1), (9, 1, 2)

4: (2, 3, 2)

6: (3, 4, 1)

IP Formulation Based on Relocation Sequences

$$\min \sum_{i \in \mathcal{B}} \sum_{\pi_{ik} \in \mathcal{E}_i} R(\pi_{ik}) x_{ik},$$

$$\text{s.t.} \quad \sum_{\pi_{ik} \in \mathcal{E}_i} x_{ik} = 1, \quad i \in \mathcal{B}, \quad (A)$$

$$\sum_{\substack{\pi_{jl} \in \mathcal{E}_j \\ (\pi_{jl}, \pi_{ik}) \in \mathcal{G}}} x_{jl} + x_{ik} \leq 1, \quad \pi_{ik} \in \mathcal{E}_i, \quad i, j \in \mathcal{B}, \quad j < i, \quad (B)$$

(constraints on the stack capacity)

$$x_{ik} \in \{0, 1\}, \quad \pi_{ik} \in \mathcal{E}_i, \quad i \in \mathcal{B}.$$

$\mathcal{B}$: Set of badly-placed blocks

$\mathcal{E}_i$: Set of relocation sequences π_{ik} of blocks $i \in \mathcal{B}$

$\mathcal{G}$: Graph representing conflicts between sequences

$R(\pi_{ik})$: Number of relocations in sequence π_{ik}

(A): assignment constraints

(B): aggregated conflict constraints (a pair of sequences may be inconsistent)

IP formulations for lower and upper bounds

The IP formulation is difficult to handle because of an exponential number of relocation sequences

IP formulation for lower bound (IP-LB)

Truncated (incomplete) sequences are permitted

Example of truncated sequence

11: (2, 3, •) $\left\{ \begin{array}{l} \text{The destination stack of the first relocation is arbitrary} \\ \text{The sequence length is at least one} \end{array} \right.$

IP formulation for upper bound (IP-UB)

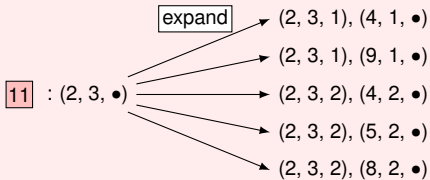
(IP-LB) without truncated sequences

(IP-LB) and (IP-UB) are solved by turns while truncated sequences are gradually refined

Algorithm

Algorithm

1. Solve (IP-LB) to compute a lower bound LB.
2. Expand truncated sequences in the solution.



3. Update (IP-LB) (with expanded sequences and updated constraints).
4. Solve (IP-UB) to compute an upper bound UB.
5. If $LB = UB$, terminate. Otherwise, go to 1.

Computational Results

Methods

IDB&B: Iterative deepening branch-and-bound [Tanaka and Takii, 2016][Tanaka and Mizuno, 2018]

B&C: Branch-and-cut [Bacci et al. 2020]

IP-based: Proposed [Tanaka and Voß, 2021]

Experimental setup

Instances: Standard benchmark instances (CVS benchmark set)

Language: C++

IP Solver: Gurobi Optimizer 9.0.1

PC: desktop computer equipped with an Intel Core i9-9900K CPU and 64 GiB RAM

Time Limit: 3600 s per one instance

Results for CVS Benchmark Instances

Introduction

Related Problems

Block Relocation
Problem

Definition of BRP

Classification of BRP

Badly-placed and

Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark
Instances

Parallel Stack
Loading Problem

Summary

<i>H</i>	<i>S</i>	<i>N</i>	IDB&B			B&C			IP-based		
			time (s)			time (s)			time (s)		
			#solved	average	maximum	#solved	average	maximum	#solved	average	maximum
5	3	9	40	0.00	0.00	40	0.00	0.00	40	0.00	0.00
	4	12	40	0.00	0.00	40	0.00	0.00	40	0.00	0.00
	5	15	40	0.00	0.00	40	0.00	0.01	40	0.00	0.00
	6	18	40	0.00	0.00	40	0.01	0.02	40	0.00	0.00
	7	21	40	0.00	0.00	40	0.01	0.02	40	0.00	0.00
	8	24	40	0.00	0.00	40	0.02	0.03	40	0.00	0.01
6	4	16	40	0.00	0.00	40	0.00	0.01	40	0.00	0.00
	5	20	40	0.00	0.00	40	0.01	0.02	40	0.00	0.01
	6	24	40	0.00	0.00	40	0.02	0.05	40	0.00	0.02
	7	28	40	0.00	0.00	40	0.03	0.08	40	0.00	0.03
7	4	20	40	0.00	0.00	40	0.01	0.08	40	0.01	0.23
	5	25	40	0.00	0.01	40	0.04	0.33	40	0.02	0.22
	6	30	40	0.00	0.02	40	0.08	0.74	40	0.02	0.11
	7	35	40	0.01	0.13	40	0.11	0.47	40	0.02	0.10
	8	40	40	0.12	2.03	40	0.18	0.71	40	0.04	0.35
	9	45	40	0.16	1.68	40	0.48	4.97	40	0.04	0.32
	10	50	40	83.00	2859.04	40	0.59	2.97	40	0.10	0.96
8	6	36	40	0.89	15.05	40	0.88	8.67	40	0.24	1.45
	10	60	31	906.56	3600.00	40	15.92	233.66	40	0.80	6.32
12	6	60	12	2748.22	3600.00	20	2049.60	3600.00	35	937.06	3600.00
	10	100	0	3600.00	3600.00	2	3488.05	3600.00	29	1412.74	3600.00
total			763			782			824		

S: number of stacks, *H*: stack capacity, *N*: number of items

IP-based: 27 hours to solve the hardest instance with *N* = 100

Results for CVS Benchmark Instances

Introduction

Related Problems

Block Relocation
Problem

Definition of BRP

Classification of BRP

Badly-placed and

Well-placed

Complexity of BRP

Reduction from MES

Exact Approaches

Basic Idea

Formulation

IP for LB and UB

Algorithm

Computational Results

Results for Benchmark
Instances

Parallel Stack
Loading Problem

Summary

<i>H</i>	<i>S</i>	<i>N</i>	IDB&B			B&C			IP-based		
			time (s)			time (s)			time (s)		
			#solved	average	maximum	#solved	average	maximum	#solved	average	maximum
5	3	9	40	0.00	0.00	40	0.00	0.00	40	0.00	0.00
	4	12	40	0.00	0.00	40	0.00	0.00	40	0.00	0.00
	5	15	40	0.00	0.00	40	0.00	0.01	40	0.00	0.00
	6	18	40	0.00	0.00	40	0.01	0.02	40	0.00	0.00
	7	21	40	0.00	0.00	40	0.01	0.02	40	0.00	0.00
	8	24	40	0.00	0.00	40	0.02	0.03	40	0.00	0.01
6	4	16	40	0.00	0.00	40	0.00	0.01	40	0.00	0.00
	5	20	40	0.00	0.00	40	0.01	0.02	40	0.00	0.01
	6	24	40	0.00	0.00	40	0.02	0.05	40	0.00	0.02
	7	28	40	0.00	0.00	40	0.03	0.08	40	0.00	0.03
7	4	20	40	0.00	0.00	40	0.01	0.08	40	0.01	0.23
	5	25	40	0.00	0.01	40	0.04	0.33	40	0.02	0.22
	6	30	40	0.00	0.02	40	0.08	0.74	40	0.02	0.11
	7	35	40	0.01	0.13	40	0.11	0.47	40	0.02	0.10
	8	40	40	0.12	2.03	40	0.18	0.71	40	0.04	0.35
	9	45	40	0.16	1.68	40	0.48	4.97	40	0.04	0.32
	10	50	40	83.00	2859.04	40	0.59	2.97	40	0.10	0.96
8	6	36	40	0.89	15.05	40	0.88	8.67	40	0.24	1.45
	10	60	31	906.56	3600.00	40	15.92	233.66	40	0.80	6.32
10	6	60	12	2748.22	3600.00	20	2049.60	3600.00	35	937.06	3600.00
	10	100	0	3600.00	3600.00	2	3488.05	3600.00	29	1412.74	3600.00
total			763			782			824		

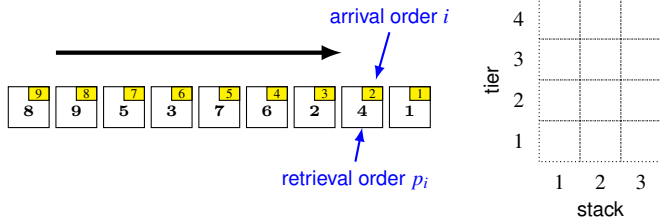
S: number of stacks, *H*: stack capacity, *N*: number of items

IP-based: 27 hours to solve the hardest instance with *N* = 100

Table of Contents

- 1 Introduction
- 2 Related Problems
- 3 Block Relocation Problem
- 4 Parallel Stack Loading Problem**
- 5 Summary

Parallel Stack Loading Problem (PSLP)

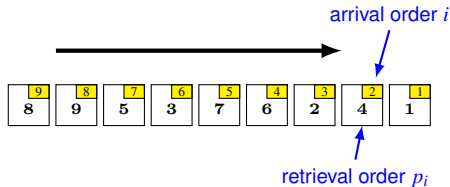


Parallel stack loading problem

Load incoming items (blocks) into stacks according to the arrival order

- number of stacks: S
- stack capacity: H
- Items are retrieved one by one after loading all items.
- Each item i is given a retrieval order p_i
- Objective: find an optimal loading plan minimizing **retrieval efforts**

Parallel Stack Loading Problem (PSLP)



4			9
3	3		8
2	2	6	7
1	1	4	5
	1	2	3
	stack		

Parallel stack loading problem

Load incoming items (blocks) into stacks according to the arrival order

- number of stacks: S
- stack capacity: H
- Items are retrieved one by one after loading all items.
- Each item i is given a retrieval order p_i
- Objective: find an optimal loading plan minimizing **retrieval efforts**

Retrieval Efforts (Objective Function)

4			9
3	3		8
2	2	6	7
1	1	4	5
	1	2	3

Exact objective function

- Minimum number of relocations for retrieving items
 - We must solve the block relocation problem for every loading plan
 $\Rightarrow$ **difficult to compute**

Surrogate objective functions

- Number of unordered stackings of adjacent items [Boysen and Emde, 2016]
- Number of badly-placed items [Boge and Knust, 2020]
- Number of double relocations [ElWakil et al., 2022]
- Stacking quality [ElWakil et al. 2022]

Retrieval Efforts (Objective Function)

Introduction

Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Parallel Stack Loading
Problem (PSLP)

Retrieval Efforts

NP-hardness of PSLP

Polynomial solvability of
PSLP_∞

MKIS and MES

Proof Outline

PSLPR and PSLPLR

Step 1

Step 2

Step 3

Summary of the Proof

Exact Approaches to
PSLP

IP Formulation

IP formulation

Valid Inequalities

Benders Decomposition

Master Problem

Benders Master
Problem

Benders Master
Problem (cnt'd)

Benders Subproblem

Computational Results

Results ($S = 5, 10$)

Results ($S = 15, 20$)

Summary

4			9
3	3		8
2	2	6	7
1	1	4	5
	1	2	3

Exact objective function

- Minimum number of relocations for retrieving items
 - We must solve the block relocation problem for every loading plan
⇒ **difficult to compute**

Surrogate objective functions

- Number of unordered stackings of adjacent items [Boysen and Emde, 2016]
- **Number of badly-placed items [Boge and Knust, 2020]**
- Number of double relocations [ElWakil et al., 2022]
- Stacking quality [ElWakil et al. 2022]

Retrieval Efforts (Objective Function)

4				9
3	3			8
2	2		6	7
1	4	3		5
	1	6		7
	1	2	3	

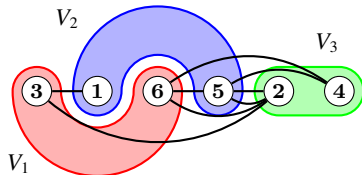
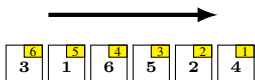
Exact objective function

- Minimum number of relocations for retrieving items
 - We must solve the block relocation problem for every loading plan
 ⇒ **difficult to compute**

Surrogate objective functions

- Number of unordered stackings of adjacent items [Boysen and Emde, 2016]
- **Number of badly-placed items [Boge and Knust, 2020]**
- Number of double relocations [ElWakil et al., 2022]
- Stacking quality [ElWakil et al. 2022]

NP-hardness of PSLP



NP-hardness of PSLP when $N > S$ and $6 \leq H < \infty$ [Boysen and Knust, 2020]

Reduction from MES on permutation graphs

MES on permutation graphs

Given a permutation graph $G = (V, E)$ and positive integers m and t , is there a partition of V into t independent sets $V_1, V_2, \dots, V_t$ with $|V_i| \leq m$?

Polynomial solvability of PSLP- ∞

PSLP- ∞

PSLP under unlimited stack capacity

Polynomial solvability of PSLP- ∞ [Tanaka et al., 2026]

Reduction to the maximum cardinality K -independent set problem for a permutation graph (MKIS)

MKIS for a permutation graph

Given a permutation graph $G = (V, E)$ and a positive integer K , find disjoint and independent subsets $V_1, V_2, \dots, V_K \subset V$ that maximize the total cardinality

$$|V_1 \cup V_2 \cup \dots \cup V_K| = |V_1| + |V_2| + \dots + |V_K|$$

MES for a permutation graph

Given a permutation graph $G = (V, E)$ and positive integers m and t , is there a partition of V into t independent subsets $V_1, V_2, \dots, V_t$ with $|V_i| \leq m$?

Differences?

MKIS and MES

MKIS for a permutation graph

Given a permutation graph $G = (V, E)$ and a positive integer K , find disjoint and independent subsets $V_1, V_2, \dots, V_K \subset V$ that maximize the total cardinality

$$|V_1 \cup V_2 \cup \dots \cup V_K| = |V_1| + |V_2| + \dots + |V_K|$$

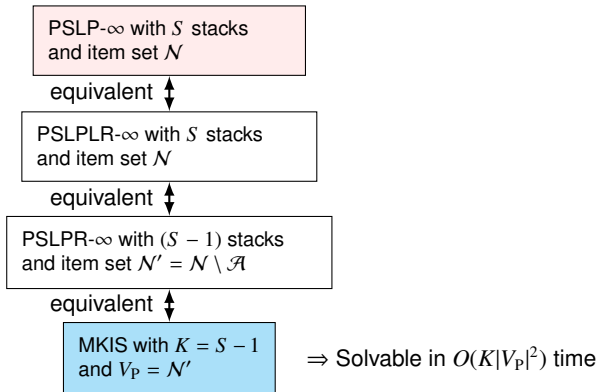
MES for a permutation graph

Given a permutation graph $G = (V, E)$ and positive integers m and t , is there a partition of V into t independent subsets $V_1, V_2, \dots, V_t$ with $|V_i| \leq m$?

Differences

- independent subsets
 - MKIS: $V_1 \cup V_2 \cup \dots \cup V_K \subset V$
 - MES: $V_1 \cup V_2 \cup \dots \cup V_t = V$
- objective
 - MKIS: total cardinality $|V_1| + |V_2| + \dots + |V_K|$
 - MES: individual cardinality $|V_i| \leq m$ (decision problem)
- complexity
 - MKIS: **polynomially solvable**
 - MES: **NP-complete**

Proof Outline for PSLP- ∞



$$\mathcal{N} = \{1, 2, \dots, N\}, \quad \mathcal{A} = \left\{ i \in \mathcal{N} \mid p_i = \min_{1 \leq j \leq i} p_j \right\} \quad (\text{set of always well-placed items})$$

PSLPR: PSLP with Rejection, PSLPLR: PSLP with Limited Rejection

PSLPR and PSLPLR

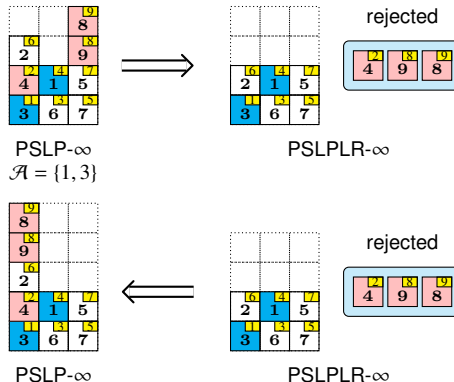
PSLPR: PSLP with Rejection

- Some items can be rejected
- Accepted items must be well-placed
- Objective: the number of accepted items (maximization)

PSLPLR: PSLP with Limited Rejection

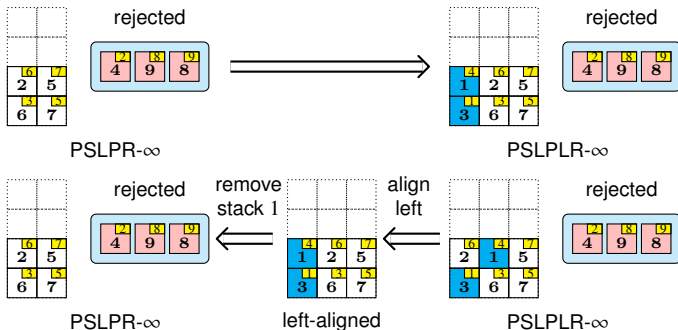
- Items that always become well-placed must be accepted
- $\mathcal{A}$: set of always well-placed items $\mathcal{A} = \left\{ i \in \mathcal{N} \mid p_i = \min_{1 \leq j \leq i} p_j \right\}$
- Objective: same as PSLPR

Step 1: PSLP- $\infty \Leftrightarrow$ PSLPLR- ∞



- PSLP- $\infty \Rightarrow$ PSLPLR- ∞
 - Reject badly-placed items
 - Items in $\mathcal{A}$ are always well-placed by its definition, thus never rejected
- PSLP- $\infty \Leftarrow$ PSLPLR- ∞
 - Rejected items are loaded into the same stacks as the items in $\mathcal{A}$

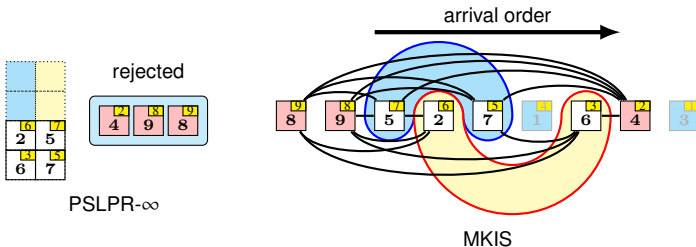
Step 2: PSLPLR $_{\infty} \Leftrightarrow$ PSLPR $_{\infty}$



- PSLPR $_{\infty} \Rightarrow$ PSLPLR $_{\infty}$
 - Load items in $\mathcal{A}$ into an additional stack
- PSLPLR $_{\infty} \Leftarrow$ PSLPLR $_{\infty}$
 - Load each well-placed item into the left-most stack
 - Remove stack 1 together with the items

PSLPR $_{\infty}$: for $S - 1$ stacks and items in $\mathcal{N} \setminus \mathcal{A}$

Step 3: PSLPR- $\infty \Leftrightarrow$ MKIS

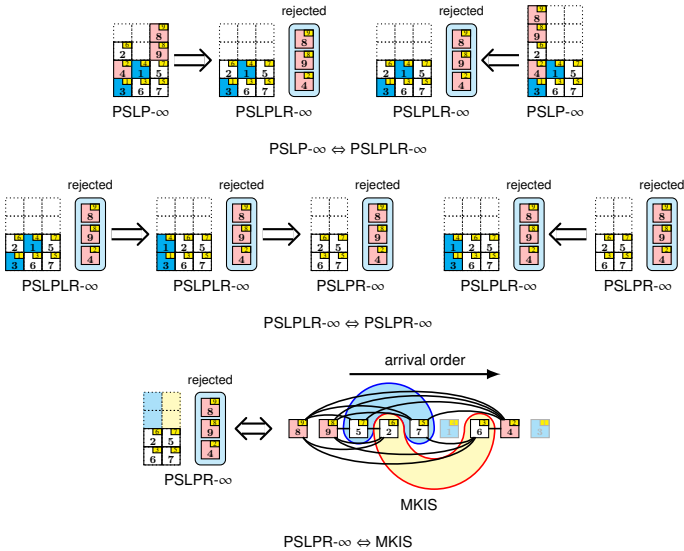


PSLPR- $\infty \Leftrightarrow$ MKIS

- item $\Leftrightarrow$ node
- $(S - 1)$ independent subsets $\Leftrightarrow (S - 1)$ stacks

MKIS for $S - 1$ stacks and items in $\mathcal{N} \setminus \mathcal{A}$: solvable in $O(SN^2)$

Summary of the Proof



Exact Approaches to PSLP

Exact approaches to PSLP

- IP formulations [Lu et al., 2016][Boge and Knust, 2020]
- IP formulation by Boge and Knust + valid inequalities [Tanaka et al. 2026]
- Combinatorial Benders decomposition [Tanaka et al. 2026]

IP formulation [Boge and Knust, 2020] can handle instances with up to 100 or 150 items

Exact Approaches to PSLP

Exact approaches to PSLP

- **IP formulations** [Lu et al., 2016][Boge and Knust, 2020]
- IP formulation by Boge and Knust + **valid inequalities** [Tanaka et al. 2026]
- **Combinatorial Benders decomposition** [Tanaka et al. 2026]

IP formulation [Boge and Knust, 2020] can handle instances with up to 100 or 150 items

IP Formulation [Boge and Knust, 2020]

Notation

- Set of items: $\mathcal{N} = \{1, 2, \dots, N\}$
- Set of stacks: $\mathcal{S} = \{1, 2, \dots, S\}$
- Set of item pairs such that item j becomes badly-placed if loaded into the same stack as item i : $\mathcal{P} = \{(i, j) \mid i < j \wedge p_i < p_j\}$
 eg. $p_2 = 4$ and $p_5 = 6 \Rightarrow (2, 5) \in \mathcal{P}$

Decision variables b_{is} and w_{is} ($i \in \mathcal{N}, s \in \mathcal{S}$)

b_{is} : item i is badly-placed in stack s

w_{is} : item i is well-placed in stack s

IP Formulation (IP)

$$\begin{aligned}
 \min \quad & \sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{S}} b_{is}, && \text{(number of badly-placed items)} \\
 \text{s.t.} \quad & \sum_{s \in \mathcal{S}} (w_{is} + b_{is}) = 1, \quad i \in \mathcal{N}, && \text{(item assignment)} \\
 & \sum_{i \in \mathcal{N}} (w_{is} + b_{is}) \leq H, \quad s \in \mathcal{S}, && \text{(stack capacity)} \\
 & w_{is} + b_{is} + w_{js} \leq 1, \quad (i, j) \in \mathcal{P}, \quad s \in \mathcal{S}, && \text{(items conflict)} \\
 & w_{is} \in \{0, 1\}, \quad i \in \mathcal{N}, \quad s \in \mathcal{S}, \\
 & b_{is} \in \{0, 1\}, \quad i \in \mathcal{N}, \quad s \in \mathcal{S}.
 \end{aligned}$$

If item i is loaded into stack s , item j cannot be well-placed there as $p_i < p_j$.

Valid Inequalities

Basic idea

$$p_3 = 2, p_4 = 5, p_7 = 6, p_9 = 8$$

⇒ only item 3 is well-placed in the same stack

⇒ only n_s items can be well-placed in n_s stacks

Valid inequalities (clique inequalities)

- $\mathcal{L} = \{i_1, i_2, \dots, i_k\}$ satisfying $p_{i_1} < p_{i_2} < \dots < p_{i_k}$
- At most S items in $\mathcal{L}$ of $\mathcal{N}$ can be well-placed.

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{L}} w_{is} \leq S$$

Separation algorithm

- Find inequalities violated by the current LP solution
- Maximization of the righthand side ⇒ **dynamic programming**

Benders Decomposition Algorithm

Benders decomposition algorithm

- Exact approach to hard mathematical programming problems
- Decompose the original problem into master problem and subproblem
- Solve them by turns while adding Benders cuts to the master problem

Combinatorial Benders decomposition

Combinatorial Benders cuts to forbid the current optimal solution

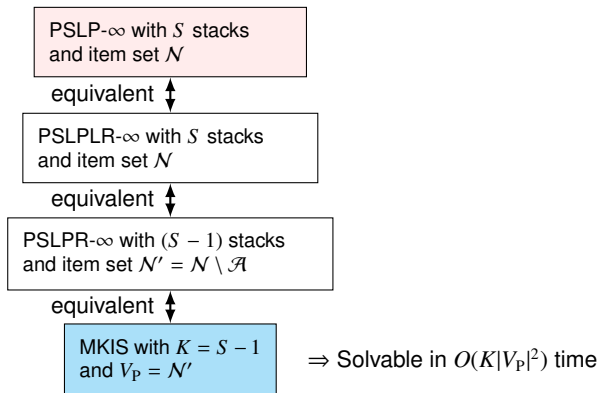
Combinatorial Benders decomposition approach to PSLP [Tanaka et al., 2026]

Master problem PSLP- ∞

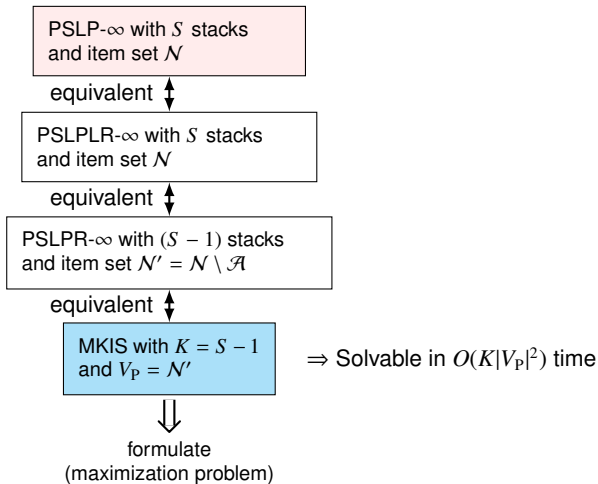
Subproblem Check if the solution is feasible for PSLP

If infeasible, the solution of PSLP- ∞ is forbidden by a Benders cut.

Formulation of Benders Master Problem



Formulation of Benders Master Problem



Benders Master Problem (MP)

Decision variables x_e ($e \in Q$) and y_i ($i \in N'$)

x_e : flow through arc $e = (i, j)$ (binary) y_i : flow through node i (continuous)

Benders master problem (MP)

$$\begin{aligned}
 \max \quad & \sum_{i \in N'} y_i, && \text{(total cardinality of independent sets)} \\
 \text{s.t.} \quad & \sum_{i \in N'} x_{(0,i)} \leq S - 1, && \text{(number of independent sets)} \\
 & \sum_{(j,i) \in Q} x_{(j,i)} = y_i, \quad i \in N', && \text{(flow conservation)} \\
 & \sum_{(i,j) \in Q} x_{(i,j)} = y_i, \quad i \in N', && \text{(flow conservation)} \\
 & x_e \in \{0, 1\}, \quad e \in Q, \\
 & 0 \leq y_i \leq 1, \quad i \in N'.
 \end{aligned}$$

- Set of considered items: $N' = N \setminus \mathcal{A}$ (+ dummy items 0 and $N + 1$)
- Set of item pairs (i, j) such that item j is well-placed even in the same stack as item i ($i < j$):
 $Q = \{(i, j) \in (N' \cup \{0\}) \times (N' \cup \{N + 1\}) \mid i < j \wedge p_i > p_j\}$

Benders Master Problem (MP) (cnt'd)

Benders master problem (MP)

$$\max \sum_{i \in \mathcal{N}'} y_i, \quad (\text{total cardinality of independent sets})$$

$$\text{s.t. } \sum_{i \in \mathcal{N}'} x_{(0,i)} \leq S - 1, \quad (\text{number of independent sets})$$

$$\sum_{(j,i) \in Q} x_{(j,i)} = y_i, \quad i \in \mathcal{N}', \quad (\text{flow conservation})$$

$$\sum_{(i,j) \in Q} x_{(i,j)} = y_i, \quad i \in \mathcal{N}', \quad (\text{flow conservation})$$

$$x_e \in \{0, 1\}, \quad e \in Q,$$

$$0 \leq y_i \leq 1, \quad i \in \mathcal{N}'.$$

Solution to (MP)

$$y_i^* = 1 \Leftrightarrow \text{node } i \text{ is chosen in an independent set in MKIS}$$

Solution to PSLP- ∞

$$\text{Set of well-placed items: } \mathcal{W} \cup \mathcal{A} \text{ where } \mathcal{W} = \{i \in \mathcal{N}' = \mathcal{N} \setminus \mathcal{A} \mid y_i^* = 1\}$$

Check if items in $\mathcal{W} \cup \mathcal{A}$ can be well-placed in PSLP by subproblem

Benders Subproblem (SP)

Decision variables z_{is} ($i \in \mathcal{N}$, $s \in \mathcal{S}$)

z_{is} : item i is loaded into stack s

Benders subproblem (SP)

min v , (dummy objective function)

s.t. $\sum_{s \in \mathcal{S}} z_{is} = 1$, $i \in \mathcal{N}$, (item assignment)

$\sum_{i \in \mathcal{N}} z_{is} \leq H$, $s \in \mathcal{S}$, (stack capacity)

$z_{is} + z_{js} \leq 1$, $(i, j) \in \mathcal{P}$, $j \in \mathcal{W}$, $s \in \mathcal{S}$, (item $j \in \mathcal{W}$ is well-placed)

$z_{is} \in \{0, 1\}$, $i \in \mathcal{N}$, $s \in \mathcal{S}$.

Benders cut

- (SP) is feasible: Current solution is optimal. Terminate.
- (SP) is infeasible: Benders cut is added to (MP). Repeat.

$$\sum_{i \in \mathcal{W}} y_i^* \leq |\mathcal{W}| - 1. \quad (\text{forbid the current solution})$$

Computational Results

Comparison with the IP formulation and the Benders decomposition algorithm

Experimental setup

Instances: Randomly generated

Language: Python

IP Solver: Gurobi Optimizer 12.0.3

PC: desktop computer equipped with an Intel Core i9-9900K CPU and 64 GiB RAM

Time Limit: 1800 s for one instance

Results ($S = 5, 10$)

S	H	N	(IP)			b&c			Benders				
			#solved	time (s)		#solved	time (s)		#solved	time (s)			
				average	maximum		average	maximum		average	maximum		
5	15	10	10	0.00	0.01	10	0.00	0.01	10	0.00	0.01		
		20	10	0.02	0.04	10	0.01	0.02	10	0.00	0.01		
		25	10	0.07	0.15	10	0.07	0.19	10	0.07	0.47		
	10	30	10	0.07	0.14	10	0.03	0.05	10	0.01	0.01		
		40	10	0.31	1.38	10	0.09	0.14	10	0.01	0.01		
		50	10	0.53	0.99	10	0.40	1.13	10	0.23	2.09		
	15	45	10	0.29	0.72	10	0.12	0.18	10	0.01	0.01		
		60	10	0.66	0.99	10	0.42	0.72	10	0.02	0.03		
		75	10	1.86	4.85	10	1.52	3.23	9	180.02	1800.00		
	20	60	10	0.46	0.65	10	0.33	0.48	10	0.02	0.02		
		80	10	1.82	3.94	10	1.33	2.10	10	0.03	0.07		
100		10	3.61	7.44	10	3.44	4.47	10	9.49	94.40			
10	5	30	10	0.05	0.38	10	0.02	0.02	10	0.01	0.01		
		40	10	0.55	4.60	10	0.03	0.06	10	0.01	0.02		
		50	10	0.65	2.37	10	0.41	1.19	10	0.51	3.27		
	10	60	10	7.67	29.77	10	0.11	0.26	10	0.02	0.03		
		80	10	24.71	133.91	10	1.15	1.78	10	0.04	0.07		
		100	10	35.87	96.84	10	8.78	13.97	10	3.06	29.47		
	15	90	10	19.74	41.55	10	2.45	6.33	10	0.04	0.05		
		120	9	259.10	1800.00	10	9.14	16.11	10	0.14	0.49		
		150	6	797.78	1800.00	10	82.59	296.45	9	181.12	1800.00		
	20	120	9	262.50	1800.00	10	10.36	18.20	10	0.09	0.17		
		160	8	526.00	1800.00	10	66.54	154.59	10	0.15	0.26		
200		5	1297.49	1800.00	10	307.15	442.16	10	34.45	273.12			
total			227				240				236		

S : number of stacks
 H : stack capacity
 N : number of items

- Valid inequalities (b&c) enable solving all instances to optimality
- Benders decomposition can be ineffective for dense instances ($N = SH$).

Results ($S = 5, 10$)

S	H	N	(IP)			b&c			Benders			
			#solved	time (s)		#solved	time (s)		#solved	time (s)		
				average	maximum		average	maximum		average	maximum	
5	15	10	10	0.00	0.01	10	0.00	0.01	10	0.00	0.01	
		20	10	0.02	0.04	10	0.01	0.02	10	0.00	0.01	
		25	10	0.07	0.15	10	0.07	0.19	10	0.07	0.47	
	30	10	10	0.07	0.14	10	0.03	0.05	10	0.01	0.01	
		40	10	0.31	1.38	10	0.09	0.14	10	0.01	0.01	
		50	10	0.53	0.99	10	0.40	1.13	10	0.23	2.09	
	45	10	10	0.29	0.72	10	0.12	0.18	10	0.01	0.01	
		60	10	0.66	0.99	10	0.42	0.72	10	0.02	0.03	
		75	10	1.86	4.85	10	1.52	3.23	9	180.02	1800.00	
	60	10	10	0.46	0.65	10	0.33	0.48	10	0.02	0.02	
20		80	10	1.82	3.94	10	1.33	2.10	10	0.03	0.07	
		100	10	3.61	7.44	10	3.44	4.47	10	9.49	94.40	
	10	30	10	10	0.05	0.38	10	0.02	0.02	10	0.01	0.01
40			10	0.55	4.60	10	0.03	0.06	10	0.01	0.02	
50			10	0.65	2.37	10	0.41	1.19	10	0.51	3.27	
60		10	10	7.67	29.77	10	0.11	0.26	10	0.02	0.03	
		80	10	24.71	133.91	10	1.15	1.78	10	0.04	0.07	
		100	10	35.87	96.84	10	8.78	13.97	10	3.06	29.47	
90		10	19.74	41.55	10	2.45	6.33	10	0.04	0.05		
		15	120	9	259.10	1800.00	10	9.14	16.11	10	0.14	0.49
			150	6	797.78	1800.00	10	82.59	296.45	9	181.12	1800.00
120			9	262.50	1800.00	10	10.36	18.20	10	0.09	0.17	
20	160	8	526.00	1800.00	10	66.54	154.59	10	0.15	0.26		
	200	5	1297.49	1800.00	10	307.15	442.16	10	34.45	273.12		
total			227			240			236			

S : number of stacks
 H : stack capacity
 N : number of items

- Valid inequalities (b&c) enable solving all instances to optimality
- Benders decomposition can be ineffective for dense instances ($N = SH$).

Results ($S = 15, 20$)

Introduction
Related Problems

Block Relocation
Problem

Parallel Stack
Loading Problem

Parallel Stack Loading
Problem (PSLP)

Retrieval Efforts

NP-hardness of PSLP

Polynomial solvability of
PSLP $_{\infty}$

MKIS and MES

Proof Outline

PSLPR and PSLPLR

Step 1

Step 2

Step 3

Summary of the Proof

Exact Approaches to
PSLP

IP Formulation

IP formulation

Valid Inequalities

Benders Decomposition

Master Problem

Benders Master
Problem

Benders Master
Problem (cnt'd)

Benders Subproblem

Computational Results

Results ($S = 5, 10$)

Results ($S = 15, 20$)

Summary

			(IP)			b&c			Benders		
			time (s)			time (s)			time (s)		
<i>S</i>	<i>H</i>	<i>N</i>	#solved	average	maximum	#solved	average	maximum	#solved	average	maximum
15	5	45	10	0.04	0.05	10	0.04	0.05	10	0.02	0.03
		60	10	0.08	0.25	10	0.07	0.13	10	0.03	0.04
		75	10	107.12	1014.91	10	5.12	17.52	10	1.08	2.80
	10	90	7	562.46	1800.00	10	0.36	0.97	10	0.04	0.05
		120	6	754.25	1800.00	10	3.48	8.79	10	0.09	0.14
		150	8	543.14	1800.00	9	323.65	1800.00	8	364.23	1800.00
	15	135	9	332.70	1800.00	10	12.02	41.34	10	0.13	0.35
		180	9	474.01	1800.00	10	73.72	183.97	10	0.23	0.47
		225	4	1682.34	1800.00	6	1123.59	1800.00	8	371.81	1800.00
	20	180	9	507.11	1800.00	10	158.12	580.02	10	0.18	0.20
240		4	1635.99	1800.00	10	1175.18	1769.08	10	1.46	8.90	
300		0	1800.00	1800.00	1	1767.32	1800.00	8	368.19	1800.00	
20	5	60	10	0.07	0.08	10	0.07	0.08	10	0.04	0.05
		80	10	0.12	0.15	10	0.12	0.15	10	0.05	0.07
		100	9	180.78	1800.00	10	2.52	13.58	10	4.54	44.00
	10	120	10	0.36	1.30	10	0.33	0.95	10	0.08	0.09
		160	2	1440.38	1800.00	10	8.80	24.11	10	0.14	0.20
		200	1	1668.31	1800.00	3	1612.79	1800.00	5	1167.35	1800.00
	15	180	2	1468.74	1800.00	10	34.72	176.57	10	0.18	0.26
		240	2	1683.00	1800.00	7	874.98	1800.00	10	0.80	2.20
		300	0	1800.00	1800.00	0	1800.00	1800.00	7	556.43	1800.00
	20	240	0	1800.00	1800.00	8	995.24	1800.00	10	0.50	0.94
320		0	1800.00	1800.00	2	1670.03	1800.00	10	5.87	23.38	
400		0	1800.00	1800.00	0	1800.00	1800.00	8	445.39	1800.00	
total			132			186			224		

S : number of stacks
 H : stack capacity
 N : number of items

- Benders decomposition is effective for larger instances.

Table of Contents

- 1 Introduction
- 2 Related Problems
- 3 Block Relocation Problem
- 4 Parallel Stack Loading Problem
- 5 Summary**

Summary

- Many problems on stacking and retrieval
block (container) relocation problem, container pre-marshalling problem, parallel stack loading problem, slab reshuffling problem Blocks World, Tower of Hanoi, train shunting problem, stack sorting, etc.
- Studies on the BRP
 - Computational complexities
 - IP-based algorithm for the restricted BRP
- Studies on the PSLP
 - Computational complexities
 - Combinatorial Benders decomposition algorithm for the PSLP

Resources

- Parallel stack loading problem
<https://sites.google.com/site/shunjitanaka/pslp>
- Block (container) pre-marshalling problem
<https://sites.google.com/site/shunjitanaka/pmp>
- Block (container) relocation problem
<https://sites.google.com/site/shunjitanaka/brp>
- Block relocation problem with a stowage plan
<https://sites.google.com/site/shunjitanaka/brpsp>