Online Storage Systems and Transportation Problems with Applications

Applied Optimization
Volume 91

Series Editors:

Panos M. Pardalos
University of Florida, U.S.A.

Donald W. Hearn
University of Florida, U.S.A.

Online Storage Systems and Transportation Problems with Applications

Optimization Models and Mathematical Solutions

by

Julia Kallrath
ITWM, Germany

Springer

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN 1-4020-2971-3 e-ISBN 0-387-23485-3 Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11161943

springeronline.com

to my parents

Contents

Dedication v

Preface xi

1. INTRODUCTION 1

 1.1 Optimization Everywhere 1

2. BATCH PRESORTING PROBLEMS. I 5

 2.1 Problem Description and Classification 5

 2.2 Formulation of the Batch Presorting Problem 6

  2.2.1 Feasible Permutations 7

  2.2.2 Mathematical Formulation of BPSP_1 8

  2.2.3 Mathematical Formulation of BPSP_2 and BPSP_3 10

  2.2.3.1 An Optimization Version of BPSP_2 11

  2.2.3.2 An Optimization Version of BPSP_3 13

 2.3 Complexity Results 15

 2.4 Polynomial Subcases 21

  2.4.1 Reformulation of BPSP_2 and BPSP_3 21

  2.4.2 An Alternative Model Formulation of D-BPSP$'$ 22

 2.5 The Case of Two Layers 25

  2.5.1 Offline Situations 26

  2.5.2 Online Situations 30

  2.5.3 Algorithms for Online Situations with Lookahead 33

  2.5.3.1 Definition of a Lookahead 33

  2.5.3.2 An Algorithm with a Weak Lookahead, size $\lambda = 1$ 34

  2.5.3.3 Algorithms with a Weak Lookahead, size $\lambda \geq 2$ 35

  2.5.3.4 An Algorithm with a Strong Lookahead 37

  2.5.4 Competitive Analysis 37

2.5.5 Comparison of D_2 and Online Algorithms with Lookahead 41

2.6 Extensions 41

2.7 Summary und Future Research 43

3. BATCH PRESORTING PROBLEMS. II 45

3.1 The Storage System Rotastore 45

3.1.1 A Brief Description of the Rotastore 45

3.1.2 Stochastic Measures 46

3.2 Numerical Tests 48

3.2.1 Models 48

3.2.2 Algorithms 53

3.3 Summary 54

4. VEHICLE ROUTING PROBLEMS IN HOSPITALS. I 57

4.1 Problem Formulation and Solution Outline 57

4.1.1 Problem Description 57

4.1.2 Discussion of one Particular Model Formulation 60

4.1.3 Discussion of Existing Solution Approaches 63

4.1.4 Outline of Proposed Methods 64

4.2 General Framework 67

4.2.1 Notation 67

4.2.2 Characterizing the Quality of Tours 69

4.3 Exact Solution Approaches 70

4.3.1 A Mixed Integer Programming Approach 70

4.3.1.1 Data, Indices and Variables 71

4.3.1.2 The Objective Function 72

4.3.1.3 The Constraints 73

4.3.1.4 Equivalent Mixed Integer Linear Formulations 79

4.3.1.5 Online Version of the MILP Model 80

4.3.1.6 Comments on the Size and the Structure of the MILP Problem 82

4.3.1.7 Tightening the Model Formulation 84

4.3.1.8 Concluding Remarks on the Model 85

4.3.2 A Branch-and-Bound Approach for Solving the Intra-Tour Problem 85

4.3.3 Column Enumeration 92

4.3.3.1 Motivating a Column Enumeration Approach 92

4.3.3.2 Comments on Column Generation Techniques 94

4.3.3.3 Developing a Column Enumeration Approach 95
4.3.3.4 Summarizing the Column Enumeration Approach 99
4.3.3.5 Complexity Aspects and Computational Issues 100
4.3.3.6 Numerical Improvements 103
4.3.3.7 Column Enumeration Coupled to Heuristics 104
4.4 Construction and Improvement Heuristics 104
4.4.1 Construction Heuristics 104
4.4.1.1 Heuristic `CONH1` 105
4.4.1.2 Heuristic `CONH2` 106
4.4.1.3 Penalty Criteria 109
4.4.2 Improvement Heuristics 110
4.4.2.1 Heuristic `IT1` 110
4.4.2.2 Heuristic `IT2` 111
4.4.2.3 Adding an Order to a Tour 111
4.4.2.4 Termination Criteria 112
4.4.3 Simulated Annealing and Vehicle Routing 114
4.4.3.1 Monte Carlo Methods and Simulated Annealing 115
4.4.3.2 Optimization Problems and Simulated Annealing 116
4.4.3.3 Improving Tours by Simulated Annealing 117
4.5 Summary 120
5. VEHICLE ROUTING PROBLEMS IN HOSPITALS. II 121
5.1 OptiTrans 121
5.2 Numerical Tests 125
5.2.1 Input data 125
5.2.2 Offline and Online Versions 125
5.2.3 MILP Model 126
5.2.3.1 Intra-Tour Optimization (Single-Vehicle Cases) 126
5.2.3.2 An Online Case Including Pre-assigned Orders 129
5.2.4 Column Enumeration Experiments 134
5.2.5 Construction and Improvement Heuristics 139
5.2.5.1 Offline Version 139
5.2.5.2 Online version 141
5.2.6 Simulated Annealing Examples 143
5.2.7 Summary of the Numerical Experiments 144
5.3 Summary 146

6. SUMMARY 149

Appendices 154

A Rotastore 155

A.1 Tabular Results for Different Models 155

A.2 Tabular Results for Different Algorithms 180

B OptiTrans 187

B.1 Input Data 187

B.1.1 Input Data Common to all Solution Approaches 187

B.1.2 Specific Input Data for the MILP Model and the Column Enumeration Approach 188

B.1.3 Specific Input Data for the Heuristic Methods 189

B.1.3.1 Penalty Criteria 189

B.1.3.2 Control Parameters of the OptiTrans Software 190

B.2 Tabular Results 191

B.2.1 Tabular Results for the MILP Model 191

B.2.2 Tabular Results for the Heuristic Methods 195

B.2.2.1 Input Data for a Whole Day - Offline Analysis 195

B.2.2.2 Results for CIH and SA 200

References 213

Index 221

Preface

This book covers the analysis and development of online algorithms involving exact optimization and heuristic techniques, and their application to solve two real life problems.

The first problem is concerned with a complex technical system: a special carousel based high-speed storage system - Rotastore. It is shown that this logistic problem leads to an NP-hard *Batch PreSorting Problem* (BPSP) which is not easy to solve optimally in offline situations. We consider a polynomial case and develope an exact algorithm for offline situations. Competitive analysis showed that the proposed online algorithm is 3/2-competitive. Online algorithms with lookahead improve the online solutions in particular cases. If the capacity constraint on additional storage is neglected the problem has a totally unimodular polyhedron.

The second problem originates in the health sector and leads to a vehicle routing problem. We demonstrate that reasonable solutions for the offline case covering a whole day with a few hundred orders can be constructed with a heuristic approach, as well as by simulated annealing. Optimal solutions for typical online instances are computed by an efficient column enumeration approach leading to a set partitioning problem and a set of routing-scheduling subproblems. The latter are solved exactly with a branch-and-bound method which prunes nodes if they are value-dominated by previous found solutions or if they are infeasible with respect to the capacity or temporal constraints. Our branch-and-bound method is suitable to solve any kind of sequencing-scheduling problem involving accumulative objective functions and constraints, which can be evaluated sequentially. The column enumeration approach developed to solve this hospital problem is of general nature and thus can be embedded into any decision-support system involving assigning, sequencing and scheduling.

The book is aimed at practioners and scientists in operation research especially those interested in online optimization. The target audience are readers interested in fast solutions of batch presorting and vehicle routing problems or software companies producing decision support systems. Students and graduates in mathematics, physics, operations research, and businesses with interest in modeling and solving real optimization problems will also benefit from this book and can experience how online optimization enters into real world problems.

Structure of this Book

This book is organized as follows. Chapter 2 addresses the BPSP, where a formal definition of the BPSP is introduced (Section 2.1) and several modeling approaches are proposed (see Section 2.2). Complexity issues of some formulations are investigated in Section 2.3 and Section 2.4. For one polynomial case of the BPSP several algorithms are presented and compared in Section 2.5. In Chapter 3 we consider a concrete application of the BPSP - carousel based storage system Rotastore. In Section 3.1 we describe the system performance, and in Section 3.2 the numerical results of the experiments are presented.

Chapter 4 focuses on the *Vehicle Routing problem with Pickup and Delivery and Time Windows* (VRPPDTW), adapted for hospital transportation problems. After introducing some notations (Subsection 4.2.1), we suggest several approaches we have developed to solve this problem, including a MILP formulation (Subsection 4.3.1), a branch-and-bound approach (Subsection 4.3.2), a column enumeration approach (Subsection 4.3.3), and heuristic methods (Section 4.4). In Chapter 5 we describe a problem related to a hospital project with the University Hospital in Homburg. Detailed numerical results for our solution approaches related to the VRPPDTW are collected in Section 5.2.

Conventions and Abbreviations

The following table contains in alphabetic order the abbreviations used in this book.

Abbreviation	*Meaning*
B&B	Branch-and-Bound
B&C	Branch-and-Cut
BPSP	Batch PreSorting Problem
CEA	column enumeration approach
IP	Integer Programming
LP	Linear Programming
MCP	Mixed Complementarity Problem
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Nonlinear Programming
RH	reassignment heuristic
SA	simulated annealing
SAT	satisfiability problem
SH	sequencing heuristic
s.t.	subject to
TS	tabu search
VNS	variable neighborhood search
VRP	Vehicle Routing Problem
VRPPDTW	VRP with Pickup and Delivery and Time Windows
w.r.t.	with respect to

Acknowledgements

First of all I want to thank Prof. Dr. Stefan Nickel (ITWM Kaiserslautern and Universität Saarbrücken), the head of our department, who was leading the two real world projects; without him the book would not have appeared. Prof. Dr. Christodoulos Floudas (Princeton University) and Prof. Dr. Linus Schrage (University of Chicago) - for their interest in this work and encouraging comments on the vehicle routing problem. Prof. Dr. Susanne Albers (Freiburg Universität) - for the useful feedback on the topic of online optimization and the competitive analysis for the batch pre-sorting problem. Prof. Dr. Alexander Lavrov (ITWM Kaiserslautern and NTUU Kiev) - for his constant support and help during all my time at the ITWM in Kaiserslautern. Dr. Teresa Melo (ITWM Kaiserslautern)- for fruitful discussions about the vehicle routing problem and for the proof reading. Martin Müller (Siemens AG, München) - for constructive talks and discussions about the batch presorting problem. Prof. Dr. Robert E. Wilson (University of Florida), Steffen Rebennack (Universität Heidelberg) and Dr. Anna Schreieck (BASF, Ludwigshafen) - for careful reading parts of this book. And last but not least, my husband Josef Kallrath - for his positive spirit encouraging and supporting me never to give up.

JULIA KALLRATH

Chapter 1

INTRODUCTION

What do a logistics manager responsible for an inventory storage system and a vehicle fleet dispatcher in a hospital campus have in common? They both have to consider new objects arriving at short notice and to decide on what to do with them, how to assign them to given resources or how to modify previously made decisions. This means they both need to make decisions based on data suffering from incomplete knowledge about near future events. Online optimization is a discipline in mathematical optimization and operations research which provides the mathematical framework and algorithms for dealing appropriately with such situations.

1.1. Optimization Everywhere

The need for applying optimization arises in many areas: finance, space industry, biosystems, textile industry, mineral oil, process and metal industry, and airlines to name a few. Mathematical programming is a very natural and powerful way to solve problems appearing in these areas. In particular, see [12], [18], [23], [37] and [83] for application examples. One might argue that low structure systems can probably be handled well without optimization. However, for the analysis and development of real life complex systems (that have many degrees of freedom, underlying numerous restrictions *etc.*) the application of optimization techniques is unavoidable. It would not be an exaggeration even to say that any decision problem is an optimization problem. Despite their diversity real world optimization problems often share many common features, *e.g.*, they have similar mathematical kernels such as flow, assignment or knapsack structures.

One further common feature of many real life decision problems is the online nature aspect, *i.e.*, decision making is based on partial, insufficient information or without any knowledge of the future. One approach (not treated in this book) to solve problems with only partial or insufficient information is optimization under uncertainty (*cf.* [45], [50], or [88]), and especially, stochastic programming (*cf.* [14], [53], [77], or [78]). In that case, the problem is still solved as an offline problem.

However, it is not always appropriate to solve a problem offline. If we cannot make any assumptions on future data, only the currently available data can be used. In such situations online optimization is recommended. We can list a number of problems that were originally formulated as offline problems but which in many practical applications are used in their online versions: the bin packing problem, the list update problem, the k-server problem, the vehicle routing problem, and the pickup and delivery problem to name a few.

Special optimization techniques for online applications exploit the online nature of the decision process. Usually, a sequence of online optimization problems is solved when advancing in time and more data become available. Therefore, online optimization can be much faster than offline optimization (which uses the complete input data). To estimate the quality of a sequence of solutions obtained by online optimization one can only compare it with the overall solution produced by an offline algorithm afterwards. A powerful technique to estimate the performance of online algorithms is the competitive analysis (*cf.* [11]). A good survey on online optimization and competitive analysis can be found in [4], [11], [30]. Online optimization and competitive analysis are based on generic principles and can be beneficial in completely different areas such as the storage system and transportation problem considered in this book.

At first we consider an example of a complex technical system, namely a special carousel based high-speed storage system - *Rotastore* [73], which not only allows storing ([56], [57]) but also performs sorting ([49], [70]). Sorting actions and assignment to storing locations are fulfilled in real time, but the information horizon may be rather narrow. The quality of the corresponding decisions strongly influences the performance of the system in general; thus the need to improve the quality of the decisions. Due to the limited information horizon online optimization is a promising approach to solve these problems.

In our second case study, the conditions for the decision making process in hospital transportation are similar: the orders often are not known in advance, the transportation network may be changed dynamically. The efficiency of order assignment and scheduling of the transport system can influence the operation of the whole hospital. That assumes, in this case, not only economical aspects, but, at first of all, human health and life issues.

As will be shown in this book, the mathematical base for the first problem is the *Batch PreSorting Problem* (BPSP), for the second one we naturally can use an online variant of the *Vehicle Routing problem with Pickup and Delivery and Time Windows* (VRPPDTW). The efficient application of the corresponding solution methods allows to improve the performance of both systems compared to the current real life situation.

Chapter 2

BATCH PRESORTING PROBLEMS. I MODELS AND SOLUTION APPROACHES

This chapter is organized as follows: at first, we describe the problem and give a short classification. In Section 2.2 different formulations of the BPSP are presented. In Subsection 2.2.2 we consider an optimization version of $BPSP_1$. In Subsection 2.2.3 we formulate $BPSP_2$ and $BPSP_3$ as decision problems and additionally introduce optimization models. The complexity status of $BPSP_2$ is investigated in Section 2.3, and in Section 2.4 we show that there is a polynomial version of the BPSP. Also we consider a special subcase of a BPSP with $N^L = 2$ in offline and online situations and present corresponding algorithms in Section 2.5. Finally, in Section 2.6, some results derived for BPSPs with $N^L = 2$ are adapted to general BPSP.

2.1. Problem Description and Classification

We consider the problem of finding a finite sequence of objects of different types, that guarantees an optimal assignment of objects to given physical storage layers with a pre-sorting facility of limited capacity. This problem will be called the *Batch PreSorting Problem* (BPSP), because the objects have to be sorted within one batch before they are assigned to the layers. After sorting, the object with number i will be assigned to layer i. For a more transparent presentation we speak of colors instead of types and thus consider all objects of type k as having the same color k. We present three types of BPSP with different objective functions. The objective function, z, of BPSP_1 minimizes the total number of layers not yet occupied by objects of a certain color k; such objects can be considered as occupying an empty layer (empty *w.r.t.* to k) at zero cost. Once each layer has an object of a given color, the cost does not change with further additions of that color. If the forgoing is true for all colors, z gives the number of all objects to be distributed minus those already assigned to the layers. In BPSP_2 the objective is to minimize the maximum number of objects of the same color on the same layer. Finally, BPSP_3 aims to minimize the sum of the maximum number of objects of the same color over all layers.

We use the following example to illustrate the problem:

EXAMPLE 2.1 *Suppose, there are six objects of two different colors in the input sequence (see Fig. 2.1.1) and three layers.*

Objects can be sorted within one batch, i.e., the objects 1, 2, 3 can be sorted, then they are assigned to the layers. After this the objects 4,5 and 6 can be sorted and assigned to the layers. Fig. 2.1.1 displays the content of the layers without pre-sorting. For this assignment the objective function value of $BPSP_1$ is 2*, because the objects of the first batch occupy the layers at zero cost (layers were empty); the objects 4 and 5 occupy the layers 1 and 2, respectively, each with cost one, and object 6 occupies layer 3 at zero cost. The objective function value of $BPSP_2$ is* 2*, because the maximal number of objects of any color on all layers is* 2*. Finally, the objective function value of $BPSP_3$ is* 4*, because the maximal number of objects of the colors 1 and 2 over all layers is* 2 *for both colors. Clearly, this assignment is not optimal w.r.t. none of the three objective functions. The optimal objective function values for $BPSP_1$, $BPSP_2$, and $BPSP_3$ are* 0, 1, *and* 2, *respectively (see Fig. 2.1.2).*

2.2. Formulation of the Batch Presorting Problem

At first we introduce some notations used in this chapter:

- N^O is the number of objects of different colors in a given sequence. These objects are indexed by i or j (for simplicity the positions of the objects are identified by their index values). S_k is the set of objects of color k, and $i \in S_k$ means that the object at position i in the sequence (also called "i^{th} object" or "object i" for short) has color k;
- N^K is the number of colors;
- N^L is the number of layers;
- N^S is the capacity of the pre-sorting facility.

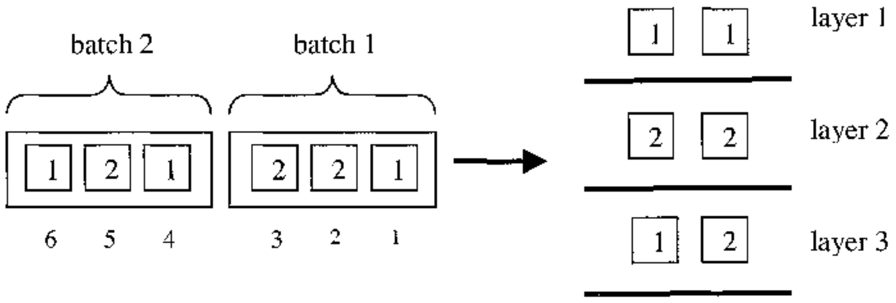

Figure 2.1.1. The input sequence and the content of the storage layers without presorting. On the left part of the figure, the numbers 1, 2, ..., 6 refer to the objects while the numbers 1 and 2 in the squares denote the colors.

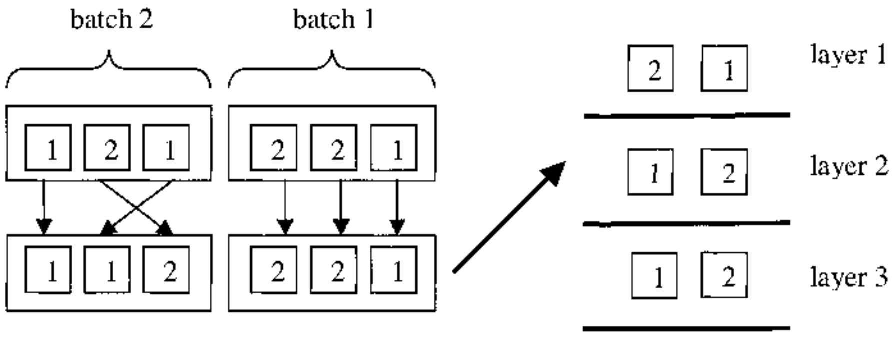

Figure 2.1.2. Optimal permutation and content of the storage layers after the assignment

2.2.1 Feasible Permutations

Before we talk about feasible permutations, recall the definition of permutation:

DEFINITION 2.2 *A permutation δ on a set of N^O objects is a one-to-one mapping of set $\{1,\dots,N^O\}$ onto itself, i.e., $\delta : \{1,\dots,N^O\} \longrightarrow \{1,\dots,N^O\}$. Thus $\delta(i) = j$ if the object originally positioned at i, is placed onto position j.*

In other words, if δ is a permutation, $\delta(i)$ denotes the position of object i in the output sequence. In our case, only a subset of all possible permutations can be performed using the pre-sorting facility.[1]

[1]For the concrete technical functionality see Chapter 3.

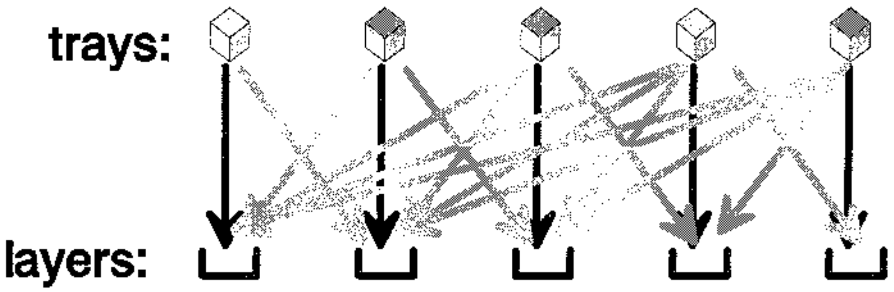

Figure 2.2.3. The set of all possible permutations for $N^S = 1$

THEOREM 2.3 *Let N^S be the capacity of the pre-sorting facility. A permutation δ is realizable, if and only if for each object i, $\delta(i) \geq i - N^S$. If $N^O \leq N^S$ then there exist $N^O!$ realizable permutations, otherwise there will be $N^S!(N^S+1)^{N^O-N^S}$.*
Proof. *(see, for instance, [39])* ■

Fig. 2.2.3 illustrates the result of Theorem 2.3. Notice, that if $N^O \leq N^S$ then there exist $N^O!$ realizable permutations and $N^S!(N^S+1)^{N^O-N^S}$ otherwise. In this work the terms *realizable* and *feasible* permutations are equivalent. Now we formally introduce the notion of a feasible permutation.

DEFINITION 2.4 *A permutation δ is feasible if for any $i = 1, ..., N^O$*

$$\delta(i) \geq i - N^S \tag{2.2.1}$$

is fulfilled.

2.2.2 Mathematical Formulation of BPSP$_1$

As was defined above, only the permutations with $\delta(i) \geq i - N^S$ are feasible. Note that only the objects at the permuted positions $\delta(i) = j$ will be placed onto layer l, where $l \equiv j \bmod N^L$, *i.e.*, l is a function of j. For example, if $N^O = 5$, $N^L = 2$, then objects with positions $j = 1, 3, 5$ will be placed onto layer $l = 1$, those with positions $j = 2, 4$ onto layer $l = 2$.

In addition we introduce the following notations:

$$\delta_{ij} = \begin{cases} 1, & \text{if } \delta(i) = j \\ 0, & \text{otherwise} \end{cases},$$

and

$$C'_{ij} = \begin{cases} 1, & \text{if layer } l,\ l \equiv j \bmod N^L, \text{ has already an object of the same color as object } i \\ 0, & \text{otherwise} \end{cases} \quad . \tag{2.2.2}$$

The optimal permutation can be constructed from the solution of the following linear program [39]:

$$\min \sum_{i=1}^{N^O} \sum_{j=1}^{N^O} C'_{ij}\delta_{ij} \quad , \tag{2.2.3}$$

$$\sum_{j=1}^{N^O} \delta_{ij} = 1, \quad 1 \leq i \leq N^O \quad , \tag{2.2.4}$$

$$\sum_{i=1}^{N^O} \delta_{ij} = 1, \quad 1 \leq j \leq N^O \quad , \tag{2.2.5}$$

$$\delta_{ij} = 0, \quad \forall i, j : j < i - N^S \quad , \tag{2.2.6}$$

$$\delta_{ij} \in \{0, 1\}, \quad 1 \leq i, j \leq N^O \quad . \tag{2.2.7}$$

We can interpret the coefficient C'_{ij} as the cost of placing object i onto position j (which uniquely identifies layer l). As (2.2.3) minimizes the total placing cost, it minimizes hence the total number of layers not yet occupied by objects of a certain color k. Such objects can populate an empty layer (empty *w.r.t.* to k) at zero cost. Infeasible permutations are excluded (depending on N^S), a priori by (2.2.6). Obviously, (2.2.6) corresponds to (2.2.1).

It is well known that this kind of integer program is totally unimodular (*cf.* [61]) and, thus, may be solved efficiently by some versions of the Simplex algorithm. Many special matching algorithms solve the problem in polynomial time (*cf.*, [72]). In practical applications (see Chapter 3), the performance very often depends on the number of attempts needed to output completely a set of orders (an order is a set of objects of different types). An attempt is considered successful if there exists at least one object of a given color on each layer (*i.e.*, belonging to the requested order). Therefore, for a given set of orders, the number of attempts needed for complete output is the maximum number of objects in these orders found on a single layer.

Consider, for instance, the following example: $N^O = 8$, $N^K = 2$ (*e.g.*, blue and yellow), $N^L = 2$, $C'_{ij} = 1$ for all i, j (*i.e.*, one blue and one yellow object already exist on each layer). Let the first four objects be blue and the others yellow. Suppose, BPSP_1 has two optimal solutions with objective function value 8:

1 Three blue objects are assigned to the first layer and one to the second; one yellow object to the first layer and three to the second.

2 Two objects of each color are assigned to both layers.

The numbers of attempts for complete output are $4+4=8$ in the first case and $3+3=6$ in the second (see Fig. 2.2.4). In terms of suffi-

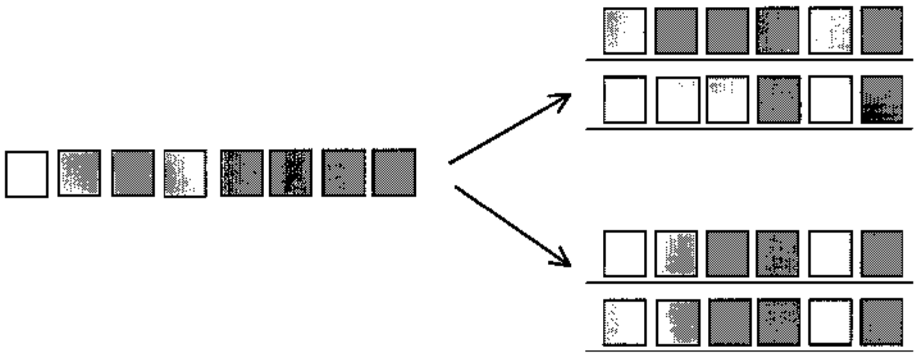

Figure 2.2.4. The example of two different assignments of objects to the storage layers.

ciency the second solution is preferable, because it needs fewer attempts for complete output. For practical applications we want to produce a solution with minimal number of attempts. Since BPSP_1 does not necessarily do so, we developed the following problem formulations.

Note that the formulation above does not contain the index k, because the information about the color of objects is hidden in the coefficients C'_{ij}. More precise, $C'_{ij} = C'_{ij}(k)$. Example 2.5 illustrates how the coefficients C'_{ij} are constructed.

2.2.3 Mathematical Formulation of BPSP_2 and BPSP_3

In this section we formulate BPSP_2 and BPSP_3 as decision problems. Most of the notations used in the previous section will be kept. Analogous to the notation C'_{ij} from Section 2.2.2 we use the notation C_{kl} - the number of objects of color k already present on layer l. Additionally we define:

- an integer bound B;

- constants

$$S_{ik} = \begin{cases} 1, & \text{if } i \in \mathcal{S}_k \\ 0, & \text{otherwise} \end{cases} \quad , \tag{2.2.8}$$

$$M_{jl} = \begin{cases} 1, & \text{if } j \equiv l \mod N^L \\ 0, & \text{otherwise.} \end{cases} \quad . \tag{2.2.9}$$

This allows us to define

$$D_{ikjl} := S_{ik} M_{jl} \quad . \tag{2.2.10}$$

Now we can formulate the following decision problems:

D-BPSP$_2$: Is there a feasible permutation δ such that the maximal cost

$$\max_{k=1,\dots,N^K,\, l=1,\dots,N^L} \left(C_{kl} + \sum_{i=1}^{N^O} \sum_{j=1}^{N^O} D_{ikjl} \delta_{ij} \right) \tag{2.2.11}$$

does not exceed B?

D-BPSP$_3$: Is there a feasible permutation δ such that the total cost

$$\sum_{k=1}^{N^K} \max_{l=1,\dots,N^L} \left(C_{kl} + \sum_{i=1}^{N^O} \sum_{j=1}^{N^O} D_{ikjl} \delta_{ij} \right) \tag{2.2.12}$$

does not exceed B?

Remark: The term $D_{ikjl}\delta_{ij}$ takes the value 1 if an additional object i of color k is placed onto layer l by permutation δ. As C_{kl} denotes the number of objects of color k already present on that layer, the cost $C_{kl}+\sum_{i=1}^{N^O}\sum_{j=1}^{N^O} D_{ikjl}\delta_{ij}$ yields the number of objects after the permuted objects have all been placed in the layers. In other words, D-BPSP$_2$ is the problem of finding a permutation of objects such that the maximal number of objects of the same color on any layer is less than or equal to B for all colors. Thus, for practical applications, the total cost term of D-BPSP$_2$ can be interpreted as a worst-case estimation of the performance and the total cost of D-BPSP$_3$, analogously, represents the average performance over all colors.

2.2.3.1 An Optimization Version of BPSP$_2$

Since the objective is to minimize the maximal cost (2.2.11), we now formulate the decision problem as an optimization problem:

$$\min \max_{k=1,\dots,N^K,\, l=1,\dots,N^L} \left(C_{kl} + \sum_{i=1}^{N^O} \sum_{j=1}^{N^O} D_{ikjl} \delta_{ij} \right) \quad .$$