TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

D I S S E R T A T I O N

Hybrid Optimization Methods for Warehouse Logistics and the Reconstruction of Destroyed Paper Documents

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

ao. Univ.-Prof. Dipl.-Ing. Dr. Günther Raidl
Institut für Computergraphik und Algorithmen E186
Technische Universität Wien

und

ao. Univ.-Prof. Dr. Ulrich Pferschy
Institut für Statistik und Operations Research
Karl-Franzens-Universität Graz

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Mag.rer.soc.oec. Dipl.-Ing. Matthias Prandtstetter
Matrikelnummer 0025600
Am Neubau 28, 2100 Korneuburg

Wien, am

Matthias Prandtstetter

Kurzfassung

DIESE Dissertation beschäftigt sich mit dem Lösen von kombinatorischen Optimierungsproblemen aus zwei unterschiedlichen Anwendungsbereichen: der *Platzverwaltung* und *Berechnung von Kommissionierungstouren* aus dem Bereich der Lagerverwaltung sowie die *Rekonstruktion von zerstörten Papierdokumenten* aus dem Bereich der Forensik. Obwohl diese beiden Probleme aus Sicht der Anwendung wenig gemeinsam haben, kann man dennoch Parallelen feststellen, wenn man sie im Detail betrachtet, da sie Varianten von wohlbekannten kombinatorischen Optimierungsprobleme sind. So ist die Lagerplatzverwaltung mit dem als *blocks world* bekannten Problem verwandt und die Berechnung von Touren sowie die Dokumentenrekonstruktion stehen stark in Beziehung zum *Handlungsreisendenrpoblem.* Zusätzlich wird eine kurze Darstellung von Standardmethoden zum Lösen schwerer kombinatorischer Optimierungsprobleme, die im Weiteren für die untersuchten Problemstellungen adaptiert werden, präsentiert.

Zuerst wird eine Variante der Lagerplatzverwaltung betrachtet, die unter anderem in der Papierindustrie angewandt wird. Die dort verwendeten Lagerhäuser zeichnen sich durch auf einander orthogonal stehende Lagergänge aus. Die Lagerplätze selbst werden mittels einer Last-In, First-Out-Strategie verwaltet. Das heißt, nur auf die letzte an einem Lagerplatz eingelagerte Papierrolle kann direkt zugegriffen werden. Will man hingegen eine weiter hinten stehende Papierrolle ausfassen, muss man alle davor platzierten Rollen entnehmen und (temporär) an anderen Lagerplätzen zwischenlagern. Das Ziel der in dieser Arbeit verfolgten Variante der Lagerplatzzuweisung besteht darin Einlagerungen zu berechnen, sodass die Anzahl der Umlagerungen während der Auslagerung, und somit die benötigte Zeit, minimiert wird. Unglücklicherweise ist die genaue Produktionsreihenfolge der Papierrollen im Vorhinein nicht bekannt, da es immer wieder zu Maschinenausfällen kommt. Weiters sind die exakten Lieferdaten nicht bekannt beziehungsweise können sich unvorhergesehen ändern, zum Beispiel aufgrund von Verspätungen der Frächter. Neben einer Ad-hoc-Zuweisungsstrategie wurden auch zwei Umordnungsalgorithmen entwickelt, die in weiterer Folge verwendet werden können, um Ad-hoc-Umlagerungen sowie größere Umorganisationen des Lagers während Stehzeiten durchzuführen. Die entwickelten Algorithmen wurden in einem Lager eines Projektpartners getestet und die erreichten Ergebnisse sowie die Rückmeldung der Arbeiter zeigten die hohe Qualität der Lösungen auf. Als zweites Problem aus dem Bereich der Lagerlogistik wurde die Berechnung von Kommissionierungstouren durch ein Lager, sodass die benötigte Zeit minimiert wird, betrachtet. Dafür wurde ein exakter Algorithmus basierend auf dynamischer Programmierung zur Berechnung von optimalen Touren in einem „klassischen“ Lagerhaus entwickelt, wobei Lagerarbeiter letztlich den Touren folgend durch das Lager gehen um die bestellten Artikel einzusammeln. Dieser exakte Ansatz wurde zum Lösen von Teilproblemen verwendet und in ein größeres Framework integriert um zusätzliche Nebenbedingungen bezüglich Lieferdaten, Kundenbestellungen und der Zuweisung von Lagerarbeitern zu Lagerwägen erfüllen zu können.

Das zweite große Themengebiet dieser Dissertation entstammt aus dem Gebiet der Forensik und beschäftigt sich mit der Rekonstruktion von zerstörten Papierdokumenten. Die berücksichtigten Aspekte können in drei Klassen eingeteilt werden: (a) das Rekonstruieren von händisch zerrissenen Papierseiten und die Wiederherstellung von (b) in Streifen oder (c) in Rechtecke geschreddertem Papier. Obwohl die Aufgabenstellung für alle drei Varianten auf den ersten Blick ähnlich ist, liegen gravierende Unterschiede im Detail: Während zum Beispiel bei der Rekonstruktion von händisch zerrissenem Papier geometrische Information ausgenutzt werden kann, sind in den beiden anderen Fällen alle Schnipsel (nahezu) gleich geformt. Deswegen wird eine Zielfunktion eingeführt, die versucht die Wahrscheinlichkeit, dass zwei Schnipsel nebeneinander platziert werden sollen, abzuschätzen. Obwohl ein endgültiges (halb-)automatisches System zur Rekonstruktion von Papierdokumenten auch Mustererkennung und Bildverarbeitung ausnutzen wird, konzentriert sich diese Arbeit primär auf einen gewissermaßen ergänzenden Ansatz. Dafür wird das Problem zuerst als kombinatorisches Optimierungsproblem formuliert, welches dann mittels Transformation auf das Handlungsreisendenproblem abgebildet und mittels variabler Nachbarschaftssuche gelöst wird. Zusätzlich werden Schranken mit Hilfe einer Lagrangerelaxierung berechnet. Ein Ameisensystem und eine variable Nachbarschaftssuche werden auf die Rekonstruktion von in (kleine) Rechtecke geschnittenem Papier angewandt. Testergebnisse zeigen, dass mit diesem Ansätzen Instanzen mit bis zu 300 Schnipseln (fast perfekt) gelöst werden können. Diese Instanzgrößen entsprechen in etwa Dokumenten mit zehn Seiten. Berücksichtigt man allerdings die Komplexität dieser Problemstellung, unterstreichen die Ergebnisse das große Potenzial der vorgestellten Lösungsansätze. Weiters konnte gezeigt werden, dass die Anzahl der nötigen Operationen eines menschlichen Rekonstruierers erheblich reduziert werden konnte.

Abstract

THE main topic of this thesis is the solving of real-world combinatorial optimization problems from two domains: *storage location assignments* and *picking tour computations* in warehouse management as well as the *reconstruction of destructed paper documents* from the field of forensics. Although from the application point of view these two topics have not much in common, parallels can be identified when analyzing them in more detail. All of them are extended versions of well-known combinatorial optimization problems, i.e., the storage location assignment is a variant of *blocks world*; whereas the tour computations as well as the reconstruction of documents are related to the *traveling salesman problem*. In addition, a short overview on the standard methods for solving hard combinatorial optimization problem is given which will then be adapted for the topics of this thesis.

First, a variant of the storage location assignment problem is examined which typically arises among others in paper industry. Related warehouses consist of aisles orthogonal to each other and storage locations are accessed using a Last-In, First-Out policy, i.e., only the last stored item is directly accessible from each storage location. In case someone wants to access a paper roll located not immediately accessible, all paper rolls placed in front of the requested one need to be removed and (temporarily) stored in other locations. The goal of the storage location assignment examined within this thesis is to compute an assignment of paper rolls to stockyards such that during shipping minimal picking times arise which is equivalent to minimizing the number of necessary relocation operations when loading. Unfortunately, the concrete production order of the paper rolls stored within the warehouse is not known in advance due to technical constraints. Even more, the shipping dates are only estimations and may suddenly change, e.g., due to delays of the carrier. However, beside the assignment of positions within the warehouse

two rearrangement strategies have been developed such that ad-hoc relocations as well as warehouse reorganizations during idle times can be performed for improving the current warehouse state. The algorithms described within this part were directly applied in the warehouse of a partner company and the results obtained with respect to the warehouse states as well as the feedback of the warehouse worker underlined the high quality of the proposed approaches.
As a second problem from the domain of logistics and warehouse management the computation of order picking tours through a warehouse such that the total order picking times are minimized is investigated. For this purpose, an exact algorithm based on dynamic programming for computing optimal tours through a "classical" stockyard which will be walked by warehouse men operating trolleys is presented. This procedure is applied as a subproblem solver within a larger framework regarding additional constraints related to shipping dates, customer orders and worker to trolley assignments.

The second large topic of this thesis originates from the field of forensics and focuses on the reconstruction of destroyed paper documents. The aspects considered here can be divided into three subdomains: reconstruction of (a) manually torn paper documents, (b) strip shredded documents and (c) cross cut shredded documents. Although the background is the same for all three of these concrete applications, they differ in important details, e.g., while for the reconstruction of manually torn paper shape information can be exploited during the restoration process, the snippets produced by strip shredders or cross cut shredders are all (almost) equally shaped. Therefore, two different error estimation functions trying to estimate the likelihood that two snippets should be aligned with each other are proposed. Although a (semi-)automatic reconstruction system will finally incorporate pattern recognition and image processing techniques, we mainly focus on a somehow complimentary approach. Therefore, this problem is firstly formulated as a combinatorial optimization problem which is then solved by first applying a transformation to the traveling salesman problem, via a hybrid variable neighborhood search incorporating human user interactions and a Lagrangian relaxation/heuristic for computing lower bounds. For the reconstruction of cross cut shredded documents additionally to a variable neighborhood search an ant colony optimization based method is applied. Experimental results document that instances with up to 300 shreds can be (almost perfectly) reconstructed using the presented approaches. This instance size, however, corresponds to documents with only a few pages, e.g., approximately ten sheets of paper using standard shredding devices. Considering the complexity of this problem the tests confirm the high potential of the proposed approaches. Even more, they show that the number of user operations when assembling destroyed documents is reduced to a minimum consisting of only a few final operations for obtaining the original document.

Danksagung

Zuallererst möchte ich Prof. Günther Raidl für seine ausdauernde und hervorragende Betreuung danken. Ohne seine Unterstützung und Ratschläge wäre vieles nicht so geworden, wie es letztlich hier niedergeschrieben ist. Auch Prof. Ulrich Pferschy, der sich bereit erklärt hat, die Arbeit zu begutachten, und der es geschafft hat, in kürzester Zeit wertvolles Feedback zu dieser Arbeit zu geben, gilt mein Dank.

Bedanken möchte ich mich weiters bei meinen Arbeitskollegen, die – jeder auf seine eigene Art – zum guten Gelingen dieser Arbeit beigetragen haben:

- Bin Hu, der mir viele Einblicke in chinesische (Essens-)Gewohnheiten gegeben hat
- Martin Gruber, dessen Unterstützung man sich in allen Belangen sicher sein kann
- Andy Chwatal, der für die eine oder andere astronomische Lehrstunde und Diskussion zur Verfügung stand
- Markus Leitner, der er schaffte, mir den Rang des „Morgens-der-erste-am-Institut-zu-Seiende“ abspenstig zu machen
- Mario Ruthmair, dessen Bodenständigkeit und Fähigkeit zur klaren Darstellung von Problemen stets die Felsen in der (organisatorischen) Brandung waren
- Philipp Neuner, Aksel Filipovic und Andreas Weiner, die stets bei technischen Problemen zur Stelle waren
- Stephanie Wogowitsch, Angela Schabel und Doris Dickelberger, die mich bei organisatorischen Tätigkeiten stets verlässlich und kompetent unterstützt haben

- sowie Ania Potocka, Raul Fechete, Patrick Klaffenböck und Thorsten Krenek, die in ihrer Funktion als Studienassistenten immer eine Stütze in der Lehre waren
- Daniel Wagner, mein ehemaliger Zimmerkollege, und Sandro Pirkwieser, mein derzeitiger Zimmerkollege, erhalten noch eine Extraportion Dank – mit ihnen habe ich über viele (kleine) Probleme diskutiert, die nach unseren Gesprächen keine mehr waren.

Selbstverständlich gilt mein Dank auch meiner Familie – vor allem meinen Eltern, die irgendwann damit aufgehört haben, mich zu fragen, wann ich denn nun endlich mit meiner Dissertation fertig werde.

Danke an meine geliebte Frau Ursula, die mich in allen Dingen stets unterstützt und mir immer motivierend zur Seite steht.

Bildung ist das, was übrig bleibt, wenn man alles,
was man in der Schule gelernt hat, vergisst.
(Albert Einstein)

Contents

Chapter 1

Introduction

Hybridization techniques are generally based on the combination of two (or more) mainly complimentary approaches lacking in some properties when being applied on their own to overcome a given problem. Only the combination of these methods, however, is in many situations capable of producing the desired output.

One prominent example for a direct application of hybridization is the *hybrid car*. This kind of automobile relies, among others, on the so called *kinetic energy recovery system* (KERS). Using KERS kinetic energy set free during brake applications is in most cases transformed into electric power, which is temporarily stored and in the following used for easing the acceleration process. It is assumed that using this technology the fuel consumption of vehicles can be optimized.

Another example arising in zoology is the *mule*, which is a crossing of a horse as mother and a donkey as father. Mules combine the power of horses while being as resilient and sure-footed as donkeys. Although mules were already of great importance in the Roman empire they are becoming more popular again since the beginning of the 21^{st} century [87].

Due to the complexity of many *combinatorial optimization problems* arising in real-world applications hybridization of methods for solving them played and plays a major role in computer sciences and operations research during the last decade and in recent years. Especially the combinations of exact methods, resulting in proven optimal solutions, with (meta-)heuristic approaches, often providing (high quality) solutions in reasonable computation times, are highly promising with respect to solution qualities as well as computation times. Nevertheless, such a “crossing” is in most cases non-trivial and

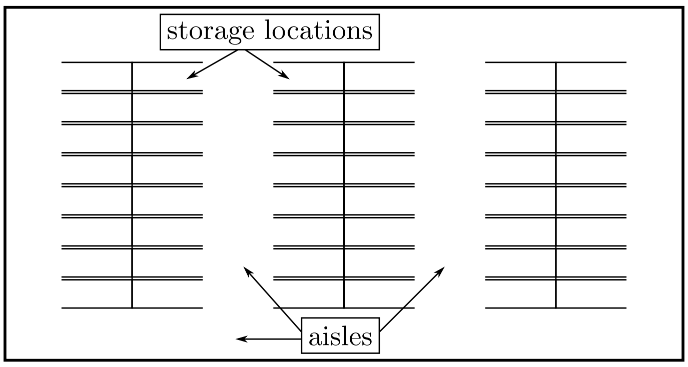


Figure 1.1.: A typical warehouse layout as considered within this thesis.

sophisticated methods for aggregating the advantages of the combined methods while minimizing their disadvantages need to be developed.

Applications are, among others, *production optimization*, *routing* and *telecommunications*, *cutting and packing*, and just to mention one of the most important application areas of combinatorial optimization: *supply chain management*—although this importance might be rooted in the fact that a vast amount of money can be earned in this field. Solving large instances of *hard* combinatorial problems is a challenging task and therefore the development of powerful algorithms for tackling them is of great importance.

This thesis deals with combinatorial optimization problems taken from two at a first glance completely different domains: While in the first part the computation of storage location assignments in warehouses as well as the computation of optimal tours through the warehouse for picking articles which were ordered by customers is considered, the second part concentrates on the reconstruction of destroyed paper documents as arising in the field of forensics.

Although the computation of storage locations is of general interest in any warehouse management system, a special case arising in paper industry is tackled within this work. A typical warehouse in this application is structured as follows, see also Fig. 1.1: all aisles are orthogonal to each other, storage locations are accessible only from one aisle and all storage locations are structured using a *Last-In, First-Out* strategy, i.e., only the article (paper roll) stored to a storage location at last can be directly accessed. In case a paper roll not directly accessible needs to be removed from the warehouse, e.g., due to shipping, all other paper rolls in front need to be relocated. Obviously, these additional movements of paper rolls slow the shipping process down. Though, it is desired that especially the time needed for loading is minimized—on the one hand to reduce the waiting time of customers and on the other hand to be able to serve as much customers as possible without increasing the for example the number of warehouse men. In addition, the likelihood that a paper roll is damaged is raised each time it is moved due to the appliance mounted on the fork trucks used for lifting paper rolls. Obviously, this leads to the requirement that paper rolls need to be assigned to storage locations such that they are sorted according to the sequence needed during shipping. Unfortunately, they are not produced according to this order. Even more, the precise production sequence is in most cases not entirely known since (additional) high priority orders may arrive lately and machine breakdowns occur from time to time. Within this work, an ad-hoc stocking strategy is developed which will be used for assigning storage locations to paper rolls on a first-come, first-serve basis, i.e., for each paper roll arriving from production a stockyard is immediately assigned. In addition, a relocation strategy is proposed reassigning paper rolls to new storage locations such that the current warehouse state can be improved according to requirements stated by the warehouse manager.

While this first investigated application assigns storage locations to items, the second topic explored within this thesis is related to computing routes through a warehouse. The underlying warehouse structure is basically the same as for the storage location assignment problem, see also Fig. 1.1, i.e., aisles orthogonal to each other and racks only accessible from one aisle. This time, however, all articles stored within one storage location are equivalent, i.e., no relocations of articles are necessary. A couple of warehouse men walk through the warehouse operating a trolley and collect articles ordered by customers by placing them on their trolley. These items are then brought to a so-called packing station where they are boxed and handed over to a shipping company. Although each customers typically orders several articles, they need not be picked by the same warehouse worker since there is an intermediate storage in the packing station. Obviously, it is tried to minimize the time needed by the worker collecting all articles, which corresponds to minimizing the lengths of the routes to be walked. For this purpose, it is self-evident that in a first step a partitioning of all ordered articles will be computed such that articles located nearby will be picked along the same tour. In a second step, the concrete determination of tours will be performed. However, there are some restrictions which will to be regarded. For example it is necessary that all articles are delivered to the packing station within a specified time and the capacities of the trolleys need to be regarded.

With respect to the restoration of destroyed paper documents, three different applications are considered: the reconstruction of *manually torn* paper documents, the reconstruction of *strip shredded* paper documents and the reconstruction of *cross cut shredded* paper documents. Although all three of these applications seem to be very similar on a first glance, crucial differences can be identified on a closer look, see also Fig. 1.2. While for the reconstruction of manually torn paper documents the shape of remnants can be exploited, all shreds obtained using a shredder device are (almost) equally shaped (and sized). One possible approach would be to reconstruct the documents based on

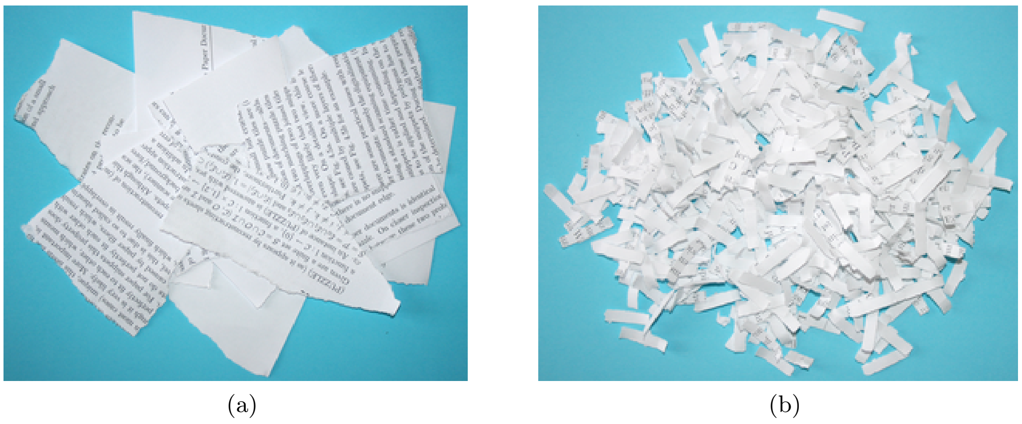


Figure 1.2.: An example for remnants as obtained when manually tearing (a) and mechanically shredding (b) paper documents.

the information contained on the front (and back) faces of the shreds. For this purpose it is convenient to apply pattern recognition and image processing techniques to gather as much information as possible. Nevertheless, after extracting useful features it is still necessary to assemble the shreds such that the original document is restored. Within this work, we entirely focus on this second step, i.e., the actual reconstruction assuming that during a preprocessing step valuable information to be exploited was obtained. The relationship of these three problems to the solving of jigsaw puzzles is obvious. In addition, it will be shown that the reconstruction of shredded documents is strongly related to the well-known traveling salesman problem. Obviously, the problem of reconstructed a potential evidence will arise in crime scene investigations. Additionally, in archeology related problems arise when trying to reconstruct clay jugs out of clay fragments or restoring frescoes possibly destructed during earth quakes.

Overview of the Thesis

This work is structured as follows: The next chapter will give a short survey of (standard) optimization techniques applied to (hard) combinatorial optimization problems including among others an introduction to linear programming, dynamic programming, variable neighborhood search and ant colony optimization.

In Chap. 3 two works related to logistics and warehouse management, namely a storage location assignment problem and a routing problem in warehouses are presented. This chapter mainly reflects the work done during a project with our industry partner Dataphone GmbH located in Vienna, Austria. Large parts of this chapter were published in

U. Ritzinger, M. Prandtstetter, and G. R. Raidl. *Computing optimized stock (re-)placements in Last-In, First-Out warehouses.* In S. Voss et al., editors, Logistik Management, pages 279–298. Physica-Verlag, 2009.

and

M. Prandtstetter, G. R. Raidl, and T. Misar. *A hybrid algorithm for computing tours in a spare parts warehouse.* In C. Cotta and P. Cowling, editors, Evolutionary Computation in Combinatorial Optimization – EvoCOP 2009, volume 5482 of LNCS, pages 25–36. Springer, 2009.

Furthermore results of this chapter have been presented at the *AIRO2008*, the annual conference of the Italian operations research society, in Italy in 2008 and the joint workshop *Entscheidungsunterstützung in der Logistik – Geographische Informationssysteme, Simulation und Optimierung* of the Austrian and German operations research societies in Salzburg, Austria, in 2008. In addition, two master theses [118, 93] related to this topic were co-supervised by the current author.

Chapter 4 focuses on the reconstruction of destructed paper documents, whereas in the first section of this chapter methods related to the reconstruction of manually torn documents are presented. This part of the chapter mainly acts as a literature overview as well as a summary of the master theses [123, 14] co-supervised by myself focusing on document reconstruction. A presentation related to this topic was given at the *11th International Workshop on Combinatorial Optimization* in Aussois, France, in 2007.

The second part of Chap. 4 deals with the reconstruction of strip shredded documents. One related master thesis [95] was supervised by us and earlier versions of this section were published in

M. Prandtstetter and G. R. Raidl. *Combining forces to reconstruct strip shredded text documents.* In M. J. Blesa et al., editors, Hybrid Metaheuristics, volume 5296 of LNCS, pages 175–189. Springer, 2008.

as well as in

M. Prandtstetter. *Two approaches for computing lower bounds on the reconstruction of strip shredded text documents.* Technical Report TR 186–1–09–01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2009. submitted to Operations Research Letters.

The third section of this chapter focuses on the reconstruction of cross cut shredded text documents and large parts were published in

M. Prandtstetter and G. R. Raidl. *Meta-heuristics for reconstructing cross cut shredded text documents.* In G. R. Raidl et al., editors, GECCO '09: Proceedings of the 11th annual conference on Genetic and Evolutionary Computation, pages 349–356. ACM Press, 2009.

Final remarks indicating possible future research directions are presented in Chap. 5.

Chapter 2

Methodologies

Within this chapter, we will focus on the presentation of some selected methods—both of exact and heuristic nature—which will be a basis for the solution approaches discussed in more detail in the remaining chapters of this thesis. First we will, however, give a definition of combinatorial optimization problems, cf. [15]:

Definition 1 (combinatorial optimization problem)**.** *A combinatorial optimization problem is a set of instances. Each instance is a pair (S, f) with S indicating a finite set of feasible solutions $x \in S$ and function $f : S \rightarrow \mathbb{R}$ assigns to each solution in S a real value $f(x)$.*

Set S is also called *search space* and the goal is to find a solution x^* such that $f(x^*) \leq f(x)$ is satisfied for all $x \in S$. Obviously x^* denotes an optimal solution.

The methods for solving hard *combinatorial optimization problems* (COPs) are as differing as the problems arising in real-world applications and academic research projects. They can, however, be classified into three main categories: *exact*, *heuristic*, and *hybrid* approaches. While exact algorithms are able to provide a proven optimal solution they are in general very time-consuming such that they can often be applied to small or moderately sized instances only. Heuristic approaches are, in contrast, often quite fast with respect to execution times, but only provide approximate solutions and usually do not provide quality guarantees. So called *approximation algorithms*—a subclass of heuristics—are capable of giving such a guarantee on the quality of obtained solutions. Hybrid algorithms, finally, try to combine advantages of both, exact and heuristic approaches, such that high quality solutions—in certain cases even including some estimation on solution quality—are returned within reasonable computation times.

On the side of exact algorithms following approaches are most prominent, among others: *dynamic programming* [13], *Branch&Bound* [139], and *constraint programming* [121], as well as the large class of approaches based on *linear programming* including *integer linear programming*, *Branch&Cut*, *Branch&Price*, *Branch&Cut&Price* [97, 100, 139] and *Lagrangian relaxation* based techniques [11].

With respect to heuristics a further classification can be done: Roughly speaking there are *construction heuristics* and *improvement and repair heuristics.* While the former aim in generating a solution to a given problem, the latter try to improve a given (possibly invalid) solution with respect to some objective function. Among all types of heuristics the most straightforward ones are so called *greedy heuristics*, which construct a solution from scratch by choosing and adding an immediately most lucrative appearing solution component until a complete solution is obtained. Hereby, they never withdraw a made decision. Other heuristics are based on the concept of *local search*, which aims to improve a given solution by small, i.e., local, changes. Improvements are always accepted, while worse solutions are discarded. These approaches are, however, often very problem specific and they are in general finally caught in local optima. To escape those valleys containing local optima *metaheuristics* are often applied, which are more general solution strategies specified in more abstract ways and can therefore be applied to a wide range of different problems. Successfully applied metaheuristics based on local search [62] are, among others, *simulated annealing* [61], *tabu search* [44], *iterated local search* [88], and *variable neighborhood search* [58]; inspired by nature, especially biology, are *ant colony optimization* [33], *particle swarm optimization* [72], and population based approaches like *evolutionary algorithms* [9], *memetic algorithms* [96], or *scatter search* [47].

2.1. Exact Methods

The large class of exact methods can further be divided in several subclasses of algorithms following different paradigms. In this work we focus, however, only on a few selected general schemes like *dynamic programming*, *(integer) linear programming* based approaches and *Lagrangian relaxation.*

2.1.1. Dynamic Programming

Dynamic programming (DP) was developed in the 1950s by Bellman [13]. The basic principle of dynamic programming is to divide a given problem P_0 into a sequence of subproblems $P_1, P_2, \ldots, P_k$ such that subproblem P_k can be (trivially) solved and a solution to problem P_i can be directly derived from P_{i+1}, with $i = 0, 1, \ldots, k-1$.

In contrast to the apparently related concept of *Divide&Conquer* the subproblems are dependent of each other.

Bellman presents a rule on how to derive subproblems for a given problem P_0:

> "Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [13]

To be able to efficiently apply DP to combinatorial optimization problems this implies that on the one hand, the subproblems are extracted such that the optimal solution to subproblem P_{i+1} can be exploited when constructing an optimal solution to P_i, with $0 \leq i \leq k-1$. On the other hand, it needs to be guaranteed that the results obtained for already solved subproblems are stored, e.g., in memory, and can be efficiently accessed when deriving other solutions, cf. [23].

Although in most cases DP based approaches are applied to polynomially solvable problems, there are situations in which even for $\mathcal{NP}$-hard problems DP is successful. One such optimization problem is the classical *0/1 knapsack problem* (KP); see [71] for a comprehensive study on knapsack problems: Given are a set of n items having profits p_j and weights w_j, for $1 \leq j \leq n$, and a constant maximum capacity c. The goal is now to find a subset S of the items such that the profit $p(S) = \sum_{j \in S} p_j$ of the selected items is maximized while the capacity constraint $\sum_{j \in S} w_j \leq c$ is respected.

Using DP the 0/1 knapsack problem can be solved in pseudo-polynomial time, i.e., in time complexity polynomially bounded by the instance size n when all w_j are integer and c is polynomially bounded by n. For this purpose, a matrix $\mathrm{m}_{i,j}$ and a matrix $\mathrm{s}_{i,j}$, with $0 \leq i \leq n$ and $0 \leq j \leq c$, is defined whereas the field $\mathrm{s}_{i,j}$ corresponds to a selection of the first i items having a capacity equal to j. The field $\mathrm{m}_{i,j}$ is set to the associated profit of the selection in $\mathrm{s}_{i,j}$. Following recursion is used to compute matrix m:

$$\mathrm{m}_{0,0} = 0 \tag{2.1}$$

$$\mathrm{m}_{0,j} = -\infty, \qquad \forall j \in \{1, \ldots, c\} \tag{2.2}$$

$$\mathrm{m}_{i,0} = 0, \qquad \forall i \in \{1, \ldots, n\} \tag{2.3}$$

$$\mathrm{m}_{i,j} = \mathrm{m}_{i-1,j}, \qquad \forall i \in \{1, \ldots, n\}, j \in \{0, \ldots, w_i - 1\} \tag{2.4}$$

$$\mathrm{m}_{i,j} = \max\left\{\mathrm{m}_{i,j}, \mathrm{m}_{i-1,j}, \mathrm{m}_{i-1,j-w_i} + p_i\right\}, \qquad \forall i \in \{1, \ldots, n\}, j \in \{w_i, \ldots, c\} \tag{2.5}$$

The values of matrix s are set accordingly. For this purpose, all values are first initialized to $\emptyset$. Then the sets yielding the profits stored in $\mathrm{m}_{i,j}$ are saved in $\mathrm{s}_{i,j}$, with $i \in \{1, \ldots, n\}$ and $j \in \{w_i, \ldots, c\}$, i.e., $\mathrm{s}_{i,j} = \arg\max_{S \in \left\{s_{i,j}, s_{i-1,j}, s_{i-1,j-w_i} \cup \{i\}\right\}} \{p(S)\}$.

The basic idea of this model is to first solve the problem of optimally packing the knapsack with just one item ($i = 1$), i.e., the item is selected or not. These two solutions correspond to the fields $\mathrm{m}_{1,0}$ and m_{1,w_1}, respectively. All other fields of row i are set to $-\infty$ using the above rules and indicate that no solutions for the corresponding fields exist. The solution for deciding which of the first two items should be packed is now derived from the solution for just one item by either packing the second item to the first one, selecting only the second one, selecting only the first one or selecting none of both. The corresponding profit values of these solutions can, as long as the total weight does not exceed c, be found in the fields m_{2,w_1+w_2}, m_{2,w_2}, m_{2,w_1} and $\mathrm{m}_{2,0}$, respectively. Obviously, the situation can occur that two different selections of items have the same weight. In this case, the solution with the better, i.e., higher, profit is stored for further computations while the worse solution is discarded. The profit of the best solution can be obtained via finding that column j with maximum associated profit for the problem regarding all solutions $\mathrm{m}_{n,j}$. Obviously, the corresponding selection of items is stored in $\mathrm{s}_{n,j}$.

2.1.2. Integer Linear Programming

There is a vast amount of combinatorial optimization problems which can be modeled as an *integer linear program*, e.g., the knapsack problem [71] already introduced in the previous section or the well-known traveling salesman problem [84]. An *integer linear programming* (ILP) formulation consists of a linear objective function to be maximized over a set of integral decision variables and linear inequalities and equalities to be fulfilled. A basic introduction into this field is given in [139]. For more advanced techniques we refer to [97]. Each ILP can also be written in the following standard form:

$$\begin{aligned} \max\ & cx \\ \text{s.t. } & Ax \le b \\ & x \ge 0 \text{ and integer} \end{aligned} \tag{ILP}$$

where A is an $m \times n$ coefficient matrix, c an n-dimensional row vector, b an m-dimensional column vector and x an n-dimensional column vector of integer variables, with $m, n \ge 1$. In some situations it is convenient to also allow some non-integer, i.e., rational, variables y. The so-called *mixed integer linear program* (MIP) can then be written as

$$\begin{aligned} \max\ & cx + hy \\ \text{s.t. } & Ax + Gy \le b \\ & x \ge 0 \text{ and integer} \\ & y \ge 0 \end{aligned} \tag{MIP}$$

where c, A, b and x is defined as above and G denotes a $m \times p$ coefficient matrix, h a p-dimensional row vector, and y a p-dimensional column vector of fractional variables.

In case variables x in (ILP) are restricted to the binary domain $\{0, 1\}$, the resulting formulation is called *binary integer program* (BIP):

$$\begin{aligned} \max\ & cx \\ \text{s.t. } & Ax \leq b \\ & x \in \{0, 1\} \end{aligned} \tag{BIP}$$

Many *combinatorial optimization problems* (COPs) can be formulated as ILP, MIP or BIP, e.g., see [71] for formulations of the knapsack problem. When modeling a certain COP it might be more natural to express the problem as minimization problem which in addition might contain equalities. It is, however, easy to express each equality by two inequalities (one with $\leq$ the other with $\geq$) and multiply one by -1. In addition, each maximization problem can be transformed into an equal minimization problem (again by multiplication with -1) [97, 139]. Therefore, we assume in the following section that we have a maximization problem.

Linear Programming

Let us now introduce some additional definitions and observations which will be used within the thesis: The *linear programming* (LP) relaxation of a MIP (or ILP or BIP) is obtained by omitting the integrality constraints for formulation (MIP) (or (ILP) or (BIP))

$$\begin{aligned} \max\ & cx + hy \\ \text{s.t. } & Ax + Gy \leq b \\ & x \geq 0 \\ & y \geq 0 \end{aligned} \tag{LP}$$

The set $P_{\text{MIP}} = \{x : Ax + Gy \leq b, x \geq 0 \text{ and integer}, y \geq 0\}$ corresponds to the set of feasible solutions to (MIP) while $P_{\text{LP}} = \{x : Ax + Gy \leq b, x \geq 0, y \geq 0\}$ denotes the set of valid solutions to (LP). It is easy to show that $P_{\text{MIP}} \subseteq P_{\text{LP}}$ since any solution $x \in P_{\text{MIP}}$ is also member of P_{LP}. Now, it is obvious that $\max_x \{cx : x \in P_{\text{MIP}}\} \leq \max_x \{cx : x \in P_{\text{LP}}\}$ holds and therefore the objective value obtained by solving the linear program, i.e., determining $x \in P_{\text{LP}}$ minimizing the objective function cx, is always greater than or equal to the value obtained for solving the original mixed integer program. Since this relationship always holds, the minimal LP relaxation value is always an *upper bound* for the optimal objective value of the original problem. Using interior point methods [73] or the ellipsoid method [70] the LP relaxation value can be obtained in time polynomially bounded in the number of variables and constraints. In practice, however, variants of the simplex algorithm are most often used due to its typically better

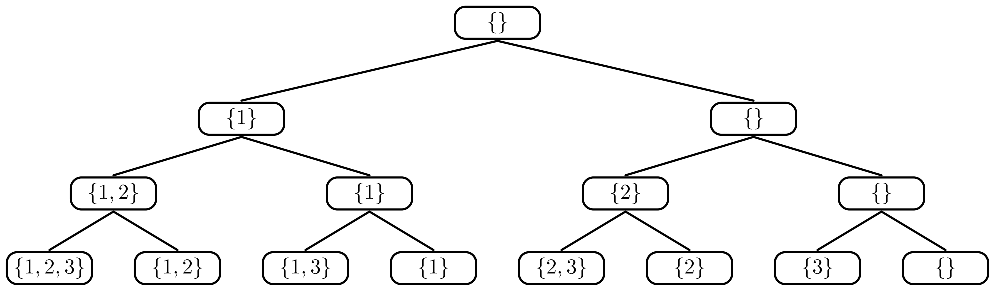


Figure 2.1.: An enumeration tree for a knapsack instance with $n = 3$ items.

runtime-behavior, although its worst case runtime is exponential. An introduction to this highly important algorithm [31] is given in [23].

The simplex algorithm exploits the following observations for set P_{MIP}:

- $P_{\text{MIP}} = \emptyset \Rightarrow$ no solution exists, i.e., the LP is *infeasible*.
- $P_{\text{MIP}} \neq \emptyset$ and $\nexists \inf\{cx : x \in P_{\text{MIP}}\} \Rightarrow$ the LP is unbounded, i.e., although it is feasible no optimal solution can be identified.
- $P_{\text{MIP}} \neq \emptyset$ and $\exists \inf\{cx : x \in P_{\text{MIP}}\} \Rightarrow$ there exists an optimal solution $x^* = \min\{cx : x \in P_{\text{MIP}}\}$. However, there exists a vertex $\overline{x}$ of P_{MIP} such that $c\overline{x} = cx^*$.

LP based Branch&Bound

Using linear programming techniques upper bounds on the objective value of the optimal solution for given integer linear programs can be computed. Lower bounds can essentially only be obtained by computing solutions to the original problem [139]. Obviously, the best lower bound corresponds to the objective value of the optimal solution. Although for some problems finding (any) solution is easy, the open question is: “How to find *good* solutions?”. This is in general $\mathcal{NP}$-hard, i.e., it is very unlikely that an algorithm exists which is polynomially bounded in the instance's input size. One naive approach for obtaining solutions to the original problem would be to apply complete enumeration on the set of decision variables. E.g., for the 0/1 knapsack problem introduced in Sec. 2.1.1 this would result in enumerating all possible selections of items to be packed. Clearly each of the selections needs to be checked whether it is valid with respect to the capacity constraint. Since all possible selections are inspected the optimal one is visited, too. Although this approach might be applicable to (very) small instances the number of solutions to be checked is exponential in the number of items n, i.e., $O(2^n)$. The corresponding enumeration tree for an instance of the 0/1 knapsack problem with $n = 3$ items is shown in Fig. 2.1. The root of the tree corresponds to the initialization. The first level corresponds to the decision whether or not item 1 should be packed into the knapsack. Level two corresponds to the decision for item 2, and so on. To keep the number of nodes in this tree as small as possible it is usual to constrain the tree. For example, if a selection of items is already violating the capacity constraint it is not necessary to further investigate all solutions containing the violating selection as subset. Furthermore, (upper) bounds on the solution quality can be computed for each subtree not yet examined. E.g., one such bound can be computed as the already packed profit plus the profit of all so far not considered items. If this sum is less than or equal to the profit of an already obtained solution, the corresponding branch of the tree needs not to be examined anymore since no further improvement can be achieved in that subtree. Obviously, the bounds generated by this heuristic will not be very tight since the constraint on the capacity is completely disregarded.

However, linear programming techniques can be incorporated in such an approach. See Alg. 1 for pseudocode of this procedure. Let us assume that polyhedron $P = \{x : Ax \leq b, x \geq 0\}$ corresponds to the LP relaxation of the ILP to be solved. If solution x^{LP} obtained is integer, i.e., the relaxed integer constraints are fulfilled, the ILP is solved to optimality. Otherwise, heuristics incorporating the solution to the LP can be used for generating a lower bound, i.e., a feasible solution. Anyhow, at least one variable x_i, with $1 \leq i \leq n$ is fractional. Let x_i^{LP} be the value of variable x_i in x^*. Now, two new subproblems P_1 and P_2 can be specified with

$$P_1 = \left\{x : Ax \leq b, x \geq 0, x_i \leq \lfloor x_i^{\mathrm{LP}} \rfloor\right\} \tag{2.6}$$

$$P_2 = \left\{x : Ax \leq b, x \geq 0, x_i \geq \lceil x_i^{\mathrm{LP}} \rceil\right\} \tag{2.7}$$

This step is also referred to as *branching*. By recursively solving the newly generated subproblems three cases can occur:

- If the best lower bound, i.e., the best obtained feasible solution, is equal to the local upper bound, i.e., the value of the current LP relaxation, then the current best solution is optimal with respect to the current subtree, i.e., this branch of the tree can be pruned.
- If the best lower bound is greater than the local upper bound, the corresponding subtree can be pruned, since the best achievable solution in this branch cannot be better than the best already obtained solution.
- In all other cases, i.e., if the gap between the lower and the upper bound is positive, the process has to be further iterated for this subtree.

More advanced techniques may be additionally applied for improving this LP-based Branch&Bound algorithm. For example, it is common to incorporate a *cutting plane* [139] approach when the number of constraints in the original formulation is large or even

Algorithm 1: Branch&Bound

```
Input: set $\{Ax \leq b, x \geq 0$ and integer$\}$
Data: $\overline{z} \ldots$ local upper bound
Output: optimal solution $x^*$
$P \leftarrow \{Ax \leq b, x \geq 0\}$;
set of subproblems $S \leftarrow \{P\}$;
while $S \neq \emptyset$ do
    // select one subproblem and remove it
    $P \leftarrow$ select one problem in $S$;
    $S \leftarrow S \setminus \{P\}$;
    // solve the subproblem
    $x^{\text{LP}} \leftarrow \min\{cx : x \in P\}$;
    if $P$ is infeasible then
        continue; // prune the tree
    $\overline{z} \leftarrow cx^{\text{LP}}$;
    if $\overline{z} < cx^*$ then
        continue; // prune the tree
    if $x^{\text{LP}}$ is integral then
        $x^* \leftarrow x^{\text{LP}}$; // set new incumbent
        continue; // prune the tree
    // compute an integral solution using heuristics
    $\overline{x} \leftarrow$ a heuristic solution to $P$;
    if $c\overline{x} > cx^*$ then
        $x^* \leftarrow \overline{x}$;
    // select a fractional $x_i \in x^{\text{LP}}$
    $P_1 \leftarrow \left\{x : x \in P, x_i \leq \lfloor x_i^{\text{LP}} \rfloor\right\}$;
    $P_2 \leftarrow \left\{x : x \in P, x_i \leq \lceil x_i^{\text{LP}} \rceil\right\}$;
    $S \leftarrow S \cup \{P_1, P_2\}$;
return $x^*$;
```

exponential in the instance size. If the number of variables is exponential *column generation* [90] is applied which iteratively adds promising variables on demand. The resulting approach is then referred to as *Branch&Cut* and *Branch&Price* [139], respectively.

2.1.3. Lagrangian Relaxation

As already outlined in the previous section the computation of lower bounds is one crucial point in the successful application of Branch&Bound algorithms. Although the computation of bounds using LP relaxations is a standard technique better bounds can sometimes be obtained by applying the so-called *Lagrangian relaxation* [11] (LR). Let us assume the following ILP formulation for a minimization problem is given

$$\min cx \tag{2.8.1}$$
$$\text{s.t. } Ax \geq b \tag{2.8.2}$$
$$Bx \geq d \tag{2.8.3}$$
$$x \in \{0,1\} \tag{2.8.4}$$

where A is an $m \times n$ and B and $p \times n$ coefficient matrix, respectively, c an n-dimensional row vector, b and d an m-dimensional and and p-dimensional column vector, respectively, and x an n-dimensional column vector of integer variables, with $m, n, p \geq 1$.

Now, it is easy to define a Lagrangian relaxation by associating *Lagrangian multipliers* λ with constraints (2.8.2) and relaxing them into the objective function:

$$\begin{aligned} &\min cx + \lambda(b - Ax) \\ &\text{s.t. } Bx \geq d \\ &\quad x \in \{0,1\} \end{aligned} \tag{LR}$$

Therefore, λ is a n-dimensional row vector. It can be shown that an optimal solution of formulation (LR) provides a lower bound on the optimum of formulation (2.8) for any $\lambda \geq 0$ [11]. Obviously, it would also be possible to relax constraints (2.8.3) instead of constraints (2.8.2). However, normally those constraints are relaxed which make a problem “hard” to solve. In addition, we are interested in finding among all possible values for λ that set of multipliers which maximizes the lower bound, i.e.,

$$\max_{\lambda \geq 0} \left\{ \begin{aligned} &\min cx + \lambda(b - Ax) \\ &\text{s.t. } Bx \geq d \\ &\quad x \in \{0,1\} \end{aligned} \right\} \tag{LD}$$

This problem is also referred to as *Lagrangian dual program* (LD). There are two issues open related to a development of an LR: Which of the constraints should be relaxed?

As already mentioned, one normally will relax the "hard" ones but at the same time it can be shown that in case formulation (LR) exhibits the *integrality property*, which states that the LP relaxation of (LR) is always integral, the bounds obtained by LR are equal to the value obtained via a standard LP relaxation. Therefore, one will in general try to find a set of constraints to be relaxed into the objective function such that the remaining formulation does not contain the integrality property.

The second issue is related to the computation of optimal λ-values, i.e., the determination of λ maximizing (LD). Fortunately, this problem turns out to be piecewise linear and concave. The standard approach in practice is to apply a *subgradient procedure* [11]. The idea is to iteratively solve the Lagrangian dual program and to compute for each element of λ a so-called subgradient indicating how much the corresponding constraint is satisfied or violated. In case a relaxed constraint is still violated the corresponding λ_i is enlarged while it is reduced when the constraint is "over-satisfied". Choosing appropriate values for strategic parameters the iterative process can be controlled and fine tuned.

Lagrangian Heuristic

Additionally to the computation of lower bounds by applying Lagrangian relaxation, it is also possible to generate solutions to the original formulation by developing a so-called *Lagrangian heuristic* (LH). The idea is to derive solutions from the values obtained via LR which will constitute upper bounds on the original problem. Again, the process can be prematurely terminated as soon as the lower bound and the upper bound coincide.

Although LR and LH both are not exact approaches on their own for obtaining solutions to their underlying problem, they are often incorporated in a Branch&Bound algorithm for computing lower and upper bounds which then results in an exact method.

2.2. Metaheuristics

Although exact methods theoretically provide optimal solutions to any given problem instance, the practical application of such methods is often limited to small and moderately sized instances—especially when applied to hard combinatorial optimization problems. Therefore, it is convenient to sacrifice proven optimal solutions for the sake of desired time and memory performance by applying (meta-)heuristics.

By the term *metaheuristic*, which has been introduced by Glover [45], a large class of algorithms is denoted which have in common that they are relatively abstractly specified, problem independent approaches guiding and controlling low-level, problem-specific heuristics. Due to their nature they can be applied to a huge amount of optimization

Algorithm 2: LocalSearch

```
Input: initial solution $x$
       neighborhood structure $N$
Output: possibly improved solution $x$
repeat
    // get neighbor of $x$
    $x' \leftarrow \mathrm{step}(N(x))$;
    // improvement or not?
    if $f(x') < f(x)$ then
        $x \leftarrow x'$;
until a stopping criterion is met ;
return $x$;
```

problems by simply adapting and/or interchanging their problem-specific parts. Based on these principles they rely on, the following classes of metaheuristics, among others, can be identified: *local search* based metaheuristics like *simulated annealing* [61], *iterated local search* [88], *variable neighborhood search* [58] and *tabu search* [44] as well as population based and nature inspired metaheuristics like *evolutionary algorithms* [9], *ant colony optimization* [33], *memetic algorithms* [96], *genetic programming* [81] and *particle swarm optimization* [72]. Within this work only a small subset of these metaheuristics is applied which will be shortly introduced in the next few sections.

2.2.1. Local Search

Local search [62] (LS) tries to improve a given initial solution by iteratively applying small changes, so-called *moves*. *Neighborhood structures* used for defining *neighborhoods* of candidate solutions of a current solutions are one main concept of LS. More formally they can be defined as follows:

Definition 2. *A neighborhood structure N is a function $N : S \rightarrow 2^S$ mapping each solution $x \in S$ in a set S of feasible solutions to a set of neighbors, the so-called neighborhood $\mathcal{N}(x)$ of x.*

Let us assume that an objective function $f : S \rightarrow \mathbb{R}$ is given assigning each candidate solution x a real objective value. The most straightforward concept of a LS is shown in Alg. 2. The function step($N_i(x)$)—also called *step function*—indicates which of the neighbors of x should be selected. The most frequently applied step functions are:

best improvement: When applying this examination strategy neighbor x' of solution x is returned for which $f(x') \leq f(x'')$, with $x'' \in N(x)$, applies, i.e., among all neighbors that one is chosen which imposes the best improvement.

first improvement: This step function is often also called *next improvement* and returns that neighbor which is the first one (according to the examination order of the neighborhood) improving the current objective value.

random neighbor: Among all neighbors in $N(x)$ for a solution x one will be randomly chosen. Although the selection of the neighbor is quite fast an improvement occurs with lower probability than for the other two step functions.

Let us now introduce the concept of a local minimum:

Definition 3 (local minimum)**.** *A solution $\overline{x}$ is called a local minimum with respect to neighborhood structure N if $f(\overline{x}) \leq f(x)$, with $x \in N(\overline{x})$, holds.*

When applying either best or first improvement as step function in Alg. 2 no further improvement can be found as soon as a local minimum is reached. Obviously, it will be convenient to terminate LS at this point. However, when using a random neighbor strategy it cannot be reliably stated whether or not a local optimum is obtained. Therefore, it is common to terminate LS in such situations as soon as for example a given number of iterations without improvement or a predefined time limit was reached. Finally, let us introduce the term of a global optimum:

Definition 4 (global optimum)**.** *A solution x^* is said to be globally optimal when $f(x^*) \leq f(x)$, with $x \in N$, for any theoretical neighborhood structure N holds. I.e., a solution is globally optimal if it is locally optimal with respect to all (possible) neighborhood structures.*

Let us note that obviously a solution which is locally optimal does not need to be globally optimal.

2.2.2. Variable Neighborhood Search

Variable Neighborhood Search [94, 56, 58] (VNS) is a local search based metaheuristic which tries to overcome the drawback of pure local search approaches often getting stuck in local optima. To escape them, perturbation moves, so-called *shakings*, are performed which, in contrast to multi-start heuristics, try to preserve large amounts of a local optimum. This behavior is based on the observation that often local minimums are relatively close to each other. Therefore, randomly changing only subparts of a local optimum raises the probability that the search can be continued in another close region

Algorithm 3: Basic Variable Neighborhood Search

```
Input: initial solution $x$
       a set of neighborhood structures $\mathrm{N}_1, \ldots, \mathrm{N}_{k_{\max}}$
Output: possibly improved solution $x$

$k \leftarrow 1$;
while $k \leq k_{\max}$ do
    // shaking
    $x' \leftarrow$ randomly choose one solution in $\mathrm{N}_k(x)$;
    // perform local search
    $x' \leftarrow$ LocalSearch($x'$);
    // improvement or not?
    if $f(x') < f(x)$ then
        $x \leftarrow x'$;
        $k \leftarrow 1$;
    else
        $k \leftarrow k + 1$;
return $x$;
```

of the search space. For this purpose it is necessary to define neighborhood structures N_i, with $1 \leq i \leq k_{\max}$ used as a basis for the shaking moves. Basically, N_{i+1}, with $1 \leq i \leq k_{\max} - 1$ will be chosen in such a way that $\mathrm{N}_i(x)$ contains in general solutions closer to x than solutions in $\mathrm{N}_{i+1}(x)$. The shaking neighborhoods will then be examined systematically, i.e., a random move to a solution contained in N_{i+1} will only be applied when the last iteration of VNS starting with a random move with respect to N_i did not improve the current best solution, with $1 \leq i \leq k_{\max} - 1$. However, as soon as an overall improvement during the local search phase could be achieved, the shaking will restart with N_1. For an outline of this metaheuristic we refer to Alg. 3.

2.2.3. Variable Neighborhood Descent

In contrast to local search *variable neighborhood descent* [56, 58] (VND) tries to systematically explore multiple neighborhood structures. However, in contrast to basic VNS the step function applied is in general either best or next improvement and not random neighbor for examining the neighborhood structures. An ordering $\mathcal{N}_1, \ldots, \mathcal{N}_{l_{\max}}$ of the neighborhood structures is defined. VND now tries to find a local minimum with respect to neighborhood structure $\mathcal{N}_l$ and proceeds with $\mathcal{N}_{l+1}$ if it was found starting with $l = 1$. Anyhow, the case can occur that an improvement could be identified in $\mathcal{N}_l$, with $1 \leq l \leq l_{\max}$. Then the search is proceeded with resetting l to 1. By using this

Algorithm 4: VariableNeighborhoodDescent

```
Input: initial solution $x$
       neighborhood structures $\mathcal{N}_1, \ldots, \mathcal{N}_{l_{\max}}$
Output: a local optimum with respect to all neighborhood structures
        $\mathcal{N}_1, \ldots, \mathcal{N}_{l_{\max}}$
$l \leftarrow 1$;
while $l \neq l_{\max}$ do
    // get neighbor of $x$
    $x' \leftarrow \mathrm{step}(\mathcal{N}_l(x))$;
    // improvement or not?
    if $f(x') < f(x)$ then
        $x \leftarrow x'$;
        // return to the first neighborhood structure
        $l \leftarrow 1$;
    else
        // proceed with the next neighborhood structure
        $l \leftarrow l + 1$;
return $x$;
```

systematic approach, the ordering of neighborhoods must be done such that $\mathcal{N}_l \not\subseteq \mathcal{N}_{l-1}$ holds, for $2 \leq l \leq l_{\max}$. If $\mathcal{N}_l \subseteq \mathcal{N}_{l-1}$ would hold for any $2 \leq l \leq l_{\max}$ it is obvious that an examination of neighborhood $\mathcal{N}_l(x)$ cannot yield better results than an (already completely performed) exploration of $\mathcal{N}_{l-1}(x)$ for any solution x. See Alg. 4 for pseudocode of this procedure.

The key concept of VND is the observation that a global optimum is a local optimum with respect to all possible neighborhood structures while the reverse does not necessarily hold. Therefore the success of a concrete application of VND to a combinatorial optimization problem is mainly based on the proper definition of neighborhood structures and an appropriate ordering of them. In general, this order will be determined based on the size or time complexities for examining the neighborhoods, i.e., small neighborhoods are explored first and only in case no improvement can be found in them the search is extended to more complex structures.

Anyhow, in some situations no truly convincing static order can be identified. Then it is promising to apply strategies for dynamically changing the sequence on neighborhood structures. In [112] Puchinger and Raidl propose to order the neighborhoods for a certain solution x according to decreasing improvement potential of the neighborhoods. This potential can, for example, be identified using linear programming techniques. In contrast to this approach which only tries to estimate the benefit of operations performed in the future, i.e., moves to be applied according to the neighborhood structures, Hu and Raidl [64] proposed a dynamic ordering based on the contribution of each neighborhood structure during the already performed search. Here the ordering of neighborhood structures during the next VND iterations is determined by computing a performance rating depending on the time used for examining the corresponding neighborhood during the last iteration and the quality of the solution obtained by this exploration.

Approaches like these can be applied whenever the order of the neighborhood structures is not induced by their definition, i.e., examination times complexities and/or inclusions of neighborhood structures in other. However, there are also situations in which neighborhood structures contribute relatively often during the beginning of the search procedure but loose their potential in later iterations, see for example Sec. 3.2.

General Variable Neighborhood Search Scheme

While VNS suffers from the weakness of the local search applied VND lacks in the ability to escape local optima with respect to all used neighborhood structures. It is therefore convenient to combine both metaheuristics by applying VND as local search procedure during VNS. The resulting metaheuristic is also referred to as *general variable neighborhood search scheme*. However, be aware that the neighborhood structures used in VNS for shaking moves are in general different to the neighborhood structures defined for the embedded VND. There is a large variety of other VNS and/or VND based approaches: For example, by omitting the local search phase of VNS a so-called *reduced variable neighborhood search* is obtained. An introduction to VNS/VND in general and to variants of them can be found in [57].

2.2.4. Ant Colony Optimization

The *ant colony optimization* [34, 33] (ACO) metaheuristic is member of the large class of nature inspired algorithms. The development of ACO was inspired by the behavior of real ants when finding paths between their home and food locations. Instead of directly communicating with each other, an indirect communication is established using so-called *pheromone*. While walking along the paths pheromone trails are laid which can be followed by other ants. Analogously to nature, a group of agents—also called (artificial) ants— is defined which is guided by (artificial) pheromone. Throughout the search process this pheromone information—typically stored in a pheromone matrix—is dynamically updated and provides a basis for decisions made by the agents. Since each ant updates the pheromone matrix according to solutions built by itself, this matrix represents somehow a long-term memory of all solutions found by the agents. However,

Algorithm 5: AntColonyOptimization

```
Input: number $m$ of ants to be used
Output: the best obtained solution

initialize pheromone matrix;
while termination condition not met do
    construct $m$ candidate solutions based on pheromone and heuristic
    information;
    apply local search; // optional
    update pheromone matrix;
return best so far found solution;
```

it is important that solutions are not only constructed by considering the pheromone matrix but also locally available information is exploited. It should be noted that ants are relatively autonomous in the sense that they are independent of each other and only communicate with each other by the pheromone trails laid.

The general ACO metaheuristic is shown in Alg. 5. As can be seen, the process is started by initializing the pheromone matrix to meaningful values. Often it is convenient to uniformly initialize the matrix but there are also applications where a more advanced initialization is applied, e.g., [52]. Afterwards each of the $m \geq 1$ available ants constructs a new solution. To these solutions a local search based procedure can be applied. Finally, an update of the pheromone matrix is done, whereas first—again in analogy to nature—a certain amount of pheromone is evaporated and then the new pheromone is laid with respect to the solutions obtained by the ants. Different formulas can be applied for this final, but crucial step. For an advanced presentation of different methods we refer to [34].

Although this concept when first presented as an *ant system* on the *traveling salesman problem* [32] did not convince due to non-competitive results in comparison with the state-of-the-art approaches, other variants and a wide range of applications like vehicle routing, scheduling and the quadratic assignment problem [41, 91, 128, 92] led to the today's importance of this metaheuristic.

2.3. Hybrid Approaches

As already outlined exact methods often suffer from the fact that the running times typically increase dramatically with increasing problem sizes. (Meta-)heuristic approaches on the other hand are often able to provide good solutions in acceptable times but without any guarantee on the quality of the solution. Approximation algorithms are somehow a special case, since they provide a quality measure under certain conditions. However, there are problems for which no constant factor approximation algorithm exists, unless $\mathcal{P} = \mathcal{NP}$, e.g., the traveling salesman problem [122, 84]. Nevertheless, another promising approach is to combine advantages of both, exact and metaheuristic, approaches by hybridizing them. Highly successful applications of such hybridization techniques have been proposed in recent years [63, 76, 102, 103, 104, 106, 113]. Hybridization approaches based on exact and heuristic methods can, however, be partitioned in two large classes [110, 113, 114]: *collaborative combinations* and *integrative combinations*. Collaborative hybridizations are characterized by the fact that two approaches exchange information with each other but do not incorporate the other approaches, i.e., they are executed in parallel, intertwined or sequential. The second class of combinations consists, however, of hybridizations where one approach incorporates the other one, i.e., either metaheuristics using exact methods as subordinates or exact methods incorporating heuristics for solving subproblems. In addition, there are also other approaches focusing on the hybridization of two (or more) metaheuristics, e.g., [135]. Furthermore, sometimes it is even tried to incorporate human problem solving abilities in (semi-)automatic systems, e.g., [79].

Prominent representatives for the incorporation of exact methods for solving subproblems are *very large-scale neighborhood search* [2] (VLNS) techniques. The basic idea is to define complex neighborhood structures resulting in (very) large neighborhoods used within a local search based approach. However, an exploration of such neighborhoods using naive enumeration methods will not be applicable and more advanced approaches are necessary. Depending on the concrete definition of the neighborhood structure, dynamic programming or (integer) linear programming based techniques will be more convenient. For example, for the traveling salesman problem exchanging simultaneously $k > 1$ edges of the current solution will in general result in $O(2^k)$ different candidate solutions. Obviously, for large k a naive (complete) enumeration of the candidate solutions is not possible. For a dynamic programming based VLNS approach for the TSP we refer to [37]. Another approach based on the incorporation of dynamic programming techniques in a *corridor method* is presented in [20]. A similar approach with respect to the basic idea of fixing some variables while others are left free was presented in [106] where the neighborhood is examined by applying integer linear programming techniques.

For exact methods incorporating metaheuristics one can refer to the *Branch&Bound* algorithm presented in Sec. 2.1.2. Here the exact algorithm is mainly dependent on the computation of proper bounds. These bounds are often computed using heuristics, e.g., for solutions to the original problem construction heuristics regarding the variables set during branches are incorporated. Furthermore, one crucial point in the efficient application of Branch&Bound to hard optimization problems is the definition of a branching rule, i.e., a (almost) deterministic method for deciding on which variable branching will be performed. In addition, during each iteration of Branch&Bound a currently unsolved subproblem needs to be selected. This can of course be done using a naive enumeration method but it will obviously be more promising when heuristics are applied for estimating the improvement potential for each subtree. Obviously, that one with maximal possible improvement should then be selected.

During *Branch&Cut* approaches it is necessary to separate violated cuts to be added to the model during the next iteration. However, even the separation of the cuts can be a hard problem which implies that this needs to be done using heuristics for large instance sizes. For approaches using this concept we refer to [109, 51]. Similar ideas can be applied in a *Branch&Prize* approach for which the generation of new columns is a hard optimization problem. For a recent work related to heuristic column generation we refer to [101].

Chapter 3

Logistics and Warehouse Management

Nowadays, logistics and warehouse management, which are subfields of supply chain management [83], are one of the most important tasks in production environments. On the one hand it has to be assured that all components needed during the production process are available and on stock but on the other hand the costs induced by storage capacity, i.e., storage space, should be minimized. Furthermore, the access times to individual items located in the storage should be kept as low as possible in order to optimally serve customers as well as to reduce idle times during production.

Recent investigations revealed that about 33 percent of money invested in logistics can be attributed to the costs arising in inventory management [138]. Therefore, a proper investigation of savings that might be achieved within this part of supply chains is necessary and is in many cases profitable. We refer to [138] for a literature review on this topic over the last 30 years.

One of the main factors in the organization of warehouses is, among others, the throughput policy to be used within the storage [129]. The best known strategies are last-in, first-out (LIFO) and first-in, first-out (FIFO) policies. Nevertheless, there are also other policies especially developed for different kinds of goods to be stored, e.g., first-produced, first-out (FPDO) or first-expire, first-out and first-deliver, first-out (FDFO). FPDO and FDFO mainly find application in lines of businesses coping with products having best-before dates assigned [129]. While these main inventory decisions are strongly dependent on the line of business and have to be made when building warehouses, there are other decisions to be made on a daily or even shorter basis, e.g., the planning of routes and tours for drivers when delivering goods or the assignment of articles to storage locations during the stocking process.

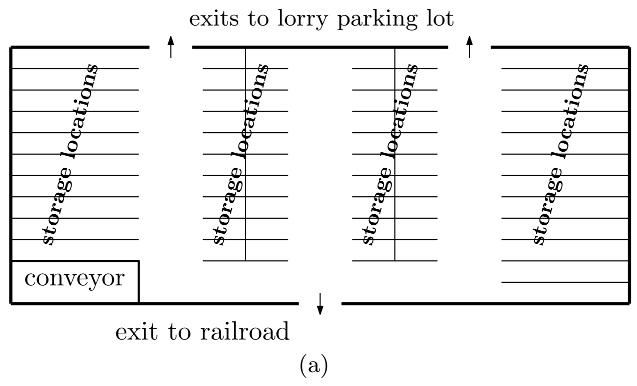
Within this chapter we focus on two specific problems arising in logistics and warehouse management: a variant of a *storage location assignment problem* as well as a *route finding problem*. While the former focuses on the computation of (optimal) storage locations for articles to be stored within the warehouse, the latter concentrates on the computation of tours throughout the warehouse such that the total length of the routes to be walked by warehousemen and therefore the overall working time of the workers is minimized. Of course, various side constraints to be described in the corresponding sections of this chapter need to be regarded in both applications.

Although the quality of an assignment of storage locations to articles has a direct influence on the lengths of walks to be processed later by warehousemen during picking, these two problems are often considered independently of each other. This is based on the fact that in most situations either the tour finding is trivial, e.g., each article has to be delivered on its own, cf. Sec. 3.1, or the storage locations are predefined, cf. Sec. 3.2. Furthermore, tours through a warehouse strongly depend on the structure of the warehouse. Therefore, a reorganization of the storage exploiting properties of the goods to be stored as well as typical customer behavior [36] may be more convenient in some situations.

Both approaches presented within this section are the outcome of a cooperation with industry partners, namely Dataphone GmbH, located in Vienna, Austria, and Hamburger-Spremberg located in Pitten, Austria, and Spremberg, Germany. The obtained results were published in [107, 119] and presented at four international conferences and workshops in the years 2008 and 2009.

3.1. Location Assignment

In this section, we focus on a storage location problem arising in paper industry. To optimize the production efficiency, e.g., minimize cutting loss, it is common to temporarily store the just produced paper rolls in an intermediate store until the customers pick up their orders. Obviously it is desired that the stocking process should be performed as fast as possible. Therefore, it is highly preferable that all paper rolls of currently served customers are directly accessible. If this is not the case blocking rolls, i.e., those paper rolls preventing the direct access of warehousemen to the requested rolls, should be reinserted into the storage at the best possible location. At the end of a working day there might be additional time left for reordering part of the warehouse such that all orders to be processed during the next day are optimally accessible.



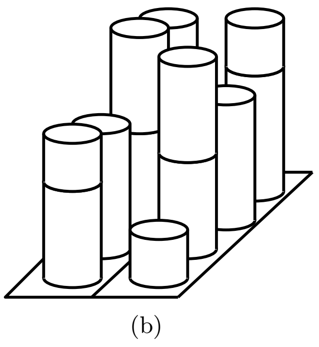


Figure 3.1.: (a) Schematic plan of the warehouse and (b) extract of the storage.

In the next two sections a detailed description of the underlying problem and a more formal definition including an evaluation function for estimating the likelihood of additional paper roll relocations during stock removal are given. Based on this evaluation function a stocking strategy is presented in Sec. 3.1.4, and Sec. 3.1.5 includes the description of relocation strategies utilizing variable neighborhood descent and greedy methods. Both approaches are evaluated on different warehouse states in Sec. 3.1.6.

3.1.1. Production Process

In paper industry it is common to apply a three stage production process: At first paper roll blanks are produced which are cut into paper rolls of individual lengths in the second stage. At last the produced rolls are shipped.

In our case, the first two steps of the production process are optimized such that all ordered rolls of same paper type and grammage are consecutively produced and that the offcut of each blank is as small as possible. Due to this the production order of rolls is not sorted according to any attribute that is relevant for shipping. After cutting the paper rolls are transported on a conveyor belt into the warehouse.

The warehouse itself is organized as follows: The storage consists of parallel aisles and each aisle contains to its left and right side storage locations for storing rolls, also called strips, see Fig. 3.1a. Although, there is no physical separation between the strips they are always accessed from the corresponding aisle. In each strip the first roll is placed at the end of it and all further rolls are put in according to the last-in, first-out throughput policy, see Fig. 3.1b for an extract of two strips with rolls stored at them. The strips are all physically identical except with respect to their capacities. There are some strips dedicated to the storage of rolls of short length, i.e., rolls with less than 1250 mm. Such rolls can be loaded in a more space-saving way and thus they should be stored together at theses special locations. There are three workers (one is responsible for placing the rolls from production and two are assigned for picking up outgoing rolls) who are equipped with forklift trucks for transporting the rolls, mobile terminals displaying various information (e.g., where the roll should be stored) and mobile bar code scanners to communicate each operation (removal and placing of rolls) to a centralized database. Therefore, the current state of the warehouse including the advance in the loading of paper rolls is known at any time.

When the assignment of rolls to storage locations is done in a simple, greedy manner it often occurs that rolls have to be rearranged to provide direct access to those rolls for shipping. This leads to increased removal times. Therefore it is important that the removal sequence of the rolls is considered at the time of storage and hence the most important criteria are the shipping dates of the rolls. Besides this there are also other attributes: It is desired that the rolls are grouped by customers, i.e., all rolls for one customer should be at the same location. In addition, each customer has a preferred type of shipment (by lorry or freight car) and therefore the rolls should be placed in those strips which are near the according exit, see also Fig. 3.1a. For a complete list of constraints and different aims regarded in this work and their influence on an evaluation function used for our optimization approach, see Section 3.1.2.

Unfortunately, it is in our case not possible—neither for the workers nor for any computer aided decision system—to gather the exact production sequence. This is mainly caused by frequent breakdowns or failures of certain machines needed for special types of paper such that for a short term other paper types are produced, or express orders of customers which have to be fulfilled almost immediately. Thus, the investigated storage location assignment problem is an *online problem* [4]. A further reason for not globally planning a fixed storage location for each roll is that customers frequently pick up their order lately or even too early.

Therefore, the depositer has to estimate the best available storage location for each new roll to be stored. In literature there exists some work for the related *storage location assignment problem* [17, 1, 59, 66], which has been shown to be $\mathcal{NP}$-hard [1]. Anyhow, according to the classification used in [17] the storage location assignment problem examined within this work uses class-based storage as stock location assignment strategy, i.e., each roll is assigned to a certain class according to its attributes and then (arbitrarily) stored at a location dedicated to that class of paper rolls. To overcome this arbitrariness in selecting the best available storage location for each paper roll, we propose a finer grained evaluation function within this work that assigns to each (possible) warehouse state a positive value approximately indicating the likelihood of occurring conflicts, i.e., paper roll reallocations, during stock removals. The most promising paper roll assignments are suggested to the warehousemen via a mobile terminal.

Although this approach works quite well, it must not be disregarded that due to stock removals there might arise the situation that an explicit rearrangement of paper rolls would significantly improve the warehouse situation, i.e., reduce the number of conflicts in future stock removals. Therefore, we additionally use a *variable neighborhood descent* [58] based approach for computing rearrangement operations that can be performed by warehousemen currently not busy. Again, there are some requirements which have to be regarded when implementing such a method. Mainly, it has to be assured that after each single rearrangement operation, the warehouse state is reasonably good such that in case the rearrangements have to be suspended the stock removal operations can still be efficiently performed. Due to the fact that rearrangement operations are only performed when no other jobs are to be completed it is not possible to count on these reallocations during storage location assignment. In addition, the number of times one roll is moved should be kept low.

3.1.2. Problem Definition

Within this section we provide a more formal definition of the introduced problem. In addition we present an evaluation function which will be used in the further context for approximately indicating the likelihood of conflicts, i.e., necessary paper roll reallocations, arising during stock removal operations.

We are given a warehouse W and a set of n_r paper rolls $R = \{1, \ldots, n_\mathrm{r}\}$, which includes all rolls in the system. The warehouse itself consists of n storage locations $i \in W = \{1, \ldots, n\}$, which are organized according to a last-in, first-out throughput policy. Therefore, we can define a tuple $S_i = (s_{i,1}, \ldots, s_{i,f_i})$ for each strip $i \in W$ indicating that roll $s_{i,l}$ has been assigned to i before $s_{i,l+1}$, with $1 \leq l < f_i$ and $f_i \in \mathbb{N}_0$ indicating the fill level of strip i, i.e., the number of rolls stored in storage location $i \in W$. While each strip $i \in W$ has a maximum capacity c_i, each paper roll $j \in R$ has a given weight w_j. Obviously, at each time $\sum_{j \in S_i} \mathrm{w}_j \leq \mathrm{c}_i$ holds. Further we are given a set of n_o orders O requested by costumers, where each order $K \in O$ is a set $K \subseteq R$ of paper rolls and define the set $\Omega(i) = \{K \in O \mid \exists\, l : s_{i,l} \in K\}$ as the set of all orders having at least one paper roll $l \in R$ stored in strip $i \in W$. In addition, we define set $D_K = \{i \in W \mid S_i \cap K \neq \emptyset\}$ as the set of storage locations containing at least one roll of an order $K \in O$. As already mentioned the exact shipping date is not known, but an expected shipping date d_K as well as the preferred shipping mode $\mathrm{m}_K \in \{\mathrm{truck}, \mathrm{train}\}$ are given for each order $K \in O$.

Each paper roll $j \in R$ has a certain positive length, and those rolls with a length shorter than 1250 mm are called *small goods*. Constants $\mathrm{w}_i^{\mathrm{small}} \in \{0, 1\}$ indicate which storage locations $i \in W$ are dedicated to storing small goods; for these locations $\mathrm{w}_i^{\mathrm{small}}$ is one and for all others zero. It is, however, possible (but not favored) that small and large paper rolls are stored in the same strip, cf. also Eq. (3.6) defined later. Similarly, constants $\mathrm{w}_i^{\mathrm{truck}} \in [0,1]$ and $\mathrm{w}_i^{\mathrm{train}} \in [0,1]$ define with which preference paper rolls shipped by truck or train should be stored at locations $i \in W$, respectively. Note that $\mathrm{w}_i^{\mathrm{truck}} + \mathrm{w}_i^{\mathrm{train}} = 1$ does not necessarily hold. Finally, we denote by $\mathcal{W}$ the (current) warehouse state $\mathcal{W}$, i.e., a snapshot of the current situation in the warehouse.

Evaluation Function

To present a method for either finding the best storage location(s) for a given paper roll to be stored or moving operations for improving the situation in the warehouse within this work, it is necessary to develop an evaluation function that approximately indicates the likelihood of future conflicts, i.e., paper roll reallocations becoming necessary, during stock removals for a given warehouse state $\mathcal{W}$. The basic concept of this evaluation function $\mathrm{E}(\mathcal{W})$ is as follows: In case $\mathrm{E}(\mathcal{W}) = 0$ holds it is very likely that during stock removals no additional reallocations of paper rolls are necessary. With increasing value of $\mathrm{E}(\mathcal{W})$ this likelihood decreases, i.e., the likelihood of reallocations increases. In addition, not only the likelihood increases but also the expected number of occurring conflicts, i.e., more reallocations will become necessary during each stock removal step. Therefore the value of $\mathrm{E}(\mathcal{W})$ is not (strongly) bounded from above, since it is almost always possible to generate a worse warehouse state by adding an additional roll to the storage that generates additional conflicts. The evaluation process, however, is based on restrictions implied by observations stated by the warehouse manager of our paper production company.

The evaluation of a current warehouse state $\mathcal{W}$ is done in two steps: firstly all possible conflicts arising in single strips $i \in W$ are computed and secondly a rating regarding the complete warehouse is done. Both values are then appropriately weighted and the sum represents the objective value of the warehouse state. The most important reason for conflicts is the expected shipping date d_K of the orders $K \in O$. Therefore, we introduce function $\mathrm{date}(i) \geq 0$ counting the number of conflicts in strip $i \in W$ occurring with respect to the shipping dates associated with the rolls stored in i. A conflict occurs for two paper rolls $s_{i,l}$ and $s_{i,l'}$, with $s_{i,l}, s_{i,l'} \in S_i$, iff $l < l'$ and $\mathrm{d}_K < \mathrm{d}_{K'}$, with $s_{i,l} \in K$, $s_{i,l'} \in K'$, respectively, and $K, K' \in O$, i.e.,

$$\mathrm{date}(i) = \sum_{l=1}^{f_i - 1} \sum_{l'=l+1}^{f_i} \chi^{\mathrm{d}}(s_{i,l}, s_{i,l'}), \qquad \forall i \in W, \tag{3.1}$$

with $\chi^{\mathrm{d}}(j, j') = 1$ iff roll j is going to be shipped before j'; otherwise $\chi^{\mathrm{d}}(j, j') = 0$, with $j, j' \in R$.

Another important reason for conflicts is the mix-up of different orders within one strip. Therefore function order(i) counts the number of different orders stored at location $i \in W$; in addition the inhomogeneity of orders stored in the same strip is considered, yielding the following definition:

$$\mathrm{order}(i) = |\Omega(i)| + \sum_{l=1}^{f_i-1} \chi^{\mathrm{o}}(s_{i,l}, s_{i,l+1}), \qquad \forall i \in W, \tag{3.2}$$

with $\chi^{\mathrm{o}}(j, j') = 1$ iff the orders of rolls $j, j' \in R$ are different; otherwise $\chi^{\mathrm{o}}(j, j') = 0$. Next, the number of strips used for storing all rolls of an order $K \in O$ is computed. Function distr($\mathcal{W}$) sums up these distribution values of all orders:

$$\mathrm{distr}(\mathcal{W}) = \sum_{K \in O} D_K. \tag{3.3}$$

Since the orders of customers are known, it can be decided if there are still some paper rolls in production or not. Of course, it should be emphasized that each order is stored at few locations (the best case is if only one strip is needed for each order). Therefore, it is meaningful to reserve space in the storage for each uncompleted order. This is done by using function cap($\mathcal{W}$) which computes the number of paper rolls not yet stocked and which cannot be assigned to the same storage location as the other paper rolls of the same order:

$$\mathrm{cap}(\mathcal{W}) = \sum_{K \in O} \max \left\{ \left| K \setminus \bigcup_{i' \in W} S_{i'} \right| - \max_{i \in W} \left(\mathrm{c}_i - \sum_{j \in S_i} \mathrm{w}_j \right), 0 \right\} \tag{3.4}$$

In addition, function compl($\mathcal{W}$) counts the number of orders that are not yet completely stocked and have at least one stocked roll blocked by another one of another order. Thus, blocking of not yet completely stocked orders is also penalized.

The process of loading the rolls on lorries or freight cars should be finished as fast as possible by minimizing the lengths of the paths the warehousemen have to move the rolls. Therefore, we store paper rolls preferably close to the exit presumably later used during stock removal. This is done according to function ship(i), with $i \in W$, under the assumption that v_j^{t} is equal to one iff roll $j \in R$ should be shipped with trucks; otherwise $v_j^{\mathrm{t}} = 0$ holds:

$$\mathrm{ship}(i) = \sum_{j \in S_i} \left(v_j^{\mathrm{t}} \cdot \left(1 - \mathrm{w}_i^{\mathrm{truck}}\right) + \left(1 - v_j^{\mathrm{t}}\right) \cdot \left(1 - \mathrm{w}_i^{\mathrm{train}}\right) \right), \qquad \forall i \in W. \tag{3.5}$$

Analogously, function small(i), with $i \in W$, increases when long paper rolls are stored in strips dedicated to small goods or when short paper rolls are stored in strips not dedicated to them. Under the assumption that $v_j^{\mathrm{s}} = 1$ for paper rolls $j \in R$ with a length shorter than 1250 mm and $v_j^{\mathrm{s}} = 0$ otherwise, small(i) can be defined as:

$$\mathrm{small}(i) = \sum_{j \in S_i} \left| v_j^{\mathrm{s}} - \mathrm{w}_i^{\mathrm{small}} \right|, \qquad \forall i \in W. \tag{3.6}$$

Empty strips are the most valuable ones because virtually all paper rolls or even orders can be stored in them while increasing the likelihood for conflicts only minimally, if at all. Therefore, it should be well considered at which time empty strips will be started to be used. For this, we define a function empty(i) increasing the objective function by a small amount if strip $i \in W$ is not empty, i.e.,

$$\mathrm{empty}(i) = \begin{cases} 0 & \text{if strip } i \text{ is empty} \\ 1 & \text{otherwise} \end{cases} \tag{3.7}$$

Finally, we define the evaluation function $\mathrm{E}(\mathcal{W})$ as a linear combination of all previously defined functions. Each of the sub-functions is weighted using an appropriate coefficient in order to balance the influence of the individual components among each other:

$$\begin{aligned} \mathrm{E}(\mathcal{W}) = \sum_{i \in W} \Big(& \gamma^{\mathrm{d}} \cdot \mathrm{date}(i) + \gamma^{\mathrm{o}} \cdot \mathrm{order}(i) + \\ & \gamma^{\mathrm{s}} \cdot \mathrm{ship}(i) + \gamma^{\mathrm{e}} \cdot \mathrm{empty}(i) + \gamma^{\sigma} \cdot \mathrm{small}(i) \Big) + \\ & \gamma^{\delta} \cdot \mathrm{distr}(\mathcal{W}) + \gamma^{\kappa} \cdot \mathrm{cap}(\mathcal{W}) + \gamma^{\mathrm{c}} \cdot \mathrm{compl}(\mathcal{W}) \end{aligned} \tag{3.8}$$

Though objective function (3.8) covers the most important aspects during stocking operations in our particular paper industry application, there are further special cases that can be considered. For a more detailed approach we refer to [118].

3.1.3. Related Problems and Complexity

Although the here proposed storage location assignment problem and the related relocation problem tackled in Sec. 3.1.5 is strongly restricted due to the underlying real-world application it is nevertheless possible to find parallels to other related problems which mainly differ in the constraints imposed by industrial settings and environments.

For example, container yards in large seaports mainly possess the same characteristics as warehouses in our storage (re-)location problem, i.e., containers are stored in so-called bays consisting of stacks of containers. Obviously, only that container placed on top of a stack can be directly accessed. To be able to face competition it is, however, for seaport container terminal operators important to keep the loading times of ships as short as possible. Therefore, it is convenient that during unproductive times in terms of loading operations reorganization operations are performed such that the containers can be directly loaded into the vessel according to the desired order imposed by the shipping company owner. However, to keep the reorganization workload as low as possible, it is desired to find a work plan which consists of a minimized number of container movements.

Depending on the time when the reorganization is performed there are different variants of this reorganization problem: The so-called *block relocation problem* (BRP) mainly focuses on the reorganization of the bay(s) during loading operations, i.e., movements of containers only occur whenever a container is blocked by other containers placed on top of it. One main characteristic of this problem variant is that the number of containers in the bay is consequently reduced, i.e., as soon as a container currently needed is on top of a stack it is shipped. Approaches tackling this problem are, among others, a *corridor method* based approach incorporating a dynamic programming formulation [20], a *Branch&Bound* based algorithm [75] as well as a heuristic decision rule [75]. In [74] an evaluation approach estimating the number of relocation operations was presented.

Another variant of this problem is the so-called *pre-marshalling problem* which differs from BRP simply by the fact that all operations are performed in idle times and therefore containers cannot be directly loaded into vessels. Obviously, they have to be stored in the bay and therefore a final arrangement of containers is search such that during loading operations no additional reshuffling operations are necessary. However, the number of container movements should be minimized. Beside an application of a *corridor method* based approach [19] a local search based heuristic incorporating ILP techniques for solving subproblem [85] as well as a multi commodity flow model [86] were proposed.

For a survey on container terminal operations and (optimization) problems related to the operation of container yards we refer to [127]. Although these problems have been extensively studied both from the practical and theoretical point of view no statements on the complexity of the related problem variants can be found literature. Nevertheless, by reformulating these problems it is possible to utilize theoretical results to other closely related problem. One of these problems is the problem of sorting a given (unordered) sequence of integers using complete network of stacks [80]. In this problem the goal is to find a minimum number of shuffles, i.e., moves of numbers between stacks, such that a sorted sequence of the inputs is achieved. In [80] it is shown that this problem is $\mathcal{NP}$-hard. Obviously, this problem corresponds to the block relocation problem as arising in container terminals. A second result is obtained for the so-called *blocks-world planning* problem which asks to find a minimal schedule of moves such that a valid rearrangement of blocks is achieved. The blocks can either be placed on top of each other or on the table. Certain further restrictions can apply, e.g., certain placements of blocks on other

Algorithm 6: Stocking Strategy

```
Input: $\mathcal{W}$: current warehouse state, $j \in R$: paper roll
Data: $bestStrip$: so far best strip for paper roll $j$,
      $bestEval$: value of best so far found warehouse state
Output: strip $i \in W$ roll $j$ should be assigned to

$bestStrip \leftarrow$ null ; $bestEval \leftarrow \infty$;
foreach $i \in W \setminus \{i'\}$ do
    $\mathcal{W}' \leftarrow \mathcal{W}$ after adding paper roll $j$ to strip $i$ /* assuming $i$ is not full */
    if $\mathrm{E}(\mathcal{W}') < bestEval$ then
        $bestStrip \leftarrow i$;
        $bestEval \leftarrow \mathrm{E}(\mathcal{W}')$;
return $bestStrip$;
```

blocks can be required and/or forbidden for valid solutions. Gupta and Nau [53] showed that this problem including the variant directly corresponding to our relocation problem is $\mathcal{NP}$-hard.

3.1.4. Stocking Strategy

Based on function (3.8) it is easy to develop a straightforward greedy stocking strategy. For this purpose, one simply needs to compute the changes in $\mathrm{E}(\mathcal{W})$ when adding the new roll alternatively to each feasible strip in the storage. That one resulting in the best warehouse state is chosen. For pseudocode of this procedure see Alg. 6.

Influence of the Weighting Coefficients

As already mentioned above each of the sub-functions of the objective function (3.8) is weighted by a factor for controlling its influence on the evaluation of a given warehouse state. Unfortunately, it is not trivial to find a parameter setup being valid for any production setting. Even more, there exists no generally good weighting factor adjustment. Although this might be disappointing for warehouse operators, this circumstance holds a crucial advantage: By tuning these parameters and adapting the relations it is possible to implement different stocking strategies. For example it might be promising to ensure for certain customers that paper rolls ordered by them are directly accessible all the time. In this case one will increase the weighting factor γ^o such that a mix-up of orders becomes very unlikely. Nevertheless, this behavior might not be appropriate for all customers. In such a case the weighting factors may even be differently instantiated in dependence of the customers which finally leads to a more complex but also significantly more flexible objective function.

In this work, we use the following fixed weighting factors which were determined after consulting the warehouse manager of our case company: $\gamma^{\mathrm{d}} = 150$, $\gamma^{\mathrm{o}} = 5$, $\gamma^{\mathrm{s}} = 1$, $\gamma^{\mathrm{e}} = 20$, $\gamma^{\sigma} = 40$, $\gamma^{\delta} = 25$, $\gamma^{\kappa} = 20$, $\gamma^{\mathrm{c}} = 50$.

3.1.5. Relocation Strategy

Due to the online and stochastic aspects of our problem, even the best stocking strategy finally results in suboptimal storage situations, which means that reallocations are necessary during shipment. Additionally, empirical data provided by our case company implies that the filling level of the storage usually is about 70–80%, i.e., the stocking opportunities are rather limited. When applying the above presented stocking strategy over a longer time, it is only able to prevent major immediate conflicts to a certain degree anymore. For generally improving the warehouse state and to exploit idle times of warehousemen it is possible to perform relocations of paper rolls. Of course, the aim of this is to reduce the number of conflicts occurring during stock removals and improve the warehouse situation.

There are different types of possible relocations: The first class of reallocations is necessarily to be performed during stock removal operations when one or more paper rolls are blocking rolls to be shipped. The second and more laborious type is performed during idle times of warehousemen. Dependent on the available time various movements can be done. To be flexible in this point the system has to accept inputs from the workers indicating the number of rolls to be reallocated resulting in a list of movements improving the current warehouse state.

One obvious approach for achieving this is a greedy method presented in the next section selecting always the next best paper roll for relocation. In addition, we present a variable neighborhood descent based approach generating movement lists to be processed by the warehouse workers.

Greedy Reallocation

The *greedy reallocation procedure* (GRP) removes a roll $j \in R$ which is directly accessible and causes conflicts during removal operations with highest probability, i.e., under the assumption that $\mathcal{W}$ denotes the current warehouse state

$$j = \arg\min_{j' \in \bigcup_{i \in W} \{s_{i,f_i} \in S_i\}} \left\{ \mathrm{E}(\mathcal{W}') \mid \mathcal{W}' = \mathcal{W} \text{ after removing roll } j' \right\}. \tag{3.9}$$

Algorithm 7: GreedyRelocationProcedure($\mathcal{W}$, n_m)

```
Input: $\mathcal{W}$: current warehouse state, $n_\mathrm{m}$: number of available relocation moves
Data: $\mathcal{W}'$, $\mathcal{W}''$: intermediate warehouse states
Output: L: list of moves to be performed
repeat
    $j \leftarrow \arg\min_{j' \in \bigcup_{i' \in W} \left\{ s_{i',f_{i'}} \in S_{i'} \right\}} \{\mathrm{E}(\mathcal{W}') \mid \mathcal{W}' = \mathcal{W}$ after removing roll $j'\}$;
    $\mathcal{W}' \leftarrow \mathcal{W}$ after removing $j$;
    $i \leftarrow \arg\min_{i' \in W} \{\mathrm{E}(\mathcal{W}'') \mid \mathcal{W}'' = \mathcal{W}'$ after storing roll $j$ at location $i'\}$;
    $\mathcal{W}'' \leftarrow \mathcal{W}'$ after moving $j$ to strip $i$;
    if $\mathrm{E}(\mathcal{W}'') < \mathrm{E}(\mathcal{W})$ then
        $\mathcal{W} \leftarrow \mathcal{W}''$;
        add appropriate movement instructions to L;
        $cnt \leftarrow cnt + 1$;
until $cnt \geq n_\mathrm{m}$ or no further improvement could be achieved ;
return L;
```

Afterwards this roll is reinserted into the storage at the best strip $i \in W$, i.e.,

$$i = \arg\min_{i' \in W} \left\{ \mathrm{E}(\mathcal{W}'') \mid \mathcal{W}'' = \mathcal{W}' \text{ after storing roll } j \text{ at location } i' \right\}, \tag{3.10}$$

where $\mathcal{W}'$ denotes the warehouse state after removing paper roll j from its current strip. This procedure is repeated until either the number of available moves n_m is reached or there is no further improvement achievable, i.e., roll j is best stored at its original storage location. An outline of the pseudocode is given in Alg. 7. For experimental results obtained using GRP we refer to Sec. 3.1.6.

Variable Neighborhood Descent Based Approach

Obviously, the main disadvantage of GRP lies in the fact that moves are selected on a purely greedy basis disregarding the improvement potential of moves to be investigated in further steps. Thus, it is not possible or at least very unlikely to resolve conflicts arising in connection with paper rolls not directly accessible. We propose an approach based on *variable neighborhood descent* (VND) [58].

VND itself basically exploits the observation that a local optimum with respect to one neighborhood is not necessarily a local optimum with respect to another and any global optimum is also locally optimal with respect to any neighborhood. Therefore, a successful application of VND mainly relies on a set of multiple appropriately defined neighborhood structures, which are systematically examined. For this purpose, the first neighborhood structure is searched until no further improvement can be achieved. Then, the next neighborhood structure is examined, but as soon as an improvement could be achieved the search is continued using the first neighborhood structure again. This is repeated until a solution is found that is locally optimal with respect to all used neighborhood structures, see also Sec. 2.2.3.

This implies that two criteria must be regarded when following a VND based approach: on the one hand the proper definition of neighborhood structures to be used and on the other hand an appropriate order for examining these neighborhood structures. Both aspects will be discussed in more detail in the following.

Neighborhood Structures The main idea of the neighborhood structures used in this work is to resolve conflicts with respect to the sub-functions of the objective function (3.8) step-by-step. The neighborhood structures are defined as follows:

N_1: A neighborhood based on this structure consists always of only one solution, namely that one which can be obtained by applying the above presented greedy reallocation procedure to a given warehouse state $\mathcal{W}$. In other words, the exclusive member of this neighborhood can be obtained by identifying that roll which increases $\mathrm{E}(\mathcal{W})$ most.

N_2: Any solution contained in a neighborhood based on this structure can be obtained by applying a move which first removes all rolls from a strip $i \in W$ such that no conflicts with respect to the shipping dates occurs in strip i, see Eq. (3.1). Afterwards the removed rolls are immediately greedily reassigned to other strips $i' \in W \setminus \{i\}$, i.e., in the same order as they are removed.

N_3: This neighborhood structure is based on the idea that the jointly relocation of multiple rolls already directly stored in the same strip and being member of the same customer order might resolve possible conflicts with respect to shipping dates, format specifications or shipping mode. Therefore, to obtain a solution according to this neighborhood structure, one reallocation of such a group of paper rolls to one other strip which is best-suited, i.e., for which the objective is minimal, is performed.

N_4: While in neighborhood structure N_3 groups of paper rolls being part of the same customer order are moved, the basic principle of this neighborhood structure is to create such groups. For this purpose, a move for this neighborhood structure consists of first removing all directly accessible rolls contained in the requested customer order. Let us denote the set of strips affected by this first part of the move by $W' \subseteq W$. During the second part of the move, the strips are then moved to one strip $i \in W'$, i.e., the underlying move concentrates (a part of) one order in a strip.

N_5**:** This neighborhood structure is defined via moves which first remove the minimum number of rolls from a specified strip i such that $S_i \subseteq K \in O$ holds. Obviously, the rolls removed are reassigned based on the greedy reallocation procedure to other strips.

N_6**:** This neighborhood structure is defined for resolving conflicts caused by rolls placed deep inside of any strip, i.e., rolls which were assigned to this storage location relatively early. Therefore, one move with respect to this neighborhood structure consists of removing all paper rolls of a strip $i \in W$ and then greedily assigning them to other strips $i' \in W \setminus \{i\}$ in the same order they were removed. Obviously, each solution contained within a neighborhood based on N_6 contains at least one strip which is totally empty.

While the size of $N_1(\mathcal{W})$ is in $O(1)$ for any warehouse state $\mathcal{W}$, the sizes of all other neighborhoods is in $O(|W|)$: for each of these neighborhoods the underlying moves effect one strip while the rest of the moves is mainly based on the greedy reallocation procedure (and therefore deterministic). Using appropriate datastructures $N_1(W)$ can be examined in time $O(|W|^2)$. While $N_2(W)$, $N_5(W)$ and $N_6(W)$ can be examined in time $O(|W|^2 \cdot \max_{i \in W} \{|S_i|\})$ the time for the search in neighborhoods $N_3(W)$ and $N_4(W)$ is bounded from above by $O(|W| \cdot \max_{K \in O} \{|K|\})$. In order to achieve this an incremental update of the evaluation function is implemented and the following two step functions are used:

Resolving Most Conflicts (RMC): This step function selects among all solutions in a certain neighborhood $N(\mathcal{W})$ that one which rearranges those rolls causing the most conflicts with respect to $\mathrm{E}(\mathcal{W})$. Although very similar, preliminary tests revealed that a *best improvement* strategy is not as promising as this RMC step function. Anyhow, in case of ties the first found is selected.

Proportionally Random Neighbor (PRN): When using this step function for examining a neighborhood $N(\mathcal{W})$ one of the candidate solutions contained in $N(\mathcal{W})$ is chosen randomly. The probability for choosing one solution is proportional to the contribution of the moved rolls to the objective function (3.8), i.e., a roulette wheel selection [50] is applied. Therefore, it is very likely to relocate rolls causing conflicts with high probability. At the same time, not so promising candidate solutions might also contribute to a finally computed list of relocation moves.

Neighborhood Order Beside the definition of neighborhood structures the sequence to be followed when examining them is of crucial importance for VND based approaches. Although there exist rules of thumb for ordering the neighborhoods [58], empirical tests in [64, 112, 107] revealed that dynamically chosen neighborhood orderings sometimes significantly improve the finally obtained solutions. Therefore, we investigated four neighborhood ordering strategies which are, however, associated with certain applications of the previously defined step functions. These combinations were identified based on preliminary tests:

Ordered: This is the classical neighborhood ordering strategy as described in [58], i.e., the neighborhoods are arranged according to increasing size and/or examination times. Although the asymptotic examination times are similar to each other in the worst case, the actual examination times experienced in practice are on average increasing for N_1 to N_6. Therefore, when using this neighborhood ordering, N_i is examined before N_{i+1} for $i = 1, \ldots, 5$. All neighborhoods are examined according to the RMC strategy.

Reversely Ordered: The contribution of neighborhoods with small indices is limited to resolving conflicts occurring for paper rolls quite recently assigned to storage locations. Since the available time for reallocations is rather short, it is very likely that conflicts for paper rolls assigned early will not be resolved during relocation phases. In addition, by first applying more time expensive rearrangements, it is possible to eliminate these conflicts. Therefore, this neighborhood ordering first examines neighborhood structure N_6 and continues with N_5, N_4, N_3, N_2 and N_1, respectively. Again, all neighborhoods are searched using the RMC step function.

Randomized: Based on the observation that the current warehouse state is permanently changing and therefore the type of arising conflicts is always in flux, a randomly chosen and constantly altering neighborhood ordering might be promising. Additionally, a variation in the utilized step function is performed such that there are nine different combinations of neighborhood structures and step functions. The application of the PRN strategy to neighborhood structures N_2, N_3 and N_6 will be denoted by N_7, N_8 and N_9, respectively, in the following. The next neighborhood N_i to be examined, with $i = 1, \ldots, 9$, is selected on a purely random basis each time a neighborhood examination is finished. In addition, a neighborhood structure is removed, i.e., no longer considered, if it did not yield a new improved warehouse state within its t last consecutive examinations. Based on preliminary tests the value of t is set to five for our application.

Dynamically Randomized: Analogously to the randomized ordering strategy the next neighborhood N_i to be examined, with $i = 1, \ldots, 9$, is chosen randomly when applying this ordering strategy. Again N_7 to N_9 denote the application of the PRN strategy to N_2, N_3 and N_6, respectively. The probabilities for selecting the neighborhoods, however, are adjusted each time a selection is performed. Detailed values for the applied probabilities will be given below. Analogously to the purely randomized ordering, neighborhood structures are removed as soon as t consecutive examinations did not provide improved warehouse states.

VND Framework Given the current warehouse state $\mathcal{W}$, a number of paper rolls to be relocated, and a preferred neighborhood ordering and examination strategy *strat*, Alg. 8 can be used for computing a list of paper roll relocations to be performed by warehousemen. After initializing all temporary variables, the main loop of the procedure is entered and is executed until either no more neighborhoods are left to be examined according to *strat* or the number of yet available paper roll movements is equal to or less than zero. In case of either the randomized or the dynamically randomized neighborhood ordering is chosen, this algorithm is repeatedly executed for r rounds. Although any arbitrary value could be chosen for the maximum number of rounds, $r = 50$ seems to be promising based on the observation that computation times on a standard PC for our paper production company are then acceptable, i.e., at most about three minutes. The list of rearrangements resulting in the best new warehouse state is then returned by the algorithm.

Depending on the value of parameter *strat* of Alg. 8 one or r iterations of the outer loop are performed. During one iteration the list of paper roll movements is, however, constantly lengthened, i.e., a once added move is never removed again. Obviously, this behavior corresponds to a greedy arrangement of warehouse operations and we think that this algorithmic design decision should be explained in detail. Based on an requirement analysis conducted in cooperation with the warehouse manager of our industry partner two fundamental requirements were identified: The reorganization of the warehouse might be interrupted at any time due to customers (unexpectedly) arriving for picking up their order. Therefore, each operation performed in the warehouse must guarantee that the warehouse state afterwards is better than before the movement of the corresponding rolls. However, each of the moves used for defining the neighborhood structures for our VND approach is assumed to be atomic, i.e., all corresponding reallocations will be completely performed before interrupting the reorganization. This also implies, however, that too complex reorganization steps are undesired. In addition, the algorithm is indented to be concurrently executed while the warehouse worker already apply the first proposed moves. Obviously, it is therefore not meaningful to withdraw already made decisions (especially if the corresponding paper roll relocations were already executed). Nevertheless, sometimes the first paper roll relocations are not immediately applied by the warehouse men which can be exploited by accordingly setting parameter *strat* of Alg. 8 such that r rounds are performed. In this case, the proposed heuristic is based on the same principles as a multistart heuristic like *greedy randomized adaptive search procedure* (GRASP) [117]. Nevertheless, be aware that in our case the starting point for the local search is always the current warehouse state, i.e., in contrast to GRASP the same solution. Furthermore, although we are interested in a warehouse state of minimal costs, i.e., having a minimum number of conflicts with respect to objective function (3.11), the output of the algorithm is a list of moves for reaching this state.

Algorithm 8: RelocationVariableNeighborhoodDescent($\mathcal{W}$, n_{m}, *strat*)

```
Input: $\mathcal{W}$ ...current warehouse state,
       $n_{\mathrm{m}}$ ...number of available paper roll movements,
       strat ...neighborhood ordering strategy and step function to be used
Data: $l$ ...index of currently examined neighborhood structure,
      $n', n''$ ...remaining number of available paper roll movements,
      $\mathcal{W}', \mathcal{W}''$ ...intermediate warehouse states,
      bestVal ...value of the best so far obtained warehouse state,
      bestL ...list of rearrangements for reaching the best so far obtained
      warehouse state
Output: L ...list of moves to be performed
bestL ← ();
bestVal ← $\infty$;
repeat
    L ← ();
    $\mathcal{W}' \leftarrow \mathcal{W}$;
    $n' \leftarrow n_{\mathrm{m}}$;
    repeat
        $l$ ← index of neighborhood to be examined next according to strat;
        $\mathcal{W}''$ ← examine neighborhood $N_l(\mathcal{W}')$ according to strat;
        if $\mathrm{E}(\mathcal{W}'') < \mathrm{E}(\mathcal{W}')$ then
            $n''$ ← $n'$−number of roll moves needed for obtaining $\mathcal{W}''$ from $\mathcal{W}'$;
            if $n'' \geq 0$ then
                add roll relocations for obtaining $\mathcal{W}''$ from $\mathcal{W}'$ to L;
                $\mathcal{W}' \leftarrow \mathcal{W}''$;
                $n' \leftarrow n''$;
    until no more neighborhoods left to be examined or $n' \leq 0$ ;
    if $\mathrm{E}(\mathcal{W}')$ < bestVal then
        bestVal ← $\mathrm{E}(\mathcal{W}')$;
        bestL ← L;
until until 1 or r repetitions are reached (depending on strat) ;
return L;
```

Table 3.1.: Selection probabilities of the neighborhoods in DRVND in dependence on the number of still available moves n_m.

n_m	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9
<20	0.50	0.20	0.10	0.05	0.03	0.02	0.04	0.04	0.02
20–100	0.35	0.15	0.10	0.10	0.02	0.15	0.05	0.05	0.03
>100	0.05	0.05	0.05	0.20	0.15	0.30	0.05	0.05	0.10

The following four different variants of VND were implemented and compared with each other (for test results see Sec. 3.1.6):

Ordered VND (OVND): For this variant of VND the parameter *strat* is set to ordered (neighborhood order is fixed and RMC strategy is used). Since the algorithm is deterministic only one round is performed.

Reversely Ordered VND (ROVND): While the number of still available moves n_m is greater than or equal to 80, this algorithm uses a fixed reversely ordered neighborhood ordering. As soon as n_m falls below a given value, the order is reversed, i.e., the same order as for OVND is used. In our setting, this value is set to 80 which was identified in preliminary tests based on the typical characteristics of customer orders. In contrast to OVND, this setting tries to first apply more complex rearrangement to the warehouse such that large improvements can be achieved. However, as soon as the number of still available moves gets closer to zero the degree of freedom with respect to rearrangements also decreases. It is therefore more likely that further improvements are obtained using short rearrangements, i.e., relocations with only a few paper roll movements. Analogously to OVND only one round needs to be performed in Alg. 8 due to the deterministic nature of this setting.

Randomized VND (RVND): For this setting the randomized neighborhood ordering is selected. Therefore, N_1 to N_6 are using a RMC strategy and N_7 to N_9 are searched by the PRN step function. Hence, r rounds are performed in the algorithm and the list of relocation moves resulting in the best warehouse state is returned.

Dynamically Randomized VND (DRVND): This variant applies dynamically randomized neighborhood ordering. The concrete probability values are shown in Tab. 3.1. Each time, the number of still available moves n_m falls below a value indicated in the column labeled n_m, the probabilities for selecting neighborhoods N_1 to N_9 are adjusted to the values shown in the corresponding columns. These values were identified during preliminary tests and were then refined with the help of the warehouse manager of our industry partner. Again, the basic concept relies on the observation that as long as the number of still available moves is relatively high, the exploration of neighborhoods based on the more complex neighborhood structures seems to be more promising. However, if only a view moves are left, the likelihood for an improvement based on these complex neighborhood structures decreases. Further, it would also be imaginable that self-adaptive neighborhood orderings are applied as presented in [64, 112, 107]. These methods are, however, proposed for applications were the number of still available moves in not limited. Furthermore, these methods always rest upon the improvement potential, cf. [112], or the number of improvements (and examination times) in the past, cf. [64, 107]. In our case these measures are, however, not reasonable due to the limitation on available moves.

3.1.6. Experimental Results

In our paper production company the stocking strategy presented in Sec. 3.1.4 is already applied in practice. Comparing warehouse states previously obtained by the old stocking strategy, which was mainly based on the experience of the warehousemen as well as the warehouse manager with warehouse states obtained after using the here proposed stocking strategy, it can be clearly seen that the situation in the warehouse significantly improved and therefore the time needed for shipping is reduced by a vast amount. Unfortunately, it is not possible to directly compare the old and the new stocking strategy with each other during real time operations. Therefore, we decided to simulate the stocking of typically produced paper rolls using the old and the new strategy.

Using this simulation based data it is possible to compare the efficiency and contributions of the relocation strategies proposed within this thesis. In fact, the main parameter of a typical warehouse state is the number of rolls stored within the warehouse. In our case at most 4400 paper rolls can be stored in the warehouse. Expert knowledge indicates that a filling level of 80% constitutes the critical level for which any further stocked paper rolls will almost always cause conflicts—even in case of optimal placement. Therefore we tested our relocation approaches on warehouse states with 2500, 3000 and 3500 paper rolls stocked. All computations were performed on a single core of a Dual Opteron processor with 2.4GHz and 4GB of RAM. As underlying database storing all production and order relevant data an Oracle 10i database was used. Although, the number of rolls to be relocated, can be chosen arbitrarily, we tested our algorithm for 10, 20, 50, 100, 300, 500 and 700 relocation moves, which corresponds to approximately 10, 25 and 60 minutes as well as 2, 6, 8 and 12 hours of working time. Although it is relatively rare that one worker might relocate all day long this might occur on weekends when only few new paper rolls are produced and the driving of trucks on highways is prohibited, which is law in some European countries.

Table 3.2.: Absolute values of E($\mathcal{W}$) for six different test instances. The values represent the objective values for the warehouse states obtained by a simulated human stocking strategy and our stocking strategy proposed in Section 3.1.4. In the last column a lower bound on the objective value for the warehouse states is given.

	simulated stocking	stocking strategy		sorted stocking
		avg.	std	
w_1	102635.0	12819.9	339.1	11807.0
w_2	130856.0	14683.0	413.0	13778.0
w_3	186835.0	23047.1	1101.4	18244.0
w_4	135203.0	17290.2	1098.9	14012.0
w_5	300881.0	49481.0	3018.7	35850.0
w_6	181877.0	64687.3	2525.2	33080.0

For testing purposes we have chosen six exemplary production data sets called w_1 to w_6 which were provided by our industry partners. The limiting factor for our stocking strategy as well as the VND approach are the number of paper rolls to be stored in the warehouse as preliminary tests revealed. While warehouses w_1 and w_2 contain 2500 paper rolls to be stored, warehouses w_3 and w_4 consist of 3000 paper rolls. Finally, w_5 and w_6 contain 3500 rolls.

For evaluating the performance of the stocking strategy we compared three different stocking approaches. The first one corresponds to a simulation of the stocking strategy used in our paper production company during the last years. This strategy is mainly based on the experience of the warehouse operator and does not provide any type of forecast—neither with respect to the shipping dates nor regarding paper rolls to be produced in future. The corresponding objective values with respect to Eq. (3.8) are given in the first data column of Tab. 3.2. The second column of this table lists the mean results over 20 runs using slightly different production sequences of paper rolls obtained by our stocking strategy including standard deviations. The final column lists values obtained by first sorting all ordered rolls according to their shipping date and customer order and then stocking them using our stocking strategy. It can be clearly seen, that our stocking strategy outperforms the formerly used strategy. It has to be emphasized that an optimal warehouse state will almost never be reached in this real world application as long as a last-in, first-out throughput policy is applied and the production process is optimized disregarding the storage structure.

Anyhow, it is still necessary to perform relocations from time to time since the shipping dates are often not met by the customers. These conflicts cannot be foreseen even by the best stocking strategy. Therefore, we did experiments using the proposed relocation

Table 3.3.: For different number of relocation moves the relative values of the finally obtained warehouse states based on those obtained via the formerly used stocking strategy are presented. Mean values are averages over 20 runs (with standard deviations in parentheses). The last column presents p-values of Wilcoxon rank sum tests for the hypothesis that the mean values of DRVND are better than those of RVND.

		GRP	OVND	ROVND	RVND			DRVND			p-val
					best	mean	std	best	mean	std	
$n_\mathrm{m} = 10$	w_1	97.2%	97.2%	97.2%	97.3%	97.7%	(0.3%)	97.1%	97.2%	(0.0%)	<0.01
	w_2	96.2%	96.2%	96.2%	96.2%	96.6%	(0.3%)	96.2%	96.2%	(0.0%)	<0.01
	w_3	98.9%	98.9%	98.9%	98.9%	99.0%	(0.1%)	98.6%	98.7%	(0.1%)	<0.01
	w_4	98.8%	98.8%	98.8%	98.6%	98.9%	(0.1%)	98.6%	98.8%	(0.0%)	<0.01
	w_5	97.7%	97.7%	97.7%	99.2%	99.4%	(0.2%)	97.7%	97.7%	(0.0%)	<0.01
	w_6	98.8%	98.8%	98.9%	98.8%	99.0%	(0.2%)	98.7%	98.8%	(0.0%)	<0.01
$n_\mathrm{m} = 20$	w_1	95.0%	95.0%	95.0%	95.8%	96.3%	(0.3%)	94.7%	95.0%	(0.2%)	<0.01
	w_2	92.9%	92.9%	92.9%	94.3%	94.7%	(0.2%)	92.9%	92.9%	(0.1%)	<0.01
	w_3	97.9%	97.9%	97.9%	97.5%	98.2%	(0.2%)	97.6%	97.8%	(0.1%)	<0.01
	w_4	98.4%	98.4%	98.4%	97.8%	98.1%	(0.2%)	97.6%	97.8%	(0.1%)	<0.01
	w_5	96.7%	96.7%	96.7%	97.5%	98.4%	(0.3%)	96.3%	96.7%	(0.2%)	<0.01
	w_6	98.0%	98.0%	98.1%	97.5%	98.2%	(0.3%)	97.0%	97.8%	(0.3%)	<0.01
$n_\mathrm{m} = 50$	w_1	89.6%	89.6%	89.6%	92.0%	92.3%	(0.1%)	89.4%	90.0%	(0.5%)	<0.01
	w_2	87.6%	87.6%	87.6%	89.4%	89.9%	(0.3%)	86.5%	87.6%	(0.7%)	<0.01
	w_3	94.7%	94.7%	94.7%	95.1%	96.1%	(0.4%)	94.2%	94.7%	(0.3%)	<0.01
	w_4	97.1%	97.1%	97.1%	95.0%	96.5%	(0.6%)	94.4%	95.8%	(0.8%)	<0.01
	w_5	93.5%	93.5%	93.5%	94.4%	96.0%	(0.4%)	92.7%	93.5%	(0.7%)	<0.01
	w_6	97.6%	96.2%	95.8%	95.6%	95.8%	(0.2%)	94.9%	95.3%	(0.3%)	<0.01
$n_\mathrm{m} = 100$	w_1	82.7%	82.7%	86.5%	86.6%	88.1%	(0.7%)	82.4%	83.3%	(0.8%)	<0.01
	w_2	84.9%	82.4%	83.1%	82.8%	84.4%	(0.8%)	81.1%	82.5%	(0.9%)	<0.01
	w_3	92.7%	92.3%	93.7%	91.8%	92.2%	(0.3%)	91.1%	92.0%	(0.3%)	0.03
	w_4	97.0%	95.8%	91.9%	89.8%	90.7%	(0.4%)	89.8%	90.2%	(0.3%)	<0.01
	w_5	90.4%	90.4%	91.7%	90.2%	90.7%	(0.2%)	89.5%	90.3%	(0.3%)	<0.01
	w_6	97.6%	93.4%	94.5%	92.7%	93.6%	(0.4%)	92.0%	92.5%	(0.5%)	<0.01
$n_\mathrm{m} = 300$	w_1	77.3%	67.7%	69.1%	71.0%	73.4%	(1.4%)	64.0%	68.6%	(3.1%)	<0.01
	w_2	84.9%	68.4%	66.4%	67.1%	68.9%	(1.0%)	65.0%	66.6%	(1.0%)	<0.01
	w_3	92.7%	85.6%	83.8%	83.3%	84.2%	(0.9%)	80.2%	82.5%	(1.3%)	<0.01
	w_4	97.0%	95.8%	84.5%	76.9%	79.9%	(1.5%)	73.2%	78.3%	(3.0%)	0.01
	w_5	90.4%	85.4%	81.0%	77.7%	78.9%	(0.7%)	77.7%	78.8%	(0.8%)	0.31
	w_6	97.6%	92.0%	95.5%	83.8%	84.9%	(0.7%)	83.4%	84.7%	(1.0%)	0.21
$n_\mathrm{m} = 500$	w_1	77.3%	63.5%	61.2%	63.0%	65.7%	(1.7%)	57.3%	59.8%	(2.1%)	<0.01
	w_2	84.9%	52.7%	51.5%	53.5%	55.3%	(1.9%)	50.4%	52.0%	(1.4%)	<0.01
	w_3	92.7%	79.1%	76.7%	74.8%	76.3%	(1.0%)	71.6%	73.6%	(1.3%)	<0.01
	w_4	97.0%	96.5%	65.7%	66.4%	67.8%	(1.3%)	64.0%	66.1%	(1.6%)	<0.01
	w_5	90.4%	77.7%	72.8%	68.9%	70.7%	(1.2%)	68.3%	70.4%	(1.2%)	0.10
	w_6	97.6%	93.4%	96.0%	76.1%	79.0%	(2.1%)	74.7%	77.2%	(1.6%)	<0.01
$n_\mathrm{m} = 700$	w_1	77.3%	59.1%	50.6%	55.3%	57.7%	(1.9%)	50.0%	51.2%	(1.3%)	<0.01
	w_2	84.9%	42.0%	43.0%	41.4%	44.0%	(2.2%)	40.4%	42.0%	(2.3%)	<0.01
	w_3	92.7%	74.8%	66.8%	64.4%	66.4%	(1.5%)	60.3%	63.2%	(2.1%)	<0.01
	w_4	97.0%	95.8%	60.0%	57.6%	58.9%	(0.9%)	53.5%	56.5%	(2.0%)	<0.01
	w_5	90.4%	76.2%	72.9%	62.5%	64.8%	(1.5%)	62.5%	64.1%	(1.4%)	0.01
	w_6	97.6%	92.8%	96.0%	68.8%	72.7%	(3.0%)	68.5%	71.2%	(1.9%)	<0.01

Table 3.4.: For different number of relocation moves the relative values of the finally obtained warehouse states obtained via the proposed stocking strategy are presented. Mean values are averages over 40 runs (with standard deviations in parentheses). The last column presents p-values of Wilcoxon rank sum tests for the hypothesis that the mean values of DRVND are better than those of RVND.

		GRP	OVND	ROVND	RVND			DRVND			p-val
					best	mean	std	best	mean	std	
$n_\mathrm{m} = 10$	w_1	100.0%	99.5%	99.4%	97.9%	98.7%	(0.2%)	97.8%	98.3%	(0.1%)	<0.01
	w_2	100.0%	99.8%	99.8%	98.2%	99.2%	(0.1%)	97.7%	98.8%	(0.1%)	<0.01
	w_3	100.0%	99.4%	99.3%	97.8%	98.8%	(0.1%)	97.8%	98.6%	(0.1%)	0.11
	w_4	99.9%	99.8%	99.8%	98.0%	98.7%	(0.1%)	97.9%	98.5%	(0.1%)	<0.01
	w_5	99.8%	99.6%	99.6%	98.1%	99.2%	(0.1%)	98.2%	99.2%	(0.1%)	0.48
	w_6	99.9%	99.5%	99.5%	98.3%	98.8%	(0.1%)	97.8%	99.0%	(0.2%)	0.93
$n_\mathrm{m} = 20$	w_1	100.0%	99.4%	99.3%	97.3%	98.2%	(0.1%)	97.2%	97.8%	(0.1%)	<0.01
	w_2	100.0%	99.7%	99.6%	98.0%	98.8%	(0.1%)	97.7%	98.4%	(0.1%)	<0.01
	w_3	100.0%	99.1%	99.0%	96.3%	98.4%	(0.1%)	97.0%	98.1%	(0.1%)	0.04
	w_4	99.9%	99.8%	99.8%	95.5%	98.1%	(0.3%)	61.9%	96.3%	(1.2%)	<0.01
	w_5	99.8%	99.4%	99.3%	96.6%	97.8%	(0.3%)	96.4%	97.9%	(0.2%)	0.67
	w_6	99.9%	99.5%	99.5%	96.5%	97.9%	(0.1%)	96.7%	98.2%	(0.2%)	1.00
$n_\mathrm{m} = 50$	w_1	100.0%	98.9%	98.7%	95.2%	96.9%	(0.1%)	95.0%	96.2%	(0.1%)	<0.01
	w_2	100.0%	98.5%	98.5%	96.0%	97.0%	(0.1%)	95.6%	96.7%	(0.1%)	<0.01
	w_3	100.0%	99.1%	99.1%	95.3%	97.9%	(0.2%)	93.9%	97.1%	(0.3%)	<0.01
	w_4	99.9%	99.8%	99.8%	92.7%	96.8%	(0.2%)	68.8%	93.3%	(1.2%)	<0.01
	w_5	99.8%	99.3%	99.3%	92.5%	94.9%	(0.3%)	93.3%	95.3%	(0.3%)	0.99
	w_6	99.9%	99.1%	99.2%	93.5%	96.0%	(0.3%)	93.5%	96.2%	(0.4%)	0.85
$n_\mathrm{m} = 100$	w_1	100.0%	98.3%	98.1%	92.7%	96.0%	(0.3%)	68.0%	92.3%	(1.9%)	<0.01
	w_2	100.0%	99.7%	99.7%	93.4%	96.6%	(0.3%)	93.2%	95.6%	(0.2%)	<0.01
	w_3	100.0%	99.1%	99.1%	92.3%	97.5%	(0.4%)	92.4%	96.1%	(0.5%)	<0.01
	w_4	99.9%	99.8%	99.9%	92.5%	96.9%	(0.4%)	69.2%	93.5%	(1.3%)	<0.01
	w_5	99.8%	99.3%	98.3%	87.6%	92.0%	(0.7%)	86.7%	91.9%	(0.7%)	0.46
	w_6	99.9%	99.0%	99.1%	89.1%	94.4%	(1.3%)	90.1%	93.7%	(0.5%)	0.25
$n_\mathrm{m} = 300$	w_1	100.0%	99.2%	99.2%	93.7%	96.2%	(0.3%)	84.8%	93.1%	(0.7%)	<0.01
	w_2	100.0%	99.7%	99.7%	93.6%	97.0%	(0.3%)	91.8%	95.3%	(0.5%)	<0.01
	w_3	100.0%	99.1%	99.1%	92.8%	97.3%	(0.4%)	90.9%	96.0%	(0.5%)	<0.01
	w_4	99.9%	99.8%	99.9%	91.3%	96.9%	(0.4%)	86.8%	93.8%	(0.7%)	<0.01
	w_5	99.8%	99.2%	97.8%	82.5%	91.1%	(0.9%)	81.3%	88.7%	(0.6%)	<0.01
	w_6	99.9%	99.0%	98.4%	81.7%	91.2%	(1.9%)	73.8%	86.4%	(2.3%)	<0.01
$n_\mathrm{m} = 500$	w_1	100.0%	99.2%	99.2%	93.3%	96.2%	(0.3%)	85.1%	92.9%	(0.7%)	<0.01
	w_2	100.0%	99.7%	99.7%	93.9%	97.0%	(0.3%)	92.5%	95.3%	(0.5%)	<0.01
	w_3	100.0%	99.1%	99.1%	92.5%	97.6%	(0.3%)	92.1%	95.9%	(0.3%)	<0.01
	w_4	99.9%	99.8%	99.9%	93.5%	97.0%	(0.5%)	70.4%	93.8%	(1.2%)	<0.01
	w_5	99.8%	99.2%	97.8%	83.2%	90.7%	(1.0%)	80.1%	88.4%	(0.8%)	<0.01
	w_6	99.9%	99.0%	98.4%	73.2%	90.6%	(1.9%)	66.4%	83.3%	(3.2%)	<0.01
$n_\mathrm{m} = 700$	w_1	100.0%	99.2%	99.2%	93.2%	96.0%	(0.3%)	86.9%	92.8%	(0.5%)	<0.01
	w_2	100.0%	99.7%	99.7%	93.4%	96.8%	(0.2%)	91.2%	95.2%	(0.4%)	<0.01
	w_3	100.0%	99.1%	99.1%	94.2%	97.7%	(0.2%)	90.4%	95.8%	(0.3%)	<0.01
	w_4	99.9%	99.8%	99.9%	93.4%	97.1%	(0.4%)	73.1%	93.7%	(0.9%)	<0.01
	w_5	99.8%	99.2%	97.8%	82.6%	91.1%	(0.7%)	78.3%	88.4%	(1.1%)	<0.01
	w_6	99.9%	99.0%	98.4%	76.7%	91.2%	(2.1%)	68.6%	83.5%	(4.0%)	<0.01

strategies. For the VND based approach we set the values of the parameters t and r based on preliminary tests to $t = 5$, i.e., the number of allowed unsuccessful exploration of a neighborhood until it is removed, and $r = 50$, i.e., the number of rounds performed for the multistart variant of our approach. While Tab. 3.3 represents values obtained for applying the relocation strategies on warehouse states generated by the formerly used stocking strategy, the values presented in Tab. 3.4 correspond to results obtained by reassigning paper rolls in warehouses obtained by our stocking strategy. A value of 98% indicates that an improvement of two percent could be achieved, i.e., the estimated probability of conflicts during removal operations with respect to Eq. (3.8) could be reduced by 2%.

The following trend can be recognized: the more available time is dedicated to relocation operations the better the obtained warehouse states become. In addition the performance of GRP seems to be poorer for a larger number of available moves than those of the different VND variants. Although the former tendency is obvious, GRP could not improve further with more than 100 moves for warehouse states considered in Tab. 3.3. It is most interesting that this behavior seems to be independent of the number of rolls stored in the warehouse. To confirm this observation, we performed additional tests investigating especially this fact. A possible explanation for this could be that within our warehouse only 175 strips exist, such that only a few conflicts can be resolved when considering always the paper rolls at the front of each strip only. A limitation of GRP to at most 100 moves seems, however, reasonable for an application in our paper production company. When applying GRP on warehouse states obtained via our stocking strategy, the improvement potential is rather limited.

Regarding the performances of our VND variants it turned out that DRVND seems to be the best VND setting for relocations consisting of many paper roll movements. If the available time is rather limited the performances of RVND, OVND and ROVND are similar. RVND, however, is outperformed almost always by DRVND. To validate this hypothesis we performed Wilcoxon rank sum tests. The resulting p-values are for nearly all tested instances below 0.01, which indicates that the assumption is in most cases correct with an error probability of at most one percent. For those warehouse states obtained via our stocking strategy an improvement could still be achieved using the VND variants which implies that conflicts induced by improper production sequences have an impact on the stocking strategy. Anyhow, it is important to assign the rolls to good storage locations from the beginning on, since the results obtained by the new stocking strategy could not be reached by the relocation procedure applied to warehouse states resulting from the formerly used stocking strategy.

With respect to runtime, GRP is the fastest approach with runtimes of at most 5 seconds. DRVND, which is the slowest VND variant, needs for computing 700 paper roll movements about 3.5 minutes, which is reasonable according to the warehouse manager of our industry partner. Finally, we investigated the number of times one paper roll is relocated during storage reassignments. We observed that even for the test runs including 700 moves, multiple moves of individual paper rolls seldom occur.

3.2. Routing

Although the computation of optimal storage locations for articles might increase the quality of a storage management system while at the same time decreasing the arising costs, there are situations for which an optimal assignment of articles to storage locations cannot be computed. One of these situations is a spare parts warehouse. In contrast to branches of industry like paper production spare parts supplier cannot produce ordered articles on customer demand for reducing overall delivery times. Since the concrete customer orders are in general unknown at the time of stocking the computation of storage locations need to be performed independently. In most cases, statistics and expert knowledge are used for determining optimal storage locations. In addition characteristics of the articles to be stored like size and weight are considered. Nevertheless, to reduce the overall response times to customer orders it is necessary to compute as short picking tours as possible, which are then processed by warehousemen.

In addition, spare parts suppliers are confronted with several problems. On the one hand, they should be able to supply spare parts both on demand and as fast as possible. On the other hand, they have to keep their storage as small as possible for various economic reasons. Storage space itself is expensive, but more importantly by adding additional capacity to the stock, the complexity of administration increases substantially. Therefore, the demand for (semi-)automatic warehouse management systems arises. Beside keeping computerized inventory lists additional planning tasks can be transferred to the computer system. For example, lists containing all articles to be reordered can be automatically generated.

Obviously the main task to be performed within a spare parts warehouse is the issuing and shipping of items ordered by customers. For this purpose, several warehousemen traverse the storage and collect ordered articles which will then be brought to a packing station where all items are boxed and shipped for each customer. Of course, the possible savings related with minimizing collecting times of items are high and therefore effort should be put into a proper tour planning. Various constraints related to capacities of trolleys used for transporting collected articles, structural conditions of the warehouse and delivery times guaranteed to the customers have to be considered. Nevertheless such a tour planning system can only provide a suggestion of tours through the warehouse since the final decisions must always be made by humans. For example it may happen that some routes through the warehouse are unpredictable blocked due to the breakdown

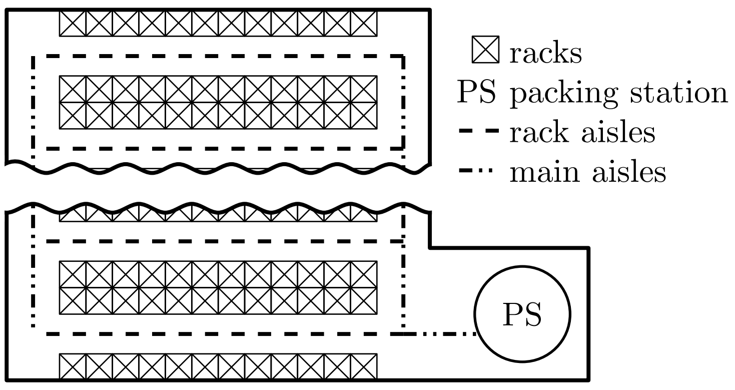


Figure 3.2.: An exemplary storage layout. Main aisles (vertical in this sketch) and rack aisles (horizontally aligned) are joining each other orthogonally.

of a trolley used by another worker. In this work we assume, however, that such situations only rarely occur.

The rest of this section is organized as follows: The next section gives a detailed problem definition. In Sec. 3.2.2 we present an overview of related work. A new hybrid approach based on *variable neighborhood search* and *dynamic programming* is presented in Sec. 3.2.3. Experimental results complete this section.

3.2.1. Problem Definition

The problem can be defined as follows: We are given a warehouse with a storage layout similar to that presented in Fig. 3.2, i.e., several racks are aligned such that two types of aisles arise: *rack aisles* and *main aisles*, whereas we assume that there are two main aisles with an arbitrary number of rack aisles lying between them. While rack aisles provide access to the racks main aisles only act as an interconnection between the rack aisles and the packing station. We denote by $\mathcal{R} = \{1, \ldots, n_{\mathrm{R}}\}$ the set of racks located in the rack aisles.

In addition, a set of articles $\mathcal{A}$, with $\mathcal{A} = \{1, \ldots, n_{\mathrm{A}}\}$, $n_{\mathrm{A}} \geq 1$, is given. Each article $a \in \mathcal{A}$ is stored at a non-empty set R_a of one or more racks, i.e., $\emptyset \neq R_a \subseteq \mathcal{R}$. The quantities of articles $a \in \mathcal{A}$ are given by $q_a : R_a \to \mathbb{N}$.

We assume that a homogeneous fleet of n_{T} trolleys used for carrying collected items exists, each having capacity γ. A group of n_{W} warehousemen, $1 \leq n_{\mathrm{W}} \leq n_{\mathrm{T}}$, is operating these trolleys and issuing ordered articles.

A set of customer orders is given with each order consisting of a list of articles with demands to be shipped to a specific address. Further, a global latest delivery time L is defined which states that all customer orders need to be shipped until that time. Although the assignment of orders to customers is important for a production system, we are only interested in the quantities of each article to be collected in the warehouse for this work, since extra workers are assigned to pack all items according to orders. Therefore, we define the set $\mathcal{O}$ of orders as the set of tuples $(a, d_a) \in \mathcal{O}$, with $|\mathcal{O}| = n_\mathrm{O}$, stating the total integer demand $d_a \geq 1$ of each article $a \in \mathcal{A}$. In addition, we assume that all articles have uniform size and the capacities of all trolleys together, i.e., $n_\mathrm{T} \cdot \gamma$, is greater than the amount of articles to be collected. However, we do not require that each order fits into one trolley. Even more, we expect that in an optimal solution multiple orders will be carried by one trolley. In addition, one order can be split over multiple trolleys.

Let us denote by set $\mathcal{S}$ a finite set of selections, whereas a selection $S \in \mathcal{S}$ is a set of triples (a, δ, r) such that $r \in R_a$, $1 \leq \delta \leq q_a(l)$, and there exists an order $(a, d_a) \in \mathcal{O}$ with $d_a \geq \delta$. Further, we denote by $\mathcal{T}$ a set of tours whereas for each $S_i \in \mathcal{S}$ a tour $T_i \in \mathcal{T}$ exists. By tour T_i we understand a walk through the warehouse visiting all locations contained in S_i such that the corresponding items of S_i can be collected. The length of tour $T_i \in \mathcal{T}$ is denoted by $c(T_i)$.

A solution $x = (\mathcal{S}, \mathcal{T}, \Pi, <_\mathcal{T})$ to the given problem consists of a set $\mathcal{T}$ of tours corresponding to the selections in $\mathcal{S}$ as well as a mapping $\Pi : \mathcal{T} \to \{1, \ldots, n_\mathrm{W}\}$ of tours to workers and an ordering $<_\mathcal{T}$ of tours such that

- $|\mathcal{S}| = |\mathcal{T}| \leq n_\mathrm{T}$,
- tour $T_i \in \mathcal{T}$ collects all articles $a \in S_i$, $S_i \in \mathcal{S}$,
- $\sum_{S \in \mathcal{S}} \sum_{(a,\delta,l) \in S} \delta = d_a$, for all $a \in \mathcal{A}$,
- worker $\Pi(T_i)$ processes tour $T_i \in \mathcal{T}$,
- tour T_i is processed before tour T_j if $T_i <_\mathcal{T} T_j$ for all $T_i, T_j \in \mathcal{T}$ with $\Pi(T_i) = \Pi(T_j)$,
- all time constraints are met, i.e., for each worker $1 \leq k \leq n_\mathrm{W}$ it has to be guaranteed that his/her last tour is finished before the global time limit L. Therefore, $\sum_{T | \Pi(T) = k} c(T) \leq L$ must hold for all workers $1 \leq k \leq n_\mathrm{W}$.

We formulate the given problem as an optimization problem in which the total length of the tours, i.e., $\sum_{T \in \mathcal{T}} c(T)$, as well as the violations of the capacity constraints defined by the trolleys, i.e., $\sum_{S \in \mathcal{S}} \max\left\{\sum_{(a,\delta,l) \in S} \delta - \gamma, 0\right\}$, should be minimized. For weighting the relative importance of violating capacity constraints compared to tour lengths, we introduce a weighting coefficient ω such that the objective function can be written as

$$\min \sum_{T \in \mathcal{T}} c(T) + \omega \cdot \sum_{S \in \mathcal{S}} \max \left\{ \sum_{(a,\delta,l) \in S} \delta - \gamma, 0 \right\} \tag{3.11}$$

For this work, ω is set to the maximum possible length of one tour picking up one article plus one. Such a choice for ω implies that it is always better to use an additional tour for collecting an item than violating a capacity constraint. Let us note that, although violations of capacities are allowed with respect to the objective function, a finally obtained solution must not contain overloaded trolleys. Due to the relatively high weighting factor ω, such solutions are, however, only considered during the starting phase of the proposed algorithms while generating a first valid solution.

Although not mentioned so far, there is a further constraint which is not directly regarded within this work: For security reasons it is demanded by the warehouse manager that no two workers are working at the same time in the same area of rack aisles. It is, however, almost impossible to compute the exact times when warehouse men are in specific aisles, e.g., worker may occasionally stop to go to the restroom or talk to other people. Nevertheless, based on preliminary tests it turned out that optimal solutions normally do not include many different tours entering the same aisle. In most cases, even only one worker needs to enter an aisle and collects all articles ordered during one traversal.

3.2.2. Related Work

Obviously, the stated problem is related to warehouse management in general. An introduction to this topic as well as an overview over tour finding, storage management and other related tasks is given in [26].

It is obvious that the stated problem forms a special variant of the well known *vehicle routing problem* (VRP) [130]. In fact, the classical VRP is extended by additional domain specific constraints. In the classical VRP one wants to find a set of tours minimal with respect to their total length starting at a depot and visiting a predefined set of customers. Further, the problem studied here is related to the *split delivery VRP* [35], the *VRP with time windows* [126] and the *capacitated VRP* [115]. To our knowledge, there exists only few previous work (e.g. [40]) considering a combination of all three of these variants of VRP. A further combination with the constraints considered in this work is, however, to our knowledge at this time untried. However, due to this strong relationship to VRP, it is obvious that the problem investigated within this section is $\mathcal{NP}$-hard. Another related routing problem arises in container terminals [83] where *automatic guided vehicles* (AGVs) are utilized for transporting containers between storing locations and vessels.

Beside this obvious relationship with VRPs, this problem is also related to the *generalized network design problems* [38] with respect to the possibility to collect one article from different locations within the warehouse. At a time only one node of such a cluster has to be visited.

3.2.3. A Hybrid Variable Neighborhood Search Approach

Based on the fact that the problem examined within this paper is strongly related to the VRP, we expect that exact approaches are limited to relatively small instances. In addition, short computation times are important, since the observation was made that new orders are committed continuously by customers which implies that the algorithm is restarted frequently. Since recently highly effective *variable neighborhood search* (VNS) [58] approaches have been reported for diverse variants of the VRP [60, 98] we also based our approach on a similar concept. Within our hybrid VNS, *variable neighborhood descent* (VND) [58] is used as embedded local search procedure, and subproblems corresponding to the computation of individual tours for collecting particular items are solved exactly by means of dynamic programming [13], exploiting the specific structure of the warehouse.

The Basic Principle

In this work we assume that $\sum_{a \in \mathcal{A}} d_a \leq n_\mathrm{T} \cdot \gamma$, i.e., the total capacities of all trolleys is greater than or at least equal to the total amount of ordered articles. (Just as a reminder: we assume that all articles are equally sized.) Anyhow, in real-world settings the problem may arise that these constraints cannot be satisfied. In that case a straightforward preprocessing step is used, which partitions the set of orders such that for each set the capacity constraints are satisfied. Each of these sets is then independently solved using the proposed approach.

The tour planning algorithm mainly consists of two parts: (1) the allocation of articles to at most n_T selections and (2) the computation of concrete routes through the warehouse for collecting all items assigned to the previously determined selections. Anyhow, both of these parts have to be executed intertwined, since the evaluation of the mapping of articles to tours is based on the lengths of these tours. Therefore, these two steps are repeated until no further improvement can be achieved. Finally, an assignment of the walks to n_W warehousemen is done, such that the latest finishing time is as early as possible and the global delivery time is respected.

In real-world settings it might happen that additional orders will be committed by customers. In this situation the algorithms needs to be restarted from the beginning. However, it is straightforward to extend this approach by an incremental update function, such that already computed solutions can be expanded to valid solutions regarding the additional orders.

Assignment of Articles to Tours

One crucial point of our algorithm is the assignment of articles to selections such that in a second step walks through the warehouse can be computed. Nevertheless, the capacity constraints stated by the trolleys as well as the maximum number of available trolleys n_{T} have to be regarded during this allocation step.

Construction Heuristic For quickly initializing our algorithm we developed a construction method called *collision avoiding heuristic* (CAH). The main idea of CAH is to divide the storage into $m \geq 1$ physically non-overlapping zones whereupon each one is operated by one trolley, i.e., m selections are generated. For this work, we set m to n_{W}.

Since the capacities of the trolleys are not regarded within this initialization procedure the solution qualities produced by this heuristic are not outstanding. The required computation times are, however, very low. Therefore, CAH can be used for providing *ad hoc* solutions such that the workers start collecting the first scheduled item while the rest of the tours is improved in the meantime.

Improvement Heuristic For improving solutions generated by CAH, we present a *variable neighborhood search* (VNS) approach using an adapted version of *variable neighborhood descent* (VND) as subordinate. The basic idea of VNS/VND is to systematically switch between different neighborhood structures until no further improvement can be achieved. In fact, the crucial task in designing such an approach is the proper definition of appropriate moves used for defining the neighborhood structures incorporated in VNS and VND, respectively. The following seven different move types were implemented:

BreakTour(i) Selection $S_i \in \mathcal{S}$ is removed from $\mathcal{S}$ and all articles assigned to S_i are randomly distributed over all other selections $S_j \in \mathcal{S} \setminus S_i$.

MergeTour(i, j) Selections $S_i \in \mathcal{S}$ and $S_j \in \mathcal{S}$ are both removed from $\mathcal{S}$ and merged with each other into a new selection $S_{i'}$, which is then added to $\mathcal{S}$.

ShiftArticle(i, j, a) Any solution generated by this move differs from the underlying solution in one article a which is moved from selection S_i to selection S_j, with $a \in S_i$ and $S_i, S_j \in \mathcal{S}$.

ShiftArticleChangeRack(i, j, a, r) Analogously to the ShiftArticle move, this move shifts an article $a \in S_i$ to selection S_j, with $S_i, S_j \in \mathcal{S}$. In addition to this, a is now collected from rack $r \in R_a$ regardless of the position it was acquired before.

SplitTour(i) By applying this move, selection $S_i \in \mathcal{S}$ is split into two new selections $S_{i'}$ and $S_{i''}$ such that $|S_{i'}| = |S_{i''}|$ or $|S_{i'}| = |S_{i''}| + 1$. Selection S_i is removed from $\mathcal{S}$, whereas $S_{i'}$ and $S_{i''}$ are added.

SwapArticle(i, j, a_1, a_2) This move swaps two articles a_1 and a_2 , with $S_i, S_j \in \mathcal{S}$ and $a_1 \in S_i$, $a_2 \in S_j$.

SwapArticleChangeRack(i, j, a_1, a_2, r_1, r_2) This move is very similar to the SwapArticle move. After swapping articles $a_1 \in S_i$ and $a_2 \in S_j$ between selections $S_i \in \mathcal{S}$ and $S_j \in \mathcal{S}$, the rack of a_1 is changed to $r_1 \in R_{a_1}$ and that of a_2 is changed to $r_2 \in R_{a_2}$.

Based on these move types, the neighborhood structures for VNS and VND are defined, whereas the neighborhoods $N_1(x), \dots, N_{k_{\max}}(x)$, with $1 \leq k_{\max} \leq |\mathcal{S}| - 1$, used within the shaking phase of VNS are purely based on BreakTour moves such that within $N_k(x)$, with $1 \leq k \leq k_{\max}$, k randomly chosen BreakTour moves are applied to x. The neighborhood structures $\mathcal{N}_1, \dots, \mathcal{N}_6$ for VND are defined by a single application of one of the other six move types, such that $\mathcal{N}_1, \dots, \mathcal{N}_6$ apply SplitTour, MergeTour, ShiftArticle, ShiftArticleChangeRack, SwapArticle, SwapArticleChangeRack, respectively.

In addition to the proper definition of neighborhood structures, a beneficial order used for systematically examining them is necessary and has a great influence on the performance of VND (cf. [64, 111]). Preliminary tests showed that the contributions of neighborhood structures $\mathcal{N}_1$ and $\mathcal{N}_2$ are relatively high during the beginning of VND but dramatically decrease after only a few iterations. This is due to the fact that splitting and merging of tours is only important as long as the capacity constraints are either violated or highly over-satisfied, i.e., there is significant capacity left in more than one trolley. Anyhow, in most of the iterations, i.e., in about 95% of the iterations, no improvement can be achieved by these neighborhoods. Therefore, some dynamic order mechanism guaranteeing that neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ are primarily examined during the beginning phase of VND while being applied less frequently during the later iterations seems to be important.

In contrast to self-adaptive VND as proposed by Hu and Raidl [64], we do not punish or reward neighborhood structures based on their examination times, but reorder the neighborhoods according to their success rates, i.e., the ratios of improvements over examinations, only. The neighborhood order is updated each time an improvement on the current solution could be achieved.

In addition, we adapted VND such that not only improvements on the current solution are accepted but also moves can be applied which leave the current objective value unchanged. To avoid infinite loops, at most $z \geq 1$ non-improving subsequent moves are allowed in our version of VND, whereas the i-th non-improving move is accepted with probability $1 - (i-1) \cdot {}^{1}\!/_{z}$, only. All counters regarding the acceptance of non-improving moves are reset as soon as an improvement could be achieved. Based on experimental results we observed that this adaption helps to escape local optima. Especially in cases where the swapping of two articles between two tours or the shifting of one tour to another tour is performed for tours which have both to enter the same (sub-)set of aisles. Although the tour lengths are not changed, the used space on the trolleys is affected which may result in the situation that afterwards a due to the capacity constraints impossible move can be performed. Preliminary tests revealed that setting parameter $z = 10$ results in about 10% to better results while the running times are still kept low. For larger values the solutions did not significantly improve while the running time exceeded the desired limit given by our industry partner. As step function a *next improvement* strategy was implemented, whereas a random examination order was chosen to uniformly sample the current neighborhood.

Computing Individual Tours

Another crucial point of the proposed algorithm is the computation of concrete tours which will be used by the warehousemen for collecting a specific set of ordered items since the evaluation of the assignment of articles to selections is based on the shortest possible tours, and therefore an efficient tour computation is needed.

For this purpose we will present an approach based on dynamic programming for computing optimal tours through the warehouse. It should be mentioned, however, that in [116] another dynamic programming approach was published. Although that method can be used for computing tours as needed in our case, the method proposed in the following is more flexible in the sense that it can be easily adapted such that not only tours but also paths or all other traversings of the warehouse can be computed using the same dynamic program with only a few minor modifications.

Please note that a tour as used within this work does not correspond to tours as used within works related to the traveling salesman problem or the vehicle routing problem. In fact, the main difference lies therein that tours within a storage are allowed to visit each point of interest, i.e., among others the *packing station*, crossings of aisles and rack positions, more than once. This is simply induced by the circumstance that in most

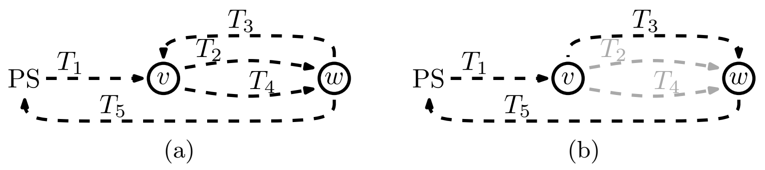


Figure 3.3.: How to construct a tour T' from a given tour T under the assumption that T visits two times location w immediately after location v.

cases no direct connection between two points of interest exists. Consequently, paths between points of interest can be walked along more than once within one tour. Anyhow, an upper bound for the number of times the same passage is walked can be provided based on the following two observations.

Theorem 1. *Given is a tour T, which is of shortest length with respect to a set of points of interest, i.e., all of these points are visited by T. Further, we assume that there exist two adjacent points of interest v and w which are twice visited immediately consecutively in T. Then the passage between v and w is once traversed from v to w and once vice versa in T.*

Proof. Let us assume that the passage between points v and w is traversed twice in the same direction. Then, we can split tour T into five subwalks T_1, T_2, T_3, T_4 and T_5 as shown in Fig. 3.3a, whereas PS denotes the packing station. A new tour T' can be built by passing segment T_1 from PS to v followed by traversing walk T_3 from v to w and finally walking along T_5 from w to PS, see Fig. 3.3b. Since v and w are adjacent, i.e., no other point of interest has to be visited when walking from v to w, T' visits the same points of interest as T. Furthermore, since subwalks T_2 and T_4 are not traversed within T', T' is shorter than T, which is a contradiction to the assumption that T is minimal. $\square$

Lemma 1. *Given is an optimal tour T with respect to a set of points of interest. Then any two adjacent points v and w are visited at most twice immediately consecutively by T.*

Proof. This lemma directly follows from Theorem 1. Under the assumption that points v and w are visited more than two times immediately consecutively the passage between these two points has to be traversed at least twice in the same direction. $\square$

Based on the special structure induced by warehouse layouts similar to that shown in Fig. 3.2, we define *aisle operations* (AOs) and *inter-aisle operations* (IOs). While AOs are representations of the walks to be performed within rack aisles, IOs correspond to movements in main aisles. In Fig. 3.4 the sets of basic AOs and IOs are shown. Each

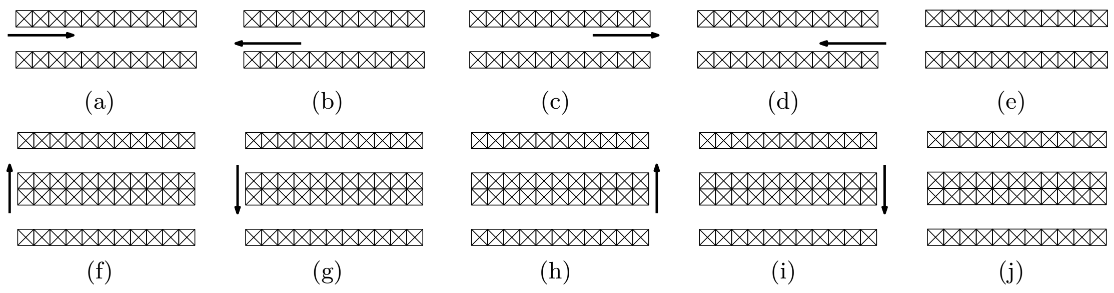


Figure 3.4.: In (a)–(e) the five basic aisle operations (AOs) are presented, whereas (f)–(j) show the basic inter-aisle operations (IOs). In this graphics the rectangles with crosses symbolize rack positions. The aisle, obviously, runs in-between of the racks.

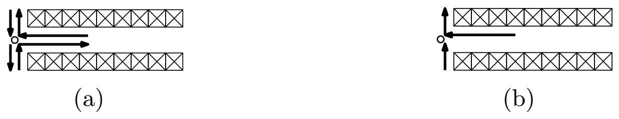


Figure 3.5.: Figure (a) shows a module representing that part of a tour entering and leaving the RA from and to the left side. This aisle operation is then suitably joined with the rest of the tour by appropriate inter-aisle operations.

of the arrows represents a part of a tour through the warehouse, e.g., in Fig. 3.4a the corresponding aisle in entered from "the left". In contrast, Fig. 3.4f shows a part of the tour which leads from the "left end" of an aisle to the "left end" of the "next" aisle. By appropriately combining these basic operations, so-called modules can be defined, which will then be used for representing parts of tours, for an example see Fig. 3.5a. Based on Theorem 1 it can be concluded that the number of different module types needed for representing a tour is limited to 1792. This number can be derived as follows: As shown in the example in Fig. 3.5a at most three incoming and three outgoing edges can be connected to the point indicated by the small circle. Obviously, there are 2^6 possibilities for selecting a subset of these edges, i.e., 64 different tour parts for each aisle end which implies that there a total of 4092 possible modules. Some of these modules are, however, invalid, e.g., see the invalid module depicted in Fig. 3.5b. Subtracting all the invalid modules result in a total of 1792 valid modules.

Although it is now obvious that tours can be built by selecting an appropriate module for each aisle to visit, it can be observed that the resulting tours may contain subtours, which are not connected to the rest of the tour, see for example Fig. 3.6a. Unfortunately, as shown in Fig. 3.6b, the decision whether a combination of modules is valid cannot always be made as soon as the next module is selected. Let us denote by $\mathcal{N}_c(j)$ the set

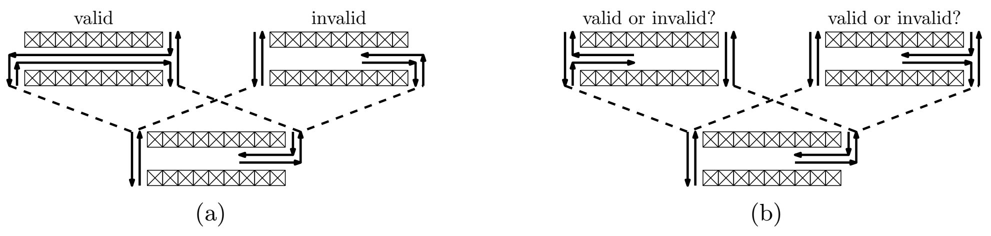


Figure 3.6.: For some combinations of modules (a) it can be directly decided whether or not they are valid. In other cases (b) this decision has to be postponed.

of those modules j' which might be connected with module j with respect to the IOs of j and j', i.e., all modules j' forming together with j possibly valid tour parts. Further, we denote by $\mathcal{N}_{\mathrm{v}}(j)$ the set of those modules $j' \in \mathcal{N}_{\mathrm{c}}(j)$ such that the usage of modules j and j' results in a definitely valid tour (part).

Therefore, we introduce two $(n+1) \times (\nu)$ matrices σ and τ, with n being the number of aisles containing items to be selected and ν indicating the maximum number of potentially used module types. An entry σ_{ij}, with $1 \leq i \leq n$ and $1 \leq j \leq \nu$, corresponds to the length of a valid tour T' which visits all rack locations in aisles 1 to i storing articles to be shipped to customers and performs in aisle i the operations corresponding to module j. Analogously, an entry τ_{ij} corresponds to the total length of tour parts which visit all racks in aisles 1 to i storing articles to be shipped and perform in aisle i the operations corresponding to module j. Anyhow, these tour parts need not to be connected with each other and therefore it has to be assured that they are going to be joined into one (big) tour by operations performed in any aisle $> i$. Now, let us assume that $c_i(j)$ denotes the length of the tour part(s) represented by module j when applied to aisle i and module μ represents the IOs necessary for reaching the first aisle from the packing station. Then, the entries of σ and τ can be computed by using the following recursive functions:

$$\sigma_{0\mu} = \tau_{0\mu} = 0 \tag{3.12}$$

$$\sigma_{0j} = \tau_{0j} = \infty \qquad \forall j \in \{1, \dots, \nu\} \setminus \{\mu\} \tag{3.13}$$

$$\sigma_{ij} = c_i(j) + \min \left\{ \begin{array}{l} \{\sigma_{i-1j'} : j' \in \mathcal{N}_{\mathrm{v}}(j)\} \cup \\ \{\tau_{i-1j'} : j' \in \mathcal{N}_{\mathrm{v}}(j)\} \end{array} \right\} \qquad \begin{array}{l} \forall i \in \{1, \dots, n\} \\ \forall j \in \{1, \dots, \nu\} \end{array} \tag{3.14}$$

$$\tau_{ij} = c_i(j) + \min \left\{ \begin{array}{l} \{\sigma_{i-1j'} : j' \in \mathcal{N}_{\mathrm{c}}(j)\} \cup \\ \{\tau_{i-1j'} : j' \in \mathcal{N}_{\mathrm{c}}(j)\} \end{array} \right\} \qquad \begin{array}{l} \forall i \in \{1, \dots, n\} \\ \forall j \in \{1, \dots, \nu\} \end{array} \tag{3.15}$$

For determining the optimal tour, one first needs to identify module J used for aisle n in an optimal tour, i.e., $J = \arg\min_{j \in \{1,\dots,\nu\}} \{\sigma_{nj}\}$. Then, the computations based on

Eq. (3.14) and (3.15) have to be performed backwards. Anyhow, it can be easily proven that $\sigma_{nJ} \neq \infty$ definitely holds. In case of ties any module can be chosen.

Assignment of Workers to Tours

In a final step, an assignment of workers to tours has to be computed such that the latest finishing time is as early as possible while regarding the guaranteed delivery times. Obviously this problem is a variant of the well known $\mathcal{NP}$-complete job shop scheduling [43] which asks to schedule a set of jobs having different lengths on a set of homogeneous machines such that the latest finishing time of all jobs is as early as possible. In our case, the tours correspond to jobs and and the warehouse worker correspond to machines. Several different approaches have been proposed for solving this problem, e.g., [49, 69, 3].

Although an application of this methods would obviously be possible, we decided to develop a VNS based approach for assigning tours to workers. In fact, what we desire in this step of the algorithm is a valid assignment of workers to tours, i.e., an assignment such that all articles are collected before the deadline. It is, however, in fact unimportant how early the last article is delivered to the packing station. Since preliminary tests revealed that in most situations the assignment arising in this problem can be easily solved and there were recently successful applications of VNS based approaches to variants of job shop scheduling published[42, 120], we integrated the following approach in our framework. It turned out that the computation of this scheduling was never the limiting factor of our approach for the instances used for testing.

The developed General VNS scheme is initialized using a greedy construction heuristic which sorts the tours ascending with respect to their lengths, and systematically assigns them to workers. For the shaking phase a random swapping of two tours between two workers are performed.

The local improvement phase is realized using a VND approach for which three neighborhood structures are defined: the first one is based on *reassign moves* which simply reassign one tour from one worker to another worker. The second one is defined via *swap moves* which swap two tours between two arbitrarily chosen workers and the last one is based on the rearrangement tours for one worker, i.e., one tour is selected and shifted to the beginning of the schedule for one worker. The neighborhood ordering is fixed and corresponds to this order. For examining neighborhoods a *first improvement* strategy is applied as step function.

Table 3.5.: Average results over 20 runs for 20 instances. The initial values, the objective values (including standard deviations in parentheses), the number of tours (n_T), the average filling degree of the trolleys used (quota) and the average computation times in seconds are opposed for instances allowing to reverse in aisle and disallowing turning around. The last column presents the p-values of an unpaired Wilcoxon rank sum test, for evaluating whether the tours with turning around are 20% shorter than those without reversing.

		reversing disabled					reversing enabled					
inst.	init	objective		n_T	quota	time	objective		n_T	quota	time	p-Val
(01)	3750	3059.0	(132.6)	3.0	74.2	37.5	2281.0	(44.7)	3.0	74.2	35.5	<0.01
(02)	4200	3213.0	(156.5)	3.7	79.3	29.2	2348.0	(85.6)	3.5	85.0	59.1	<0.01
(03)	4260	3560.0	(109.0)	4.0	75.5	47.5	2541.5	(80.0)	4.0	75.5	78.7	<0.01
(04)	3210	2962.0	(35.8)	3.0	91.8	22.6	2298.5	(4.9)	3.0	91.8	30.6	<0.01
(05)	4020	3605.0	(105.6)	4.1	88.1	37.8	2466.5	(39.1)	4.0	90.3	55.5	<0.01
(06)	5060	4671.5	(106.7)	5.7	78.3	61.2	3576.0	(74.9)	5.4	82.0	114.6	<0.01
(07)	6050	5575.5	(79.3)	7.0	81.7	83.8	4052.5	(73.5)	6.8	82.9	192.7	<0.01
(08)	5650	5602.5	(155.8)	7.0	84.9	95.6	4159.5	(98.9)	7.0	85.5	212.6	<0.01
(09)	7000	6583.0	(246.9)	8.0	86.4	132.5	4887.5	(117.6)	8.0	86.4	334.1	<0.01
(10)	5070	4995.5	(252.1)	6.0	78.8	88.2	3589.5	(104.1)	5.8	82.2	128.8	<0.01
(11)	10740	9254.0	(268.4)	12.2	81.9	391.4	6158.5	(149.3)	11.1	90.4	845.4	<0.01
(12)	9350	8155.0	(175.9)	12.6	79.3	255.4	5952.0	(136.9)	11.7	85.4	689.0	<0.01
(13)	9970	8939.0	(256.2)	12.0	83.2	323.9	6102.0	(156.7)	11.3	88.0	715.7	<0.01
(14)	9520	9082.5	(246.5)	12.6	79.6	370.7	6165.5	(181.2)	11.7	85.8	864.3	<0.01
(15)	7690	7473.0	(270.4)	11.5	86.9	279.2	5860.5	(74.9)	11.2	89.2	673.0	0.02
(16)	11510	8878.0	(240.3)	12.2	81.9	716.4	6465.0	(165.9)	11.7	85.8	1200.0	<0.01
(17)	11460	8251.5	(216.7)	12.3	80.9	782.2	6261.5	(161.7)	11.7	85.4	1200.0	<0.01
(18)	11740	8520.0	(187.8)	12.6	79.3	748.5	6238.5	(159.9)	11.5	86.9	1200.0	<0.01
(19)	11480	8990.0	(216.8)	12.1	82.6	828.3	6349.0	(207.0)	11.7	85.8	1200.0	<0.01
(20)	12260	9644.5	(286.9)	12.6	79.6	819.0	6635.0	(164.4)	11.8	85.0	1200.0	<0.01

3.2.4. Experimental Results

For evaluating the performance of the proposed method several test runs were performed. Our algorithm was implemented in Java 6 and all tests were run on a single core of a Dual-Core AMD Opteron™ Processor 2.6GHz and 4GB of RAM. All instances were randomly generated based on the characteristics of real-world data, i.e., storage layouts and typical customer orders, provided by our industry partner Dataphone GmbH. There are about 500 articles stored in the warehouse and between 25 to 200 randomly selected articles have to be picked. The concrete number of items to be collected for each of the 20 generated instances is shown in the second column of Tab. 3.6.

For each of the 20 instances, we performed 40 independent runs, whereas for the half of those runs we allowed that workers may reverse within one aisle, while for the remaining ones we assured that once an aisle is entered, it is completely traversed by the warehousemen. To avoid excessive computation times, a time limit of 1200 seconds for VNS was applied. See Tab. 3.5 for a detailed listing of the obtained results. The column labeled init presents the initial values obtained by the so called s-shaped heuristic [26], whereas we simply try to collect as much articles as possible during one tour regarding the capacities of the trolleys used. The objective values provided within the next columns correspond to the weighted sum as presented in Eq. 3.11. The standard deviations (given in parentheses) are relatively low with respect to the tour lengths. Taking a look at these average values it can be observed that the tour lengths can be reduced by about 20% on average when it is allowed to reverse within an aisle. To statistically confirm this observation, we performed an unpaired Wilcoxon rank sum test. The corresponding p-values are shown in the last column of Tab. 3.5. Regarding the number of tours as well as the filling degree of the trolleys, no significant difference between those runs allowing reversing and those forbidding it can be identified. Finally, taking a closer look at the computation times, it can be observed that for those instances including the option to reverse the running times are longer. This can be reasoned by the fact that the number of aisle operations is more restricted for those runs forbidding reversing. Since the available computation time was limited, the results obtained for instances 16–20 might be further improved when using looser time limits. Regarding the last step of the algorithm, i.e., the assignment of tours to workers, all violations of the time constraints could be resolved within a few iterations of the corresponding VNS procedure.

Regarding the performance of the proposed neighborhoods, i.e., the ratio of improvements over examinations, we observed that all of them except $\mathcal{N}_6$ contribute substantially to the final solution whereas neighborhood $\mathcal{N}_6$ did almost never add an improvement (see Tab. 3.6). Nevertheless, for the small instances with 25 articles to be shipped, some improvements could be achieved even by $\mathcal{N}_6$, and therefore we included it here. Regarding the other neighborhood structures, $\mathcal{N}_3$, i.e., the neighborhoods based on the ShiftArticle move, performed best. It is interesting that neighborhood structure $\mathcal{N}_4$ did worse for those instances forbidding turning around. However, this can be explained by the fact that for these instances the degree of freedom is less than for the others which results therein that once an aisle i has to be entered all ordered articles stored within this aisle can be collected with less expenses from aisle i than from any other aisle. The same observation holds for neighborhood structure $\mathcal{N}_5$, although this effect is less prominent for the underlying move type.

3.3. Summary

Warehouse management and especially logistics related to warehouse management are of great importance in today's production facilities—mainly due to the potential for

Table 3.6.: Average contributions of the individual neighborhood structures to the final solutions. The numbers represent the average ratios of improvements over examinations over 20 runs for 20 instances with 25, 50, 100, 200 different items to be collected (#it.).

		reversing disabled						reversing enabled					
inst.	#it.	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\mathcal{N}_5$	$\mathcal{N}_6$	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\mathcal{N}_5$	$\mathcal{N}_6$
(01)	25	3.6	10.6	60.2	34.4	46.9	0.2	5.9	15.9	70.2	61.5	18.2	0.0
(02)	25	4.1	12.0	61.0	32.2	33.8	0.0	5.6	19.7	73.0	46.2	16.3	1.1
(03)	25	3.6	11.8	60.5	40.6	49.9	0.1	4.4	17.1	73.4	55.2	27.9	0.0
(04)	25	4.5	11.1	52.1	27.4	34.6	0.0	7.4	18.2	70.9	42.8	6.9	0.4
(05)	25	6.6	15.2	65.9	29.3	45.2	0.8	9.1	19.9	74.2	58.0	32.3	0.0
(06)	50	10.4	17.6	69.6	3.7	18.7	0.0	10.7	23.3	81.8	53.5	7.6	0.0
(07)	50	12.2	20.7	74.3	6.0	42.0	0.0	12.9	25.7	85.6	59.6	48.2	0.0
(08)	50	13.5	19.5	72.3	1.7	55.9	0.0	19.1	28.6	87.9	29.9	40.2	0.0
(09)	50	13.9	20.3	74.4	41.2	46.3	0.0	17.8	27.8	85.8	46.8	37.4	0.0
(10)	50	9.8	17.4	70.8	9.2	22.3	0.0	12.6	23.1	81.8	66.9	18.5	0.0
(11)	100	16.0	27.1	76.3	7.9	24.6	0.0	21.6	29.4	88.5	44.0	37.0	0.0
(12)	100	18.9	29.9	77.8	0.9	24.7	0.0	18.4	28.8	88.3	47.2	39.2	0.0
(13)	100	16.5	26.2	75.9	5.9	26.7	0.0	23.5	29.3	87.3	60.4	44.9	0.0
(14)	100	14.3	27.6	78.2	1.1	19.5	0.0	19.4	27.2	88.0	50.5	33.8	0.0
(15)	100	18.2	23.2	73.4	1.1	12.8	0.0	20.1	26.7	85.3	51.1	30.8	0.0
(16)	200	18.0	30.3	75.3	2.9	8.1	0.0	25.7	31.5	88.8	46.4	30.5	0.0
(17)	200	17.9	28.8	75.7	0.0	7.9	0.0	27.0	31.8	88.8	39.3	49.2	0.0
(18)	200	19.2	28.6	76.0	3.5	15.0	0.0	28.1	32.2	88.8	53.8	38.6	0.0
(19)	200	16.1	28.1	74.5	1.8	17.3	0.0	26.2	29.8	88.4	35.9	35.1	0.0
(20)	200	16.4	28.2	75.4	1.1	6.7	0.0	27.5	34.7	90.2	46.6	32.8	0.0

reducing arising costs. Competitive advantages can be achieved by companies paying attention in solving the hard to solve problems arising within this field of operations research.

Within this chapter we focused on two specific problems arising during stocking operations as well as during the picking. Whereas the former one was handled by defining a newly introduced evaluation function for warehouse states and applying greedy and *variable neighborhood descent* (VND) based approaches we proposed an approach based on *variable neighborhood search* (VNS) incorporating *dynamic programming* (DP) for optimally solving arising subproblems for the latter one.

The stocking strategy proposed in Sec. 3.1.4 was experimentally applied in a real world setting. The feedback given by the warehouse manager and especially by the warehousemen revealed that using the proposed stocking strategy the warehouse states could be significantly improved in comparison to the prior applied methods. In addition, the developed relocation strategy, cf. Sec. 3.1.5, based on variable neighborhood descent can be used to reduce the likelihood of conflicts occurring during stock removal. Anyhow, these reallocation operations are only executed during idle times of warehousemen dedicated to shipment, which implies that a reliable stocking strategy is crucial for efficient warehouse operations.

For the real world scheduling and tour finding problem within a spare parts warehouse discussed in Sec. 3.2 we proposed a new hybrid algorithm combining VNS and DP. We used a new self-adaptive VND rearranging the neighborhoods according to their success rates for boosting the performance of the neighborhood structures used within the embedded VND. Individual optimal tours through the warehouse are computed by means of dynamic programming, whereas the special structure of the storage is exploited. Experimental investigations showed that this approach performs good for instances based on real-world characteristics. Further, we showed that the total tour lengths can be reduced by about 20% on average when reversing within aisles is allowed. Regarding the computation times, our approach is able to provide good results within 1200 seconds which correspond to acceptable time limits in real-world scenarios.

Although the results obtained by our methods are promising and specifically improved the warehouse management of our paper production company substantially, the further improvement potential for both discussed settings is high. For example, one main drawback is that both approaches only concentrate on the specific problem while related (optimization) problems like the paper production process or the stocking operations (in case of the spare parts warehouse) are disregarded. For this purpose, a deeper investigation of the processes running in the background needs to be done such that unnecessary intricateness is reduced and the overall production process including customer ordering, production, stocking, and shipping is enhanced.

Chapter 4

Reconstruction of Destructed Documents

Since ages sensitive data is of great importance for the human race. Beside securely transmitting and storing (secret) messages one could also gain an advantage by possessing information unavailable to ones antagonist—be it a military enemy or a trade rival. Nevertheless, at some time it is usually necessary to irrecoverably destroy this information. In the ancient world the easiest way was possibly burning a piece of papyrus or braking stone plates containing data worth of protection into small pieces.

Nowadays sensitive data is stored, among others, by writing it down on paper. Of course, burning such paper documents is still an imaginable method of destruction. Unfortunately, sparking a fire is in most offices not possible such that mostly documents are torn apart. In case of many documents and pages it is often more convenient to use a so-called shredder. This is a device for mechanically cutting sheets of paper into snippets of predefined size. The German institute for standardization (DIN) defines in DIN EN 15713 eight different levels of destruction, whereas for paper only levels 1 to 6 are realized [30]. A snippet forming the output of such a shredder is then at most $5000\,\text{mm}^2$ with a maximum width of 25 mm (level 1) to $320\,\text{mm}^2$ with a maximum width of 4 mm (level 6).

Anyhow, there are situations in which it is necessary to reconstruct destroyed—mainly shredded or manually torn—documents. Of course, there are a lot of ethically questionable applications of a document recovery system like (industry) espionage. On the other hand forensics or archeology are socially accepted fields of utilization for document restoration. In the former, it is necessary to reconstruct sheets of paper to be used as evidence in trials. In archeology, it is in the most cases necessary to assemble broken parchment documents to obtain information on ancient cultures. In addition, antiquarian studies concentrating on the near past are of great importance. Just to mention one, the reconstruction of dossiers destroyed by the *Staatssicherheit* (*Stasi*, germ. *German secret police*) during the last days of the *German Democratic Republic* (DDR) is one of the most challenging tasks of modern archeology. The task in this project is to reconstruct about 45 million document pages mostly manually torn into about 600 million snippets [22].

In the rest of this chapter we assume that the paper documents to be reconstructed are either manually torn apart or mechanically shredded using a shredding device. For the latter case, there is a further classification into approaches trying to restore documents cut into strips of arbitrary widths all having the same height as the original document sheets—so called *strip shredded* documents—and documents cut into snippets all of equal size but having heights not equal to the height of the original document pages—so called *cross cut* shredded documents.

Anyhow, if not otherwise mentioned we assume that the given documents are consisting of equally sized, rectangular sheets of paper of format DIN A4. The proposed methods can, however, be easily adapted for paper documents with arbitrary formats. In addition only one face of the sheets contains (valuable) information, i.e., the front face of each snippet is known.

Obviously, for the reconstruction of destroyed documents the application of pattern recognition and image processing methods is fundamental. Additionally, it is, however, important to find a method exploiting this visual information. Therefore, the main contribution of this section is the development of approaches focusing on the combinatorial optimization aspects of the reconstruction of destructed paper documents. Pattern recognition approaches are deliberately omitted (as far as possible). Nevertheless, in the final stage of extension any reconstruction system will incorporate both pattern recognition methods and combinatorial approaches. For this reason we designed our methods such that this combination is as easy as possible.

In the next section we will give a short overview on methods handling manually torn paper documents. Afterwards we will concentrate on strip shredded as well as cross cut shredded documents. Although the main focus of this chapter is the reconstruction of destroyed documents by means of combinatorial optimization methods we will also reference to works based on pattern recognition were appropriate and necessary. At the end of this chapter we will give a short discussion on the impact of the presented methods on data security and confidentiality.

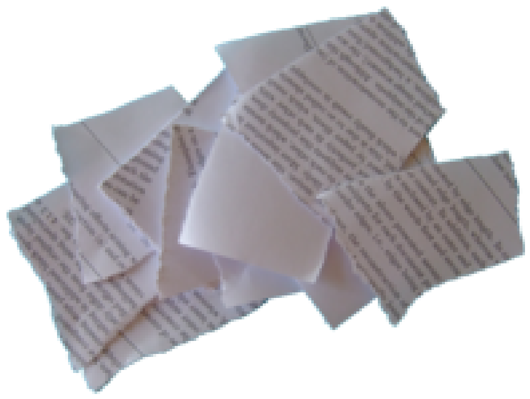


Figure 4.1.: Real world example of snippets produced by manually tearing one sheet of paper.

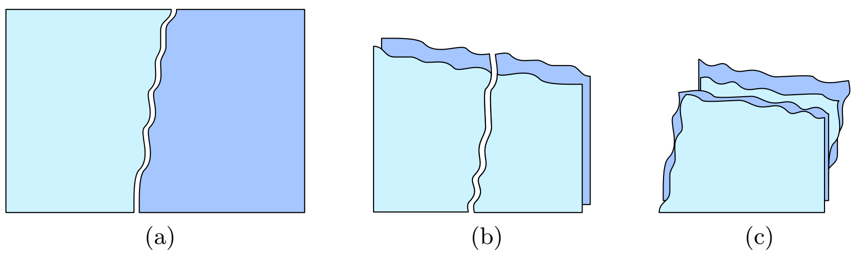


Figure 4.2.: Exemplary progress of a typical paper document destruction: (a) the first tear starts at a long edge and ends at the other long edge, (b) the two snippets are put on top of each other and torn again, and (c) finally the two stacks are put on top of each other again. The relative order of snippets is never shuffled.

4.1. Manually Torn Paper Documents

Although there are many imaginable variants of how to destruct paper most people decide to tear it multiple times until the (confidential) information is destroyed. Based on empirical experiments we observed that the applied pattern for tearing paper sheets is in most cases the same, namely the first tear runs parallel to the shorter edges of the document and the two resulting snippets are put on top of each other. These two steps are then iterated several times which results in a set of snippets as shown in Fig. 4.1, also cf. [27]. Therefore the following assumptions can be formulated for a "typical" case:

(P I) The sheets of paper are rectangular. (Each sheet has two long and two short edges.)

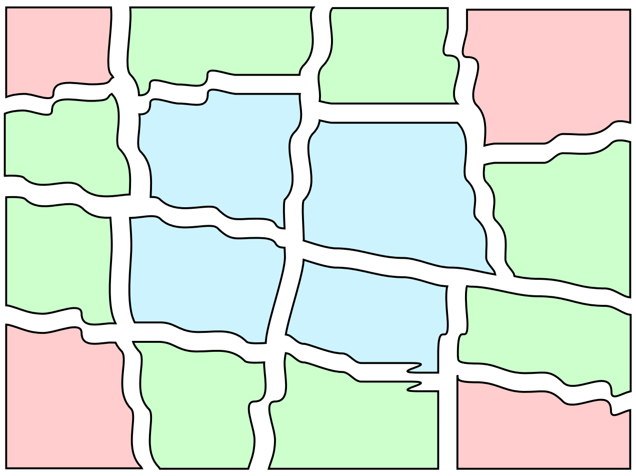


Figure 4.3.: Schematic illustration of a typical tear pattern. The blue, green, and red snippets are also called *inner*, *border*, and *corner* snippets, respectively.

(P II) The first tear starts at one long edge, runs roughly parallel to the short edges and ends at the second long edge, cf. Fig. 4.2a.

(P III) The resulting snippets of a tearing operation are put on top of each other. The next tear is again approximately parallel to the short edges, cf. Fig. 4.2b. The relative order of snippets within the stack is not shuffled during this operations, cf. Fig. 4.2c.

(P IV) Each tear cuts each snippet in half, i.e., each tear doubles the number of snippets.

(P V) At least two tearing operations are performed, i.e., there are at least four snippets.

Although the tearing process could be iterated for an arbitrarily chosen number of times, it is normally ended after four or five tears. (This implies that there are 16 or 32 snippets, respectively.) A typical tear pattern for four tears, i.e., the schematic shape and alignment of the resulting snippets, is shown in Fig. 4.3. As can be seen, there are some snippets at the border of the original document—the so called *outer snippets*—and some in the inner regions—the so called *inner snippets*. The outer snippets can further be divided into those snippets containing a document corner—they are called *corner snippets*—and the rest which are called *border snippets*, cf. Fig. 4.3. To further standardize the terminology to be used let us have a closer look at the snippets itself: It is convenient to approximate arbitrarily shaped real world snippets by polygons. In most cases quadrangles would be sufficient, cf. Fig. 4.4. This leads, however, to following properties

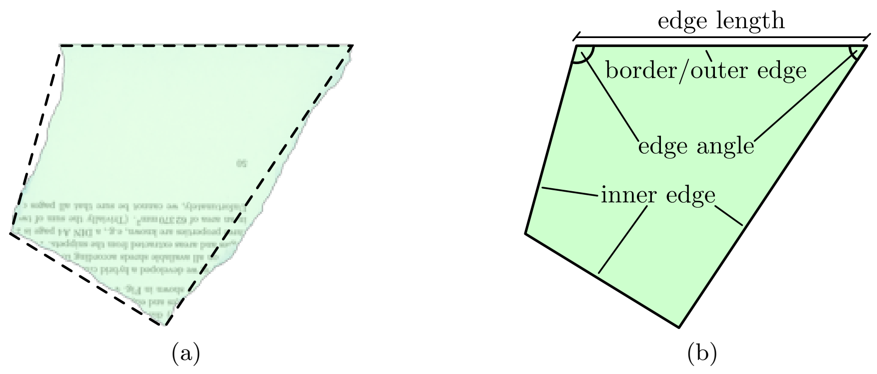


Figure 4.4.: Approximation of snippets by general polygons (a) and indication of snippet features (b).

of the approximated snippets: Each border snippet has one *border* or *outer edge* which corresponds to a part of a document edge. Obviously, corner snippets contain two outer edges enclosing a right angle and inner snippets do not have any outer edge. All other edges, i.e., those not being an outer edge, are called *inner edges*. The length of an outer edge is simply denoted by edge length. In case of corner snippets we will also talk of the first and the second edge length, whereas the concrete meaning of first and second will derive from the context. In addition, each outer snippet has so called *edge angles* which are the angles enclosed by an outer and an inner edge. Again, we sometimes distinguish both angles by the words first and second, whereas the context will clarify ambiguities.

Based on the above presented assumptions it can be observed that there must be four corner snippets for each document page. Furthermore, there is no snippet with three or more border edges, i.e., edges being part of the original document edge.

At first glance, the reconstruction of manually torn paper documents is identical with the well known (and universally loved) game named *puzzle*. For some works related to solving jigsaw puzzles we refer to [5, 18, 21, 25, 48, 140]. On closer inspection one will discover that there are small but crucial differences between these two problems. While the shape of the jigsaw puzzle tiles are (in most cases) unique, this needs not to hold for the reconstruction of documents—although it is very likely. More important is, however, the fact that two matching puzzle tiles almost perfectly fit to each other, which means that there are in particular no overlappings of two joined tiles. For paper snippets this property does not hold. It is even very likely that two snippets do not perfectly fit each other with respect to the shape: On a detailed view, this is caused by paper fibers, which result in frayed edges, see Fig. 4.5a. On a coarser level, this is due to so called shearing effects, which are caused by multiple layers of fibers that finally result in overlapping regions of snippets, see Fig. 4.5b for an example.

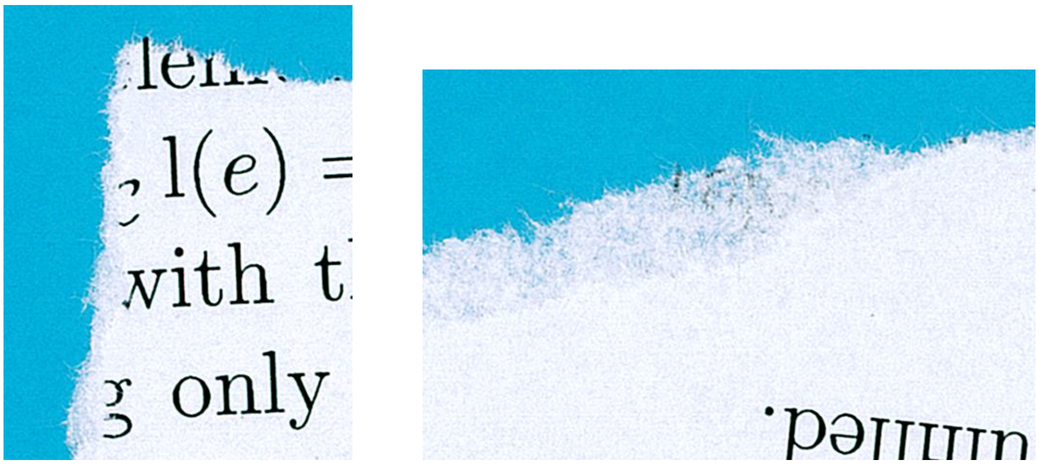


Figure 4.5.: Examples of shearing effects arising when tearing sheets of , typical office paper: Frayed edges on a close inspection (a) and shearing effects, i.e., overlapping regions (b).

Even more, there are some practical issues with respect to the reconstruction of (manually torn) paper documents: namely the digitalization of the snippets. Although this step can be done using a standard scanning equipment (flatbed image scanner), the scanning of a heap of snippets is rather time consuming. In addition, a background/foreground extraction has to be applied and depending on the selected reconstruction approach the approximation of the snippets by polygons has to be computed. In addition a rotation vector has to be determined. During all these preprocessing steps plenty of errors can be induced, e.g., by dust particles on the flatbed scanner resulting in additional black spots on the image(s), etc. such that the result might significantly differ from the original input. Nevertheless, some of the indicated issues can be handled efficiently by using specialized tools and/or approaches. In conjunction with this challenges we refer to Sec. 4.1.5 as well as to literature dedicated to computer vision techniques, e.g., [68, 133, 131, 134, 28, 29, 27].

As already mentioned we assume that the front face of the snippets can be easily detected due to the fact that only one face of the remnants has printed text on them. However, this is not always true. For example it might happen that no information is shown on a snippet at all, i.e., it is blank on both faces. Further, the original document could be printed using a duplex printer, i.e., there is information on both faces. In these cases the complexity of the problem increases since beside the decision where to place each snippet it also has to be decided whether or not a remnant should be flipped. It is, however, possible to straightforward adapt the proposed methods to these more complex situations. Therefore, we will focus for the sake of simplicity on the reconstruction of one sided documents.

In the rest of this section, we will first discuss the complexity of this problem and continue with an overview of the—in our opinion—most important works related to the reconstruction of manually torn paper documents. Sections 4.1.2 and 4.1.3 mainly present the work done together with Peter Schüller and Franz Berger who were both former master's students at our institute. An in some aspects more detailed discussion of the proposed approaches can be found in their master theses [123, 14]. The rest of this section, i.e., Sec. 4.1.4 and 4.1.5, focuses on other promising and important approaches for reconstructing manually torn paper documents recently published in literature.

4.1.1. Complexity Results

To be able to give a statement on the complexity of the reconstruction of manually torn paper documents (Rmtpd), it is in a first step necessary to formalize the given problem. For this purpose we define the following partitioning of the set $\mathcal{S}$ of all given snippets:

- $\mathcal{C}$ denotes the set of all corner snippets
- $\mathcal{B}$ is the set of all border snippets, i.e., all outer snippets except corner snippets, and
- $\mathcal{I}$ contains all inner snippets.

Further, we introduce for each border snippet a function $\mathrm{l} : \mathcal{B} \rightarrow \mathbb{Z}^+$ whose values correspond to lengths of the snippets' outer edges, cf. Fig. 4.4b. For the set of corner snippets this function is defined as $\mathrm{l} : \mathcal{C} \times \{1, 2\} \rightarrow \mathbb{Z}^+$ assigning both border edges—numbered counter clockwise—of each corner snippet a corresponding length value. Furthermore, we are given four integers c_0, c_1, c_2 and c_3 corresponding to the lengths of the edges of the original document page(s). (For rectangular document pages the equalities $c_0 = c_2$ and $c_1 = c_3$ must hold.)

Although a further formalization regarding the edge angles, the length of inner edges, etc. could be done, we think that it would neither improve the proof on the complexity of Rmtpd nor the understanding of this document.

When reconstructing manually torn documents it is among others necessary to correctly reconstruct the border of the document. Under the assumption that the given snippets belong all to the same document page, i.e., the original document to be reconstructed consisted of only one sheet of paper, a solution to this subproblem would result in four ordered sets $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, whereas the first and the second element of these sets correspond to corner snippets and all other to border snippets. The following additional properties must be fulfilled, whereas $\mathcal{E}_i(j)$ denotes the j-th element of $\mathcal{E}_i$:

$$\bigcup_{i=0}^{3} \mathcal{E}_i = \mathcal{B} \cup \mathcal{C} \tag{4.1}$$

$$\mathcal{E}_i \subset \mathcal{B} \cup \mathcal{C}, \qquad \forall i \in \{0,1,2,3\} \tag{4.2}$$

$$\mathcal{E}_i(1) = \mathcal{E}_{(i+1) \bmod 4}(2) \in \mathcal{C} \tag{4.3}$$

$$|\mathcal{E}_i \cap \mathcal{E}_{(i+2) \bmod 4}| = 0 \tag{4.4}$$

$$\sum_{j \in \mathcal{E}_i \setminus \mathcal{C}} \mathrm{l}(j) + \mathrm{l}(\mathcal{E}_i(1), 1) + \mathrm{l}(\mathcal{E}_i(2), 2) = c_i \qquad \forall i \in \{0,1,2,3\} \tag{4.5}$$

Basically these constraints state that all outer snippets must be used (4.1), only outer snippets are allowed to be assigned to borders (4.2), two document edges connected with each other must contain the same corner snippet (4.3), one snippet must not be contained in two parallel edges (4.4), and the sum of the lengths of the corresponding snippets must be equal to the length of the original document edge (4.5); see also Fig. 4.7 for an illustration.

Of course, for a practical solution of Rmtpd a solution must also propose an order of snippets along an edge. It is, however, easy to show that an extension of the above presented model respecting a concrete ordering of snippets along an edge does not change the complexity of the problem. For simplicity and for facilitating comprehension of the following proof, we disregard this extension.

The following proof of $\mathcal{NP}$-hardness is based on the reduction of Subset Sum to Rmtpd. For this purpose, we define Subset Sum as follows (see also [43]): Given are a finite set $\mathcal{A}$, a function $\mathrm{s} : \mathcal{A} \rightarrow \mathbb{Z}^+$ and a positive integer b. An instance is answered with *yes*, i.e., a valid solution is found, iff there is a subset $\mathcal{A}' \subseteq \mathcal{A}$ such that $\sum_{a \in \mathcal{A}} \mathrm{s}(a) = b$ holds.

Now let us introduce a transformation from Subset Sum to Rmtpd which is polynomial in the number of elements in $\mathcal{A}$ for a given instance of Subset Sum. Without loss of generality, let us assume that $b \geq \sum_{a \in \mathcal{A}} \mathrm{s}(a)/2$. In the case that $b < \sum_{a \in \mathcal{A}} \mathrm{s}(a)/2$, it can be easily shown that another instance of Subset Sum can be defined with the same set of elements and an integer b' such that $b' = \sum_{a \in \mathcal{A}} \mathrm{s}(a) - b$. The set $\mathcal{A} \setminus \mathcal{A}'$ is a solution to this new instance of Subset Sum. Anyhow, the given instance of Subset Sum can be transformed into an instance of Rmtpd using the following procedure:

1. Define four corner snippets with two outer edges, both with length 1. These four snippets build set $\mathcal{C}$.
2. For each element $a \in \mathcal{A}$ create a border snippet $s \in \mathcal{B}$ with $\mathrm{l}(s) = \mathrm{s}(a)$.

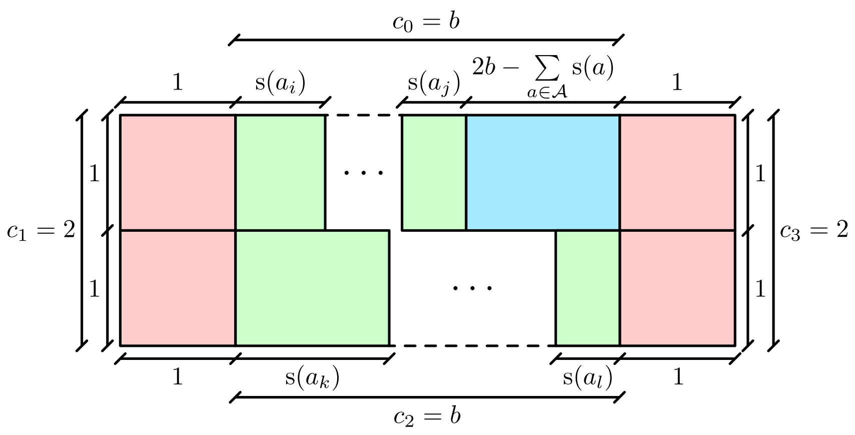


Figure 4.6.: Schematic visualization of the result obtained by transforming an instance of Subset Sum to Rmtpd. Red snippets correspond to the artificially introduced corner snippets. The green ones correspond to the elements in set $\mathcal{A}$. The blue snippet is the artificial one with edge length $2b - \sum_{a\in\mathcal{A}} \mathrm{s}(a)$.

3. If $2b - \sum_{a\in\mathcal{A}} \mathrm{s}(a) \neq 0$ create an additional border snippet $s \in \mathcal{B}$ with $\mathrm{l}(s) = 2b - \sum_{a\in\mathcal{A}} \mathrm{s}(a)$.
4. Set $\mathcal{I}$ is the empty set.
5. Define $c_0 = c_2 = b + 2$ and $c_1 = c_3 = 2$.

Obviously, this procedure is linear in the number of elements of $\mathcal{A}$. We refer to Fig. 4.6 of a schematic visualization of this transformation.

If there is a classification of the snippets into sets $\mathcal{E}_0$ to $\mathcal{E}_3$ for the derived instance of Rmtpd such that all constraints are satisfied, then equalities $\sum_{e\in\mathcal{E}_0\setminus\mathcal{C}} \mathrm{l}(e) = b$ and $\sum_{e\in\mathcal{E}_2\setminus\mathcal{C}} \mathrm{l}(e) = b$ hold. Since there is at most one element in $\mathcal{B}$ which is not derived from an element in $\mathcal{A}$, at least one of the two subsets $\mathcal{E}_0$ and $\mathcal{E}_2$ consists only of elements derived from elements in $\mathcal{A}$. Without loss of generality we can assume that this subset is $\mathcal{E}_2$, cf. Fig. 4.6. Therefore an inverse transformation of the elements in $\mathcal{E}_3$ into elements of $\mathcal{A}$ induce set $\mathcal{A}'$ with the property that $\sum_{a\in\mathcal{A}'} \mathrm{s}(a) = b$. Since the reconstruction of the borders is a subproblem of Rmtpd, i.e., any algorithm solving Rmtpd also reconstructs the document's borders, Rmtpd is $\mathcal{NP}$-hard. Obviously, Rmtpd is at the same time member of the complexity class $\mathcal{NP}$, since a solution to Rmtpd can be guessed and validated in polynomial time. Therefore Rmtpd is $\mathcal{NP}$-complete.

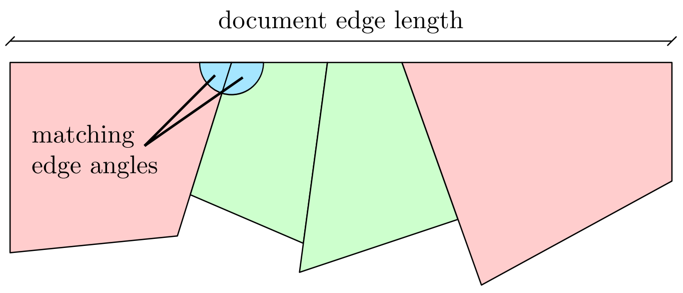


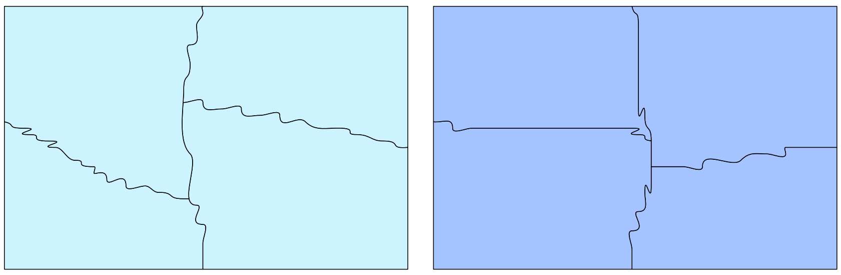
Figure 4.7.: Reconstruction of a document edge.

4.1.2. Reconstructing Edges of Paper Sheets

Analogously to the well known concept of solving the border of a jigsaw puzzle first, the basic idea of this approach is to reconstruct the borders of the original document. For this purpose, it is first necessary to approximate each physical snippet by a polygon, e.g., a general tetragon which will be convenient in most cases, see Fig. 4.4. Now, it can be discovered that the sum of two matching edge angles must be equal to 180°, see Fig. 4.7. In addition, the lengths of all outer edges assigned to the same document edge must sum up to the document edge length, cf. Fig. 4.7 and Eq. (4.5). Obviously these constraints must hold for all four document edges and all pairs of matched snippets which directly leads to an *integer linear programming* (ILP) formulation for solving this problem. Unfortunately, there will be errors made both in measuring the outer edge lengths as well as in approximating the physical snippets by polygons and therefore in calculating the edge angles. To overcome this imperfection of practical input data, it is convenient to introduce an objective function trying to minimize the imposed error, i.e., minimizing the deviations of the matching edge angles from 180° and the variations in the lengths of the reconstructed document edges to the lengths of the original document edges.

The well-disposed reader will find the complete ILP formulation in [123]. In addition, Schüller presents in that work a slightly modified ILP formulation taking only the lengths of document edges into account while omitting the matching angles constraints. As well documented in the test results section of his work, Schüller shows that this approach can be utilized for reconstructing the borders of one single page with up to 32 snippets, whereas 20 of those snippets are outer or corner snippets. The runtimes for these small instances are approximately 0.05 seconds, while the correctness is up to 80%, i.e., in 80% of all test runs the borders of the original document could be restored.

Unfortunately the runtimes dramatically increase while the output quality substantially decreases when the number of pages to be concurrently reconstructed is extended. For


(a) The two original document pages with tears indicated.

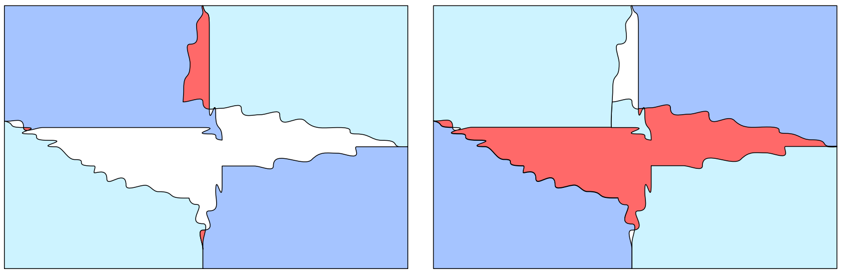

(b) The with respect to edge lengths correctly reconstructed document pages.

Figure 4.8.: An example for a perfect reconstruction with respect to document edges. Those regions with overlapping snippets are marked red in (b).

even only five different document pages runtimes of almost 300 seconds are reached. Obviously these results imply that, although this approach is promising for small sized instances, the exact reconstruction of multiple document pages is not practicable at this time. Furthermore, only the borders of the documents are reconstructed while the most valuable information is in most documents not on the page margins but on the inner regions. Nevertheless, for practical applications of an approach reconstructing only the borders of documents one has to state that for documents torn three times under properties **(P I)** to **(P V)** there are still only outer and no inner snippets. Even for four tears there are only four inner snippets while the number of outer snippets increases to twelve, cf. Fig. 4.3.

Furthermore, reconstructing the borders of document pages can on the one hand be used during a preprocessing phase constraining the solution space for further reconstruction steps focused on the inner regions of documents. On the other hand, a solution of a small subproblem by means of ILP techniques might be useful in a more coarsened approach as presented in the next section.

4.1.3. Exploiting Geometrical Information

While the approach presented in the previous section only concentrates on the reconstruction of the borders of document pages, there are still two related problems to be solved: On the one hand, this approach is limited to relatively small instances with only a few (up to five) different document pages. On the other hand, only the information about outer edges and edge angles are utilized such that at the worst infeasible solutions as exemplary shown in Fig. 4.8b are produced.

Therefore, we developed a hybrid clustering approach in [14] which is based on the idea to partition all available shreds according to geometric information like edge lengths, edge angles and areas extracted from the snippets. For each page of a given format all of these three properties are known, e.g., a DIN A4 page is $210\,\mathrm{mm} \times 297\,\mathrm{mm}$ large which results in an area of $62\,370\,\mathrm{mm}^2$. (Trivially the sum of two matching edge angles is still $180°$.) Unfortunately, we cannot be sure that all pages of the original document(s) were torn using the same number of tears. Therefore, we developed an estimation algorithm which tries to guess the number of pages torn with two, three, four, and five tears, respectively. We assumed that it is very unlikely that a page is destroyed with six or more tears, since in this situation the size of snippets becomes relatively small while the height of the snippet stack gets rather large, such that further tearing operations cannot be easily performed.

To achieve the requested partitioning of the available snippets we apply a *variable neighborhood search* (VNS) based approach, which embeds a *variable neighborhood descent* (VND) as local search procedure. As optimization criterion we decided again to minimize the deviations from the desired document page characteristics. A detailed presentation and description of the used objective function is given in [14].

In contrast to a standard VND approach we did not define multiple moves which are then used to define multiple neighborhoods but only swap moves, i.e., the exchange of two snippets from two different sets, are utilized. Different neighborhoods are then built based on the characteristics of the snippets to be swapped, e.g., one neighborhood consists of all solutions for which a predefined snippet is removed from its current page while another neighborhood contains all solutions for which the characteristics of a predefined page are improved. In addition for systematically changing between the defined neighborhoods we also exchange in our VND variant the step functions to be used for examining the neighborhoods. Namely we select that move which maximally improves the solution with respect to the edge lengths, the edge angles, or the areas, respectively, while all other features must not worsen below a given threshold.

So far this method follows the basic principles of a standard clustering approach. For a second phase of the hybrid approach, we integrated the exact approach of Schüller presented in the previous section in each iteration of the proposed VNS method. Due to

Algorithm 9: HybridClustering

```
Input: Set $\mathcal{S}$ of snippets
Output: Assignment of snippets in $\mathcal{S}$ to pages, i.e., a clustering.
Data: (intermediate) solutions $x$, $x'$, $x''$
/* guess right number of pages and compute initial solution       */
$x \leftarrow$ initial solution;
/* apply standard VNS/VND based on simple swapping operations     */
$x' \leftarrow$ standard VNS/VND on $x$;
/* apply hybrid VND using exact approach for selecting snippets to
   be moved                                                        */
$x'' \leftarrow$ hybrid VND on $x'$;
return $x''$;
```

the fact that the exact reconstruction of borders of single page instances can be done in less then 0.05 seconds on average even for instances with five tears, we are able to solve the given subproblems each time a swapping operation of snippets from one page to another page was performed. Obviously, we favor those snippets as swapping candidates for the next iteration which correct the largest error when reconstructing the document borders. The concept of the complete approach is shown in Alg. 9.

With respect to the experimental tests we refer for a detailed listing to [14]. To summarize the obtained results it can be seen that throughout the algorithm a constant improvement in the solution quality, which is measured by the percentage of correctly assigned snippets to pages, could be achieved. Nevertheless, this approach is again limited to relatively small instances with up to ten different original document pages. This is, however, not only implied by increasing runtimes but by dramatically decreasing solution qualities. Nevertheless, this approach is a first step towards the application of automatic document reconstruction systems on (very) small real-world instances.

4.1.4. Fragment Stack Analysis

Under the assumption of properties **(P III)** and **(P IV)** it can be observed that as long as the snippets are not shuffled a predefined relative order must exist within the resulting stack of snippets, cf. [27]. Under this circumstance it is rather easy to precalculate which pairs of snippets have to be matched with each other, see also Fig. 4.2. In fact, the only problem that remains is computing the actual alignment of two remnants. For a solution to this problem we refer to Sec. 4.1.5.

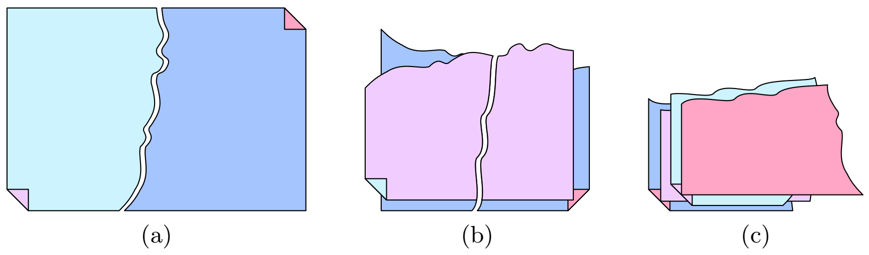


Figure 4.9.: Flipping of stacks during tearing. During both tearing operations the left stack is flipped.

Nevertheless, there are still some further problems that arise in combination with this approach: First of all, it has to be guaranteed that the relative order of the snippets is never changed—neither during the tearing process, nor during the on-site recovery, nor during transportation, etc. This implies that even the slightest unthoughtfulness of any person involved in the reconstruction process leads to major difficulties. Even if all participating persons take care, it cannot be guaranteed that the relative order of the snippets was not randomly twisted during the tearing operations. In fact, it can be observed that one human tears several paper documents using most likely always the same strategy but this strategy can vary from person to person. For example, it might occur that the two stacks resulting after a tearing operation are not put directly on each other but one (or even both) is flipped, cf. Fig. 4.9.

In addition, the problem complicates when multiple pages are either torn at the same time or the snippets of multiple pages are part of the current stack. Furthermore, any missing snippet represents a major problem. De Smet discusses these and related problems in [27]. In that work, he also suggests some algorithms and references for handling some of the related problems. Nevertheless, approaches like this might be interesting for small sized instances for which the on-site recovery is perfect, i.e., almost no shuffling of snippets occurs, but in most cases there will be too many concurrently arising issues.

4.1.5. Computer Vision Techniques

A complimentary approach to the above presented approaches is the somehow "natural" method applied by humans when trying to reassemble a set of manually torn paper documents. Humans do not measure the lengths of edges to be put together or compute the areas of the reconstructed document pages but they simply decide whether two snippets match with each other based on the contours or—to be more precise—on parts of the contours of the snippets. In addition, the rotation of the snippets is mainly derived based on the written text or other contextual information to be found on the snippets. Mapping this human approach to (semi-)automatic computer aided reconstruction systems is, however, not easy. In [28], De Smet *et al.* present some basic methods to be performed when trying the develop an approach based on computer vision techniques.

First of all it is necessary to preprocess the input data, i.e., the physical snippets, in such a way that they are digitized. This can be done by placing the snippets on a flatbed image scanner while using a uniformly colored background. Obviously, the background color should not appear somewhere on the snippets to be scanned. After scanning, the snippets have to be extracted from the background and their contours need to be computed. While De Smet *et al.* did not try to rotate the snippets during this step such that the final orientation is already determined it is imaginable that an orientation detection procedure as presented in [8] or [89] is applied.

Based on the computed contours it is possible to detect corner snippets and border snippets. In the next step, for each remnant specific features are computed which are then stored clock-wise. The matching of two snippets is then done by finding among all pairs of remnants that one which contains the best matching set of features for a determined part of the contours. The global reconstruction is then based on the idea to iteratively match a pair of snippets and continue this process by using this matched pair as if it was only one shred.

Although no detailed results are given in [28], it is indicated that the results are promising but far from good. Therefore the integration of user feedback is proposed such that the final decision(s) whether or not two snippets match are made by the human. Nevertheless, this approach shows that especially the digitalization of the physical shreds is manageable and useful information can be extracted which in turn can be utilized for refining the objective function to be used in combinatorial optimization approaches.

In [68], Justino *et al.* propose a similar approach to that of De Smet. They present, however, also some performance figures indicating that the computation times dramatically increase with the increasing number of shreds. Furthermore, even for ten snippets the performance of their approach already drops below 80%, i.e., only 80% of all matchings are correct.

4.2. Strip Shredded Text Documents

As indicated in Sec. 4.1 most people destruct paper documents by manually tearing them. In offices with documents containing more sensible data it is, however, convenient to use physical devices—so-called shredders. Such a machine consists of feed cylinders and blades cutting the paper documents into rectangular snippets. Depending on the concrete shredder model, the snippets are either strips, i.e., the rectangle's height is equal to the height of the original document, or rectangles with heights smaller than the document's page height. Within this section, we assume that strip shredding devices were used for cutting the documents. With respect to the widths of the rectangles no concrete assumptions are met. Even more, for the proposed approaches the width of the strips is disregarded and can therefore be chosen arbitrarily. Nevertheless, due to the concrete implementation of strip shredders all strip will in general have the same width.

These physical properties of strip shredders imply that—in contrast to the reconstruction of manually torn documents—the shape of the remnants cannot be exploited when trying to restore the original document(s). Therefore, other evaluation methods need to be implemented which do not depend on the shape of the snippets but on the information held on the snippets front and back faces.

The remainder of this section is structures as follows: We will first give an overview on methods published in literature for tackling the reconstruction of strip shredded (text) documents. Then we will present a formal definition of this problem including an $\mathcal{NP}$-hardness proof followed by the introduction of so called error estimation functions and the *concept of quality* for being able to objectively evaluate the proposed methods. The discussed approaches will include a reformulation of the reconstruction of strip shredded text documents as traveling salesman problem, a variable neighborhood search based method and an approach for computing lower bounds via Lagrangian relaxation. This section is completed by a discussion of related issues.

4.2.1. Related Work

Although the *reconstruction of strip shredded text documents* (RSSTD) is of major interest for intelligence agencies as well as forensics, there exists little (scientific) work related to this topic. Obviously, the reconstruction of strip shredded documents differs from other reconstruction fields like the reconstruction of manually torn paper documents (see Sec. 4.1) or jigsaw puzzle solving (see for example [21]) mainly by the fact that the shapes of the shreds cannot be exploited. Therefore, the most methods known so far rely on pattern recognition and image processing techniques.

Skeoch [125] focuses on the reconstruction of strip shredded documents by mainly discussing issues related to the scanning process and properties caused by the structure of paper (strips). She also presents a genetic algorithm including crossover and mutation operators as well as heuristics for generating initial solutions to restore shredded images. In contrast to text documents, a large amount of different colors usually exists in images and soft color transitions dominate. This aspect can be efficiently exploited.

In [133], Ukovich *et al.* try not to reconstruct the original document pages but to build clusters of strips belonging to the same sheet of paper by using MPEG-7 descriptors for this task. In [134] they introduce additional features like background and text color, line spacing and the number of lines to be extracted and exploited when reconstructing strip shredded text documents and finally discuss the potential of these clustering methods. However, the original documents still need to be reconstructed by hand (or using other methods).

De Smet *et al.* [29] discuss in their work mainly the principle design of (semi-)automatic reconstruction systems. They also present some basic approaches for determining text lines on shreds, top and bottom margins as well as empty or useless shreds. Furthermore a method is proposed based on these features to join pairs of matching strips into larger ones as well as a heuristic iterating this process until only one strip representing the (reconstructed) document is left. However, no performance results are presented in this work and only a basic outlook on future research is given.

In his thesis [95], Morandell, a former master's student at our institute, summarizes our work on metaheuristic methods based on the observation that RSSTD can be (re-)formulated as combinatorial optimization problem. Beside the analysis of different alternatives for the objective function, optimization approaches based on iterated local search, variable neighborhood search and simulated annealing are presented. Results documented are promising and represent the base of the approaches described in the following.

4.2.2. Formulation as Combinatorial Optimization Problem

For the *reconstruction of strip shredded text documents* (RSSTD) we are given a finite set of rectangularly shaped and (almost) equally sized paper snippets—so-called strips—which have been produced by shredding one or more sheets of paper. In this thesis the widths of the strips are not further investigated since no information to be exploited in our approaches can be extracted from them. Furthermore, the heights of all strips are assumed to be the same. If this is not the case, then a preprocessing step using clustering methods as proposed in [134] can be performed such that (smaller) subproblems are generated consisting of (sub)sets of strips having all the same heights which are then solved independently from each other.

Although many office printers are capable of duplex printing nowadays, most documents—especially in offices, one of the main application areas of shredders—are still blank on the back face. Motivated by this observation and for simplicity our presented model only regards the front face of the scanned strips. However, an extension to handle two-sided documents is possible in a straightforward way.

Our methods to be presented in the following solely focus on the information held on the borders of the strips, cf. Sec. 4.2.4. Therefore we neglect all strips with completely blank faces as well as strips with blank borders but non-empty inner regions. Beside the fact that the number of shreds to be regarded during the reconstruction process is reduced this blank strip elimination procedure removes symmetries implied by arbitrarily swapping blank strips, i.e., the search space is significantly reduced. For modeling the circumstance that in general paper documents have white margins at their left and right boundaries an additional virtual strip is added to the input of any instance which finally results in a finite set $\mathcal{S} = \{1, \ldots, n\}$ of (almost) equally sized, rectangular shreds forming the output of a shredding process of one or more pages of paper documents. Whereas shreds 1 to $n-1$ are non-empty, we request that the virtual shred n is blank. The basic idea of this modeling is that the first, i.e., the leftmost, (non-blank) shred is placed right next to n while the last, i.e., the rightmost, shred is placed left to n yielding a cycle. The virtual shred is therefore something like a connector between the first and last shred marking the start and the end, i.e., the left and right edge, of a page.

A solution $x = \langle \pi, o \rangle$ to RSSTD consists of a permutation $\pi : \mathcal{S} \setminus \{n\} \rightarrow \{1, \ldots, n-1\}$ of the elements in set $\mathcal{S}$ as well as a vector $o = \langle o_1, \ldots o_n \rangle \in \{0,1\}^n = \mathcal{O}^n$ which assigns an orientation to each strip $i \in \mathcal{S}$:

$$o_i = \begin{cases} 0 & \text{if strip } i \text{ is to be placed in its original orientation,} \\ 1 & \text{if strip } i \text{ is rotated by } 180°. \end{cases} \tag{4.6}$$

In the following π_i, with $i \in \mathcal{S} \setminus \{n\}$, denotes the position of strip i according to π and $\pi_i = n$. Additionally, by s_k we denote the strip placed at position k, with $1 \leq k \leq n$, i.e., $\pi_i = k \Leftrightarrow s_k = i$, with $i \in \mathcal{S}$ and $1 \leq k \leq n$. Possibly empty (sub-)sequences of strips will be denoted by $\sigma = \langle s_k, \ldots, s_{k'} \rangle$, with $1 \leq k < k' \leq n$. Please note that the orientation of strip n is of no concrete impact since n is blank.

In the following we make use of a cost function $c(i, j, \omega) \geq 0$ to be defined later in detail It provides an approximate measure for the possible error made when two strips i and j appear side-by-side and are oriented according to $\omega = (o_i, o_j) \in \mathcal{O}^2$ in the reconstructed document. This implies that in case two shreds perfectly fit, the corresponding value of $c(i, j, \omega)$ will be low while in cases of matching two rather different borders the value of the error estimation function will be relatively high.

The overall objective of RSSTD is to find a solution, i.e., a permutation and a corresponding orientation vector, such that the following total costs, i.e., the estimated error made during reconstruction, are minimized:

$$\mathrm{obj}(x) = \sum_{k=1}^{n-1} \mathrm{c}(\pi_k, \pi_{k+1}, \omega) + \mathrm{c}(\pi_n, \pi_1, \omega) \tag{4.7}$$

One crucial task in solving RSSTD as stated above is a proper definition of the cost function $c(i,j,\omega)$. A detailed discussion on this topic is given in Sec. 4.2.4. In any case, an error estimation function used for RSSTD must have the property that $c(i,j,\omega) = c(j,i,\overline{\omega})$, with $i,j \in \mathcal{S}$, $\omega = (o_i, o_j) \in \mathcal{O}^2$ and $\overline{\omega} = (o_j^C, o_i^C)$, where $o_i^C = 1 - o_i$. This means rotating two strips and then swapping their positions must lead to the same error estimation.

4.2.3. Complexity Results

For this section we will define the decision variant of RSSTD (DRSSTD) as follows: Given an RSSTD instance and a integer β. The instance is answered with *yes*, i.e., is solved, if there is a permutation and orientation vector such that the arising costs are less than or equal to β.

We will now show that DRSSTD is $\mathcal{NP}$-complete by reducing the decision variant of the (symmetric) *traveling salesman problem* (TSP) to DRSSTD. In addition, it is obvious that DRSSTD is in $\mathcal{NP}$ since an instance can be solved in non-deterministic polynomial time by guessing a permutation and orientation vector and then verifying them in polynomial time.

We define an instance of the decision variant of TSP (also denoted by DTSP) as follows (see also [43]): Given is a set $\mathcal{C}$ of n cities and distances $d(c_i, c_j) \in \mathbb{Z}^+$ for each pair of cities $c_i, c_j \in \mathcal{C}$. Furthermore, we are given a positive integer constant B. An instance of this decision problem is answered with *yes*, i.e., is solved, if there is a tour of all cities in $\mathcal{C}$ having length B or less, i.e., a permutation $\langle c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(n)} \rangle$ of $\mathcal{C}$ is searched, with

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)}) \le B \tag{4.8}$$

and $c_{\pi(i)}$ denoting the i-th city along the computed tour.

Such an instance of DTSP can now easily be transformed in polynomial time using the following algorithm:

1. For each city $c_i \in \mathcal{C}$ introduce a strip $i \in \mathcal{S}$.
2. Without loss of generality assume the city c_n corresponds to strip n.
3. Further, let us assume that the value of the error estimation function $c(i,j,\omega)$ is set to $d(c_i, c_j)$, for all $\omega \in \mathcal{O}^2$, $i,j \in \mathcal{S}$, $c_i, c_j \in \mathcal{C}$.
4. Set $\beta = B$.

This transformation has an effort of $O(n)$, i.e., linear in the number of cities of the instance of DTSP. Obviously, any solution to a so generated DRSSTD instance can be transformed into a solution to the original DTSP instance. Furthermore, the DRSSTD instance is answered with *yes*, if and only if the corresponding DTSP instance is answered with *yes*. Therefore, any algorithm for DRSSTD also solves DTSP. This implies that DRSSTD is $\mathcal{NP}$-complete. Based on the above transformation any TSP instance can be transformed into it a RSSTD instance with same optimal value which implies that RSSTD cannot be approximated by a constant factor unless $\mathcal{P} = \mathcal{NP}$, cf. [122].

4.2.4. Error Estimation Function

One crucial point in RSSTD is the definition of an appropriate cost function $c(i, j, \omega)$ for estimating the error made when placing two strips i and j, with $i, j \in \mathcal{S}$, next to each other under their given orientations $\omega \in \mathcal{O}^2$. There are several different ways on how this can be done (see also [95] on this topic), and none will be perfect in every possible situation. Nevertheless, there are some properties which can (and should) be demanded when specifying such a cost function.

First of all it must be regarded that in the best case two shreds are correctly matched with each other, i.e., no error is made at all. In that case we expect to associate a cost value of zero. In all other cases, i.e., non-perfect matches, values greater than zero should be assigned. That is,

$$c(i, j, \omega) \geq 0 \qquad \forall i, j \in \mathcal{S}, \omega \in \mathcal{O}^2 \tag{4.9}$$

must hold. In addition, it must be guaranteed that equation (4.10) is satisfied:

$$c(i, j, \omega) = c(j, i, \overline{\omega}) \qquad \forall i, j \in \mathcal{S}, \omega \in \mathcal{O}^2 \tag{4.10}$$

This means that placing strip i left to strip j yields the same error as placing shred j left to shred i but both rotated by 180°.

In contrast to pattern recognition based approaches like presented in [133, 131, 134, 29] we propose an evaluation function which does not rely on the information primarily imposed by the inner regions and features of the strips but we solely want to exploit (color) information on the borders of the strips. For this purpose we denote the number of (image) pixels along the y-axis of a strip $i \in \mathcal{S}$ as h_i. As it is unlikely that the images of two strips i and j with the same physical height and scanned with the same resolution will significantly differ in h_i and h_j. We therefore assume for this work $h_i = h_j$ holds for all strips $i, j \in \mathcal{S}$ and thus will omit the indices, i.e., write h for short.

To simplify the next definitions, we consider eventual rotations of strips in the following as already performed; i.e., when speaking about the left side of a strip $i \in \mathcal{S}$ for which

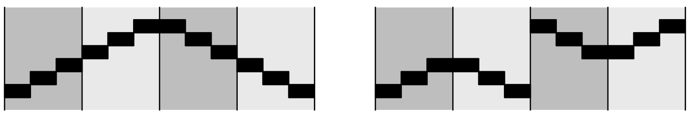

(a) (b)

Figure 4.10.: Two possible reconstructions of a document with four strips. Both of these two solutions might be correct, but as a human reader one expects (a) to be correct.

$o_i = 1$, we actually refer to its original right side. The pixels on the left or right edge are those pixels which form the left or right border, respectively.

Since the majority of text documents are composed of black text on (almost) white background and we mainly focus on the reconstruction of text documents, we only consider black-and-white (B/W) image data as input here. In fact, preliminary tests have shown that the usage of finer grained color or gray-scale information does not significantly increase the quality of the solutions obtained by our approaches. We remark, however, that in cases where documents contain a significant amount of different colors or gray values, an extension of our model might be meaningful and can be achieved in a more or less straightforward way.

Let $v_\mathrm{l}(i, y, o_i), v_\mathrm{r}(i, y, o_i) \in \{0, 1\}$ be the B/W values of the y-th pixel, with $1 \leq y \leq h$ at the left and right borders of strip $i \in \mathcal{S}$ under orientation o_i, respectively. Then, the first and most straightforward approach for defining a cost function $c_1(i, j, \omega)$ is by simply iterating over all pixels on the right border of strip i and compare them to the corresponding pixels on the left border of strip j. Since we defined RSSTD as a minimization problem the value of $c_1(i, j, \omega)$ is increased by one if two corresponding pixels do not have the same values:

$$c_1(i, j, \omega) = \sum_{y=1}^{h} \left| v_\mathrm{r}(i, y, o_i) - v_\mathrm{l}(j, y, o_j) \right| \tag{4.11}$$

Although the evaluation of this cost function (4.11) can be efficiently done, there are some special cases in which the human intuition would result in another preferred solution. For an example see the case depicted in Fig. 4.10. Although both solutions shown might be the correct one, it is intuitively more likely that the reconstruction in Fig. 4.10a corresponds to the original document. Therefore, we want the alignment 4.10a to receive a better error estimation, i.e., a lower value, than arrangement 4.10b. Hence, we adopt the idea presented in [10] which additionally considers the values of two pixels above and two pixels below the currently evaluated position. It is then the weighted average of these five pixel values compared with the corresponding average value on the other strip:

$$c_2(i,j,\omega) = \sum_{y=3}^{h_s-2} p(i,j,\omega,y) \tag{4.12}$$

$$p(i,j,\omega,y) = \begin{cases} 1 & \text{if } p'(i,j,\omega,y) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

$$\begin{aligned} p'(i,j,\omega,y) = \Big| \; & 0.7 \cdot v_r(i,o_i,y) - 0.7 \cdot v_l(j,o_j,y) \\ & + 0.1 \cdot \big(v_r(i,o_i,y+1) - v_l(j,o_j,y+1)\big) \\ & + 0.1 \cdot \big(v_r(i,o_i,y-1) + v_l(j,o_j,y-1)\big) \\ & + 0.05 \cdot \big(v_r(i,o_i,y+2) + v_l(j,o_j,y+2)\big) \\ & + 0.05 \cdot \big(v_r(i,o_i,y-2) + v_r(j,o_j,y-2)\big) \Big| \end{aligned} \tag{4.14}$$

Obviously the weighting factors and the threshold value τ used in the definition of $p(i,j,\omega,y)$ and $p'(i,j,\omega,y)$ have to be chosen carefully. Preliminary tests revealed, however, that the proposed definition in addition of setting $\tau = 0.1$ results in relatively high quality solutions—in particular, the special case depicted in Fig. 4.10 is also handled by this concrete parameter setting.

4.2.5. The Concept of Quality

As already shown in the previous section, there are different ways for defining an error estimation function for RSSTD. For any objective function it is, however, possible to create a worst case scenario which results in optimal solutions with respect to the objective value, i.e., in solutions having a minimal estimated error, but does not correspond to the original document. For such a worst cases scenario we refer to Fig. 4.10. Obviously, an objective function is searched for which optimal solutions correspond to the actually originally documents in as many situations as possible. Therefore, it is desirable to directly compare two (different) objective functions with each other. This leads to the *concept of quality* which should serve as a measure on how good a particular error estimation function is (with respect to a given set of RSSTD instances).

For this purpose, we proposed in [105] a pseudometric $Q(x)$ for any solution x of RSSTD. The basic concept of $Q(x)$ is to count the number of correctly reconstructed subsequences of strips with respect to the original document. This implies that a value of 1 corresponds to an entirely correctly restored document while values greater than 1 indicate that some placements of strips with respect to their neighbors and/or rotations are wrong. Obviously, $1 \leq Q(x) \leq n$ holds for any solution x, with n being the number of strips.

Empirical tests based on the results obtained with the reconstruction methods proposed in this thesis revealed that humans are most likely able to decode, i.e., read, documents with a quality of five or lower. Furthermore, it is easy to reconstruct the original document by hand as soon as only five subsequences of strips have to be (correctly) rearranged.

4.2.6. Solving RSSTD via Reformulation as a Traveling Salesman Problem

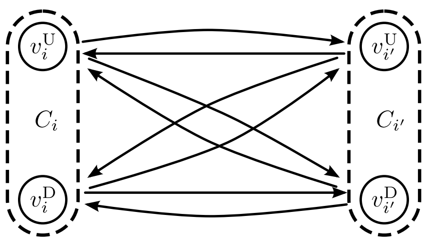
Motivated by the observation that TSP is strongly related to RSSTD one approach for solving RSSTD is to apply well-known methods for solving TSP to transformed RSSTD instances. For this purpose, we present a polynomial transformation procedure translating any RSSTD instance into a TSP instance. Anyhow, although the reverse transformation, i.e., from TSP to RSSTD, can be achieved directly, cf. Sec. 4.2.3, for this procedure an intermediate step is necessary. Therefore, we first represent RSSTD as an *asymmetric generalized traveling salesman problem* (AGTSP) which can then be transformed to TSP. We will, however, show that in contrast to the standard transformation from AGTSP to TSP in our case it is not necessary to introduce additional nodes which is caused by the special structure implied by the error estimation function, cf. Sec. 4.2.4.

Formulation as Asymmetric Generalized Traveling Salesman Problem

In the *asymmetric generalized traveling salesman problem* (AGTSP) a directed graph $G = (V, A)$, with V being the set of nodes and A being the set of arcs, as well as a partitioning of V into m disjoint, non-empty clusters C_i, $i = 1, \dots, m$, is given. Furthermore, a weight $w_a \geq 0$ is associated with each $a \in A$. An optimal solution to AGTSP is a tour $T \subseteq A$ that visits exactly one node of each cluster C_i while minimizing the objective function $\sum_{a \in T} w_a$, cf. [54].

The following steps have to be performed for representing RSSTD as AGTSP:

1. For each strip $i \in \mathcal{S} \setminus \{n\}$, introduce a cluster C_i consisting of two vertices v_i^{U} and v_i^{D} representing the possible orientations of the corresponding strip i.
2. Introduce a cluster C_n for the (virtual) blank strip n and insert one vertex v_n into this cluster. Since n is blank no orientation information is necessary for this strip.
3. Each pair (i, j) of strips, with $i, j \in \mathcal{S} \setminus \{n\}$ and $i \neq j$, induces eight arcs representing the possible placements of i and j in relation to each other, see also Fig. 4.11a. For example, arc $(v_i^{\mathrm{D}}, v_j^{\mathrm{U}})$ represents the case that strip i is placed right to strip j. While strip i is rotated by 180°, strip j is positioned upright. Since any strip



(a)

	v_i^{U}	v_i^{D}	$v_{i'}^{\mathrm{U}}$	$v_{i'}^{\mathrm{D}}$
v_i^{U}	–	0	a	b
v_i^{D}	0	–	c	d
$v_{i'}^{\mathrm{U}}$	d	b	–	0
$v_{i'}^{\mathrm{D}}$	c	a	0	–

(b)

	v_i^{U}	v_i^{D}	$v_{i'}^{\mathrm{U}}$	$v_{i'}^{\mathrm{D}}$
v_i^{U}	–	0	$c+M$	$d+M$
v_i^{D}	0	–	$a+M$	$b+M$
$v_{i'}^{\mathrm{U}}$	$c+M$	$a+M$	–	0
$v_{i'}^{\mathrm{D}}$	$d+M$	$b+M$	0	–

(c)

Figure 4.11.: In (a) a subgraph representing two strips s and s' in an AGTSP instance is depicted while in (b) the same subgraph after performing the transformation to TSP is shown. The bold lines indicate two corresponding tours.

$i \in \mathcal{S}$ cannot be placed left (or right) to itself, it is obvious that there are no arcs between two nodes representing the same strip.

4. Additionally, vertex v_n is connected via two oppositely directed arcs with each other node representing a strip.

5. The weights of the arcs are chosen such that for any arc $a = (v_i^{o_i}, v_j^{o_j})$, with $i, j \in \mathcal{S} \setminus \{n\}$, $w_a = \mathrm{c}(i, j, \omega)$, with $\omega = (o_i, o_j)$. The weights for arcs leaving or entering v_n are chosen according to $\mathrm{c}(n, i, \omega)$ or $\mathrm{c}(i, n, \omega)$, respectively.

Obviously, an optimal solution to the AGTSP instance derived using this algorithm also forms a solution to the original RSSTD instance with equal costs when starting the tour at the virtual strip represented by v_n.

There are several methods for solving AGTSP directly, e.g., see [39] for a branch-and-cut algorithm or [124, 65, 54] for various heuristic approaches based on genetic algorithms, memetic algorithms or variable neighborhood search combined with dynamic programming techniques. Beside applying one of those algorithms specifically designed for solving AGTSP another possibility is to further transform an AGTSP instance into a classical TSP instance and solve the latter with one of the many existing methods. In the next section we concentrate on such an approach.

Further Reformulation as TSP

The classical, i.e., symmetric, TSP consists of finding the shortest tour in a weighted undirected graph $G = (V, E)$ such that each vertex in V is visited exactly once. Let $w_e \geq 0$ be the weight associated with each edge $e \in E$. The length of a tour in TSP is computed as the sum of the tour's edge weights.

Based on the presented transformation of RSSTD to AGTSP, RSSTD can be further translated into a TSP by first applying the polynomial time transformation into an *asymmetric traveling salesman problem* (ATSP) proposed in [12] and finally applying the polynomial transformation of ATSP into TSP described in [82]. Taking a closer look at these works, two major drawbacks can be identified. On one hand, the maximum costs for edges are dramatically increased by a value in $O(m \cdot \sum_{e \in E} w_e)$ during the transformation from AGTSP into ATSP, which might lead to practical problems when trying to solve such transformed instances. On the other hand, the number of nodes in G is doubled during the translation from the asymmetric TSP to the symmetric case. Due to the nature of TSP this has a substantial impact on the running times when computationally solving these TSP instances. Fortunately, both drawbacks can be avoided or at least reduced when applying the following transformation method which adopts the idea of first introducing directed cycles of zero costs within each cluster while modifying the (costs of the) outgoing arcs presented in [12]:

1. Add two additional arcs—one in each direction—between nodes v_i^{D} and v_i^{U} for each strip $i \in \mathcal{S} \setminus \{n\}$.
2. Set the weights of these new arcs equal to zero.
3. In a next step, swap the weights for $(v_i^{\mathrm{D}}, v_j^{\mathrm{D}})$ and $(v_i^{\mathrm{U}}, v_j^{\mathrm{D}})$ as well as $(v_i^{\mathrm{D}}, v_j^{\mathrm{U}})$ and $(v_i^{\mathrm{U}}, v_j^{\mathrm{U}})$, with $i, j \in \mathcal{S} \setminus \{n\}$ and $i \neq j$. After swapping two arcs we add a constant $M > 0$ to the associated arc weights.
4. Since the cluster C_n consists of only one node, no transformation needs to be done for this cluster.

In Fig. 4.11b the adjacency matrix of a subgraph of an AGTSP instance for RSSTD is presented. Fig. 4.11c depicts the adjacency of this subgraph after applying the transformation to TSP. It can be easily checked that the resulting graph is undirected.

Theorem 2. *Any weight-minimal Hamiltonian tour on a graph obtained by the presented transformation from RSSTD can be re-transformed into an optimal placement of strips with respect to objective function* (4.7)*.*

Proof. Due to the fact, that the costs for arcs connecting the nodes within a cluster are zero, any optimal tour will visit both nodes in a cluster consecutively. Assuming that there is one cluster C_k whose nodes are not visited consecutively, the tour has to enter cluster C_k at least two times. Since the costs for all arcs except for those within a cluster are equal to or greater than M, the costs of such a tour have to be greater than $(m+1) \cdot M$, with m being the number of clusters. Therefore, if M is chosen large enough, any tour, entering each cluster only once is cheaper. An appropriate value for M is $1 + m \cdot \max_{(i,j) \in \mathcal{S}^2} c(i, j, \omega)$, with $i, j \in \mathcal{S}$. Since each cluster is entered only once, we can decode the Hamiltonian tour as a permutation of the clusters which are representing the strips in RSSTD. Cluster C_n marks the beginning and the end of the strips' permutation. The orientation of each strip is set according to the node the cluster is entered by. If the first node visited in a cluster corresponds to the orientation 0 then the strip is oriented according to its original orientation in the corresponding solution. Analogously, orientation 1 is decoded. Furthermore, any optimal permutation π of strips can be transformed into an optimal tour T using the relationship described above. Assuming that there exists a tour T' with lower costs than T, we can transform T' into a permutation π' with lower costs than π, which contradicts the assumption that π is minimal. $\square$

Using this transformation of RSSTD to TSP it is now possible to apply any method for solving TSP. However, since the number of vertices is always twice the number of strips which on the other hand can be quite large, it cannot be expected that (current) exact algorithms will be applicable for (large sized) real-world instances of RSSTD. In addition, any error estimation function suffers from the fact that one cannot guarantee that all special cases are handled correctly. Therefore, we decided to use the implementation of the Chained Lin-Kernighan heuristic [7] by Applegate *et al.* [6] for solving the transformed RSSTD.

Experimental Results

To investigate the performance of such an approach we executed some computational tests. For this purpose, we shredded ten different pages using various strip widths, such that in total 50 instances were generated. The shredding process itself was done virtually, i.e., the strips are all sharp-cut. Although this approach is somehow idealistic it is also used by others, e.g., cf. [134]. Any comparison is therefore fair. Somehow more problematic is the selection of a set of pages to be used as (original) documents. We tried to choose ten pages with different characteristics. They contain among others, plain text, plain text with (multiple) headings, table of contents, figures and tables (including horizontal and vertical lines), and two-columned text. We refer to these pages as p01 to p10. For a pictorial illustration of the instances used we refer to App. A. For having the opportunity to compare with already published results found in literature we also used the data set provided by Ukovich *et al.* in [134]. This set consists of ten pages containing mainly type writer written text with hand written notes (in a different color). In contrast to the tests performed on p01 to p10, we adopted the settings of Ukovich *et al.* and tried to reconstruct all pages at the same time, i.e., instances based on this set contain multiple pages.

Detailed results of the test runs based on the transformation of RSSTD to TSP are shown in Tab. 4.1. The shown numbers represent averages over 30 runs, each, with standard

Table 4.1.: Average qualities of final solutions from the TSP solver comparing cost functions c_1 and c_2. Standard deviations are given in parentheses.

page	*time*	30 strips		50 strips		100 strips		150 strips		300 strips	
		c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2
p01	5 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	23.0 (0.0)	1.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	1.0 (0.0)	23.0 (0.0)	1.0 (0.0)
p02	5 s	2.9 (1.5)	1.0 (0.0)	7.6 (0.5)	8.9 (0.5)	16.3 (0.7)	14.8 (1.4)	38.1 (0.9)	34.3 (0.7)	105.7 (0.5)	80.2 (1.1)
	50 s	4.0 (0.0)	1.0 (0.0)	8.0 (0.0)	10.0 (0.0)	16.0 (0.0)	16.0 (0.0)	39.0 (0.0)	33.0 (0.0)	104.0 (0.0)	71.0 (0.0)
p03	5 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	16.0 (0.0)	1.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	14.0 (0.0)	1.0 (0.0)
p04	5 s	1.0 (0.0)	1.0 (0.0)	5.0 (0.0)	1.0 (0.0)	19.9 (0.6)	4.0 (0.0)	28.5 (0.7)	20.0 (0.0)	68.9 (0.6)	42.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	5.0 (0.0)	1.0 (0.0)	20.0 (0.0)	4.0 (0.0)	29.0 (0.0)	20.0 (0.0)	66.0 (0.0)	43.0 (0.0)
p05	5 s	1.0 (0.0)	1.0 (0.0)	4.3 (0.4)	1.0 (0.0)	12.0 (0.0)	6.0 (0.0)	45.0 (0.0)	8.0 (0.0)	83.2 (0.5)	30.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	1.0 (0.0)	12.0 (0.0)	6.0 (0.0)	39.0 (0.0)	8.0 (0.0)	84.0 (0.0)	25.0 (0.0)
p06	5 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	1.0 (0.0)
p07	5 s	16.9 (0.8)	17.0 (1.2)	34.3 (0.9)	33.1 (1.0)	76.5 (1.1)	68.5 (0.8)	126.8 (0.8)	116.3 (1.4)	258.2 (0.8)	244.8 (0.9)
	50 s	15.0 (0.0)	17.0 (0.0)	35.0 (0.0)	34.0 (0.0)	76.0 (0.0)	67.0 (0.0)	127.0 (0.0)	121.0 (0.0)	258.0 (0.0)	241.0 (0.0)
p08	5 s	1.9 (0.3)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.2)	1.4 (0.5)	1.4 (0.5)	3.0 (0.0)	3.0 (0.0)
	50 s	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	1.0 (0.0)	3.0 (0.0)	3.0 (0.0)
p09	5 s	2.0 (0.2)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)
	50 s	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)
p10	5 s	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	4.0 (0.0)	8.0 (0.0)	10.0 (0.0)	12.0 (0.0)	8.0 (0.0)	33.2 (0.4)	39.0 (0.0)
	50 s	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	4.0 (0.0)	8.0 (0.0)	10.0 (0.0)	12.0 (0.0)	8.0 (0.0)	33.0 (0.0)	40.0 (0.0)
m00	5 s	11.8 (1.1)	14.4 (1.0)	9.6 (0.7)	9.7 (0.4)	9.4 (0.7)	10.9 (0.3)	15.4 (0.6)	16.9 (0.4)	35.5 (0.9)	28.1 (1.3)
	50 s	11.0 (0.0)	13.0 (0.0)	9.0 (0.0)	10.0 (0.0)	8.0 (0.0)	11.0 (0.0)	15.0 (0.0)	17.0 (0.0)	34.9 (0.8)	31.3 (1.0)

deviations shown in parentheses. The code was primarily implemented in Java 6 using JNI (Java Native Interface) for calling the C implementation of the CONCORDE Chained Lin-Kernighan heuristic[1]. The results were obtained on a single core of an Intel® Core™2 Quad CPU with 8 GB RAM and 2.83 GHz.

The values presented in Tab. 4.1 should be read as follows: The first two columns indicate the page and the time provided to the Chained Lin-Kernighan heuristic for solving the corresponding TSP instance. Columns labeled with c_1 represent the qualities achieved while optimizing with respect to objective function c_1 (see Eq. (4.11) in Sec. 4.2.4) while the columns labeled c_2 correspond to the outputs obtained with respect to c_2 (cf. Eq. (4.12) in Sec. 4.2.4). Finally, always two columns—one labeled with c_1 the other with c_2—are superscribed with the number of strips the original document page was cut into. Rows labeled with m00 correspond to the instance set provided by Ukovich *et al.* [134].

It can be seen that the obtained results differ dramatically from instance to instance. For example, page p01 could always be solved to optimality in even five seconds using error estimation function c_2 while the smallest instance of page p07, i.e., with 30 strips, was

[1]Code available at `http://www.tsp.gatech.edu/concorde/`.

only solved to an average quality of 15 in 50 seconds using c_1. Obviously, this is motivated by the fact that, for instance, page p07 consists of a colorful image which is transformed into a B/W image. Choosing the wrong threshold value results in a document almost completely black. Nevertheless, with respect to those instances containing almost only text it can, for example, be observed that the average qualities obtained using c_2 are in most cases better or at least equal to the qualities obtained with respect to c_1. Even more, for instance p01 cut into 300 strips the second error estimation function led to the original document in all 30 runs (even in five seconds) while the runs based on c_1 only achieved an average quality of 23.

Please note that in some cases the qualities after five seconds are better than those obtained after 50 seconds. This is caused by the fact that due to the imperfectness of any error estimation function these instances are “overoptimized”, i.e., the original solution does not correspond to the solution having the best objective value. Altogether, the results lead to the conclusion the assumption that error estimation function c_2 suits better for B/W text documents than c_1.

Further, we want to discuss another interesting observation related to the results obtained for page p09 containing two-columned text. It seems that instances based on this page could (almost) never be solved to optimality. Actually, p09 was always completely restored but the order of the two columns was not correctly identified, i.e., the left text column was placed right next to the (originally) right column. The same observation also applies to page p02 which consists of a table of contents with horizontal lines of dots. Therefore, there are some strips having only dots printed on them. Symmetries are introduced by these strips and many with respect to any error estimation function optimal solutions (with different qualities) exist. Although in most cases humans might decide the correct order due to the information contained in the text, there are some cases for which even a human cannot make this decision—so neither an automatic system should be expected to correctly guess the alignment of these columns. Nevertheless, this implies that in some situations the integration of human intelligence into a (semi-)automatic reconstruction system might not only be valuable but even required.

Finally, we want to focus on the results obtained for the page set m00. For this set, the column labeled with “30 strips” indicates that each of the ten pages was cut into 30 strips, i.e., a total of 300 strips were realigned, which also relativizes the (rather bad) results in comparison to the other instances shown in the same column. A closer inspection of the result files shows that although a value of, for example, 14.4 for c_2 and 30 strips is obtained the single pages were reconstructed with qualities less than or equal to three. Please note, that with respect to the running times of five or 50 seconds, respectively, the results obtained for 300 strips, i.e., a total of 3000 strips, are particularly good. To be able to directly compare this approach with the results obtained by Ukovich *et al.* presented in [134], we also performed tests with 34 strips (not listed in the table), a time limit of five seconds and optimizing with respect to error estimation function c_2. The results obtained showed that in 16 of 30 runs all pages were optimally reconstructed, whereas in the remaining 14 runs only one page was reconstructed to quality 2 while all others were correctly restored. Considering the time limit of five seconds for the execution of the Chained Lin-Kernighan heuristic and additional ten seconds used for computing the error estimation function our method clearly outperforms that from Ukovich *et al.* presented in [134]—especially under the consideration of the fact that the approach by Ukovich *et al.* only identifies strips to be very likely on the same original document page while no concrete alignment of these strips is obtained.

4.2.7. Solving RSSTD via Variable Neighborhood Search and Human Interaction

As already indicated in the previous section, even the "most precise" cost function and an exact solution of our RSSTD model will not always yield a correct arrangement fully representing the original document before destruction. The reason is that the cost function only is an (approximate) indicator for the likelihood of two strips appearing next to each other. However, documents also may contain unlikely scenarios. Furthermore, text may be arranged in columns with empty parts in between. It is then impossible to find the correct order of the separated text blocks without having more specific knowledge of the documents content. Additionally applying heavier pattern recognition and knowledge extraction techniques might be feasible for certain applications but will also dramatically increase running times.

Instead, we leverage here the power of human knowledge, experience, and intuition in combination with a variable neighborhood search metaheuristic. When confronted with a candidate solution, a human often can decide quite easily which parts are most likely correctly arranged, which strips should definitely not be placed side-by-side, or which parts have a wrong orientation.

The idea of systematically integrating human interaction in an optimization process is not new. Klau *et al.* [79, 77, 78] give a survey on such approaches and present a framework called *Human Guided Search* (HuGS). The implementation is primarily based on tabu search, and the success of this human/metaheuristic integration is demonstrated on several applications.

Variable Neighborhood Search in HuGS

Since preliminary tests for solving RSSTD with tabu search as implemented in the HuGS framework [77] did not convince, we considered also other metaheuristics and finally decided to use a *(general) variable neighborhood search* (VNS) with embedded *variable neighborhood descent* (VND) for local improvement. In addition to a standard VND/VNS approach we incorporated user actions into the search procedure such that the final decision on the quality of an obtained solution is made by humans.

For this approach, we represent a solution to RSSTD by three arrays corresponding to the strips permutation π, a vector p storing the position for each strip, and the orientation vector o. Note that π and p are redundant, but the evaluation of the neighborhoods to be discussed in the following can be more efficiently implemented when both are available.

Neighborhoods for VNS and VND

Several different move types are used within VND and VNS. The most intuitive move is called *shifting* (SH) and simply shifts one strip by a given amount to the right or left. More formally it can be written as

$$\mathrm{SH}(\sigma_1 \cdot \langle i \rangle \cdot \sigma_2 \cdot \langle j \rangle \cdot \sigma_3, i, j) = \sigma_1 \cdot \langle j \rangle \cdot \langle i \rangle \cdot \sigma_2 \cdot \sigma_3 \tag{4.15}$$

or

$$\mathrm{SH}(\sigma_1 \cdot \langle j \rangle \cdot \sigma_2 \cdot \langle i \rangle \cdot \sigma_3, i, j) = \sigma_1 \cdot \sigma_2 \cdot \langle i \rangle \cdot \langle j \rangle \cdot \sigma_3 \tag{4.16}$$

with $i, j \in \mathcal{S} \setminus \{n\}$. In this (and the following) context σ_k denotes a possibly empty subsequence of strips. A second move, called *swapping* (SW), is defined by swapping two arbitrary elements with each other. In a formal matter, this can be written as

$$\mathrm{SW}(\sigma_1 \cdot \langle s_k \rangle \cdot \sigma_2 \cdot \langle s_{k'} \rangle \cdot \sigma_3, k, k') = \sigma_1 \cdot \langle s_{k'} \rangle \cdot \sigma_2 \cdot \langle s_k \rangle \cdot \sigma_3 \tag{4.17}$$

with $1 \leq k < k' \leq n-1$. Both moves, shifting and swapping, can be extended to block moves. In the latter case, called *block swapping* (BS), this results in a move swapping two arbitrarily long, non-overlapping subsequences of strips with each other. The other block move, namely *block shifting*, is equivalent to swapping two adjacent blocks with each other. Therefore, it is not explicitly defined in our environment. A block swap move can be formally written as

$$\begin{aligned}\mathrm{BS}(\sigma_1 \cdot \langle s_k, .., s_{k+m} \rangle \cdot \sigma_2 \cdot \langle s_{k'}, .., s_{k'+m'} \rangle \cdot \sigma_3, k, m, k', m') = \\ \sigma_1 \cdot \langle s_{k'}, .., s_{k'+m'} \rangle \cdot \sigma_2 \cdot \langle s_k, .., s_{k+m} \rangle \cdot \sigma_3 \end{aligned} \tag{4.18}$$

with $1 \leq k < k+m < k' < k'+m' \leq n-1$. In addition to this four move types related to the assignment of strips to positions, two further moves for changing the orientation

Table 4.2.: Neighborhood structures defined for VND.

neighborhood structure	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\mathcal{N}_5$
move type	R	SH	SW	BR	BS
size of $\mathcal{N}_x$	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^4)$

of a strip or a block of strips, called *rotating* (R) and *block rotating* (BR), respectively, are defined. Rotating simply rotates one strip by 180°, while block rotating executed on positions k to k', with $1 \leq k < k' \leq n-1$ first rotates all strips in this interval and in a second step swaps strips at positions k and k', $k+1$ and $k'-1$, and so on. Using incremental evaluation schemes each presented move can be evaluated in constant time.

In our VND, the five neighborhood structures induced by our moves are considered in the order shown in Table 4.2, thus, sorted by their sizes. As step function *best improvement* as well as *next improvement* have been implemented. For shaking in VNS, k random swap moves, with $1 \leq k \leq 5$, are performed. As initial solution either a random solution or a solution provided by the Lin-Kernighan based approach is used (details are given in the experimental results section).

User Interactions

For the integration of user interaction into the optimization process a set of valid user moves has to be defined. All previously described move types are contained in this set of allowed user actions. Additionally, the user can

- forbid “wrong” neighborhood relations between pairs of strips;
- lock “correct” subsequences of strips, which are concatenated and in the further optimization process considered as atomic *meta-strips*;
- lock the orientation of strips.

All of these actions also can be reverted, should the user reconsider his earlier made decisions. Our extensions of the HuGS framework provide an easy and intuitive way to visualize candidate solutions, perform the mentioned user actions, or to let VNS or the Lin-Kernighan based approach continue for a while.

A main advantage of integrating human power into the search procedure is in fact that with each additional lock of strips or forbidden neighborhood relation the solution space is pruned. For example, locking two neighboring strips into a meta-strip reduces the number of valid solutions to $1/m$, where m is the number of yet unmerged strips.

Table 4.3.: Average qualities of final solutions from the TSP solver comparing cost functions c_1 and c_2. Standard deviations are given in parentheses.

page	*step*	30 strips		50 strips		100 strips		150 strips		300 strips	
		c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2
p01	*next*	6.7 (6.5)	3.9 (5.9)	22.5 (13.5)	12.1 (13.5)	66.0 (5.6)	53.8 (19.5)	105.9 (2.1)	102.5 (5.3)	216.0 (1.5)	214.4 (2.3)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.1 (0.4)	1.0 (0.0)	59.3 (10.6)	34.2 (11.6)	160.0 (20.1)	96.5 (38.7)
p02	*next*	8.2 (4.5)	5.3 (4.9)	29.8 (3.7)	27.8 (5.1)	64.8 (2.3)	64.1 (2.3)	97.6 (1.4)	97.4 (1.5)	192.3 (1.7)	192.3 (1.1)
	best	3.6 (2.1)	1.4 (0.7)	13.7 (3.5)	11.8 (2.7)	36.6 (5.1)	31.1 (6.4)	64.5 (5.6)	56.5 (7.1)	177.7 (4.6)	171.7 (7.8)
p03	*next*	3.8 (5.1)	1.9 (3.2)	11.7 (12.7)	11.5 (13.0)	61.6 (15.1)	44.0 (29.2)	100.5 (10.6)	94.8 (19.3)	215.0 (1.8)	203.6 (17.4)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	2.4 (5.2)	1.0 (0.0)	64.3 (22.7)	42.7 (23.1)	169.0 (40.7)	104.9 (58.2)
p04	*next*	3.7 (3.7)	1.8 (2.4)	19.7 (6.8)	16.8 (7.1)	53.9 (4.3)	52.4 (5.7)	84.5 (2.2)	85.1 (1.4)	173.5 (1.2)	173.1 (1.0)
	best	1.1 (0.6)	1.2 (1.3)	7.5 (2.5)	1.7 (1.7)	25.3 (5.9)	13.1 (5.6)	54.2 (6.1)	36.5 (8.3)	157.9 (6.6)	151.6 (7.6)
p05	*next*	1.6 (3.1)	3.1 (4.1)	12.0 (11.1)	10.6 (14.0)	40.7 (26.4)	41.3 (28.4)	104.9 (10.7)	93.0 (24.5)	216.5 (3.8)	215.2 (4.2)
	best	1.0 (0.0)	1.0 (0.0)	5.4 (2.8)	1.0 (0.0)	17.3 (6.9)	6.0 (0.2)	102.9 (3.4)	91.0 (15.0)	207.6 (7.9)	199.2 (7.9)
p06	*next*	9.0 (8.5)	8.7 (8.5)	24.7 (15.1)	25.9 (14.9)	84.6 (12.6)	83.9 (13.5)	139.9 (2.0)	138.6 (3.9)	282.6 (1.2)	281.6 (4.4)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.5 (2.6)	110.6 (20.5)	87.0 (33.1)	247.6 (11.6)	248.0 (9.7)
p07	*next*	21.8 (5.6)	19.2 (4.1)	47.3 (2.2)	46.0 (4.4)	98.2 (1.8)	97.3 (1.9)	149.1 (0.9)	148.4 (1.4)	298.6 (1.3)	298.7 (1.1)
	best	17.0 (1.2)	17.2 (0.9)	39.7 (2.9)	36.4 (3.0)	88.5 (4.2)	86.1 (4.9)	145.5 (2.5)	142.9 (3.8)	294.8 (2.5)	290.3 (4.6)
p08	*next*	19.7 (4.5)	19.5 (3.9)	40.8 (3.5)	40.5 (3.6)	87.1 (2.9)	88.0 (2.3)	138.0 (3.0)	137.8 (1.9)	280.4 (1.5)	279.7 (1.4)
	best	2.0 (0.8)	1.8 (0.6)	1.8 (0.8)	1.8 (0.6)	2.3 (2.6)	2.3 (1.6)	68.5 (8.9)	68.9 (12.9)	180.6 (14.6)	155.2 (22.2)
p09	*next*	13.7 (7.0)	12.0 (7.0)	31.7 (7.9)	33.5 (5.6)	75.9 (4.6)	77.6 (4.2)	121.4 (1.8)	120.9 (1.9)	243.9 (1.5)	243.6 (2.1)
	best	1.7 (0.4)	1.7 (0.4)	1.6 (0.5)	1.6 (0.5)	1.6 (0.5)	2.1 (1.4)	57.1 (14.9)	42.6 (16.1)	154.5 (26.4)	132.9 (28.7)
p10	*next*	12.0 (9.0)	11.1 (8.7)	37.2 (6.8)	37.4 (5.8)	81.8 (2.4)	81.6 (2.4)	126.5 (1.7)	125.9 (1.9)	255.1 (1.3)	254.9 (1.4)
	best	1.6 (1.9)	1.1 (0.7)	5.0 (1.0)	4.0 (0.0)	8.2 (5.6)	12.9 (6.8)	98.3 (12.0)	88.2 (18.4)	211.2 (21.3)	201.7 (19.5)
m00	*next*	243.1 (1.4)	243.1 (1.3)	390.2 (1.2)	390.3 (1.4)	781.8 (1.2)	780.9 (1.6)				
	best	128.0 (30.2)	148.9 (35.7)	225.9 (35.5)	269.6 (46.7)	457.4 (51.6)	509.8 (122.1)				

A usual approach for a semi-automatic reconstruction of strip shredded text documents would be to first execute the TSP solver to obtain a good initial solution. Then, assuming that this solution is not already perfect, either some user moves are applied or, if there is no obvious correct subsequence of strips to be concatenated or wrongly rotated strips, VNS would be executed. Afterwards, a human inspection combined with user moves is performed. The last two steps will be repeated until either no improvement can be achieved or a solution of desired quality is obtained.

Experimental Results

For testing the VNS based approach we used the same sets of data and CPU as for the experiments presented in the previous section, i.e., ten pages named p01 to p10 and one data set consisting ten document pages denoted by m00. Detailed results obtained using VNS only, i.e., by initializing VNS to a random solution, are shown in Tab. 4.3. Again, the numbers shown represent average results over 30 runs and standard deviations are presented in parentheses. In contrast to the previous test settings, the running times were not limited, i.e., after five consecutive shaking iterations without improvement during VNS the algorithm was terminated. Anyhow, we performed tests using *next improvement* and *best improvement* strategy as step function. Accordingly, the results are shown in the corresponding rows of the result table. Analogously to the results for the

Lin-Kernighan based approach, each column represents results according to the number of strips and error estimation function used. Due to the size of neighborhood structure $\mathcal{N}_5$, we decided to omit its exploration whenever more than 100 strips are to be realigned.

Although the results are worse than those obtained incorporating the transformation of RSSTD to TSP at a first glance, some relevant information can be gathered from these tests. At first, it is obvious that best improvement suits better for these runs than next improvement. At the same time, it can be observed that with increasing number of shreds the performance with respect to the obtained quality rapidly decreases. Nevertheless, having a closer look at the running times (not shown in Tab. 4.3) it can be seen that especially for small instances with 30 or 50 strips the total computation time for the VNS approach is in most cases below one second. Unfortunately, it can clearly be seen that for instances with 150 strips (or more) the absence of $\mathcal{N}_5$, i.e., block swapping, has severe consequences on the solution quality.

With respect to data set m00 it can be seen that the results obtained are rather bad. Please consider, that again the block swapping neighborhood is missing due to the large number of shreds. For the instances with 150 and 300 strips it was not possible to obtain meaningful results—neither with respect to the used computation time nor with respect to the achieved qualities.

Within a third test setting we initialized our VNS based approach using the Lin-Kernighan based method. The results obtained for this setting are shown in Tab. 4.4. As can be seen, the numbers are very similar to those presented in Tab. 4.1. Nevertheless, for some instances improvements could be achieved, e.g., p05 with 300 strips or p08 with 30 to 150 strips. Unfortunately, there are also a few instances for which worsenings occurred. Having a closer look at these results, it can be observed that although the quality decreased the corresponding objective value with respect to the error estimation function improved. Even for those instances with no improvement with respect to the quality measure, improvements with respect to the objective values could be achieved. Therefore, it can be concluded that the VNS approach could contribute to the final solution. With respect to the multiple pages instances, it can be seen that using the combination of the TSP transformation based approach and the VNS method even the very large instances with up to 9000 strips could be solved. It also should be mentioned that the standard deviations are very low for all instances.

Finally, we performed a few tests with our semi-automatic system as it would be used in practice for reconstructing strip shredded text documents. For this purpose we initialized the VNS using the Lin-Kernighan based approach and performed some user moves as soon as VNS reached a local optimum. Within only a few user interactions we were able to quickly restore all original documents by exploiting the benefits of the hybridization of machine and human power.

Table 4.4.: Average qualities of final solutions from the TSP solver comparing cost functions c_1 and c_2. Standard deviations are given in parentheses.

page	*step*	30 strips		50 strips		100 strips		150 strips		300 strips	
		c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2
p01	*next*	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	23.0 (0.0)	1.0 (0.0)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	23.0 (0.0)	1.0 (0.0)
p02	*next*	2.3 (1.5)	1.0 (0.0)	7.7 (0.8)	9.3 (0.5)	16.3 (0.7)	14.5 (1.4)	39.5 (0.7)	34.0 (1.0)	105.9 (0.6)	80.2 (1.3)
	best	2.5 (1.5)	1.0 (0.0)	7.8 (0.9)	9.5 (0.6)	16.3 (0.7)	14.7 (1.5)	39.6 (1.0)	34.1 (1.2)	106.1 (0.5)	80.5 (1.4)
p03	*next*	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	16.0 (0.0)	1.0 (0.0)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	16.0 (0.0)	1.0 (0.0)
p04	*next*	1.0 (0.0)	1.0 (0.0)	5.0 (0.0)	1.0 (0.0)	19.9 (0.5)	4.0 (0.0)	29.6 (0.7)	20.0 (0.0)	66.3 (0.6)	42.0 (0.0)
	best	1.0 (0.0)	1.0 (0.0)	5.0 (0.0)	1.0 (0.0)	19.6 (0.9)	4.0 (0.0)	29.6 (0.5)	19.9 (0.3)	66.3 (0.6)	42.0 (0.3)
p05	*next*	1.0 (0.0)	1.0 (0.0)	4.4 (0.5)	1.0 (0.0)	12.0 (0.2)	6.0 (0.0)	43.6 (0.7)	8.0 (0.0)	84.4 (1.3)	27.4 (0.8)
	best	1.0 (0.0)	1.0 (0.0)	4.3 (0.4)	1.0 (0.0)	12.0 (0.0)	6.0 (0.0)	43.6 (0.6)	8.0 (0.0)	84.8 (1.5)	27.5 (1.0)
p06	*next*	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	2.5 (1.5)	1.0 (0.0)
	best	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	2.5 (1.5)	1.0 (0.0)
p07	*next*	16.6 (0.8)	16.9 (1.1)	34.3 (1.0)	33.0 (0.9)	76.7 (1.0)	66.7 (1.8)	128.0 (1.1)	120.6 (0.9)	258.4 (1.3)	245.6 (0.7)
	best	17.0 (1.0)	17.3 (1.0)	34.4 (0.8)	32.9 (0.9)	76.0 (1.1)	67.0 (2.0)	127.6 (1.0)	120.6 (1.0)	258.6 (1.6)	245.4 (0.8)
p08	*next*	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	1.9 (0.3)	2.0 (0.0)	2.0 (0.2)	1.4 (0.5)	1.4 (0.5)	3.0 (0.0)	3.0 (0.0)
	best	2.0 (0.2)	1.9 (0.3)	1.9 (0.3)	2.0 (0.0)	1.9 (0.3)	2.0 (0.2)	1.3 (0.5)	1.3 (0.4)	3.0 (0.0)	3.0 (0.0)
p09	*next*	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)
	best	1.9 (0.3)	1.9 (0.3)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)	1.9 (0.3)	2.0 (0.0)	2.0 (0.2)	2.0 (0.0)	2.0 (0.0)
p10	*next*	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	4.0 (0.0)	7.0 (1.0)	8.3 (2.7)	12.0 (0.0)	8.0 (0.0)	33.9 (0.3)	39.0 (0.0)
	best	1.0 (0.0)	1.0 (0.0)	4.0 (0.0)	4.0 (0.0)	7.0 (1.0)	9.9 (0.8)	12.0 (0.0)	8.0 (0.0)	33.8 (0.4)	39.0 (0.0)
m00	*next*	10.7 (0.8)	13.5 (0.9)	8.3 (0.9)	8.8 (0.4)	8.4 (0.8)	9.9 (0.3)	19.9 (1.1)	15.4 (0.6)	33.8 (0.9)	29.9 (1.2)
	best	10.9 (0.5)	13.3 (0.9)	8.1 (0.8)	8.9 (0.3)	7.7 (1.1)	10.0 (0.2)	19.9 (1.1)	15.6 (0.7)	33.8 (0.9)	29.9 (1.3)

Although the results for the pure user independent VNS approach are rather discouraging, it should be emphasized that the VNS was especially designed for integrating user actions. Therefore, the approach was implemented in such a way that “forbidding” or “enforcing” certain neighborhood relations of strips is possible and can be done efficiently. Extending the Lin-Kernighan based approach to this concept would include the adaption of the values provided by the error estimation function such that “user-forbidden” strip alignments cannot occur. Due to the fact that additional edges with costs dependent on all other edges were introduced during the transformation of RSSTD to TSP this adaption also effects other edges in the derived instance which results in long runtimes on the one hand and impractical large edge weights on the other hand.

4.2.8. Computing Bounds for RSSTD via Lagrangian Relaxation

All approaches presented so far suffer from one main drawback: at no time, a lower bound on the solution is given, i.e., non of these methods indicates how to evaluate the improvement potential for a given solution provided by the automatic system. To overcome this drawback, we propose a new method for computing lower bounds via

Lagrangian relaxation (LR). For this approach it is necessary to first introduce an *integer linear programming* (ILP) formulation which is then used as a base for the LR approach. In addition, we present another method for computing lower bounds related to ILP formulation(s) for the TSP.

Core ILP Formulation

Let us assume that variable $s^{\omega}_{jj'} \in \{0,1\}$, with $1 \leq j, j' \leq n$ and $\omega \in \mathcal{O}^2$, is equal to 1 iff strip j' is the right neighbor of strip j and both are oriented according to ω. For the artificial strip n we define $s_{nj'} = 1$, iff strip j' is placed at position 1, i.e., the artificial strip is considered to be followed by the first strip. Using this variable definition the following model can be expressed, which provides a basis for the later proposed ILP formulations (for short we write d instead of $1 \in \mathcal{O}$ and u instead of $0 \in \mathcal{O}$):

$$\min \sum_{j\in\mathcal{S}} \sum_{j'\in\mathcal{S}} \sum_{\omega\in\mathcal{O}^2} s^{\omega}_{jj'} \cdot c(j,j',\omega) \tag{4.19.1}$$

$$\sum_{j'\in\mathcal{S}} \sum_{\omega\in\mathcal{O}^2} s^{\omega}_{jj'} = 1, \qquad \forall j \in \mathcal{S} \tag{4.19.2}$$

$$\sum_{j\in\mathcal{S}} \sum_{\omega\in\mathcal{O}^2} s^{\omega}_{jj'} = 1, \qquad \forall j' \in \mathcal{S} \tag{4.19.3}$$

$$\sum_{j'\in\mathcal{S}} s^{(\mathrm{d},\mathrm{u})}_{jj'} + s^{(\mathrm{d},\mathrm{d})}_{jj'} = \sum_{j'\in\mathcal{S}} s^{(\mathrm{u},\mathrm{d})}_{j'j} + s^{(\mathrm{d},\mathrm{d})}_{j'j}, \qquad \forall j \in \mathcal{S} \tag{4.19.4}$$

$$\sum_{j'\in\mathcal{S}} s^{(\mathrm{u},\mathrm{d})}_{jj'} + s^{(\mathrm{u},\mathrm{u})}_{jj'} = \sum_{j'\in\mathcal{S}} s^{(\mathrm{d},\mathrm{u})}_{j'j} + s^{(\mathrm{u},\mathrm{u})}_{j'j}, \qquad \forall j \in \mathcal{S} \tag{4.19.5}$$

$$\sum_{\omega\in\mathcal{O}} s^{\omega}_{jj'} + \sum_{\omega\in\mathcal{O}} s^{\omega}_{j'j} \leq 1, \qquad \forall j, j' \in \mathcal{S} \tag{4.19.6}$$

$$s^{\omega}_{jj} = 0, \qquad \forall j \in \mathcal{S},\ \omega \in \mathcal{O}^2 \tag{4.19.7}$$

$$s^{\omega}_{jj'} \in \{0,1\}, \qquad \forall \omega \in \mathcal{O}^2,\ j, j' \in \mathcal{S} \tag{4.19.8}$$

While the total costs for an assignment of strips to each other should be minimized according to expression (4.19.1), constraints (4.19.2) and (4.19.3) state that each strip j, with $1 \leq j \leq n$, has to be followed and preceded by exactly one strip, i.e., exactly one strip has to be assigned to the position right to strip j and one left to j. If a strip j precedes strip j' it is obvious that strip j follows another strip. Anyhow, the orientation of strip j has to be the same for both relations, see Eq. (4.19.4) and (4.19.5). As soon as one strip j is preceding another strip j' strip j cannot be placed right next to j', cf. Eq. (4.19.6).

Cycle Elimination Cuts

Due to the strong relationship of RSSTD to (A)TSP it is obvious that optimal solutions with respect to formulation (4.19) can in general contain subtours, which are not valid for RSSTD. Therefore, we decided to implement and compare two different approaches for preventing subtours. The first one is based on cycle elimination constraints, which can be expressed as follows:

$$\sum_{k\in\mathcal{C}}\sum_{\omega\in\mathcal{O}^2} s^{\omega}_{kk+1} \le |\mathcal{C}|-1, \qquad \forall\emptyset\neq\mathcal{C}\subset\mathcal{S}, \text{for } |\mathcal{C}| \text{ is a cycle,} \tag{4.20}$$

whereas $\mathcal{C}$ corresponds to cycles of length less than $|\mathcal{S}|$ and $k+1$ denotes the strip placed right to strip k on this cycles.

Since the number of constraints specified by expression (4.20) is exponential in the number of strips, an efficient dynamic separation of these constraints as cutting planes is necessary for computing practical results. This is done by first building a complete graph $G(V,E)$ whose nodes $v\in V$ correspond to strips. The weights of the edges $(i,j)\in E$ are set to $1-\sum_{\omega\in\mathcal{O}^2} s^{\omega,\text{LP}}_{ij}$, where $s^{\omega,\text{LP}}_{ij}$ are the current values of the LP solution. Any cycle $\emptyset\neq\mathcal{C}\subset E$ in this graph, whose length is less than 1 corresponds to a violated cut. Therefore, these cuts can be separated by computing shortest paths from i to j after removing the corresponding edge $(i,j)\in E$ from the graph, with $i,j\in V$.

Although, from a theoretical point of view, cycle elimination cuts are in general weaker than subtour elimination cuts (4.21)

$$\sum_{k\in\mathcal{C}}\sum_{k'\in\mathcal{C}}\sum_{\omega\in\mathcal{O}^2} s^{\omega}_{kk'} \le |\mathcal{C}|-1 \qquad \forall\emptyset\neq\mathcal{C}\subset\mathcal{S}, \tag{4.21}$$

i.e., every cycle elimination cut is satisfied in a model including subtour elimination cuts, the separation of the latter is more complex and in most cases more time demanding [99]. We therefore decided to use the more efficient cut separation method to be able to obtain as fast as possible lower bounds on RSSTD, although slightly better bounds might be achieved by considering subtour elimination cuts.

Compact ILP Formulation

The second approach for eliminating cycles is based on the introduction of additional variables $p_{ij}\in\{0,1\}$, with $1\le i,j\le n$, whereas p_{ij} is equal to 1 iff strip j is assigned to position i and otherwise 0. Then the following constraints can be defined:

$$\sum_{i=1}^{n} p_{ij} = 1, \qquad \forall j\in\mathcal{S} \tag{4.22.1}$$

$$\sum_{j \in \mathcal{S}} p_{ij} = 1, \qquad \forall i = 1, \ldots, n \tag{4.22.2}$$

$$p_{1j'} = \sum_{\omega \in \mathcal{O}^2} s^{\omega}_{nj'}, \qquad \forall j' \in \mathcal{S} \tag{4.22.3}$$

$$p_{n-1j} = \sum_{\omega \in \mathcal{O}^2} s^{\omega}_{jn}, \qquad \forall j \in \mathcal{S} \tag{4.22.4}$$

$$p_{ij} + p_{i+1j'} - 1 \leq \sum_{\omega \in \mathcal{O}^2} s^{\omega}_{jj'}, \qquad \forall \begin{cases} i = 1, \ldots, n-2, \\ j, j' \in \mathcal{S} \end{cases} \tag{4.22.5}$$

$$p_{nn} = 1 \tag{4.22.6}$$

$$p_{ij} \in \{0,1\}, \qquad \forall i = 1, \ldots, n, \ j \in \mathcal{S} \tag{4.22.7}$$

Due to constraints (4.22.1) and (4.22.2) it is assured that each strip is assigned to exactly one position and vice versa. Anyhow, a connection between variables p and s has to be established. This is done by Eq. (4.22.3), (4.22.4) and (4.22.5). If strip j is assigned to position i and strip j' to position $i+1$ then the according variables $s^{\omega}_{jj'}$, with $\omega \in \mathcal{O}^2$, have to be set to one. Finally, constraint (4.22.6) ensures that the artificial strip is assigned to position n.

In contrast to a formulation based on cycle elimination cuts the number of constraints (and variables) is polynomially bounded for a model based on Eq. (4.22.1) to (4.22.7). Obviously this comes with the advantage that all constraints can be included in the model from the beginning and therefore a time demanding separation procedure is not needed.

In the further context, we will denote the two above presented formulations by *cycle elimination based formulation* (CEF) and *position assignment based formulation* (PAF), whereas CEF corresponds to the core formulation (4.19) amended by constraints (4.20) and PAF refers to formulation (4.19) together with expressions (4.22). For practical results regarding the direct solution of these two formulations by using CPLEX we refer to the results presented at the end of this section.

An obviously interesting question now is, whether the bounds obtained from the LP relaxations of CEF or PAF are better, i.e., which of the two formulations are tighter [139]. It can be shown by an example that there exists at a fractional solution which is valid with respect to the relaxed version of PAF but contains subcycles, i.e., for which at least one constraint contained in expression (4.20) is violated. For a pictorial presentation of this example see Fig. 4.12, whereas circles represent strips and squares correspond to positions. Accordingly, the arrows from circles to square represent variables p_{ij} and variables $s_{jj'}$ are represented by arrows between circles. Only variables with values greater than zero are shown. The concrete values of the variables are as follows: Let

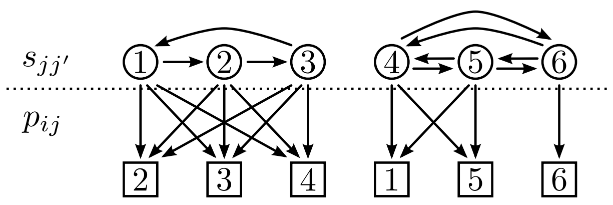


Figure 4.12.: A schematic presentation of a solution valid with respect to PAF. Strips are presented by circles and positions by rectangles.

us assume that $n = 6$, then the following assignment of values to the variables forms a valid PAF solution: $p_{21} = p_{22} = p_{23} = 1/3$, $p_{31} = p_{32} = p_{33} = 1/3$, $p_{41} = p_{42} = p_{43} = 1/3$, $p_{14} = p_{15} = 1/2$, $p_{54} = p_{55} = 1/2$ and $p_{66} = 1$. All other p-variables are set to 0. For the sequence variables we set the values $s^{\omega}_{12} = s^{\omega}_{23} = s^{\omega}_{31} = 1$ and $s^{\omega}_{45} = s^{\omega}_{54} = s^{\omega}_{56} = s^{\omega}_{65} = s^{\omega}_{46} = s^{\omega}_{64} = 1/2$, with $\omega = (0, 0)$. Since $p_{ij} \leq 1/2$, for $i = 1, \ldots, 5$ and $j \in \mathcal{S}$, $p_{ij} + p_{i+1j'} - 1 \leq 0$, for $i = 1, \ldots, 4$ and $j, j' \in \mathcal{S}$, holds. Since $0 \leq s^{\omega}_{jj'}$, for $j, j' \in \mathcal{S}$, $\omega \in \mathcal{O}^2$, constraints (4.22.5) are fulfilled. It can be easily checked that all other constraints (4.22.1), (4.22.2), (4.22.3), (4.22.4) and (4.22.6) are fulfilled, too. However, constraint (4.20) is violated by setting $C = \{1, 2, 3\}$. Although by this example it is shown that PAF is not stronger than CEF the reverse, i.e., whether CEF is stronger than PAF, remains an open question.

Lagrangian Relaxation for RSSTD

Preliminary tests revealed that the application of exact approaches to RSSTD, e.g., a direct solution of CEF and PAF using general purpose ILP solvers, is limited to relatively small instances. Therefore, heuristic methods are of great importance when trying to solve real-world instances. Anyhow, one main drawback of many heuristics is the lack of providing (tight) bounds on the solution quality. To overcome this problem one could solve the *linear programming* (LP) relaxation of CEF or PAF; see the end of this section for computational results. In addition, we developed a *Lagrangian relaxation* (LR) approach based on PAF. The main idea of LR is to substitute complicating constraints by corresponding penalty terms in the objective function. For this purpose, each relaxed constraint is associated with a so called *Lagrangian multiplier*. Subsequently, one tries to find a set of Lagrangian multipliers that maximizes the associated lower bound for the original minimization problem.

For this purpose we relax the linking constraints (4.22.3)–(4.22.5) of PAF resulting in the following new objective function:

$$\begin{aligned}\min \sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}\sum_{\omega\in\mathcal{O}^2} & s^{\omega}_{jj'}\cdot c(j,j',\omega)+\\ & \sum_{j'\in\mathcal{S}}\lambda^1_{j'}\cdot\left(p_{1j'}-\sum_{\omega\in\mathcal{O}^2}s^{\omega}_{nj'}\right)+\\ & \sum_{j\in\mathcal{S}}\lambda^2_{j}\cdot\left(p_{n-1j}-\sum_{\omega\in\mathcal{O}^2}s^{\omega}_{jn}\right)+\\ & \sum_{i=1}^{n-2}\sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}\lambda^3_{i,j,j'}\cdot\left(p_{ij}+p_{i+1j'}-1-\sum_{\omega\in\mathcal{O}^2}s^{\omega}_{jj'}\right)\end{aligned} \tag{4.23}$$

After applying some basic transformations and substituting constant expressions by (newly introduced) coefficients ρ_{ij}, $\sigma^{\omega}_{jj'}$ and δ, with $1\le i,j,j'\le n$ and $\omega\in\mathcal{O}^2$, the LR approach can be reformulated as follows:

$$\min \underbrace{\sum_{j\in\mathcal{S}}\sum_{i=1}^{n-1}(\rho_{ij}\cdot p_{ij})}_{\text{SP I}}+\underbrace{\sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}\sum_{\omega\in\mathcal{O}^2}\left(\sigma^{\omega}_{jj'}\cdot s^{\omega}_{jj'}\right)}_{\text{SP II}}+\delta \tag{4.24}$$

subject to Eq. (4.19.2)–(4.19.8), (4.22.1), (4.22.2), (4.22.6), and (4.22.7)

with

$$\rho_{1j}=\lambda^1_j+\sum_{j'=1}^{n}\lambda^3_{i,j,j'}+\sum_{j'=1}^{n}\lambda^3_{i,j',j}, \qquad \forall j\in\mathcal{S} \tag{4.25}$$

$$\rho_{ij}=\lambda^2_j+\sum_{j'=1}^{n}\lambda^3_{i,j,j'}+\sum_{j'=1}^{n}\lambda^3_{i,j',j}, \qquad \forall\begin{cases} i=2,\dots,n-2\\ j\in\mathcal{S}\end{cases} \tag{4.26}$$

$$\rho_{n-1j}=\lambda^2_j+\sum_{j'=1}^{n}\lambda^3_{n-1,j',j}, \qquad \forall j\in\mathcal{S} \tag{4.27}$$

$$\sigma^{\omega}_{j,j'}=c(j,j',\omega)-\sum_{i=1}^{n-2}\lambda^3_{i,j,n}, \qquad \forall\begin{cases} \omega\in\mathcal{O}^2\\ j,j'\in\mathcal{S}\setminus\{n\}\end{cases} \tag{4.28}$$

$$\sigma^{\omega}_{n,j}=c(n,j,\omega)-\lambda^1_j-\sum_{i=1}^{n-2}\lambda^3_{i,n,j}, \qquad \forall\begin{cases} \omega\in\mathcal{O}^2\\ j\in\mathcal{S}\setminus\{n\}\end{cases} \tag{4.29}$$

$$\sigma^{\omega}_{j,n} = c(j,n,\omega) - \lambda^2_j - \sum_{i=1}^{n-2} \lambda^3_{i,j,n}, \qquad \forall \begin{cases} \omega \in \mathcal{O}^2 \\ j \in \mathcal{S} \setminus \{n\} \end{cases} \tag{4.30}$$

$$\sigma^{\omega}_{n,n} = c(n,n,\omega) - \lambda^1_n - \lambda^2_n - \sum_{i=1}^{n-2} \lambda^3_{i,n,n}, \qquad \forall \omega \in \mathcal{O}^2 \tag{4.31}$$

$$\delta = -\sum_{j\in\mathcal{S}} \sum_{j'\in\mathcal{S}} \sum_{i=1}^{n-2} \lambda^3_{i,j,j'} \tag{4.32}$$

Based on the fact, that the coefficients ρ, σ and δ are composed of linear combinations of λ^1, λ^2, λ^3 and the cost function c, see Eq. (4.25)–(4.32), it can be observed that the above formulation decomposes into two independent subproblems only linked by the objective function (4.24). The first subproblem SP I formulated via variables p_{ij}, with $1 \leq i \leq n-1$ and $j \in \mathcal{S}$, corresponds to a linear assignment problem. It is well known that this problem can be efficiently solved. The second subproblem SP II formulated via variables $s^{\omega}_{jj'}$, with $j, j' \in \mathcal{S}$ and $\omega \in \mathcal{O}^2$, corresponds to the generalized version of the so-called *cycle cover problem* which is polynomially solvable in the non-generalized variant [67]. In our case it was, however, shown that this problem is $\mathcal{NP}$-hard [55]. Consequently, it can be easily shown that the integrality property does not hold for SP II, which implies that bounds provided by our LR approach might be better than those provided by an LP relaxation of PAF [11].

For computing lower bounds by means of LR, we implemented a standard subgradient method as described in [11] by initializing all Lagrangian multipliers to 0 and setting the strategic parameter $\pi = 2$. The value of π is halved as soon as 30 subgradient iterations without improvement on the lower bound were performed. In contrast, π is doubled when an improvement could be achieved and $\pi \leq 1$ holds. This iterative process is terminated once π falls below 0.001 or the lower bound provided by this method corresponds to the best known upper bound, which is iteratively updated based on the solutions generated by the Lagrangian heuristic presented within the next section. For solving subproblems SP I and SP II we directly applied the general purpose ILP solver CPLEX 11.2. Again, we refer to results presented at the end of this section for a detailed listing including a comparison of bounds obtained via LP relaxations and those obtained via LR.

A Lagrangian Heuristic

Based on the LR presented in the previous section, we further developed a *Lagrangian heuristic* (LH) which provides feasible solutions to the original problem based on the values of the relaxed ILP. The main idea is to decode the neighborhood relations and orientations of strips such that a feasible solution is generated. Since the absolute positions of strips, i.e., the values of variables p_{ij}, are not necessarily consistent with the relative positions, i.e., the values of variables $s^{\omega}_{jj'}$, we decided to neglect the information about the absolute position within this decoding step and derive a feasible solution from the relative positions only, which also primarily contribute to the objective function. Since the virtual strip n is placed at the last position (see Eq. (4.22.6)), we start the decoding by placing this strip at position n. According to the values of s^{ω}_{jn}, with $1 \leq j \leq n-1$ and $\omega \in \mathcal{O}^2$, we place that strip $\overline{j}$ at position $n-1$ which has a corresponding variable $s^{\omega}_{\overline{j}n}$ equal to 1. Of course, the orientation of the strip is also regarded. This method is applied iteratively as long as not already positioned strips are concerned. In the case of a cycle, we restart the method by placing a randomly chosen and so far not positioned strip at the last yet free position.

Since any permutation of strips with the artificial strip placed at the last position forms a valid solution, this method always provides feasible solutions. Further, by using appropriate datastructures the runtime of this approach is in $O(n^2)$ as for each position at most $4n$ variables have to be evaluated.

Experimental Results

To evaluate the performances and the contributions of the above presented approaches, we applied them to instances of RSSTD. For generating instances, we used those documents introduced by Ukovich *et al.* in [134], which were then converted into B/W images and were (virtually) cut into 80 to 135 strips, each. These settings correspond to strip widths of 2.6mm to 1.5mm. The test results presented within this section were obtained on a single core of an Intel® Core™2 Quad CPU with 8 GB RAM and 2.83 GHz and ILOG CPLEX 11.2 has been used as general purpose (I)LP solver.

For computing lower bounds by means of LR we implemented the standard subgradient method, whereas the upper bound is updated based on the solutions provided by the proposed LH. The Lagrangian multipliers were all initialized to 0. Obviously, the execution of the subgradient method is aborted as soon as the lower and upper bound are identical. We analyzed the bounds provided by LR and the LP relaxation of CEF and PAF on 560 instances in total and the main result is that in most cases, i.e., in 517 out of 560, the obtained bounds are equal. Only for 43 instances of which all where generated based on the first document page of the test set introduced by Ukovich *et al.* a difference in the quality of the bounds could be identified. The corresponding results are shown in Tab. 4.5, whereas the first column indicates the number of strips the page was cut into and the second column lists the absolute objective values of the original document pages. The columns labeled with UB represent the lower bound obtained via LR, CEF and PAF, respectively. These numbers represent the relative values in relation to the objective value of the original document. The column labeled with LB represents the upper bound provided by LH during LR. The number of iterations performed during LR

Table 4.5.: Results comparing the bounds obtained by the proposed LR and the LP relaxation of CEF in relation to the original document page (orig.). In addition the number of LR iterations until LR was terminated are provided.

		LR				CEF		PAF	
strips	orig.	LB	UB	iter.	time [s]	LB	time [s]	LB	time [s]
80	29408	99.8232%	99.8232%	1.0	2.2 (0.3)	99.5103%	0.1	99.5103%	86.8
81	29408	99.8232%	99.8232%	1.0	2.3 (0.4)	99.5103%	0.1	99.5103%	129.2
86	31494	99.6444%	99.6444%	1.0	1.3 (0.2)	99.4253%	0.1	99.4253%	24.7
87	31494	99.6444%	99.6444%	1.0	1.3 (0.3)	99.4253%	0.1	99.4253%	20.6
88	31494	99.6444%	99.6444%	1.0	1.3 (0.2)	99.4253%	0.1	99.4253%	14.0
89	32774	99.8047%	99.8047%	1.0	2.4 (0.5)	99.6217%	0.1	99.6217%	14.3
90	32774	99.8047%	99.8047%	1.0	2.5 (0.4)	99.6217%	0.1	99.6217%	28.2
91	32440	100.0000%	100.0000%	1.0	2.8 (0.4)	99.7534%	0.1	99.7534%	148.6
92	32440	100.0000%	100.0000%	1.0	2.8 (0.5)	99.7534%	0.1	99.7534%	235.3
93	32440	100.0000%	100.0000%	1.0	2.5 (0.4)	99.7534%	0.1	99.7534%	115.7
96	36256	100.0000%	100.0000%	1.0	4.7 (0.6)	99.7269%	0.1	99.7269%	192.1
97	36256	100.0000%	100.0000%	1.0	4.8 (0.8)	99.7269%	0.1	99.7269%	430.3
98	36256	100.0000%	100.0000%	1.0	4.1 (0.7)	99.7269%	0.1	99.7269%	302.6
106	37122	99.9407%	99.9407%	1.0	5.1 (0.9)	99.6875%	0.1	99.6875%	345.9
107	37122	99.9407%	99.9407%	1.0	4.7 (0.8)	99.6875%	0.2	99.6875%	504.8
108	37122	99.9407%	99.9407%	1.0	4.7 (0.9)	99.6875%	0.1	99.6875%	237.0
109	38694	99.8346%	103.9565%	331.0	1612.6 (83.8)	99.6614%	0.2	99.6614%	377.2
110	38694	99.8346%	104.2399%	331.0	1637.4 (105.0)	99.6614%	0.2	99.6614%	254.1
111	38694	99.8346%	104.5810%	331.0	1601.1 (93.7)	99.6614%	0.1	99.6614%	302.4
112	38694	99.8346%	103.0356%	331.0	1606.8 (86.6)	99.6614%	0.1	99.6614%	367.5
113	39836	99.9699%	99.9699%	1.0	4.2 (0.7)	99.6034%	0.1	99.6034%	473.5
114	39836	96.6375%	96.6375%	1.0	4.1 (0.8)	99.6034%	0.1	99.6034%	380.0
115	39836	99.9699%	99.9699%	1.0	4.0 (0.8)	99.6034%	0.1	99.6034%	458.9
116	39836	93.3052%	93.3052%	1.0	4.0 (0.8)	99.6034%	0.2	99.6034%	425.7
117	39926	99.8397%	104.2131%	331.0	2195.9 (167.0)	99.6569%	0.1	99.6569%	449.3
118	39926	99.8397%	104.3895%	331.0	2157.3 (134.1)	99.6569%	0.2	99.6569%	426.0
119	39926	99.8397%	103.6462%	331.0	2137.8 (151.1)	99.6569%	0.1	99.6569%	508.1
120	39962	99.8398%	103.9471%	331.0	2387.2 (282.4)	99.6572%	0.3	99.6572%	313.9
121	42422	99.7737%	99.7737%	1.0	7.4 (0.7)	99.5780%	0.2	99.5780%	554.6
122	42422	99.7737%	99.7737%	1.0	6.2 (1.0)	99.5780%	0.2	99.5780%	516.6
123	42422	99.7737%	99.7737%	1.0	7.1 (1.0)	99.5780%	0.2	99.5780%	629.1
124	42422	99.7737%	99.7737%	1.0	6.9 (1.2)	99.5780%	0.3	99.5780%	610.2
125	42454	96.4481%	96.4481%	1.0	8.4 (1.6)	99.5784%	0.3	99.5784%	629.1
126	44682	93.2247%	93.2247%	1.0	7.7 (1.7)	99.6598%	0.2	99.6598%	626.5
127	44682	96.5542%	96.5542%	1.0	7.8 (1.5)	99.6598%	0.2	99.6598%	597.9
128	44682	96.5542%	96.5542%	1.0	7.7 (1.6)	99.6598%	0.3	99.6598%	677.3
129	44728	96.5543%	96.5543%	2.0	12.1 (2.4)	99.6602%	0.3	99.6602%	829.8
130	44728	99.8837%	99.8837%	2.0	12.6 (2.1)	99.6602%	0.3	99.6602%	851.7
131	45698	99.9912%	99.9912%	1.0	9.3 (1.9)	99.7505%	0.2	99.7505%	705.8
132	45698	96.6582%	96.6582%	1.0	10.1 (2.0)	99.7505%	0.3	99.7505%	678.9
133	45698	99.9912%	99.9912%	1.0	9.8 (1.6)	99.7505%	0.3	99.7505%	656.5
134	45698	99.9912%	99.9912%	1.0	10.4 (1.2)	99.7505%	0.2	99.7505%	694.0
135	45698	96.6582%	96.6582%	1.0	10.2 (1.7)	99.7505%	0.3	99.7505%	809.1

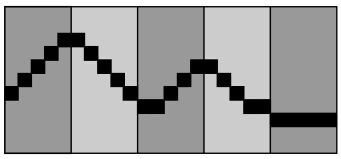


Figure 4.13.: If this set of strips has to be reconstructed, not all Lagrangian multipliers are set to zero in the set of optimal multipliers when using the LR approach.

is shown in column iter and obviously, the column labeled with time represents the time used for computing the lower bounds. The values for the LR approach are averages over 30 iterations. The standard deviations for the times are shown in the parentheses. For the lower and upper bounds as well as the number of iterations the standard deviation is equal to 0 and therefore omitted. In case the number of iterations is equal to 1 the solution derived by our LH approach by setting all Lagrangian multipliers to 0, i.e., solving the core formulation (4.19) solely, is proven optimal.

The following two observations can be made based on the test results: first of all the bounds obtained by our LR approach are typically equal or better than the bounds provided by an LP formulation using cycle elimination constraints. We assume, however, that this behavior is mainly based on the objective function used for estimating the likelihood of placing two strips next to each other. Furthermore we expect to emphasize this positive property of our cost function when considering more problem specific information by calculating the concrete cost values, e.g., by considering the character orientations, applying optical character recognition (OCR), or incorporating the likelihood that two patterns identified on the corresponding strip edges match with each other. In that case we assume that the error made by the cost function is even further minimized.

The second conclusion which can be drawn from the results is that the number of iterations until our LR approach terminates is typically low (even for those instances not listed in this table). In most cases there is even only one iteration. For some instances, however, it was not possible to improve the bound obtained during the first iteration of LR, but at the same time LH was not able to provide a primal feasible solution with identical objective value. Again, we expect to improve on this issue by adapting the cost function as already indicated above.

When comparing the CPU times, it can be seen that the CEF approach was clearly the fastest one. Especially the results for PAF are extremely bad (with respect to the CPU times). The LR performed worst for those instances where the optimal bound could not be found within few iterations.

Based on this observation the initialization of the Lagrangian multipliers to zero seems not only to be valuable but to be the only reasonable approach for providing good bounds

Table 4.6.: Comparison of computation times and solution qualities of PAF and CEF when directly solved using CPLEX 11.2. Numbers without parentheses indicate CPU times in seconds until the optimal solution was obtained (including optimality proof) whereas numbers in parentheses indicate the relative gap of current best integer and best dual bounds after 1200 seconds of computation time.

strips	20	30	40	50	60	70	80	90	100
inst.	PAF/CEF	PAF/CEF	PAF/ CEF	PAF/ CEF	PAF/ CEF	PAF/ CEF	PAF/ CEF	PAF/CEF	PAF/ CEF
m01	0.3/ 0.3	2.0/ 0.6	(0.04)/(0.07)	(0.50)/(0.02)	(0.01)/(0.10)	(0.80)/(0.01)	(0.80)/(0.01)	1200.2/ 42.5	(0.81)/ 68.2
m02	0.5/ 0.1	7.9/ 0.3	311.1/ 0.7	(0.52)/ 1.2	(0.83)/ 7.8	(0.84)/ 4.0	(0.84)/ 6.8	(0.83)/ 19.8	(0.83)/ 149.1
m03	0.2/ 0.1	72.2/ 0.6	1.3/ 0.3	30.7/ 0.6	(0.44)/ 1.1	(0.77)/ 2.4	102.6/ 2.7	(0.80)/ 4.1	(0.80)/ 4.2
m04	0.1/ 0.1	7.3/ 0.2	18.7/ 0.2	117.5/ 0.6	(0.77)/ 1.3	(0.12)/ 0.9	(0.16)/ 2.8	(0.11)/ 28.1	(0.80)/ 5.6
m05	0.1/ 0.1	0.5/ 0.1	1.2/ 0.5	101.8/ 0.2	(0.18)/ 2.4	249.9/ 1.2	290.4/ 0.7	(0.56)/ 7.1	(0.71)/(0.06)
m06	1.1/ 0.0	67.0/ 0.2	1.1/ 0.4	(0.18)/ 0.7	(0.29)/ 0.7	(0.09)/ 1.9	148.9/ 5.2	813.1/ 23.2	720.5/ 4.7
m07	0.1/ 0.1	0.4/ 0.1	4.0/ 0.5	281.9/ 0.2	(0.34)/ 1.8	(0.20)/ 1.1	(0.41)/ 0.8	(0.69)/ 2.4	(0.77)/ 17.3
m08	0.2/ 0.1	0.8/ 0.1	108.4/ 0.8	7.7/ 0.2	(0.42)/ 0.8	(0.20)/ 1.0	(0.75)/ 0.8	(0.76)/ 4.4	174.1/ 2.5
m09	0.2/ 0.1	0.8/ 0.3	147.3/ 0.7	(0.24)/ 1.0	(0.21)/ 0.9	676.7/ 0.5	(0.64)/ 1.6	150.8/ 2.5	(0.76)/ 2.5
m10	0.5/ 0.1	1.8/ 0.3	306.6/ 0.3	(0.78)/ 1.1	637.1/ 2.0	(0.78)/ 2.6	(0.79)/ 4.7	(0.78)/ 14.4	(0.78)/ 6.2

as well as solving RSSTD. Nevertheless, not for all instances all Lagrangian multipliers are set to zero in the optimal set of multipliers. See for example the document shown in Fig. 4.13. When realigning these strips some multipliers have to be set to values not equal to zero for eliminating the cycles implied by the first two strips as well as the third and the fourth strip.

In addition to the experiments listed in Tab. 4.5 we tested to directly solve the above presented ILP formulations via CPLEX. The corresponding results are listed in Tab. 4.6. For this test setting we used again the document pages introduced by Ukovich *et al.* This time, however, they were cut into 20 to at most 100 strips each, since preliminary tests revealed that the direct application of the general purpose ILP solver CPLEX to the above presented ILP formulations can be very time-consuming and for more than 110 strips the computation times did in most cases exceed a given time limit of 1200 seconds.

The numbers presented in Tab. 4.6 should be interpreted as follows: We present for each document page (m01–m10) and number of strips (20–80) the time (in seconds) until the optimal solution was found (and its optimality was proven). In case the optimal solution was either not reached or was not proven to be optimal within 1200 seconds of available computation time we present the relative gap of the so far best found integer solution and the dual bound computed by CPLEX in parentheses.

As can be seen, the numbers in Tab. 4.6 show that by directly applying CPLEX to the two ILP formulations, CEF leads to far better results than PAF. More specifically, for almost all instances with 50 or more strips optimal solutions could be obtained via CEF in some seconds of computation time. For only a few instances of that sizes even CEF could not lead to proven optimal solutions. Furthermore, for those instances with less than 50 strips, CEF provided more often the optimal solution and even in case both formulations could achieve optimality the computation times for the approach based on CEF where in most cases shorter.

Although Tab. 4.6 implies that solving a model based on CEF via CPLEX is much more efficient, the bounds obtained via the LR/LH approach are a little bit more promising than the results computed by the LP relaxation of CEF. Since the runtimes until the bounds were achieved did relatively strongly vary for both approaches no clear statement can be given which of the two different approaches for computing dual bounds is in the given case faster. Nevertheless, both the LR/LH approach and the computations of LP relaxations provide a good toolkit for producing valuable (lower) bounds. Furthermore, the LH often provides the optimal solution within a few iterations of the LR/LH approach.

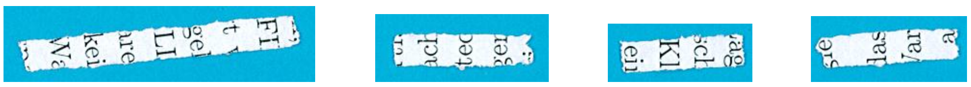
4.2.9. Discussion of Related and Arising Problems

Based on the results presented in the above sections, it can be seen that reconstructing strip shredded text documents involving few pages can be done quite effective—at least from the algorithmic point of view. However, there are some issues which should be discussed a little bit more in detail.

Issues Related to the Error Estimation Function

The most crucial part of the above presented methods is the error estimation function. Although many (preliminary) tests with different definitions of error estimation functions revealed that the two used within this thesis are the most promising ones, it is very easy to find examples of documents for which the error estimation completely fails. Nevertheless, as already discussed previously, there exists no generally "perfect" error estimation function since in some cases even humans are not able to decide which of two possible strip alignments is the correct one. In many situations the intuition of humans is, however, reliable and user can provide valuable information such that the reconstruction process can be optimally performed. Therefore, any automatic document recovery system must finally rely on the input of users indicating whether or not the reconstructed pages are sound.

Another point of critique which might be formulated in relation to the methods presented in this chapter is the fact that all test instances were automatically generated, i.e., especially the cutting process was not performed using real shredders but by doing it virtually. As shown by Ukovich *et al.* [132] the extraction of features from "real" shredded documents performs equally good as from virtually shredded documents. Based on this observation we performed some experiments using a standard shredder as found in our



(a) (b) (c) (d)

Figure 4.14.: A scan of shreds indicating that the amount of information lost along the edges is minimal.

office as well as a flatbed scanner. For an exemplary scan of two matching shreds see Fig. 4.14. In addition the reconstructing of virtually shredded documents were performed for testing the applicability of the proposed approaches and methods. Clearly, for any real-world reconstruction system it is necessary that an automatic device is developed for scanning, extracting and rotating strips which in the following can be evaluated using an error estimation function respecting noise induced during the cutting and scanning process. For example, the error estimation function could be advanced in such a way that not the pixels directly located at the strip's edges are used for extracting edge information but those pixels being located two or three layers away from the edge, cf. [10].

However, it turned out that for the reliability of the error estimation function the resolution used for scanning the images is much more impact then the "perfectness" of the cuts. When using higher resolutions, the number of pixels and therefore the information along the edge is obviously higher. Let us remark that on the one hand the appropriate resolution can be chosen by the user of the automatic reconstruction system since in most cases the data acquisition process will be part of such a system. On the other hand, tests showed that using a "standard" resolution of 150dpi is in most cases adequate.

To further enhance the results obtained by our methods we additionally tried a set of more "advanced" error estimation functions trying the incorporate the complexity of the pattern shown along the edges. For example, we defined a relative error estimation which is simply the normalization of the absolute value obtained by Eq. (4.12) to values in the interval $[0; 1]$. Additionally, we tried methods for counting (and matching) larger blocks of black pixels to blocks of black pixels on the corresponding edge of the second shred. A third approach tried to compute a indicator on how good a matching between two shreds is based on the number of (other) shreds having a similar absolute error estimation value with respect to Eq. (4.12). However, it turned out that all of the approaches increased the obtained results for some instances but simultaneously worsened the results for other instances. On average error estimation function c_2, see Eq. (4.12), yielded the best results.

Obviously, for any recovery system to be used for real-world documents during forensic investigations it is necessary that the error estimation function respects information gathered from the strips using pattern recognition and/or image processing methods like line spacings, top and bottom margins, text color, background color and many others. Obviously, the times for computing any such error estimation function will raise but at the same time the robustness of the method should increase too.

Finally, it was mentioned in the beginning of this section that we assume that the back face of the strips is blank. It is, however, relatively easy to extend the presented error estimation functions such that the information on the back face is regarded too. Nevertheless one should keep in mind that, obviously, beside the decision on the orientation of the strips it would also be necessary to decide which of the two faces is the front of the strip. Therefore, the search space is enlarged and obviously the one expects the running times to increase too. At the same time, this extension should positively effect the correctness of the error estimation function since the information included is doubled.

Multilevel Refinement Strategy

While the methods proposed so far, mainly focus on the relative alignment of strips to each other, i.e., the neighborhood relations, it would also be imaginable that absolute position information is computed for individual strips. This would be of high interest, especially for shredders with either various strip widths or if some of the knives are blunt or even missing and therefore the properties of certain strips are significantly different from other shreds, e.g., sharp versus frayed cuts or even twice as broad strips.

Another, from the algorithmic point of view, interesting extension would be the application of so called *multilevel refinement strategies* [136, 137]. The basic idea of such a heuristic is to iteratively solve a given problem instance on different levels of abstraction whereas representations on higher levels normally correspond to easier to solve instances, e.g., due to smaller instance size. Using the concepts of *coarsening* and *refinement* the different entities of the instance can be transformed into each other such that solutions on a higher level can be "extended" to solutions on a lower level and vice versa. Obviously, this process can be iterated in both directions until no further improvements can be achieved. For a survey on multilevel refinement strategies including successful applications to several combinatorial optimization problems, including graph partitioning and the traveling salesman problem we refer to [136, 137].

In our case, the coarsening step would include the building of *blocks* or *meta-strips* which consist of two or more matched strips. It is therefore easy to generate instances of smaller size, i.e., with less strips, which can then be solved using the above presented methods. During the refinement the meta-strips are then loosened such that previously fix matched strips can be separated and moved independently of each other. While the building of meta-strips can be done based on heuristics using the error estimation function, this step can also be performed based on the input of humans using an user guided search as proposed in Sec. 4.2.7.

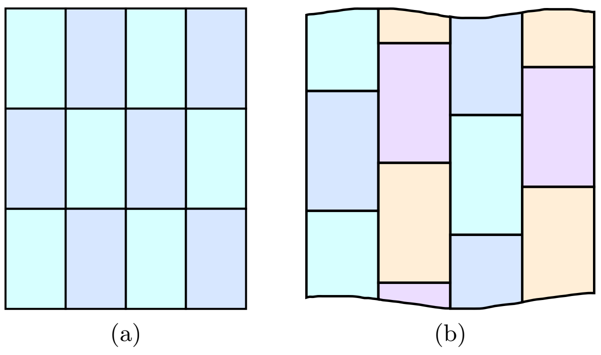


Figure 4.15.: Two different possible cutting patterns.

4.3. Cross Cut Shredded Text Documents

Beside the option to cut documents into strips there is also the possibility to use so-called cross cut shredders which cut the document along both the x- and the y-axis into small rectangular shreds whose heights and widths differ from the height and width of the original document, respectively. While the vertical cutting is done in a similar way as for strip shredders, i.e., each cut starts at the top of the page and continues until the end of the page, the horizontal cut is either also running from one document edge to the other or the cuts are much shorter, i.e., reaching only from one vertical cut to the next, and vertically shifted, see for an illustration Fig. 4.15.

In this section, we focus on cutting patterns like the one shown in Fig. 4.15a only, i.e., we are given a set of strips all having (almost) the same rectangular shape with equal widths and heights. Analogously to the reconstruction of strip shredded text documents we assume that the back face of the shreds is blank and completely empty shreds are supposed to be removed from the input. Furthermore, to simplify the problem we assume that the orientation of the shreds is known. An extension of the model based on the ideas already presented in the previous section is, however, straightforward possible.

While there is already a multitude of different approaches for the reconstruction of strip shredded or manually riped up (text) documents, there is, to our best knowledge, no work published focusing on the reconstruction of cross cut shredded text documents (RCCSTD). However, some of the methods originally designed for the reconstruction of strip shredded documents can be performed as preprocessing step, e.g., we refer to the cluster methods presented in [134].

In the next section, we give a formal model for this problem, which also describes how candidate solutions are represented in our approach. For quickly obtaining reasonable initial solutions, different construction heuristics are described afterwards. Then, a general variable neighborhood search (VNS) metaheuristic that utilizes several different neighborhood structures within an embedded variable neighborhood descent local improvement procedure is described. As an alternative, an ant colony optimization (ACO) approach that makes use of the same local improvement as VNS is proposed. Finally, experimental results conclude this section documenting that the ACO usually obtains better results than the VNS at the costs of longer running times.

4.3.1. Formal Problem Definition

We assume that a set $\mathcal{S} = \{1, \dots, n\}$ of rectangular, geometrically identical shreds is given, which represent the output of a shredding device. Let shreds $1, \dots, n-1$ be the shreds on which (parts of) the original document's text is printed, while all blank shreds are replaced by the single special shred n for modeling reasons. For simplicity, we assume here that the orientation of all the shreds is known or is identified during a preprocessing step based on pattern recognition techniques, see for example [16].

In addition, two error estimation functions (or cost functions) $\mathrm{c}(i,j) \geq 0$ and $\overline{\mathrm{c}}(i,j) \geq 0$ are given. They estimate the potential error introduced when placing shred j right next to shred i or by placing shred i on top of shred j, with $i, j \in \mathcal{S}$, respectively. For this section we assume that the error estimation function c_2 represented by Eq. (4.12) is straightforward adapted for representing $\mathrm{c}(i,j)$ and $\overline{\mathrm{c}}(i,j)$, respectively. Unfortunately, preliminary tests revealed that the performance of the above presented error estimation functions suffer from the fact that due to the shorter edges of the shreds to be aligned the information content is considerably reduced. Therefore, we extended the image data from B/W images to grayscale images. Although this extension did not significantly improve the results for the reconstruction of strip shredded text documents, it enhanced the output of RCCSTD.

The goal of RCCSTD is to find an assignment of shreds to positions within a solution such that the total costs induced by all realized neighborhoods are minimized. For this purpose, we define a solution of RCCSTD as an injection $\Pi : \mathcal{S} \setminus \{n\} \rightarrow \mathbb{D}^2$ of shreds to positions $p = (x, y)$ in the two-dimensional (Euclidean) space, with $x, y \in \mathbb{D} = \{1, \dots, n-1\}$, i.e., to each position is at most one shred assigned.

Furthermore, let

$$\mathrm{s}(p) = \begin{cases} i & \text{if there exists a shred } i \in \mathcal{S} \text{ such that } \Pi(i) = p \\ n & \text{otherwise} \end{cases}, \quad \forall p \in {\mathbb{D}_0}^2, \tag{4.33}$$

with $\mathbb{D}_0 = \{0, \dots, n\}$. I.e., if a shred is placed at position p, it is returned by $\mathrm{s}(p)$; otherwise the position is assumed to be filled with the special empty shred n. Let

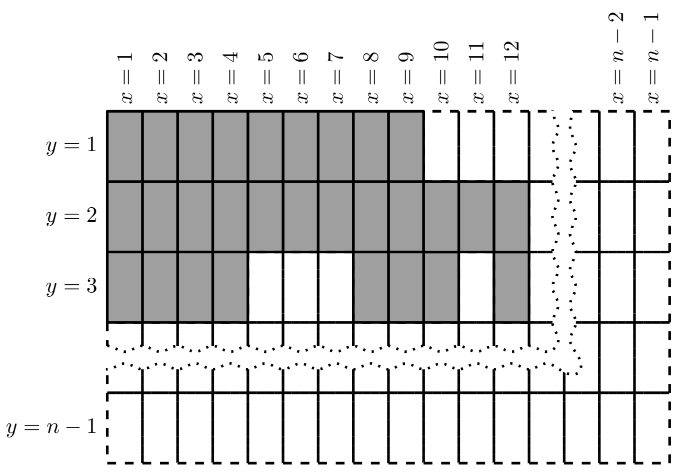


Figure 4.16.: Sketch of a solution. Any of the $(n-1) \cdot (n-1)$ available positions may be occupied (dark shaded); all others are left free (light shaded).

us denote by $\mathrm{s}_\mathrm{l}(p)$, $\mathrm{s}_\mathrm{r}(p)$, $\mathrm{s}_\mathrm{t}(p)$ and $\mathrm{s}_\mathrm{b}(p)$, with $p = (x, y) \in \mathbb{D}^2$, shreds $\mathrm{s}((x-1, y))$, $\mathrm{s}((x+1, y))$, $\mathrm{s}((x, y-1))$ and $\mathrm{s}((x, y+1))$, respectively, such that the costs of a solution, i.e., the total potential error, can be defined as

$$c(\Pi) = \sum_{p \in \{1,\dots,n\}^2} \mathrm{c}(\mathrm{s}_\mathrm{l}(p), \mathrm{s}(p)) + \overline{\mathrm{c}}(\mathrm{s}_\mathrm{t}(p), \mathrm{s}(p)). \tag{4.34}$$

A sketch of a solution is shown in Fig. 4.16. Note that rows and columns of the solution may contain multiple entries of the virtual shred n, whereas all other shreds may not be contained more than once.

Although this solution representation might look unhandy and many positions $p \in \mathbb{D}^2$ are empty, i.e., $\mathrm{s}(p) = n$, this representation allows that well matching sequences of shreds are not frequently forced to be wrapped at the end of a row or column due to a limited number of rows or columns. Anyhow, an efficient implementation must always bear in mind that there are large regions of the potential solution space $\mathbb{D}^2$ containing no assigned shreds in $\mathcal{S} \setminus \{n\}$. If the dimensions of the original document are known, the solution space may be defined smaller. Here, however, we want to stay more general.

Remark on Complexity

The problem as considered in this paper is obviously a generalization of the reconstruction of strip shredded text documents (RSSTD) since any RSSTD instance can also be solved by any algorithm for RCCSTD. This can be achieved by requesting $\overline{\mathrm{c}}(i, j) = \infty$

for $i, j \in \mathcal{S} \setminus \{n\}$. Obviously, any (optimal) solution to this so derived instance consists of only one row having all shreds placed next to each other. Due to the $\mathcal{NP}$-hardness of RSSTD RCCSTD is $\mathcal{NP}$-hard, too.

4.3.2. Construction Heuristics

For quickly creating reasonable initial solutions to be used by a VNS as well as an ACO, we propose five different construction heuristics based on different ideas and observations. They mainly try to achieve good neighborhood relationships according to function $\mathrm{c}(i, j)$ only, with $i, j \in \mathcal{S}$, since function $\overline{\mathrm{c}}(i, j)$ is merely conditionally meaningful due to the observation that the width of a shred is in comparison to its height typically relatively small in practice, see also Fig. 4.16. Furthermore, if a horizontal cut occurred between two lines of written text, i.e., no letters or other printed characters were cut, the corresponding edges are blank and therefore no reliable conclusion on the shreds vertical placement can be drawn.

Greedy Matching Heuristic

In the *greedy matching heuristic* (GMH) a first intermediate solution is generated by grouping the shreds into pairs. In each iteration, the pair of shreds that is most likely placed side by side in horizontal direction, i.e., the pair (i, j) that minimizes function $\mathrm{c}(i, j)$, with $i, j \in \mathcal{S} \setminus \{n\}$, is chosen. These two shreds are then removed from further consideration and the search for pairs is continued until all shreds got assigned partners. In the case of an odd number of snippets, a remaining one is not matched. Now, the whole process is iterated, trying to find best matchings of larger and larger sequences, until one long sequence of shreds is obtained. Finally, this single sequence is broken apart into multiple lines such that the end of each row except the last one, which contains all remaining shreds, is a shred having a blank right edge.

Perfect Matching Heuristic

Similarly to GMH the *perfect matching heuristic* (PMH) tries to iteratively find matchings of shreds in $\mathcal{S} \setminus \{n\}$. In contrast to GMH, this is not done using a greedy procedure but by finding a perfect minimum costs matching in each iteration. Obviously, a nearly perfect matching is computed if $|\mathcal{S} \setminus \{n\}|$ is odd. The resulting single row of snippets is, analogously to GMH, split into multiple lines. The matching is obtained via directly solving the following *integer linear programming* (ILP) formulation (4.35) by applying

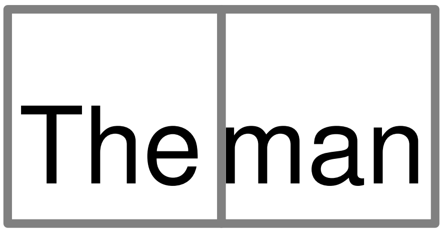


Figure 4.17.: An example for a cutting such that a blank edge and a non-blank edge have to be matched in a perfect solution.

the general purpose ILP solver CPLEX 11.2:

$$\min \sum_{i \in \mathcal{S} \setminus \{n\}} \sum_{j \in \mathcal{S} \setminus \{n\}} x_{i,j} \cdot \mathrm{c}(i,j) \tag{4.35.1}$$

$$\text{s.t.} \sum_{i \in \mathcal{S} \setminus \{n\}} x_{i,j} \leq 1, \qquad \forall j \in \mathcal{S} \setminus \{n\} \tag{4.35.2}$$

$$\sum_{i \in \mathcal{S} \setminus \{n\}} \sum_{j \in \mathcal{S} \setminus \{n\}} x_{i,j} \geq \frac{|\mathcal{S} \setminus \{n\}|}{2} \tag{4.35.3}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i,j \in \mathcal{S} \setminus \{n\} \tag{4.35.4}$$

Obviously, the goal is to minimize the costs with respect to the matched strips, cf. Eq. (4.35.1), while computing a nearly perfect matching, see constraints (4.35.2) and (4.35.3). The domain of the variables is specified by expression (4.35.4), whereas $x_{i,j} = 1$ corresponds to matching strip i and j, with $i, j \in \mathcal{S} \setminus \{n\}$.

Row Building Heuristic

The *row building heuristic* (RBH) is based on the observation that in a perfect solution (under the assumption that all shreds are available) each reconstructed row of shreds starts with a shred having a blank left edge and ends with a shred having a blank right edge. Therefore, RBH places a randomly chosen blank-left-edge snippet at the first position of the current row and continues by placing the best fitting shred with respect to $\mathrm{c}(i,j)$ next to it. This greedy best fit procedure is repeated until a snippet is reached with a blank right edge, which constitutes the end of the current row. Unfortunately, two special cases can occur: Firstly, it may happen that not all shreds are utilized when constructing a solution according to this procedure. In this case, the remaining shreds are purely randomly placed at the bottom of the constructed solution. Secondly, the number of shreds having a blank left edge needs not to be equal to the number of shreds having a blank right edge; for an example see Fig. 4.17. Additionally, more than one shred having a blank left edge might be used during the construction of the current row. If no more blank-left-edge shreds are available, the situation results in the first case. If no more blank-right-edge shreds are available, all other shreds have been used (including all blank left edge shreds). Therefore, no further actions have to be performed and the resulting solution is returned.

Multiple Paths Heuristic

Based on the same idea as RBH, the *multiple paths heuristic* (MPH) tries to find a set of rows to be aligned with each other such that the original document is reconstructed. In contrast to RBH, the rows are not built greedily but a solution is searched which is globally optimal with respect to cost function $\mathrm{c}(i,j)$. In addition, it is assured that each available shred is assigned to exactly one row, i.e., there are no shreds to be positioned randomly in the last row. For this purpose, the following *integer linear programming* (ILP) formulation is used:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{c}(i,j)\cdot x_{ij} \tag{4.36.1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i \in \mathcal{S}\setminus\{n\} \tag{4.36.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \mathcal{S}\setminus\{n\} \tag{4.36.3}$$

$$x_{ii} = 0, \qquad \forall i \in \mathcal{S} \tag{4.36.4}$$

$$x_{ij} + x_{ji} \le 1, \qquad \forall i,j \in \mathcal{S}\setminus\{n\} \tag{4.36.5}$$

$$\sum_{j=1}^{n-1} x_{nj} \ge 1 \tag{4.36.6}$$

$$\sum_{i=1}^{n-1} x_{in} \ge 1 \tag{4.36.7}$$

$$\sum_{i\in S'}\sum_{j\in S'} x_{ij} \le |S'| - 1, \qquad \forall S' \subseteq \mathcal{S}\setminus\{n\} \tag{4.36.8}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j \in \mathcal{S} \tag{4.36.9}$$

Within this model, the binary variable x_{ij}, with $i,j \in \mathcal{S}$, is set to one iff the right edge of shred i is matched with the left edge of shred j. While the objective (4.36.1) is to minimize the (potential) error introduced by these matchings, a solution is searched such that

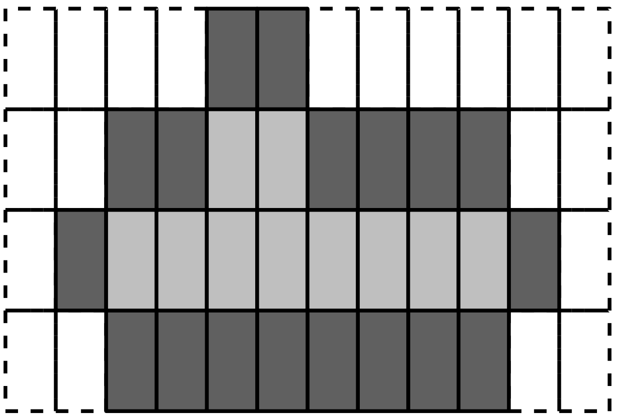


Figure 4.18.: Prim iteration. Potential placements (dark shaded) of the next shred for expanding the current solution (light shaded).

each shred except the special shred n has exactly one shred assigned to its left and exactly one to its right edge (Eqs. (4.36.2) and (4.36.3)). By constraints (4.36.4) and (4.36.5) it is assured that no loops and cycles of length two occur, respectively. Equations (4.36.6) and (4.36.7) ensure that at least one row is built. Finally, expression (4.36.8) avoids arbitrary length cycles not including the virtual shred n.

For obtaining solutions based on this ILP formulation, we apply the general purpose ILP solver CPLEX 11.2. Due to the fact that the number of constraints represented by Eq. (4.36.8) is not polynomially bounded, we add only violated constraints during the *Branch&Bound* process by first checking the solution on validity, i.e., checking whether there are cycles not containing the virtual shred n, and adding the corresponding violated constraint. The check whether or not a solution is valid can be performed during the decoding of the solution in complexity $O(n)$. When decoding the obtained solution, the rows are randomly arranged since no information with respect to this order is given by the above presented model.

Prim-Based Heuristic

In contrast to the so far presented construction heuristics the *Prim-Based Heuristic* (PBH) follows the idea exploited by the algorithm of Prim [108] for finding minimum spanning trees. Analogously to this well-known greedy algorithm, the solution is constructed by starting with an arbitrarily chosen shred that is placed at position $p = (1, 1)$. During the next steps, the intermediate solution is extended by adding one shred at a time which currently is the best matching one, i.e., which minimizes the additional error introduced by assigning it. Anyhow, possible positions for the next shred to be placed are just those positions having at least one of its four neighbors, i.e., the positions directly left, right, on top or at bottom, occupied, see also Fig. 4.18. In case that the best position for the next shred would be either $p = (0, y)$ or $p = (x, 0)$, with $1 \leq x, y \leq n-1$, all shreds of the current intermediate solution are shifted one position to the right or to the bottom, respectively. Of course, the finally obtained solution can be of arbitrary shape, i.e., any placement of shreds can be obtained, as long as all shreds are connected to one component.

Experimental Results

Within this section a comparison of the proposed construction heuristics is done. All were implemented in Java and the computational tests were performed on a single core of an Intel® Core™2 Quad CPU with 8 GB RAM and 2.83 GHz. The input instances were generated as follows: As a foundation for the instances we used the first five of the document pages already used for the experiments on the reconstruction of strip shredded documents, i.e., p01 to p05, also cf. Appx. A. This time, however, the documents were transformed into grayscale images and then shredded into nine instances with 9×9 to 15×15 snippets each, which results in a total of 45 different input instances. The adapted error estimation function as described above was used for evaluating solutions.

Table 4.7 shows the results obtained using RBH, PBH, GMH and MPH for all of these instances. Since preliminary tests revealed that PMH performs in most cases worse than GMH and in all cases worse than any other construction heuristic, no detailed results are presented for this method.

The first three columns (x and y) indicate characteristics of the corresponding instance, i.e., the page and the number of shreds along the x- and y-axis, respectively. The fourth column shows the objective value of the perfectly reconstructed document page, i.e., the original sheet of paper. In the following columns the mean percentage gaps over 20 runs with respect to the objective value of the original document page as well as the standard deviations in parentheses are presented for each construction heuristic, i.e., a value of 100% indicates that the found solution is twice as bad as the arrangement of shreds representing the original document. We can observe that MPH often yields the best, i.e., lowest, objective value. Wilcoxon rank sum tests have been performed to check in which cases MPH actually yields statistically better solutions than RBH, PBH and GMH, respectively. The results are given in the corresponding columns labeled p, whereas an entry of $>$ indicates that MPH is significantly better with an error level of 5% and $<$ states that the other heuristic performed better. If none of these two cases holds, then $\approx$ is shown in the corresponding field.

Table 4.7.: Average percentage gaps and corresponding standard deviations for the four construction heuristics are listed. Results of Wilcoxon rank sum tests for the hypothesis that MPH performs better than each of the other construction heuristics are given in columns p (using a 5% error level).

				RBH			PBH			GMH			MPH	
	x	y	orig	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev
instance p01	9	9	2977	147.7%	(27.9)	$\approx$	172.2%	(9.0)	$>$	201.6%	(0.0)	$>$	**131.9**%	(31.6)
	9	12	4051	152.3%	(15.0)	$\approx$	**142.3**%	(10.6)	$\approx$	149.3%	(0.0)	$\approx$	151.1%	(28.2)
	9	15	4215	169.0%	(29.0)	$>$	160.8%	(8.0)	$>$	194.9%	(0.0)	$>$	**147.2**%	(22.4)
	12	9	4125	115.6%	(18.1)	$\approx$	140.2%	(9.8)	$>$	123.2%	(0.0)	$>$	**109.1**%	(25.1)
	12	12	4937	145.4%	(13.0)	$>$	**123.6**%	(8.1)	$\approx$	155.4%	(0.0)	$>$	129.8%	(19.5)
	12	15	5147	172.8%	(17.4)	$\approx$	130.7%	(8.0)	$<$	167.7%	(0.0)	$<$	**179.7**%	(20.5)
	15	9	4099	166.9%	(21.1)	$>$	161.1%	(8.7)	$>$	113.1%	(0.0)	$>$	**101.2**%	(24.2)
	15	12	4922	150.2%	(20.0)	$>$	142.6%	(8.6)	$>$	156.9%	(0.0)	$>$	**113.3**%	(15.2)
	15	15	5142	150.5%	(17.6)	$>$	135.7%	(6.6)	$\approx$	158.9%	(0.0)	$>$	**138.0**%	(14.2)
instance p02	9	9	1786	229.2%	(15.5)	$>$	186.6%	(28.1)	$>$	159.8%	(0.0)	$>$	**141.7**%	(11.2)
	9	12	1538	335.2%	(22.3)	$>$	235.6%	(32.6)	$>$	191.4%	(0.0)	$>$	**190.0**%	(15.7)
	9	15	2462	249.4%	(10.6)	$>$	144.6%	(21.5)	$\approx$	**132.0**%	(0.0)	$<$	145.6%	(12.3)
	12	9	1757	175.2%	(24.5)	$>$	228.3%	(38.5)	$>$	173.6%	(0.0)	$>$	**132.8**%	(14.8)
	12	12	1568	251.2%	(15.7)	$>$	273.5%	(28.6)	$>$	199.0%	(0.0)	$>$	**181.3**%	(7.4)
	12	15	2398	200.6%	(12.5)	$>$	168.3%	(18.8)	$>$	**123.4**%	(0.0)	$<$	134.2%	(11.9)
	15	9	2116	129.5%	(11.1)	$>$	243.7%	(22.6)	$>$	139.1%	(0.0)	$>$	**109.6**%	(24.8)
	15	12	2075	150.8%	(11.1)	$>$	255.5%	(18.6)	$>$	150.7%	(0.0)	$>$	**129.0**%	(12.4)
	15	15	2864	118.2%	(13.8)	$>$	183.4%	(13.1)	$>$	123.9%	(0.0)	$>$	**108.9**%	(9.5)
instance p03	9	9	3245	128.9%	(17.3)	$\approx$	130.6%	(13.4)	$\approx$	172.4%	(0.0)	$>$	**123.3**%	(25.4)
	9	12	3398	164.6%	(18.0)	$\approx$	**153.0**%	(12.3)	$<$	175.3%	(0.0)	$\approx$	169.2%	(34.3)
	9	15	3294	169.9%	(25.0)	$\approx$	171.5%	(16.4)	$\approx$	**153.9**%	(0.0)	$\approx$	159.2%	(36.7)
	12	9	4049	**96.8**%	(15.8)	$\approx$	137.3%	(9.6)	$>$	129.7%	(0.0)	$>$	105.7%	(18.7)
	12	12	4330	146.7%	(16.7)	$\approx$	**129.8**%	(13.0)	$<$	135.1%	(0.0)	$<$	147.5%	(15.9)
	12	15	4264	143.3%	(16.4)	$>$	136.2%	(10.4)	$\approx$	142.3%	(0.0)	$>$	**129.4**%	(20.1)
	15	9	4195	120.4%	(16.6)	$>$	140.0%	(7.9)	$>$	102.6%	(0.0)	$>$	**89.1**%	(24.1)
	15	12	4242	200.5%	(10.3)	$>$	149.5%	(11.6)	$\approx$	**110.9**%	(0.0)	$<$	142.0%	(15.5)
	15	15	4270	182.5%	(11.3)	$>$	140.1%	(11.7)	$\approx$	142.1%	(0.0)	$\approx$	**138.6**%	(16.7)
instance p04	9	9	1411	184.4%	(39.5)	$\approx$	207.6%	(21.0)	$>$	229.6%	(0.0)	$>$	**182.7**%	(24.3)
	9	12	1892	221.9%	(27.3)	$>$	**176.9**%	(19.9)	$\approx$	220.7%	(0.0)	$>$	184.5%	(26.5)
	9	15	1979	157.8%	(19.4)	$>$	150.9%	(20.8)	$\approx$	171.0%	(0.0)	$>$	**142.7**%	(18.2)
	12	9	2037	188.6%	(23.2)	$>$	194.0%	(16.2)	$>$	152.8%	(0.0)	$\approx$	**149.4**%	(16.7)
	12	12	2689	139.1%	(14.5)	$>$	143.5%	(14.1)	$>$	159.1%	(0.0)	$>$	**121.8**%	(17.1)
	12	15	2734	119.2%	(9.8)	$>$	122.9%	(10.0)	$>$	122.9%	(0.0)	$>$	**107.2**%	(3.4)
	15	9	2193	164.3%	(12.5)	$>$	206.5%	(19.1)	$>$	140.4%	(0.0)	$>$	**113.7**%	(23.9)
	15	12	2481	**135.3**%	(13.9)	$\approx$	184.9%	(13.8)	$>$	145.7%	(0.0)	$\approx$	144.3%	(24.4)
	15	15	2719	**106.2**%	(18.6)	$\approx$	138.6%	(14.3)	$>$	122.8%	(0.0)	$>$	107.6%	(17.3)
instance p05	9	9	923	211.0%	(36.3)	$\approx$	343.0%	(50.7)	$>$	245.3%	(0.0)	$>$	**197.3**%	(38.0)
	9	12	1293	325.6%	(40.5)	$\approx$	**302.6**%	(34.6)	$\approx$	333.9%	(0.0)	$>$	316.8%	(33.5)
	9	15	2123	264.7%	(19.8)	$>$	245.8%	(16.9)	$\approx$	265.7%	(0.0)	$>$	**243.4**%	(27.6)
	12	9	1365	177.8%	(15.8)	$>$	251.8%	(28.4)	$>$	171.5%	(0.0)	$>$	**154.4**%	(17.6)
	12	12	1841	236.7%	(30.0)	$>$	258.7%	(21.1)	$>$	222.9%	(0.0)	$>$	**204.8**%	(26.7)
	12	15	2588	219.8%	(15.0)	$\approx$	**204.1**%	(15.0)	$<$	214.5%	(0.0)	$<$	216.6%	(17.1)
	15	9	1317	175.6%	(21.1)	$>$	304.5%	(41.1)	$>$	193.5%	(0.0)	$>$	**117.0**%	(27.8)
	15	12	1634	304.6%	(21.6)	$>$	299.0%	(24.6)	$>$	260.4%	(0.0)	$>$	**233.5**%	(34.9)
	15	15	2460	253.3%	(14.1)	$>$	225.7%	(17.8)	$>$	**197.1**%	(0.0)	$<$	205.1%	(13.0)

Regarding the mean values—the best obtained are printed bold—MPH yielded 30 times the best average value while PBH obtained only seven times the best result. GMH and RBH achieved the best value in five and two cases, respectively. Nevertheless, the page could never be perfectly reconstructed. MPH obtained for 32, 30 and 28 instances statistically better results than GMH, RBH and PBH, respectively.

Since GMH is completely deterministic, the standard deviations are zero. Since the standard deviations seem to be rather high for the other three construction heuristics, it has to be mentioned that according to the cost function used, even the swapping of two shreds can significantly increase (or decrease) the objective function. Therefore, these high values have to be relativized. Nevertheless, they show that certain fluctuations are existent.

4.3.3. Variable Neighborhood Search based Approach

As already indicated the construction heuristics presented in the previous section are not intended to be used as stand-alone reconstruction approaches but as initialization methods for more elaborated optimization approaches. Within this section, we propose a variable neighborhood search (VNS) with variable neighborhood descent (VND) as embedded local improvement procedure.

Naturally, it is therefore convenient to first introduce a set of moves which will then be used for defining neighborhood structures to be systematically examined during a local improvement phase of VNS. The neighborhood structures used within the shaking step of VNS are analogously defined.

Definition of Neighborhood Structures

The underlying move types are on the one hand inspired by the natural behavior of humans when trying to reconstruct cross cut shredded text documents. Namely, the insertion of shreds to specific positions while considering the direct neighborhood of the affected position(s). This results in movements either horizontally and/or vertically shifting one shred to another position or by swapping two shreds with each other. On the other hand, it is tried to keep the moves as simple as possible since more complex operations also induce long and inefficient evaluation and reorganization steps which obviously negatively influences especially the runtime performance of the developed approach(es). Therefore, the following two move types can be defined:

SwapMove(i, j): When applying a swap move, two shreds i and j, with $i, j \in \mathcal{S}$ and $i \neq j$, are swapped with each other.

ShiftMove(p, w, h, d, a): In a first step, a rectangular region of snippets to be moved is defined. Parameter $p = (x, y) \in \mathbb{D}^2$ corresponds to the position of the top-left shred to be moved. The integer values $w \geq 1$ and $h \geq 1$ define the number of shreds along the x-axis and along the y-axis to be shifted. The direction, i.e., horizontally or vertically, is declared by d and the shift amount is given by $a \geq 1$. Therefore, after the application of the shift move all shreds contained within the specified region are moved according to d and a. Previously adjacent shreds are suitably shifted.

This definition leads straightforward to the following seven neighborhood structures:

$\mathcal{N}_1$: Within this neighborhood structure one single swap move is applied to the current solution.

$\mathcal{N}_2$: Neighborhood $\mathcal{N}_2(\Pi)$ of a solution Π consists of all solutions obtained from Π by arbitrarily shifting one single shred in either x or y direction.

$\mathcal{N}_3$: All solutions generated by applying a shift move with at least one of the parameters w and h set to one are part of this neighborhood structure.

$\mathcal{N}_4$: Within neighborhood structure $\mathcal{N}_4$ one shift move is applied, whereas the width and the height of the rectangular region of shreds to be shifted can be chosen arbitrarily.

$\mathcal{N}_5$: Two consecutive moves are applied to a single shred, whereas the first move shifts along the x-axis and the second one shifts along the y-axis.

$\mathcal{N}_6$: Neighborhood $\mathcal{N}_6(\Pi)$ consists of all solutions obtained by shifting a given rectangle of either width or height one first along the x-axis and then along the y-axis.

$\mathcal{N}_7$: This last neighborhood structure is defined analogously to $\mathcal{N}_6$, but this time the width and the height of the rectangular region can be both arbitrarily chosen.

As can be easily seen, neighborhood structure $\mathcal{N}_i$ contains $\mathcal{N}_{i-1}$ for $i = 2, 3, 4, 6, 7$. Thus, the number of candidate solutions within $\mathcal{N}_i(\Pi)$ is in general greater than the number of solutions contained in $\mathcal{N}_{i-1}(\Pi)$ for a given solution Π. Therefore, it is obvious to order the neighborhood structures according to their increasing size such that the smallest one is examined first.

Variable Neighborhood Descent

To efficiently implement a VND, the following two practical improvements are made: Firstly, all neighbors are evaluated using an incremental update function, i.e., only the changes in function (4.34) are computed. Secondly, and more importantly, two properties have to be fulfilled for a feasible move to be part of a neighborhood structure: At least one position $p \in \mathbb{D}^2$ with a shred assigned to p, i.e., $\mathrm{s}(p) \neq n$, has to be affected by this move and in case of shift moves the dimensions of the rectangle to be shifted as well as the shift amount (and direction) have to be chosen such that each row and column of the region to be shifted is not empty, i.e., at least one non-empty shred is part of the row/column, and $\mathrm{s}((x+w+a-1, y)) \neq n$ holds, if a horizontal shift is performed; otherwise $\mathrm{s}((x, y+h+a-1)) \neq n$ must hold. In our VND approach all neighborhoods are examined using a *next improvement* strategy since preliminary tests revealed that in this case this strategy outperforms *best improvement*.

Variable Neighborhood Search

While the above presented VND is used as local improvement procedure we define the neighborhood structures N_i, with $1 \leq i \leq 5$, used during the shaking phase of VNS as follows: in the i-th neighborhood structure i^2 randomly chosen shift moves of single shreds are performed. For computational results we refer to Sec. 4.3.5.

4.3.4. Ant Colony Optimization Based Approach

While in nature ants are guided along paths between food locations and their home by pheromone trails laid by other ants in most computer system inspired by this ant behavior additional locally available knowledge is incorporated in the solution construction process. For our *ant colony optimization* (ACO) approach, two pheromone matrices τ and $\overline{\tau}$ exist, whereas values τ_{ij} and $\overline{\tau}_{ij}$ correspond to the amount of pheromone laid for placing shred j right next to shred i and placing shred i on top of shred j, respectively. Both matrices are initialized within two steps, whereas during the first step five solutions $\Pi_1, \ldots, \Pi_5$ are computed with the construction heuristics presented in Sec. 4.3.2, i.e., GMH, PMH, RBH, MPH and PBH. Based on the best obtained solution within this first step, an initial value τ^0 is computed by

$$\tau^0 = \frac{m}{\min_{i=1,\ldots,5} c(\Pi_i)}, \tag{4.37}$$

whereas m denotes the number of ants being used within the ACO. Subsequently, all values τ_{ij} and $\overline{\tau}_{ij}$, with $i, j \in \mathcal{S}$, are set to τ^0. In the second step, a regular pheromone update (see the corresponding section in the following) is performed using initial solutions Π_1 to Π_5.

Table 4.8.: Results obtained by VNS and ACO. The mean percentage gaps over 20 runs and standard deviations are presented for two independent test sets of VNS initialized using PBH and MPH as well as the mean gaps (over 20 runs) and standard deviations of 4 different ACO variants incorporating RPBH, RGMH, RRBH and all three of them, respectively. Values in columns p correspond to the results of Wilcoxon rank sum tests using a 5% error level.

				VNS-PBH			VNS-MPH			ACO			ACO-RPBH			ACO-RGMH			ACO-RRBH	
	x	y	orig.	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev
instance p01	9	9	2977	40.8%	(12.5)	$>$	**10.0**%	(12.7)	$>$	4.0%	(5.9)	$>$	23.8%	(8.1)	$>$	29.1%	(9.5)	$>$	**0.0**%	(0.0)
	9	12	4051	28.7%	(11.6)	$>$	**20.8**%	(8.8)	$\approx$	**14.6**%	(5.1)	$>$	27.0%	(4.9)	$>$	27.8%	(3.7)	$>$	21.0%	(4.8)
	9	15	4215	34.7%	(8.0)	$>$	**28.6**%	(7.7)	$\approx$	31.7%	(3.6)	$>$	34.8%	(2.8)	$>$	35.6%	(2.7)	$>$	**30.7**%	(2.1)
	12	9	4125	**26.4**%	(5.9)	$\approx$	28.4%	(9.3)	$>$	27.6%	(3.3)	$>$	27.2%	(4.0)	$>$	27.0%	(3.0)	$>$	**25.2**%	(3.2)
	12	12	4937	26.5%	(6.1)	$\approx$	**24.8**%	(5.9)	$\approx$	29.6%	(2.2)	$>$	34.5%	(4.0)	$>$	30.9%	(3.0)	$>$	**27.5**%	(2.9)
	12	15	5147	**30.4**%	(6.6)	$\approx$	31.0%	(10.6)	$\approx$	34.0%	(2.7)	$\approx$	33.1%	(2.9)	$\approx$	34.7%	(2.8)	$>$	**32.7**%	(3.2)
	15	9	4099	37.5%	(11.2)	$>$	**30.1**%	(7.8)	$\approx$	32.8%	(3.7)	$>$	32.8%	(4.7)	$>$	33.7%	(3.6)	$>$	**29.1**%	(4.1)
	15	12	4922	32.0%	(5.8)	$>$	**28.2**%	(7.6)	$<$	31.9%	(3.6)	$\approx$	34.9%	(2.7)	$>$	**31.4**%	(3.1)	$\approx$	32.5%	(2.6)
	15	15	5142	**32.3**%	(5.2)	$\approx$	**32.3**%	(5.8)	$\approx$	35.3%	(3.1)	$>$	36.5%	(2.3)	$>$	36.6%	(3.9)	$>$	**33.2**%	(3.2)
instance p02	9	9	1786	**2.1**%	(9.0)	$\approx$	3.2%	(12.5)	$\approx$	0.2%	(3.7)	$<$	**-4.1**%	(5.7)	$<$	-2.5%	(5.1)	$<$	4.8%	(4.6)
	9	12	1538	**23.0**%	(9.6)	$<$	34.6%	(14.1)	$\approx$	17.0%	(3.7)	$<$	**16.9**%	(4.5)	$<$	17.6%	(3.9)	$<$	35.3%	(5.1)
	9	15	2462	**8.4**%	(4.4)	$<$	15.6%	(6.2)	$<$	8.0%	(2.5)	$<$	**6.9**%	(2.6)	$<$	14.0%	(2.5)	$<$	21.7%	(3.6)
	12	9	1757	**6.0**%	(9.7)	$\approx$	9.6%	(11.1)	$\approx$	9.0%	(6.3)	$<$	**3.3**%	(6.3)	$<$	11.3%	(7.1)	$\approx$	13.4%	(4.2)
	12	12	1568	**22.6**%	(9.1)	$<$	29.4%	(9.8)	$\approx$	24.8%	(4.7)	$<$	**21.9**%	(4.1)	$<$	25.7%	(4.1)	$\approx$	27.7%	(5.0)
	12	15	2398	**9.4**%	(6.0)	$<$	19.0%	(10.9)	$\approx$	14.4%	(3.1)	$<$	**12.4**%	(2.4)	$<$	20.4%	(3.1)	$\approx$	19.7%	(3.5)
	15	9	2116	19.8%	(10.1)	$>$	**14.0**%	(10.7)	$\approx$	16.3%	(3.5)	$>$	25.6%	(3.4)	$>$	16.8%	(3.6)	$\approx$	**13.8**%	(4.0)
	15	12	2075	27.3%	(11.4)	$\approx$	**24.6**%	(9.0)	$>$	17.2%	(3.4)	$<$	34.3%	(5.4)	$>$	**15.9**%	(3.8)	$<$	19.8%	(3.3)
	15	15	2864	16.2%	(5.0)	$\approx$	**17.5**%	(8.1)	$>$	13.8%	(1.9)	$>$	22.2%	(2.8)	$>$	17.1%	(2.9)	$>$	**12.6**%	(1.7)
instance p03	9	9	3245	25.2%	(8.2)	$>$	**21.3**%	(8.0)	$>$	11.5%	(4.3)	$>$	18.9%	(3.4)	$>$	18.5%	(4.0)	$>$	**6.7**%	(4.8)
	9	12	3398	**34.1**%	(4.1)	$\approx$	34.4%	(11.0)	$>$	**29.9**%	(3.6)	$\approx$	37.4%	(3.5)	$>$	32.0%	(4.5)	$\approx$	30.0%	(3.6)
	9	15	3294	25.4%	(14.3)	$>$	**11.6**%	(6.7)	$<$	21.7%	(5.3)	$>$	31.1%	(6.4)	$>$	24.3%	(2.2)	$>$	**18.0**%	(2.6)
	12	9	4049	23.4%	(7.3)	$>$	**16.4**%	(6.1)	$>$	14.7%	(2.0)	$>$	23.6%	(3.1)	$>$	17.4%	(2.6)	$>$	**11.6**%	(2.9)
	12	12	4330	25.1%	(4.6)	$>$	**19.9**%	(6.1)	$<$	24.5%	(3.3)	$>$	39.1%	(3.4)	$>$	25.7%	(3.3)	$>$	**23.3**%	(2.5)
	12	15	4264	**20.9**%	(6.9)	$\approx$	23.2%	(6.2)	$>$	17.4%	(2.4)	$>$	33.0%	(4.9)	$>$	18.5%	(2.5)	$>$	**16.3**%	(2.1)
	15	9	4195	**20.6**%	(6.2)	$\approx$	21.8%	(10.0)	$>$	16.1%	(3.1)	$>$	31.1%	(5.0)	$>$	24.2%	(2.2)	$>$	**13.6**%	(2.3)
	15	12	4242	**33.1**%	(6.4)	$\approx$	35.7%	(8.1)	$\approx$	37.5%	(3.2)	$\approx$	55.7%	(5.2)	$>$	38.9%	(3.5)	$>$	**36.3**%	(2.1)
	15	15	4270	**27.3**%	(4.7)	$\approx$	30.0%	(5.1)	$>$	26.3%	(2.2)	$\approx$	35.8%	(4.0)	$>$	**25.7**%	(2.4)	$<$	27.0%	(1.4)
instance p04	9	9	1411	**28.7**%	(16.5)	$\approx$	35.8%	(11.5)	$>$	21.5%	(6.4)	$\approx$	**18.3**%	(7.8)	$\approx$	20.7%	(9.4)	$\approx$	18.4%	(7.2)
	9	12	1892	**21.7**%	(9.9)	$<$	28.0%	(10.1)	$>$	15.0%	(4.8)	$\approx$	14.5%	(5.0)	$\approx$	**14.2**%	(3.4)	$\approx$	15.6%	(4.5)
	9	15	1979	**9.4**%	(8.5)	$<$	16.0%	(6.3)	$>$	3.9%	(3.8)	$\approx$	**2.4**%	(3.3)	$\approx$	5.8%	(2.8)	$>$	3.8%	(3.3)
	12	9	2037	**36.6**%	(10.4)	$\approx$	38.3%	(11.0)	$>$	29.1%	(4.5)	$>$	35.7%	(6.8)	$<$	38.4%	(5.8)	$>$	**26.3**%	(5.3)
	12	12	2689	**20.8**%	(5.1)	$<$	28.5%	(5.0)	$>$	19.2%	(2.4)	$<$	**15.9**%	(2.6)	$<$	21.9%	(2.4)	$\approx$	20.6%	(1.9)
	12	15	2734	**6.1**%	(3.6)	$<$	12.2%	(5.7)	$>$	**6.8**%	(2.8)	$\approx$	7.1%	(1.6)	$\approx$	10.4%	(1.7)	$>$	7.6%	(1.8)
	15	9	2193	41.4%	(13.4)	$>$	**26.6**%	(12.8)	$>$	18.2%	(3.5)	$>$	43.5%	(5.2)	$>$	26.0%	(3.5)	$>$	**12.5**%	(4.1)
	15	12	2481	**36.9**%	(10.5)	$\approx$	38.0%	(9.7)	$>$	29.4%	(2.7)	$>$	41.7%	(5.0)	$>$	30.7%	(3.4)	$>$	**27.0**%	(2.6)
	15	15	2719	15.7%	(6.6)	$\approx$	**14.4**%	(5.4)	$>$	5.1%	(2.6)	$>$	20.8%	(3.7)	$>$	6.3%	(1.8)	$>$	**3.9**%	(1.6)
instance p05	9	9	923	57.1%	(20.6)	$\approx$	**54.7**%	(23.6)	$>$	21.1%	(8.9)	$>$	66.7%	(11.2)	$>$	36.5%	(12.0)	$>$	**13.6**%	(8.2)
	9	12	1293	**82.0**%	(20.3)	$\approx$	82.1%	(21.9)	$>$	70.4%	(9.5)	$\approx$	72.5%	(6.9)	$\approx$	78.9%	(8.7)	$>$	**69.0**%	(7.4)
	9	15	2123	**48.4**%	(9.1)	$\approx$	51.8%	(8.7)	$>$	39.3%	(6.6)	$\approx$	45.8%	(4.6)	$>$	44.2%	(5.1)	$>$	**36.2**%	(4.5)
	12	9	1365	43.5%	(8.7)	$\approx$	**42.4**%	(16.3)	$>$	27.5%	(4.2)	$>$	75.4%	(8.1)	$>$	29.7%	(3.7)	$>$	**23.4**%	(4.5)
	12	12	1841	**47.1**%	(14.4)	$\approx$	51.4%	(13.6)	$>$	40.0%	(5.1)	$>$	54.3%	(7.6)	$>$	44.5%	(5.7)	$>$	**36.6**%	(3.1)
	12	15	2588	45.9%	(6.0)	$\approx$	**44.1**%	(7.9)	$>$	35.9%	(3.1)	$>$	45.4%	(2.9)	$>$	37.6%	(3.6)	$>$	**34.5**%	(2.6)
	15	9	1317	35.5%	(29.9)	$>$	**12.8**%	(15.0)	$>$	-4.3%	(4.2)	$>$	43.7%	(13.2)	$>$	-1.2%	(3.2)	$>$	**-9.4**%	(1.4)
	15	12	1634	**53.9**%	(11.0)	$\approx$	58.7%	(15.9)	$>$	55.4%	(4.7)	$>$	67.6%	(7.1)	$>$	59.0%	(5.2)	$>$	**51.5**%	(3.9)
	15	15	2460	**42.2**%	(6.4)	$\approx$	42.9%	(8.3)	$>$	39.5%	(3.4)	$\approx$	44.9%	(4.7)	$>$	41.0%	(3.5)	$>$	**38.5**%	(3.7)

Solution Construction

New candidate solutions are constructed within the ACO by one of the following alternative methods, which are based on the construction heuristics GMH, RBH, and PBH presented in Sec. 4.3.2. Each candidate solution created in such a way is then also locally improved by applying a restricted version of the above presented VND (see Sec. 4.3.3) using only neighborhood structures $\mathcal{N}_1$ to $\mathcal{N}_3$ with a CPU-time limit of 500ms.

Randomized Greedy Matching Heuristic Analogously to GMH, the *randomized greedy matching heuristic* (RGMH) greedily matches shreds such that finally one long sequence of snippets is produced, which is then split into multiple rows. But instead of always fixing that pair of shreds which matches best within each iteration, we now perform this selection in a probabilistic way in dependence of pheromone values and the cost function $\mathrm{c}(i,j)$. The probability $\mathfrak{p}_{ij}$ of a match for pair (i,j), with $i,j \in \mathcal{S}'$, whereas $\mathcal{S}'$ denotes the set of shreds not yet matched, is equal to

$$\mathfrak{p}_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,j)}\right)^{\beta}}{\sum_{k \in \mathcal{S}'} \sum_{k' \in \mathcal{S}'} \tau_{kk'}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(k,k')}\right)^{\beta}}, \qquad \forall i \in \mathcal{S} \setminus \mathcal{S}', j \in \mathcal{S}' \tag{4.38}$$

In case of $\mathrm{c}(i,j) \leq \epsilon = 0.25$ for any $i,j \in \mathcal{S}'$, $\mathrm{c}(i,j)$ is assumed to be equal to ϵ. As usual within an ACO, parameters α and β are controlling the influence of pheromones versus the influence of heuristic information.

Randomized Row Building Heuristic The randomized version of RBH—called *randomized row building heuristic* (RRBH)—tries to reconstruct a set of rows based on the following probability distribution, i.e., the next matching shred is not selected solely relying on the values of $\mathrm{c}(i,j)$, with i being the last placed shred and j being any shred currently not placed, but using the probability values $\mathfrak{p}_{ij}$:

$$\mathfrak{p}_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,j)}\right)^{\beta}}{\sum_{k \in \mathcal{S}'} \tau_{ik}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,k)}\right)^{\beta}}, \qquad \forall i \in \mathcal{S} \setminus \mathcal{S}', j \in \mathcal{S}' \tag{4.39}$$

Again, set $\mathcal{S}'$ is defined as the set of all shreds not used within the current intermediate solution.

Randomized Prim Based Heuristic The *Randomized Prim based heuristic* (RPBH) is the non-deterministic variant of PBH. The decision at which position the next (randomly chosen) shred is placed is based on the following definition of probability values $\mathfrak{p}_p^i$ for placing shred $i \in \mathcal{S}'$ to position p, with set $\mathcal{S}'$ being the set of shreds not yet used:

$$\mathfrak{p}_p^i = \frac{\delta(i,p)}{\sum_{p' \in \mathbb{D}_0{}^2} \sum_{k \in \mathcal{S}'} \delta(k,p')}, \qquad \forall i \in \mathcal{S}', p \in \mathbb{D}_0{}^2 \tag{4.40}$$

Function $\delta(i,p)$, with $i \in \mathcal{S}', p \in \mathbb{D}_0{}^2$ computes the additionally introduced error when placing shred i to position p. The value of $\delta(i,p)$ is equal to zero if p is either already used by another shred $k \in \mathcal{S} \setminus \mathcal{S}'$ or all neighbor positions of p are free, i.e., no shred $k \in \mathcal{S} \setminus \mathcal{S}'$ is positioned on them (see also Fig. 4.18). Analogously to PBH, all shreds are shifted one position to the right or to the bottom if the next shred should be assigned to any position outside of $\mathbb{D}^2$.

Pheromone Update

The pheromone update is done according to the following expressions, whereas we assume that k, with $1 \le k \le m$, refers to the solution obtained by ant k during the last iteration of ACO and Π_0 represents the best so far found solution:

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta_{ij}^k + \Delta_{ij}^0, \qquad \forall i,j \in \mathcal{S},\ i \neq j \tag{4.41}$$

$$\overline{\tau}_{ij} = (1-\rho) \cdot \overline{\tau}_{ij} + \sum_{k=1}^{m} \overline{\Delta}_{ij}^k + \overline{\Delta}_{ij}^0, \qquad \forall i,j \in \mathcal{S},\ i \neq j \tag{4.42}$$

$$\Delta_{ij}^k = \begin{cases} \frac{1}{c(\Pi_k)} & \text{if } j \text{ is placed right next to } i \text{ in the } k\text{-th solution} \\ 0 & \text{otherwise} \end{cases} \qquad \begin{array}{l} \forall i,j \in \mathcal{S} \\ \forall k = 0,\dots,m \end{array} \tag{4.43}$$

$$\overline{\Delta}_{ij}^k = \begin{cases} \frac{1}{c(\Pi_k)} & \text{if } j \text{ is placed on top of } i \text{ in the } k\text{-th solution} \\ 0 & \text{otherwise} \end{cases} \qquad \begin{array}{l} \forall i,j \in \mathcal{S} \\ \forall k = 0,\dots,m \end{array} \tag{4.44}$$

The idea behind these definitions is that the placing of two shreds next to each other should be emphasized when the costs of this placement are low.

4.3.5. Experimental Results

Results obtained using the VNS based approach are presented in Tab. 4.8 together with results obtained using an ACO approach to be discussed in detail in the next section. The first four columns of this table correspond to the first four columns of

Tab. 4.7. The next two columns labeled with VNS-PBH and VNS-MPH correspond to the experiments performed with VNS initialized using PBH and MPH, respectively. Again, the values represent average gaps to the original document over 20 runs with respect to the objective function. They were obtained on the same hardware as the results presented in the previous section. The column labeled p for VNS-PBH lists the return of Wilcoxon rank sum tests comparing the two VNS variants with each other, i.e., $<$ indicates that VNS-PBH performed significantly better with an error level of 5% on the corresponding instance while $>$ indicates that VNS-MPH performed better; $\approx$ implies that no statement can be given for the corresponding instance. For each instance the better of the two average gaps is printed bold.

Although we performed tests initializing VNS with all previously presented construction heuristics, it turned out that only the variants using PBH and MPH were successful. The performance of the others was not that good and we therefore omit here the detailed results. However, it can be observed that the initial solutions could be significantly improved by the VNS approach. Nevertheless, it is not easy to decide which of these two variants performs better. Interestingly, the performance of both variants seems to be strongly dependent on the underlying page. Whereas, VNS-MPH is clearly better for page p01, VNS-PBH outperforms VNS-MPH on page p02. Unfortunately, it is not possible to select the appropriate variant based on the (then reconstructed) document in advance.

In addition, Tab. 4.8 shows test results obtained by the ACO for instances based on pages p01 to p05 are presented together with the results obtained using the VNS approach. The results obtained for different ACO settings are presented in the columns labeled with ACO, ACO-RPBH, ACO-RRBH and ACO-RGMH. The concrete settings were chosen as follows: For each variant of ACO we set the number of ants $m = 18$. The construction heuristics used by the ants were RPBH, RRBH and RGMH, respectively. The fourth setting corresponding to the column labeled with ACO was chosen such that six of the 18 ants used RPBH, another six RRBH and the last six RGMH. The value of parameter m was chosen based on preliminary tests, which also revealed that the fixing of α and β to 1 and 5, respectively, is reasonable for our ACO variants. The values presented in Tab. 4.8 are again mean percentage gaps over 20 runs, and the conclusions of selected Wilcoxon tests are given in columns labeled with p, whereas VNS-MPH, ACO, ACO-RPBH and ACO-RGMH were compared to ACO-RRBH. The corresponding p columns indicate again whether the first ($<$) or the second heuristic ($>$) yielded statistically better results on an error level of 5%. If none of these two cases occur, a $\approx$ sign is printed in the according field. In addition the best mean values of the four ACO variants is emphasized.

For the ACO variants a clear conclusion can be drawn: ACO-RRBH performs best on the considered test instances. Therefore, we decided to compare VNS-MPH with ACO-RRBH and observed that the results obtained by the latter one were for 28 instances significantly better. When comparing VNS and ACO in general, the two VNS variants achieved best mean results only on 11 instances whereas the ACO variant reached 35 times the best mean value (29 times this value was provided by ACO-RRBH).

Taking a closer look at the values in Tab. 4.8 it can be seen that for instance p01 with 9×9 shreds ACO-RRBH could always reconstruct the original document page. For some runs, the percentage gap is even negative, which can be explained by the fact that for any error estimation function it is not assured that the original document is evaluated best, see also Sec. 4.2.4 and Sec. 4.2.9 for a discussion related to this topic.

Regarding running times, we can summarize that the construction heuristics performed within hundreds of milliseconds. The VNS approaches needed between one and 100 seconds computation time until termination, and the computation times for ACO lie between approximately 100 seconds and 800 seconds. It can be concluded that although the results obtained by ACO are better in most cases, the computation times needed are significantly higher.

In general further improvements are necessary to address large practical instances especially also involving multiple pages. However, considering the complexity of the problem, the achieved results on small and medium sized instances are remarkable. Especially for those pages containing mainly text, large parts of the documents could be reconstructed.

4.3.6. Concluding Remarks

Within this section we presented five different construction heuristics, a VNS as well as an ACO. Altogether it can be concluded that these approaches are promising especially for moderately sized instances. Nevertheless, there are several questions open, which should be further investigated in future research. For example, as already indicated in Sec. 4.2.9 an integration of pattern recognition and image processing methods should be done for a more reliable definition of the error estimation function.

Misleading Cuts and Integration of User Actions

Anyhow, another even more severe problem arises in connection to the reconstruction of cross cut shredded text documents. Unfortunately it very often happens that horizontal cuts are placed such that they run between two lines of text. Then there are multiple shreds having blank top or bottom edges, respectively. Obviously, no useful information can be extracted from such cuts. Therefore, a robust reconstruction of cross cut shredded text documents becomes very unlikely. It is, however, possible to reliably reconstruct (large) parts of the original documents, whereas the final and complete reconstruction can then only be done exploiting the context of the contained text. Therefore, it is highly

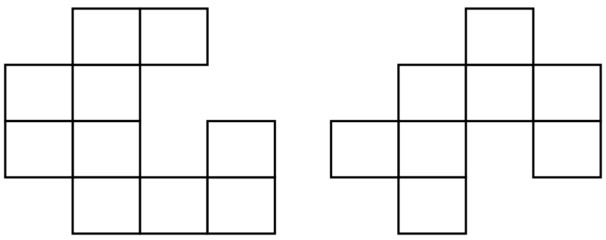


Figure 4.19.: Two arbitrarily shaped meta-shreds. A meaningful alignment of them is not straightforward.

meaningful to integrate user interaction as already proposed for the reconstruction of strip shredded text documents, cf. Sec. 4.2.7. Even more, such a human guided search approach can generally improve the performance of any reconstruction system.

Multilevel Refinement Strategies

Similar as proposed in the discussion for the reconstruction of strip shredded text documents, an extension of the above approaches to incorporate coarsening and refinement operations could be very promising. In this case, the building of larger blocks or meta-shreds might, however, be not as straightforward as for strips. Arbitrarily shaped meta-shreds introduce extreme complex situation when trying to combine them, see also Fig. 4.19. Therefore, it would be more convenient to demand that meta-shreds are rectangular. Then, however, the possibilities for combining shreds are rather restricted.

4.4. Impact on Confidentiality

Based on the results presented in the previous section, it can be concluded that whenever paper documents should be destroyed, it is extremely important that an appropriate method of destruction is chosen. Although the methods for reconstructing manually torn or mechanically shredded documents are not yet so far developed that fully automatic reconstruction can be achieved for larger documents, the approaches are advanced enough that even documents supposed to be destroyed, e.g., by cutting them into about 300 strips which—for a DIN A4 page—corresponds to strips of 0.7 mm width, could be reconstructed with the help of computer systems in an interactive way. Of course, the data acquisition process, i.e., the scanning of the strips, still needs to be improved, e.g., by developing methods for automatically scanning a large amount of shreds. It is therefore important that especially offices containing confident data think about the methods used for destructing (paper) documents.

In fact, there are methods for which it can be guaranteed that no reconstruction is possible. For example, paper documents could be burned or destroyed using chemical methods, e.g., some acid. However, it can be observed that e.g. newspapers burned in the fire place can still be read as long as the ashes is not stirred.

Chapter 5

Conclusions and Future Work

Within this work selected combinatorial optimization problems arising in different two domains were investigated: On the one hand a *storage location assignment problem* as well as a *tour planning problem* related to "classical" warehouses, i.e., warehouse consisting of aisles orthogonal to each other. On the other hand, we presented methods for reconstructing destructed paper documents.

The applications were formulated as combinatorial optimization problems and it was shown that they are $\mathcal{NP}$-hard and also very difficult to solve in practice. Various different solution approaches were developed including greedy heuristics, applications of variable neighborhood search (VNS) and variable neighborhood descent (VND), ant colony optimization (ACO), integer linear programming based techniques including a Lagrangian relaxation and a Lagrangian heuristic, dynamic programming methods and hybridizations of these techniques as well as with human guided search.

For the computation of concrete tours through the warehouse an exact algorithms with polynomially bounded running times was presented exploiting the specialized structure of the warehouse. This algorithm was then applied in a larger framework incorporating a variable neighborhood search approach for making decisions which articles to collect along the next tour. Within the embedded variable neighborhood descent (VND) an adaption of the self-adaptive neighborhood ordering presented in [65] was applied. It could be shown that this dynamic rearrangement of the neighborhoods could improve the overall solution process since the number of examinations of two neighborhood structures highly promising during the start of the search but loosing ground during the later iterations of VND was dramatically reduced. Obviously, this concept can be easily adopted for other applications where the characteristics of solutions significantly change depending on the region of the search space currently examined. It is further imaginable to apply similar methods to the selection of the step function to be used for the examination of neighborhoods. For example one could switch from random neighbor via first improvement to best improvement based on the observation that improvements will be very likely during the beginning of the search while the improvement rate is decreasing with longer running times. Therefore, random neighbor will get inefficient as well as next neighbor will in the worst case almost completely explore the neighbor.

Another, more static but still changing neighborhood ordering was applied during the computation of paper roll rearrangements for warehouses applying a Last-In, First-Out strategy to their storage locations. However, the more demanding challenge for this application area was the uncertainty of the production and shipping dates. Nevertheless, it could be shown during tests in a real-world environment, i.e., in the warehouse of a partner company, that the proposed approaches were robust enough to endure sudden worsenings in the objective function due to abrupt changes of the production sequence and/or late (or early) arriving customers. Even more, the warehouse states could be repaired in the sense that during (ad hoc) relocations optimal storage locations could be found.

It would, however, be highly interesting whether the results obtained could be further improved by solving both problems, the storage location assignment as well as the order picking tour, at the same time, i.e., possibly accepting slightly worsenings in the objective with respect to the storage location assignment but at the same time increasing the solution quality for the routes through the warehouse. Even more, it should be investigated whether further improvements could be achieved by using different storage location strategies, e.g., First-In, First-Out policy. Independently of this, it is, however, necessary to incorporate knowledge of the production process into the storage locations assignment.

For the application of combinatorial optimization techniques to the reconstruction of destroyed paper documents it can be summarized that, although only small sized instances with respect to the number of document pages could be solved, the results are nevertheless promising since the applied (and adapted) methods could exploit certain features of the problem instances. Especially the incorporation of user actions into the search procedure was not only effective but in fact necessary since only a human can decide whether (or not) a document was correctly reconstructed. Due to the fact a human user could "forbid" or "enforce" certain solutions features the search space could dramatically reduced, e.g., by setting the rotations of the strips. Obviously, this basic concept of human guided search can be extended to other areas where human intelligence is much more powerful then (current) computers, e.g., in packing irregularly shaped items into containers human knowledge and expertise should be exploited. Furthermore, it was shown that using a user integrated approach even combinatorial optimization problems with imperfect objective functions can be (semi-)automatically solved since only in the final phase of the optimization process the integration of human knowledge is necessary.

One interesting point is the fact that the computation of tours in “classical” warehouses forms a polynomially solvable case of the traveling salesman problem (TSP) while the reconstruction of strip shredded text documents corresponds to a generalization of TSP. So although the this two applications areas have not much in common on a first sight, it turns out that in fact one is a special case of the other.

Within this work two major hybridizations were examined: For the computation of routes through the warehouse an exact approach based on dynamic programming was integrated into a VNS/VND approach for computing sets of articles to be picked on the tours. It should, however, be mentioned that the objective function of the assignment of articles to tours is directly dependent on the actual tour lengths. This means that for an evaluation of an solution generated by the outer framework a set of subproblems needs to be solved. Thus the efficiency of the tour computation is of high relevance. The second hybridization addressed the incorporation of user actions into VNS based approach. As already mentioned above, on the one hand the search space could dramatically be reduced for some instances and on the other hand the final evaluation with respect to the quality of the solution could only be done by humans. Therefore, any reconstruction system disregarding input from users cannot be complete.

Finally, it should be mentioned that there is a lot of work to be done in the future for both application areas. Related to warehouse logistics further improvements could be achieved by regarding more information during the optimization, e.g., the production order of paper rolls. For the reconstruction of shredded text documents multilevel refinement strategies [136] should be developed for the reconstruction of shredded text documents. In addition, there is one major point which should be done in future works: Within this work, it was possible to show that combinatorial optimization techniques can be successfully applied to the reconstruction of shredded text documents. But in fact, not all problem specific information is currently exploited by the methods. Using pattern recognition and image processing methods more reliable and robust error estimation functions should be defined such that the number of instances for which the original document does not correspond to the solution having a minimum objective value (with respect to the error estimation function) is significantly reduced. However, in that domain the integration of user interactions into the search process is vital.

Bibliography

[1] A. A.-A. Abdel-Hamid and R. Borndörfer. On the complexity of storage assignment problems. Technical Report SC-94-14, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany, 1994.

[2] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1–3):75–102, 2002.

[3] S. Albers. Better bounds for online scheduling. *SICOMP: SIAM Journal on Computing*, 29(2):459–473, 1999.

[4] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1):3–26, 2003.

[5] T. Altman. Solving the jigsaw puzzle problem in linear time. *Applied Artificial Intelligence*, 3(4):453–462, 1989.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding tours in the TSP. Technical Report Number 99885, Research Institute for Discrete Mathematics, Universität Bonn, 1999.

[7] D. Applegate, W. Cook, and A. Rohe. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003.

[8] B. T. Ávila and R. D. Lins. A fast orientation and skew detection algorithm for monochromatic document images. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document Engineering*, pages 118–126, New York, NY, USA, 2005. ACM.

[9] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996.

[10] J. Balme. Reconstruction of shredded documents in the absence of shape information. Technical report, Yale University, USA, 2007.

[11] J. E. Beasley. Lagrangian relaxation. In C. R. Reeves, editor, *Modern heuristic techniques in combinatorial problems*, chapter 6, pages 243–303. John Wiley & Sons, Inc., 1993.

[12] A. Behzad and M. Modarres. A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. In *Proceedings of the 15th International Conference of Systems Engineering*, pages 6–8, 2002.

[13] R. E. Bellman. *Dynamic Programming.* Dover Publications Inc., 2003.

[14] F. Berger. Ein hybrides Verfahren zur automatischen Rekonstruktion von handzerrissenen Dokumentenseiten mittels geometrischer Informationen. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Austria, September 2008. supervised by G. Raidl and M. Prandtstetter.

[15] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[16] P. Bose, J.-D. Caron, and K. Ghoudi. Detection of text-line orientation. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG'98)*, 1998.

[17] H. Brynzér and M. I. Johansson. Storage location assignment: Using the product structure to reduce order picking times. *International Journal of Production Economics*, 46–47:595–603, 1996. Proceedings of the 8th International Working Seminar on Production Economics.

[18] H. Bunke and G. Kaufmann. Jigsaw puzzle solving using approximate string matching and best-first search. In D. Chetverikov and W. G. Kropatsch, editors, *Computer Analysis of Images and Patterns*, volume 719 of *LNCS*, pages 299–308. Springer, 1993.

[19] M. Caserta and S. Voß. A corridor method-based algorithm for the pre-marshalling problem. In M. Giacobini et al., editors, *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 788–797, Berlin, Heidelberg, 2009. Springer.

[20] M. Caserta, S. Voß, and M. Sniedovich. Applying the corridor method to a blocks relocation problem. *OR Spectrum*, 2006. doi:10.1007/s00291-009-0176-5.

[21] M. G. Chung, M. Fleck, and D. Forsyth. Jigsaw puzzle solver using shape and color. In *Fourth International Conference on Signal Processing Proceedings 1998, ICSP'98*, volume 2 of *Signal Processing Proceedings*, pages 877–880, October 1998.

[22] K. Connolly. "Puzzlers" reassemble shredded Stasi files, bit by bit. Los Angeles Times, November 1, 2009.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[24] D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price, editors. *New ideas in optimization*. McGraw-Hill Ltd., UK, 1999.

[25] J. De Bock, P. De Smet, W. Philips, and J. D'Haeyer. Constructing the topological solution of jigsaw puzzles. In *IEEE International Conference on Image Processing – ICIP '04*, volume 3, pages 2127–2130, 2004.

[26] R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 2007.

[27] P. De Smet. Reconstruction of ripped-up documents using fragment stack analysis procedures. *Forensic science international*, 176(2):124–136, April 2008.

[28] P. De Smet, J. De Bock, and E. Corluy. Computer vision techniques for semiautomatic reconstruction of ripped-up documents. In *AeroSense Conference Proceedings 2108B*, SPIE, Orlando, 2003. SPIE.

[29] P. De Smet, J. De Bock, and W. Philips. Semiautomatic reconstruction of strip-shredded documents. In A. Said and J. G. Apostolopoulos, editors, *Image and Video Communications and Processing 2005*, volume 5685 of *Proceedings of SPIE*, pages 239–248, San Jose, CA, USA, 2005. SPIE.

[30] Sichere Vernichtung von vertraulichen Unterlagen – Verfahrensregeln; Deutsche Fassung EN 15713:2009, August 2009.

[31] J. Dongarra and F. Sullivan. The top 10 algorithms. *Computing in Science and Engineering*, 2(1):33–50, 2000.

[32] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: An autocatalytic approach to the traveling salesman problem. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[33] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In Glover and Kochenberger [46], pages 251–285.

[34] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[35] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254, 1994.

[36] O. Eggenhofer. Optimales Lager-Layout: Kommissionierung, Material- und Verkehrsfluss im Fokus. *Getränkegrosshandel*, (5):30–34, 2007.

[37] Ö. Ergun and J. B. Orlin. A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. *Discrete Optimization*, 3(1):78–85, 2006.

[38] C. Feremans, M. Labbe, and G. Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1–13, 2003.

[39] M. Fischetti, J. J. S. González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

[40] P. W. Frizzell and J. W. Giffin. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research*, 22(6):655–667, 1995.

[41] L. M. Gambardella, È. D. Taillard, and G. Agazzi. Macs-vrptw: a multiple ant colony system for vehicle routing problems with time windows. In Corne et al. [24], pages 63–76.

[42] J. Gao, L. Sun, and M. Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):2892–2907, 2008.

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, 1979.

[44] M. Gendreau. An introduction to tabu search. In Glover and Kochenberger [46], pages 37–54.

[45] F. Glover. Future paths for integer programming and links to artificial intelligence. *Decision Sciences*, 8:156–166, 1977.

[46] F. W. Glover and G. A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science.* Kluwer Academic Publishers, New York, 2003.

[47] F. W. Glover, M. Laguna, and R. Marti. Scatter search and path relinking: Advances and applications. In Glover and Kochenberger [46], pages 1–35.

[48] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. *Computational Geometry*, 28(2–3):165–174, 2004.

[49] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, XLV(9):1563–1581, November 1966.

[50] J. Grefenstette. Proportional selection and sampling algorithms. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Evolutionary Computation 1: Basic Algorithms and Operators*, pages 172–180. Institute of Physics Publishing, Bristol and Philadelphia, 2000.

[51] M. Gruber. *Exact and Heuristic Approaches for Solving the Bounded Diameter Minimum Spanning Tree Problem.* PhD thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, May 2009. supervised by G. Raidl.

[52] M. Gruber, J. van Hemert, and G. R. Raidl. Neighborhood searches for the bounded diameter minimum spanning tree problem embedded in a VNS, EA, and ACO. In M. Keijzer et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO 2006*, volume 2, pages 1187–1194. ACM, 2006.

[53] N. Gupta and D. S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2-3):223–254, 1992.

[54] G. Gutin, D. Karapetyan, and N. Krasnogor. Memetic algorithm for the generalized asymmetric traveling salesman problem. In N. Krasnogor, G. Nicosia, M. Pavone, and D. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, volume 129 of *Studies in Computational Intelligence*, pages 199–210. Springer Berlin / Heidelberg, 2008.

[55] G. Gutin and A. Yeo. Assignment problem based algorithms are impractical for the generalized TSP. *Australasian Journal of Combinatorics*, 27:149–153, 2003.

[56] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[57] P. Hansen and N. Mladenović. A tutorial on variable neighborhood search. Technical Report G-2003-46, Les Cahiers du GERAD, HEC Montréal and GERAD, Canada, 2003.

[58] P. Hansen and N. Mladenović. Variable neighborhood search. In Glover and Kochenberger [46], pages 145–184.

[59] E. Hassini and R. G. Vickson. A two-carousel storage location problem. *Computers & Operations Research*, 30(4):527–539, 2003.

[60] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 2007. In Press. doi:10.1016/j.ejor.2007.08.048.

[61] D. Henderson, S. H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In Glover and Kochenberger [46], pages 287–319.

[62] H. H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2004.

[63] B. Hu, M. Leitner, and G. R. Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14(5):473–499, 2008.

[64] B. Hu and G. R. Raidl. Variable neighborhood descent with self-adaptive neighborhood-ordering. In C. Cotta et al., editors, *Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*, 2006.

[65] B. Hu and G. R. Raidl. Effective neighborhood structures for the generalized traveling salesman problem. In J. van Hemert and C. Cotta, editors, *Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2008*, volume 4972 of *LNCS*, pages 36–47, Naples, Italy, 2008. Springer.

[66] C. C. Jane. Storage location assignment in a distribution center. *International Journal of Physical Distribution & Logistics Management*, 30(1):55–71, 2000.

[67] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. Experimental analysis of heuristics for the atsp. In G. Gutin and A. Punnen, editors, *The TSP and Its Variations*, volume 12 of *Combinatorial Optimization*, pages 445–488. Kluwer Academic Publisher, May 2002.

[68] E. Justino, L. S. Oliveira, and C. Freitas. Reconstructing shredded documents through feature matching. *Forensic Science International*, 160(2–3):140–147, July 2006.

[69] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. In *SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–140, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[70] N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[71] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2005.

[72] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Series in Evolutionary Computation. Morgan Kaufmann, 2001.

[73] L. Khachiyan. A polynomial algorithm in linear programming (english translation). *Soviet Mathematics Doklady*, 20:191–194, 1979.

[74] K. H. Kim. Evaluation of the number of rehandles in container yards. *Computers & Industrial Engineering*, 32(4):701–711, 1997.

[75] K. H. Kim and G.-P. Hong. A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954, 2006.

[76] G. Klau, I. Ljubić, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G. Raidl, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. In K. Deb et al., editors, *Genetic and Evolutionary Computation – GECCO 2004*, volume 3102 of *LNCS*, pages 1304–1315. Springer, 2004.

[77] G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-guided tabu search. In *Proceedings of Eighteenth National Conference on Artificial Intelligence*, pages 41–47. American Association for Artificial Intelligence, 2002.

[78] G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-guided search. *Journal of Heuristics*, 2009. In press. doi:10.1007/s10732-009-9107-5.

[79] G. W. Klau, N. Lesh, J. Marks, M. Mitzenmacher, and G. T. Schafer. The HuGS platform: a toolkit for interactive optimization. In M. De Marsico, S. Levialdi, and E. Panizzi, editors, *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 324–330, New York, NY, USA, 2002. ACM.

[80] F. König and M. E. Lübbecke. Sorting with complete networks of stacks. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, *Algorithms and Computation – ISAAC 2008*, volume 5369 of *LNCS*, pages 895–906. Springer Berlin/Heidelberg, Dec. 2008.

[81] J. R. Koza. Genetic programming; automatic systhesis of topologies and numerical parameters. In Glover and Kochenberger [46], pages 83–104.

[82] R. Kumar and L. Haomin. On asymmetric TSP: Transformation to symmetric TSP and performance bound, 1994. Available online (lasted checked December 1, 2009) `http://www.ece.iastate.edu/~rkumar/PUBS/atsp.pdf`.

[83] P. D. Larson and A. Halldorsson. Logistics versus supply chain management: An international survey. *International Journal of Logistics Research and Applications*, 7(1):17–31, 2004.

[84] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Series in Discrete Mathematics & Optimization. John Wiley & Sons, Inc., 1985.

[85] Y. Lee and S.-L. Chao. A neighborhood search heuristic for pre-marshalling export containers. *European Journal of Operational Research*, 196(2):468–475, 2009.

[86] Y. Lee and N.-Y. Hsu. An optimization model for the container pre-marshalling problem. *Computers & Operations Research*, 34(11):3295–3313, 2007.

[87] P. Lehmann. Maultier. Historisches Lexikon der Schweiz (HLS), May 25, 2009. http://www.hls-dhs-dss.ch/textes/d/D26237.php.

[88] H. R. Lourenco, O. C. Martin, and T. Stützle. Iterated local search. In Glover and Kochenberger [46], pages 321–353.

[89] S. Lu and C. L. Tan. Automatic detection of document script and orientation. In *International Conference on Document Analysis and Recognition – ICDAR 2007*, volume 1, pages 237–241, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[90] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[91] S. Luo, C. Wang, and J. Wang. Ant colony optimization for resource-constrained project scheduling with generalized precedence relations. In *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 284. IEEE Computer Society, 2003.

[92] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346, 2002.

[93] T. Misar. Ein hybrides Verfahren basierend auf Variabler Nachbarschaftssuche und Dynamischer Programmierung zur Tourenfindung in einem Ersatzteillager mit domänenspezifischen Nebenbedingungen. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Austria, April 2009. supervised by G. Raidl and M. Prandtstetter.

[94] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

[95] W. Morandell. Evaluation and reconstruction of strip-shredded text documents. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Austria, May 2008. supervised by G. Raidl and M. Prandtstetter.

[96] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In Glover and Kochenberger [46], pages 105–144.

[97] G. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1999.

[98] A. Ostertag, K. F. Doerner, and R. F. Hartl. A variable neighborhood search integrated in the POPMUSIC framework for solving large scale vehicle routing problems. In M. J. Blesa et al., editors, *Hybrid Metaheuristics*, volume 5296 of *LNCS*, pages 29–42. Springer, 2008.

[99] M. W. Padberg and L. A. Wolsey. Trees and cuts. *Annals of Discrete Mathematics*, 17:511–517, 1983.

[100] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Prentice-Hall, 1982.

[101] S. Pirkwieser and G. R. Raidl. A column generation approach for the periodic vehicle routing problem with time windows. In M. G. Scutellà et al., editors, *Proceedings of the International Network Optimization Conference 2009*, Pisa, Italy, 2009.

[102] S. Pirkwieser, G. R. Raidl, and J. Puchinger. Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In C. Cotta and J. van Hemert, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2007*, volume 4446 of *LNCS*, pages 176–187. Springer, 2007.

[103] M. Prandtstetter. Exact and heuristic methods for solving the car sequencing problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, August 2005. supervised by G. Raidl and B. Hu.

[104] M. Prandtstetter and G. R. Raidl. A variable neighborhood search approach for solving the car sequencing problem. In *Proceedings of the XVIII Mini EURO Conference on VNS*, Tenerife, Spain, 2005.

[105] M. Prandtstetter and G. R. Raidl. Combining forces to reconstruct strip shredded text documents. In M. J. Blesa et al., editors, *Hybrid Metaheuristics*, volume 5296 of *LNCS*, pages 175–189. Springer, 2008.

[106] M. Prandtstetter and G. R. Raidl. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research*, 191(3):1004–1022, December 2008.

[107] M. Prandtstetter, G. R. Raidl, and T. Misar. A hybrid algorithm for computing tours in a spare parts warehouse. In C. Cotta and P. Cowling, editors, *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2009*, volume 5482 of *LNCS*, pages 25–36. Springer, 2009.

[108] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 3:1389–1401, 1957.

[109] J. Puchinger. *Combining Metaheuristics and Integer Programming for Solving Cutting and Packing Problems.* PhD thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria, January 2006. supervised by G. R. Raidl and U. Pferschy.

[110] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, Part II*, volume 3562 of *LNCS*, pages 41–53. Springer, 2005.

[111] J. Puchinger and G. R. Raidl. Relaxation guided variable neighborhood search. In *Proceedings of the XVIII Mini EURO Conference on VNS*, 2005.

[112] J. Puchinger and G. R. Raidl. Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. *Journal of Heuristics*, 14(5):457–472, 2008.

[113] G. R. Raidl and J. Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In C. Blum, M. J. B. Augilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics – An Emergent Approach for Combinatorial Optimization*, volume 114 of *Studies in Computational Intelligence*, pages 31–62. Springer, 2008.

[114] G. R. Raidl, J. Puchinger, and C. Blum. Metaheuristic hybrids. In M. Gendreau and J. Y. Potvin, editors, *Handbook of Metaheuristics*. Springer, 2nd edition, accepted 2009, to appear.

[115] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2–3):343–359, 2003.

[116] H. D. Ratliff and A. S. Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.

[117] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In Glover and Kochenberger [46], pages 219–249.

[118] U. Ritzinger. Generierung von Ein- und Umlagervorschlägen in Lagern mit einer Last-In First-Out Strategie und kundenspezifischen Auslagerpräferenzen. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Austria, December 2008. supervised by G. Raidl and M. Prandtstetter.

[119] U. Ritzinger, M. Prandtstetter, and G. R. Raidl. Computing optimized stock (re-)placements in Last-In, First-Out warehouses. In S. Voss et al., editors, *Logistik Management*, pages 279–298. Physica-Verlag, 2009.

[120] V. Roshanaei, B. Naderi, F. Jolai, and M. Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654–661, 2009.

[121] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. Elsevier, 2006.

[122] S. Sahni and T. Gonzales. P-Complete Approximation Problems. *Journal of the ACM*, 23(3):555–565, July 1976.

[123] P. Schüller. Reconstructing borders of manually torn paper scheets using integer linear programming. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Austria, January 2008. supervised by G. Raidl and M. Prandtstetter.

[124] J. Silberholz and B. Golden. The generalized traveling salesman problem: A new genetic algorithm approach. In E. K. Baker, A. Joseph, A. Mehrotra, and M. A. Trick, editors, *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, volume 37 of *Operations Research/Computer Science Interfaces*, pages 165–181. Springer US, April 2007.

[125] A. Skeoch. *An Investigation into Automated Shredded Document Reconstruction using Heuristic Search Algorithms*. PhD thesis, University of Bath, UK, 2006.

[126] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[127] D. Steenken, S. Voß, and R. Stahlbock. Container terminal operation and operations research — a classification and literature review. In H.-O. Günther and K. H. Kim, editors, *Container Terminals and Automated Transport Systems*. Springer, 2005.

[128] T. Stützle and M. Dorigo. Aco algorithms for the quadratic assignment problem. In Corne et al. [24], pages 33–50.

[129] T. Thron, G. Nagy, and N. Wassan. Evaluating alternative supply chain structures for perishable products. *The International Journal of Logistics Management*, 18(3):364–384, 2007.

[130] P. Toth and D. Vigo. *The Vehicle Routing Problem.* Number 9 in Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2002.

[131] A. Ukovich and G. Ramponi. Features for the reconstruction of shredded notebook paper. *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, 3:93–96, September 2005.

[132] A. Ukovich and G. Ramponi. System architecture for the digital recovery of shredded documents. volume 5672, pages 1–11. SPIE, 2005.

[133] A. Ukovich, G. Ramponi, H. Doulaverakis, Y. Kompatsiaris, and M. Strintzis. Shredded document reconstruction using MPEG-7 standard descriptors. *Signal Processing and Information Technology, 2004. Proceedings of the Fourth IEEE International Symposium on*, pages 334–337, December 2004.

[134] A. Ukovich, A. Zacchigna, G. Ramponi, and G. Schoier. Using clustering for document reconstruction. In E. R. Dougherty, J. T. Astola, K. O. Egiazarian, N. M. Nasrabadi, and S. A. Rizvi, editors, *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064 of *Proceedings of SPIE*. International Society for Optical Engineering, February 2006.

[135] G. Villa, S. Lozano, J. Racero, and D. Canca. A hybrid vns/tabu search algorithm for apportioning the european parliament. In J. Gottlieb and G. R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006*, volume 3906 of *LNCS*, pages 284–292. Springer, 2006.

[136] C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research*, 131:325–372, 2004.

[137] C. Walshaw. Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In C. Blum, M. J. B. Aguilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*, pages 261–289. Springer, 2008.

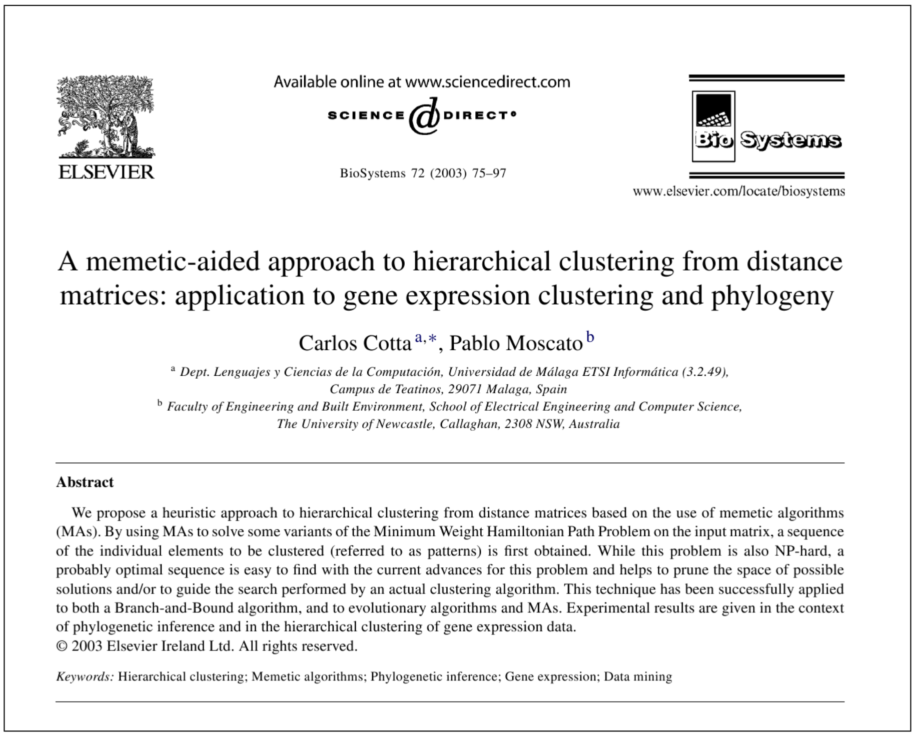
[138] B. D. Williams and T. Tokar. A review of inventory management research in major logistics journals: Themes and future directions. *The International Journal of Logistics Management*, 19(2):212–232, 2008.

[139] L. A. Wolsey. *Integer Programming.* Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1998.

[140] F.-H. Yao and G.-F. Shao. A shape and image merging technique to solve jigsaw puzzles. *Pattern Recognition Letters*, 24(12):1819–1835, 2003.

Appendix A

Pages Used for Generating Instances


Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

BioSystems 72 (2003) 75–97

BioSystems

www.elsevier.com/locate/biosystems

A memetic-aided approach to hierarchical clustering from distance matrices: application to gene expression clustering and phylogeny

Carlos Cotta[a,*], Pablo Moscato[b]

[a] Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga ETSI Informática (3.2.49), Campus de Teatinos, 29071 Malaga, Spain

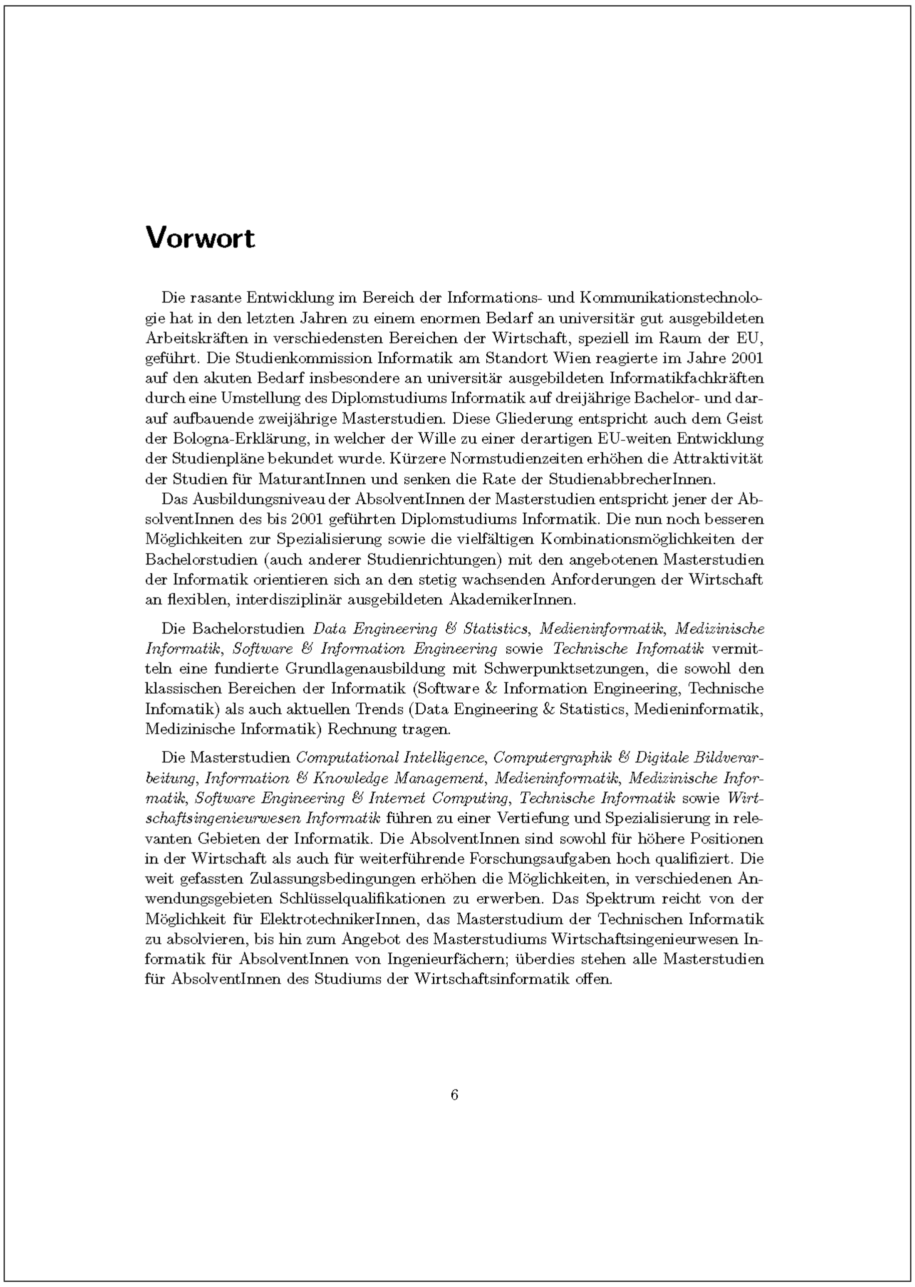
[b] Faculty of Engineering and Built Environment, School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan, 2308 NSW, Australia

Abstract

We propose a heuristic approach to hierarchical clustering from distance matrices based on the use of memetic algorithms (MAs). By using MAs to solve some variants of the Minimum Weight Hamiltonian Path Problem on the input matrix, a sequence of the individual elements to be clustered (referred to as patterns) is first obtained. While this problem is also NP-hard, a probably optimal sequence is easy to find with the current advances for this problem and helps to prune the space of possible solutions and/or to guide the search performed by an actual clustering algorithm. This technique has been successfully applied to both a Branch-and-Bound algorithm, and to evolutionary algorithms and MAs. Experimental results are given in the context of phylogenetic inference and in the hierarchical clustering of gene expression data.
© 2003 Elsevier Ireland Ltd. All rights reserved.

Keywords: Hierarchical clustering; Memetic algorithms; Phylogenetic inference; Gene expression; Data mining

Figure A.1.: Instance p06.



Vorwort

Die rasante Entwicklung im Bereich der Informations- und Kommunikationstechnologie hat in den letzten Jahren zu einem enormen Bedarf an universitär gut ausgebildeten Arbeitskräften in verschiedensten Bereichen der Wirtschaft, speziell im Raum der EU, geführt. Die Studienkommission Informatik am Standort Wien reagierte im Jahre 2001 auf den akuten Bedarf insbesondere an universitär ausgebildeten Informatikfachkräften durch eine Umstellung des Diplomstudiums Informatik auf dreijährige Bachelor- und darauf aufbauende zweijährige Masterstudien. Diese Gliederung entspricht auch dem Geist der Bologna-Erklärung, in welcher der Wille zu einer derartigen EU-weiten Entwicklung der Studienpläne bekundet wurde. Kürzere Normstudienzeiten erhöhen die Attraktivität der Studien für MaturantInnen und senken die Rate der StudienabbrecherInnen.

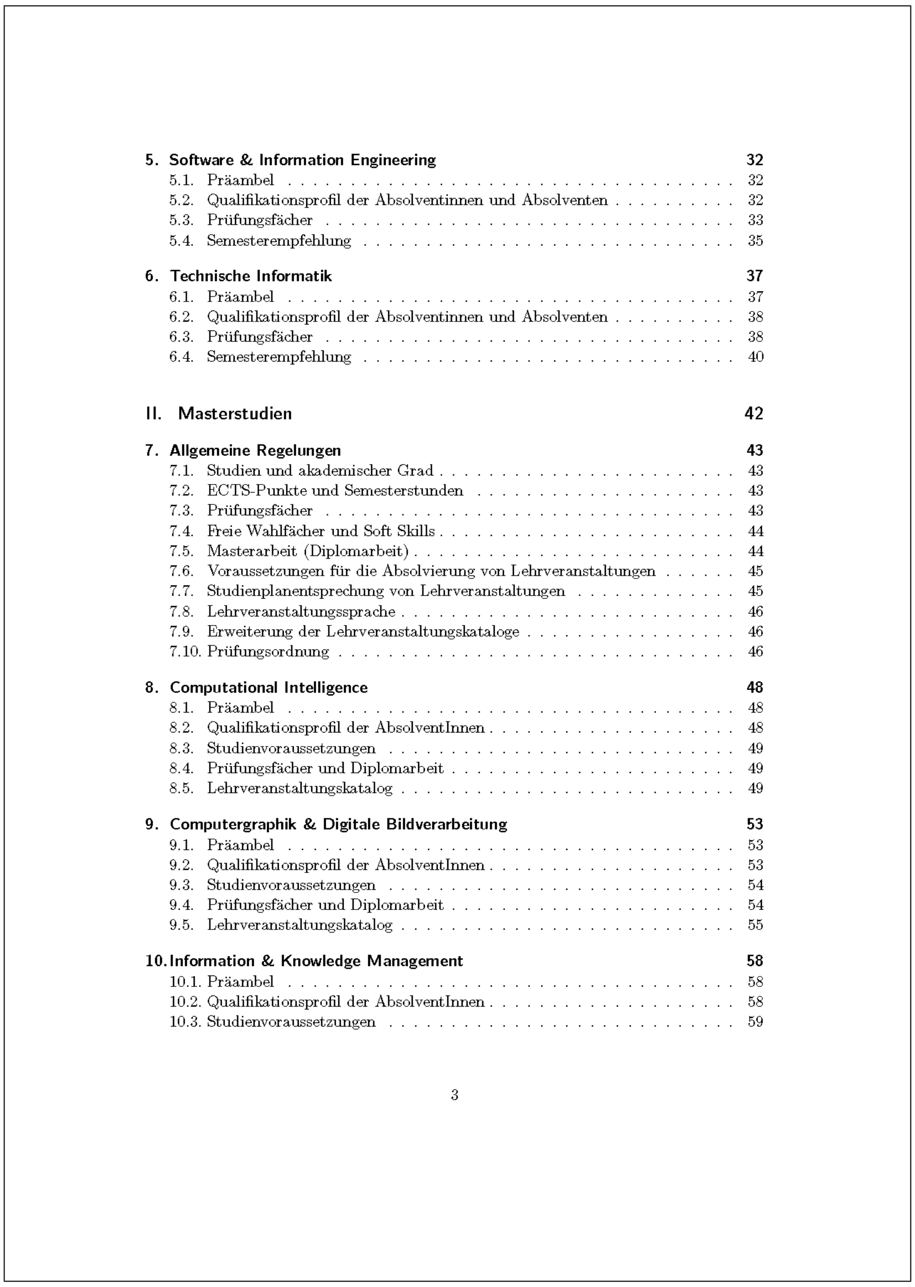
Das Ausbildungsniveau der AbsolventInnen der Masterstudien entspricht jener der AbsolventInnen des bis 2001 geführten Diplomstudiums Informatik. Die nun noch besseren Möglichkeiten zur Spezialisierung sowie die vielfältigen Kombinationsmöglichkeiten der Bachelorstudien (auch anderer Studienrichtungen) mit den angebotenen Masterstudien der Informatik orientieren sich an den stetig wachsenden Anforderungen der Wirtschaft an flexiblen, interdisziplinär ausgebildeten AkademikerInnen.

Die Bachelorstudien *Data Engineering & Statistics*, *Medieninformatik*, *Medizinische Informatik*, *Software & Information Engineering* sowie *Technische Infomatik* vermitteln eine fundierte Grundlagenausbildung mit Schwerpunktsetzungen, die sowohl den klassischen Bereichen der Informatik (Software & Information Engineering, Technische Infomatik) als auch aktuellen Trends (Data Engineering & Statistics, Medieninformatik, Medizinische Informatik) Rechnung tragen.

Die Masterstudien *Computational Intelligence*, *Computergraphik & Digitale Bildverarbeitung*, *Information & Knowledge Management*, *Medieninformatik*, *Medizinische Informatik*, *Software Engineering & Internet Computing*, *Technische Informatik* sowie *Wirtschaftsingenieurwesen Informatik* führen zu einer Vertiefung und Spezialisierung in relevanten Gebieten der Informatik. Die AbsolventInnen sind sowohl für höhere Positionen in der Wirtschaft als auch für weiterführende Forschungsaufgaben hoch qualifiziert. Die weit gefassten Zulassungsbedingungen erhöhen die Möglichkeiten, in verschiedenen Anwendungsgebieten Schlüsselqualifikationen zu erwerben. Das Spektrum reicht von der Möglichkeit für ElektrotechnikerInnen, das Masterstudium der Technischen Informatik zu absolvieren, bis hin zum Angebot des Masterstudiums Wirtschaftsingenieurwesen Informatik für AbsolventInnen von Ingenieurfächern; überdies stehen alle Masterstudien für AbsolventInnen des Studiums der Wirtschaftsinformatik offen.

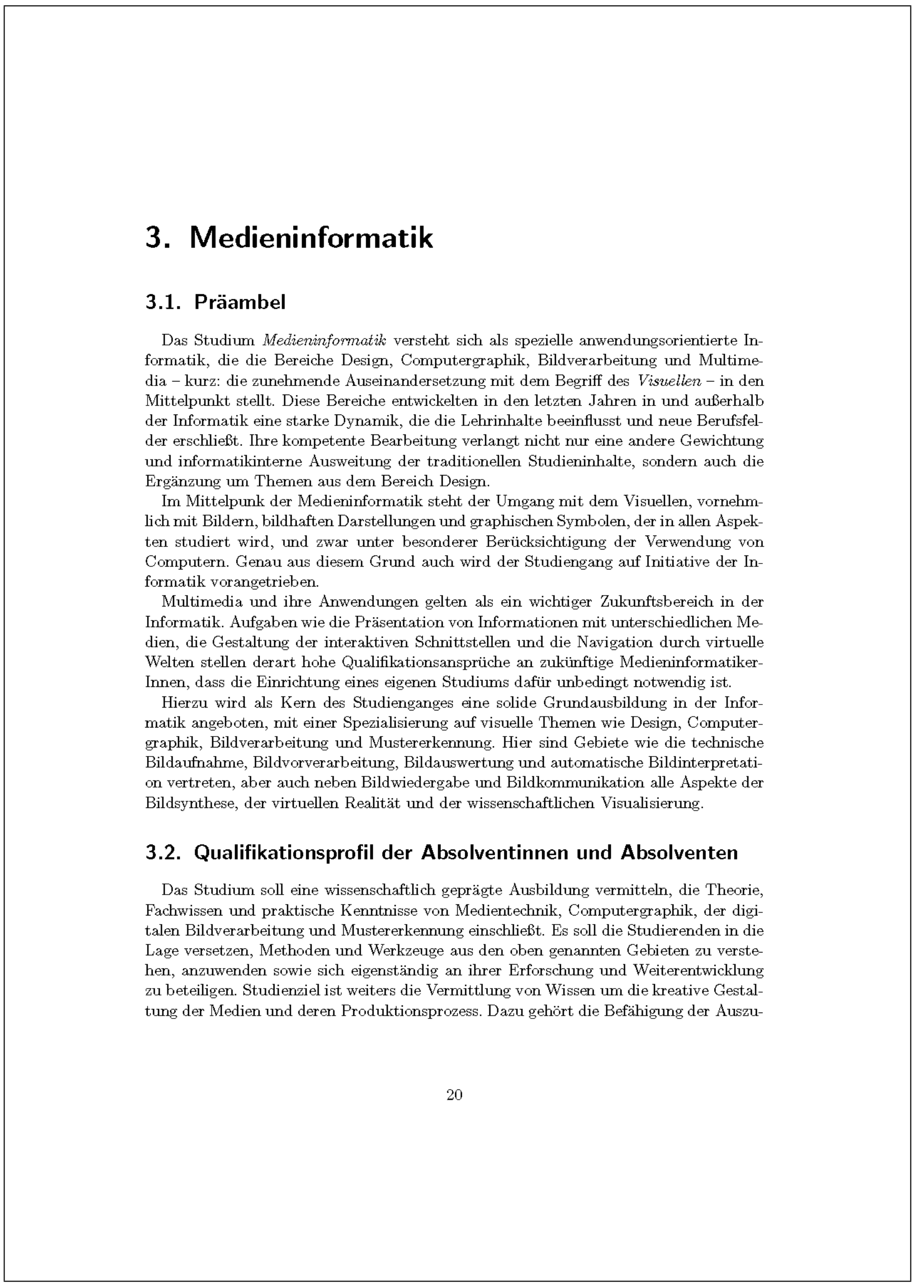
6

Figure A.2.: Instance p01.



3

Figure A.3.: Instance p02.



3. Medieninformatik

3.1. Präambel

Das Studium *Medieninformatik* versteht sich als spezielle anwendungsorientierte Informatik, die die Bereiche Design, Computergraphik, Bildverarbeitung und Multimedia – kurz: die zunehmende Auseinandersetzung mit dem Begriff des *Visuellen* – in den Mittelpunkt stellt. Diese Bereiche entwickelten in den letzten Jahren in und außerhalb der Informatik eine starke Dynamik, die die Lehrinhalte beeinflusst und neue Berufsfelder erschließt. Ihre kompetente Bearbeitung verlangt nicht nur eine andere Gewichtung und informatikinterne Ausweitung der traditionellen Studieninhalte, sondern auch die Ergänzung um Themen aus dem Bereich Design.

Im Mittelpunk der Medieninformatik steht der Umgang mit dem Visuellen, vornehmlich mit Bildern, bildhaften Darstellungen und graphischen Symbolen, der in allen Aspekten studiert wird, und zwar unter besonderer Berücksichtigung der Verwendung von Computern. Genau aus diesem Grund auch wird der Studiengang auf Initiative der Informatik vorangetrieben.

Multimedia und ihre Anwendungen gelten als ein wichtiger Zukunftsbereich in der Informatik. Aufgaben wie die Präsentation von Informationen mit unterschiedlichen Medien, die Gestaltung der interaktiven Schnittstellen und die Navigation durch virtuelle Welten stellen derart hohe Qualifikationsansprüche an zukünftige MedieninformatikerInnen, dass die Einrichtung eines eigenen Studiums dafür unbedingt notwendig ist.

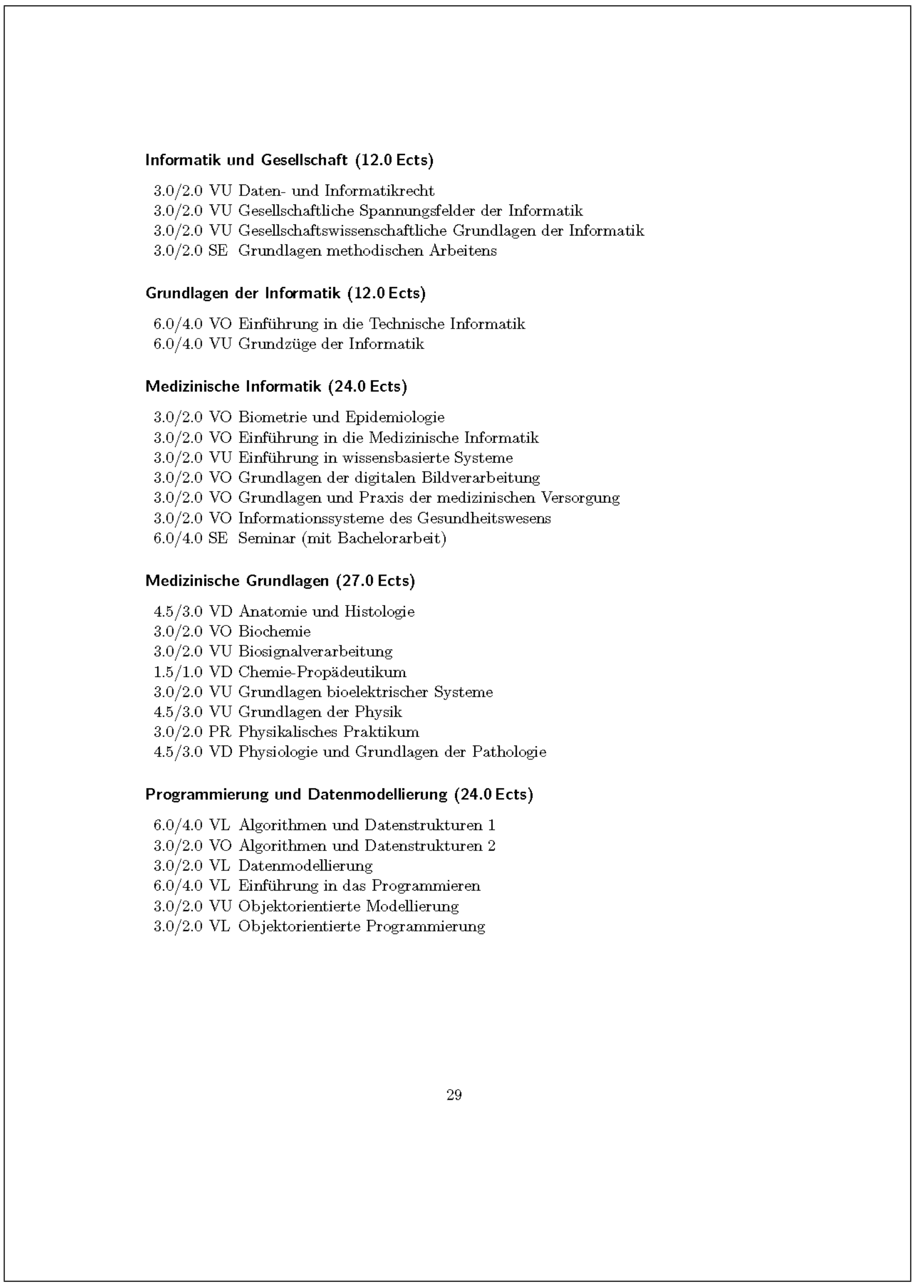
Hierzu wird als Kern des Studienganges eine solide Grundausbildung in der Informatik angeboten, mit einer Spezialisierung auf visuelle Themen wie Design, Computergraphik, Bildverarbeitung und Mustererkennung. Hier sind Gebiete wie die technische Bildaufnahme, Bildvorverarbeitung, Bildauswertung und automatische Bildinterpretation vertreten, aber auch neben Bildwiedergabe und Bildkommunikation alle Aspekte der Bildsynthese, der virtuellen Realität und der wissenschaftlichen Visualisierung.

3.2. Qualifikationsprofil der Absolventinnen und Absolventen

Das Studium soll eine wissenschaftlich geprägte Ausbildung vermitteln, die Theorie, Fachwissen und praktische Kenntnisse von Medientechnik, Computergraphik, der digitalen Bildverarbeitung und Mustererkennung einschließt. Es soll die Studierenden in die Lage versetzen, Methoden und Werkzeuge aus den oben genannten Gebieten zu verstehen, anzuwenden sowie sich eigenständig an ihrer Erforschung und Weiterentwicklung zu beteiligen. Studienziel ist weiters die Vermittlung von Wissen um die kreative Gestaltung der Medien und deren Produktionsprozess. Dazu gehört die Befähigung der Auszu-

20

Figure A.4.: Instance p03.



Informatik und Gesellschaft (12.0 Ects)

3.0/2.0	VU	Daten- und Informatikrecht
3.0/2.0	VU	Gesellschaftliche Spannungsfelder der Informatik
3.0/2.0	VU	Gesellschaftswissenschaftliche Grundlagen der Informatik
3.0/2.0	SE	Grundlagen methodischen Arbeitens

Grundlagen der Informatik (12.0 Ects)

6.0/4.0	VO	Einführung in die Technische Informatik
6.0/4.0	VU	Grundzüge der Informatik

Medizinische Informatik (24.0 Ects)

3.0/2.0	VO	Biometrie und Epidemiologie
3.0/2.0	VO	Einführung in die Medizinische Informatik
3.0/2.0	VU	Einführung in wissensbasierte Systeme
3.0/2.0	VO	Grundlagen der digitalen Bildverarbeitung
3.0/2.0	VO	Grundlagen und Praxis der medizinischen Versorgung
3.0/2.0	VO	Informationssysteme des Gesundheitswesens
6.0/4.0	SE	Seminar (mit Bachelorarbeit)

Medizinische Grundlagen (27.0 Ects)

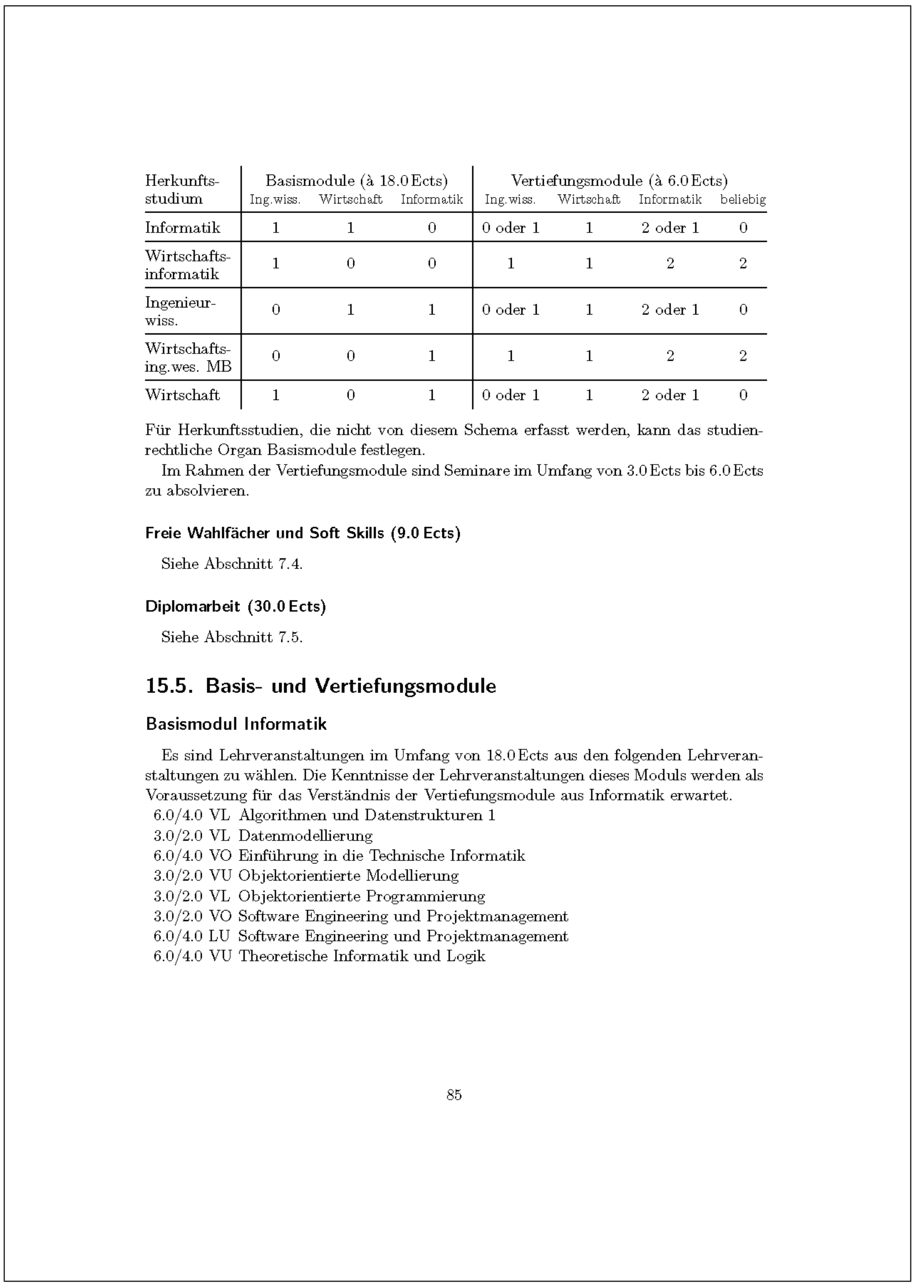
4.5/3.0	VD	Anatomie und Histologie
3.0/2.0	VO	Biochemie
3.0/2.0	VU	Biosignalverarbeitung
1.5/1.0	VD	Chemie-Propädeutikum
3.0/2.0	VU	Grundlagen bioelektrischer Systeme
4.5/3.0	VU	Grundlagen der Physik
3.0/2.0	PR	Physikalisches Praktikum
4.5/3.0	VD	Physiologie und Grundlagen der Pathologie

Programmierung und Datenmodellierung (24.0 Ects)

6.0/4.0	VL	Algorithmen und Datenstrukturen 1
3.0/2.0	VO	Algorithmen und Datenstrukturen 2
3.0/2.0	VL	Datenmodellierung
6.0/4.0	VL	Einführung in das Programmieren
3.0/2.0	VU	Objektorientierte Modellierung
3.0/2.0	VL	Objektorientierte Programmierung

29

Figure A.5.: Instance p04.



Herkunfts-studium	Basismodule (à 18.0 Ects)			Vertiefungsmodule (à 6.0 Ects)			
	Ing.wiss.	Wirtschaft	Informatik	Ing.wiss.	Wirtschaft	Informatik	beliebig
Informatik	1	1	0	0 oder 1	1	2 oder 1	0
Wirtschafts-informatik	1	0	0	1	1	2	2
Ingenieur-wiss.	0	1	1	0 oder 1	1	2 oder 1	0
Wirtschafts-ing.wes. MB	0	0	1	1	1	2	2
Wirtschaft	1	0	1	0 oder 1	1	2 oder 1	0

Für Herkunftsstudien, die nicht von diesem Schema erfasst werden, kann das studienrechtliche Organ Basismodule festlegen.

Im Rahmen der Vertiefungsmodule sind Seminare im Umfang von 3.0 Ects bis 6.0 Ects zu absolvieren.

Freie Wahlfächer und Soft Skills (9.0 Ects)

Siehe Abschnitt 7.4.

Diplomarbeit (30.0 Ects)

Siehe Abschnitt 7.5.

15.5. Basis- und Vertiefungsmodule

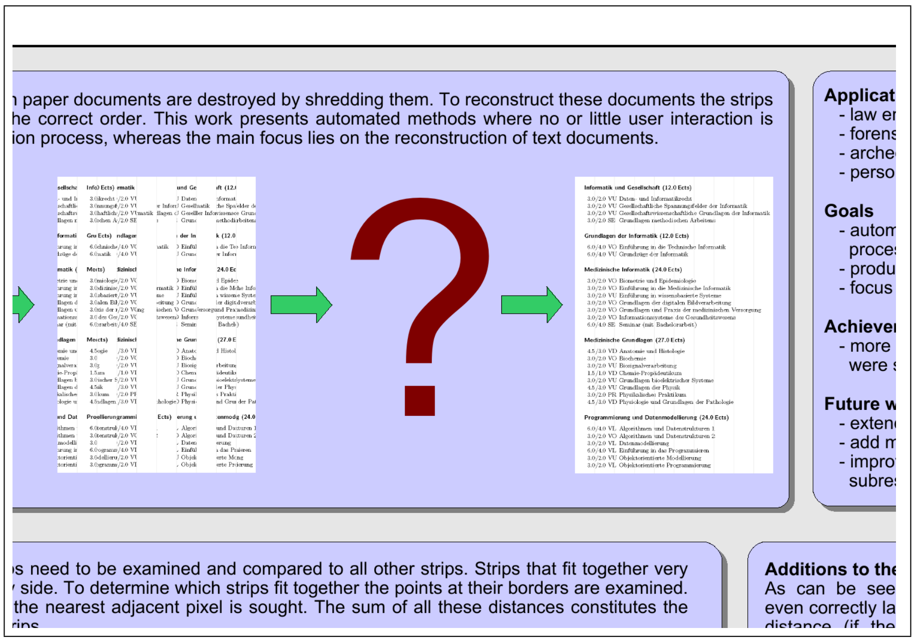
Basismodul Informatik

Es sind Lehrveranstaltungen im Umfang von 18.0 Ects aus den folgenden Lehrveranstaltungen zu wählen. Die Kenntnisse der Lehrveranstaltungen dieses Moduls werden als Voraussetzung für das Verständnis der Vertiefungsmodule aus Informatik erwartet.

6.0/4.0	VL	Algorithmen und Datenstrukturen 1
3.0/2.0	VL	Datenmodellierung
6.0/4.0	VO	Einführung in die Technische Informatik
3.0/2.0	VU	Objektorientierte Modellierung
3.0/2.0	VL	Objektorientierte Programmierung
3.0/2.0	VO	Software Engineering und Projektmanagement
6.0/4.0	LU	Software Engineering und Projektmanagement
6.0/4.0	VU	Theoretische Informatik und Logik

85

Figure A.6.: Instance p05.



ı paper documents are destroyed by shredding them. To reconstruct these documents the strips
he correct order. This work presents automated methods where no or little user interaction is
ion process, whereas the main focus lies on the reconstruction of text documents.

Informatik und Gesellschaft (12.0 Ects)
3.0/2.0 VU Daten- und Informatikrecht
3.0/2.0 VU Gesellschaftliche Spannungsfelder der Informatik
3.0/2.0 VU Gesellschaftswissenschaftliche Grundlagen der Informatik
3.0/2.0 SE Grundlagen methodischen Arbeitens

Grundlagen der Informatik (12.0 Ects)
6.0/4.0 VO Einführung in die Technische Informatik
6.0/4.0 VU Grundzüge der Informatik

Medizinische Informatik (24.0 Ects)
3.0/2.0 VO Biometrie und Epidemiologie
3.0/2.0 VO Einführung in die Medizinische Informatik
3.0/2.0 VU Einführung in wissensbasierte Systeme
3.0/2.0 VO Grundlagen der digitalen Bildverarbeitung
3.0/2.0 VO Grundlagen und Praxis der medizinischen Versorgung
3.0/2.0 VO Informationssysteme des Gesundheitswesens
6.0/4.0 SE Seminar (mit Bachelorarbeit)

Medizinische Grundlagen (27.0 Ects)
4.5/3.0 VD Anatomie und Histologie
3.0/2.0 VO Biochemie
3.0/2.0 VU Biosignalverarbeitung
1.5/1.0 VD Chemie-Propädeutikum
3.0/2.0 VU Grundlagen bioelektrischer Systeme
4.5/3.0 VU Grundlagen der Physik
3.0/2.0 PR Physikalisches Praktikum
4.5/3.0 VD Physiologie und Grundlagen der Pathologie

Programmierung und Datenmodellierung (24.0 Ects)
6.0/4.0 VL Algorithmen und Datenstrukturen 1
3.0/2.0 VO Algorithmen und Datenstrukturen 2
3.0/2.0 VL Datenmodellierung
6.0/4.0 VL Einführung in das Programmieren
3.0/2.0 VU Objektorientierte Modellierung
3.0/2.0 VL Objektorientierte Programmierung

Applicat
- law er
- forens
- arche
- perso

Goals
- autom
proce
- produ
- focus

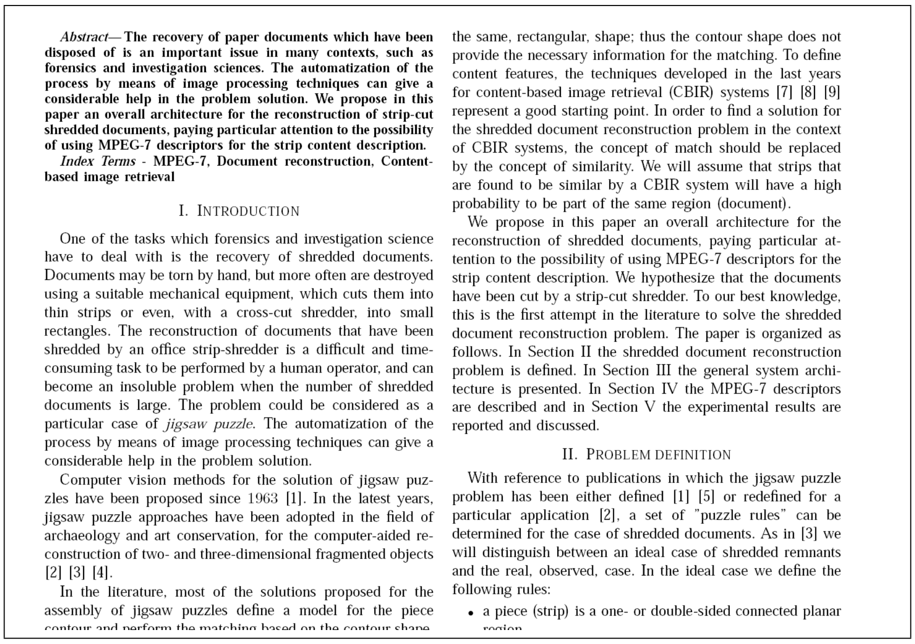
Achieve
- more
were

Future w
- exten
- add m
- impro
subre

ıs need to be examined and compared to all other strips. Strips that fit together very
ı side. To determine which strips fit together the points at their borders are examined.
the nearest adjacent pixel is sought. The sum of all these distances constitutes the

Additions to the
As can be see
even correctly la
distance (if the

Figure A.7.: Instance p07.



Abstract—**The recovery of paper documents which have been disposed of is an important issue in many contexts, such as forensics and investigation sciences. The automatization of the process by means of image processing techniques can give a considerable help in the problem solution. We propose in this paper an overall architecture for the reconstruction of strip-cut shredded documents, paying particular attention to the possibility of using MPEG-7 descriptors for the strip content description.**

Index Terms **- MPEG-7, Document reconstruction, Content-based image retrieval**

I. Introduction

One of the tasks which forensics and investigation science have to deal with is the recovery of shredded documents. Documents may be torn by hand, but more often are destroyed using a suitable mechanical equipment, which cuts them into thin strips or even, with a cross-cut shredder, into small rectangles. The reconstruction of documents that have been shredded by an office strip-shredder is a difficult and time-consuming task to be performed by a human operator, and can become an insoluble problem when the number of shredded documents is large. The problem could be considered as a particular case of *jigsaw puzzle*. The automatization of the process by means of image processing techniques can give a considerable help in the problem solution.

Computer vision methods for the solution of jigsaw puzzles have been proposed since 1963 [1]. In the latest years, jigsaw puzzle approaches have been adopted in the field of archaeology and art conservation, for the computer-aided reconstruction of two- and three-dimensional fragmented objects [2] [3] [4].

In the literature, most of the solutions proposed for the assembly of jigsaw puzzles define a model for the piece contour and perform the matching based on the contour shape

the same, rectangular, shape; thus the contour shape does not provide the necessary information for the matching. To define content features, the techniques developed in the last years for content-based image retrieval (CBIR) systems [7] [8] [9] represent a good starting point. In order to find a solution for the shredded document reconstruction problem in the context of CBIR systems, the concept of match should be replaced by the concept of similarity. We will assume that strips that are found to be similar by a CBIR system will have a high probability to be part of the same region (document).

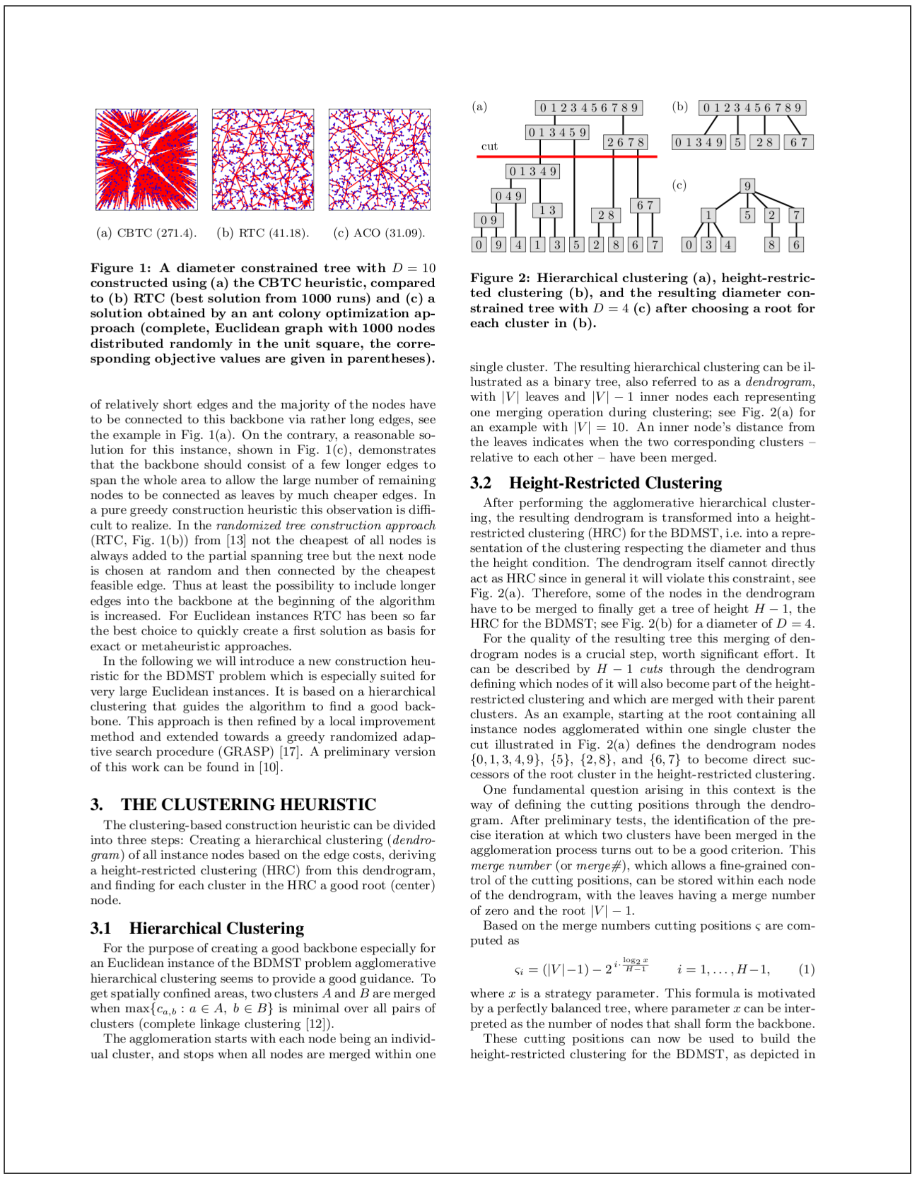
We propose in this paper an overall architecture for the reconstruction of shredded documents, paying particular attention to the possibility of using MPEG-7 descriptors for the strip content description. We hypothesize that the documents have been cut by a strip-cut shredder. To our best knowledge, this is the first attempt in the literature to solve the shredded document reconstruction problem. The paper is organized as follows. In Section II the shredded document reconstruction problem is defined. In Section III the general system architecture is presented. In Section IV the MPEG-7 descriptors are described and in Section V the experimental results are reported and discussed.

II. Problem Definition

With reference to publications in which the jigsaw puzzle problem has been either defined [1] [5] or redefined for a particular application [2], a set of "puzzle rules" can be determined for the case of shredded documents. As in [3] we will distinguish between an ideal case of shredded remnants and the real, observed, case. In the ideal case we define the following rules:

- a piece (strip) is a one- or double-sided connected planar

Figure A.8.: Instance p08.



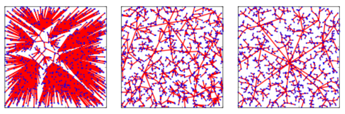

(a) CBTC (271.4). (b) RTC (41.18). (c) ACO (31.09).

Figure 1: A diameter constrained tree with $D = 10$ constructed using (a) the CBTC heuristic, compared to (b) RTC (best solution from 1000 runs) and (c) a solution obtained by an ant colony optimization approach (complete, Euclidean graph with 1000 nodes distributed randomly in the unit square, the corresponding objective values are given in parentheses).

of relatively short edges and the majority of the nodes have to be connected to this backbone via rather long edges, see the example in Fig. 1(a). On the contrary, a reasonable solution for this instance, shown in Fig. 1(c), demonstrates that the backbone should consist of a few longer edges to span the whole area to allow the large number of remaining nodes to be connected as leaves by much cheaper edges. In a pure greedy construction heuristic this observation is difficult to realize. In the *randomized tree construction approach* (RTC, Fig. 1(b)) from [13] not the cheapest of all nodes is always added to the partial spanning tree but the next node is chosen at random and then connected by the cheapest feasible edge. Thus at least the possibility to include longer edges into the backbone at the beginning of the algorithm is increased. For Euclidean instances RTC has been so far the best choice to quickly create a first solution as basis for exact or metaheuristic approaches.

In the following we will introduce a new construction heuristic for the BDMST problem which is especially suited for very large Euclidean instances. It is based on a hierarchical clustering that guides the algorithm to find a good backbone. This approach is then refined by a local improvement method and extended towards a greedy randomized adaptive search procedure (GRASP) [17]. A preliminary version of this work can be found in [10].

3. THE CLUSTERING HEURISTIC

The clustering-based construction heuristic can be divided into three steps: Creating a hierarchical clustering (*dendrogram*) of all instance nodes based on the edge costs, deriving a height-restricted clustering (HRC) from this dendrogram, and finding for each cluster in the HRC a good root (center) node.

3.1 Hierarchical Clustering

For the purpose of creating a good backbone especially for an Euclidean instance of the BDMST problem agglomerative hierarchical clustering seems to provide a good guidance. To get spatially confined areas, two clusters A and B are merged when $\max\{c_{a,b} : a \in A,\ b \in B\}$ is minimal over all pairs of clusters (complete linkage clustering [12]).

The agglomeration starts with each node being an individual cluster, and stops when all nodes are merged within one

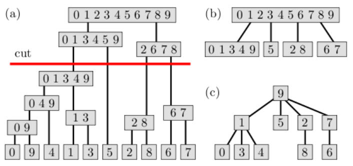


Figure 2: Hierarchical clustering (a), height-restricted clustering (b), and the resulting diameter constrained tree with $D = 4$ (c) after choosing a root for each cluster in (b).

single cluster. The resulting hierarchical clustering can be illustrated as a binary tree, also referred to as a *dendrogram*, with $|V|$ leaves and $|V| - 1$ inner nodes each representing one merging operation during clustering; see Fig. 2(a) for an example with $|V| = 10$. An inner node's distance from the leaves indicates when the two corresponding clusters – relative to each other – have been merged.

3.2 Height-Restricted Clustering

After performing the agglomerative hierarchical clustering, the resulting dendrogram is transformed into a height-restricted clustering (HRC) for the BDMST, i.e. into a representation of the clustering respecting the diameter and thus the height condition. The dendrogram itself cannot directly act as HRC since in general it will violate this constraint, see Fig. 2(a). Therefore, some of the nodes in the dendrogram have to be merged to finally get a tree of height $H - 1$, the HRC for the BDMST; see Fig. 2(b) for a diameter of $D = 4$.

For the quality of the resulting tree this merging of dendrogram nodes is a crucial step, worth significant effort. It can be described by $H - 1$ *cuts* through the dendrogram defining which nodes of it will also become part of the height-restricted clustering and which are merged with their parent clusters. As an example, starting at the root containing all instance nodes agglomerated within one single cluster the cut illustrated in Fig. 2(a) defines the dendrogram nodes $\{0, 1, 3, 4, 9\}$, $\{5\}$, $\{2, 8\}$, and $\{6, 7\}$ to become direct successors of the root cluster in the height-restricted clustering.

One fundamental question arising in this context is the way of defining the cutting positions through the dendrogram. After preliminary tests, the identification of the precise iteration at which two clusters have been merged in the agglomeration process turns out to be a good criterion. This *merge number* (or *merge#*), which allows a fine-grained control of the cutting positions, can be stored within each node of the dendrogram, with the leaves having a merge number of zero and the root $|V| - 1$.

Based on the merge numbers cutting positions ς are computed as

$$\varsigma_i = (|V|-1) - 2^{i \cdot \frac{\log_2 x}{H-1}} \qquad i = 1, \ldots, H-1, \qquad (1)$$

where x is a strategy parameter. This formula is motivated by a perfectly balanced tree, where parameter x can be interpreted as the number of nodes that shall form the backbone.

These cutting positions can now be used to build the height-restricted clustering for the BDMST, as depicted in

Figure A.9.: Instance p09.

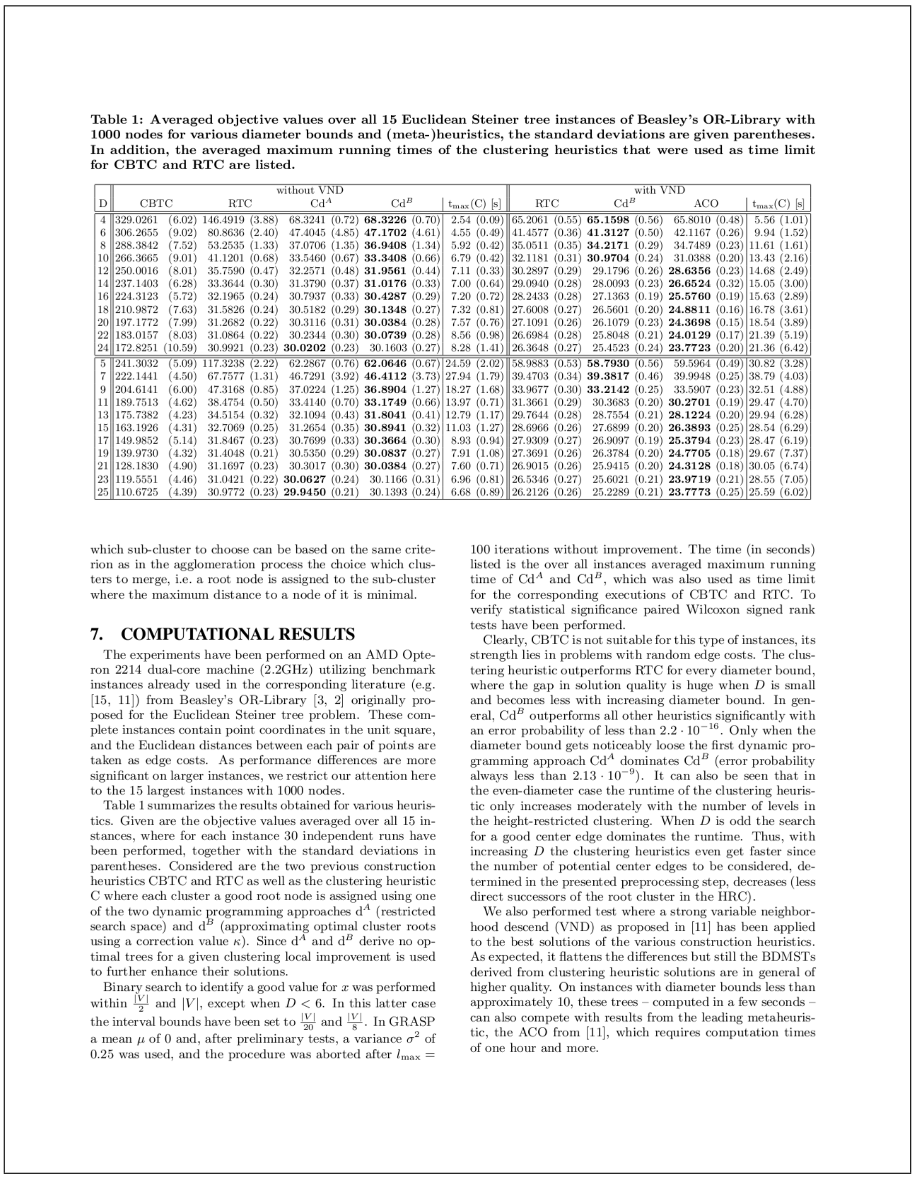


Table 1: Averaged objective values over all 15 Euclidean Steiner tree instances of Beasley's OR-Library with 1000 nodes for various diameter bounds and (meta-)heuristics, the standard deviations are given parentheses. In addition, the averaged maximum running times of the clustering heuristics that were used as time limit for CBTC and RTC are listed.

	without VND					with VND			
D	CBTC	RTC	Cd^A	Cd^B	t$_{\max}$(C) [s]	RTC	Cd^B	ACO	t$_{\max}$(C) [s]
4	329.0261 (6.02)	146.4919 (3.88)	68.3241 (0.72)	**68.3226** (0.70)	2.54 (0.09)	65.2061 (0.55)	**65.1598** (0.56)	65.8010 (0.48)	5.56 (1.01)
6	306.2655 (9.02)	80.8636 (2.40)	47.4045 (4.85)	**47.1702** (4.61)	4.55 (0.49)	41.4577 (0.36)	**41.3127** (0.50)	42.1167 (0.26)	9.94 (1.52)
8	288.3842 (7.52)	53.2535 (1.33)	37.0706 (1.35)	**36.9408** (1.34)	5.92 (0.42)	35.0511 (0.35)	**34.2171** (0.29)	34.7489 (0.23)	11.61 (1.61)
10	266.3665 (9.01)	41.1201 (0.68)	33.5460 (0.67)	**33.3408** (0.66)	6.79 (0.42)	32.1181 (0.31)	**30.9704** (0.24)	31.0388 (0.20)	13.43 (2.16)
12	250.0016 (8.01)	35.7590 (0.47)	32.2571 (0.48)	**31.9561** (0.44)	7.11 (0.33)	30.2897 (0.29)	29.1796 (0.26)	**28.6356** (0.23)	14.68 (2.49)
14	237.1403 (6.28)	33.3644 (0.30)	31.3790 (0.37)	**31.0176** (0.33)	7.00 (0.64)	29.0940 (0.28)	28.0093 (0.23)	**26.6524** (0.32)	15.05 (3.00)
16	224.3123 (5.72)	32.1965 (0.24)	30.7937 (0.33)	**30.4287** (0.29)	7.20 (0.72)	28.2433 (0.28)	27.1363 (0.19)	**25.5760** (0.19)	15.63 (2.89)
18	210.9872 (7.63)	31.5826 (0.24)	30.5182 (0.29)	**30.1348** (0.27)	7.32 (0.81)	27.6008 (0.27)	26.5601 (0.20)	**24.8811** (0.16)	16.78 (3.61)
20	197.1772 (7.99)	31.2682 (0.22)	30.3116 (0.31)	**30.0384** (0.28)	7.57 (0.76)	27.1091 (0.26)	26.1079 (0.23)	**24.3698** (0.15)	18.54 (3.89)
22	183.0157 (8.03)	31.0864 (0.22)	30.2344 (0.30)	**30.0739** (0.28)	8.56 (0.98)	26.6984 (0.28)	25.8048 (0.21)	**24.0129** (0.17)	21.39 (5.19)
24	172.8251 (10.59)	30.9921 (0.23)	**30.0202** (0.23)	30.1603 (0.27)	8.28 (1.41)	26.3648 (0.27)	25.4523 (0.24)	**23.7723** (0.20)	21.36 (6.42)
5	241.3032 (5.09)	117.3238 (2.22)	62.2867 (0.76)	**62.0646** (0.67)	24.59 (2.02)	58.9883 (0.53)	**58.7930** (0.56)	59.5964 (0.49)	30.82 (3.28)
7	222.1441 (4.50)	67.7577 (1.31)	46.7291 (3.92)	**46.4112** (3.73)	27.94 (1.79)	39.4703 (0.34)	**39.3817** (0.46)	39.9948 (0.25)	38.79 (4.03)
9	204.6141 (6.00)	47.3168 (0.85)	37.0224 (1.25)	**36.8904** (1.27)	18.27 (1.68)	33.9677 (0.30)	**33.2142** (0.25)	33.5907 (0.23)	32.51 (4.88)
11	189.7513 (4.62)	38.4754 (0.50)	33.4140 (0.70)	**33.1749** (0.66)	13.97 (0.71)	31.3661 (0.29)	30.3683 (0.20)	**30.2701** (0.19)	29.47 (4.70)
13	175.7382 (4.23)	34.5154 (0.32)	32.1094 (0.43)	**31.8041** (0.41)	12.79 (1.17)	29.7644 (0.28)	28.7554 (0.21)	**28.1224** (0.20)	29.94 (6.28)
15	163.1926 (4.31)	32.7069 (0.25)	31.2654 (0.35)	**30.8941** (0.32)	11.03 (1.27)	28.6966 (0.26)	27.6899 (0.20)	**26.3893** (0.25)	28.54 (6.29)
17	149.9852 (5.14)	31.8467 (0.23)	30.7699 (0.33)	**30.3664** (0.30)	8.93 (0.94)	27.9309 (0.27)	26.9097 (0.19)	**25.3794** (0.23)	28.47 (6.19)
19	139.9730 (4.32)	31.4048 (0.21)	30.5350 (0.29)	**30.0837** (0.27)	7.91 (1.08)	27.3691 (0.26)	26.3784 (0.20)	**24.7705** (0.18)	29.67 (7.37)
21	128.1830 (4.90)	31.1697 (0.23)	30.3017 (0.30)	**30.0384** (0.27)	7.60 (0.71)	26.9015 (0.26)	25.9415 (0.20)	**24.3128** (0.18)	30.05 (6.74)
23	119.5551 (4.46)	31.0421 (0.22)	**30.0627** (0.24)	30.1166 (0.31)	6.96 (0.81)	26.5346 (0.27)	25.6021 (0.21)	**23.9719** (0.21)	28.55 (7.05)
25	110.6725 (4.39)	30.9772 (0.23)	**29.9450** (0.21)	30.1393 (0.24)	6.68 (0.89)	26.2126 (0.26)	25.2289 (0.21)	**23.7773** (0.25)	25.59 (6.02)

which sub-cluster to choose can be based on the same criterion as in the agglomeration process the choice which clusters to merge, i.e. a root node is assigned to the sub-cluster where the maximum distance to a node of it is minimal.

7. COMPUTATIONAL RESULTS

The experiments have been performed on an AMD Opteron 2214 dual-core machine (2.2GHz) utilizing benchmark instances already used in the corresponding literature (e.g. [15, 11]) from Beasley's OR-Library [3, 2] originally proposed for the Euclidean Steiner tree problem. These complete instances contain point coordinates in the unit square, and the Euclidean distances between each pair of points are taken as edge costs. As performance differences are more significant on larger instances, we restrict our attention here to the 15 largest instances with 1000 nodes.

Table 1 summarizes the results obtained for various heuristics. Given are the objective values averaged over all 15 instances, where for each instance 30 independent runs have been performed, together with the standard deviations in parentheses. Considered are the two previous construction heuristics CBTC and RTC as well as the clustering heuristic C where each cluster a good root node is assigned using one of the two dynamic programming approaches d^A (restricted search space) and d^B (approximating optimal cluster roots using a correction value κ). Since d^A and d^B derive no optimal trees for a given clustering local improvement is used to further enhance their solutions.

Binary search to identify a good value for x was performed within $\frac{|V|}{2}$ and $|V|$, except when $D < 6$. In this latter case the interval bounds have been set to $\frac{|V|}{20}$ and $\frac{|V|}{8}$. In GRASP a mean μ of 0 and, after preliminary tests, a variance σ^2 of 0.25 was used, and the procedure was aborted after $l_{\max} =$ 100 iterations without improvement. The time (in seconds) listed is the over all instances averaged maximum running time of Cd^A and Cd^B, which was also used as time limit for the corresponding executions of CBTC and RTC. To verify statistical significance paired Wilcoxon signed rank tests have been performed.

Clearly, CBTC is not suitable for this type of instances, its strength lies in problems with random edge costs. The clustering heuristic outperforms RTC for every diameter bound, where the gap in solution quality is huge when D is small and becomes less with increasing diameter bound. In general, Cd^B outperforms all other heuristics significantly with an error probability of less than $2.2 \cdot 10^{-16}$. Only when the diameter bound gets noticeably loose the first dynamic programming approach Cd^A dominates Cd^B (error probability always less than $2.13 \cdot 10^{-9}$). It can also be seen that in the even-diameter case the runtime of the clustering heuristic only increases moderately with the number of levels in the height-restricted clustering. When D is odd the search for a good center edge dominates the runtime. Thus, with increasing D the clustering heuristics even get faster since the number of potential center edges to be considered, determined in the presented preprocessing step, decreases (less direct successors of the root cluster in the HRC).

We also performed test where a strong variable neighborhood descend (VND) as proposed in [11] has been applied to the best solutions of the various construction heuristics. As expected, it flattens the differences but still the BDMSTs derived from clustering heuristic solutions are in general of higher quality. On instances with diameter bounds less than approximately 10, these trees – computed in a few seconds – can also compete with results from the leading metaheuristic, the ACO from [11], which requires computation times of one hour and more.

Figure A.10.: Instance p10.

Appendix B

Curriculum Vitae

Personal Information

- Name: Matthias Prandtstetter
- Date of birth: September 18, 1980
- Place of birth: Vienna, Austria

Education

- since 10/2005:
  PhD student at the Vienna University of Technology, Austria Main research “Hybrid Optimization Methods for Warehouse Logistics and the Reconstruction of Destroyed Paper Documents”.
- 10/2006–03/2006:
  Computer Management studies (Informatikmanagement) at the Vienna University of Technology, Austria, with graduation to ”Mag.rer.soc.oec.” (equivalent to MSocEcSc).
- 10/2000–10/2005:
  Computer Science studies (Informatik) at the Vienna University of Technology, Austria, with graduation to ”Diplom Ingenieur” (equivalent to MSc). Diploma Thesis: ”Exact and heuristic methods for solving the Car Sequencing Problem”

- 09/1991–06/1999:
  Comprehensive school (with focus on languages) in Stockerau, Austria
- 09/1987–06/1991:
  Primary school in Korneuburg, Austria

Work Experience (Academic)

- since 03/2006:
  Research and teaching assistant, Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology
- 07/2005–01/2006:
  Employed in the FWF project *Combining Memetic Algorithms with Branch and Cut and Price for Some Network Design Problem* under grant P16263-N04, Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology
- 03/2002–06/2005:
  Teaching assistant (Tutor), Algorithms and Data Structures Group, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Publications

- Ulrike Ritzinger, Matthias Prandtstetter and Günther R. Raidl.
  Computing optimized stock (re-)placements in last-in, first-out warehouses. In Stefan Voss et al., editors, Logistik Management, pages 279–298. Physica-Verlag, 2009.
- Matthias Prandtstetter and Günther R. Raidl.
  Meta-heuristics for reconstructing cross cut shredded text documents. In Günther R. Raidl et al., editors, GECCO '09: Proceedings of the 11th annual conference on Genetic and evolutionary computation, pages 349–356. ACM Press, 2009.
- Matthias Prandtstetter, Günther R. Raidl and Thomas Misar.
  A hybrid algorithm for computing tours in a spare parts warehouse. In Carlos Cotta and Peter Cowling, editors, Evolutionary Computation in Combinatorial Optimization – EvoCOP 2009, volume 5482 of LNCS, pages 25–36. Springer, 2009.
- Matthias Prandtstetter.
  Two approaches for computing lower bounds on the reconstruction of strip shred-

ded text documents. Technical Report TR 186–1–09–01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2009.

- Matthias Prandtstetter and Günther R. Raidl.
  An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. European Journal of Operational Research, 191(3):1004–1022, 2008.
- Matthias Prandtstetter and Günther R. Raidl.
  Combining forces to reconstruct strip shredded text documents. In M. J. Blesa et al., editors, Hybrid Metaheuristics, volume 5296 of LNCS, pages 175–189. Springer, 2008.
- Matthias Prandtstetter and Günther R. Raidl.
  An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. Technical Report TR 186–1–05–01, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2005. Submitted to EJOR (European Journal of Operational Research).
- Matthias Prandtstetter and Günther R. Raidl.
  A variable neighborhood search approach for solving the car sequencing problem. In Proceedings of the XVIII Mini EURO Conference on VNS, Tenerife, Spain, 2005.
- Matthias Prandtstetter.
  Exact and heuristic methods for solving the car sequencing problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, August 2005. Supervised by G.R. Raidl and B. Hu.

Organizational and Reviewing Activities

- Program Commitee Member: 43^{rd} Hawaii International Conference on System Sciences
- since 02/2008:
  Reviewer for Computers & Operations Research, Elsevier.