TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

DIPLOMARBEIT

Ein hybrides Verfahren zur automatischen Rekonstruktion von handzerrissenen Dokumentenseiten mittels geometrischer Informationen

ausgeführt am

Institut für Computergraphik und Algorithmen
der Technischen Universität Wien

unter der Anleitung von

Univ.Prof. Dipl.-Ing. Dr. Günther Raidl
und
Univ.Ass. Mag. Dipl.-Ing. Matthias Prandtstetter

durch

Franz Berger
Märzstraße 76-78/1/15
A - 1150, Wien

September 2008

_____

Zusammenfassung

Diese Diplomarbeit ist dem großen Gebiet der Rekonstruktion von zerstörten Papierdokumenten zuzuordnen. Ziel dieser Arbeit ist die Zuweisung verschiedener durch manuelles Zerreißen erzeugter Schnipsel auf einzelne Seiten, sodass im Weiteren eine vollständige Rekonstruktion der originalen Dokumente vorgenommen werden kann. Alle in dieser Arbeit entwickelten Ansätze basieren auf der Idee als zusätzliche, sozusagen komplementäre Methoden zu Ansätzen aus dem Bereich der Bildverarbeitung zu fungieren.

In dieser Arbeit werden drei Ansätze genauer betrachtet: Eine *Lokale Suche* soll im Wesentlichen in möglichst kurzer Zeit mit Hilfe einfacher Strategien gute Lösungen liefern. Weiters wird basierend auf dieser lokalen Suche ein *Variable Neighborhood Descent* (VND) Ansatz definiert, bei dem nicht im klassischen Sinn Nachbarschaften sondern die Schrittfunktionen systematisch ausgetauscht werden. Letztlich wird noch eine *hybride Methode* vorgestellt, die eine lokale Suche mit einem exakten Verfahren basierend auf ganzzahliger linearer Programmierung kombiniert. Dabei wird versucht die Abweichung von „optimalen“ Seiten nach der Rekonstruktion anhand unterschiedlicher Merkmale zu messen und möglichst zu minimieren. Alle drei Ansätze werden durch die Anwendung einer für die Problemstellung entwickelten Datenstruktur effizient implementiert.

Anhand ausführlicher Testergebnisse wird die Qualität der in dieser Arbeit vorgestellten Methoden miteinander verglichen und im Weiteren detailliert diskutiert. Zusammenfasssend kann gesagt werden, dass die Kombination von VND zum Berechnen von Ausgangslösungen mit der hybriden Methoden zum Verfeinern dieser Startlösung angewandt auf Seiten mit jeweils zwei Rissen (entspricht vier Schnipseln) am erfolgversprechendsten ist. Erwartungsgemäß nimmt die Lösungsqualität mit zunehmender Schnipselzahl pro Seite und zunehmender Seitenanzahl ab.

Abstract

This thesis focuses on the field of the reconstruction of manually destroyed paper documents. The goal of this work is the assignment of different manually destroyed snippets to single pages in such a way, that afterwards a complete reconstruction of the original documents can be carried out. All approaches developed in this thesis are based on the idea that they act as a complementary approach on the field of image processing.

In this thesis three approaches are examined: A *Local Search* should essentially yield good solutions in the shortest possible time with the help of simple strategies. Furthermore, based on the local search, a *Variable Neighborhood Descent* (VND) approach is defined which does not interchange neighborhoods in the common sense. Instead a systematic exchange of different step functions is performed. Finally, a *hybrid method* is presented, which combines local search with an exact algorithm based on integer linear programming. Thereby it is attempted to measure the deviation from “optimal“ pages after the reconstruction by means of different features and minimize this deviation as much as possible. All three approaches are implemented efficiently by using a data structure which has been developed for the problem.

Based on detailed test results the quality of the presented methods are checked against each other and discussed in detail. To sum it up, it can be concluded that the combination of VND for the calculation of a start solution and the hybrid method to improve this solution applied on pages which are shredded twice (equates four snippets) are most promising. As expected, the quality of the solutions decreases with increasing number of snippets per page and increasing number of pages.

Inhaltsverzeichnis

1 Einleitung

Die Rekonstruktion von zerstörten Dokumenten ist von großer Bedeutung auf dem Gebiet der Forensik und investigativen Wissenschaft. Folgendes Beispiel zeigt die Relevanz: Im Herbst 1989 vernichtete die Stasi tausende von geheimen Dokumenten durch händisches Zerreißen in kleine Stücke. Das Resultat davon sind ungefähr 16000 Säcke voller Papierschnipsel. Da heutzutage die geheimen Informationen, die sich auf den Dokumenten befinden, von großem Interesse sind, möchte man die Dokumente wieder rekonstruieren.

Die Zerstörung von Dokumenten kann neben dem händischen Zerreißen auch durch den Gebrauch eines Schredders erfolgen. Wobei, je nach verwendeten Typ des Schredders, das Papier entweder in dünne Streifen oder sehr kleine Rechtecke zerschnitten wird. Diese Arbeit beschäftigt sich mit dem Problem von handzerrissenen Dokumenten, das in der Folge als *Tearing Paper Problem* (TPP) bezeichnet wird.

Abhängig von der Größe und Anzahl der Schnipsel kann die Wiederherstellung der originalen Dokumente sehr zeitaufwendig sein, womit diese für Menschen beinahe unmöglich ist. Aus diesem Grund werden automatische Rekonstruierungsmethoden benötigt. Für diese Ansätze müssen aber zuerst die beteiligten Schnipsel eingescannt werden. Mithilfe von Mustererkennung und Bildverarbeitungsprozessen wird dabei die Form der Schnipsel und weitere hilfreiche Merkmale wie Papierfarbe, Schriftfarbe, Schriftart, usw. ermittelt, die für die Rekonstruktion verwendet werden können. Es existieren auch Problemfälle, bei denen die Informationen, die sich auf den Schnipseln befinden, nicht ermittelt oder verwendet werden können. Das kann daran liegen, weil zum Beispiel die Muster auf den Schnipseln nicht zu erkennen sind oder alle Schnipsel ein beinahe identes Muster aufweisen.

Aufgrund dieser Tatsache wird hier ein Ansatz vorgestellt, der nur mithilfe der geometrischen Daten der Schnipsel arbeitet. Dieser soll die Mustererkennung nicht ersetzen sondern erweitern und in den Fällen, in denen die Mustererkennung zu nichts führt eine hilfreiche Alternative darstellen. Der hier vorgestellte Ansatz ordnet die einzelnen Schnipsel entsprechenden Seiten zu, ohne dabei eine exakte Anordnung der Schnipsel auf den jeweiligen Seiten zu ermitteln. Dazu wird eine abstrakte Datenstruktur vorgestellt, die es ermöglicht effizient nach bestimmten Schnipseln und deren Eigenschaften zu suchen. Die folgenden drei Methoden werden zur Lösung des TPP beschrieben: *Lokale Suche, Variable Neighborhood Descent* und eine *Hybridmethode*.

Aufbau der Diplomarbeit

Zunächst erfolgt eine detaillierte Beschreibung des Problems mit seinen speziellen Herausforderungen. In Kapitel 3 gebe ich einen Überblick über verwandte und bisherige Arbeiten auf diesem Themengebiet. Grundlegenden Methoden, die als Basis für die hier gewählten Ansätze dienen, werden im Kapitel 4 beschrieben. In Kapitel 5 werden die Lösungsansätze beschrieben. Dazu wird zunächst eine abstrakte Datenstruktur vorgestellt und anschließend drei Methoden für die Lösung des Problems präsentiert. Die von den verschiedenen Methoden gelieferten Lösungen werden im Kapitel 6 besprochen. Die Arbeit endet mit Schlussfolgerungen.

2 Problembeschreibung

2.1 Problem

Beim *Tearing Paper Problem* (TPP) wird von einer Menge von Schnipsel $\mathcal{S}$ (kleinen Papierstücken) ausgegangen, die von unterschiedlichen Seiten stammen. Erzeugt wurden die Schnipsel durch händisches Zerreißen von mehreren Papierseiten. Viele Verfahren verwenden zur Rekonstruktion der originalen Seiten Informationen, die sie von den Mustern auf den jeweiligen Schnipseln extrahiert haben (*Pattern Recognition*). Es gibt aber Problemfälle, bei denen dieser Ansatz nicht funktioniert, weil zum Beispiel die Muster auf den Schnipseln nicht zu erkennen sind oder Seiten zerrissen wurden, die alle ein beinahe identisches Muster besitzen, etc.

Aus diesem Grund wird hier eine hilfreiche Alternative zur Mustererkennung vorgestellt, die diese nicht ersetzen sondern erweitern soll. Der hier präsentierte Ansatz verwendet deshalb nur die geometrischen Daten der Schnipsel. Etwaige Muster auf den Schnipsel, Oberflächenbeschaffenheit des Papiers, etc. werden in diesem Ansatz nicht berücksichtigt. Ziel des TPP ist es, die gegebenen Schnipsel mithilfe ihrer geometrischen Daten einzelnen Seiten zuzuordnen. Wobei eine exakte Anordnung der Schnipsel auf den einzelnen Seiten, zur Rekonstruktion der originalen Seite, nicht Gegenstand dieser Arbeit ist.

Annahmen

Für das *Tearing Paper Problem* wurden einige Annahmen getroffen, die bestimmte Eigenschaften der Schnipsel festlegen. Mithilfe dieser Annahmen wird die Lösung der Problems vereinfacht, da aus ihnen abgeleitet werden kann wie viele Schnipsel sich von jedem Typ (siehe Abschnitt 2.2) auf einer Seite befinden müssen.

- Alle gegebenen Schnipsel sind durch händisches Zerreißen von Papierseiten der Größe DIN-A4 entstanden. Durch diese Annahme sind die Kantenlängen der originalen Seiten bekannt und somit auch deren Umfang und Fläche.
- Zusätzlich wird auch davon ausgegangen, dass alle Schnipsel vorhanden sind. Es gibt keine Seiten von denen nur Teile der Schnipsel existieren.
- Jede Seite besitzt genau vier Eckschnipsel. Durch diese Annahme kann sehr einfach die gesamte Anzahl der vorliegenden Seiten berechnet werden, indem man die Anzahl aller Eckschnipsel durch vier dividiert.

- Jedes Eckschnipsel besitzt genau zwei Außenkanten. Diese Annahme muss in der Praxis nicht zwingend der Fall sein, denn es kann auch vorkommen, dass eine Seite genau durch das Eck zerrissen wurde oder dass zwei Ecken Teil eines Schnipsels sind.
- Alle gegebenen Schnipsel, die keine Eckschnipsel sind, dürfen maximal eine Außenkante besitzen.
- Jede Seite wurde in 2^i Schnipsel zerrissen. Das bedeutet: die Anzahl der Schnipsel verdoppelt sich bei jedem Zerreißvorgang. Der Vorgang selbst wird im Abschnitt 3 beschrieben. Durch die vorherige Annahme, dass jede Seite genau vier Eckschnipsel besitzt, muss jede Seite mindestens zweimal zerrissen werden. Zusätzlich wird noch die Annahme getroffen, dass eine Seite durch maximal fünf Risse zerrissen wird und somit maximal 32 Schnipsel besitzt. Denn es ist physikalisch äußerst schwierig eine Seite, per Hand, sechsmal zu zerreißen. Dadurch ergeben sich für i die Werte: 2, 3, 4 und 5.

2.2 Instanz

Eine Instanz für das *Tearing Paper Problem* ist als eine Liste von n Schnipsel gegeben. Ein Schnipsel ist ein Polygon, deren Koordinaten in Millimeter angegeben sind. Für jede Kante eines Schnipsels ist zusätzlich angegeben ob es sich um eine Innenkante oder um eine Außenkante handelt. Wobei die Innenkanten durch den Prozess des Zerreißens entstanden sind und die Außenkanten ein Teil der originalen Seitenkanten sind.

In Abbildung 2.1 ist eine zerrissene Seite mit einigen ihrer Eigenschaften dargestellt. Zur Vereinfachung sind hier die Innenkanten als gerade Linien dargestellt. Die für die Beschreibung der Instanz verwendeten mathematischen Symbole werden im nächsten Absatz erklärt und sind noch einmal in der Tabelle 2.1 zusammengefasst.

Die Menge $\mathcal{S}$ aller Schnipsel wird nach der Anzahl ihrer Außenkanten unterschieden. Schnipsel, die keine Außenkante besitzen, werden als Menge $\mathcal{I}$ der *Innenschnipsel* bezeichnet. Schnipsel mit Außenkanten nennt man *Außenschnipsel* $\mathcal{O}$, wobei diese noch weiter unterschieden werden. Schnipsel mit genau einer Außenkante werden als *Randschnipsel* $\mathcal{B}$ und Schnipsel mit genau zwei Außenkanten als *Eckschnipsel* $\mathcal{C}$ bezeichnet. Wir gehen davon aus, dass es $m = \frac{|\mathcal{C}|}{4}$ originale Dokumentenseiten $\mathcal{P}$, mit $|\mathcal{P}| = m$, gibt. Die Länge der Außenkanten von den Außenschnipsel werden mit $\mathrm{L}_i^{\mathrm{S}}$ für alle $i \in \mathcal{O}$ bezeichnet. Jedes Außenschnipsel $i \in \mathcal{O}$ besitzt zwei Winkel α_i^1 und α_i^2 zu Außenkanten, wobei bei einem Eckschnipsel keiner dieser beiden Winkel in einem Eck beginnen oder enden darf. Mit $\mathrm{A}_i^{\mathrm{S}}$ für alle $i \in \mathcal{S}$ werden die Flächen der einzelnen Schnipsel bezeichnet.

Da bekannt ist, dass die originalen Seiten vor dem Zerreißen die Größe eines DIN-A4 Blattes hatten, stehen noch weitere Informationen zur Verfügung. Die längere Kante einer DIN-A4

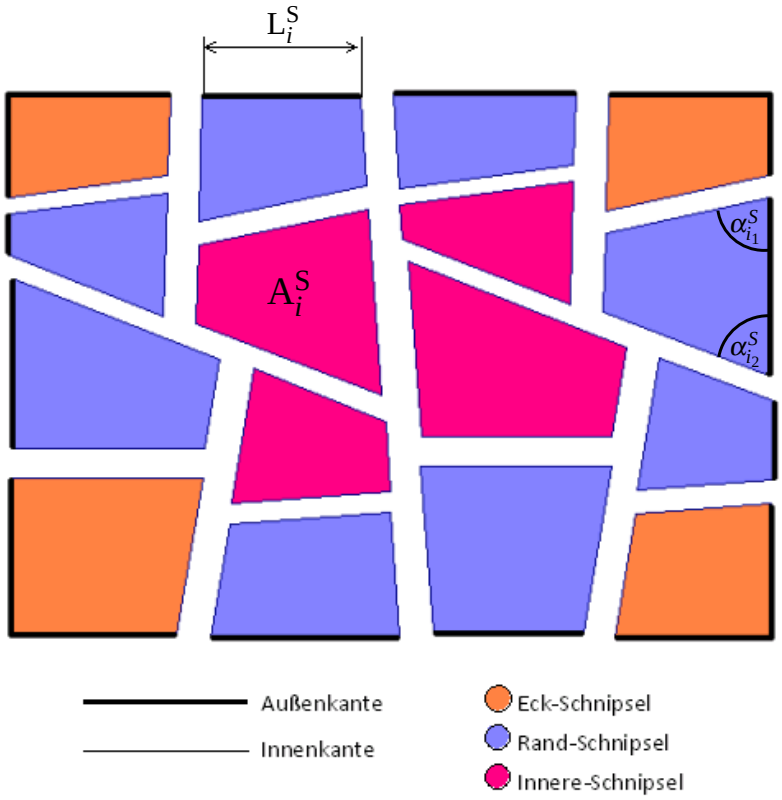


Abbildung 2.1: Darstellung einer zerrissenen Seite

Symbol	Erklärung
m	Anzahl der Seiten auf welche die Schnipsel zugeordnet werden sollen
$\mathcal{P}$	Menge aller Seiten auf welche die Schnipsel zugeordnet werden sollen
n	Anzahl der gegebenen Schnipsel
$\mathcal{S}$	Menge aller Schnipsel. $\mathcal{S} = \mathcal{I} \cup \mathcal{O}$
$\mathcal{I}$	Menge aller Innenschnipsel.
$\mathcal{O}$	Menge aller Außenschnipsel . $\mathcal{O} = \mathcal{B} \cup \mathcal{C}$
$\mathcal{B}$	Menge aller Randschnipsel
$\mathcal{C}$	Menge aller Eckschnipsel
L_i^S	Länge der Außenkante vom Schnipsel i
α_i^1	Erster Winkel zu einer Außenkante vom Schnipsel i
α_i^2	Zweiter Winkel zu einer Außenkante vom Schnipsel i
A_i^S	Fläche des Schnipsels i
L_l^P	Kantenlänge von der längeren (*long*) Seitenkante der originalen Seiten. $L_l^P = 295\ mm$
L_s^P	Kantenlänge von der kürzeren (*short*) Seitenkante der originalen Seiten. $L_s^P = 210\ mm$
P_{origin}^P	Umfang (*Perimenter*) der originalen Seiten. $P_{origin}^P = 1010mm$
A_{origin}^P	Fläche (*Area*) der originalen Seiten. $A_{origin}^P = 61950mm^2$

Tabelle 2.1: Zusammenfassung der verwendeten mathematischen Symbole

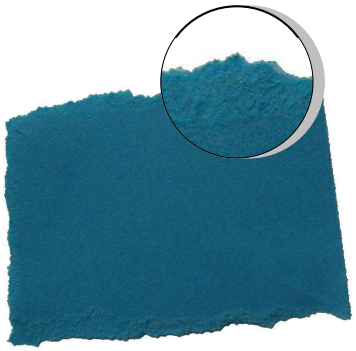


Abbildung 2.2: Schnipsel mit Schereffekt

Seite hat eine Länge von $L_l^P = 295\ mm$ und die kürzere eine von $L_s^P = 210\ mm$. Mithilfe dieser Kantenlängen erhält man auch noch den Umfang $P_{origin}^P = 1010\ mm$ und die Fläche $A_{origin}^P = 61950\ mm^2$ der originalen Seiten.

2.3 Herausforderungen

Eine Herausforderung die durch das händische Zerreißen von Papier entsteht ist der so genannte Schereffekt. Er führt dazu, dass die Breite der Risse größer als Null ist [1], siehe Abbildung 2.2.

Der Schereffekt entsteht, weil beim Zerreißen per Hand nicht jede Faser des Papiers an derselben Stelle getrennt wird, so wie es beim Zerschneiden mit einer Schere der Fall wäre. Deswegen bekommt man auch Risse die unregelmäßig, zackig und nicht geradlinig sind. Zusätzlich hat der Schereffekt auch noch die Eigenschaft, dass sich die Fläche der Schnipsel vergrößert. Für eine korrekte Wiederherstellung der Seiten müssen entlang der Risse viele Regionen überlappt werden [1]. Deshalb ist die Summe der Schnipselflächen einer Seite j deutlich größer als die originale Fläche A_{origin}^P der Seite.

$$\sum_{i \in P_j} A_i^S > A_{origin}^P \tag{2.1}$$

Dieselbe Eigenschaft ist auch beim Umfang zu bemerken, jedoch nicht in diesem Ausmaß da sich hier der Schereffekt nur an den Außenkanten der Schnipsel auswirkt und nicht wie bei der Fläche entlang aller Innenkanten.

$$\sum_{i \in P_j} L_i^S > P_{origin}^P \tag{2.2}$$

Die Summe der Kantenlängen L_i^S aller Schnipsel einer Seite j ist größer als der originale Umfang P_{origin}^P der Seite.

Eine weitere Herausforderung, die jedoch nicht Teil dieser Diplomarbeit ist, besteht darin, die gegebenen Schnipsel einzuscannen und in digitale Form darzustellen. Dazu müssen aus den Bilddaten Polygone erzeugt werden und jede Kante eines Schnipsel darauf hin überprüft werden, ob es sich um einen innere (ausgefranste) Kante oder um eine äußere (gerade) Kante handelt. Einen Grenzwert für die minimale Länge einer Kante liefert ein „Grobheitsmaß“. Je nach Wahl des Grenzwertes kann es vorkommen, dass sehr kurze Kanten nicht mehr als eigenständige Kanten erkannt werden, sondern Teil einer benachbarten Kante werden. Bei sehr kurzen Kanten kann es auch noch zu Schwierigkeiten bei der Zuordnung des Kantentyps (Innen- oder Außenkante) kommen [1].

In dieser Arbeit gehen wir davon aus, dass schon alle Schnipsel korrekt, in digitaler Form, vorhanden sind.

3 Verwandte Arbeiten

Es gibt verschiedene Problem die dem *Tearing Paper Problem* (TPP) ähnlich sind. In diesem Kapitel werden dazu einige verwandte Arbeiten vorgestellt, wobei eine von diesen Arbeiten etwas genauer betrachtet wird, da gewisse Teile von ihr auch in dieser Diplomarbeit verwendet wurden.

Eine verwandte Thematik, welche mit dem TPP verwandt ist, aber doch sehr unterschiedliche Herausfoderungen besitzt, beschäftigt sich mit dem automatischen Lösen von Puzzle (*Jigsaw Puzzles*). Der Hauptunterschied zum TPP liegt darin, dass beim Puzzle beinahe alle Teile eine nahezu einheitliche und klare Form besitzen. Aus diesem Grund können ihre geometrischen Daten sehr gut für die Rekonstruktion verwendet werden. Chung *et. al.* [2] beschreibt in ihrer Arbeit eine Methode, die versucht die verfügbaren Informationen effizienter zu nutzen. Dazu hat sie drei Methoden zur Rekonstruktion von Puzzle entwickelt, die neben den geometrischen Daten der Puzzle Teile auch ihre Farbinformationen verwendet. Denn im Gegensatz zu den meisten Textdokumenten können zum Lösen von Puzzle diese Informationen sehr gut ausgenutzt werden.

Ein anderes verwandtes Thema beschäftigt sich mit der Rekonstruktion von in Streifen geschnittenen Dokumenten. In diesem Fall wird von Streifen ausgegangen, die annähernd alle eine identische Form besitzen. Skeoch behandelt in [3] hauptsächlich die Rekonstruierung von geschredderten Bildern. Im Gegensatz zu Textdokumenten existieren in Bildern üblicherweise eine große Menge von unterschiedlichen Farben, sowie weiche Farbübergänge. Dieser Aspekt kann sehr effizient ausgenutzt werden. Die Hauptstrategie besteht darin die Pixel von den Rändern der Streifen mit benachbarten Pixel auf den Rändern von anderen Streifen, auf derselben Position, zu vergleichen. Ein großer Teil in [3] beschäftigt sich außerdem mit dem Scannvorgang und den ähnlichen Eigenschaften der Papierstreifen. Zusätzlich wird ein genetischer Algorithmus mit Crossover und Mutationsopterationen vorgestellt, sowie eine Heuristik für die Generierung von Anfangslösungen für geschredderte Bilder.

Ukovich *et al.* [4] beschreibt einen Ansatz, der davon ausgeht, dass die Rekonstruktion von geschredderten Dokumenten als eine spezielle Form des Puzzles gesehen werden kann. Wegen der einheitlichen Form der Streifen versucht Ukovich zusätzlich Informationen vom Inhalt, der sich auf den Streifen befindet zu verwenden. Für die Beschreibung dieser Informationen werden dabei die MPEG-7 Standard Deskriptoren verwendet. Ausgehen von diesen Informationen wird versucht ähnliche Streifen zu gruppieren (zum Beispiel, eine Menge für alle Farbstreifen, eine Menge für alle Streifen mit kursiver Schrift, ...). Für die Gruppierung wird im Allgemeinen die Farbe, Textur und die Form verwendet. Die Ergebnisse sind für Farbbilder vielversprechend, jedoch für Textdokumente liegen sie hinter den Erwartungen.

In [5] arbeitet Ukovich an der Ermittlung von zusätzlichen Eigenschaften, abgesehen von den bereits erwähnten MPEG-7 Deskriptoren. In dieser Arbeit wird von Papier aus einem Notizbuch ausgegangen, welches im Vergleich zu Büropapier leichte Unterschiede aufweist, zum Beispiel in der Größe und der Papierfarbe. Ukovich beschäftigt sich in dieser Arbeit ausschließlich mit handgeschriebenen Textdokumenten. Sie schlägt die Verwendung von Schrifterkennung vor. Da aber die Teile so klein sind, dass sich nicht einmal ein einzelnes Wort auf ihnen befindet, wird stattdessen das MPEG-7 Kanten Histogramm verwendet. Eine weitere wichtige Eigenschaft die hier erwähnt wird, ist die Erkennung von kariertem Papier. Ukovich verwendet für die Mustererkennung, die Hough Transformation. Diese Transformation ist ein universelles Werkzeug für die Gewinnung von Eigenschaften und liefert gute Ergebnisse.

Ukovich *et al.* beschreibt in [6] einen weiteren Ansatz, der versucht die geschredderten Streifen in Cluster einzuteilen. Vergleichbar mit dem manuellen Lösen eines Puzzles werden Teile mit einem ähnlichen Inhalt gruppiert. Das Clustering hat eine doppelte Auswirkung: einerseits wird die Komplexität der Teilprobleme erheblich verringert und andererseits wird die Qualität der Lösung erhöht, da Streifen nur mehr in ihren spezifischen Cluster gesucht werden. Ausgehend von der Annahme, dass die Einteilung korrekt ist. Ein wesentlicher Teil des Clustering Problems beschäftigt sich damit, die Anzahl der Cluster zu bestimmen auf welche die Streifen aufgeteilt werden sollen.

Morandell beschreibt in [7] einen Ansatz, der die Rekonstruierung von in Streifen geschnittenen Textdokumenten als ein kombinatorisches Optimierungsproblem, verwandt mit dem Handelsreisenden-Problem, formuliert. Außerdem werden grundlegende Ideen für die Lösung dieser neuen Formulierung vorgestellt. Dazu gehören Metaheuristiken wie Variable Nachbarschaftssuche, Iterierte Lokale Suche und Simulated Annealing.

Prandtstetter beschreibt in [8], basierend auf den Ansatz von Morandell [7], eine Transformation des Problems, der Rekonstruktion von geschredderten Textdokumenten, in das symmetrische Handelsreisenden-Problem. Es wird die verkettete Lin Kernigham Heuristik sowie eine neu eingeführte Variable Nachbarschaftssuche angewandt. Da die beiden Ansätze von ihrer Zielfunktion und somit von Schätzwerten und Wahrscheinlichkeiten bezüglich der Anordnung der Streifen abhängig sind, wurden die Algorithmen in den HuGS-Framework eingebettet. Dies ermöglicht den Benutzer in den Optimierungsvorgang einzugreifen. Die Ergebnisse zeigen, dass dieser halbautomatische Weg sehr gut Lösungen liefert.

Abschließend wollen wir uns noch Arbeiten ansehen, die sich mit der Rekonstruktion von handzerrissenen Dokumenten beschäftigt. Der Hauptunterschied, zum Problem mit geschredderten Dokumenten liegt im Schereffet, der die Lösung des Problems erheblich komplizierter macht.

Justino *et al.* [9] beschreibt eine Methode zur Rekonstruktion von handzerrissen Dokumenten. Die vorgestellte Methode wird in drei Hautschritte unterteilt. Um das Problem mit den unregelmäßigen Rissen zu bewältigen, werden zu Beginn der Methode die einzelnen Schnipsel

mithilfe eines Algorithmus, der eine Polylinienvereinfachung durchführt, vorbehandelt. Dieser Algorithmus verwendet dazu die Nähe eines Knoten zu einem Kantensegment, um die Komplexität der Ränder zu reduzieren. Nach der Vorbehandlung werden verschiedene Eigenschaften von den Schnipseln extrahiert. Diese Extrahierung stellt auch eine Reduzierung der Komplexität dar, da es die Polygone in eine Sequenz von Eigenschaften umwandelt. Zu den extrahierten Eigenschaften gehören: der Winkel von jedem Knoten bezüglich seiner beiden Nachbarn, die Überprüfung ob der jeweilige Winkel konvex oder konkav ist und der euklidische Abstand zwischen ein Knoten und seinen Nachbarn. Mithilfe der ermittelten Eigenschaften wird anschließend ein *Matching* durchgeführt. Dabei wird ein Schnipsel mit allen anderen verglichen und die beiden, die am besten zusammenpassen zu einem neuen Schnipsel zusammengefügt. Wiederholt wird dieser Vorgang solange, bis entweder nur mehr ein Schnipsel vorhanden ist oder keine Schnipsel mehr verschmelzt werden können. Die Ergebnisse zeigen, dass die Methode limitiert ist für eine kleine Anzahl von Schnipsel, da mit einer steigender Schnipselzahl die Lösungen deutlich schlechter werden.

De Smet [10] versucht die Informationen der Schnipsel über ihre relative Stackposition auszunutzen. Dadurch muss bei der Bergung der Schnipsel aufgepasst werden, dass die relative Stackposition von jedem Schnipsel bezüglich aller anderen verfügbar ist. Es wird in [10] davon ausgegangen, dass alle Entscheidungen die über Paarungen von Schnipsel getroffen wurden korrekt sind. Bei der Schnipselerzeugung wird ein Modell verwendet, bei dem ein Papier immer in der Mitte zerrissen wird und anschließend die beiden Hälften übereinander gelegt werden. Das Ergebnis nach dem Zerreißen wird als Folge von Zeichen dargestellt. Jeder dieser Folgen stellt die Entwicklungsinformation, zu welcher Hälfte das Schnipsel vor dem Zerreißen gehört hat, dar. Bei der Rekonstruktion wird versucht den Zerreißvorgang umzukehren und durch die schrittweise Paarung von Schnipsel die Zerreißfolge zu reduzieren. De Smet stellt dafür zwei unterschiedliche Methoden vor, die für Szenarien begrenzt sind bei denen alle Schnipsel vorhanden sind und die Stackanordnung ideal ist. Weiters wurden auch Lösungsansätze für nicht ideale Situation vorgestellt, wenn zum Beispiel ein Block von Schnipsel im Stack nicht vorhanden ist. Es sind jedoch keine detaillierten Informationen gegeben, wie der Lösungsprozess auf nicht idealisierte Situationen angepasst werden kann.

Die von Schüller [1] vorgestellten Verfahren verwenden *Integer Linear Programming* (ILP), um die Ränder von handzerrissen Papierseiten wieder herzustellen. Es werden zwei ILP Formulierungen zur Lösung des Problems beschrieben die beide nur die Außenschnipsel zur Rekonstruktion verwenden. Die erste Formulierung wird LILP genannt und verwendet die Längen von den Außenkanten der Außenschnipsel. Die zweite Formulierung wird LAILP genannt und verwendet zusätzlich die Winkel zwischen zwei benachbarten Schnipseln. Die LILP- und LAILP-Formulierung wurden mit Hilfe des CPLEX Solvers von ILOG, Inc. Version 10.0 realisiert. Anschließend erfolgt die Beschreibung der LAILP Formulierung die auch in dieser Diplomarbeit verwendet wurde.

LILP Formulierung

Diese Formulierung ordnet die Außenkanten der Schnipsel den Seitenkanten zu, sodass die Summen der zugeordneten Außenkantenlängen so gut wie möglich den Längen der originalen Seitenkanten entsprechen. Innenschnipsel werden bei dieser Formulierung nicht berücksichtigt.

Die folgenden binären Variablen stellen die Zuordnung der Schnipselkanten zu den Seitenkanten dar [1]:

$$x_{i,w} = \begin{cases} 1 & \text{wenn die Schnipselkante } i \text{ zur Seitenkante } w \text{ gehört} \\ 0 & \text{sonst} \end{cases} \tag{3.1}$$
$$\forall i \in \mathcal{O}, w \in K$$

Bedingungen

Für die LILP sind folgenden Bedingungen festgelegt:

- Jede Schnipselkante ist Teil von genau einer Seitenkante:

$$\sum_{w \in K} x_{i,w} = 1 \qquad \forall i \in \mathcal{O} \tag{3.2}$$

- Jeder Seitenkante sind mindestens zwei Schnipselkanten zugeordnet:

$$\sum_{i \in \mathcal{O}} x_{i,w} \geq 2 \qquad \forall w \in K \tag{3.3}$$

- Eckschnipsel verbinden immer lange und kurze Seitenkanten, wie in den folgenden Bedingungen gezeigt wird:

  Von den zwei Kanten eines gegebenen Eckschnipsels wird immer genau eine Kante der langen Seitenkante zugeordnet [2]:

$$\sum_{w \in K^{\mathrm{L}}} (x_{C_v^1,w} + x_{C_v^2,w}) = 1 \qquad \forall v \in \mathcal{C} \tag{3.4}$$

  Von den zwei Kanten eines gegebenen Eckschnipsels wird immer genau eine Kante der kurzen Seitenkante zugeordnet[3]:

---

[1] K : Menge aller Seitenkanten.

[2] C^1: Menge aller ersten Kanten der Eckschnipsel.
C^2: Menge aller zweiten Kanten der Eckschnipsel.
K^{L}: Menge aller langen (long) Seitenkanten

[3] K^{S}: Menge alle kurzen (short) Seitenkanten

$$\sum_{w \in K^{\mathrm{S}}} (x_{C_v^1,w} + x_{C_v^2,w}) = 1 \qquad \forall v \in \mathcal{C} \tag{3.5}$$

- Wenn eine Kante eines Eckschnipsels Teil einer Seitenkante ist, dann ist die andere Kante des Eckschnipsels ein Teil der nachfolgenden Seitenkante vom selben Blatt.

$$\begin{aligned} x_{C_v^1,w} &= x_{C_v^2,w_{next}} \\ w_{next} &= w - (w \bmod 4) + ((w+1) \bmod 4) \end{aligned} \qquad \forall w \in K \tag{3.6}$$

- Jede Seitenkante muss genau eine erste Kante eines Eckschnipsels und eine zweite Kante eines Eckschnipsels beinhalten:

$$\sum_{i \in C^1} x_{i,w} = 1 \qquad \forall w \in K \tag{3.7}$$

$$\sum_{i \in C^2} x_{i,w} = 1 \qquad \forall w \in K \tag{3.8}$$

Zielfunktion

Alle ganzzahligen Lösungen definiert durch die oben beschriebenen Bedingungen sind gültige Lösungen des Rekonstruktionsproblems. Ziel ist es aber eine Lösung zu finden, die so gut wie möglich mit den originalen Seiten übereinstimmt. Aus diesem Grund wird eine Zielfunktion definiert, welche die Differenz zwischen der Länge einer Seitenkante und der Summe aller Außenkanten die dieser Seitenkante zugeordnet sind minimiert. Es werden verschiedene Annäherungen zur Verfügung gestellt, welche dieses Ziel als ganzzahliges lineares Programm implementieren. Von den verschiedenen Annäherungen die als Deltamodus bezeichnet werden, wird hier nur jener Modus beschrieben der auch in dieser Arbeit verwendet wurde.

twovars In diesem Deltamodus wird der Absolutwert in die Zielfunktion eingefügt. Das erfolgt mithilfe einer einzelnen Bedingung und zwei nicht negativer Variablen: $\Delta_w^{\mathrm{L}_{pos}}$ für die positive Längendifferenz und $\Delta_w^{\mathrm{L}_{neg}}$ für die negative Längendifferenz bezüglich jeder Seitenkante. Diese Methode erlaubt es die positiven Differenzen anders wie die negative Differenzen zu gewichten und zu limitieren:

$$\text{minimiere} \sum_{w \in K} \Delta_w^{\mathrm{L}_{pos}} + \sum_{w \in K} \Delta_w^{\mathrm{L}_{neg}} \tag{3.9}$$

$$S_w + \Delta_w^{\mathrm{L}_{pos}} = \Delta_w^{\mathrm{L}_{neg}} + \sum_{i \in N} L_i \cdot x_{i,w} \qquad \forall w \in K \tag{3.10}$$

$$0 \leq \Delta_w^{\mathrm{L}_{pos}} \leq \Delta_{Max} \qquad \forall w \in K \tag{3.11}$$

$$0 \leq \Delta_w^{\mathrm{L}_{neg}} \leq \Delta_{Max} \qquad \forall w \in K \tag{3.12}$$

Die Variable Δ_{Max} ist die obere Schranke für die maximale Längendifferenz bezüglich einer Seitenkante. Berechnet wird diese Schranke wie folgt:

- Die Hilfskonstante $\Delta_{tearWidth}$ wurde durch Beobachtungen festgelegt und gibt die maximal erwartete Weite eines Risses an, der durch den Schereffekt entstanden ist. Üblicherweise liegt der Wert zwischen fünf und zehn Millimeter.
- Die maximal erlaubte Längendifferenz Δ_{Max} berechnet sich aus der maximalen Weite der Risse $\Delta_{tearWidth}$ und der maximalen Anzahl von Rissen entlang einer Seitenkante, die als fünf angenommen wird. Diese Annahme wurde aufgrund der Beobachtungen von Instanzen aus dem realen Leben getroffen. Jeder dieser höchstens fünf Rissen kann $\Delta_{tearWidth}$ weit sein (trifft für beide Schnipsel eines Risses zu):

$$\Delta_{Max} = 2 \cdot 5 \cdot \Delta_{tearWidth} = 10 \cdot \Delta_{tearWidth} \tag{3.13}$$

Das Ergebnis der LILP Formulierung ist eine Funktion, die jede Schnipselkante, eines Außenschnipsels, auf eine Seitenkante einer bestimmten Seite abbildet. Die Anzahl der Ergebnisseiten ist durch die Anzahl der gegebenen Eckschnipsel bestimmt.

Die LAILP Formulierung ist eine Erweiterung von der LILP. Sie verwendet das gesamte LILP Model und erweitert die Bedingungen und die Zielfunktion um die Anordnung der zugeordneten Schnipsel rund um eine Seite, sodass man einen Kreis von Schnipsel bekommt. Das Ziel der Optimierung ist es eine Anordnung der Schnipsel zu finden, bei der die Summe der Winkel von benachbarten Schnipseln so gut wie möglich 180° erreichen. Für weitere Details siehe [1].

Tools

In der Arbeit von Peter Schüller [1] wurden verschiedenen Tools entwickelt, die einerseits die Erstellung der Testinstanzen und andererseits den Prozess der Wiederherstellung von zerrissenen Seiten simulieren. Dies beinhaltet das Zerreißen von Seiten, das Mischen der so erhaltenen Schnipsel, das anschließende Einscannen der Schnipsel um eine digitale Kopie zu erhalten und, die abschließende Anwendung eines Lösungsalgorithums auf die gescannten Schnipsel.

Das Zerreißen einer Seite wird durch den `simulator` nachgebildet. Als Eingabedatei bekommt dieses Tool eine Parameterdatei, die verschiedene Parameter des Zerreißprozesses und der Randomisierung beinhaltet. Die Ausgabe des `simulator`s besteht aus den angegebenen Parametern, einer Liste der durchgeführten Zerreißaktionen, sowie der Abreißhierarchie. Die Abreißhierarchie ist eine Baumstruktur deren Wurzelknoten die Ausgangsseite darstellt. Die Kinder von jedem Knoten sind die Schnipsel die durch das Zerreißen des im Knoten gespeicherten Schnipsels entstehen. Die Blätter der Hierarchie stellen die endgültigen Schnipsel dar die als Input für den `mixer` dienen. Diese Hierarchie beinhaltet Informationen über das Zerreißen und somit auch über die korrekte Lösung.

Der `mixer` bearbeitet eine oder mehrere Ausgabedateien vom `simulator`. Er simuliert den Schereffekt an den Risskanten der Schnipsel, sowie auch das Problem der Kantenerkennung während des Scannvorgangs. Die Ausgabedatei vom `mixer` enthält eine gemischte Liste von veränderten Schnipseln die anschießend als Probleminstanz verwendet werden kann. Der `mixer` ist zusätzlich auch in der Lage optimale Lösungen zu berechnen da er die Abreißhierarchie als Eingabe erhält.

Der `solver` kann entweder mit der LILP Formulierung oder mit der LAILP Formulierung konfiguriert werden. Er bearbeitet die vom `mixer` gelieferte Probleminstanz und erzeugt eine Ausgabedatei, die dasselbe Format besitzt, wie das vom `mixer` generierte Lösungsfile. Bei der Konfiguration mit der LILP Formulierung erhält man eine Zuordnung der äußeren Schnipselkanten zu den einzelnen Seitenkanten. Wird mit LAILP gearbeitet, wird zusätzlich auch die Anordnung der Schnipsel rund um die Seiten geliefert.

Zusätzlich gibt es noch ein `verifier` Tool, das es ermöglicht, die Lösung vom `solver` mit der optimalen Lösung vom `mixer` zu vergleichen und somit ein Maß über die Qualität der Lösung zu bekommen. Die Darstellung der Zerreißhierarchien, der generierten Probleminstanzen und der Lösungen ist mithilfe des `displayer` Tools möglich.

Die in dieser Diplomarbeit verwendeten Probleminstanzen wurden durch die Verwendung des `simulator`s und `mixer`s erzeugt. Anschließend folgt eine kurze Beschreibung ihrer Funktionsweise.

`simulator`

Der `simulator`, wie sein Name schon sagt, simuliert das Zerreißen von Papier. Er verwendet die CGAL Bibliothek [15] für die geometrische Berechnungen und Randomisierungen. In Abbildung 3.1 ist die Ausgabe des `simulator`s mithilfe des `displayer`s dargestellt.

Damit die Probleminstanzen so gut wie möglich denen im realen Leben ähneln wird hier ein Algorithmus verwendet, der versucht das beobachtete Verhalten, von Leuten beim Zerreißen von Papierseiten, zu simulieren. Ein Beispiel für zwei Risse ist in Abbildung 3.2 dargestellt.

Die Parameter die den Algorithmus steuern werden den Simulator in einer XML-Datei zur Verfügung gestellt. Dieser Algorithmus arbeitet wie folgt:

- Begonnen wird mit einem zentrierten Rechteck rund um die ursprüngliche Darstellung der originalen Seite im Querformat (Abbildung 3.2a).
- Ein Riss wird durch eine gerade Linie simuliert. Ein Endpunkt dieser Schnittlinie befindet sich auf der oberen Seite des Rechtecks, der andere Punkt auf der unteren Seite des Rechtecks. Die horizontale Position der beiden Punkte erhält man durch die Parameter *cut.top* und *cut.bottom*. Mithilfe dieser Schnittlinie wird das Rechteck in zwei Schnipsel

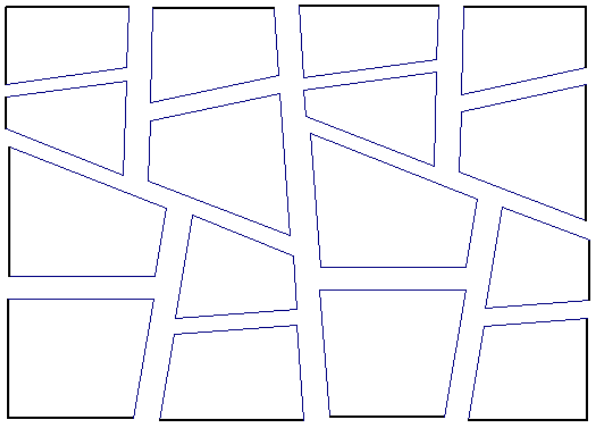


Abbildung 3.1: Darstellung einer typischen `simulator` Ausgabe

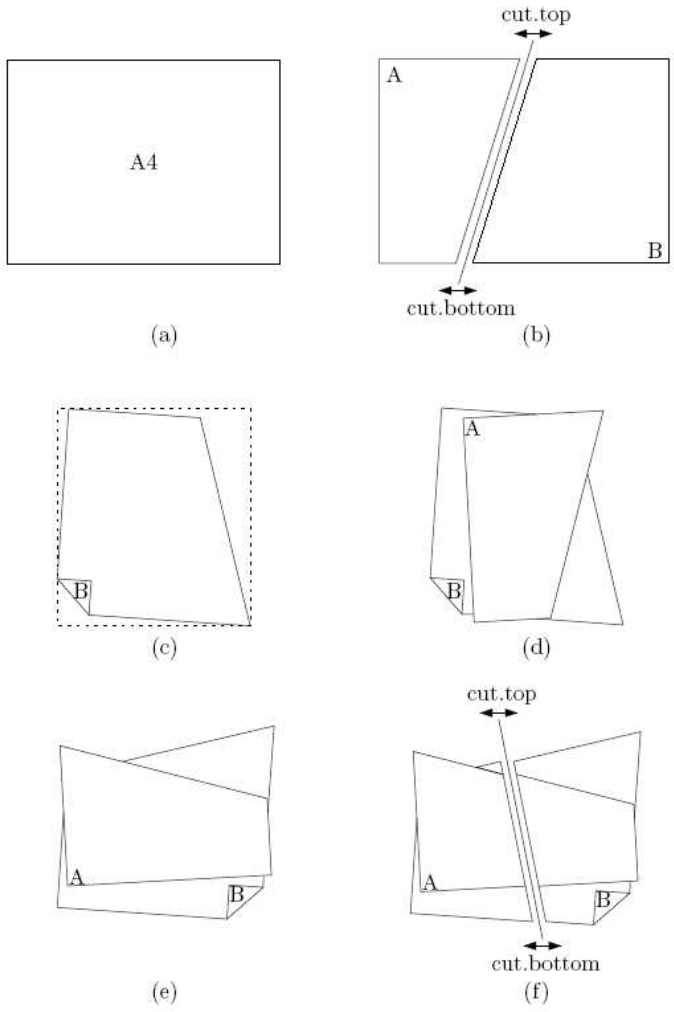


Abbildung 3.2: Zerreißprozess mit 2 Rissen (entnommen aus [1])

zerrissen und zwar in das Schnipsel rechts von der Schnittlinie und in das Schnipsel links von der Schnittlinie (Abbildung 3.2b).

- Platziere den rechten Teil durch die Anwendung der folgenden Umformungen, wobei der Ursprung als Referenzpunkt benutzt wird (Abbildung 3.2c):
  - zentriere das begrenzende Rechteck,
  - spiegle so wie in den Parametern *mirror.x* und *mirror.y* angegeben,
  - rotiete so wie im Parameter *rotate* angegeben,
  - verschiebe so wie in den Parametern *translate.x* und *translate.y* angegeben.
- Wende die selben Umformungen auf den linken Teil an jedoch ohne die Spiegelung, denn um einen Teil gegen einen anderen zu spiegeln ist es nur notwendig einen davon zu spiegeln. Anschließend werden die beiden Teile aufeinander gelegt und zu einem Stack zusammengefasst (Abbildung 3.2d).
- Drehe diesen Stack um 90° nach links oder rechts entsprechend der Angabe durch den Parameter *turn.left* (Abbildung 3.2e).

Um die gewünschte Anzahl von Rissen, wie im Parameter *cut.count* angegeben, zu simulieren müssen die letzten vier Punkte dementsprechenden oft wiederholt werden. Der letzte Schritt der Simulation besteht nur noch aus einem Schnitt (Abbildung 3.2f).

`mixer`

Der `mixer` wird verwendet um aus einen oder mehreren Ausführungen des `simulator`s eine ungeordnete Sammlung von Polygonen zu erzeugen. Die vom `mixer` gelieferte Ausgabe dient als Eingabe für den `solver`. Zusätzlich kann der `mixer` auch optimale Lösungsdateien erzeugen, die der `verifier` benutzt um eine Bewertung der vom `solver` kreierten Lösungen zu erhalten.

Um sicher zu stellen, dass die Lösungen nicht aufgrund der Position oder der Reihenfolge der Schnipsel erzeugt wurden, erstellt der `mixer` eine zufällige Permutation der Schnipsel, normiert die Position jedes Schnipsel indem es sie um ihren Ursprung zentriert und rotiert sie um einen beliebigen Betrag. Abbildung 3.3 zeigt Teile einer typischen `mixer` Ausgabe.

Zwei bedeutende Eigenschaften die beim realen Zerreißprozess auftreten werden durch den `mixer` simuliert: der Schereffekte und die Kantenerkennung während des Scannprozesses.
Die Simulation des Schereffektes ist in Abbildung 3.4 dargestellt und erfolgt durch die Verschiebung der Knoten eines Schnipsel nach folgenden Regeln:

- verschiebe keine Eckpunkte,
- verschiebe Punkte entlang Seitenkanten nur längs dieser Kanten,

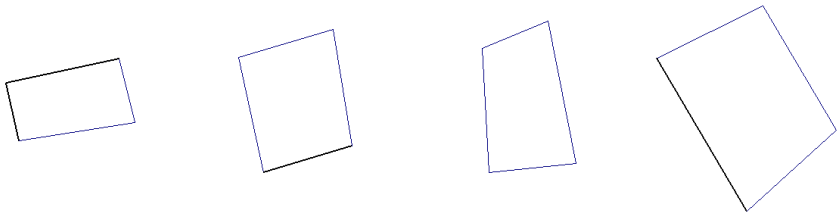


Abbildung 3.3: Typische `mixer` Ausgabe (entnommen aus [1])

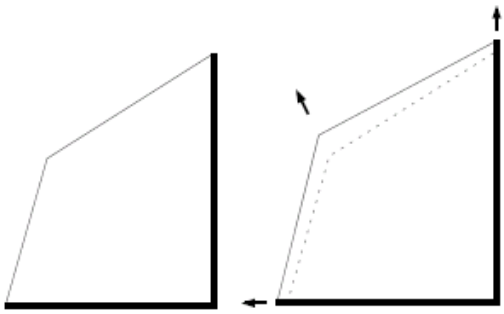


Abbildung 3.4: Simulation des Schereffekts (entnommen aus [1])

- verschiebe andere Punkte vom Zentrum des Schnipsels weg, sodass sich die Fläche des Schnipsel vergrößert.

Die Probleme der Kantenerkennung während des Einscannvorgangs der Schnipsel werden durch das Löschen von Schnipselkanten, welche kleiner als eine angegebene minimale Länge sind, simuliert. Wie in Abbildung 3.5 dargestellt ist, kann dies zu einer Reduzierung der Fläche führen.

Um die oben beschriebenen Tools zu implementieren wurde eine SNIPLIB C++ Bibliothek entwickelt. Diese Bibliothek soll die Umsetzung der Tools erleichtern und zusätzlich auch als Hilfe dienen um die Implementierung von neu entwickelten Lösungsansätzen zu vereinfachen.

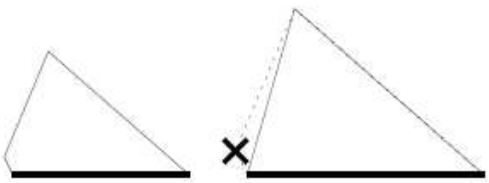


Abbildung 3.5: Kantenerkennung durch den `mixer` (entnommen aus [1])

4 Grundlegende Methoden

In diesem Kapitel werden grundlegende Methoden vorgestellt, die als Basis für die in dieser Arbeit gewählten Ansätze dienen. Mittels sogenannten *AVL-Bäume* als interne Datenstruktur, wird der effiziente Zugriff auf unterschiedliche Schnipsel und der Parameter garantiert. Auf der Seite des Algorithmus werden die *Lokale Suche* (LS) und die *Variable Neigborhood Descent* (VND) näher vorgestellt.

4.1 AVL-Baum

Unter einem *AVL-Baum* versteht man einen *balancierten binären Suchbaum*, bei dem sich für jeden Knoten die Höhe seines linken und rechten Teilbaumes um maximal eins unterscheidet. Benannt ist der *AVL-Baum* nach seinen Erfindern Adelson-Velskii und Landis[1962] [11].

Die *Balance* eines Knoten p ist definiert als die Höhe seines rechten Teilbaums minus der Höhe seines linken Teilbaums.

$$balance(p) = height(p.right) - height(p.left)$$

Durch das Einfügen/Löschen von Knoten in/aus dem Baum kann es vorkommen, dass die Eigenschaft der oben beschriebenen Balance verletzt wird. In diesem Fall muss diese durch geeignete *Rebalancierungsoperationen* wiederhergestellt werden. Die Operationen manipulieren einige Knoten in der Nähe des unbalancierten Knoten, um durch Anheben von einem Teilbaum und Absenken des anderen die Ausgeglichenheit wieder herzustellen. Im balancierten Fall nimmt die Balance die Werte {-1,0,+1} an und im unbalancierten Fall {-2,+2} [11]. Abbildung 4.1 zeigt einen *AVL-Baum* mit Balancen.

Im unbalanciertem Fall stehen vier unterschiedliche *Rebalancierungsoperationen* zur Verfügung, die die Ausgeglichenheit des Baumes wiederherstellen: *Linksrotation, Rechtsrotation, Rechtslinksrotation* und *Linksrechtsrotation* [11].

Linksrotation:

Die Linksrotation wird angewandt, wenn die *Balance* eines Knoten größer als $+1$ (also +2) ist und sein rechter Teilbaum eine *Balance* von 0 oder $+1$ besitzt. In diesem Fall wird der rechte Nachfolger an die Stelle des unbalancierten Knoten verschoben und der unbalancierte Knoten selbst wird zum linken Nachfolger vom seinem ehemaligen rechten Nachfolger. Der

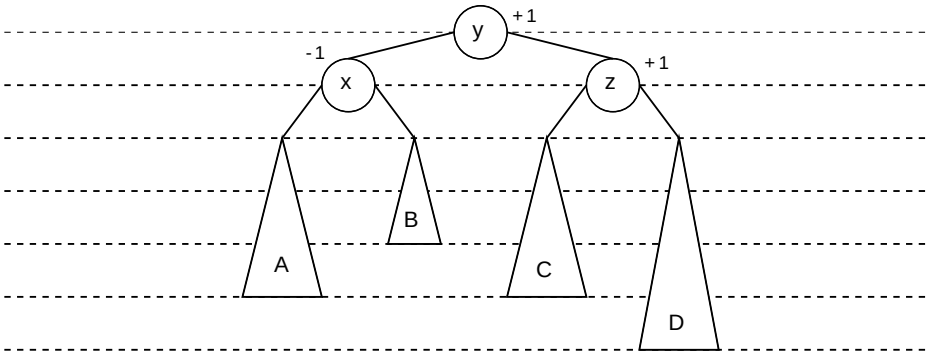


Abbildung 4.1: AVL-Baum mit Balancen

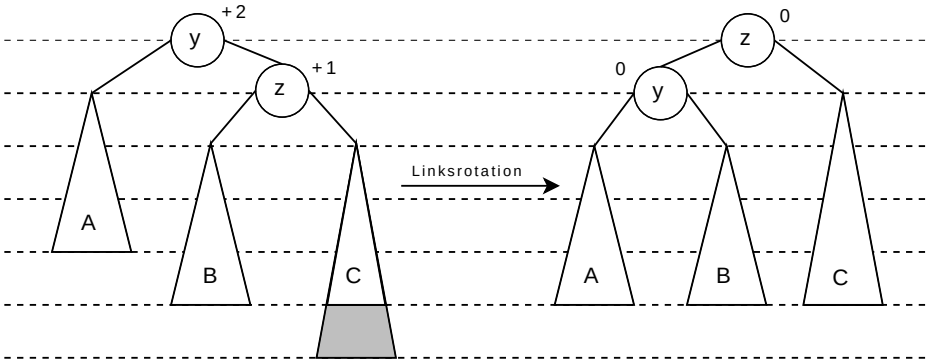


Abbildung 4.2: Linksrotation

linke Nachfolger vom ehemaligen rechten Nachfolger wird zum rechten Nachfolger des unbalancierten Knotens. In Abbildung 4.2 ist ein Beispiel einer Linksrotation dargestellt [11].

- Durch das Einfügen von einem Knoten in den Teilbaum *C* entsteht am Knoten *y* eine Balance von $+2$.
- Da sein rechter Nachfolger *z* eine Balance von $+1$ besitzt, ist eine Linksrotation durchzuführen.
- Der Knoten *z* wird nach oben verschoben und *y* wird zum linken Nachfolger von *z*. Der Teilbaum *B* wird zum rechten Nachfolger von *y*.

Rechtsrotation:

Die Rechtsrotation wird im spiegelverkehrten Fall zur Linksrotation verwendet. Also wenn die *Balance* eines Knoten -2 ist und sein linker Teilbaum eine *Balance* von -1 oder 0 besitzt. In diesem Fall erfolgt die Rebalancierung ebenfalls spiegelverkehrt zur Linksrotation.

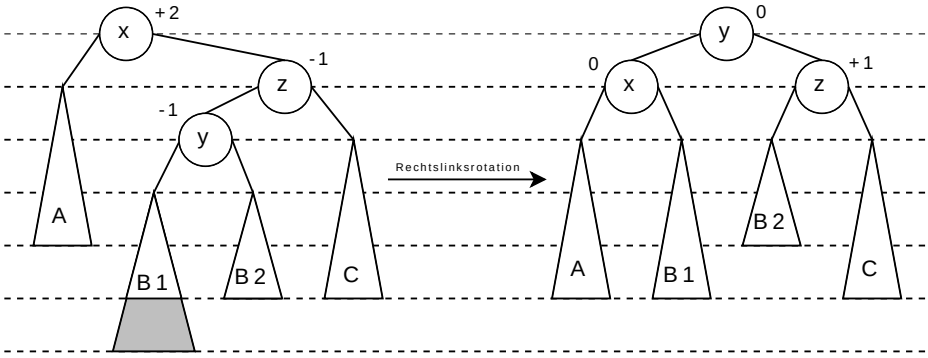


Abbildung 4.3: Rechtslinksrotation

Rechtslinksrotaion:

Die Rechtslinksrotation wird verwendet, wenn die *Balance* eines Knoten +2 ist und die *Balance* seines rechten Nachfolgers -1 ist. In einem solchen Fall reicht eine Operation nicht aus, um die *Balance* wiederherzustellen. Es wird daher zuerst eine Rechtsrotation des rechten Nachfolgers und anschließend eine Linksrotation des unbalancierten Knoten durchgeführt. Ein Beispiel für eine Rechtslinksrotation ist in Abbildung 4.3 dargestellt. Aufgrund der doppelten Rotation wird der Teilbaum *B* genauer betrachtet [11].

- Durch das Einfügen eines Knoten in den Teilbaum *B*1 entsteht am Knoten *x* ein Balance von $+2$.
- Sein rechter Nachfolger *z* hat ein Balance von -1, weshalb die Rechtslinksrotation zu verwenden ist.
- Zuerst wird eine Rechtsrotation vom rechten Nachfolger *z* durchgeführt.
- Anschließend erfolgt eine Linksrotation von Knoten *x*.

Linksrechtsrotation:

Wenn die *Balance* eines Knoten -2 ist und die *Balance* seines linken Nachfolgers den Wert +1 besitzt ist die Linksrechtsrotation zu verwenden. Da dies der umgekehrte Fall zur Rechtslinksrotation ist, wird auch die Rebalancierung umgekehrt zur Rechtslinksrotation durchgeführt.

Der Vorteil von *AVL-Bäumen* liegt in ihrer Ausgeglichenheit. Sie besitzen eine Höhe von *O(log n)* und somit auch eine Suchfunktion die in einer Zeit von *O(log n)* realisiert ist. Der Nachteil liegt in den aufwendigen Rebalancierungsschritten beim Einfügen/Löschen von Knoten, um deren Ausgeglichenheit zu gewährleisten [11].

Algorithmus 4.1 : Lokale Suche

```
Beginn
    x ⟵ Startlösung;
    wiederhole
        wähle x' ∈ N(x);
        wenn f(x') ≤ f(x) dann
            x ⟵ x'
    bis Abbruchkriterium erfüllt ;
Ende
```

4.2 Lokale Suche

Die *Lokale Suche* (LS) ist ein weit verbreiteter Ansatz zum Lösen von kombinatorischen Optimierungsproblemen (KOP). Die Instanz eines KPO ist ein Paar$(\mathcal{L}, f)$, wobei $\mathcal{L}$ die Menge aller möglichen Lösungen ist und die Kostenfunktion $f : \mathcal{L} \rightarrow \mathbb{R}$ jeder Lösung einen numerischen Wert zuordnet. Ziel ist es eine optimale Lösung mit minimalen (maximalen) Kosten zu finden, z.B. ein $x \in \mathcal{L}$, sodass $f(x^*) \leq f(x) \forall x \in \mathcal{L}$, wobei $f^* = f(x^*)$ die optimalen Kosten bezeichnet [12]. Um die Begrifflichkeit zu vereinfachen gehen wir im Folgenden von einem Minimierungsproblem aus. Es sei aber angemerkt, dass jedes Maximierungsproblem in ein äquivalentes Minimierungsproblem umgewandelt werden kann.

Die Grundidee der LS ist von einer gegebenen aktuellen Lösung x, durch geringfügige lokale Änderungen neue, potenziell bessere Lösungen x' zu generieren. Die Menge dieser Lösungen wird auch als Nachbarschaft $N(x)$ von x bezeichnet. Je nachdem welche Schrittfunktion verwendet wird, wird aus $N(x)$ die beste (*best improvement*), die erstbeste (*next improvement*) oder eine zufällig gefundene Lösung für die nächste Iteration in der LS verwendet [12].

Eine Lösung x' in $N(x)$ wird auch als Nachbar von x bezeichnet. Nachbarschaften sind meist implizit durch die Definition von möglichen Veränderungsvorschriften beschrieben. Neben der sinnvollen Wahl einer oder mehrerer Nachbarschaften, sowie der Definition einer Schrittfunktion, muss eine Methode zum Berechnen der Startlösung gewählt werden. Meist wird hier auf zufällige Initialisierung oder auf Greedy Heuristiken basierenden Verfahren zurückgegriffen. Algorithmus 4.1 bildet die prinzipielle Methode der LS ab [12].

Der größte Nachteil der *Lokalen Suche* besteht darin, dass sie im Allgemeinen nur lokal optimale Lösungen liefert. Das sind solche Lösungen, die durch die Anwendung von vordefinierten Zügen nicht mehr verbessert werden können, nicht aber mit einer global optimalen Lösung übereinstimmen müssen [12].

Algorithmus 4.2 : Basic-VND

Eingabe : Menge von Nachbarschaftsstrukturen N_l, für $l = 1, \ldots, l_{max}$
Ausgabe : Lösung x, die bezüglich aller Nachbarschaftsstrukturen lokal optimal ist

```
Beginn
    x ⟵ Startlösung;
    l ⟵ 1;
    wiederhole
        finde ein x' mit f(x') ≤ f(x''), ∀x'' ∈ N_l(x);
        wenn f(x') ≤ f(x) dann
            x ⟵ x';
            l ⟵ 1;
        sonst
            l ⟵ l + 1;
    bis l < l_max ;
    zurück x
Ende
```

4.3 Variable Neighborhood Descent

Um dem prinzipiellen Problem der *Lokalen Suche* zu entkommen, kann man Verfahren wie die *Variable Neiborhood Descent* (VND) verwenden. VND basiert auf der Idee, dass ein lokales Optimum in Bezug auf eine Nachbarschaftsstruktur nicht unbedingt ein lokales Optimum in Bezug auf eine andere Nachbarschaftsstruktur ist. Es kann somit von Vorteil sein verschiedene Nachbarschaften zu kombinieren und diese systematisch zu durchsuchen [13].

Bei VND werden die ausgewählten Nachbarschaftsstrukturen $N_1 \ldots N_{l_{max}}$ der Reihen nach angewandt. Wenn dabei in einer Nachbarschaftstruktur N_l keine Lösung x' gefunden, welche die aktuelle Lösung x verbessert, wird die nächste Nachbarschaftstruktur N_{l+1} verwendet. Wird jedoch in irgendeiner Nachbarschaft N_l eine bessere Lösung gefunden, wird wieder in der ersten Nachbarschaftstruktur N_1 weitergesucht. Die VND terminiert, wenn in keiner der Nachbarschaftsstrukturen $N_1 \ldots N_{l_{max}}$ mehr eine besser Lösung gefunden werden kann [14].

Letztendlich erreicht VND eine Lösung, die bezüglich aller gegebenen Nachbarschaftsstrukturen $N1 \ldots N_{l_{max}}$ lokal optimal ist. Als Schrittfunktion wird üblicherweise *best improvement* oder *next improvement* verwendet. Die Anordnung der Nachbarschaftsstrukturen erfolgt meist nach ihrer Größe $|N_l(x)|$ bzw. der Komplexität ihrer Auswertung, kann aber auch zufällig sowie mittels komplexeren Verfahren erfolgen. Die prinzipielle Funktionsweise ist im Algorithmus 4.2 dargestellt [14].

Bei der Anwendung dieses Schemas sollte man hinsichtlich einiger Punkte genauere Überlegungen anstellen. Zum einen ist die Komplexität der unterschiedlichen *Moves* von Interesse. Wenn zu viele elementare Änderungen gemacht werden (z.B. Komplementierung von drei

Elementen oder mehr eines 0-1 Vektors), wird die Heuristik sehr langsam und kann bei kleinen oder mittleren Instanzen oft mehr Zeit benötigen als ein exakter Algorithmus. Wichtig ist es auch darauf zu achten, dass für manche Probleme einfache *Moves* oft nicht ausreichen um aus einem engen Tal (lokalen Optimum) zu entkommen. Heuristiken, die solche *Moves* verwenden, liefern nur sehr schlechte Ergebnisse. Die gewünschte Genauigkeit der VND hängt davon ab, ob sie alleine verwendet wird oder ob sie ein Teil eines größeren Systems ist. Im ersten Fall ist man bestrebt die bestmögliche Lösung innerhalb der zur Verfügung stehenden Zeit zu erhalten. Im zweiten Fall bevorzugt man eine schnelle gute Lösung, die anschließend mit weiteren Verfahren verbessert wird [13].

5 Lösungsansatz

Der in diesem Kapitel vorgestellte Algorithmus `astp` (*assign snips to pages*) ist mithilfe der von Peter Schüller entwickelten SNIPLIB C++ Bibliothek [1] implementiert worden.

Der Algorithmus `astp` ist in zwei Hauptschritte unterteilt, siehe Abbildung 5.1. Im ersten Teil (`preprocess`) wird die Eingabedatei verarbeitet und die berechneten Informationen der Schnipsel in eine Datenstruktur eingefügt. Als Eingabedateien erhält der Algorithmus Instanzen die durch die Anwendung des `simulator` und `mixer` Tools (Siehe Abschnitt 3) generiert wurden. Der zweite Teil (`solver`) stellt unterschiedliche Methoden für die Lösung des *Tearing Paper Problems* zur Verfügung. Je nach ausgewählter Methode wird eine entsprechende Lösung ermittelt.

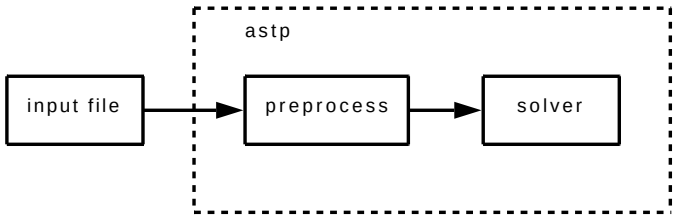


Abbildung 5.1: Komponenten des Algorithmus `astp`

5.1 Aufbereitung der Daten

Zu Beginn des Algorithmus erfolgt eine Vorverarbeitung der Instanz (`preprocess`) bei der alle Informationen der Schnipsel ermittelt werden, welche für die anschließende Lösungsberechnung benötigt werden. Die einzelnen Bearbeitungsschritte dieser Vorverarbeitung sind in Abbildung 5.2 zu sehen.

Zuerst werden die Schnipsel aus der Eingabedatei eingelesen. Für jedes eingelesene Schnipsel $i \in \mathcal{S}$ werden anschließend alle relevanten Informationen berechnet. Dazu gehören unter anderem die Fläche A_i^S eines Schnipsels, die Kantenlänge L_i^S, deren Winkeln α_1^i und α_2^i zu den Außenkanten und der Typ des Schnipsels (Eck-, Rand- oder Innenschnipsel). Alle ermittelten Informationen des Schnipsels *i* werden in dem Objekt $snipinfo_i$ der Klasse `SnipInfo` gespeichert.

Die berechneten Schnipselinformationen ($snipinfo_i$) werden anschließend als Knoten in eine

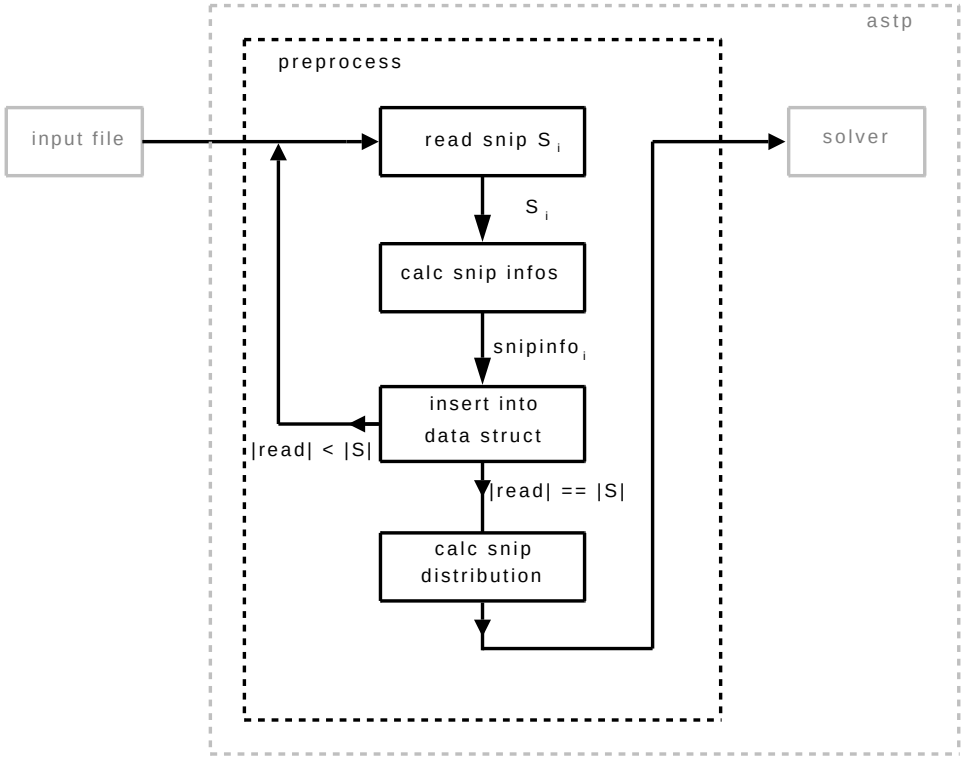


Abbildung 5.2: Komponenten der Vorbehandlung

geeignete Datenstruktur eingefügt, wobei hier zwischen drei verschiedenen Datenstrukturen unterschieden wird. Je nach Typ des eingelesenen Schnipsel wird $snipinfo_i$ entweder in den *CornerTree, BorderTree* oder in den *InnerTree* eingefügt.

Nachdem alle Schnipsel eingelesen, ihre Daten berechnet und in die entsprechende Datenstruktur einfügt wurden, wird abschließend die Verteilung der unterschiedlichen Seitentypen berechnet. Also die Anzahl der Seiten mit 2, 3, 4 oder 5 Rissen, auf welche die gegebenen Schnipsel später zugeordnet werden sollen.

5.1.1 Daten einlesen

Zu Beginn des Algorithmus `astp` werden die einzelnen Schnipsel des *Tearing Paper Problems* aus einer Datei eingelesen. Da für die Implementierung des Algorithmus und die Generierung der Testdaten die SNIPLIB C++ Bibliothek [1] verwendet wurde, wollen wir uns die dort verwendete Formatierung genauer betrachten.

Format der Eingabedatei

Die SNIPLIB C++ Tools, `simulator` und `mixer`, werden für die Erzeugung der Instanzen verwendet. Die Anwendung dieser Tools erlaubt es Probleminstanzen zu generieren, die denen eines realen Zerreißprozesses ähneln. Mithilfe des `simulator`s wird der Vorgang des Zerreißens simuliert. Anschließend verwendet man den `mixer`, um für die einzelnen Schnipsel den Schereffekt und das Problem der Kantenerkennung beim Einscannen nachzubilden. Der `mixer` ist zusätzlich in der Lage die Schnipsel zu mischen, sie zu rotieren, zu spiegeln und um den eigenen Ursprung zu zentrieren. Als Ergebnis der beiden Tools erhält man eine Kollektion von Schnipsel, die ähnliche Eigenschaften wie reale Schnipsel aufweisen und deren Reihenfolge und Anordnung keine Rückschlüsse für die Rekonstruktion der ursprünglichen Seiten liefert.

Die von den beiden Tools (`simulator` und `mixer`) erzeugten Instanzen werden in XML-Dateien abgespeichert. Eine Instanz besteht aus einer Kollektion von einem oder mehreren Schnipsel. Jedes dieser Schnipsel enthält als Attribut eine eindeutige ID. Ein Schnipsel besteht wiederum aus einem Polygon mit einem oder mehreren Punkten die gegen den Uhrzeigersinn aufgelistet sind. Als Attribute besitzt jeder Punkt eine Positionsangabe durch die `x` und `y` Koordinaten, eine eindeutige ID und die Information, ob es sich bei der von diesem Punkt ausgehenden Kante (gegen den Uhrzeigersinn) um eine Außenkante oder um eine Innenkante handelt.

5.1.2 Schnipselinformationen berechnen

Im nächsten Schritt werden aus den eingelesenen Schnipseldaten, den angegebenen Koordinatenpunkten eines Polygons und der Information, ob es sich bei einer Kante um eine innere oder äußere handelt, alle Informationen berechnet die später für die Generierung einer Lösung für das TPP verwendet werden. Nach ihrer Berechnung werden alle Informationen eines Schnipsels in der Klasse `SnipInfo` gespeichert.

Aus den gegebenen Schnipseldaten werden die folgenden Informationen ermittelt:

- Der Flächeninhalt eines Schnipsel
- Der Schnipseltyp (Eck-, Rand-,Innenschnipsel)
- Die Länge aller Außenkanten
- Die Winkeln zu den Außenkanten

Fläche

Die Fläche A_i^S entspricht der Fläche des Schnipsels *i*. Diese Information wird später verwendet, um eine möglichst gute Auswahl treffen zu können, damit die Gesamtfläche aller einer

Seite zugeordneten Schnipsel, der Fläche einer Originalseite entspricht. Die Flächenberechnung selbst wird mit Hilfe der CGAL-Bibliothek [15] durchgeführt.

$$\mathrm{A}_i^{\mathrm{S}} = S_i.area() \tag{5.1}$$

Typ

Jedem Schnipsel *i* wird ein Typ $type_i$ zugeteilt, der anhand der Außenkanten des Schnipsels ermittelt wird. Je nach Anzahl dieser wird zwischen Innen-, Rand- und Eckschnipsel unterschieden.

$$type = \begin{cases} inner & \text{wenn Anzahl der Außenkanten = 0)} \\ border & \text{wenn Anzahl der Außenkanten = 1)} \\ corner & \text{wenn Anzahl der Außenkanten = 2)} \end{cases} \tag{5.2}$$

Ausgehend von der Annahme, dass jeder Riss alle vorkommenden Schnipsel jeweils in zwei Teile teilt und zwei aufeinanderfolgende Risse jeweils orthogonal aufeinander ausgeführt werden, kann man annehmen, dass ab zwei Rissen jedes Schnipsel maximal zwei Außenkanten besitzen kann. Ist dies der Fall handelt es sich um ein Eckschnipsel (*corner*). Existiert nur eine Außenkante, spricht man von einem Randschnipsel (*border*). Alle anderen Schnipsel werden als Innenschnipsel (*innen*) bezeichnet.

Außenkantenlänge

Zusätzlich zur Anzahl der Außenkanten wird auch deren Länge $\mathrm{L}_i^{\mathrm{S}}$ berücksichtigt. Hierbei wird allerdings nicht die Länge der einzelnen Kanten gespeichert, sonder nur die Summe aller Außenkantenlängen. Daraus ergibt sich, dass für Innenschnipsel $\mathrm{L}_i^{\mathrm{S}} = 0$ gilt.

$$L_i^S = \begin{cases} 0 & \text{wenn } i \text{ = Innenschnipsel} \\ \text{Länge der Außenkanten} & \text{sonst} \end{cases} \tag{5.3}$$

Winkel

Zuletzt wird noch der gesamte Winkel α_i^{S} eines Schnipsel *i* ermittelt, der sich aus der Summe der Winkel zwischen Außen- und Innenkanten zusammensetzt. Da Innenschnipsel keine Außenkanten besitzen, gilt für diese $\alpha_i^{\mathrm{S}} = 0$.

$$\alpha_i^S = \begin{cases} 0 & \text{wenn } i \text{ = Innenschnipsel} \\ \sum(\text{Winkeln zw. Innnen- und Außenkanten}) & \text{sonst} \end{cases} \tag{5.4}$$

In den Abbildungen 5.3a-5.3c sind die drei unterschiedlichen Typen von Schnipsel mit ihren berechneten Eigenschaften dargestellt.

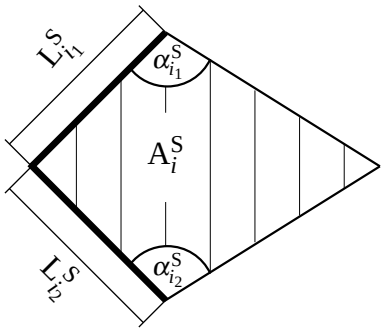


Abbildung 5.3a: Eckschnipsel

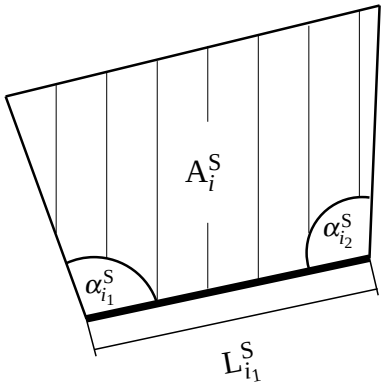


Abbildung 5.3b: Randschnipsel

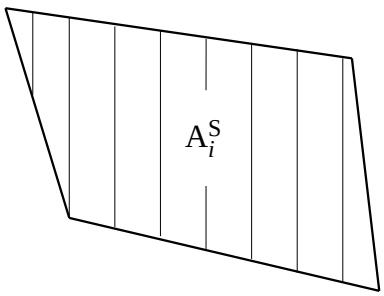


Abbildung 5.3c: Innenschnipsel

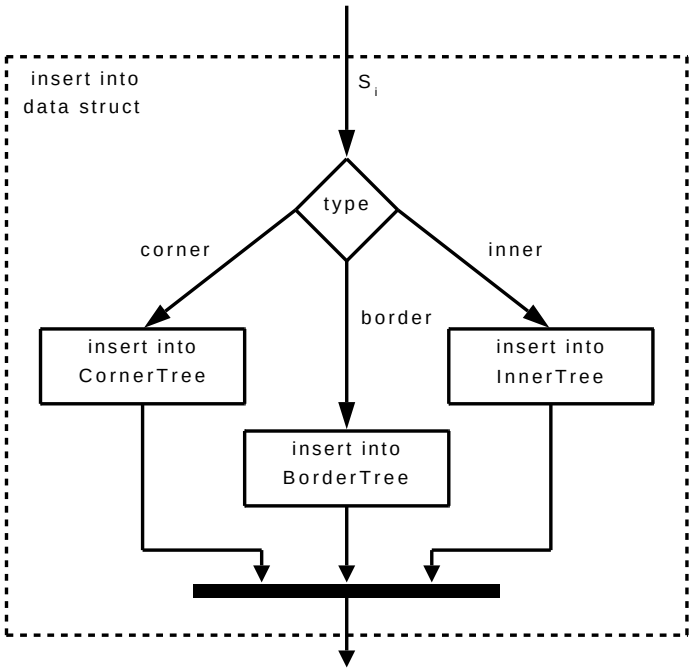


Abbildung 5.4: Einfügen eines Knoten in den entsprechenden Baum

SnipInfo Klasse

Die im vorherigem Abschnitt vom Schnipsel $i \in \mathcal{S}$ ermittelten Informationen (Fläche, Typ, Kantenlänge und Winkel) werden anschließend im Objekt $snipinfo_i$ der Klasse `SnipInfo` gespeichert.

Zusätzlich zu diesen Informationen werden noch Verwaltungsfunktionen gespeichert, die es in weiterer Folge ermöglichen, die Schnipsel mit Hilfe einer eigens dafür entwickelten Datenstruktur schnell und effizient zu finden.

5.1.3 Datenstruktur

Um einen effizienten Zugriff auf die einzelnen Schnipsel zu ermöglichen wurde eine neue abstrakte Datenstruktur entwickelt, die sich im Grunde aus mehreren miteinander verschachtelten und miteinander verknüpften AVL-Bäumen zusammensetzt. Ein AVL-Baum ist ein binärer Suchbaum, für den gilt, dass die Höhe des Baumes in $ld(n)$ liegt, wenn n die Anzahl der Knoten im Baum bezeichnet (siehe Abschnitt 4.1).

Um eine Zeitersparnis beim Suchen von Schnipsel zu erlangen, werden die Papierstücke je nach Typ entweder in einen *CornerTree, BorderTree* oder *InnerTree* eingefügt, wie in Abbildung 5.4 zu sehen ist.

Während der *InnerTree* aus einem AVL-Baum, bei dem die Knoten nach der Fläche sortiert

	A	B	C	D	E	F
A_i^S	1	2	3	4	5	6
L_i^S	6	4	5	1	2	3
α_i^S	4	6	1	5	2	3

Tabelle 5.1: Beispieldaten für die Datenstruktur

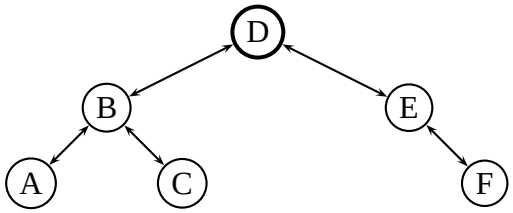


Abbildung 5.5a: Area-Tree

sind, besteht, setzen sich *CornerTree* und *BorderTree* aus drei AVL-Bäumen zusammen, wobei jeweils einer für Fläche, einer für Länge der Außenkanten und einer für Winkel zur Verfügung steht.

Um den Vorteil, der Vereinigung der AVL-Bäume besser zu verstehen, sehen wir uns dazu ein Beispiel an: Gegebenen sind sechs Schnipsel *A, B, C, D, E, F* mit den in Tabelle 5.1 angegebenen Informationen.

Der AVL-Baum, aufgebaut nach den Flächen der Schnipsel, ist Abbildung 5.5a dargestellt. Wenn wir dieselben Knoten ($snipinfo_i$) nach ihren Kantenlängen aufbauen ändert sich die Anordnung der Knoten im AVL-Baum, siehe Abbildung 5.5b. Auch beim Aufbau des Baumes nach den Winkeln der einzelnen Schnipsel erhält man wieder eine neue Anordnung der Knoten, wie in Abbildung 5.5c zu sehen ist.

Somit erhält man drei unterschiedliche Wurzelknoten und Anordnungen der Knoten im Baum. Verwendet man als Datenstruktur zum Beispiel einen AVL-Baum der nach den Flächen A_i^S

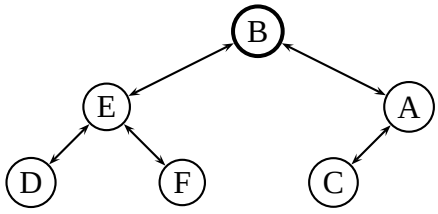


Abbildung 5.5b: Edge-Tree

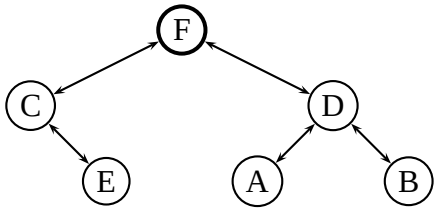


Abbildung 5.5c: Angle-Tree

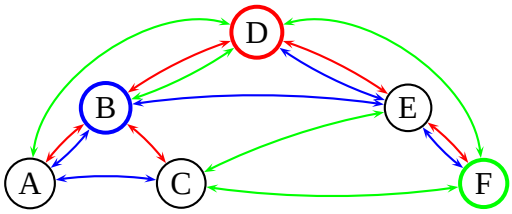


Abbildung 5.6: Beispiel der Datenstruktur: 3 verbundene AVL-Bäume

aufgebaut ist, so kann man den Baum sehr gut nach Schnipseln mit einer gewissen Fläche durchsuchen und zwar in $O(log\ n)$. Möchte man jetzt aber nach einem Schnipsel mit einer bestimmten Kantenlänge L_i^S oder nach einem Schnipsel mit einem bestimmten Winkel α_i^S suchen, erhöht sich der Suchaufwand auf $O(n)$, da der gesamte Baum durchsucht werden muss. Derselbe Effekt tritt auf, wenn man in einem AVL-Baum, der nach den Kantenlängen L_i^S aufgebaut ist, nach einer bestimmten Fläche oder einen Winkel sucht. Ähnliches gilt auch bei einem AVL-Baum der nach den Winkeln α_i^S der Schnipseln entwickelt wurde.

Um jetzt effizient nach allen drei Schnipselinformationen suchen zu können werden die drei AVL-Bäume miteinander verbunden, wie in Abbildung 5.6 zu sehen ist. Die roten Linien stellen den Baum nach den Flächen aufgebaut dar, wobei der Knoten D die dazugehörige Wurzel ist. Der Baum nach den Kantenlängen aufgebaut wird durch die blauen Linien und den Wurzelknoten B dargestellt. Abschließend beschreiben die grünen Linien mit der Wurzel F den Baum der nach den Winkeln der Schnipsel erstellt wurde.

Einfügen eines Knoten

Das Einfügen eines Knoten ($snipinfo_i$) in die Datenstruktur erfolgt durch Setzen der entsprechenden Verknüpfungen zu den anderen Knoten. Beim Einfügen muss aber darauf geachtet werden, dass die Balanceeigenschaft (siehe Abschnitt 4.1) für jeden einzelnen AVL-Baum nicht verletzt wird. Aus diesem Grund werden für jeden Knoten zusätzlich auch seine Höhen in den entsprechenden Bäumen (*area, edge, angle*) mitgespeichert, um daraus die Balance berechnen zu können.

Bezeichung	Beschreibung
a	Anzahl der Seiten mit 4 Schnipsel (2 Risse)
b	Anzahl der Seiten mit 8 Schnipsel (3 Risse)
c	Anzahl der Seiten mit 16 Schnipsel (4 Risse)
d	Anzahl der Seiten mit 32 Schnipsel (5 Risse)
#Snips	Anzahl der gegebenen Schnipsel: $\|\mathcal{S}\|$
#Corner	Anzahl der gegebenen Eckschnipsel: $\|\mathcal{C}\|$
#Border	Anzahl der gegebenen Randschnipsel: $\|\mathcal{B}\|$
#Inner	Anzal der gegebenen Innenschnipsel: $\|\mathcal{I}\|$
$\#Page_{345}$	Anzahl der Seiten mit 3, 4 und 5 Rissen: $b+c+d$

Tabelle 5.2: Bezeichungen für die Berechnung der Seitenverteilung

Befindet sich die Balance eines Knoten, in einem der Bäume, außerhalb der zulässigen Toleranz von {-1, 0, +1}, müssen entsprechende *Rebalancierungsoperationen* durchgeführt werden, um die Ausgeglichenheit des Baums wieder herzustellen. Dazu werden einfach im entsprechenden Baum die Verknüpfungen der von der Rebalancierung betroffenen Knoten neu gesetzt.

5.1.4 Berechnung der Seitenverteilung

Nachdem nun alle Schnipsel eingelesen, deren Informationen berechnet und diese in der Datenstruktur gespeichert wurden, wird im nächsten Schritt die Verteilung der unterschiedlichen Seitentypen berechnet. Dazu werden in Tabelle 5.2 zunächst einige Bezeichnungen vorgestellt, welche in der folgenden Erklärung verwendet werden.

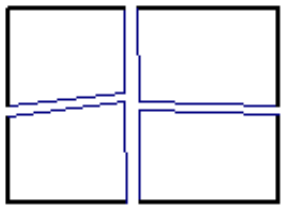
Durch die in Abschnitt 2.1 angegebenen Annahmen gibt es vier verschiedene Möglichkeiten, wie sich die Schnipsel auf einer Seite verteilen. Abbildung 5.7 zeigt diese vier Möglichkeiten mit einer Auflistung der jeweils beteiligten Schnipsel.

Aus den dargestellten vier Verteilungsmöglichkeiten der Schnipsel lassen sich die folgenden Informationen ermitteln:

$$4a + 8b + 16c + 32d = \#Snips \tag{5.5}$$
$$4a + 4b + 4c + 4d = \#Corner \tag{5.6}$$
$$4b + 8c + 16d = \#Border \tag{5.7}$$
$$4c + 12d = \#Inner \tag{5.8}$$

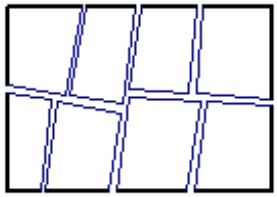
Die Anzahl der Seiten mit zwei Rissen (a) kann ohne großen Aufwand sofort berechnet werden, indem von der Gleichung (5.5) zweimal die Gleichung (5.7) subtrahiert wird. Dadurch

2 Risse



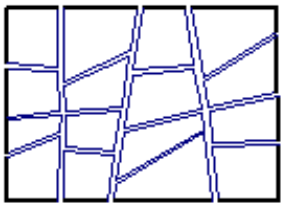
4 Eckschnipsel

3 Risse



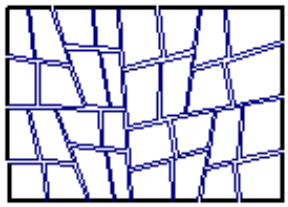
4 Eckschnipsel
4 Randschnipsel

4 Risse



4 Eckschnipsel
8 Randschnipsel
4 Innenschnipsel

5 Risse



4 Eckschnipsel
16 Randschnipsel
12 Innenschnipsel

Abbildung 5.7: Unterschiedliche Schnipselverteilungen

ergibt sich für a:

$$a = \frac{(\#Snips - 2 \cdot \#Border)}{4} \tag{5.9}$$

Somit verbleiben für die weitere Berechnung die Gleichungen (5.7), (5.8) und (5.6) in umgeformter Form:

$$b + c + d = \frac{\#Corner}{4} - a = \#Page_{345} \tag{5.10}$$

Da, nach der Ermittlung von a, die Gleichung (5.5) ident mit (5.7) ist, wird für die weitere Berechnung, nur mehr die Gleichung (5.7) verwendet. Wenn keine Innenschnipsel vorhanden sind ($\#Inner = 0$), ist die Lösung des Gleichungssystems einfach:

$$\begin{aligned} b &= \#Page_{345} \\ c &= 0 \\ d &= 0 \end{aligned} \tag{5.11}$$

Wenn allerdings die Anzahl der Innenschnipsel ungleich null ist, ist die Lösung dieses Gleichungssytems nicht eindeutig möglich, da (5.8) eine Linearkombination von (5.6) und (5.7) ist. Um aus den verschiedenen, zulässigen, Lösungen diejenige Verteilung zu erhalten die am besten die eingelesene Instanz annähert, wird wie folgt vorgegangen:

- Zuerst wird c berechnet:

$$c = \frac{3}{2}\#Pages_{345} - \frac{\#Inner}{8} - \frac{3}{2}b \tag{5.12}$$

- Aus der Gleichung (5.12) kann bestimmen werden ob b gerade oder ungerade ist, indem man zuerst c_{sub} mit:

$$c_{sub} = \frac{3}{2}\#Pages_{345} - \frac{\#Inner}{8} \tag{5.13}$$

  ermittelt. Da c ganzzahlig sein muss, denn unvollständige Seiten sind durch die getroffenen Annahmen nicht zulässig, ergibt sich für b folgendes:

$$b = \begin{cases} \text{gerade} & \text{wenn } c_{sub} = \text{ gerade} \\ \text{ungerade} & \text{wenn } c_{sub} = \text{ ungerade} \end{cases} \tag{5.14}$$

- Im nächsten Schritt wir die maximale Anzahl von erlaubten Seiten mit drei Rissen (b_{max}) berechnet. Dies erfolgt durch die folgenden Punkte:
  - Da mindestens eine Seite mit vier oder fünf Rissen vorhanden ist (#Innen $\neq 0$), erhält b_{max} zu Beginn den Wert: $b_{max} = \#Page_{345} - 1$.

– Im nächsten Punkt wird überprüft ob b_{max} gerade oder ungerade ist und dementsprechend an b (5.14) angepasst:

$$b_{max} = \begin{cases} b_{max} & \text{wenn } (b = \text{ gerade } \wedge\ b_{max} = \text{ gerade }) \quad \vee \\ & \quad (b = \text{ ungerade } \wedge\ b_{max} = \text{ ungerade }) \\ b_{max} - 1 & \text{wenn } (b = \text{ gerade } \wedge\ b_{max} = \text{ ungerade }) \quad \vee \\ & \quad (b = \text{ ungerade } \wedge\ b_{max} = \text{ gerade }) \end{cases} \tag{5.15}$$

– Zuletzt wird b_{max} solange um zwei reduziert bis die Bedingung:

$$(b_{max} > 1 \quad \wedge \quad c_{sub} - \frac{3}{2} \cdot b_{max} > 0) \tag{5.16}$$

erfüllt ist.

- Alle zulässigen Lösungen erhält man, indem ausgehend von $b = b_{max}$, b solange um zwei reduziert wird, solange $b \geq 0$ und $b + c \leq \#Pages_{345}$ ist. Um nun aus allen zulässigen Lösungen diejenige zu ermitteln, die die gegebene Instanz am besten annähert, wurden die Schnipsel $i \in \mathcal{S}$ bei der Berechnung ihrer Fläche in vier unterschiedliche Bereiche $(\mathrm{T}_2, \mathrm{T}_3, \mathrm{T}_4, \mathrm{T}_5)$ eingeteilt, mit:

$$\begin{aligned} \mathrm{T}_2 &= \text{Menge aller Eckschnipsel } i \in \mathcal{C} \text{ für die gilt:} \quad \mathrm{A}_i^{\mathrm{S}} > \frac{3}{16} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \\ \mathrm{T}_3 &= \text{Menge aller Außenschnipsel } i \in \mathcal{O} \text{für die gilt:} \quad \frac{3}{16} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \leq \mathrm{A}_i^{\mathrm{S}} < \frac{3}{32} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \\ \mathrm{T}_4 &= \text{Menge aller Schnipsel } i \in \mathcal{S} \text{ für die gilt:} \quad \frac{3}{32} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \leq \mathrm{A}_i^{\mathrm{S}} < \frac{3}{64} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \\ \mathrm{T}_5 &= \text{Menge aller Schnipsel } i \in \mathcal{S} \text{ für die gilt:} \quad \mathrm{A}_i^{\mathrm{S}} \leq \frac{3}{64} \cdot \mathrm{A}_{origin}^{\mathrm{P}} \end{aligned} \tag{5.17}$$

Wobei bei der Einteilung von der Überlegung ausgegangen wird, dass Schnipsel, die von Seiten mit zwei Rissen (T_2) stammen, eine eher größere Fläche besitzen als Schnipsel, die von Seiten mit drei Rissen (T_3) stammen, usw. Diese Annahme ist nicht zu hundert Prozent korrekt, aber für den größten Teil der gegebenen Schnipsel trifft sie zu. Natürlich kann es auch vorkommen, dass Schnipsel existieren, die größer oder kleiner sind als der für sie zugedachte Bereich. Somit liefert diese Einteilung zwar keine exakten Ergebnisse, dafür aber eine gute Tendenz über die tatsächliche Verteilung der Seiten. Die Anzahl der Schnipsel im jeweiligen Bereich wird in der Folge mit $t_2 = |\mathrm{T}_2|, t_3 = |\mathrm{T}_3|, t_4 = |\mathrm{T}_4|$ und $t_5 = |\mathrm{T}_5|$ bezeichnet.

Da die Anzahl von a in (5.9) berechnet wurde, ist die genaue Anzahl von Schnipseln in T_2 bekannt ($4 \cdot a$). Dadurch kann t_3 angepasst werden:

$$t_3 = t_3 + (t_2 - 4 \cdot a) \tag{5.18}$$

Empirische Tests über die Verteilung der Schnipsel in den angegebenen Bereichen (5.17), entsprechend ihrer Fläche und der Anzahl der Risse, haben folgende Aufteilung ergeben:

$$\begin{aligned} t_{3_{calc}} &= 8b \cdot 87\% + 16c \cdot 7\% \\ t_{4_{calc}} &= 8b \cdot 13\% + 16c \cdot 76\% + 32d \cdot 13\% \\ t_{5_{calc}} &= 16c \cdot 17\% + 32d \cdot 87\% \end{aligned} \tag{5.19}$$

Von allen berechneten, zulässigen, Lösungen wird diejenige Verteilung der Seiten verwendet, welche in Summe die geringste Abweichung zwischen der gezählten und berechneten Schnipselzahl, für die jeweiligen Bereiche, liefert:

$$best_{distribution} = min\{|t_{3_{calc}} - t_3| + |t_{4_{calc}} - t_4| + |t_{4_{calc}} - t_4|\} \tag{5.20}$$

Im Fall, dass mehrere Verteilungen dieselbe minimale Abweichung liefern, wird zufällig eine von ihnen ausgewählt.

5.2 Lösungsverfahren

Nachdem nun die Instanz eingelesen und die berechneten Schnipselinformationen in der entsprechenden Datenstruktur gespeichert wurden, werden in diesem Abschnitt unterschiedliche Methoden für die Lösung des *Tearing Paper Problems* vorgestellt. In Abbildung 5.8 ist der generelle Ablauf des Lösungsverfahren (`solver`) dargestellt.

Zunächst wird eine Initialisierung der zu rekonstruierenden Dokumentenseiten vorgenommen. Anschließend wird eine gültige Startlösung für das TPP, mittels einer greedy Konstruktionsheuristik, erzeugt. Ausgehend von dieser Startlösung wird versucht, diese durch die Anwendung unterschiedlicher Verbesserungsverfahren zu verfeinern. Um eine Aussage über die Qualität unseres Lösungsverfahrens treffen zu können, wird für Testzwecke abschließend noch eine Bewertung der endgültigen Lösung vorgenommen.

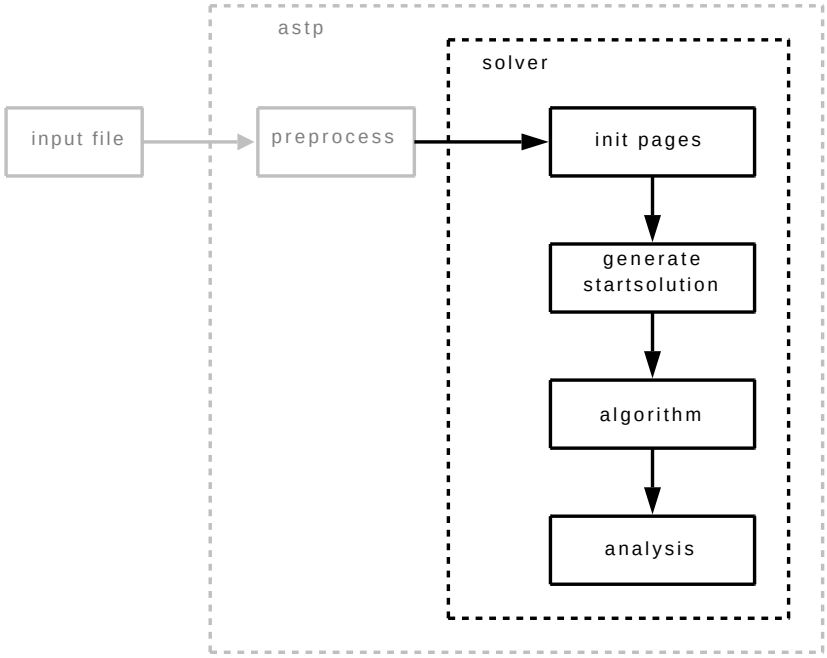


Abbildung 5.8: Komponenten des Lösungsverfahrens

5.2.1 Seiten initialisieren

Zu Beginn des Lösungsverfahrens werden jene Seiten initialisiert, auf welche die eingelesenen Schnipsel zugeordnet werden sollen.

Durch die in der Vorbehandlung (`preprocess`) ermittelten Anzahl von Eckschnipsel (*#Corner*), kann sehr einfach die Zahl der Seiten *m* berechnet werden, auf der die Schnipsel verteilt werden sollen.

$$m = \frac{\#Corner}{4} \tag{5.21}$$

Zusätzlich besitzt man über die berechnete Seitenverteilung auch noch die Information, wie viele Schnipsel auf die jeweiligen Seiten zugeordnet werden sollen. Die folgenden binären Variablen stellen die Zuordnung der Schnipsel zu den Seiten dar.

$$y_{i,j} = \begin{cases} 1 & \text{wenn Schnipsel } i \text{ der Seite } j \text{ zugeordnet ist} \\ 0 & \text{sonst} \end{cases} \tag{5.22}$$
$$\forall i \in \mathcal{S}, j \in \mathcal{P}$$

Wegen des in Abschnitt 2.3 beschriebenen Schereffekts tritt die Eigenschaft auf, dass bei einer optimalen Zuordnung der Schnipsel zu den Seiten die Gesamtfläche, der Umfang und die Winkelsumme einer Seite von den Werten der ursprünglichen Seite abweichen:

- Die Gesamtfläche $\mathrm{A}_j^{\mathrm{P}}$ aller Schnipsel einer Seite $j \in \mathcal{P}$ ist größer als die ursprüngliche Fläche der Seite.

$$\sum_{i \in \mathcal{S}} \mathrm{A}_i^{\mathrm{S}} \cdot y_{i,j} = \mathrm{A}_j^{\mathrm{P}} > \mathrm{A}_{origin}^{\mathrm{P}} \qquad \forall j \in \mathcal{P} \tag{5.23}$$

- Die Summe der Außenkanten $\mathrm{L}_j^{\mathrm{S}}$ aller Schnipsel einer Seite $j \in \mathcal{P}$ ist größer als der ursprüngliche Umfang P_{origin}^{P} der Seite.

$$\sum_{i \in \mathcal{S}} \mathrm{L}_i^{\mathrm{S}} \cdot y_{i,j} = \mathrm{L}_j^{\mathrm{P}} > P_{origin}^{\mathrm{P}} \qquad \forall j \in \mathcal{P} \tag{5.24}$$

- Die Winkelsumme α_j^{P} ist ein Wert, der in der ursprünglichen Seiten nicht vorkommt. Er entsteht erst durch das Zerreißen und ist abhängig von der Anzahl der Außenschnipsel auf einer Seite. Jedoch haben auch hier empirische Tests ergeben, dass sich die Winkelsumme durch den Schereffekt vergrößert.

$$\sum_{i \in \mathcal{S}} \alpha_i^{\mathrm{S}} \cdot y_{i,j} = \alpha_j^{\mathrm{P}} > \alpha_{ideal_j}^{\mathrm{P}} \qquad \forall j \in \mathcal{P} \tag{5.25}$$

  mit

$$\alpha_{ideal_j}^{P} = 180° \cdot (\text{Anzahl der Außenschnipsel von } j \in \mathcal{P}\,) \tag{5.26}$$

Wegen dieser Abweichungen sind die ursprünglichen Werte einer Seite $(\mathrm{A}_{origin}^{\mathrm{P}}, \mathrm{L}_{origin}^{\mathrm{P}}, \alpha_{origin}^{\mathrm{P}})$ für das Lösungsverfahren nicht verwendbar. Stattdessen werden für eine Seite jene Werte benötigt, die aufgrund des Schereffekts zu erwarten sind, um anschließend die Schnipsel so zuzuordnen, dass sie diese sogenannten Referenzwerte annähern und somit eine möglichst gute Zuordnung zu liefern. Bei der Berechnung dieser Referenzwerte wird folgendermaßen vorgegangen:

- Zu Beginn werden die gesamten Abweichungen der Fläche, Außenkantenlänge und Winkel zu ihren idealen Werten ermittelt:

$$\mathrm{A}_{add} = \sum_{i \in \mathcal{S}} \mathrm{A}_i^{\mathrm{S}} - \mathrm{A}_{origin}^{\mathrm{P}} \cdot m \tag{5.27}$$

$$\mathrm{L}_{add} = \sum_{i \in \mathcal{S}} \mathrm{L}_i^{\mathrm{S}} - \mathrm{P}_{origin}^{\mathrm{P}} \cdot m \tag{5.28}$$

$$\alpha_{add} = \sum_{i \in \mathcal{S}} \alpha_i^{\mathrm{S}} - \sum_{j \in \mathcal{P}} \alpha_{ideal_j}^{\mathrm{P}} \tag{5.29}$$

  Mithilfe von empirischen Tests wurden die entstehenden Abweichungen untersucht und somit entdeckt, dass sich die Abweichungen je nach Seitentyp unterscheiden. Wobei

die Seiten durch die Anzahl ihrer Schnipsel $(4S, 8S, 16S, 32S)$ unterschieden werden. Die resultierenden Informationen aus den Tests sind durch die folgenden Koeffizienten dargestellt:

- Die Abweichung, der zu erwartenden Fläche von der originalen, steigt im gleichen Ausmaß wie die Anzahl der Schnipsel auf einer Seite. Bei einer doppelten Anzahl von Schnipsel ist auch die Abweichung doppelt so hoch.

$$\begin{aligned} c^{\mathrm{A}}_{4S} &= 1 \\ c^{\mathrm{A}}_{8S} &= 2 \\ c^{\mathrm{A}}_{16S} &= 3 \\ c^{\mathrm{A}}_{32S} &= 4 \end{aligned} \tag{5.30}$$

- Die Abweichung vom originalen Umfang steigt mit der Anzahl der Außenschnipsel pro Seite. Bei doppelt so vielen Außenschnipseln ist auch die Abweichung vom Umfang doppelt so hoch.

$$\begin{aligned} c^{\mathrm{L}}_{4S} &= 1 \\ c^{\mathrm{L}}_{8S} &= 2 \\ c^{\mathrm{L}}_{16S} &= 3 \\ c^{\mathrm{L}}_{32S} &= 4 \end{aligned} \tag{5.31}$$

- Auch beim Winkel tritt die Eigenschaft auf, dass mit einer steigenden Anzahl von Schnipsel die Abweichung zum originalen Wert größer wird. Hier besteht allerdings kein konstantes Wachstumsverhältnis.

$$\begin{aligned} c^{\mathrm{W}}_{4S} &= 1 \\ c^{\mathrm{W}}_{8S} &= 4 \\ c^{\mathrm{W}}_{16S} &= 7 \\ c^{\mathrm{W}}_{32S} &= 17 \end{aligned} \tag{5.32}$$

- Mithilfe der oben gegeben Koeffizienten und der berechneten Seitenverteilung (a, b, c, d), können die Zusammensetzungen der Abweichungen folgendermaßen angegeben werden:

$$\mathrm{A}_{add} = \mathrm{A}_t(a \cdot c^{\mathrm{A}}_{4S} + b \cdot c^{\mathrm{A}}_{8S} + c \cdot c^{\mathrm{A}}_{16S} + d \cdot c^{\mathrm{A}}_{32S}) \tag{5.33}$$

$$\mathrm{L}_{add} = \mathrm{L}_t(a \cdot c^{\mathrm{L}}_{4S} + b \cdot c^{\mathrm{L}}_{8S} + c \cdot c^{\mathrm{L}}_{16S} + d \cdot c^{\mathrm{L}}_{32S}) \tag{5.34}$$

$$\alpha_{add} = \alpha_t(a \cdot c^{\mathrm{W}}_{4S} + b \cdot c^{\mathrm{W}}_{8S} + c \cdot c^{\mathrm{W}}_{16S} + d \cdot c^{\mathrm{W}}_{32S}) \tag{5.35}$$

Daraus ergibt sich für die verschiedenen Abweichungsanteile:

$$\mathrm{A}_t = \frac{\mathrm{A}_{add}}{a \cdot c_{4S}^{\mathrm{A}} + b \cdot c_{8S}^{\mathrm{A}} + c \cdot c_{16S}^{\mathrm{A}} + d \cdot c_{32S}^{\mathrm{A}}} \tag{5.36}$$

$$\mathrm{L}_t = \frac{\mathrm{L}_{add}}{a \cdot c_{4S}^{\mathrm{L}} + b \cdot c_{8S}^{\mathrm{L}} + c \cdot c_{16S}^{\mathrm{L}} + d \cdot c_{32S}^{\mathrm{L}}} \tag{5.37}$$

$$\alpha_t = \frac{\alpha_{add}}{a \cdot c_{4S}^{\mathrm{W}} + b \cdot c_{8S}^{\mathrm{W}} + c \cdot c_{16S}^{\mathrm{W}} + d \cdot c_{32S}^{\mathrm{W}}} \tag{5.38}$$

- Aus den berechneten Abweichungsanteilen $\mathrm{A}_d, \mathrm{L}_d, \alpha_d$ und den ermittelten Koeffizienten (5.30) - (5.32) werden nun abschließend die Referenzwerte für die unterschiedlichen Seitentypen berechnet:

$$A_{4S}^{ref} = A_t \cdot c_{4S}^{\mathrm{A}} \qquad L_{4S}^{ref} = L_t \cdot c_{4S}^{\mathrm{L}} \qquad \alpha_{4S}^{ref} = \alpha_t \cdot c_{4S}^{\mathrm{W}} \tag{5.39}$$

$$A_{8S}^{ref} = A_t \cdot c_{8S}^{\mathrm{A}} \qquad L_{8S}^{ref} = L_t \cdot c_{8S}^{\mathrm{L}} \qquad \alpha_{8S}^{ref} = \alpha_t \cdot c_{8S}^{\mathrm{W}} \tag{5.40}$$

$$A_{16S}^{ref} = A_t \cdot c_{16S}^{\mathrm{A}} \qquad L_{16S}^{ref} = L_t \cdot c_{16S}^{\mathrm{L}} \qquad \alpha_{16S}^{ref} = \alpha_t \cdot c_{16S}^{\mathrm{W}} \tag{5.41}$$

$$A_{32S}^{ref} = A_t \cdot c_{32S}^{\mathrm{A}} \qquad L_{32S}^{ref} = L_t \cdot c_{32S}^{\mathrm{L}} \qquad \alpha_{32S}^{ref} = \alpha_t \cdot c_{32S}^{\mathrm{W}} \tag{5.42}$$

In weiterer Folge wird versucht eine Zuordnung der Schnipsel zu den Seiten zu finden, sodass diese Referenzwerte, für die verschiedenen Seitentypen, so gut wie möglich angenähert werden. Im nächsten Schritt wird dazu eine Startlösung für das TTP erzeugt.

5.2.2 Startlösung

Zuerst werden die Bedingungen erklärt, die für die Zuordnung der Schnipsel zu den Seiten erfüllt sein müssen, damit es sich um eine gültige Lösung für das *Tearing Paper Problem* handelt. Im Anschluss daran wird die Generierung von zwei unterschiedlichen Startlösungen für das TPP beschrieben.

Damit es sich bei einer Zuordnung der Schnipsel zu den Seiten um eine gültige Lösung handelt, müssen die folgenden Bedingungen erfüllt sein:

- Den Seiten müssen, entsprechend der berechneten Seitenverteilung, entweder 4, 8, 16 oder 32 Schnipsel zugeordnet werden:

$$\sum_{i \in \mathcal{S}} y_{i,j} = \begin{cases} 4 & \text{wenn } j = \text{Seite mit 2 Rissen} \\ 8 & \text{wenn } j = \text{Seite mit 3 Rissen} \\ 16 & \text{wenn } j = \text{Seite mit 4 Rissen} \\ 32 & \text{wenn } j = \text{Seite mit 5 Rissen} \end{cases} \tag{5.43}$$

- Jeder Seite müssen genau vier Eckschnipsel zugeordnet werden [1]:

$$\sum_{i \in \mathcal{C}} y_{i,j} = 4 \qquad \forall j \in \mathcal{P} \tag{5.44}$$

- Je nachdem um welchen Seitentyp es sich handelt, muss eine unterschiedliche Anzahl von Randschnipsel zugeordnet werden:

$$\sum_{i \in \mathcal{B}} y_{i,j} = \begin{cases} 0 & \text{wenn } j = \text{Seite mit 2 Rissen} \\ 4 & \text{wenn } j = \text{Seite mit 3 Rissen} \\ 8 & \text{wenn } j = \text{Seite mit 4 Rissen} \\ 16 & \text{wenn } j = \text{Seite mit 5 Rissen} \end{cases} \tag{5.45}$$

- Dem Seitentyp entsprechend muss auch die Anzahl der Innenschnipsel zugeordnet werden:

$$\sum_{i \in \mathcal{I}} y_{i,j} = \begin{cases} 0 & \text{wenn } j = \text{Seite mit 2 Rissen} \\ 0 & \text{wenn } j = \text{Seite mit 3 Rissen} \\ 4 & \text{wenn } j = \text{Seite mit 4 Rissen} \\ 12 & \text{wenn } j = \text{Seite mit 5 Rissen} \end{cases} \tag{5.46}$$

Die Generierung einer Startlösung kann auf zwei unterschiedliche Methoden durchgeführt werden: *zufällig* oder *deterministisch*. Während bei ersterer eine zufällige Zuteilung von Schnipsel zu Seiten, unter der Einhaltung der Regeln (5.43) - (5.46), vorgenommen wird, erfolgt diese bei der zweiten Methode nach bestimmten Vorgaben.

Hier wird von der Überlegung ausgegangen, dass im Normalfall größere Schnipsel eher von Seiten stammen, die nicht so oft zerrissen wurden. Schnipsel mit einer kleineren Fläche hingegen, werden eher von Seiten stammen, die öfter zerrissen wurden. Aus diesem Grund werden die verschiedenen Schnipseltypen (*corner, border, inner*) vom Schnipsel mit der größten Fläche bis zum Schnipsel mit der kleinsten Fläche durchlaufen. Wobei die größeren Schnipsel den Seiten mit weniger Rissen und die kleineren Schnipsel den Seiten mit mehr Rissen, unter Einhaltung der Regeln (5.43) - (5.46), zugeordnet werden. Im Algorithmus 5.1 ist das Zuordnungsverfahren der Schnipsel zu den Seiten dargestellt.

Nachdem nun die Startlösung erzeugt wurde, werden als nächstes drei Methoden vorgestellt, die diese Lösung verbessern: *Lokale Suche, Variable Neighborhood Descent* und eine *Hybridmethode*

[1] Muss wegen der in Abschnitt 2.1 getroffenen Annahme erfüllt sein.

Algorithmus 5.1 : Deterministische-Startlösung

Beginn
- **für jedes** *Eckschnipsel (beginnend beim größten)* **tue**
  - beginne beim Seitentyp mit 2 Rissen;
  - **wenn** *alle Seiten dieses Typs* (5.44) *erfüllen* **dann**
    - verwende Seitentyp mit der nächst höheren Risszahl;
  - **sonst**
    - ordne das Schnipsel abwechselnd einer Seite des Typs zu, die (5.44) nicht erfüllt;
- **für jedes** *Randschnipsel (beginnend beim größten)* **tue**
  - beginne beim Seitentyp mit 3 Rissen;
  - **wenn** *alle Seiten dieses Typs* (5.45) *erfüllen* **dann**
    - verwende Seitentyp mit der nächst höheren Risszahl;
  - **sonst**
    - ordne das Schnipsel abwechselnd einer Seite des Typs zu, die (5.45) nicht erfüllt;
- **für jedes** *Innenschnipsel (beginnend beim größten)* **tue**
  - beginne beim Seitentyp mit 4 Rissen;
  - **wenn** *alle Seiten dieses Typs* (5.46) *erfüllen* **dann**
    - verwende Seitentyp mit der nächst höheren Risszahl;
  - **sonst**
    - ordne das Schnipsel abwechselnd einer Seite des Typs zu, die (5.46) nicht erfüllt;

Ende

5.2.3 Lokale Suche

Um von allen gültigen Lösungen für das TPP jene zu erhalten, bei der die Zuordnung der Schnipsel zu den Seiten so gut wie möglich mit den originalen Seiten übereinstimmt, wird zuerst eine Zielfunktion definiert.

Zielfunktion

Ziel ist es, jene Zuordnung von Schnipseln zu den Seiten zu finden, bei der die Abweichungen von den Seiteneigenschaften (Fläche, Umfang, Winkel) zu ihren vorher berechneten Referenzwerten minimal ist.

Die Abweichungen der Seiteneigenschaften zu den Referenzwerten werden wie folgt ermittelt:

$$\Delta_j^{\mathrm{A}} = |\mathrm{A}^{\mathrm{P}}{}_j - \mathrm{A}_j^{ref}| \qquad \forall j \in \mathcal{P} \tag{5.47}$$

$$\Delta_j^{\mathrm{L}} = |\mathrm{L}^{\mathrm{P}}{}_j - \mathrm{L}_j^{ref}| \qquad \forall j \in \mathcal{P} \tag{5.48}$$

$$\Delta_j^{\mathrm{W}} = |\alpha_j^{P} - \alpha_j^{ref}| \qquad \forall j \in \mathcal{P} \tag{5.49}$$

Dadurch erhält man als Zielfunktion:

$$\text{minimiere} \sum_{j \in \mathcal{P}} (\Delta_j^{\mathrm{A}} + \Delta_j^{\mathrm{L}} + \Delta_j^{\mathrm{W}}) \tag{5.50}$$

Die grundlegende Funktionsweise der *Lokalen Suche* wurden schon in Abschnitt 4.2 erklärt. Bei der hier vorgestellten Methode erfolgt die schrittweise Verbesserung der zuvor generierten Startlösung durch austauschen (*swap*) von einzelnen Schnipsel zwischen unterschiedlichen Seiten, um somit einen besseren Zielfunktionswert zu erhalten. Es kann zwischen drei unterschiedlich großen Nachbarschaften ausgewählt werden: $\mathcal{N}_{snip}, \mathcal{N}_{page}, \mathcal{N}_{best}$.

Nachbarschaften

$\mathcal{N}_{\mathbf{snip}}$: Diese Nachbarschaft besteht aus allen Austauschmöglichkeiten, die für ein zufällig ausgewähltes Schnipsel S_a mit einem anderen Schnipsel $S_b \in \mathcal{S}$ existieren. Wobei das Schnipsel S_b denselben Schnipseltyp wie S_a besitzen muss, aber nicht derselben Seiten zugeordnet sein darf. Die Größe der Nachbarschaft liegt in $O(n)$.

$$\begin{aligned} \mathcal{N}_{snip} = \{(S_a, S_b) \mid\ & S_a \text{ zufällig gewählt aus } \mathcal{S},\ S_b \in \mathcal{S}, \\ & S_a \text{ und } S_b \text{ unterschiedlichen Seiten zugeordnet}, \\ & S_a.type = S_b.type\ \} \end{aligned} \tag{5.51}$$

$\mathcal{N}_{\mathbf{page}}$**:** Beinhaltet die Austauschmöglichkeiten für alle Schnipsel einer zufällig ausgewählten Seite $P_c \in \mathcal{P}$. Auch hier müssen die jeweiligen Schnipsel von einem Paar denselben Typ besitzen und unterschiedlichen Seiten zugeordnet sein. In den meisten Fälle besitzt diese Nachbarschaft ein Größenordnung von $O(n)$, kann aber im ungünstigsten Fall auch $O(n^2)$ erreichen.

$$\begin{aligned}\mathcal{N}_{page} = \{(S_a, S_b) \mid\ & S_a \text{ ist der zufällig gewählten Seite } P_c \text{ zugeordnet,} \\ & S_a \text{ und } S_b \text{ unterschiedlichen Seiten zugeordnet,} \\ & S_a.type = S_b.type,\ S_a \in \mathcal{S},\ S_b \in \mathcal{S},\ P_c \in \mathcal{P}\ \}\end{aligned} \tag{5.52}$$

$\mathcal{N}_{\mathbf{best}}$**:** Die dritte Nachbarschaft enthält alle möglichen Kombinationen von Schnipselvertauschungen, wobei die Schnipselpaare dieselben Bedingungen erfüllen müssen wie bei den anderen Nachbarschaften. Die Größen der Nachbarschaft liegt in $O(n^2)$.

$$\begin{aligned}\mathcal{N}_{best} = \{(S_a, S_b) \mid\ & S_a \in \mathcal{S},\ S_b \in \mathcal{S}, \\ & S_a \text{ und } S_b \text{ unterschiedlichen Seiten zugeordnet,} \\ & S_a.type = S_b.type\ \}\end{aligned} \tag{5.53}$$

Aufgrund der Tatsache, dass für die Vertauschung eines Schnipsels nur jene mit gleichen Typ in Frage kommen, befinden sich alle Schnipsel für eine mögliche Vertauschung, entweder im *CornerTree, BorderTree* oder *InnerTree*. Dadurch müssen nicht alle Schnipsel betrachtet werden, was zu einer Zeitersparnis beim Suchen führt.

Wie die jeweilige Nachbarschaft durchsucht wird und welches Schnipselpaar für den Austausch verwendet werden soll, wird durch die entsprechende Schrittfunktion bestimmt.

Schrittfunktionen

Die Nachbarschaften können nach drei verschiedenen Möglichkeiten durchsucht werden. Je nachdem, welche der Seiteneigenschaften $(\mathrm{A}^{\mathrm{P}}, \mathrm{L}^{\mathrm{P}}, \alpha^{\mathrm{P}})$ verbessert werden soll, wird in der Nachbarschaft das Schnipselpaar gesucht, durch dessen Vertauschung der gewünschte Wert die größte Verbesserung erfährt. Gleichzeitig wird bei der Auswahl des Paares auch darauf geachtet, dass die beiden anderen Seiteneigenschaften nicht zu sehr verschlechtert werden. Aus diesem Grund ist die Berechnung der Änderungen aller Eigenschaften, die durch die Vertauschung entstehen, notwendig.

Nachfolgenden ist die Berechnung der Flächenänderung f_{imp}^{A}, die durch die Vertauschung der Schnipsel (S_a, S_b) entsteht, beschrieben. Die Berechnungen für die Änderungen der Außenkantenlängen f_{imp}^{L} und Winkeln f_{imp}^{W} erfolgt auf die gleiche Weise:

Zuerst werden die beiden neuen Seitenflächen der durch die Vertauschung von (S_a, S_b) betroffenen Seiten P_a und P_b ermittelt.

$$\mathrm{A}^{\mathrm{P}}_{a_{new}} = \mathrm{A}^{\mathrm{P}}_{a} - \mathrm{A}^{\mathrm{S}}_{a} + \mathrm{A}^{\mathrm{S}}_{b} \tag{5.54}$$

$$\mathrm{A}^{\mathrm{P}}_{b_{new}} = \mathrm{A}^{\mathrm{P}}_{b} - \mathrm{A}^{\mathrm{S}}_{b} + \mathrm{A}^{\mathrm{S}}_{a} \tag{5.55}$$

Anschließend werden die prozentuellen Abweichungen der Seitenflächen, vor und nach der Vertauschung, zu ihren Referenzwerten berechnet.

$$\delta_{a_{old}} = 100 \cdot (\frac{\mathrm{A}^{\mathrm{P}}_{a}}{\mathrm{A}^{ref}_{a}} - 1) \tag{5.56}$$

$$\delta_{b_{old}} = 100 \cdot (\frac{\mathrm{A}^{\mathrm{P}}_{b}}{\mathrm{A}^{ref}_{b}} - 1) \tag{5.57}$$

$$\delta_{a_{new}} = 100 \cdot (\frac{\mathrm{A}^{\mathrm{P}}_{a_{new}}}{\mathrm{A}^{ref}_{a}} - 1) \tag{5.58}$$

$$\delta_{b_{new}} = 100 \cdot (\frac{\mathrm{A}^{\mathrm{P}}_{b_{new}}}{\mathrm{A}^{ref}_{b}} - 1) \tag{5.59}$$

Mithilfe dieser Abweichungen wird abschließend ermittelt, um wie viel Prozent sich die Flächen der betroffenen Seiten (P_a, P_b) an ihre Referenzflächen annähern.

$$f^{\mathrm{A}}_{imp}(S_a, S_b) = (|\delta_{a_{old}}| - |\delta_{a_{new}}|) + (|\delta_{b_{old}}| - |\delta_{b_{new}}|) \tag{5.60}$$

Die berechneten Änderungen der Seiteneigenschaften werden verwendet, um in der Nachbarschaft $\mathcal{N}$ nach jenem Schnipselpaar zu suchen, durch dessen Vertauschung die größtmögliche Verbesserung für die gewünschte Seiteneigenschaft entsteht und gleichzeitig die beiden anderen Seiteneigenschaften keine Verschlechterung aufweisen, die größer als ein vorgegebener Wert (max_{worse}) ist. Dadurch ergeben sich für die *Lokale Suche* drei unterschiedliche Schrittfunktionen: $f^{\mathrm{A}}_{step}, f^{\mathrm{L}}_{step}, f^{\mathrm{W}}_{step}$:

$$\begin{aligned} f^{\mathrm{A}}_{step} : \mathcal{N} \rightarrow (S_a, S_b) \quad \text{mit:} \quad & f^{\mathrm{A}}_{imp}(S_a, S_b) > 0 \ \wedge \ f^{\mathrm{A}}_{imp}(S_a, S_b) \text{ ist maximal} \ \wedge \\ & f^{\mathrm{L}}_{imp}(S_a, S_b) \geq -max_{worse} \ \wedge \ f^{\mathrm{W}}_{imp}(S_a, S_b) \geq -max_{worse} \end{aligned} \tag{5.61}$$

$$\begin{aligned} f^{\mathrm{L}}_{step} : \mathcal{N} \rightarrow (S_a, S_b) \quad \text{mit:} \quad & f^{\mathrm{L}}_{imp}(S_a, S_b) > 0 \ \wedge \ f^{\mathrm{L}}_{imp}(S_a, S_b) \text{ ist maximal} \ \wedge \\ & f^{\mathrm{A}}_{imp}(S_a, S_b) \geq -max_{worse} \ \wedge \ f^{\mathrm{W}}_{imp}(S_a, S_b) \geq -max_{worse} \end{aligned} \tag{5.62}$$

$$\begin{aligned} f^{\mathrm{A}}_{step} : \mathcal{N} \rightarrow (S_a, S_b) \quad \text{mit:} \quad & f^{\mathrm{W}}_{imp}(S_a, S_b) > 0 \ \wedge \ f^{\mathrm{W}}_{imp}(S_a, S_b) \text{ ist maximal} \ \wedge \\ & f^{\mathrm{A}}_{imp}(S_a, S_b) \geq -max_{worse} \ \wedge \ f^{\mathrm{L}}_{imp}(S_a, S_b) \geq -max_{worse} \end{aligned} \tag{5.63}$$

Abbruchkriterium

Die *Lokale Suche* terminiert, wenn entweder die maximal erlaubte Anzahl von Iterationen überschritten wird, oder wenn innerhalb einer bestimmte Anzahl von Iterationen durchgehend keine Verbesserung erfolgte.

5.2.4 Variable Neighborhood Descent

In einer weiteren Variante wird versucht die Startlösung mittels einer *Variable Neighborhood Descent* (VND) zu verbessern. Die grundlegende Funktionsweise einer VND wurde schon im Abschnitt 4.3 detailliert beschrieben.

Bei der hier beschriebenen Variante der VND, für das TPP, wird nicht zwischen verschiedenen Nachbarschaften gewechselt, sondern es werden für eine ausgewählte Nachbarschaft die oben beschriebenen Schrittfunktionen ($f^{\mathrm{A}}_{step}, f^{\mathrm{L}}_{step}, f^{\mathrm{W}}_{step}$) der Reihe nach angewandt. Diese Schrittfunktionen liefern, für dieselbe Nachbarschaft, verschiedene Nachbarlösungen, da sie die aktuelle Lösung entweder bezüglich der Fläche, Kantenlänge oder Winkel verbessern. Daher wird versucht durch die abwechselnde Anwendung der Schrittfunktionen das Ziel, einer minimalen Abweichung der Seiteneigenschaften (5.50), zu erreichen. Je nachdem welche Nachbarschaft ($\mathcal{N}_{snip}, \mathcal{N}_{page}, \mathcal{N}_{best}$) für die VND verwendet wird, spricht man von der VND-Methode: *snip, page* oder *best*.

Die jeweilige VND-Methode durchsucht die ausgewählte Nachbarschaft zuerst nach der Fläche (f^{A}_{step}), anschließend nach der Außenkantenlänge (f^{L}_{step}) und zuletzt nach dem Winkel (f^{W}_{step}). Liefert irgendeine der drei Schrittfunktionen ein geeignetes Schnipselpaar (S_a, S_b), so werden diese Schnipsel vertauscht und wieder die erste Schrittfunktion (f^{A}_{step}) verwendet. Wenn jedoch eine Schrittfunktion kein geeignetes Schnipselpaar findet, wechselt man zur nächsten und sucht mit dieser eine Verbesserung. Im Algorithmus 5.2 ist die VND für das TPP dargestellt.

Abbruchkriterium

Das Abbruchkriterium der VND ist erfüllt, wenn entweder die maximale erlaubte Anzahl der Iterationen überschritten wurde, oder von allen drei Schrittfunktionen, eine bestimmte Anzahl von Iterationen lang kein Schnipselpaar für eine Verbesserung gefunden wurde.

5.2.5 Hybrid

In dieser Methode wird versucht, durch die Kombination von einem exakten Algorithmus für das TTP und einer entsprechenden Heuristik, die Startlösung schrittweise zu verbessern. Der exakte Algorithmus wird für jede Seite separat aufgerufen und liefert ein Maß über die Qualität der zugeordneten Schnipsel. Die vom exakten Algorithmus erhaltenen Eigenschaften einer

Algorithmus 5.2 : TPP-VND

Eingabe : Startlösung l, Nachbarschaft $\mathcal{N}$

Beginn

$i \longleftarrow 1$;

wiederhole

unterscheide i **tue**

Fall *1*

$(S_a, S_b) \longleftarrow (f^{\mathrm{A}}_{step} : \mathcal{N}(l))$;

Fall *2*

$(S_a, S_b) \longleftarrow (f^{\mathrm{L}}_{step} : \mathcal{N}(l))$;

Fall *3*

$(S_a, S_b) \longleftarrow (f^{\mathrm{W}}_{step} : \mathcal{N}(l))$;

wenn (S_a, S_b) *gefunden* **dann**

$i \longleftarrow 1$;

$l \longleftarrow$ (vertausche (S_a, S_b) in l);

sonst

wenn $i == 3$ **dann**

$i \longleftarrow 1$;

bis *Abbruchkriterium erfüllt* ;

Ende

Seite versucht man anschließend, mithilfe einer Heuristik, zu verbessert. Dazu werden einzelne Schnipsel zwischen verschiedenen Seitenkanten ausgetauscht, um eine bessere Lösung zu erhalten.

Exakter Algorithmus

Es wird der von Schüller [1] entwickelte exakte Algorithmus, für die Rekonstruktion der Ränder von handzerrissenen Seiten, verwendet. Konfiguriert wird der Algorithmus mit der in Abschnitt 3 beschriebenen LILP Formulierung.

Der exakte Algorithmus liefert die bestmögliche Zuordnung aller Schnipsel i einer Seite $j \in \mathcal{P}$ zu ihren vier Seitenkanten k. Durch eine kleine Erweiterung die im exakten Algorithmus vorgenommen wurde, ist es möglich die Abweichung $\Delta_{k,j}$ jeder Seitenkante von ihrem Idealwert auszulesen. Berechnet wird die Abweichung $\Delta_{k,j}$, indem von der Summe der Außenkantenlängen L_i^S , aller einer Kante k zugeordneten Schnipsel i, die ideale Kantenlänge $\mathrm{L}^{\mathrm{E}}_{k_{ideal}}$ subtrahiert wird:

$$\Delta_{k,j} = \sum_{i \in \mathcal{S}} \mathrm{L}_i^{\mathrm{S}} \cdot x_{i,k} - \mathrm{L}^{\mathrm{E}}_{k_{ideal}} \qquad \forall j \in \mathcal{P}, k \in \{1,2,3,4\} \tag{5.64}$$

mit

$$x_{i,k} = \begin{cases} 1 & \text{wenn Schnipsel } i \text{ der Kante } k \text{ zugeordnet ist} \\ 0 & \text{sonst} \end{cases} \tag{5.65}$$

Bei der Berechnung muss darauf geachtet werden, dass eine Seite zwei kurzen Seitenkanten ($k \in \{2,4\}$) und zwei langen Seitenkanten ($k \in \{1,3\}$) besitzt, die hier gegen den Uhrzeigersinn nummeriert sind. Somit ergibt sich:

$$\mathrm{L}^{\mathrm{E}}_{k_{ideal}} = \begin{cases} \mathrm{L}^{\mathrm{P}}_s = 210\ mm & \text{wenn } k \in \{2,4\} \quad \text{(kurze Seitenkante)} \\ \mathrm{L}^{\mathrm{P}}_l = 295\ mm & \text{wenn } k \in \{1,3\} \quad \text{(lange Seitenkante)} \end{cases} \tag{5.66}$$

Ermitteln der zu Verbessernden Seitenkanten

Bei der Ermittlung der zu verbessernden Seitenkanten wird von der Annahme ausgegangen, dass Kanten, bei denen die Zuordnung der Schnipsel korrekt sind, eine Abweichung besitzen, die innerhalb einer angegebenen Toleranz liegen. Hingegen werden Kanten, bei denen die Zuordnung „wahrscheinlich“ nicht korrekt ist, eine Abweichung besitzen, die sich außerhalb dieser Toleranz befindet.

Somit werden alle Kanten $\mathrm{E}_{k,j}$ in die Menge $\mathcal{E}$ aufgenommen, deren Abweichung $\Delta_{k,j}$ entweder negativ ist, oder die maximal erlaubte Abweichung $\Delta^{max}_{k,j}$ überschreitet:

$$E_{k,j} \in \mathcal{E} \quad \text{wenn} \quad \Delta_{k,j} < 0 \ \vee \ \Delta_{k,j} > \Delta^{max}_{k,j} \qquad \forall j \in \mathcal{P},\ k \in \{1,2,3,4\} \tag{5.67}$$

Die maximal erlaubte Abweichung ist folgendermaßen gegeben:

$$\Delta_{k,j}^{max} = distance \cdot gaps_{k,i} \tag{5.68}$$

Durch die Variable *distance* wird die Länge angegeben, um wie viel sich durch einen Riss, aufgrund des Schereffekts, die Außenkantenlänge maximal vergrößern darf. Mit der unterschiedlichen Anzahl von Rissen, die für das Zerreißen eine Originalseite verwendet werden, ändert sich auch die Anzahl der Risse, mit denen die kurzen und langen Kanten einer Seite zerrissen wurden. Aus diesem Grund muss die maximal erlaubte Abweichung von der Anzahl der Risse auf einer Seitenkanten abhängen. Je mehr Risse sich auf einer Kante befinden, desto größer muss auch die Abweichung sein. Die Variable $gaps_{k,j}$ gibt dazu an, wie viel Risse auf der Kante k, von der Seite j, vorhanden sind:

$$gaps_{k,j} = \begin{cases} 1 & \text{wenn} \quad j \text{ Seite mit 2 Rissen} \ \wedge\ k \in \{1,2,3,4\} \ \vee \\ & \quad\ \ j \text{ Seite mit 3 Rissen} \ \wedge\ k \in \{2,4\} \\ 3 & \text{wenn} \quad j \text{ Seite mit 3 Rissen} \ \wedge\ k \in \{1,3\} \ \vee \\ & \quad\ \ j \text{ Seite mit 4 Rissen} \ \wedge\ k \in \{1,2,3,4\} \ \vee \\ & \quad\ \ j \text{ Seite mit 5 Rissen} \ \wedge\ k \in \{2,4\} \\ 7 & \text{wenn} \quad j \text{ Seite mit 5 Rissen} \ \wedge\ k \in \{1,3\} \end{cases} \tag{5.69}$$

Die Menge $\mathcal{E}$ enthält somit all jene Kanten, bei denen die Zuordnung der Schnipsel sozusagen nicht als *gültig* angenommen wird.

Zielfunktion

Ziel der Hybridmethode ist es, eine möglichst gute Zuordnung der Schnipsel für die einzelnen Seitenkanten zu erhalten. Sprich eine Zuordnung bei der die Anzahl der Abweichungen $\Delta_{k,j}$, die außerhalb der Toleranz liegen, minimal ist. Es soll somit die Anzahl der Kanten in $\mathcal{E}$ minimiert werden:

$$\text{Zielfunktion} = \quad \text{minimiere} \ |\mathcal{E}| \tag{5.70}$$

Um dieses Ziel zu erreichen, werden Schnipsel zwischen unterschiedlichen, nicht *gültigen* Kanten ausgetauscht (*swap*), sodass die Abweichungen der Seitenkanten verringert werden. Dazu werden zwei unterschiedliche großen Nachbarschaften definiert: $\mathcal{N}^{\mathrm{H}}_{snip}, \mathcal{N}^{\mathrm{H}}_{edge}$.

Nachbarschaft

$\mathcal{N}^{\mathrm{H}}_{\mathbf{snip}}$ Die Nachbarschaft $\mathcal{N}^{\mathrm{H}}_{snip}$ beinhaltet alle Vertauschungen (Schnipselpaare (S_a, S_b)), die für eine bestimmte Anzahl (g) von zufällig gewählten Schnipsel möglich sind. Wobei für jedes Schnipselpaar folgende Bedingungen erfüllt sein müssen: Die Schnipsel müssen denselben Schnipseltyp besitzen aber unterschiedlichen Seiten zugeordnet sein. Weiters müssen beide Kanten E_a und E_b, auf welche die Schnipsel zugeordnet sind, in der Menge $\mathcal{E}$, der nicht *gültigen* Kanten, enthalten sein. Je nach Anzahl der zufällig gewählten Schnipsel liegt die Größe der Nachbarschaft entweder in $O(n)$ oder $O(n^2)$.

$$\begin{aligned}\mathcal{N}^{\mathrm{H}}_{snip} = \{(S_a, S_b) \mid\ & S_a \text{ eines von } g \text{ zufällig gewählten Schnipsel},\\ & S_a \text{ und } S_b \text{ unterschiedlichen Seiten zugeordnet},\\ & S_a \in E_a,\ S_b \in E_b,\ E_a \text{ und } E_b \in \mathcal{E},\\ & S_a.type = S_b.type,\ S_b \in \mathcal{S}\}\end{aligned} \tag{5.71}$$

$\mathcal{N}^{\mathrm{H}}_{\mathbf{page}}$ Die Nachbarschaft $\mathcal{N}^{\mathrm{H}}_{page}$ beinhaltete alle Austauschmöglichkeiten für jene Schnipsel, die einer bestimmten Anzahl (g) von, zufällig ausgewählten, Seitenkanten zugeordnet sind. Wobei für die Schnipselpaare dieselben Bedingungen erfüllt sein müssen, wie es bei $\mathcal{N}^{\mathrm{H}}_{snip}$ der Fall ist. Abhängig von der Anzahl der zufällig ausgewählten Kanten besitzt die Nachbarschaft eine Größenordnung, die in $O(n)$ oder $O(n^2)$ liegt.

$$\begin{aligned}\mathcal{N}^{\mathrm{H}}_{page} = \{(S_a, S_b) \mid\ & S_a \text{ einer von } g \text{ zufällig gewählten Seitenkanten zugeordnet},\\ & S_a \text{ und } S_b \text{ unterschiedlichen Seiten zugeordnet},\\ & S_a \in E_a,\ S_b \in E_b,\ E_a \text{ und } E_b \in \mathcal{E},\\ & S_a.type = S_b.type,\ S_b \in \mathcal{S}\}\end{aligned} \tag{5.72}$$

Schrittfunktion

Es wird in den Nachbarschaften nach jenem Schnipselpaar durchsucht, welches für die durch den Austausch betroffenen Kanten die größte Verbesserung liefert.

Dazu berechnet die Funktion $f_{hybrid}(S_a, S_b)$ die Änderung, um wie viel sich, durch den Austausch der Schnipsel S_a und S_b, die Abweichung $\Delta_{k,j}$ der Kante k, mit $S_a \in k$, in Richtung erlaubten Toleranzbereich bewegt.

Bei der Berechnung der Änderung wird zunächst die neue Abweichung $\Delta^{new}_{k,j}$ berechnet, die nach dem Austausch der Schnipsel S_a und S_b für die Kante k zu erwarten ist:

$$\Delta^{new}_{k,j} = \Delta_{k,j} - S_a.edge + S_b.edge \tag{5.73}$$

Anschließend werden die Differenzen Δ^{old}_{tol} und Δ^{new}_{tol} ermittelt, die den Abstand der Kantenabweichungen, vor und nach einer möglichen Vertauschung, zur erlaubten Toleranz angeben:

$$\Delta^{old}_{tol} = \begin{cases} |\Delta_{k,j}| & wenn \quad \Delta_{k,j} < 0 \\ \Delta_{k,j} - \Delta^{max}_{k,j} & wenn \quad \Delta_{k,j} \geq 0 \end{cases} \tag{5.74}$$

$$\Delta^{new}_{tol} = \begin{cases} |\Delta^{new}_{k,j}| & wenn \quad \Delta^{new}_{k,j} < 0 \\ \Delta^{new}_{k,j} - \Delta^{max}_{k,j} & wenn \quad \Delta^{new}_{k,j} \geq 0 \end{cases} \tag{5.75}$$

Als Resultat liefert $f_{hybrid}(S_a, S_b)$ die Differenz der beiden Toleranzabweichungen Δ_{tol}^{old} und Δ_{tol}^{new}:

$$f_{hybrid}(S_a, S_b) = \Delta_{tol}^{old} - \Delta_{tol}^{new} \tag{5.76}$$

Zu erwähnen ist, dass bei der hier beschriebenen Bewertungsfunktion von einer *angenommenen* Veränderung der beteiligten Kantenlängen ausgegangen wird, die durch den Austausch der beiden Schnipsel zwischen den Kanten entsteht. *Angenommene* Veränderung aus dem Grund, weil nicht gewährleistet ist, dass nach der neuerlichen Anwendung des exakten Algorithmus auf die geänderten Seiten, dieser wieder dieselbe Zuordnung der Schnipsel zu den Kanten liefert, wie zuvor. Es kann durchaus möglich sein, dass der Algorithmus eine gänzlich unterschiedliche Zuordnung und somit auch unterschiedliche Kantenabweichungen liefert. Daher ist auch eine inkrementelle Anwendung der Zielfunktion nicht möglich.

Für die Auswahl der gewünschten Schnipsel wird die entsprechende Nachbarschaft nach folgender Regel durchlaufen: Es werden nur Schnipselpaare (S_a, S_b) betrachtet bei denen, durch eine mögliche Vertauschung, die Verbesserung für die Kante E_a größer als null ist und die zweite Kante E_b keine Verschlechterung aufweist, die größer als ein gegebener Wert ($worse_{hybrid}$) ist.

$$f_{hybrid}(S_a, S_b) > 0 \tag{5.77}$$
$$f_{hybrid}(S_b, S_a) \geq worse_{hybrid} \tag{5.78}$$

Das Schnipselpaar (S_a, S_b) mit der größten Verbesserung, wird mithilfe der folgenden vier Fallunterscheidungen ermittelt.

1. Wenn sich durch einen möglichen Austausch von (S_a, S_b) beide Abweichungen, $\Delta_{E_a}^{new}$ und $\Delta_{E_b}^{new}$, innerhalb der Toleranz befinden, wird die Suche abgebrochen und dieses Paar verwendet. In diesem Fall wird angenommen, dass durch diese Vertauschung beiden Kanten nur mehr korrekte Schnipsel zugeordnet sind und somit keine noch größere Verbesserung gefunden werden kann.

2. Befindet sich für mehrere Schnipselpaare die Abweichung $\Delta_{E_a}^{new}$ innerhalb der Toleranz, so wird jenes Paar verwendet, bei dem die Verbesserung für die Kante E_b am größten ist ($max(f_{hybrid}(S_b, S_a))$).

3. Beim Vergleich eines Paares, bei dem die Abweichung $\Delta_{E_a}^{new}$ innerhalb der Toleranz liegt, mit einem Paar bei dem sich, nach einem möglichen Austausch, keine Abweichung innerhalb der Toleranz befindet, wird ersteres verwendet.

4. Sind für mehrere Paare beide Abweichungen außerhalb der erlaubten Toleranz, so verwendet man jenes Paar, dass für die Kante E_a die größte Verbesserung ($f_{hybrid}(S_a, S_b)$ ist maximal) aufweist.

Algorithmus 5.3 : TPP-Hybrid

Eingabe : Startlösung l

Beginn

```
    E ⟵ alle Kanten, deren Abweichungen außerhalb des Toleranzbereichs liegen;
    unterscheide Hybridvariante tue
        Fall snipsingle
            snipSingle(E);
        Fall snipall
            snipAll(E);
        Fall edgesingle
            edgeSingle(E);
        Fall edgeall
            edgeAll(E);
```

Ende

In der Hybridmethode wird daher zuerst, mithilfe des exakten Algorithmus, die Menge aller Kanten $\mathcal{E}$ berechnet, deren Zuordnung nicht *gültig* ist. Mithilfe der oben beschriebenen Nachbarschaften und Schrittfunktionen wird anschließend versucht die Abweichungen der Kanten durch Vertauschen von Schnipsel zwischen ihnen zu verringern. Es werden vier Varianten unterschieden, wie der Austausch der Schnipsel und somit auch die Verbesserung der Abweichungen durchgeführt werden soll: *snipSingle, snipAll, edgeSingle, edgeAll*. Im Algorithmus 5.3 ist die Hybridmethode für das TPP dargestellt.

snipsingle

Bei der *snipsingle* Variante wird zuerst aus der Nachbarschaft $\mathcal{N}^{\mathrm{H}}_{snip}$ ein entsprechendes Schnipselpaar (S_a, S_b) zum Vertauschen gesucht. Wird eines gefunden, so werden die Schnipsel vertauscht und anschließend mithilfe des exakten Algorithmus die neuen Kantenabweichungen, für die vom Austausch betroffenen Seiten, ermittelt. Abschließend wird überprüft, ob die neu berechneten Kantenabweichungen nun innerhalb der gewünschten Toleranz liegen. Dementsprechend wird die Menge $\mathcal{E}$ aktualisiert und die Kanten entweder aus $\mathcal{E}$ entfernt oder eingefügt. Im Algorithmus 5.4 ist die Hybridvariante *snipsinlge* dargestellt.

Der oben beschriebene Vorgang wird solange wiederholt, solange in der Menge $\mathcal{E}$ zwei Kanten von unterschiedlichen Seiten enthalten sind, die maximal erlaubte Anzahl von Iterationen nicht überschritten wurde und in einer bestimmten Anzahl vorheriger Iterationen eine Verbesserung (Vertauschung) durchgeführt wurde.

edgeSingle

Bei *edgeSingle* Variante wird abweichend zur *snipSingle* Variante die Nachbarschaft $\mathcal{N}^{\mathrm{H}}_{edge}$ verwendet. Sonst unterscheiden sich die beiden Varianten in ihrer Funktionsweise nicht.

Algorithmus 5.4 : snipSingle

Eingabe : Startlösung l, Menge $\mathcal{E}$ der nicht *gültigen* Kanten

Beginn

  solange *Abbruchkriterium nicht erfüllt* **tue**

    $(S_a, S_b) \longleftarrow f^{hybrid}(N^{\mathrm{H}}_{snip}(l))$;

    wenn (S_a, S_b) *gefunden* **dann**

      vertausche (S_a, S_b);

      berechne mit dem exakten Algorithmus die neuen Kantenabweichungen der vom Austausch betroffenen Seiten;

      aktualisiere die Menge $\mathcal{E}$;

Ende

snipAll

Wie auch bei der *snipSingle* Variante wird zuerst in der Nachbarschaft ${}^{\mathrm{H}}N_{snip}$ nach dem Schnipselpaar (S_a, S_b) zum Vertauschen gesucht, welches die größte Verbesserung liefert. Wird hier ein Paar gefunden, werden diese Schnipsel vertauscht und die Kanten, denen die beiden Schnipsel zugeordnet sind, aus der Menge $\mathcal{E}$ entfernt. Im Gegensatz zu der *snipSingle* Variante wird hier nicht sofort der exakte Algorithmus aufgerufen, sondern es wird solange in der Menge $\mathcal{E}$ nach weiteren Schnipselpaaren zum Tauschen gesucht, solange in $\mathcal{E}$ mindestens zwei Kanten von unterschiedlichen Seiten enthalten sind und in einer bestimmten Anzahl vorheriger Suchvorgänge eine Verbesserung gefunden wurde.

Nachdem in der Menge $\mathcal{E}$ keine weiteren Austauschmöglichkeiten mehr gefunden wurden, wird anschließend für alle Seiten, die von einer Vertauschung betroffen waren, der exakte Algorithmus aufgerufen. Dadurch erhält man die neuen Schnipselzuordnungen und Kantenabweichungen. Abschließend wird überprüft, ob sich neuen Kantenabweichungen nun innerhalb der erlaubten Toleranz befinden und gegebenenfalls wird die Menge $\mathcal{E}$ aktualisiert. Im Algorithmus 5.5 ist die *snipall* Variante der Hybridmethode zu sehen.

Wiederholt wird der gesamte Vorgang, solange die maximal erlaubte Anzahl von Iterationen nicht erreicht wurde, in der Menge $\mathcal{E}$ Kanten von unterschiedlichen Seiten enthalten sind und in einer bestimmten Anzahl vorheriger Iterationen mindestens eine Vertauschung durchgeführt wurde.

edgeAll

Die *edgeAll* Variante funktioniert analog zur *snipAll* Variante, basiert jedoch auf der Nachbarschaft $\mathcal{N}^{\mathrm{H}}_{edge}$.

Algorithmus 5.5 : snipAll

Eingabe : Startlösung l, Menge $\mathcal{E}$ der nicht *gültigen* Kanten

Beginn

solange *Abbruchkriterium nicht erfüllt* **tue**

solange $\mathcal{E}$ *Kanten von verschiedenen Seiten enthält und in einer bestimmte Anzahl vorheriger Suchvorgänge eine Verbesserung gefunden wurde* **tue**

$(S_a, S_b) \longleftarrow f^{hybrid}(\mathfrak{N}^{\mathrm{H}}_{snip}(l))$;

wenn (S_a, S_b) *gefunden* **dann**

vertausche (S_a, S_b);

entferne die Kanten, denen die Schnipsel S_a und S_b zugeordnet sind aus $\mathcal{E}$;

wenn *Vertauschungen durchgeführt wurden* **dann**

berechne mit dem exakten Algorithmus die neuen Kantenabweichungen der von den Vertauschungen betroffenen Seiten;

aktualisiere die Menge $\mathcal{E}$;

Ende

5.2.6 Analyse der Lösung

Nachdem der Algorithmus eine entsprechende Zuordnung der Schnipsel zu den Seiten berechnet hat, wird abschließend für Testzwecke diese Lösung bewertet, um eine Aussage über ihre Qualität treffen zu können.

Zu diesem Zweck wollen wir zuerst die erzeugten Lösungsseiten so den originalen Seiten zugeordnet, dass die Anzahl der korrekt zugeordneten Schnipsel einer Seite maximal ist. Dazu wird zunächst ermittelt, wie viele Schnipsel sich von welcher originalen Seite auf einer Lösungsseite befinden. Da für die in dieser Arbeit verwendeten Instanzen die optimalen Lösungen bekannt sind, kann die Verteilung der Schnipsel mithilfe ihrer ID ermittelt werden. In Tabelle 5.3 ist ein Beispiel einer möglichen Schnipselverteilung von vier Seiten dargestellt.

	L1	L2	L3	L4
O1	1	1	2	0
O2	1	1	1	1
O3	2	1	1	4
O4	0	5	0	3

Tabelle 5.3: Beispiel einer Schnipselverteilung

Wobei $L1$-$L4$ die vom Algorithmus erzeugten Lösungsseiten bezeichnen und $O1$-$O4$ jene Seiten beschreiben, zu denen die Schnipsel vor dem Zerreißen gehört haben. Somit sind im angegebenen Beispiel der Lösungsseite $L1$ ein Schnipsel von der originalen Seite $O1$, eines von

Algorithmus 5.6 : Ungarische-Methode

Eingabe : $n \times n$ Matrix

Beginn

subtrahiere von jedem Element in einer Zeile das Zeilenminimum;
subtrahiere von jedem Element in einer Spalte das Spaltenminimum;
solange *true* **tue**
finde minimale Anzahl von Linien, mit welchen sämtliche Nullen der Matrix gestrichen werden können;
wenn *Anzahl der Linien* = n **dann Abbruch**;
sonst
finde kleinstes Element, dass nicht durchgestrichen ist;
subtrahiere es von allen Einträgen die nicht durchgestrichen sind;
addiere es zu allen Einträgen die zweimal durchgestrichen sind;
Suche eine Kombination von Nullen bei der in jeder Zeile und in jeder Spalte nur eine Null ausgewählt ist;

Ende

$O2$ und zwei von $O3$ zugeordnet.

Für die Lösung des hier gegebenen linearen Zuordnungsproblems wird die *Ungarische Methode* [16] verwendet. Es existieren zwar Methoden die effizienter sind, jedoch zur Analyse der Lösung ist diese Methode ausreichend. Die Funktionsweise der *Ungarische Methode* ist im Algorithmus 5.6 dargestellt:

Bevor man jedoch die *Ungarische Methode* anwenden kann, muss das gegebene Maximierungsproblem noch in ein äquivalentes Minimierungsproblem umgewandelt werden. Dafür wird die prozentuelle Verteilung der Schnipsel auf den Seiten ermittelt und die jeweiligen Werte von 100% (Maximalwert) subtrahiert. Für unsere Schnipselverteilung ergeben sich dadurch die in Tabelle 5.4 angegeben Werte.

	L1	L2	L3	L4
O1	75	75	50	100
O2	75	75	75	75
O3	75	87,5	87,5	50
O4	100	37,5	100	62,5

Tabelle 5.4: Minimierungsproblem

Die *Ungarische Methode* liefert, für diese Matrix, eine Kombination von Nullen, bei der in jeder Zeile und in jeder Spalte nur eine Null ausgewählt ist. Die Plätze der Nullen in dieser Kombination repräsentieren die optimale Zuordnung. Für unser Beispiel ergibt sich dadurch

folgende Zuordnung: $O1 \leftarrow L3, O2 \leftarrow L1, O3 \leftarrow L4, O4 \leftarrow L2$.

Nachdem man die Zuordnung ermittelt hat, bei der die Anzahl der korrekt zugeordneten Schnipsel maximal ist, wird anschließend die Qualität der einzelnen Seiten bestimmt. Wobei die Qualität Q_j einer Seite $j \in \mathcal{P}$ angibt, wie viel Prozent der Schnipsel auf der Seite richtig zugeordnet sind:

$$\mathrm{Q}_j = \frac{(\text{Anzahl der korrekt zugeordneten Schnipsel von } j)}{(\text{Anzahl der Schnipsel von } j)} \cdot 100 \qquad \forall j \in \mathcal{P} \qquad (5.79)$$

Der Mittelwert von allen Einzelbewertungen Q_j ergibt abschließend die Qualität Q_{all} der gesamten Lösung:

$$\mathrm{Q}_{all} = \frac{\sum_{j \in \mathcal{P}} \mathrm{Q}_j}{|\mathcal{P}|} \qquad (5.80)$$

Für das oben angegebene Beispiel erhält man somit die folgenden Lösungsqualitäten:

$$\mathrm{Q}_{\mathrm{L}_1} = 50\% \qquad \mathrm{Q}_{\mathrm{L}_2} = 25\% \qquad \mathrm{Q}_{\mathrm{L}_3} = 50\% \qquad \mathrm{Q}_{\mathrm{L}_4} = 62,5\%$$

Dadurch ergibt sich für die gesamte Zuordnung eine Lösungsqualität von:

$$Q_{all} = \frac{50\% + 25\% + 50\% + 62,5\%}{4} = 46,9\%$$

6 Ergebnisse

In diesem Kapitel wird die Generierung der unterschiedlichen Instanzen, sowie die Testergebnisse für den, im vorherigen Abschnitt, beschrieben Algorithmus `astp` dargestellt. Die Tests wurden durchgeführt, indem für die jeweilige Instanz zuerst eine *VND-Methode* aufgerufen wurde und anschließend eine der verschiedenen *Hybridmethoden* angewandt wurde, um eine weitere Verbesserung der Lösungen zu erhalten.

Obwohl auch Tests mit reiner *Lokalen Suche* durchgeführt wurden, werden diese Testergebnisse hier nicht weiter besprochen, da bezogen auf das *Tearing Paper Problem* keine feststellbaren Verbesserungen erzielt werden konnten.

Die bei den Testläufen verwendeten Instanzen wurden mithilfe dem von Schüller in [1] vorgestellten `simulator` und `mixer` generiert (siehe Abschnitt 3). Erzeugt wurden dabei Instanzen, die einerseits nur Seiten mit derselben Rissanzahl beinhalten und andererseits Instanzen, welche auch Seiten mit unterschiedlicher Anzahl von Rissen aufweisen.

Durchgeführt wurden die verschiedenen Tests auf einem Dual Core AMD Opteron(tm) 270 Prozessor mit 2 GHz und 8 GB RAM. Als ILP-Löser wurde das kommerzielle Softwarepaket CPLEX 10.0 verwendet.

Die Ergebnisse des Algorithmus `astp` beschreiben, wie viel Prozent der Schnipsel, durchschnittlich auf den Seiten, korrekt zugeordnet wurden. Ein Wert von 0% bedeutet, dass kein einziges Schnipsel richtig zugeordnet wurde. Im Gegensatz besagt ein Wert von 100%, dass alle Schnipsel den richtigen Seiten zugeordnet wurden und es sich somit um eine optimale Lösung handelt.

6.1 Generierung der Instanzen

Für die Generierung der Instanzen wurden der `simulator` und `mixer` verwendet. Wobei der `simulator` das Zerreißen des Papiers simuliert und der `mixer` die, vom `simulator`, erhaltenen Schnipsel durchmischt, rotiert, spiegelt und den Schereffekt und das Problem der Kantenerkennung simuliert (siehe Abschnitt 3).

Als Ergebnis der beiden Tools erhält man eine Menge von Schnipsel, deren Verhalten jenes von realen Schnipseln ähnelt und deren Anordnung keine Rückschlüsse auf die Lösung des TPP liefert.

Parameter	Relation	Generator	Werte
`paper.witdh`	absolut	fixed	295
`paper.heigth`	absolut	fixed	210
`cut.count`	absolut	fixed	{2,3,4,5}
`cut.top`	relativ	gauss	center = 0 dev = 0.1
`cut.bottom`	relativ	gauss	center = 0 dev = 0.2
`mirrow.x`	boolean	bool	50%
`mirrow.y`	boolean	bool	50%
`rotate`	absolut	gauss	center = 0 dev = 0.13
`translate.x`	absolut	gauss	center = 0 dev = 3
`translate.y`	absolut	gauss	center = 0 dev = 3
`turn.left`	boolean	bool	50%

Tabelle 6.1: Parameter für den `simulator`

Simulator

Wie eine Seite zerrissen wird, wird durch die `simulator` Parameter gesteuert. In Tabelle 6.1 sind die, für die Erzeugung, verwendeten Parameter dargestellt, die dem `simulator` über die Datei *params.xml* übergeben werden.

Der Aufruf des `simulator`s erfolgte durch den folgenden Befehl:

```
./simulator -m simulate -p params.xml -o torn.xml
```

Mixer

Die vom `simulator` erzeugten Dateien werden anschließend als Eingabedateien für den `mixer` verwendet. Aufgerufen wurde der `mixer` mit dem nachstehenden Befehl:

```
./mixer -o instance.xml -c -r -m -s -d {1,2} torn1.xml
torn2.xml ...
```

Erzeugt wurden zum Einen Instanzen, die nur Seiten mit derselben Anzahl von Rissen (2, 3, 4 oder 5) beinhalten. Die kleinsten dieser Instanzen enthalten dabei zwei Seiten und die größten zehn. Zusätzlich wurden auch Instanzen generiert, die Seiten mit einer unterschiedlichen Anzahl von Rissen beinhalten. Es werden in diesem Fall Instanzen unterschieden, die entweder Seiten mit zwei und drei Rissen enthalten, oder Instanzen, die Seiten mit zwei, drei und vier Rissen enthalten.

Bezeichnung der Instanzen

Um die in den Testergebnissen verwendeten Instanzenbezeichnung zu verstehen, sehen wir uns deren Zusammensetzung genauer an.

Die Bezeichnung der Instanzen bezieht sich auf die Seiten, die in der jeweiligen Instanz beinhaltet sind. Zuerst wird die Anzahl der Risse, gefolgt von einem `t` angegeben. Anschließend folgt die Anzahl der Seiten, die in dieser Instanz, mit der entsprechenden Rissanzahl, vorhanden sind. Wenn eine Instanz Seiten mit einer unterschiedlichen Anzahl von Rissen beinhaltet, werden die Bezeichnungen durch einen Unterstrich voneinander getrennt. Abschließend wird für jede Instanz noch der vom `mixer` verwendete *d-Parameter* angeben, indem zuerst ein Unterstrich gefolgt vom Buchstaben `d` und dem gewählten Wert, an das Ende der Instanzbezeichnung angehängt wird.

Zum Beispiel: Eine Instanz, die zwei Seiten mit zwei Rissen und vier Seiten mit drei Rissen beinhaltet und die bei der Generierung mit dem `mixer` Wert `d`=1 aufgerufen wurde, erhält somit die Bezeichnung: `2t2_3t4_d1`.

6.2 Testergebnisse

In diesem Abschnitt werden die Ergebnisse des Algorithmus `astp` für die unterschiedliche Instanzen präsentiert.

Die Durchführung der Tests erfolgte, indem zuerst für eine beliebige Instanz eine zufällige oder deterministische Startlösung erzeugt wurde. Auf die Startlösung wurde anschließend die VND angewandt, wobei hier nur die Varianten *snip* und *page* dargestellt sind, denn die *best* Variante benötigte zu viel Zeit, um eine Lösung für das TPP zu liefern. Abschließend wurde auf die Lösung der VND noch eine der vier Hybridmethoden (*edgeall*, *edgesingle*, *snipall*, *snipsingle*) angewandt, um die aktuelle Lösung weiter zu verbessern. Obwohl auch Testläufe für Instanzen, die Seiten mit fünf Rissen enthalten, durchgeführt wurden, werden diese Ergebnisse hier nicht präsentiert, da für diese Instanzen, mit den hier angewandten Verfahren, keine erkennbaren Verbesserungen erzielt wurden.

Für jede Kombination der Methoden (*Startlösung* - *VND* - *Hybrid*) wurden für eine Instanz zehn Durchläufe durchgeführt. In den Ergebnissen sind die Mittelwerte dieser zehn Durchläufe dargestellt.

Die Ergebnisse zeigen, dass die Heuristik, die für die Berechnung der Seitenverteilung in Abschnitt 5.1.4 vorgestellt wurde für alle getesteten Instanzen eine korrekte Verteilung ermittelt.

Die vom Algorithmus `astp` für die jeweiligen Instanzen verwendeten Parameter wurden durch aufwendige Tests eruiert. Eine detaillierte Beschreibung dieser Parameter ist im Anhang A aufgelistet. Die Darstellung von allen Ergebnissen, die zur Ermittlung der Parameterwerte

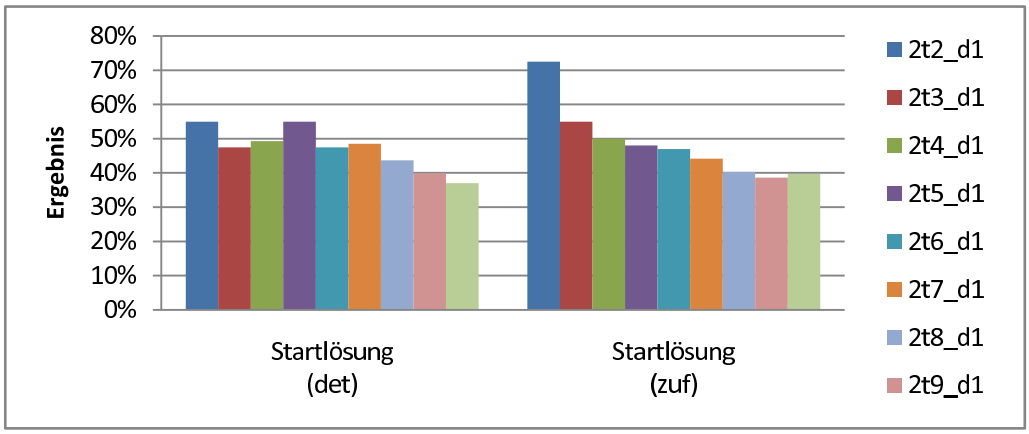


Abbildung 6.1: Startlösungen der Instanzen mit 2 Rissen

beigetragen haben, würde den Rahmen hier sprengen. Bis auf die beiden Parameter `t` und `g`, die sich für die verschiedenen Instanztypen unterscheiden, wurden die dargestellten Ergebnisse mithilfe der Parameter: `r`=35, `w`=5, `m`=10000000, `b`=30, `c`=5, `o`=10 und `x`=100000 erzeugt.

Instanzen mit zwei Rissen

Bei allen Instanzen, die nur Seiten mit zwei Rissen enthalten, wurden für die Parameter `t` und `g` die folgenden Werte verwendet: Für 2t{2-10}_d1: `t`=1.8, `g`=4 und für 2t{2-10}_d2: `t`=3.6, `g`=4.

Zu Beginn wollen wir uns die zufällige (*zuf*) und die deterministische (*det*) Startlösung ansehen. Vergleicht man die beiden Lösungen, so ist zu erkennen, dass die zufälligen Startlösungen im Durchschnitt besser sind als die deterministischen. Bei den zufälligen Lösungen sinkt, bis auf eine Ausnahme, mit steigender Anzahl der Seiten die Qualität der Lösung. Bei den deterministischen Startlösungen ist kein derart kontinuierlicher Verlauf zu erkennen. Das liegt an der Tatsache, dass nur Seiten mit der selben Anzahl von Rissen vorhanden sind, bei der die im vorhinein definierte Zuordnung der Schnipsel dazu führt, dass einige Seiten eine Gesamtfläche besitzen, die eindeutig größer ist als die Referenzfläche und andere wiederum ein Fläche, die eindeutig kleiner ist (siehe auch Abschnitt 5.2.2). In Abbildung 6.1 sind die beiden Startlösungen gegenübergestellt.

Im nächsten Schritt vergleichen wir die beiden VND-Methoden, *snip* und *page*, welche auf die beiden Startlösungen (*zuf* und *det*) angewandt wurden. Bei nur zwei Seiten liefern alle Methoden, bis auf eine Ausnahme, 100%. Nur die VND-Methode *snip-zuf* erreicht keine optimale Zuordnung. Ab drei Seiten fällt die Lösungsqualität erheblich und sinkt mit steigender Seitenanzahl weiter. Zu erkennen ist aber, dass die *page* Methode bessere Ergebnisse liefert als die *snip* Methode. Vergleicht man die Ergebnisse der VND bezüglich der unterschiedlichen Startlösungen, so ist zu erkennen, dass die Qualität der Startlösung keinen Einfluss auf die Lö-

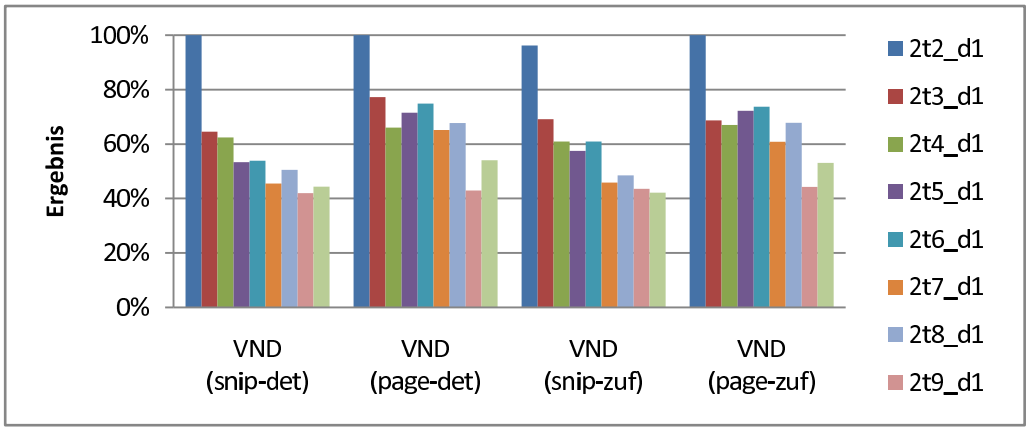


Abbildung 6.2: VND-Lösungen der Instanzen mit 2 Rissen

sungsqualität der VND hat. Denn egal, ob die zufällige Startlösung oder die etwas schlechtere deterministische verwendet wurde, liefert die *page* bzw. *snip* Methode, im Durchschnitt, die gleichen Ergebnisse. Die Lösungen der VND-Methoden sind in Abbildung 6.2 dargestellt.

Vergleicht man anschließend die vier Hybridmethoden (*edgeal*, *edgesingle*, *snipall*, *snipsingle*), die auf die Lösungen der VND angewandt wurden, erhält man Ergebnisse, die sich nur minimal voneinander unterscheiden. Lediglich die *edgesingle* Methode, angewandt auf die *VND(page-det)* Lösung, liefert für die Instanzen `2t5_d1` bis `2t8_d1` Lösungen, die durchschnittlich um 5% niedriger sind als die Lösungen der anderen Methoden. Alle anderen liefern für Instanzen mit bis zu acht Seiten fast ausschließlich die optimale Zuordnung (100%). Ab neun Seiten sinkt die Lösungsgüte, erreicht aber bei zehn Seiten noch immer Werte die zwischen 85% und 90% liegen. In Abbildung 6.3 sind die Ergebnisse aller vier Hybridmethoden dargestellt, welche auf die Lösung der *VND(page-det*) angewandt wurden.

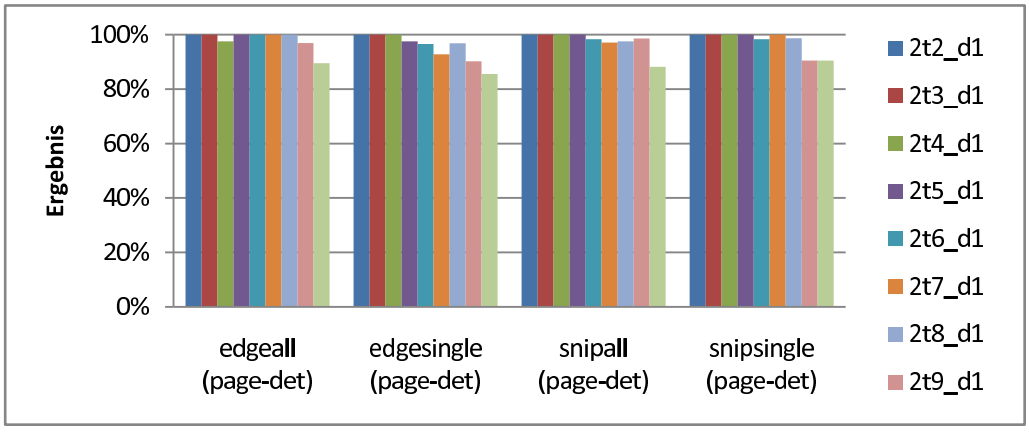


Abbildung 6.3: Hybridlösungen der Instanzen mit 2 Rissen

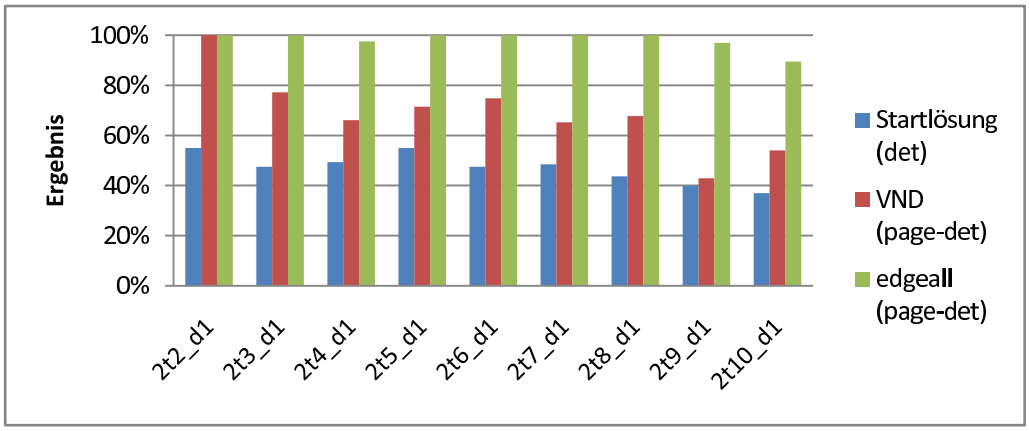


Abbildung 6.4: Vergleich der Lösungsansätze bei Instanzen mit 2 Rissen

Da alle Hybridmethoden, bis auf eine, ähnliche Ergebnisse liefern, kann keine Aussage getroffen werden, welche von ihnen am besten für das TPP geeignet ist. In Abbildung 6.4 werden die Lösungen der unterschiedlichen Ansätze (*Start-VND-Hybrid*) gegenübergestellt. Es ist sehr gut zu erkennen, dass durch VND die Startlösungen deutlich verbessert werden und für die Instanz mit zwei Seiten sogar die optimale Zuordnung erreicht wird. Durch die weitere Anwendung einer Hybridmethode erhält man dann für Instanzen mit bis zu acht Seiten, bis auf eine Ausnahme, immer die optimale Zuordnung und auch für Instanzen mit neun oder zehn Seiten liegen die Ergebnisse noch immer bei etwa 90%.

Betrachtet man die Einzelergebnisse der zehn Testdurchläufe sieht man, dass für Instanzen bis neun Seiten bei allen Testläufen, bis auf drei Ausnahmen, immer die optimale Zuordnung erreicht wurde. Nur ein Testlauf der Instanz mit drei Seiten und zwei der Instanz mit neun Seiten erreichten nicht die optimale Lösung, lagen aber auch noch über 75%. Auch für die Instanz mit zehn Seiten wurde noch dreimal die optimale Zuordnung erreicht, wobei die restlichen Ergebnisse ebenfalls, abgesehen von zwei Fällen, noch über 85% liegen. Aufgrund dieser Lösungen ist deutlich zu erkennen, dass der hier vorgestellte Algorithmus für Instanzen mit zwei Rissen sehr gut geeignet ist. Die Ergebnisse der einzelnen Testläufe sind in der Tabelle 6.2 zusammengefasst.

Vergleicht man die Ergebnisse der Instanzen `2t{2-10}_d1` mit den von `2t{2-10}_d2` ergeben sich bei den Startlösungen keine Unterschiede. Bei der *VND(snip)* Methode sinken die Ergebnisse im Durchschnitt um 2% und bei der *VND(page)* Methode um 6%. Das ist auf die Tatsache zurückzuführen, dass durch die größere Streuung beim Schereffekt die berechneten Referenzwerte auch eine größere Abweichung zu den idealen Werten aufweisen. Wie die VND liefern auch die Hybridmethoden Lösungen, die etwas schlechter sind, wobei der Unterschied bis zu Instanzen mit sieben Seiten minimal ist. Erst ab acht Seiten ist zu erkennen, dass die Instanzen `2t{2-10}_d2` deutlich schlechtere Ergebnisse liefern. Der Grund dafür ist folgender: mit dem größeren Schereffekt erhöhen sich die Abweichungen der Außenkanten. Dadurch muss sich aber ebenfalls der maximale Toleranzbereich für die Zuordnung der

Instanz	Testläufe										∅(σ)
	1	2	3	4	5	6	7	8	9	10	
2t2_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t3_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t4_d1	100	100	100	100	100	100	100	75	100	100	97,5 (7,9)
2t5_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t6_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t7_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t8_d1	100	100	100	100	100	100	100	100	100	100	100 (0)
2t9_d1	100	100	100	100	100	80,6	88,9	100	100	100	96,9 (6,7)
2t10_d1	95	70	87,5	100	85	100	75	87,5	100	95	89,5 (10,6)

Tabelle 6.2: Testergebnisse der *edgeall(page-det)* Variante für Instanzen mit 2 Rissen

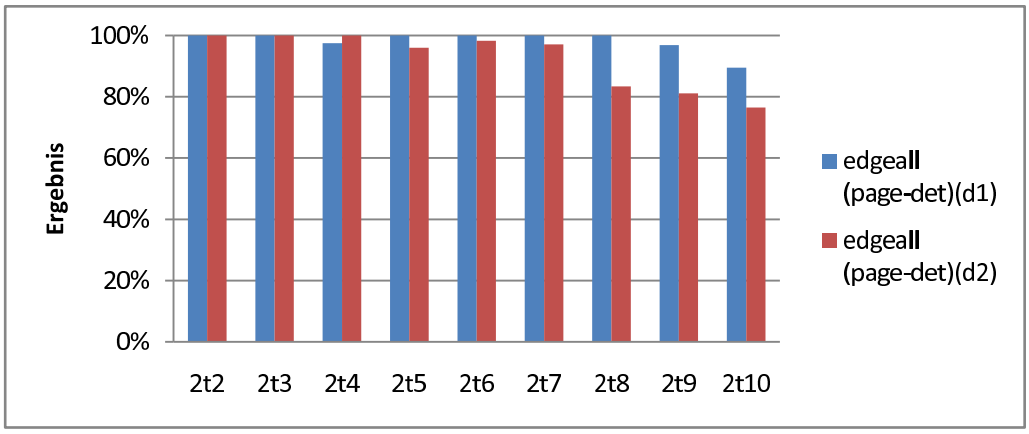


Abbildung 6.5: Hybridlösungen mit unterschiedlichen d Werten bei Instanzen mit 2 Rissen

Schnipsel zu den Seitenkanten erhöhen (siehe Abschnitt 5.2.5). Der größere Toleranzbereich führt dazu, dass die Zahl der möglichen Zuordnungen, die innerhalb der Toleranz liegen, steigt und sich mit der Anzahl der Schnipsel noch weiter erhöht. Dadurch erhöht sich aber auch die Wahrscheinlichkeit, dass die Hybridmethode eine nicht ideale Schnipselzuordnung als *gültig* annimmt, was zur Verschlechterung der Lösungen führt. Abbildung 6.5 stellt die Ergebnisse der *edgeall(page-det)* Methode für unterschiedliche d-Werte gegenüber.

Die für die Lösung benötigten Zeiten weisen eine sehr große Streuung auf. Zu erkennen ist aber, dass die VND für alle Instanzen Lösungen innerhalb von einer Minute liefert. Auch die Hybridmethoden benötigen für fast alle Instanzen mit bis zu sechs Seiten nur zirka eine Minute. Ab sieben Seiten pro Instanz steigt die benötigte Zeit für die Hybridmethode mit jeder zusätzlichen Seite rapide an und kann bei zehn Seiten über 15 Minuten dauern.

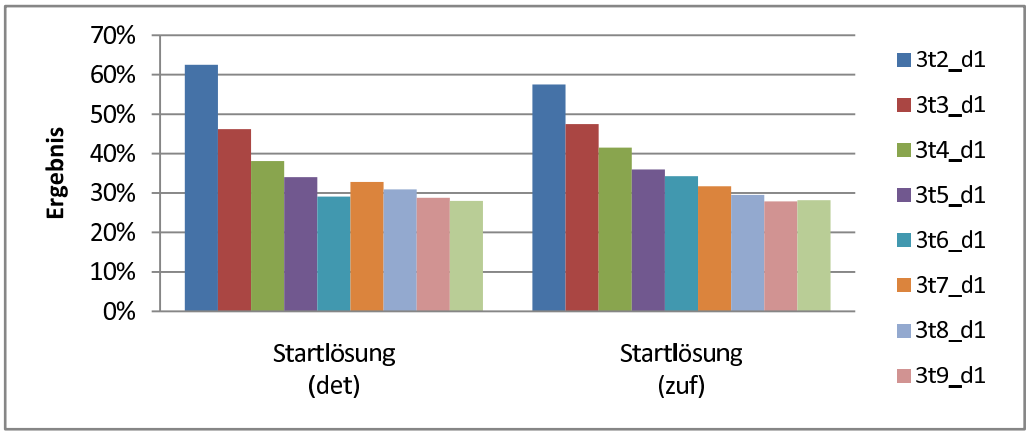


Abbildung 6.6: Startlösungen der Instanzen mit 3 Rissen

Instanzen mit drei Rissen

Für Instanzen, die nur Seiten mit drei Rissen enthalten, wurden für `t` und `g` die folgenden Werte verwendet: Für `3t{2-10}_d1`: `t`=1.4, `g`=3 und für `3t{2-10}_d2`: `t`=3.2, `g`=1.

Vergleicht man hier die zufälligen und deterministischen Startlösungen, so ist ein ähnlicher Verlauf wie bei den Instanzen, die nur Seiten mit zwei Rissen enthalten, zu erkennen. Bis auf die Instanz mit zwei Seiten, bei denen die deterministische Startlösung eindeutig besser ist, ist zu erkennen, dass die zufälligen Startlösungen im Durchschnitt besser als die deterministischen sind. Auch hier sinkt bei den zufälligen Lösungen, bis auf eine Ausnahme, mit steigender Seitenanzahl die Qualität der Lösung. Bei den deterministischen Startlösungen ist das nicht der Fall, was auf die Methode der Schnipselzuordnung, wie schon bei den Instanzen mit zwei Rissen erklärt, zurückzuführen ist. Abbildung 6.6 stellt die Ergebnisse der verschiedenen Startlösungen dar.

Betrachtet man die Ergebnisse der unterschiedlichen VND-Methoden sieht man, dass bis auf die Ergebnisse bei Instanzen mit zwei und drei Seiten, alle Methoden durchschnittlich ähnliche Ergebnisse liefern: Mit steigender Anzahl der Seiten sinkt die Lösungsqualität. Nur die *page* Methode, angewandt auf die deterministische Startlösung *VND(page-det)*, fällt für zwei und drei Seiten aus der Reihe. Sie liefert für zwei Seiten die mit Abstand schlechteste Lösung, jedoch für drei Seiten eindeutig die beste. Die anderen Methoden unterscheiden sich nur bei der Instanz mit zwei Seiten eindeutig. Auch hier ist, wie schon bei den Instanzen mit nur zwei Rissen, zu erkennen, dass die Qualität der Startlösung keinen Einfluss auf die Lösungsqualität der VND besitzt. Zusätzlich wurde bemerkt, dass bei der *page* Methode ab fünf Seiten und bei der *snip* Methode ab sieben Seiten die maximale Iterationszahl (*Parameter-m*) erreicht wurde. In Abbildung 6.7 sind die Lösungen der VND-Methoden gegenübergestellt.

Wenn man die Ergebnisse der Hybridmethoden betrachtet ist zu erkennen, dass sich die Lösungen im Durchschnitt nur minimal voneinander unterscheiden. Lediglich bei den Instanzen

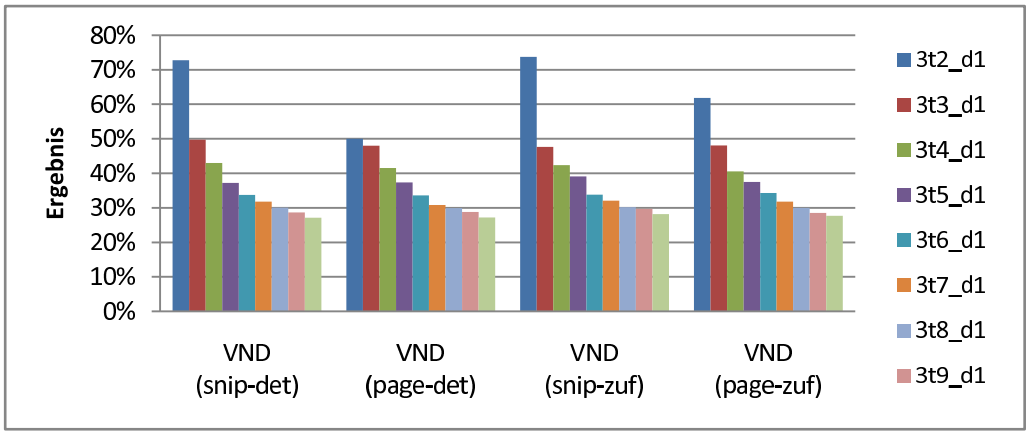


Abbildung 6.7: VND-Lösungen der Instanzen mit 3 Rissen

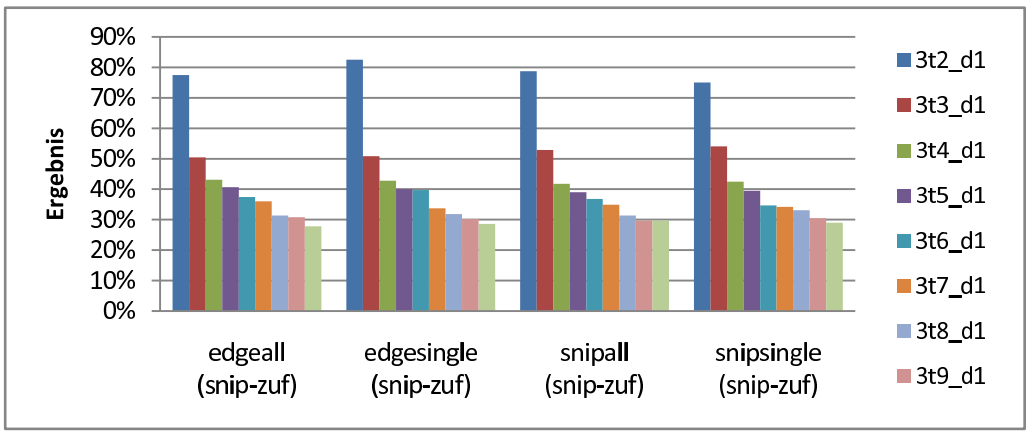


Abbildung 6.8: Hybridlösungen der Instanzen mit 3 Rissen

mit zwei und drei Seiten sind Abweichungen von bis zu 8% zu erkennen. Es kann somit auch für diese Instanzen keine Aussage getroffen werden, welche der Hybridmethoden am besten zur Lösung geeignet ist. Abbildung 6.8 zeigt die Ergebnisse der Hybridmethoden, welche auf die Lösung der *VND(snip-zuf)* angewandt wurden.

Der Vergleich der Start-, VND- und Hybridlösungen zeigt, dass die VND die Startlösung für zwei Seiten noch erheblich verbessert. Ab drei Seiten sind die Verbesserungen nicht mehr so deutlich und für die Instanzen mit sechs bzw. zehn Seiten sogar geringfügig schlechter. Die weitere Anwendung einer Hybridmethode liefert noch einmal eine kleine Steigerung der VND-Ergebnisse. Auch wenn man hier keine optimalen Lösungen erhält zeigt sich eindeutig, dass eine Verbesserung der Startlösung durch den Algorithmus erzielt wurde. In Abbildung 6.9 sind die Lösungen der verschiedenen Ansätze zu sehen.

In Tabelle 6.3 sind die Ergebnisse der einzelnen Testläufe, der Hybridmethode *snipsingle(snip-zuf)*, zusammengefasst. Es wird nur mehr für die Instanz mit zwei Seiten viermal eine optimale

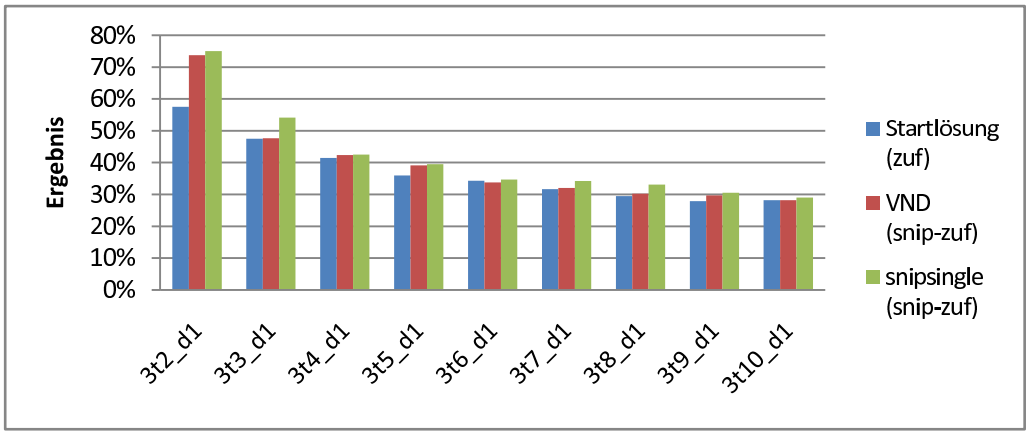


Abbildung 6.9: Vergleich der Lösungsansätze bei Instanzen mit 3 Rissen

Instanz	Testläufe										$\varnothing(\sigma)$
	1	2	3	4	5	6	7	8	9	10	
3t2_d1	50	100	50	62,5	100	100	75	100	62,5	50	75 (22,8)
3t3_d1	62,5	54,2	54,2	70,8	50	41,7	58,3	50	50	50	54,2 (8,1)
3t4_d1	50	40,3	37,5	43,8	34,4	40,6	43,8	40,6	43,8	50	42,5 (4,9)
3t5_d1	35	35	42,5	40	42,5	37,5	35	42,5	37,5	47,5	39,5 (4,2)
3t6_d1	39,6	33,3	33,3	39,6	33,3	33,3	35,4	29,2	33,3	37,5	34,8 (3,2)
3t7_d1	37,5	33,9	37,5	35,7	33,9	37,5	33,9	32,1	32,1	30,4	34,3 (2,4)
3t8_d1	32,8	29,7	42,2	29,7	34,4	32,8	31,3	31,3	31,3	35,9	33,1 (3,7)
3t9_d1	29,2	33,3	30,6	31,9	30,6	27,8	30,6	30,6	30,6	30,6,	30,6 (1,5)
3t10_d1	31,3	26,3	27,5	27,5	28,8	25	30	30	30	33,8	29 (2,6)

Tabelle 6.3: Testergebnisse der *snipsingle(snip-zuf)* Variante für Instanzen mit 3 Rissen

Zuordnung erreicht. Bei allen anderen Instanzen liegen die Lösungen unter 100%.

Beim Vergleich der Ergebnisse für die Instanzen `3t{2-10}_d1` mit denen für die Instanzen `3t{2-10}_d2`, wurden bei den Start- und VND-Lösungen keine Unterschiede in der Lösungsqualität festgestellt. Die Hybridmethoden liefern hingegen für `3t{2-10}_d2` Ergebnisse, die im Vergleich um ein bis zwei Prozent schlechter sind. Das ist wieder auf den größeren Toleranzbereich für die Seitenkanten und der dadurch größeren Möglichkeiten an *gültigen* Zuordnungen zurückzuführen.

Wie schon bei den Lösungen für Instanzen mit nur zwei Rissen, weisen auch hier die benötigten Zeiten eine sehr große Streuung auf. Die Zeiten für die VND reichen von ungefähr einer Minute für Instanzen mit zwei Seiten bis zu ungefähr fünf Minuten für Instanzen mit zehn Seiten. Die Hybridmethoden benötigten nur für zwei Seiten eine Zeit unterhalb von drei Minuten, für alle anderen Instanzen werden Zeiten bis zu 15 Minuten und manchmal auch mehr benötigt.

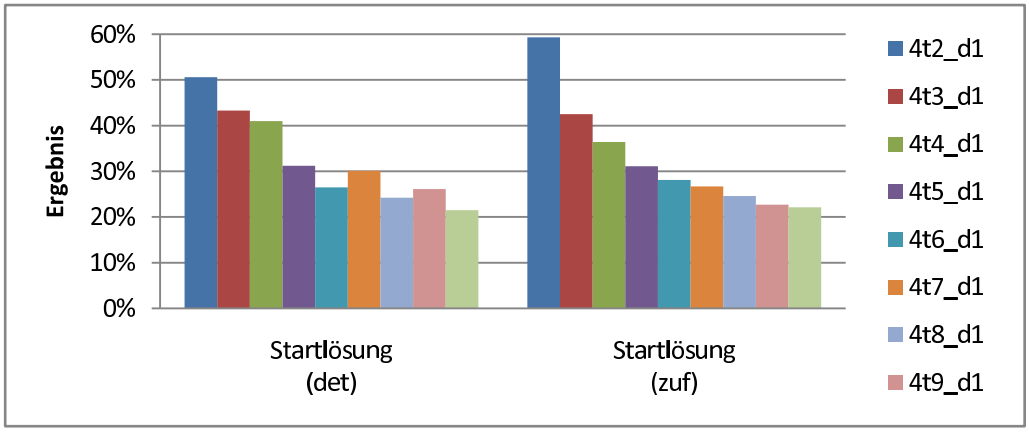


Abbildung 6.10: Startlösungen der Instanzen mit 4 Rissen

Instanzen mit vier Rissen

Für diesen Instanztyp wurden die folgenden Parameter verwendet: Für `4t{2-10}_d1`: `t`=1.2, `g`=1 und für `4t{2-10}_d2`: `t`=3.0, `g`=1.

Die Startlösungen zeigen wieder ein ähnliches Verhalten wie wir schon bei den Instanzen mit zwei Rissen und drei Rissen gesehen haben. Die zufälligen Lösungen zeigen einen kontinuierlichen Verlauf und werden mit steigender Seitenanzahl schlechter. Die deterministischen Startlösungen haben auch hier wieder einen unregelmäßigen Verlauf bezüglich steigender Seitenzahl. Grund dafür ist auch hier die Methode der Schnipselzuordnung, wie bei den Instanzen mit zwei Rissen schon erklärt wurde. Im Durchschnitt über alle Instanzen liefern aber beide Startlösungen Ergebnisse, die sich nur minimal voneinander unterscheiden. In Abbildung 6.10 sind die beiden Startlösungen dargestellt.

Alle VND-Methoden liefern beinahe idente Lösungen. Nur die *page* Methode auf die deterministische Lösung angewandt, liefert für zwei Seiten eine Lösung, die, verglichen mit den anderen, um ungefähr 8% schlechter ist. Durch die größere Anzahl von Schnipsel erreichte die VND für beide Methoden (*page*, *snip*) ab sechs Seiten die maximal erlaubten Iterationen. Die Lösungen der VND sind in Abbildung 6.11 zu sehen.

Beim Vergleich der unterschiedlichen Hybridmethoden sind keine merkbaren Unterschiede zu erkennen. Alle Ergebnisse weisen nur minimale Abweichungen auf, deshalb kann auch für diese Instanzen nicht entschieden werden, welche Methode die besseren Ergebnisse liefert. Die Ergebnisse der vier Hybridmethoden sind in Abbildung 6.12 dargestellt.

Betrachtet man die Lösungen der VND und Hybridmethode im Vergleich zu den Startlösungen, so bemerkt man, dass die VND bezüglich der Startlösung im Durchschnitt sogar kleine Verschlechterungen der Lösungen liefert. Die Hybridmethode liefert zwar geringfügig bessere Ergebnisse als die VND, jedoch bezogen auf die Startlösung ist nur bei der Instanz mit drei Seiten eine deutliche Verbesserung zu bemerken. Für alle anderen sind wiederum nur leichte

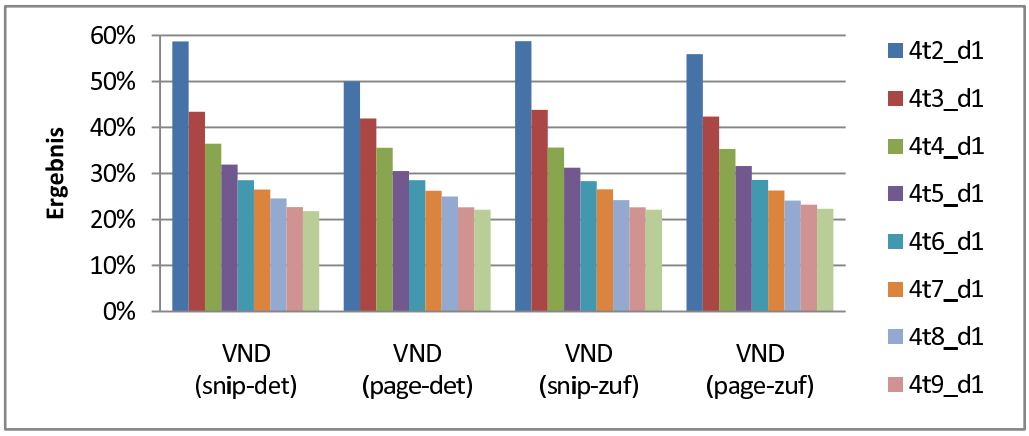


Abbildung 6.11: VND-Lösungen der Instanzen mit 4 Rissen

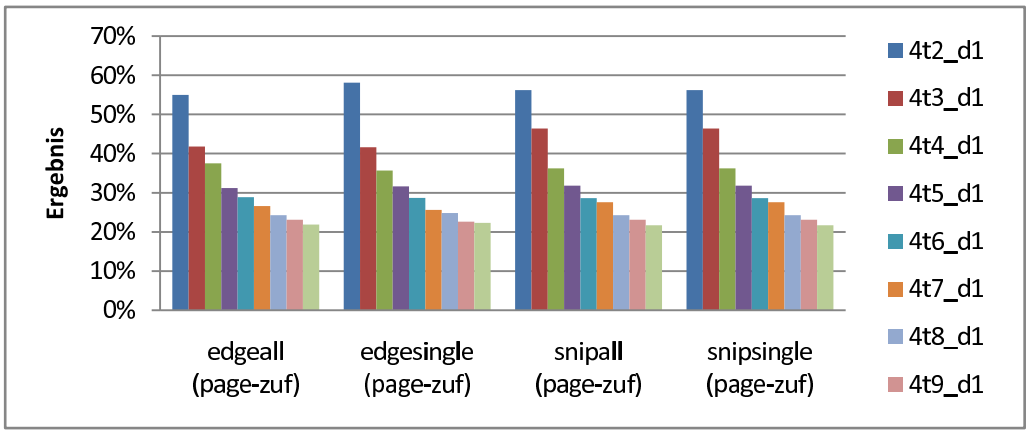


Abbildung 6.12: Hybridlösungen der Instanzen mit 4 Rissen

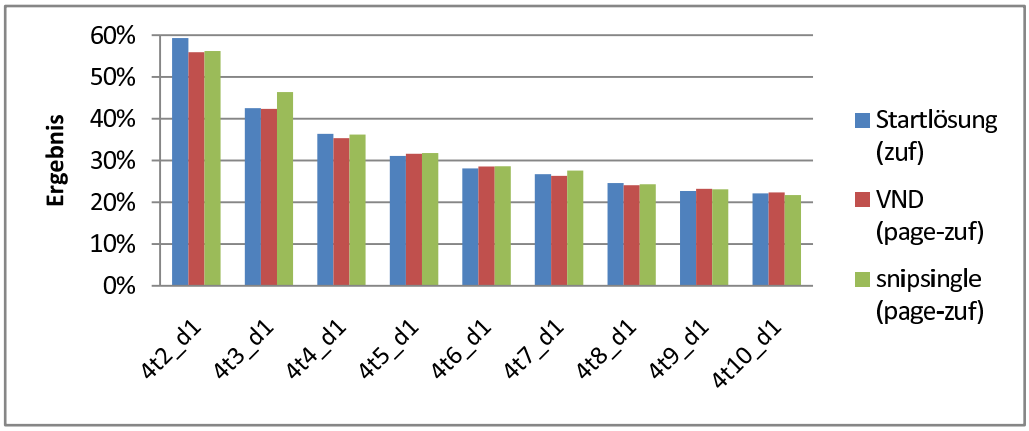


Abbildung 6.13: Vergleich der Lösungsansätze bei Instanzen mit 4 Rissen

Instanz	Testläufe										$\varnothing(\sigma)$
	1	2	3	4	5	6	7	8	9	10	
4t2_d1	50	56,3	56,3	56,3	62,5	56,3	62,5	56,3,5	50	56,3	56,3 (4,2)
4t3_d1	47,9	45,8	47,9	43,8	47,9	43,8	45,8	41,7	45,8	45,8	45,6 (2,1)
4t4_d1	35,9	39,1	35,9	35,9	40,6	34,4	34,4	32,8	37,5	35,9	36,3 (2,3)
4t5_d1	33,8	30	28,8	26,3	33,8	33,8	35	30	33,8	30	31,5 (2,9)
4t6_d1	32,3	33,3	28,2	29,2	25	31,3	29,2	29,2	31,3	26	29,5 (2,6)
4t7_d1	28,6	25	28,6	26,8	26,8	28,6	28,6	26,8	25	26,8	27,2 (1,4)
4t8_d1	24,2	21,9	27,3	25	21,9	24,2	24,2	23,4	25,8	23,4	24,2 (1,7)
4t9_d1	22,9	23,6	21,6	27,1	22,2	25,7	24,3	29,2	21,5	20,2	23,8 (2,8)
4t10_d1	21,9	20,6	21,3	21,9	22,5	22,5	21,3	22,5	21,9	20	21,6 (0,9)

Tabelle 6.4: Testergebnisse der *snipsingle(page-zuf)* Variante für Instanzen mit 4 Rissen

Verbesserungen oder auch Verschlechterungen zu beobachten. Abbildung 6.13 stellt die Lösungen der verschiedenen Ansätze dar.

Bei den Ergebnissen der einzelnen Testläufe zeigt sich, dass kein einziges Mal eine optimale Zuordnung erreicht wurde und nur die Instanzen mit zwei Seiten Zuordnungen von über 50% erreichen. Eine Auflistung aller Testläufe, der *snipsingle(page-zuf)* Variante, ist in Tabelle 6.4 zu sehen.

Zwischen den Ergebnissen der Instanzen `4t{2-10}_d1` und `4t{2-10}_d2` gibt es keine Unterschiede. Der größere Schereffekt hat somit auf die Ergebnisse für Instanzen, die nur Seiten mit vier Rissen enthalten, keinen Einfluss.

Wie schon zuvor weisen auch hier die Zeiten eine große Streuung auf. Die VND benötigt für zwei Seiten ungefähr 3 Minuten, mit steigender Seitenanzahl erhöhen sich auch die Zeiten und erreichen für zehn Seiten an die 10 Minuten. Die Hybridmethoden liefern hingegen für alle Instanzen Lösungen innerhalb von einer Minute.

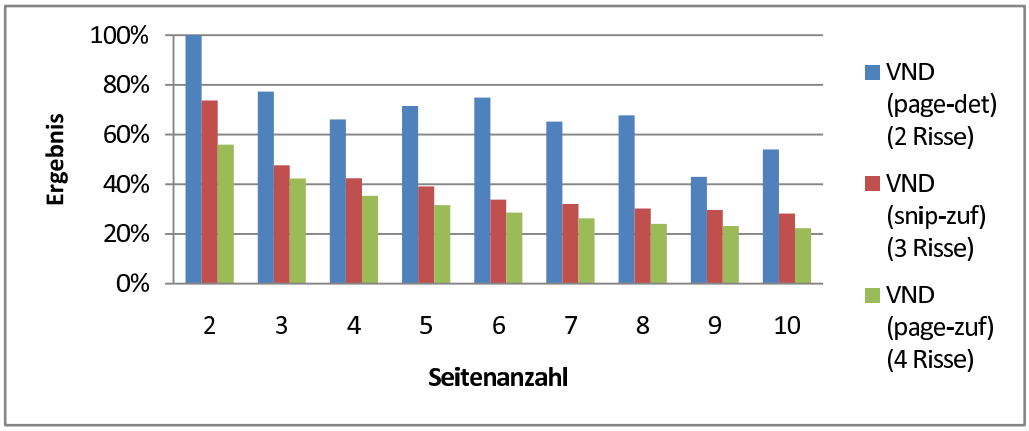


Abbildung 6.14: Vergleich der VND-Lösungen der Instanzen mit 2,3 oder 4 Rissen

Als nächstes vergleichen wir die Lösungen der VND für die Instanzen mit zwei, drei oder vier Rissen. Es zeigt sich, dass die VND für Instanzen mit zwei Rissen bei zwei Seiten die optimale Lösung liefert und auch für mehrere Seiten noch Lösungen von über 60% erreicht. Für Instanzen mit drei Rissen sinkt die Qualität der Lösungen deutlich und verschlechtert sich für vier Risse weiter. Der Grund dafür ist, dass sich bei einer höheren Anzahl der Risse auf einer Seite auch die Anzahl der ähnlichen Schnipsel erhöht, welche zugeordnet werden müssen. Zusätzlich steigen mit einer größeren Schnipselzahl ebenfalls die Referenzwerte und somit auch die Ungenauigkeiten bezüglich der Idealwerte. Dadurch erhöhen sich die Zuordnungsmöglichkeiten, die es der VND erschweren eine optimale Zuordnung zu liefern. Dies führt auch dazu, dass mit einer steigenden Schnipselzahl die Zahl der benötigen Iterationen steigt und ab etwa 60 Schnipsel die maximal erlaubten Iterationen (`m`-Parameter) erreicht wurden. Die größere Anzahl von Iterationen ist auch der Grund für die höheren Zeiten. Eine Gegenüberstellung der VND-Lösungen ist in Abbildung 6.14 zu sehen.

Bei den Hybridmethoden sieht man, dass diese für Instanzen mit zwei Rissen sehr gut geeignet sind. Denn man erhält für Instanzen mit bis zu acht Seiten, bis auf eine Ausnahme, immer eine optimale Zuordnung und auch für zehn Seiten liegt die Zuordnung noch bei etwa 90%. Doch auch wie bei der VND werden auch bei den Hybridmethoden die Lösungen mit steigender Anzahl der Risse deutlich schlechter. Dies lässt sich auf die folgende Tatsache zurückzuführen: Durch eine größere Anzahl von Rissen steigt auch die Zahl der ähnlichen Schnipsel, welche auf eine Seitenkante zugeordnet werden müssen. Jedes Schnipsel, das auf eine Seitenkante zugeordnet werden muss, erhöht aufgrund des Schereffekts auch den Toleranzbereich für diese Seitenkante. Die größere Toleranz führt aber dazu, dass neben der idealen Zuordnung die Zahl der zusätzlichen Zuordnungsmöglichkeiten steigt, welche für den jeweiligen Toleranzbereich ebenfalls als *gültig* erkannt werden. Aus diesem Grund steigt die Wahrscheinlichkeit einer falschen Zuordnung und führt somit zu den nicht so guten Ergebnissen für Instanzen mit einer höheren Anzahl von Rissen. Auf diese Tatsache lassen sich auch die geringen Zeiten für die Instanzen mit vier Rissen zurückführen, denn es werden für diesen Instanztyp sehr schnell

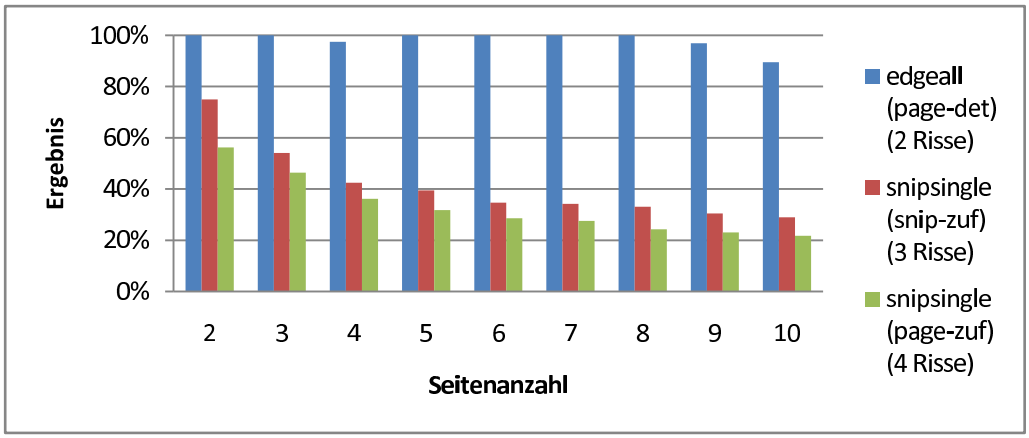


Abbildung 6.15: Vergleich der Hybridlösungen der Instanzen mit 2,3 oder 4 Rissen

Zuordnungen zu Seitenkanten gefunden, die innerhalb des jeweiligen Toleranzbereichs liegen. Ein Vergleich der Hybridlösungen ist in Abbildung 6.15 dargestellt.

Als nächstes werden wir Ergebnisse für Instanzen betrachten, die Seiten mit einer unterschiedlichen Anzahl von Rissen beinhalten.

Instanzen mit zwei und drei Rissen

Für die Instanzen, die Seiten mit zwei Rissen und Seiten mit drei Rissen beinhalten, wurden die folgenden Parameter verwendet: Für alle Instanzen mit `d`=1: `t`=1.6, `g`=3 und für alle Instanzen mit `d`=2: `t`=3.2, `g`=4.

Wenn man hier die unterschiedlichen Startlösungen miteinander vergleicht sieht man, dass, durch die Zuordnung der größeren Schnipsel zu den Seiten mit weniger Rissen, die deterministischen Startlösungen im Durchschnitt um 6% bis 7% besser als die zufälligen Lösungen sind. Abbildung 6.16 zeigt die beiden Startlösungen.

Betrachtet man die Lösungen der VND-Methoden so ist zu erkennen, dass die Qualität der Startlösung wieder keinen Einfluss auf die Lösungen der VND hat. Außerdem erkennt man, dass für Instanzen, die Seiten mit zwei Rissen und Seiten mit drei Rissen beinhalten, die *page* Methoden Ergebnisse liefert, die durchschnittlich um 2% bis 3 % besser als jene der *snip* Methoden sind. In Abbildung 6.17 sind die Lösungen der verschiedenen VND-Methoden dargestellt.

Bei den Ergebnissen der Hybridmethoden zeigt sich, dass die Methoden, welche auf die *VND(page-det)* oder *VND(page-zuf)* angewandt wurden, im Durchschnitt um ein Prozent besser sind als die, die auf eine der *VND(snip)* Lösungen basieren. Der Vergleich der vier Hybridmethoden zeigt nur bei der Instanz `2t1_3t3_d1` eine größere Abweichung, alle anderen

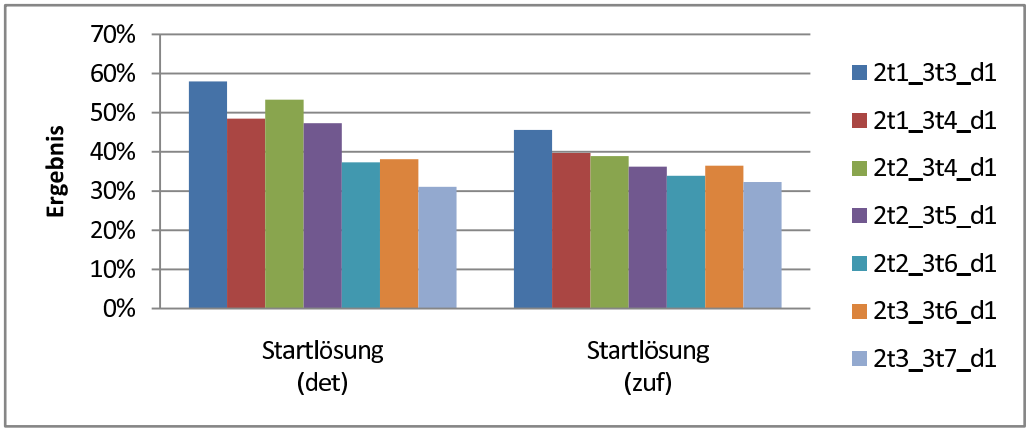


Abbildung 6.16: Startlösungen der Instanzen mit 2 und 3 Rissen

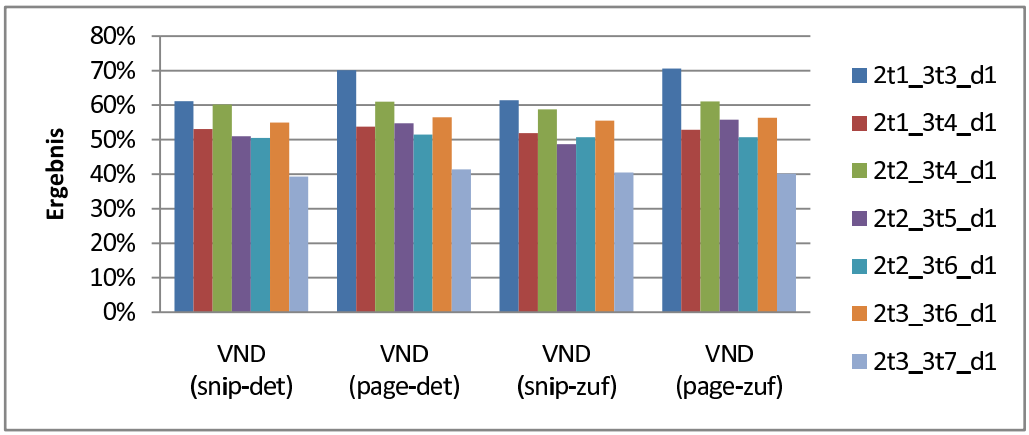


Abbildung 6.17: VND-Lösungen der Instanzen mit 2 und 3 Rissen

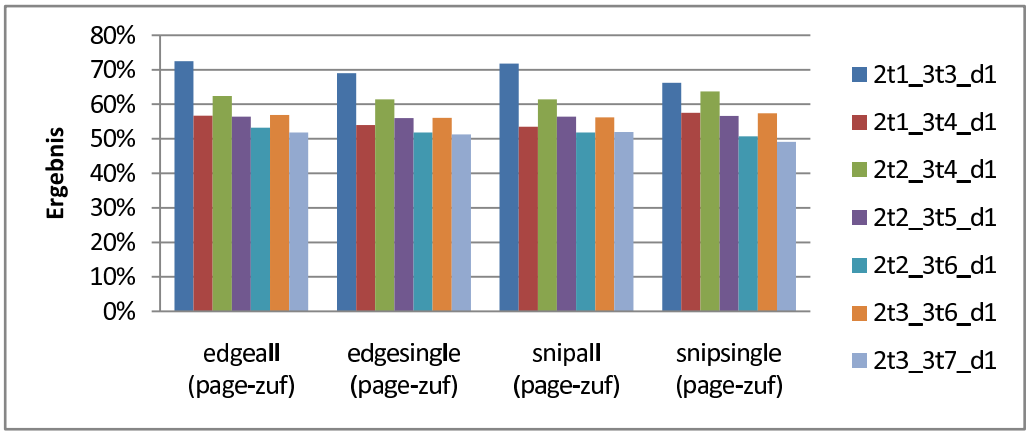


Abbildung 6.18: Hybridlösungen der Instanzen mit 2 und 3 Rissen

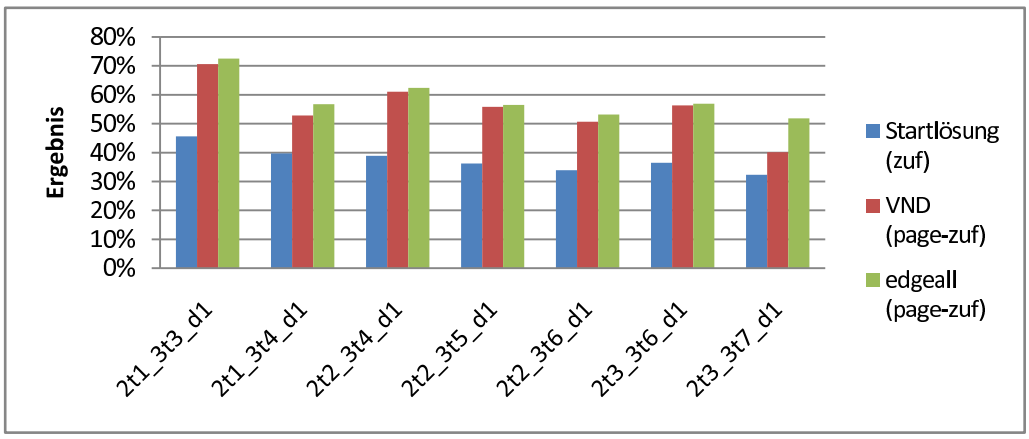


Abbildung 6.19: Vergleich der Lösungsansätze bei Instanzen mit 2 und 3 Rissen

liefern durchschnittlich dieselben Ergebnisse. Abbildung 6.18 stellt die Ergebnisse der Hybridmethoden, angewandt auf die *VND(page-zuf)* Lösungen, dar.

Vergleicht man die Lösungen von den unterschiedlichen Ansätzen sieht man eine deutliche Verbesserung der Startlösungen durch die VND. Auch die Anwendung einer Hybridvariante liefert für alle Instanzen weitere Verbesserungen der Ergebnisse. Eine Gegenüberstellung der verschiedenen Lösungen ist in Abbildung 6.19 zu sehen, wobei die Ergebnisse der Hybridmethode in der Tabelle 6.5 aufgelistet sind.

Als nächstes werden die Teilergebnisse für die unterschiedlichen Seitentypen (Seiten mit zwei oder drei Rissen) getrennt voneinander betrachtet. Dabei ist zu erkennen, dass für die Seiten mit zwei Rissen, bis auf eine Ausnahme, immer die optimale Zuordnung gefunden wird, aber auch die Instanz `2t3_3t7_d1` noch einen Wert von 95% erreicht. Die Zuordnung zu den Seiten mit drei Rissen ist deutlich schlechter, was wieder auf den größeren Toleranzbereich der Seitenkanten, zurückzuführen ist. Abbildung 6.20 stellt die Teilergebnisse für die unter-

Instanz	Testläufe										$\varnothing(\sigma)$
	1	2	3	4	5	6	7	8	9	10	
2t1_3t3_d1	81,3	65,3	65,3	59,4	100	75	68,8	81,3	62,5	65,3	72,5 (12,2)
2t1_3t4_d1	52,5	65	55	52,5	65	55	55	52,5	57,5	57,5	56,8 (4,7)
2t2_3t4_d1	64,6	64,6	62,5	56,3	58,3	60,4	64,6	64,6	64,6	64,6	62,5 (3,1)
2t2_3t5_d1	53,6	57,1	55,4	57,1	55,4	57,1	58,9	51,8	58,9	53,6	55,9 (2,4)
2t2_3t6_d1	54,7	57,8	50	53,1	56,3	53,1	56,3	48,4	53,1	50	53,3 (3,1)
2t3_3t6_d1	55,6	59,7	56,9	54,2	62,5	56,9	59,7	58,3	52,8	52,8	56,9 (3,2)
2t3_3t7_d1	57,5	51,3	50	48,8	51,3	53,8	46,3	55	53,8	51,3	51,9 (3,2)

Tabelle 6.5: Testergebnisse der *edgeall(page-zuf)* Variante für Instanzen mit 2 und 3 Rissen

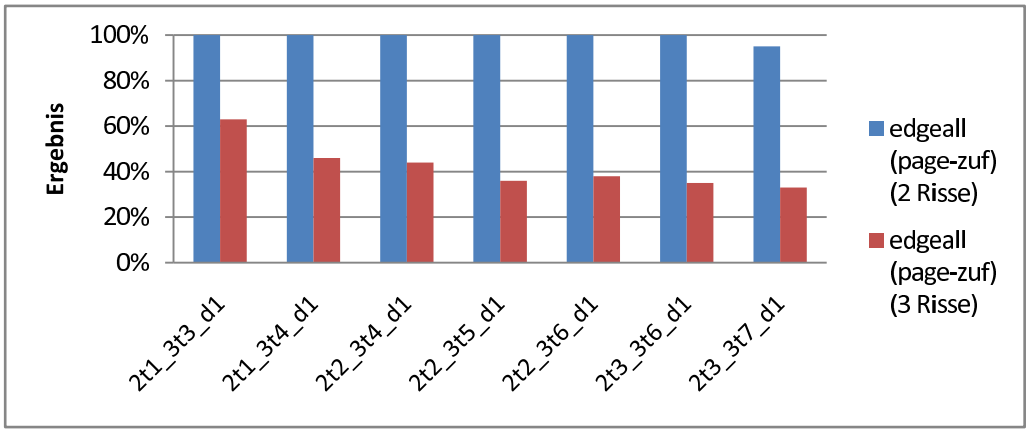


Abbildung 6.20: Teilergebnisse der unterschiedlichen Seitentypen (2 oder 3 Risse) bei Instanzen mit 2 und 3 Rissen

schiedlichen Seitentypen dar.

Der Vergleich zwischen den Instanzen mit d=1 und denen mit d=2 zeigt, dass sich die Startlösungen nicht unterscheiden. Die Ergebnisse der VND und der Hybridmethoden für Instanzen mit d=2 sind um zirka 2% bis 3% schlechter als jene für d=1. Der Grund dafür liegt, wie vorher schon erwähnt, bei der VND an der größeren Abweichung der Referenzwerte von ihren Idealwerten und bei der Hybridvariante am etwas größeren Toleranzbereich für die Seitenkanten.

Auch hier weisen die für die Berechnung benötigten Zeiten eine hohe Streuung auf. Die VND liefert dabei Lösungen innerhalb von einer halben Minute, da durch die unterschiedlich großen Schnipsel weniger Möglichkeiten zum Austauchen vorhanden sind. Im Gegensatz dazu benötigen die Hybridvarianten eine Zeit, die zwischen einer Minute und über 15 Minuten liegen kann.

Im Anschluss werden noch Instanzen betrachtet, die Seiten mit zwei Rissen, Seiten mit drei Rissen und Seiten mit vier Rissen beinhalten.

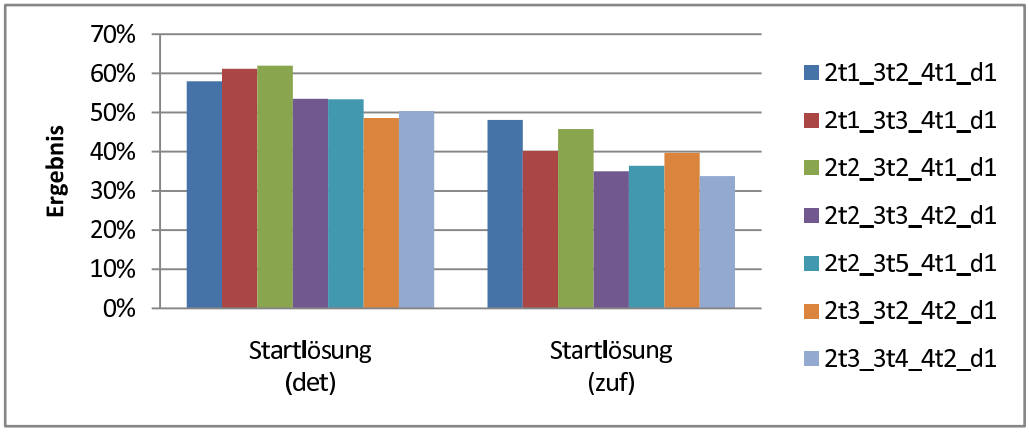


Abbildung 6.21: Startlösungen der Instanzen mit 2,3 und 4 Rissen

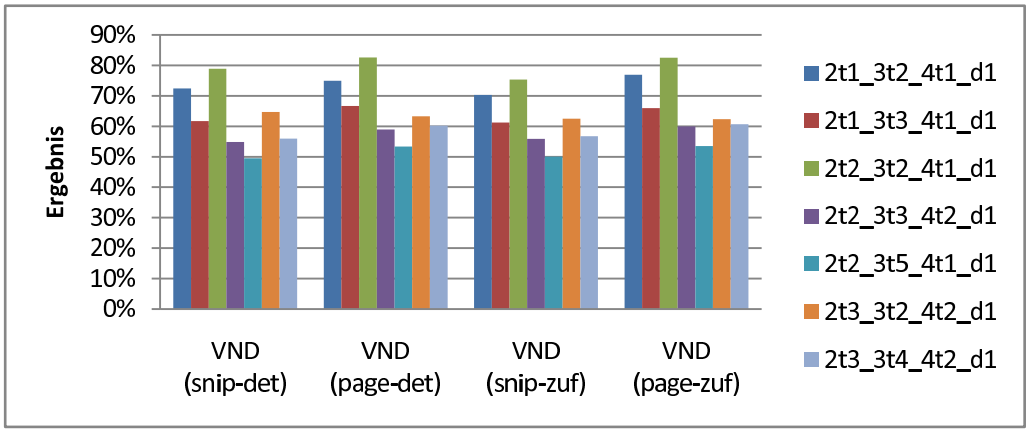


Abbildung 6.22: VND-Lösungen der Instanzen mit 2,3 und 4 Rissen

Instanzen mit zwei, drei und vier Rissen

Für diesen Instanztyp wurden die folgenden Parameter angewandt: Für alle Instanzen mit d=1: t=1.4, g=3 und für alle Instanzen mit d=2: t=3.2, g=2.

Auch hier zeigt sich, wie bei den Instanzen mit zwei und drei Rissen, dass, durch die Zuordnung der größeren Schnipsel zu den Seiten mit weniger Rissen, die deterministischen Startlösungen eindeutig besser als die zufälligen sind. Im Durchschnitt unterscheiden sich die beiden um etwa 15%, wie in Abbildung 6.21 zu sehen ist.

Bei der VND liefert die *page* Methode Ergebnisse, die um etwa 3% bis 4% besser als jene der *snip* Methode sind. Dabei ist es egal auf welche Startlösung die VND angewandt wurde. Die Unterschiede der VND-Methoden sind in Abbildung 6.22 gut zu erkennen.

Ebenfalls, wie bei den Instanzen mit zwei unterschiedlichen Rissen, liefern hier jene Hybridmethoden, die auf Lösungen der *VND(page)* angewandt wurden, besser Ergebnisse als die,

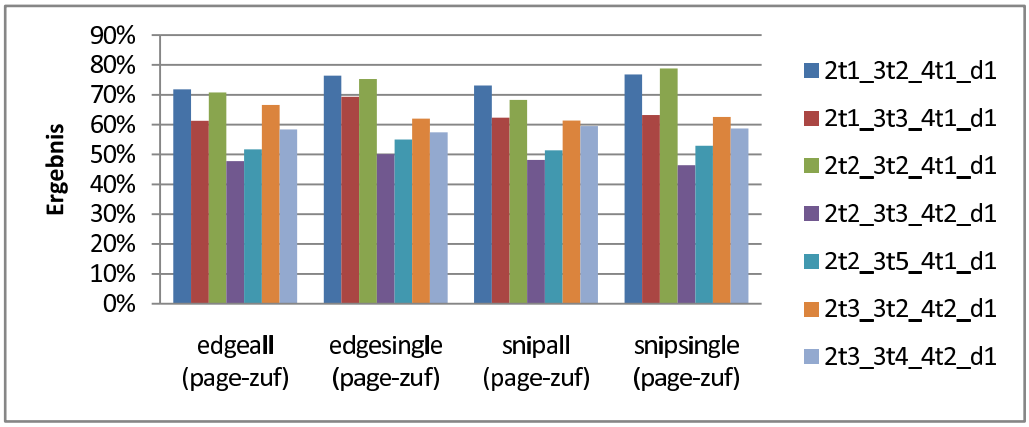


Abbildung 6.23: Hybridlösungen der Instanzen mit 2,3 und 4 Rissen

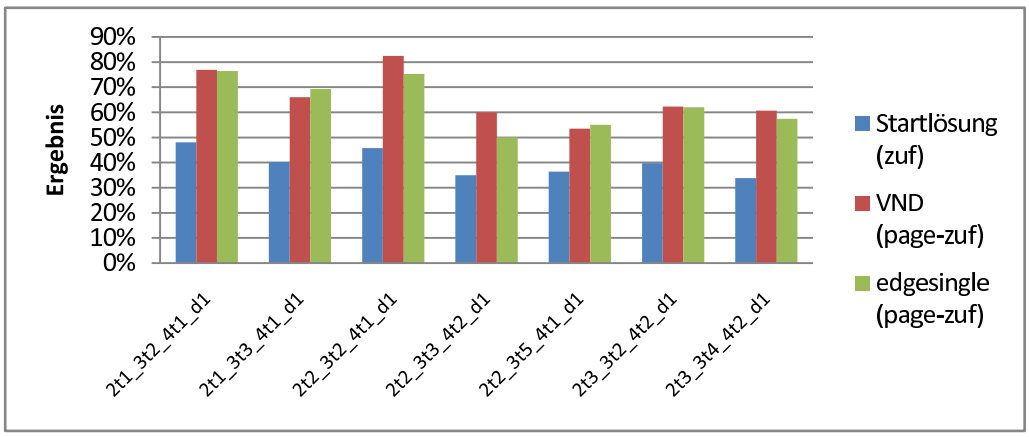


Abbildung 6.24: Vergleich der Lösungsansätze bei Instanzen mit 2,3 und 4 Rissen

die auf *VND(snip)* Lösungen basieren (im Durchschnitt um etwa 3%). Beim Vergleich der Hybridmethoden zeigt sich außerdem, dass die *edgesingle* und *snipsingle* Varianten geringfügig besser als die *edgeall* und *snipall* Varianten sind. In Abbildung 6.23 ist dieses Verhalten gut zu sehen.

Beim Vergleich der Lösungen der unterschiedlichen Ansätze ist eine deutliche Verbesserung der Startlösung durch die VND zu erkennen. Die Anwendung der Hybridmethoden auf die Instanzen, die Seiten mit zwei Rissen, Seiten mit drei Rissen und Seiten mit vier Rissen enthalten, liefert jedoch keine weitere Verbesserung mehr. Im Gegenteil, die Hybridvariante verschlechtert die Ergebnisse der VND sogar minimal (siehe Abbildung 6.24). Die Ergebnisse von den einzelnen Testläufen der VND sind in Tabelle 6.6 dargestellt.

Betrachtet man die Teilergebnisse für die unterschiedlichen Seitentypen (Seiten mit 2, 3 oder 4 Rissen) getrennt voneinander, so zeigt sich, dass für die Seiten mit zwei Rissen, bis auf zwei Ausnahmen, immer ein Zuordnung von über 90% erreicht wurde, wobei diese für die

Instanz	Testläufe										$\varnothing(\sigma)$
	1	2	3	4	5	6	7	8	9	10	
2t1_3t2_4t1_d1	100	73,4	65,6	75	70,3	70,3	70,3	100	70,3	76,6	77,2 (12,4)
2t1_3t3_4t1_d1	58,8	65	70	78,8	67,5	65	65	67,5	66,3	65	66,9 (5,1)
2t2_3t2_4t1_d1	96,3	80	83,8	85	60	96,3	80	80	83,8	80	82,5 (10,1)
2t2_3t3_4t2_d1	62,5	59,8	66,1	61,6	64,3	58,9	46,4	62,5	52,7	60,7	59,6 (5,8)
2t2_3t5_4t1_d1	59,4	59,4	57,8	40,6	45,3	57,8	53,9	59,4	61,7	42,2	53,8 (7,9)
2t3_3t2_4t2_d1	58	67,9	74,1	60,7	60,7	61,6	58	64,3	64,3	60,7	63 (4,9)
2t3_3t4_4t2_d1	64,6	58,3	47,9	61,1	70,8	62,5	56,9	66	66	59,7	61,4 (6,3)

Tabelle 6.6: Testergebnisse der *VND(page-zuf)* Variante für Instanzen mit 2,3 und 4 Rissen

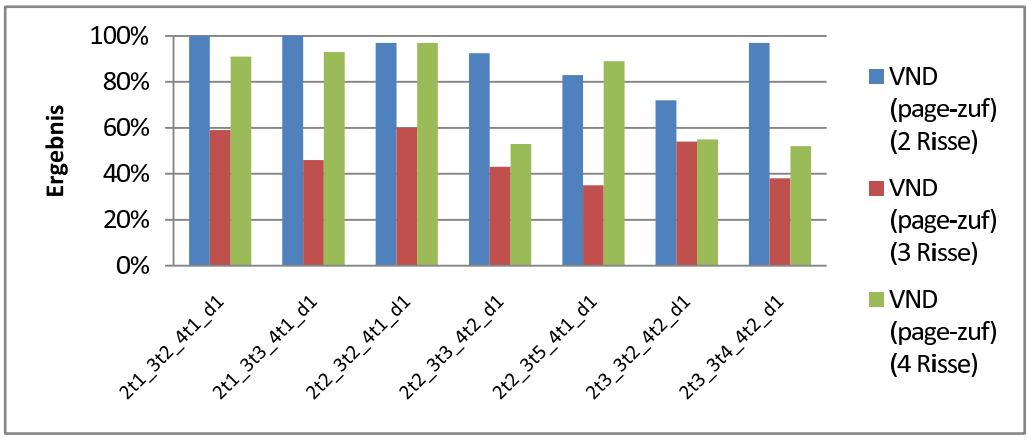


Abbildung 6.25: Teilergebnisse der unterschiedlichen Seitentypen (2,3 oder 4 Risse) bei Instanzen mit 2,3 und 3 Rissen

Instanzen `2t1_3t2_4t1_d1` und `2t1_3t3_4t1_d1` sogar optimal sind. Für Seiten mit drei Rissen sind die Ergebnisse schon wieder deutlich schlechter. Bei den Seiten mit vier Rissen erhält man dann eine gute Zuordnung, solange nur eine Seite dieses Typs in der Instanz vorhanden ist. Ab zwei Seiten mit vier Rissen sinkt auch hier die Qualität der Lösung rapide. Da die VND die besseren Ergebnisse liefert sind in Abbildung 6.25 die Lösungen für die unterschiedlichen Seitentypen von *VND(page-zuf)* dargestellt.

Wie bei allen anderen Instanzen gibt es auch hier kleine Unterschiede in den Ergebnissen der Instanzen mit d=1 und jenen Instanzen mit d=2. Bei den Startlösungen sind noch keine Abweichungen zu erkennen. Die VND liefert jedoch für Instanzen mit d=2 Lösungen die ungefähr um 3% bis 4% schlechter als die für d=1 sind. Bei den Hybridlösungen tritt, im Vergleich zu den vorherigen Instanzen, ein umgekehrtes Verhalten auf. In diesem Fall sind die Lösungen der Instanzen mit d=2 um etwa 2% besser als die für d=1. Das liegt daran: Die Hybridmethoden verschlechtern die Ergebnisse der VND für alle Instanzen mit zwei, drei und vier Rissen. Da aber die Instanzen mit d=2 einen größeren Toleranzbereich für die Seitenkanten aufweisen, gibt es auch mehrere Möglichkeiten an *gültigen* Zuordnungen. Dadurch werden weniger Kanten von der VND Lösung verändert, wodurch auch die Verschlechterungen geringer ausfallen.

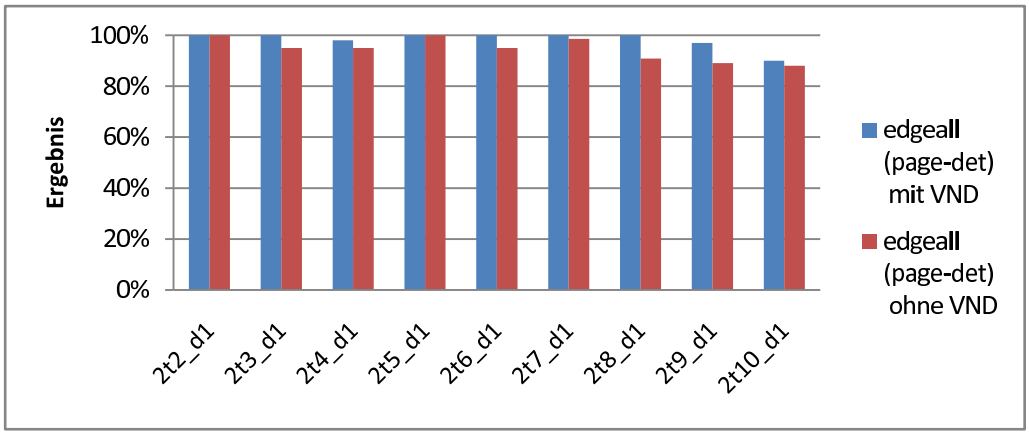


Abbildung 6.26: Lösungen der Hybridmethoden mit/ohne VND für Instanzen mit 2 Rissen

Die VND liefert für die Instanzen, die Seiten mit drei unterschiedlichen Rissen beinhaltet, Lösungen innerhalb von einer Minute, was wieder auf die unterschiedlich großen Schnipsel und der dadurch geringeren Austauschmöglichkeiten zurückzuführen ist. Bei den Hybridmethoden gibt es hingegen wieder eine sehr große Streuung der benötigten Zeiten. Sie liegen zwischen einigen Sekunden und manchmal auch über 15 Minuten.

Einfluss der VND auf die Lösungsqualität

Zum Abschluss betrachten wir noch, welchen Einfluss die Anwendung der VND auf die Lösungsqualität der Hybridmethoden hat. Dazu wurden Testläufe durchgeführt, bei denen die Hybridmethoden direkt auf die Startlösungen angewandt wurden. Für die Instanzen mit zwei Rissen zeigt sich dabei, dass die Ergebnisse der Hybridmethoden durch die Verwendung der VND gleich gut oder oftmals sogar besser sind (siehe Abbildung 6.26).

Bei den Instanzen mit drei Rissen sind kaum Abweichungen zwischen den Lösungen zu bemerken. Wobei die Lösungen ohne VND geringfügig schlechter oder höchstens gleich gut sind wie jene, die mit der VND erzeugt wurden. Für die Instanzen mit zwei und drei Rissen sind die Ergebnisse, die Mithilfe der VND ermittelt wurden, wieder eindeutig besser. Eine Gegenüberstellung der beide Lösungen ist in Abbildung 6.27 dargestellt.

Somit zeigt sich, dass die Hybridmethoden bessere Ergebnisse liefern, wenn sie auf die von der VND verbesserten Startlösungen angewandt werden.

6.3 Zusammenfassung

Zusammenfassend kann man sagen, dass der Algorithmus `astp` die Startlösungen durch die VND verbessert und die anschließende Anwendung einer Hybridmethode eine weitere Erhöhung der Lösungsqualität liefert. Wobei für Instanzen, die nur Seiten mit zwei Rissen bein-

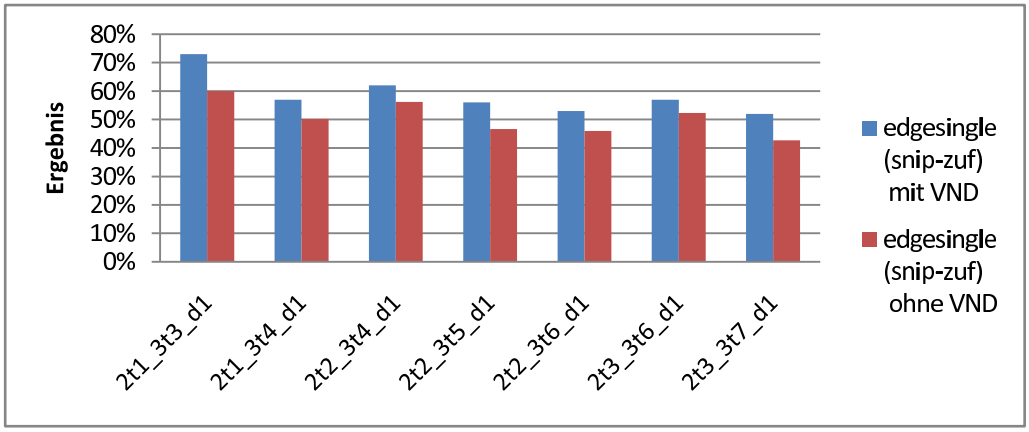


Abbildung 6.27: Lösungen der Hybridmethoden mit/ohne VND für Instanzen mit 2 und 3 Rissen

halten, sehr schnell sehr gute Ergebnisse gefunden werden. Für bis zu acht Seiten erhält man beinahe immer eine optimale Zuordnung und bei zehn Seiten noch immer einen Wert von 90%. Für Instanzen, die Seiten mit drei Rissen enthalten, geht die Qualität der Lösung deutlich zurück und verschlechtert sich für Seiten mit vier Rissen noch weiter. Der Rückgang der Lösungsqualität ist bei der VND auf die größere Anzahl von Schnipsel und die daraus resultierende höhere Ungenauigkeit von den Referenzwerten zurückzuführen. Die schlechteren Ergebnisse der Hybridvarianten beruhen auf den größer werdenden Toleranzbereich der Seitenkanten und der dadurch steigenden Anzahl an möglichen Zuordnungen, die als *gültig* erkannt werden. Bei den Ergebnissen für Instanzen mit unterschiedlicher Anzahl von Rissen zeigt sich, dass auch hier der Algorithmus für die Seiten mit nur zwei Rissen eine sehr gute Zuordnung liefert. Für Seiten mit mehr Rissen sinkt jedoch die Lösungsqualität wieder.

Beobachtet wurde auch, dass die zufälligen Startlösungen für Instanzen, mit der gleichen Anzahl von Rissen, immer besser als die deterministischen sind. Denn hier kommt es durch die Methode der Schnipselzuordnung dazu, dass einige Seiten eine Gesamtfläche besitzen, die eindeutig größer ist als die Referenzfläche und andere wiederum ein Fläche, die eindeutig kleiner ist. Bei Instanzen mit einer unterschiedlichen Anzahl von Rissen sind hingegen die deterministischen Startlösungen besser. Durch die vorgegebene Zuordnung werden die größeren Schnipsel den Seiten mit weniger Rissen zuordnet und die kleineren den Seiten mit mehr Rissen. Auf den weiteren Lösungsverlauf hat die Qualität der Startlösungen aber keinen Einfluss.

Bei den VND-Methoden wurden bei den Instanzen, die entweder Seiten mit drei Rissen oder Seiten mit vier Rissen beinhalten, in den Ergebnissen keine Unterschiede beobachtet. Für alle anderen Instanzen liefert die *page* Methode die besseren Ergebnisse, weshalb diese Methode verwendet werden sollte.

Die vier Hybridmethoden liefern für alle Instanzen, die nur Seiten mit der gleichen Anzahl von Rissen beinhalten, unabhängig von der Qualität der Lösung auf welche sie angewandt

wurden, Ergebnisse die sich nur minimal voneinander unterscheiden. Für Instanzen, die Seiten mit unterschiedlichen Risszahlen beinhalten, sind die Lösungen der Hybridmethoden, welche auf die *VND(page)* Ergebnisse angewandt wurden leicht besser. Beim Vergleich der vier Methoden wurde nur bei den Instanzen mit drei unterschiedlichen Rissen eine Abweichung in den Lösungen festgestellt. Bei diesen Testfällen liefern die *snipsingle* und *edgesingle* Methode leicht besser Ergebnisse. Für alle anderen Instanzen kann keine Aussage getroffen werden, welche Methode die besseren Ergebnisse liefert. Aus diesem Grund sollte eine der *single* Methoden, angewandt auf die *VND(page)* Lösung, bevorzugt werden. Es sei noch erwähnt, dass für Instanzen mit drei unterschiedlichen Rissen, die Hybridmethoden geringfügig schlechtere Ergebnisse liefert als die VND und es deswegen besser ist, für solche Instanzen nur die VND zu verwenden.

Weiters wurde beobachtet, dass bei einer größeren Streuung beim Schereffekt (`d`=2) die Lösung des TPP wegen größerer Ungenauigkeiten schwieriger wird und aus diesem Grund die Qualität der Lösungen sinkt.

Für die VND gilt folgender Zusammenhang bezüglich der Zeiten: Bei Instanzen mit nur einer Risszahl steigt mit der Anzahl der Risse und Schnipselzahl auch die benötigten Zeiten. Sie liegen bei kleinen Instanzen bei etwa einer Minute und steigen bei den größeren auf 10 Minuten an. Aufgrund der höheren Komplexität bei einer größeren Anzahl von ähnlichen Schnipseln, erreicht die VND ab etwa 60 Schnipsel die maximal erlaubten Iterationen (gegeben durch den Parameter `m`). Für Instanzen mit unterschiedlichen Risszahlen ist wegen der unterschiedlich großen Schnipsel die Zuordnung einfacher, da nicht soviel Austauschmöglichkeiten existieren. Dadurch liefert die VND für diese Instanzen wieder Lösungen innerhalb einer Minute.

Die Zeiten der Hybridmethoden variieren sehr stark. Für die kleineren Instanzen mit zwei Rissen (bis zu sechs Seiten) wird nur zirka eine Minute benötigt, ab sieben Seiten steigen jedoch die Zeiten mit jeder zusätzliche Seite rapide an und erreicht bei zehn Seiten über 15 Minuten. Bei den Instanzen mit drei Rissen liegen die Zeiten für zwei Seiten bei 3 Minuten und steigen für zehn Seiten auf über 15 Minuten. Für Instanzen mit vier Rissen sinken jedoch für alle Lösungen die Zeiten wieder unter eine Minute, da durch den größeren Toleranzbereich schneller *gültige* Lösungen gefunden werden. Die Zeiten für Instanzen mit unterschiedlichen Risszahlen weisen eine sehr hohe Streuung auf und liegen zwischen einigen Sekunden und Werten über 15 Minuten.

Beim Vergleich der Lösungen die mit oder ohne Anwendung der VND erzielt wurden zeigt sich, dass die Hybridmethoden bessere Ergebnisse liefern, wenn sie auf die von der VND verbesserten Startlösungen angewandt werden.

7 Schlussfolgerungen

In dieser Arbeit werden drei Lösungsansätze für das *Tearing Paper Problem* (TPP) vorgestellt: *Lokale Suche* und *Variable Neigborhood Descent* (VND) versuchen durch Vertauschen von Schnipseln zwischen einzelnen Seiten die Gesamtfläche, den Umfang und die Winkelsumme der Seiten möglichst gut an berechnete Referenzwerte anzunähern. Bei der *Hybridmethode* werden durch die Kombination eines exakten Algorithmus und einer entsprechenden Heuristik die Schnipsel so den Seitenkanten zugeordnen, dass die Abweichungen der Kantenlängen innerhalb eines bestimmten Toleranzbereichs liegen.

Es wurde eine Vielzahl von Testläufen, mit unterschiedlichen Instanzen durchgeführt. Aus den erhaltenen Ergebnissen können die folgenden Schlüsse gezogen werden: Die Ergebnisse der Hybridmethode sind abhängig von ihren Startlösungen. Je besser diese sind, umso höher ist auch die Qualität der Hybridlösungen. Die besten Resultate für die Instanzen erzielt man somit, wenn auf die Lösungen der VND anschließend noch die Hybridmethode angewandt wird. Für die VND zeigt sich, dass ihre Ergebnisse davon abhängen, wie sehr sich die gegebenen Schnipsel ähneln. Je mehr sich die Form der Schnipsel von einer Instanz unterscheiden, desto besser sind die Resultate der VND. Lokale Suche hingegen liefert für das TPP keine entsprechenden Verbesserungen.

Betrachtet man die Ergebnisse von den unterschiedlichen Instanzen zeigt sich Folgendes: Für Seiten die nur zweimal zerrissen wurden, werden exzellente Ergebnisse geliefert. Bei drei Rissen pro Seite ist die erhaltene Lösungsqualität nicht mehr so gut wie für zwei Risse. Aber auch hier werden eindeutige Verbesserungen der Ausgangslösungen erzielt. Die Ergebnisse mit vier Rissen pro Seite zeigen, dass man für diese Instanzen keine eindeutige Verbesserung der Startlösung erreichen kann. Es sind somit nur für zwei Risse pro Seiten die geometrischen Informationen der Schnipsel ausreichend, um eine sehr gute Zuordnung zu erhalten. Mit steigender Zahl von Rissen pro Seite und der dadurch größeren Anzahl von ähnlichen Schnipseln, wird es schwieriger mit den gegebenen geometrischen Daten eine gute Zuordnung der Schnipsel zu Seiten zu ermitteln.

Um jedoch auch für Seiten, die mit mehr als zwei Rissen zerstört wurden sehr gute Ergebnisse zu erhalten, ist für weiterführenden Arbeiten folgender Ansatz von Interesse: Eine Erweiterung der in dieser Arbeit vorgestellten Methoden mit zusätzlichen Schnipselinformationen (Papierfarbe, Schriftfarbe, ..), um dadurch die Zusammengehörigkeit der Schnipsel eindeutiger zu ermitteln. Interessant wäre auch ein halbautomatisches Verfahren, dass Menschen den Eingriff in den Optimierungsprozess ermöglicht. Hier könnten Menschen verschiedene Zuordnungen durchführen, welche für einen Algorithmus sehr schwierig oder gar nicht zu erkennen sind.

Literaturverzeichnis

[1] Schüller, P.:, *Reconstructing borders of manually torn paper sheets using Integer Linear Programming*. Diplomarbeit, Technische Universität Wien, Österreich (2008)

[2] Chung, M.G., Fleck, M., Forsyth, D.: *Jigsaw puzzle solver using shape and color*. Fourth International Conference of on Signal Processing 1998, ICSP'98. Volume 2. (1998) 877-880

[3] Skeoch, A.: *An Investigation into Automated Shredded Document Reconstruction using Heuristic Search Algorithms*. PhD thesis, University of Bath, United Kingdom (2006)

[4] Ukovich, A., Ramponi, G., Doulaverakis, H., Kompatsiaris, Y., Strintzis, M.: *Shredded document reconstruction using MPEG-7 standard descriptors*. Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004. (2004) 334-337

[5] Ukovich, A., Ramponi, G.: *Features for the reconstruction of shredded notebook pager*. IEEE International Conference on Image Processing 2005, ICIP(3) (2005) 93-96

[6] Ukovich, A., Zacchigna, A., Ramponi, G. Schoier, G.: *Using clustering for document reconstruction*. In Dougherty, E.R., *et al*, eds.: Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning. Volume 6064 of Proceedings of SPIE, International Society for Optical Engineering (2006) 168-179

[7] Morandell, W.: *Evaluation and reconstruction of strip-shredded text documents*. Diplomarbeit, Technische Universität Wien, Österreich (2008)

[8] Prandtstetter, M., Raidl, G.R.: *Combining forces to Reconstruct Strip Shreeded Text Documents*. In M.J. Blesa et al., editors, Hybrid Metaheuristics 2008, Volume 5269 of LNCS, pages 175-189, Oktober 2008. Springer-Verlag Berlin Heidelberg. (2008)

[9] Justino, E., Oliveira,L.S., Freitas, C.: *Reconstructing shredded documents through feature matching*. Forensic Sience International 160(2-3) (2006) 140-147

[10] De Smet, P.: *Reconstruction of ripped-up documents using fragement stack analysis procedures*. Forensic sience international 176(2) (2008) 124-136

[11] Güting R.H, Dieker, S.: *Datenstrukturen und Algorithmen 3. Auflage*. Stuttgart [u.a.]: Teubner (2004), 141-152

[12] Aarts, E., Lenstra, J.K.: *Local Search in Combinatorial Optimization*. Chichester [u.a.]: Wiley (1998), 57-90

[13] Hansen, P., Mladenovic, N.: *A Tutorial on Variable Neighborhood Search*, Les Cahiers du GERAD, HEC Montréal and GERAD, G-2003-46, Canada (2003)

[14] Puchinger, J., Raidl, G.R.: *Bringing order into the neighborhoods: Relaxation guided variable neighborhood search*. Technical Report TR 186-1-06-02, Technische Universität Wien (2006)

[15] CGAL, *Computational Geometry Algorithms Library*. http://www.cgal.org. (26.05.2008)

[16] Kuhn, H.W.: *The Hungarian Method for the assignment problem*. Naval Research Logistics Quarterly, Volume 2 (1955) 83-97

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

A Anhang

Der Algorithmus `astp` wird mit dem folgenden Kommandozeilenbefehl aufgerufen:

```
astp -i <inputFile> -a <assignMode> [ -hvs -n <neighborMode>
     -p <stepMode> -e <hybridMode> -r <maxNoImprHeuristic>
     -m <maxItrHeuristic> -w <worseningHeuristic> -t <gap>
     -b <maxNoImprHybrid> -x <maxItrHybrid> -o <worseningHybrid>
     -c <maxNoChange> -g <edges> ]
```

Die wichtigsten Argumente sind:

`-i <inputFile>` Angabe der Eingabedatei, welche die Instanz im XML-Format beinhaltet.

`-s` Es wird eine zufällige Startlösung erzeugt (Standard: deterministische Startlösung).

`-a <assignMode>` Bestimmt mit welchem Lösungsansatz die Zuordung durchgeführt wird:
- `localsearch:` Lokale Suche
- `vnd:` Variable Neighborhood Descent VND
- `hybrid:` Hybridmethode

`-n <neighborMode>` Legt fest, welche Nachbarschaft in der Lokalen Suche oder VND verwendet wird:
- `snip:` Nachbarschaft $\mathcal{N}_{snip}$
- `page:` Nachbarschaft $\mathcal{N}_{page}$
- `best:` Nachbarschaft $\mathcal{N}_{best}$

`-p <stepMode>` Angabe der Schrittfunktion, die in der Lokalen Suche verwendet wird:
- `area:` Schrittfunktion f^{A}_{step}
- `edge:` Schrittfunktion f^{L}_{step}
- `angle:` Schrittfunktion f^{W}_{step}

`-e <hybridMode>` Bestimmt welche Hybridvariante benutzt wird:
- `snipsingle:` SnipSingle
- `snipall:` SnipAll
- `edgesingle:` EdgeSingle
- `edgeall:` EdgeAll

`-r <maxNoImpr>` Legt für die Lokale Suche und VND die Anzahl der Iterationen fest, in der eine Verbesserung erfolgen muss.

`-m <maxItr>` Gibt die maximale Anzahl von Iterationen für die Lokale Suche und VND an.

`-w <worsening>` Bestimmt die maximal erlaubte Verschlechterung, die bei einer Schnipselvertauschung in der Lokalen Suche und VND entstehen darf.

`-t <distance>` Legt fest, um wieviel sich durch einen Riss der Umfang einer Seite maximal vergrößert.

`-b <maxNoImprHybrid>` Für die Hybridvarianten `snipsingle` und `edgesingle` wird die Anzahl der Iterationen angegeben, in der eine Verbesserung erfolgen muss. Für die Varianten `snipall` und `edgeall` legt dieses Argument die Anzahl der Iterationen fest, bei denen in der Menge der noch zu verbessernden Kanten eine weiter Vertauschung gefunden werden muss.

`-x <maxItrHybrid>` Maximale Anzahl von Iterationen der Hybridmethode.

`-o <worseningHybrid>` Bestimmt die maximal erlaubte Verschlechterung, die bei einer Schnipselvertauschung in der Hybridmethode entstehen darf.

`-c <maxNoChange>` Legt für die Hybridvarianten `snipall` und `edgeall` die Anzahl der Iterationen fest, in der mindestens ein Schnipselaustausch erfolgen muss.

`-g <edges>` Bestimmt die Anzahl der zufällig ausgewählten Kanten/Schnipsel, für die in der Hybridmethode Nachbarn ermittelt werden.