Combining Forces to Reconstruct Strip Shredded Text Documents

Matthias Prandtstetter and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{prandtstetter,raidl}@ads.tuwien.ac.at

Abstract. In this work, we focus on the *reconstruction of strip shredded text documents* (RSSTD) which is of great interest in investigative sciences and forensics. After presenting a formal model for RSSTD, we suggest two solution approaches: On the one hand, RSSTD can be reformulated as a (standard) traveling salesman problem and solved by well-known algorithms such as the chained Lin Kernighan heuristic. On the other hand, we present a specific variable neighborhood search approach. Both methods are able to outperform a previous algorithm from literature, but nevertheless have practical limits due to the necessarily imperfect objective function. We therefore turn to a semi-automatic system which also integrates user interactions in the optimization process. Practical results of this hybrid approach are excellent; difficult instances can be quickly resolved with only few user interactions.

1 Introduction

In the fields of forensics and investigative sciences it is often required to reconstruct the information hidden on destructed paper documents. Usually, paper is destroyed by ripping up the sheets or—more professionally—by using appropriate shredding devices either producing thin strips or even small rectangles or other geometric shapes like hexagons. In this work we focus on the topic of reconstructing strip shredded text documents.

Depending on the shape, size, and the number of remnants the process of reconstructing an original document in order to restore the lost information can be very time consuming or practically almost impossible for a human. Therefore, an automatic reconstruction process is desirable. Any such approach has to acquire the strips in a first step by scanning the remnants using a (high end) scanner. Pattern recognition and image processing tasks are applied to identify the bounding boxes and orientations of the scanned strips and to gather information about features like background/paper color, text color, and other helpful features. In a second step, these attributes can be used to derive clusters of strips potentially belonging to the same original document page(s) [1]. Unfortunately any such system suffers from two drawbacks: Firstly, after the clustering process no information is directly available on how the strips have to be concatenated to form the original page(s). Secondly, any clustering approach can only marginally reduce the problem size or even fails if many pages containing the same or similar features are shredded; examples are forms, tables, and any other regularly structured document.

Motivated by these two drawbacks, we propose a new approach to the *reconstruction of strip shredded text documents* (RSSTD) by firstly specifying the problem as a combinatorial optimization problem and secondly reformulating it as the well known *traveling salesman problem* (TSP). Furthermore, to overcome problems implied by the special structure of the resulting TSP and unavoidable inaccuracies introduced by the general modeling, a new *variable neighborhood search* (VNS) that is embedded in a system allowing user interaction is presented. Our practical results show that this approach combines and leverages machine power and human experience, knowledge, and intuition in an effective way, enabling the resolution of larger and/or more difficult RSSTD instances.

This article is structured as follows: In the next section an overview on previous and related work is given. Afterwards, our problem is formally specified. In Section 4 the transformation to the TSP is described, and Section 5 discusses possible definitions of the cost function related to the formulation as combinatorial optimization problem. Then two approaches for solving the given problem are presented—one based on the well known Lin Kernighan heuristics for the TSP and one based on a VNS and a system for integrating human interaction. Section 7 discusses results obtained by using our methods. Conclusions are drawn in Section 8.

2 Related and Previous Work

Although RSSTD is of great interest not only for intelligence agencies or forensics but also for different scientific communities, there exists not much work covering exactly this topic. A related but at the same time very different challenge is the automated solving of *jigsaw puzzles*. The major difference is the fact that for jigsaw puzzles each piece has a mostly unique shape and therefore the pure geometric information of an element can be exploited well in the reconstruction process. Furthermore and in contrast to most text documents, the image and color information on the puzzle pieces can be utilized efficiently [2].

Another related topic is the *reconstruction of manually torn paper documents*. There, shape information can also be exploited to some degree but may also be misleading due to shearing effects. The first of three major approaches was presented by Justino *et al.* [3]. They extract characteristics of the edges of snippets and then try to cling them together by iteratively matching the extracted features [3]. They state in their work that the application of the proposed method is limited to small instances of up to 15 snippets from one page.

In his master thesis, Schüller [4] proposed to use *integer linear programming* based methods for exactly reconstructing manually torn documents. The techniques presented in this work rely only on geometric information extracted from the remnants and solely focus on the borders of pages to be reconstructed since border pieces provide more reliable information and are easier to handle. Again, the application of the algorithms is limited to small instances.

De Smet [5] tries to exploit information implied by the relative order of snippets in a stack of recovered remnants. The proposed methods are limited to scenarios without missing snippets as well as a perfect snippet order. No details on how to adapt the solution process to non perfect situations are given.

In contrast to the above mentioned methods, Skeoch [6] focuses on the reconstruction of strip shredded documents but mainly discusses the scanning process and related properties of paper strips. Further, she presents a *genetic algorithm* including crossover and mutation operators as well as heuristics for generating initial solutions to restore shredded images. In contrast to text documents, a large amount of different colors usually exists in images and soft color transitions dominate. This aspect can be efficiently exploited.

Ukovich *et al.* [7] tried not to reconstruct the original document pages but to build clusters of strips belonging to the same sheet of paper by using MPEG-7 descriptors for this task. In [1], they introduced among others features like background and text color, line spacing and number of lines to be extracted from documents and discussed the potential of clustering methods.

Lately, Morandell [8] formulated the RSSTD as a combinatorial optimization problem related to the TSP. He also presents basic ideas on how to solve this new formulation by means of metaheuristics including variable neighborhood search, iterated local search, and simulated annealing. The results presented within this thesis are promising and encouraged us to pursue this approach in more detail.

3 Formal Problem Specification

In this section, we present a formal problem description of RSSTD as a combinatorial optimization problem.

We are given a finite set $\mathcal{S}$ of n rectangular shaped and (almost) equally sized paper snippets—so called strips—which have been produced by shredding one ore more sheet(s) of paper. In this work the widths of the strips are not further investigated since no information exploited in our approach can be extracted from them. Furthermore, the heights of all strips are assumed to be the same. If this is not the case, then a preprocessing step using clustering methods as proposed in [1] can be performed. Each set of strips having the same heights in the resulting partitioning can be used as input for our approach to RSSTD.

Although many printers are capable of duplex printing nowadays, most documents—especially in offices, one of the main application areas of shredders—are still blank on the back face. Motivated by this observation and for simplicity our presented model only regards the front face of the scanned strips. However, an extension to handle two-sided documents is possible in a straightforward way. Further, we neglect all strips of any input instance with no useful information on them. That is, all completely blank strips as well as strips with blank borders but non-empty inner regions are eliminated. Applying such a blank strip elimination procedure has two advantages. Firstly, symmetries implied by arbitrarily swapping blank strips are removed, and secondly—and more importantly—the search space is significantly reduced.

A solution $x = \langle \pi, o \rangle$ to RSSTD consists of a permutation $\pi : \mathcal{S} \to \{1, \dots, n\}$ of the elements in set $\mathcal{S}$ as well as a vector $o = \langle o_1, \dots o_n \rangle \in \{\text{up}, \text{down}\}^n$ which assigns an orientation to each strip $s \in \mathcal{S}$:

$$o_s = \begin{cases} \text{up} & \text{if strip } s \text{ is to be placed in its original orientation,} \\ \text{down} & \text{if strip } s \text{ is rotated by } 180°. \end{cases} \tag{1}$$

While π_i denotes the strip at position i, $i = 1, \dots, n$, we denote the position of a given strip $s \in S$ by $p_s \in \{1, \dots, n\}$; i.e. $\pi_i = s \leftrightarrow p_s = i$. By $\sigma = \langle s_j, \dots, s_k \rangle$, with $1 \leq j, k \leq n$, we denote a possibly empty (sub-)sequence of strips in a given solution. Two sequences are concatenated by the $\cdot$ operator.

In the following we make use of a cost function $c(s, s', o_s, o_{s'}) \geq 0$ to be explained later in detail, which shall provide an approximate measure for the likelihood that two strips s and s' appear side-by-side and oriented according to o_s and $o_{s'}$ in the original document, i.e. correct solution. A value of zero indicates that the contacting borders match perfectly; the larger the cost value, the more different are these borders. The overall objective is to find a solution, i.e. permutation and corresponding orientation vector, such that the following total costs are minimized:

$$\text{obj}(x) = \text{obj}_l + \sum_{i=1}^{n-1} \text{c}(\pi_i, \pi_{i+1}, o_i, o_{i+1}) + \text{obj}_r \tag{2}$$

$$\text{obj}_l = \text{c}(\beta, \pi_1, o_\beta, o_1) \tag{3}$$

$$\text{obj}_r = \text{c}(\pi_n, \beta, o_n, o_\beta) \tag{4}$$

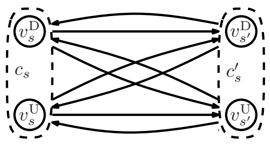
Hereby β denotes an additional (artificial) blank strip which is inserted at the beginning and the end of the page(s) to be reconstructed. This is motivated by the fact, that in most cases—especially if all strips of the original sheets of paper have been recovered—the left and right document margins are blank. As the costs of matching two blank borders are zero, omitting the additional terms obj_l and obj_r would most likely lead to a solution where the first and last strips of a correct solution are placed side-by-side. Since strip β is blank, its orientation o_β does not have any impact.

One crucial part in solving RSSTD as stated above is a proper definition of the cost function $c(s, s', o_s, o_{s'})$. A detailed discussion on this topic is given in Section 5. In any case, a cost function used for RSSTD has to have the so called *skew-symmetry property* which states that the costs for placing strip s' right to strip s have to be the same as for rotating both strips by 180° and placing strip s right to strip s'.

Before considering approaches for solving RSSTD, we show the following complexity result.

Theorem 1. *RSSTD is $\mathcal{NP}$-hard.*

Proof. Any (symmetric) *traveling salesman problem* (TSP) instance can be transformed into a RSSTD instance by introducing a strip for each city and defining



(a)

	v_s^U	v_s^D	$v_{s'}^U$	$v_{s'}^D$
v_s^U	—	—	a	b
v_s^D	—	—	c	d
$v_{s'}^U$	d	b	—	—
$v_{s'}^D$	c	a	—	—

(b)

	v_s^U	v_s^D	$v_{s'}^U$	$v_{s'}^D$
v_s^U	—	0	c_{+M}	d_{+M}
v_s^D	0	—	a_{+M}	b_{+M}
$v_{s'}^U$	c_{+M}	a_{+M}	—	0
$v_{s'}^D$	d_{+M}	b_{+M}	0	—

(c)

Fig. 1. In (a) a subgraph representing two strips s and s' in an AGTSP instance is depicted while in (b) the same subgraph after performing the transformation to TSP is shown. The bold lines indicate two corresponding tours.

the cost function $c(s, s', o_s, o_{s'})$ in correspondence to the TSP's distances; orientations are ignored. An arbitrary city can be chosen as RSSTD's artificial blank strip β corresponding to the left and right margins. An optimal solution to the RSSTD instance obtained in this way obviously will also correspond to an optimal solution of the original TSP. $\square$

4 Reformulation as Traveling Salesman Problem

In this section, we present a polynomial time transformation from for the RSSTD into a TSP, thus the reverse direction than in the proof above, with the motivation to find RSSTD solutions via algorithms for the TSP. To achieve this, a representation of RSSTD as an *asymmetric generalized traveling salesman problem* is developed first, and in a second step, we transform this problem into a TSP.

4.1 Formulation as Asymmetric Generalized Traveling Salesman Problem

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

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

1. Introduce a cluster C_s for each strip $s \in \mathcal{S}$ consisting of two vertices v_s^{U} and v_s^{D} representing the possible orientations of the corresponding strip s.
2. Introduce a cluster C_β for the virtual blank strip β and insert one vertex v_β into this cluster. Since β is blank no orientation information is necessary for this strip.
3. Each pair (s, s') of strips induces eight arcs representing the possible placements of s and s' in relation to each other, see also Fig. 1a. For instance, arc $(v_s^{\mathrm{D}}, v_{s'}^{\mathrm{U}})$ represents the case that strip s' is placed right to strip s. While strip s is rotated by 180°, strip s' is positioned upright. Since strip s cannot

be placed left (or right) to itself, it is obvious that there are no arcs between two nodes representing the same strip.

4. Additionally, vertex v_β is connected via two reversely directed arcs with each other node representing a strip.
5. The weights of the arcs are chosen such that for any arc $a = (v_s^{o_s}, v_{s'}^{o_{s'}})$, with $s, s' \in \mathcal{S}$, $w_a = \mathrm{c}(s, s', o_s, o_{s'})$. The weights for arcs leaving or entering v_β are chosen according to $\mathrm{c}(\beta, s, o_\beta, o_s)$ or $\mathrm{c}(s, \beta, o_s, o_\beta)$, respectively.

Obviously, an optimal solution to the AGTSP instance derived in the described way also forms a solution to the original RSSTD instance with equal costs when starting the tour at the virtual strip represented by v_β.

Several methods for solving AGTSP already exist like exact approaches, e.g. a branch-and-cut algorithm [9], as well as metaheuristics, e.g. a genetic algorithm [10]. Beside applying one of those algorithms specifically designed for solving AGTSP another possibility is to transform an AGTSP instance into a classical TSP instances and solve the latter with one of the many existing methods. In the next section we concentrate on such an approach.

4.2 Further Reformulation as TSP

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

Based on the presented transformation of RSSTD to AGTSP, RSSTD can be further translated into a TSP by first applying the polynomial time transformation into a *asymmetric traveling salesman problem* (ATSP) proposed in [11] and finally applying the polynomial transformation of ATSP into TSP described in [12]. Taking a closer look at these works, two major drawbacks can be identified. On one hand, the maximum costs for edges are dramatically increased during the transformation from AGTSP into ATSP, which might lead to practical problems when trying to solve such transformed instances. On the other hand, the number of nodes in G is doubled during the translation from the asymmetric TSP to the symmetric case. Fortunately, both drawbacks can be avoided when applying a new transformation method we specifically developed for RSSTD.

Each instance of RSSTD can be transformed into an instance of TSP when first applying the reformulation as ATSP presented above and then executing the following steps. For this we adopt the idea of introducing directed cycles of zero costs within each cluster while changing the (costs of the) outgoing arcs as suggested by Behzad *et al.* in [11]:

1. We add two additional arcs—one in each direction—between nodes v_s^D and v_s^U for each strip $s \in \mathcal{S}$.
2. The weights of these new arcs are all set to zero.
3. In a next step, we swap the weights for $(v_s^{\mathrm{D}}, v_{s'}^{\mathrm{D}})$ and $(v_s^{\mathrm{U}}, v_{s'}^{\mathrm{D}})$ as well as $(v_s^{\mathrm{D}}, v_{s'}^{\mathrm{U}})$ and $(v_s^{\mathrm{U}}, v_{s'}^{\mathrm{U}})$. After swapping two arcs we add a constant M to the associated arc weights.

4. Since the cluster C_β consists of only one node, no transformation needs to be done for this cluster.

In Figure 1b the adjacency matrix of a subgraph of an AGTSP instance for RSSTD is presented. Figure 1c depicts the adjacency of this subgraph after applying the transformation to TSP. It can be easily checked that the resulting graph is undirected.

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

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

5 Definition of a Cost Function

One crucial point in RSSTD is the definition of an appropriate cost function $c(s, s', o_s, o_{s'})$ for judging the likelihood, that two strips s and s' match under their given orientations o_s and $o_{s'}$. There are several different ways on how this can be done (see also [8] on this topic), and none will be perfect in any possible situation. In this section, we discuss some important aspects on how to design a meaningful cost function for RSSTD.

As already mentioned above, any cost function for RSSTD needs to have the skew-symmetry property, i.e. placing strip s' right to strip s has to be as expensive as placing strip s right to strip s' but both rotated by 180°. To simplify the process of computing (good) lower bounds on RSSTD, we demand $c(s, s', o_s, o_{s'}) \geq 0$ always holds.

Since it is unlikely that the images of two strips with the same physical height and scanned with the same resolution significantly differ in the number of pixels

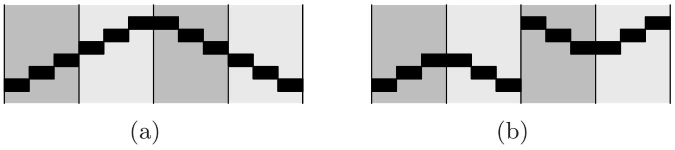


Fig. 2. Both solutions might be correct, but (a) is more likely

along the vertical edges, we assume for this work, that the number of pixels h_s along the y-axis is the same for all strips.

To simplify the next definitions, we consider eventual rotations of strips in the following as already performed; i.e. when speaking about the left side of a strip s for which $o_S =$ down, we actually refer to its original right side. The pixels on the left or right edge are those pixels which form the left or right border, respectively.

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

Let $v_\mathrm{l}(s, y, o_s), v_\mathrm{r}(s, y, o_s) \in \{0, 1\}$ be the black-and-white values of the y-th pixel at the left and right borders of strip s under orientation o_s, respectively.

The first and most straightforward approach for defining a cost function $c_1(s, s', o_s, o_{s'})$ is by simply iterating over all pixels on the right border of strip s and compare it to the corresponding pixel on the left border of strip s'. Since we defined RSSTD as a minimization problem the value of $c_1(s, s', o_s, o_{s'})$ is increased by one if two corresponding pixels do not have the same values:

$$c_1(s, s', o_s, o_{s'}) = \sum_{y=1}^{h_s} |v_\mathrm{r}(s, y, o_s) - v_\mathrm{l}(s', y, o_{s'})| \tag{5}$$

The evaluation of this cost function can be performed efficiently, but there are some situations in which it returns misleading information. For an example see the cases depicted in Figs. 2a and 2b. Of course, it is not possible to automatically decide which of the two alignments always is the correct one. Nevertheless, the situation in Fig. 2a is intuitively much more likely. Therefore, we want this alignment to receive a better evaluation than the arrangement of Fig. 2b. Hence, we adopt the idea presented in [13] to additionally consider the values of two pixels above and two pixels below to the currently evaluated position:

$$c_2(s, s', o_s, o_{s'}) = \sum_{y=3}^{h_s-2} p(s, s', o_s, o_{s'}, y) \tag{6}$$

$$p(s, s', o_s, o_{s'}, i) = \begin{cases} 1 & \text{if } p'(s, s', o_s, o_{s'}, i) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\begin{aligned} p'(s, s', o_s, o_{s'}, i) = |&0.7 \cdot v_r(s, o_s, i) - 0.7 \cdot v_l(s', o_{s'}, i) \\ &+ 0.1 \cdot (v_r(s, o_s, i+1) - v_l(s', o_{s'}, i+1)) \\ &+ 0.1 \cdot (v_r(s, o_s, i-1) + v_l(s', o_{s'}, i-1)) \\ &+ 0.05 \cdot (v_r(s, o_s, i+2) + v_l(s', o_{s'}, i+2)) \\ &+ 0.05 \cdot (v_r(s, o_s, i-2) + v_r(s', o_{s'}, i-2))| \end{aligned} \tag{8}$$

The threshold value τ used in the definition of $p(s, s', o_s, o_{s'}, i)$ has to be chosen carefully. A good value, in particular also for handling the special case depicted in Fig. 2, is 0.1.

6 Solving RSSTD

In this section we present our concrete solution approaches for RSSTD.

6.1 Solving RSSTD Via Its Reformulation as a TSP

Using the transformation of RSSTD to TSP as presented in Section 4.2 and cost function $c_2(s, s', o_s, o_{s'})$ defined in Section 5 it is obvious to apply approaches developed for the TSP on RSSTD. Since the number of nodes in the graph underlying the TSP is always twice the number of strips in the original RSSTD instance and this number can be quite large exact algorithms might not be applicable for real world instances. Therefore, we decided to use the implementation of Applegate *et al.* [14] of the Chained Lin-Kernighan heuristic [15] for solving the transformed RSSTD. Detailed results are presented in Section 7.

6.2 Solving RSSTD Via VNS and Human Interaction

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

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

The idea of systematically integrating human interaction in an optimization process is not new. Klau *et al.* [16,17] give a survey on such approaches and present a framework called *Human Guided Search* (HuGS). The implementation is primarily based on tabu search, and the success of this human/metaheuristic integration is demonstrated on several applications.

Variable Neighborhood Search in HuGS. Since preliminary tests for solving RSSTD with tabu search as implemented in HuGS did not convince, we considered also other metaheuristics and finally decided to use a *(general) variable neighborhood search* (VNS) [18] with embedded *variable neighborhood descent* (VND) for local improvement. VNS is a metaheuristic based on the general observation that the global optimum always has to be a local optimum with respect to any possible neighborhood. The key-idea is to perform a local search and switch between multiple neighborhood structures in a well-defined way, whenever a local optimum has been reached. For more details on the general algorithm we refer to [18].

In our approach, a solution to RSSTD is represented by three arrays corresponding to the strips permutation π, the vector p storing the position for each strip, and the orientation vector o. Note that π and p are redundant, but the evaluation of the neighborhoods can be more efficiently implemented when both are available.

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

$$\mathcal{SH}(\sigma_1 \cdot \langle s_i \rangle \cdot \sigma_2 \cdot \langle s_j \rangle \cdot \sigma_3, i, j) = \sigma_1 \cdot \langle s_j \rangle \cdot \langle s_i \rangle \cdot \sigma_2 \cdot \sigma_3 \tag{9}$$

or

$$\mathcal{SH}(\sigma_1 \cdot \langle s_j \rangle \cdot \sigma_2 \cdot \langle s_i \rangle \cdot \sigma_3, i, j) = \sigma_1 \cdot \sigma_2 \cdot \langle s_i \rangle \cdot \langle s_j \rangle \cdot \sigma_3 \tag{10}$$

with $1 \leq i, j \leq n$. In this context σ_k denotes a possibly empty subsequence of strips. A second move, called *swapping* ($\mathcal{SW}$), is defined by swapping two arbitrary elements with each other. In a formal matter, this can be written as

$$\mathcal{SW}(\sigma_1 \cdot \langle s_i \rangle \cdot \sigma_2 \cdot \langle s_j \rangle \cdot \sigma_3, i, j) = \sigma_1 \cdot \langle s_j \rangle \cdot \sigma_2 \cdot \langle s_i \rangle \cdot \sigma_3 \tag{11}$$

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

$$\begin{split}\mathcal{BS}(\sigma_1 \cdot \langle s_i, .., s_{i+k} \rangle \cdot \sigma_2 \cdot \langle s_j, .., s_{j+k'} \rangle \cdot \sigma_3, i, j, k, k') = \\ \sigma_1 \cdot \langle s_j, .., s_{j+k'} \rangle \cdot \sigma_2 \cdot \langle s_i, .., s_{i+k} \rangle \cdot \sigma_3\end{split} \tag{12}$$

Table 1. Neighborhood structures defined for VND

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

with $1 \leq i < i + k < j < j + k' \leq n$. In addition to this four move types related to the assignment of strips to positions, two further moves for changing the orientation of a strip or a block of strips, called *rotating* ($\mathcal{R}$) and *block rotating* ($\mathcal{BR}$) respectively, are defined. Rotating simply rotates one strip by 180°, while block rotating executed on positions i to j first rotates all strips in this interval and in a second step swaps strips at positions i and j, $i+1$ and $j-1$, and so on. Using incremental evaluation schemes, each presented move can be evaluated in constant time.

In our VND, the five neighborhood structures induced by our moves are considered in the order shown in Table 1, thus, sorted by their sizes. As step function *best improvement* as well as *next improvement* have been implemented. For shaking in VNS, i random swap moves, with $1 \leq i \leq 4$, are performed. As initial solution a random solution is used.

6.3 User Interactions

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

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

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

A main advantage of integrating human power into the search procedure is in fact that with each additional lock of strips or forbidden neighborhood relation the solution space is pruned. For example, by fixing the relative order of two strips, the number of valid solutions in the search space is divided by n.

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

7 Experimental Results

In this section we present computational results comparing both introduced objective functions c_1 and c_2 and the different approaches. All experiments were performed on a Dual Core AMD Opteron 2214 with 4GB RAM. Both the HuGS framework and our VNS approach were implemented in Java. The Concorde TSP solver implemented by Applegate[1] was used and integrated into the Java evironment by using the *Java Native Interface*. The test instances were generated by virtually shredding paper documents, i.e. by either using scanned images or images extracted from PDF-files and cutting them into a defined number of equally sized strips. We remark that a real cutting and scanning process may loose some information or introduce errors, but neglect such effects in this work.

Quality of Solutions. As we want to find out which objective function introduced before is better suited for reconstructing strip shredded text documents, we define the *quality* of a solution as the number of correctly reconstructed subsequences of strips w.r.t. the original document. Note that the length of a correctly identified subsequence, i.e. the number of its strips, has no effect on our quality measure. This is motivated by the empiric observation that the text contained on reconstructed pages up to quality five usually can be read relatively easily. For any solutions with quality values larger than six it is typically very hard or almost impossible to the read the contained text. Further, this rating method enables us to compare results obtained for different strip widths and/or number of strips for one document.

Comparison of Results. For the results shown here we used six test instances that were shredded using different numbers of strip widths. While instances p1 to p5 consist of single text pages possessing different features (p1 and p3 are composed of continuous text only, instance p2 contains an image of a table, p4 offers a listing, and p5 shows a table with horizontal and vertical lines), instance p6 is the instance presented in [1] and consists of 10 pages with both printed and handwritten text. After virtually shredding the pages, a preprocessing step is performed on all instances, such that blank strips are eliminated.

Table 2 lists results obtained by applying the TSP solver on instances p1 to p6. We solved the instances using objective function c_1 as well as objective function c_2 and limited the CPU-time to 5 and 50 seconds, respectively. All values are average qualities over 30 runs. It can be observed that—especially for instances p2, p4 and p6—the qualities obtained by using function c_2 are remarkable better than those obtained by using c_1. Even for the short runs the standard deviations are very small and the improvement on the quality is not notable if the time limit is raised to 50 seconds. Log files show that in most cases the final solution was found after 0.5 seconds. In particular for the 10-page instance p6, the results are remarkably good. For 150 strips and cost function c_2 only 3 or 4 of the 10 pages were solved to quality 2; all others have quality 1. For 300 strips only 2 pages were always solved to quality 1 but for comparison with

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

Table 2. Average qualities of final solutions from the TSP solver comparing cost functions c_1 and c_2. Standard deviations are given in parentheses.

page		p1		p2		p3		p4		p5		p6	
time		5 s.	50 s.	5 s.	50 s.	5 s.	50 s.	5 s.	50 s.	5 s.	50 s.	5 s.	50 s.
30 strips	c_1	1.4 (0.5)	2.0 (0.0)	2.4 (1.4)	4.0 (0.0)	1.5 (0.5)	1.0 (0.0)	1.5 (0.5)	2.0 (0.0)	1.3 (0.5)	2.0 (0.0)	1.6 (0.5)	2.0 (0.0)
	c_2	1.4 (0.5)	1.0 (0.0)	1.5 (0.5)	2.0 (0.0)	1.6 (0.5)	2.0 (0.0)	1.7 (0.5)	2.0 (0.0)	1.5 (0.5)	1.0 (0.0)	1.6 (0.5)	2.0 (0.0)
50 strips	c_1	1.6 (0.5)	2.0 (0.0)	9.4 (0.7)	9.0 (0.0)	1.6 (0.5)	1.0 (0.0)	5.4 (0.5)	5.0 (0.0)	9.4 (0.5)	10.0 (0.0)	1.3 (0.5)	2.0 (0.0)
	c_2	1.4 (0.5)	2.0 (0.0)	4.1 (0.7)	5.0 (0.0)	1.5 (0.5)	2.0 (0.0)	1.4 (0.5)	1.0 (0.0)	1.4 (0.5)	2.0 (0.0)	1.5 (0.5)	2.0 (0.0)
100 strips	c_1	4.6 (0.5)	2.0 (0.0)	18.2 (0.8)	18.0 (0.0)	1.5 (0.5)	1.0 (0.0)	20.4 (0.5)	17.0 (0.0)	15.4 (0.5)	15.0 (0.0)	1.3 (0.4)	1.4 (0.5)
	c_2	1.5 (0.5)	2.0 (0.0)	11.8 (1.2)	13.0 (0.0)	1.4 (0.5)	1.0 (0.0)	3.8 (1.6)	5.0 (0.0)	5.5 (0.5)	6.0 (0.0)	1.4 (0.5)	2.0 (0.0)
150 strips	c_1	5.5 (0.6)	7.0 (0.0)	31.9 (0.7)	34.0 (0.0)	1.5 (0.5)	2.0 (0.0)	27.2 (1.0)	29.0 (0.9)	37.7 (0.5)	34.5 (0.5)	14.8 (0.8)	4.6 (0.5)
	c_2	1.5 (0.5)	2.0 (0.0)	26.5 (0.5)	25.0 (0.0)	1.5 (0.5)	1.0 (0.0)	16.7 (0.9)	16.0 (0.0)	9.4 (0.5)	6.0 (0.0)	4.5 (0.5)	5.0 (0.0)
300 strips	c_1	38.6 (0.7)	27.6 (0.5)	108.1 (0.8)	103.3 (1.1)	7.5 (0.5)	8.0 (0.0)	67.5 (0.6)	65.3 (0.9)	93.3 (1.1)	83.8 (0.7)	107.1 (1.6)	15.7 (1.0)
	c_2	1.6 (0.5)	2.0 (0.0)	78.3 (0.6)	73.0 (0.0)	1.5 (0.5)	1.0 (0.0)	41.5 (0.5)	43.0 (0.0)	27.4 (0.5)	27.0 (0.0)	14.3 (0.7)	14.0 (0.0)

Table 3. Average qualities of final solutions when applying VNS comparing cost functions c_1 and c_2. Standard deviations are given in parentheses.

page		p1		p2		p3		p4		p5		p6	
impr		next	best	next	best	next	best	next	best	next	best	next	best
30 strips	c_1	2.0 (0.0)	2.0 (0.0)	2.8 (1.3)	3.0 (1.4)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	c_2	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	1.0 (0.0)	1.0 (0.0)
50 strips	c_1	4.0 (0.0)	4.0 (0.0)	11.6 (1.4)	11.6 (1.6)	2.0 (0.0)	2.0 (0.0)	4.3 (1.2)	4.7 (1.7)	10.2 (0.4)	10.1 (0.5)	1.0 (0.0)	1.0 (0.0)
	c_2	2.0 (0.0)	2.0 (0.0)	4.7 (1.2)	5.8 (2.3)	2.0 (0.0)	2.0 (0.0)	3.2 (0.4)	3.3 (0.5)	2.1 (0.4)	2.2 (0.9)	1.0 (0.0)	1.0 (0.0)
100 strips	c_1	2.5 (1.5)	3.0 (2.1)	20.5 (2.2)	20.7 (2.2)	2.1 (0.7)	2.4 (1.2)	13.2 (3.3)	14.0 (2.8)	17.8 (2.8)	19.0 (3.0)	1.0 (0.0)	1.0 (0.2)
	c_2	2.0 (0.0)	2.0 (0.0)	14.8 (2.5)	15.5 (3.1)	2.0 (0.0)	2.0 (0.0)	7.1 (1.7)	6.6 (1.8)	6.2 (0.6)	6.5 (0.9)	1.0 (0.0)	1.0 (0.0)
150 strips	c_1	27.7 (6.7)	26.8 (8.4)	37.3 (2.0)	38.9 (2.4)	25.6 (7.6)	27.8 (9.6)	27.8 (2.2)	28.7 (3.1)	41.4 (7.3)	45.6 (7.4)	4.8 (1.5)	4.9 (1.4)
	c_2	19.5 (7.1)	22.4 (6.6)	26.0 (1.8)	27.2 (1.7)	16.8 (6.8)	16.7 (9.6)	18.7 (2.5)	18.7 (1.9)	19.6 (7.6)	23.8 (9.6)	5.6 (1.4)	4.4 (0.8)

the results presented in [1] we performed also tests with 340 strips on instance p6. This time 16 out 30 runs were solved to optimality for all other only one page was solved to quality 2 while all other were completely reconstructed. Especially when considering the time limit of 5 seconds, our methods clearly outperform those from Ukovich *et al.* [1].

Average results obtained when applying VNS without human interaction are presented in Table 3. For examining the neighborhoods we tested with both *next* as well as *best improvement* strategies, and no iteration or time limit was given. Again, the values presented are from 30 runs. We used the order of neighborhoods as presented in Section 6.2 but omitted the examination of the block swapping neighborhood $\mathcal{N}_5$ for instances with more than 100 strips as the size of this neighborhood is in $O(n^4)$. We can observe that the results obtained for objective c_2 are in general better than or equal to the results obtained for c_1, but no conclusions can be drawn which step function performs better for RSSTD. Based on the poorer performance of VNS on instances with more than 100 strips we conclude that neighborhood $\mathcal{N}_5$ substantially contributes to the success of VNS.

Finally we tested out semi-automatic system as it would be used in practice for reconstructing strip shredded text documents. With only few user interactions we were able to quickly restore all original documents by exploiting the benefits of the hybridization of machine and human power.

8 Conclusions

In this work, we presented a polynomial time transformation of the RSSTD to the symmetric TSP. We applied a chained Lin Kernighan heuristic as well as a newly introduced VNS for solving the RSSTD and showed that both methods are competitive with each other. In particular they clearly outperform the previous method from Ukovich *et al.*

Anyway, both approaches suffer from the necessarily imperfect objective function, which is only based on estimations of the likelihoods that strips shall be placed side-by-side under given orientations. Therefore, we embedded the algorithms in the HuGS-framework and gave the user the possibility to interact with the optimization in flexible ways. This turned out to work excellently. In this semi-automatic way, all test instances could be completely restored in very short time with only few user interactions. We consider the reconstruction of strip shredded text documents therefore as a superior example, where neither metaheuristics (and other other automated optimization techniques) nor human are able to produce satisfactory results, but a hybrid approach performs very well due to the combination of the different strengths.

References

1. 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. Proceedings of SPIE., International Society for Optical Engineering, vol. 6064, pp. 168–179 (2006)
2. Chung, M.G., Fleck, M., Forsyth, D.: Jigsaw puzzle solver using shape and color. In: Fourth International Conference on Signal Processing 1998, ICSP 1998, vol. 2, pp. 877–880 (1998)
3. Justino, E., Oliveira, L.S., Freitas, C.: Reconstructing shredded documents through feature matching. Forensic Science International 160(2–3), 140–147 (2006)

4. Schüller, P.: Reconstructing borders of manually torn paper scheets using integer linear programming. Master's thesis, Vienna Univ. of Technology, Austria (2008)
5. De Smet, P.: Reconstruction of ripped-up documents using fragment stack analysis procedures. Forensic science international 176(2), 124–136 (2008)
6. Skeoch, A.: An Investigation into Automated Shredded Document Reconstruction using Heuristic Search Algorithms. PhD thesis, University of Bath, UK (2006)
7. Ukovich, A., Ramponi, G., Doulaverakis, H., Kompatsiaris, Y., Strintzis, M.: Shredded document reconstruction using MPEG-7 standard descriptors. In: Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004, pp. 334–337 (2004)
8. Morandell, W.: Evaluation and reconstruction of strip-shredded text documents. Master's thesis, Vienna University of Technology, Austria (2008)
9. Fischetti, M., González, J.J.S., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research 45, 378–394 (1997)
10. Silberholz, J., Golden, B.: The generalized traveling salesman problem: A new genetic algorithm approach. In: Baker, E.K., et al. (eds.) Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies. Operations Research/Computer Science Interfaces, vol. 37, pp. 165–181. Springer, Heidelberg (2007)
11. Behzad, A., Modarres, M.: A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. In: Proceedings of the 15th International Conference of Systems Engineering, pp. 6–8 (2002)
12. Kumar, R., Haomin, L.: On asymmetric TSP: Transformation to symmetric TSP and performance bound (submitted, 1994); Journal of Operations Research
13. Balme, J.: Reconstruction of shredded documents in the absence of shape information. Working paper, Dept.of Computer Science, Yale University, USA (2007)
14. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding tours in the TSP. Technical Report Number 99885, Research Institute for Discrete Mathematics, Universität Bonn (1999)
15. Applegate, D., Cook, W., Rohe, A.: Chained lin-kernighan for large traveling salesman problems. INFORMS Journal on Computing 15(1), 82–92 (2003)
16. Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M., Schafer, G.T.: The HuGS platform: A toolkit for interactive optimization. In: Proc. Advanced Visual Interfaces, AVI, pp. 324–330. ACM Press, New York (2002)
17. Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided search: Survey and recent results. Technical Report TR2003-07, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA (2003); Submitted to Journal of Heuristics
18. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 145–184. Kluwer Academic Publishers, Dordrecht (2003)