Meta-Heuristics for Reconstructing Cross Cut Shredded Text Documents

Matthias Prandtstetter
prandtstetter@ads.tuwien.ac.at

Günther R. Raidl
raidl@ads.tuwien.ac.at

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/186-1, 1040 Vienna, Austria

ABSTRACT

In this work, we present two new approaches based on variable neighborhood search (VNS) and ant colony optimization (ACO) for the reconstruction of cross cut shredded text documents. For quickly obtaining initial solutions, we consider four different construction heuristics. While one of them is based on the well known algorithm of Prim, another one tries to match shreds according to the similarity of their borders. Two further construction heuristics rely on the fact that in most cases the left and right edges of paper documents are blank, i.e. no text is written on them. Randomized variants of these construction heuristics are applied within the ACO. Experimental tests reveal that regarding the solution quality the proposed ACO variants perform better than the VNS approaches in most cases, while the running times needed are shorter for VNS. The high potential of these approaches for reconstructing cross cut shredded text documents is underlined by the obtained results.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; G.1.6 [**Numerical Analysis**]: Optimization—*constrained optimization,integer programming*; G.1.10 [**Numerical Analysis**]: Applications; I.7.m [**Document and Text Processing**]: Miscellaneous

General Terms

Algorithms

Keywords

Document reconstruction, variable neighborhood search, ant colony optimization, integer linear programming

1. INTRODUCTION

Over the last years the interest in the reconstruction of shredded or manually ripped up paper documents has immensely grown. Due to the time and effort needed for doing this task manually, more and more systems evolved for enabling an at least semi-automatic reconstruction of destroyed documents. On the one hand this development is caused by the increasing interest of investigative bureaus for solving criminal cases and/or occurrences of industrial espionage. On the other hand there is also a great interest on this and related topics posed by archaeologists in reconstructing historical documents written on papyrus or parchment as well as putting together clay fragments containing ancient texts and paintings. Unfortunately, those systems already available lack in many different properties—mainly regarding output quality, number of shreds to be processed concurrently, as well as computation times. While Ukovich et al. [12] proposed a clustering approach which can be used as a preprocessing step for any semi-automatic system, De Smet [3] tried to exploit typical disposal behavior of humans after tearing paper. Prandtstetter and Raidl [7] presented an effective framework for restoring *strip shredded* documents which is based on variable neighborhood search and includes a user-interaction component. Many modern shredding devices, however, do not cut documents just into stripes but a much larger number of smaller pieces by applying *cross cutting*. Obviously, a reconstruction then becomes much more difficult. In this work, we concentrate on the case where a rectangular text document is cut into a regular grid of equally-sized smaller rectangles, and call this problem *reconstruction of cross cut shredded text documents* (RCCSTD).

In the next section, we give a formal model for this problem, which also shows how candidate solutions are represented in our approach. For quickly obtaining reasonable initial solutions, different construction heuristics are described in Sec. 4. Then, in Secs. 5 and 6 a general variable neighborhood search (VNS) metaheuristic that utilizes several different neighborhood structures within an embedded variable neighborhood descent local improvement procedure is described. As an alternative, Sec. 7 describes an ant colony optimization (ACO) approach that makes use of the same local improvement. The experimental results presented in Sec. 8 document that the ACO usually obtains better results than the VNS at the costs of longer running times.

2. FORMAL PROBLEM DEFINITION

We assume that a set $\mathcal{S} = \{1, \ldots, n\}$ of rectangular, geometrically identical shreds is given, which represent the output of a shredding device. Let shreds $1, \ldots, n-1$ be the shreds on which (parts of) a text is printed, while all

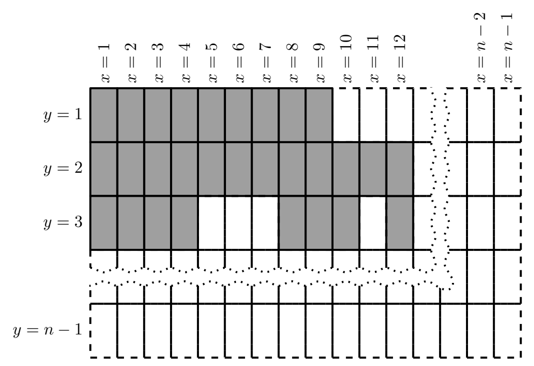

Figure 1: Sketch of a solution. Any of the $(n-1)\cdot(n-1)$ available positions may be occupied (dark shaded); all others are left free (light shaded).

blank shreds are replaced by the single special shred n for modeling reasons. For simplicity, we assume here that the orientation of all the shreds is known or identified during a preprocessing step based on pattern recognition techniques, see for example [1].

In addition, two error estimation functions (or cost functions) $\mathrm{c}(i,j) \geq 0$ and $\bar{\mathrm{c}}(i,j) \geq 0$ are given. They estimate the potential error introduced when placing shred j right next to shred i or by placing shred i on top of shred j, with $i,j \in \mathcal{S}$, respectively. This is achieved by counting for each pair of aligned pixels along the edges of shreds i and j the number of mismatches. A mismatch occurs, if the mean gray values of the corresponding pixels as well as the two pixels above and below differs more than a given threshold. A perfect match of shreds i and j, i.e. if these two shreds perfectly fit together, would then yield value 0, while larger values indicate the unlikeliness of this neighborhood relation. For a detailed definition and discussion of this error estimation function see [7].

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

$$\mathrm{s}(p) = \begin{cases} i & \text{if } \exists\, i \in \mathcal{S} \setminus \{n\} | \Pi(i) = p \\ n & \text{otherwise} \end{cases}, \quad \forall p \in {\mathbb{D}_0}^2, \qquad (1)$$

with $\mathbb{D}_0 = \{0,\ldots,n\}$. I.e. if a shred is placed at position p, it is returned by $\mathrm{s}(p)$; otherwise the position is assumed to be filled with the special empty shred n. Let us denote by $\mathrm{s_l}(p)$, $\mathrm{s_r}(p)$, $\mathrm{s_t}(p)$ and $\mathrm{s_b}(p)$, with $p = (x,y) \in \mathbb{D}^2$, shreds $\mathrm{s}((x-1,y))$, $\mathrm{s}((x+1,y))$, $\mathrm{s}((x,y-1))$ and $\mathrm{s}((x,y+1))$, respectively, such that the costs of a solution, i.e. the total potential error, can be defined as

$$c(\Pi) = \sum_{p \in \{1,\ldots,n\}^2} \mathrm{c}(\mathrm{s_l}(p), \mathrm{s}(p)) + \bar{\mathrm{c}}(\mathrm{s_t}(p), \mathrm{s}(p)). \qquad (2)$$

A sketch of a solution is shown in Fig. 1. Note that rows and columns of the solution may contain multiple entries of the virtual shred n, whereas all other shreds may not be contained more than once.

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

3. RELATED WORK

Many analogies can be drawn between RCCSTD and other problems found in literature like the reconstruction of *strip shredded* (text) documents [7, 10, 11], the restoration of *ripped up* and *manually destructed* paper work [3, 6, 9] and the solving of *jigsaw puzzles* [2].

Beside obvious relationships there are also essential differences. For example, methods for solving jigsaw puzzles mainly rely on the fact that the shapes of the pieces are unique. Even more, the smooth color transitions typically contained in depicted motives can be efficiently exploited [2], which is not true for text documents.

When reconstructing strip shredded text documents, different strategies can be used. Among others, Ukovich et al. [11] tried to find matching strips using MPEG-7 descriptors. In contrast, Skeoch [10] applied a genetic algorithm while Prandtstetter and Raidl [7] used a variable neighborhood search approach combined with user interactions. In [7] the authors showed that the reconstruction of strip shredded text documents is $\mathcal{NP}$-hard, which states a special case of RCCSTD.

For the reconstruction of manually torn paper feature matching methods [6] were applied. Further, heuristics based on the observation that ripped up documents are typically disposed as a stack of remnants are introduced in [3]. On the exact side, methods for reconstructing the border of torn documents based on integer linear programming techniques are presented in [9].

Finally, methods were proposed which can be applied to most of the above problems and which can be utilized during preprocessing steps for generating smaller instances that can then be handled more efficiently. Among these methods the approaches of Ukovich et al. [12] should be mentioned where clustering methods are used for identifying sets of shreds being part of the same original document page with the utmost probability.

4. CONSTRUCTION HEURISTICS

For quickly creating reasonable initial solutions used by the VNS as well as the ACO, we propose four different construction heuristics based on different ideas and observations. They mainly try to achieve good neighborhood relationships according to function $\mathrm{c}(i,j)$ only, with $i,j \in \mathcal{S}$, since function $\bar{\mathrm{c}}(i,j)$ is merely conditionally meaningful due to the observation that the width of a shred is in comparison to its height typically relatively small in practice, see also Fig. 1.

4.1 Greedy Matching Heuristic

In the *greedy matching heuristic* (GMH) a first intermediate solution is generated by grouping the shreds into pairs.

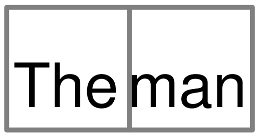

Figure 2: An example for a cutting such that a blank edge and a non-blank edge have to be matched in a perfect solution.

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

Instead of greedily identifying likely pairs of shreds, an initial solution could also be generated by iteratively computing (nearly) perfect matchings. This variant of GMH is called *perfect matching heuristic* (PMH).

4.2 Row Building Heuristic

The *row building heuristic* (RBH) is based on the observation that in a perfect solution (under the assumption that all shreds are available) each reconstructed row of shreds starts with a shred having a blank left edge and ends with a shred having a blank right edge. Therefore, RBH places a randomly chosen blank-left-edge snippet at the first position of the current row and continues by placing the best fitting shred with respect to $\mathrm{c}(i, j)$ next to it. This greedy best fit procedure is repeated until a snippet is reached with a blank right edge, which constitutes the end of the current row. Unfortunately, two special cases can occur: Firstly, it may happen that not all shreds are utilized when constructing a solution according to this procedure. In this case, the remaining shreds are purely randomly placed at the bottom of the constructed solution. Secondly, the number of shreds having a blank left edge needs not to be equal to the number of shreds having a blank right edge; for an example see Fig. 2. Additionally, more than one shred having a blank left edge might be used during the construction of the current row. If no more blank-left-edge shreds are available, the situation results in the first case. If no more blank-right-edge shreds are available, all other shreds have been used (including all left edge blank shreds). Therefore, no further actions have to be performed and the resulting solution is returned.

4.3 Multiple Paths Heuristic

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

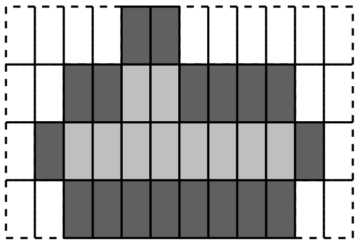

Figure 3: Prim iteration. Potential placements (dark shaded) of the next shred for expanding the current solution (light shaded).

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{c}(i, j) \cdot x_{ij} \tag{3}$$

$$\text{subject to} \sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i \in \mathcal{S} \setminus \{n\} \tag{4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \mathcal{S} \setminus \{n\} \tag{5}$$

$$x_{ii} = 0, \qquad \forall i \in \mathcal{S} \tag{6}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall i, j \in \mathcal{S} \setminus \{n\} \tag{7}$$

$$\sum_{j=1}^{n-1} x_{nj} \geq 1 \tag{8}$$

$$\sum_{i=1}^{n-1} x_{in} \geq 1 \tag{9}$$

$$\sum_{i \in S'} \sum_{j \in S'} x_{ij} \leq |S'| - 1, \qquad \forall S' \subseteq \mathcal{S} \setminus \{n\} \tag{10}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in \mathcal{S} \tag{11}$$

Within this model, the binary variable x_{ij}, with $i, j \in \mathcal{S}$, is set to one iff the right edge of shred i is matched with the left edge of shred j. While the objective (3) is to minimize the (potential) error introduced by these matchings, a solution is searched such that each shred except the special shred n has exactly one shred assigned to its left and exactly one to its right edge (Eqs. (4) and (5)). By constraints (6) and (7) it is assured that no loops and cycles of length two occur, respectively. Equations (8) and (9) ensure that at least one row is built. Finally, expression (10) avoids arbitrary length cycles not including the virtual shred n.

For obtaining solutions based on this ILP formulation, we apply the general purpose ILP solver CPLEX. Due to the fact that the number of constraints represented by Eq. (10) is not polynomially bounded, we add only violated constraints as lazy constraints during the *Branch&Bound* process. When decoding the obtained solution, the rows are randomly arranged since no information with respect to this order is given by the above presented model.

4.4 Prim Based Heuristic

In contrast to the so far presented construction heuristics the *Prim Based Heuristic* (PBH) follows the idea exploited by the algorithm of Prim [8] for finding minimum spanning trees. Analogously to this well known greedy algorithm, the solution is constructed by starting with an arbitrarily chosen shred that is placed at position $p = (1, 1)$. During the next steps, the intermediate solution is extended by adding

Algorithm 1: VariableNeighborhoodDescent(Π)

```
Input: initial solution Π
Data: neighborhood structures $\mathcal{N}_1, \ldots, \mathcal{N}_{l_{\max}}$
$l \leftarrow 1$;
while $l \neq l_{\max}$ do
    $\Pi' \leftarrow$ exploreNeighborhood $\mathcal{N}_l(\Pi)$;
    if $c(\Pi) > c(\Pi')$ then
        $\Pi \leftarrow \Pi'$;
        $l \leftarrow 1$;
    else
        $l \leftarrow l + 1$;
return Π;
```

Algorithm 2: GeneralVariableNeighborhoodSearch

```
Data: neighborhood structures $\mathrm{N}_1, \ldots, \mathrm{N}_{k_{\max}}$
$\Pi \leftarrow$ generate initial solution;
$k \leftarrow 1$;
while $k \neq k_{\max}$ do
    $\Pi' \leftarrow$ randomly choose one solution in $\mathrm{N}_k(\Pi)$;
    $\Pi' \leftarrow$ VariableNeighborhoodDescent($\Pi'$);
    if $c(\Pi) > c(\Pi')$ then
        $\Pi \leftarrow \Pi'$;
        $k \leftarrow 1$;
    else
        $k \leftarrow k + 1$;
```

one shred at a time which currently is the best matching one, i.e. which minimizes the additional error introduced by assigning it. Anyhow, possible positions for the next shred to be placed are just those positions having at least one of its four neighbors, i.e. the positions directly left, right, on top or at bottom, occupied, see also Fig. 3. In case that the best position for the next shred would be either $p = (0, y)$ or $p = (x, 0)$, with $1 \leq x, y \leq n - 1$, all shreds of the current intermediate solution are shifted one position to the right or to the bottom, respectively. Of course, the finally obtained solution can be of arbitrary shape, i.e. any placement of shreds can be obtained, as long as all shreds are connected to one component.

5. LOCAL IMPROVEMENT

Variable neighborhood descent (VND) is a deterministic local improvement (meta-)heuristic originally introduced by Hansen and Mladenović [5]. Based on the observation that a global optimum has always to be a local optimum with respect to defined neighborhoods, VND mainly relies on the proper definition of neighborhood structures to be systematically examined within local search iterations, see Alg. 1 for an outline in pseudocode. In each inner iteration the current neighborhood $\mathcal{N}_l(\Pi)$ is searched according to a next improvement strategy. For this purpose, we specify the following two basic move types, which will then be used to define neighborhood structures $\mathcal{N}_1$ to $\mathcal{N}_7$ utilized within our VND:

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

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

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

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

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

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

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

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

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

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

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

6. VARIABLE NEIGHBORHOOD SEARCH

Variable neighborhood search (VNS) [5] is a metaheuristic which combines a local search procedure with methods for escaping local optima. The local search part can be undertaken by a VND and perturbation is achieved by applying random moves within neighborhood structures of increasing variety. An outline of general VNS scheme is shown in Alg. 2.

Algorithm 3: AntColonyOptimization

```
Data: $m$ being the number of ants used
initialize pheromone matrix;
while termination condition not met do
    construct $m$ candidate solutions based on
      pheromone and heuristic information;
    apply local search; // optional
    update pheromone matrix;
```

For our purpose, we define the neighborhood structures N_i, with $1 \leq i \leq 5$ used within VNS as follows: in the i-th neighborhood structure i^2 randomly chosen shift moves of single shreds are performed. Computational results obtained by applying our VNS for reconstructing cross cut shredded text documents are presented in Sec. 8.

7. ANT COLONY OPTIMIZATION

Inspired by the behavior of natural ants, an *ant colony optimization* metaheuristic (ACO) [4] tries to guide multiple independent agents for constructing good solutions. This is mainly done by subsequently incorporating information based on solutions and their qualities achieved during elapsed iterations. In nature, this is achieved by so called pheromone trails, which are laid by ants when walking along paths between food locations and their home. Other ants follow these trails with a given probability in dependence on the amount of pheromone accumulated along these paths. In most computer systems based on this natural behavior, additional locally available knowledge is also incorporated into the solution construction process. An outline of the ant colony optimization principle is given in Alg. 3.

For our ACO, two pheromone matrices τ and $\overline{\tau}$ exist, whereas values τ_{ij} and $\overline{\tau}_{ij}$ correspond to the amount of pheromone laid for placing shred j right next to shred i and placing shred i on top of shred j, respectively. Both matrices are initialized within two steps, whereas during the first step five solutions $\Pi_1, \dots, \Pi_5$ are computed with the construction heuristics presented in Sec. 4, i.e. GMH, PMH, RBH, MPH and PBH. Based on the best obtained solution within this first step, an initial value τ^0 is computed by

$$\tau^0 = \frac{m}{\min_{i=1,\dots,5} c(\Pi_i)}, \tag{12}$$

whereas m denotes the number of ants being used within the ACO. Subsequently, all values τ_{ij} and $\overline{\tau}_{ij}$, with $i, j \in \mathcal{S}$, are set to τ^0. In the second step, a regular pheromone update (see Sect. 7.2) is performed using initial solutions Π_1 to Π_5.

7.1 Solution Construction

New candidate solutions are constructed within the ACO by one of the following alternative methods, which are based on the construction heuristics GMH, RBH, and PBH presented in Sec. 4. Each candidate solution created in such a way is then also locally improved by applying a restricted version of the above presented VND using only neighborhood structures $\mathcal{N}_1$ to $\mathcal{N}_3$ with a CPU-time limit of 500ms.

7.1.1 *Randomized Greedy Matching Heuristic*

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

$$\mathfrak{p}_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,j)}\right)^{\beta}}{\sum_{k \in \mathcal{S}'} \sum_{k' \in \mathcal{S}'} \tau_{kk'}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(k,k')}\right)^{\beta}}. \tag{13}$$

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

7.1.2 *Randomized Row Building Heuristic*

The randomized version of RBH—called *randomized row building heuristic* (RRBH)—tries to reconstruct a set of rows based on the following probability distribution:

$$\mathfrak{p}_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,j)}\right)^{\beta}}{\sum_{k \in \mathcal{S}'} \tau_{ik}^{\alpha} \cdot \left(\frac{1}{\mathrm{c}(i,k)}\right)^{\beta}} \tag{14}$$

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

7.1.3 *Randomized Prim Based Heuristic*

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

$$\mathfrak{p}_p^i = \frac{\delta(i,p)}{\sum_{p' \in \mathbb{D}_0{}^2} \sum_{k \in \mathcal{S}'} \delta(k,p')}, \qquad \begin{array}{l} \forall i \in \mathcal{S}' \\ \forall p \in \mathbb{D}_0{}^2 \end{array} \tag{15}$$

Function $\delta(i, p)$, with $i \in \mathcal{S}', p \in \mathbb{D}_0{}^2$ computes the additionally introduced error when placing shred i to position p. The value of $\delta(i, p)$ is equal to zero if p is either already used by another shred $k \in \mathcal{S} \setminus \mathcal{S}'$ or all neighbor positions of p are free, i.e. no shred $k \in \mathcal{S} \setminus \mathcal{S}'$ is positioned on them (see also Fig. 3). Analogously to PBH, all shreds are shifted one position to the right or to the bottom if the next shred should be assigned to any position outside of $\mathbb{D}^2$.

7.2 Pheromone Update

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

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta_{ij}^k + \Delta_{ij}^0, \quad \forall i, j \in \mathcal{S},\ i \neq j \tag{16}$$

$$\overline{\tau}_{ij} = (1-\rho) \cdot \overline{\tau}_{ij} + \sum_{k=1}^{m} \overline{\Delta}_{ij}^k + \overline{\Delta}_{ij}^0, \quad \forall i, j \in \mathcal{S},\ i \neq j \tag{17}$$

$$\Delta_{ij}^k = \begin{cases} \frac{1}{\mathrm{c}(\Pi_k)} & \text{if } j \text{ is placed right next to } i \text{ in the } k\text{-th solution} \\ 0 & \text{otherwise} \end{cases}, \qquad \begin{array}{l} \forall i, j \in \mathcal{S} \\ \forall k = 0, \dots, m \end{array} \tag{18}$$

$$\overline{\Delta}_{ij}^{k} = \begin{cases} \frac{1}{c(\Pi_k)} & \text{if } i \text{ is placed on top of } i \text{ in the } k\text{-th solution} \\ 0 & \text{otherwise} \end{cases}, \quad \begin{array}{l} \forall i,j \in \mathcal{S} \\ \forall k = 0,\ldots,m \end{array} \qquad (19)$$

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

8. EXPERIMENTAL RESULTS

Within this section a comparison of the proposed construction heuristics, different ACO settings and the presented VNS is done. For this purpose, we implemented all approaches in Java and performed several test runs on a single core of a Dual Opteron with 2.6GHz and 4GB of RAM.

We scanned five different pages of a text document containing a table of contents, a table with numbers and text as well as plain, formatted text. All document pages were transformed into gray scale images and then shredded into 9 instances with 9×9 to 15×15 snippets each, which results in a total of 45 different input instances to be reconstructed. For the definition of the used cost function (2) we refer to [7].

Table 1 shows the results obtained using RBH, PBH, GMH and MPH for all of these instances. Since preliminary tests revealed that PMH performs in most cases worse than GMH and in all cases worse than any other construction heuristic, no detailed results are presented for this method.

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

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

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

For testing our VNS approach we performed four independent tests, each one initializing VNS using another construction heuristic and consisting of 20 runs. Due to space limitations only the better results for the runs of VNS initialized with PBH and MPH (indicated by VNS-PBH and VNS-MPH) are shown in Tab. 2. In addition, the results obtained by applying our ACO approach to RCCSTD are presented, whereas the following settings for ACO were used: RPBH, RRBH and RGMH have been alternatively used with $m = 18$ ants. The results obtained for these settings are presented in the columns labeled ACO-RPBH, ACO-RRBH and ACO-RGMH, respectively. A fourth ACO setting was tested using again $m = 18$ ants but six of them applied RPBH, six RRBH and six RGMH. The value of parameter m was chosen based on preliminary tests, which also revealed that the fixing of α and β to 1 and 5 is promising for our ACO variants. The results of these test runs are listed in the columns labeled ACO. Again, the conclusions of selected Wilcoxon tests are presented in columns labeled with p, whereas VNS-PBH was compared to VNS-MPH and VNS-MPH, ACO, ACO-RPBH and ACO-RGMH were compared to ACO-RRBH. The corresponding p columns indicate again whether the first ($<$) or the second heuristic ($>$) yielded statistically better results on an error level of 5%. If none of these two cases occur, a $\approx$ sign is printed in the according field.

The values presented in Tab. 2 are again mean percentage gaps over 20 runs. When comparing VNS-PBH and VNS-MPH we cannot observe a general advantage for one of them. While VNS-PBH obtained for 25 instances the better mean gaps and VNS-MPH only for 19 instances, VNS-MPH was 10 times significantly better than VNS-PBH which was only 8 times significantly better. The best mean gaps of these two VNS variants are again printed bold. In addition the best mean values of the four ACO variants is emphasized.

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

Again, for each of these settings of VNS and ACO, mean values over 20 runs and standard deviations are listed. If ACO-RRBH was significantly better on an error level of 5% than another ACO setting the entry in the corresponding column labeled p is set to $>$; $<$ indicates that the corresponding ACO setting obtained significantly better results. If no conclusion could be drawn $\approx$ is given. The results comparing the two VNS settings with each other are presented in the column labeled p for VNS-PBH ($<$ indicates that VNS-PBH was statistically better on an error level of 5%). The results of the comparison of VNS-MPH and ACO-RRBH with each other, are presented in the p column of VNS-MPH.

Taking a closer look at the values in Tab. 2 it can be seen that for instance p001 with 9×9 shreds ACO-RRBH could always reconstruct the original document page. For some runs, the percentage gap is even negative, which can be easily explained by the fact that for any error estimation function it is not assured that the original document is evaluated best.

Regarding running times, we can summarize that the construction heuristics performed within hundreds of milliseconds. The VNS approaches needed between one and 100 seconds computation time until termination and the com-

Table 1: Average percentage gaps and corresponding standard deviations for the four construction heuristics are listed. Results of Wilcoxon rank sum tests for the hypothesis that MPH performs better than each of the other construction heuristics are given in columns p (using a 5% error level).

				RBH			PBH			GMH			MPH	
	x	y	orig	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev
instance p001	9	9	2977	147.7%	(27.9)	≈	172.2%	(9.0)	>	201.6%	(0.0)	>	**131.9**%	(31.6)
	9	12	4051	152.3%	(15.0)	≈	**142.3**%	(10.6)	≈	149.3%	(0.0)	≈	151.1%	(28.2)
	9	15	4215	169.0%	(29.0)	>	160.8%	(8.0)	>	194.9%	(0.0)	>	**147.2**%	(22.4)
	12	9	4125	115.6%	(18.1)	≈	140.2%	(9.8)	>	123.2%	(0.0)	>	**109.1**%	(25.1)
	12	12	4937	145.4%	(13.0)	>	**123.6**%	(8.1)	≈	155.4%	(0.0)	>	129.8%	(19.5)
	12	15	5147	172.8%	(17.4)	≈	130.7%	(8.0)	<	167.7%	(0.0)	<	**179.7**%	(20.5)
	15	9	4099	166.9%	(21.1)	>	161.1%	(8.7)	>	113.1%	(0.0)	>	**101.2**%	(24.2)
	15	12	4922	150.2%	(20.0)	>	142.6%	(8.6)	>	156.9%	(0.0)	>	**113.3**%	(15.2)
	15	15	5142	150.5%	(17.6)	>	135.7%	(6.6)	≈	158.9%	(0.0)	>	**138.0**%	(14.2)
instance p002	9	9	1786	229.2%	(15.5)	>	186.6%	(28.1)	>	159.8%	(0.0)	>	**141.7**%	(11.2)
	9	12	1538	335.2%	(22.3)	>	235.6%	(32.6)	>	191.4%	(0.0)	>	**190.0**%	(15.7)
	9	15	2462	249.4%	(10.6)	>	144.6%	(21.5)	≈	**132.0**%	(0.0)	<	145.6%	(12.3)
	12	9	1757	175.2%	(24.5)	>	228.3%	(38.5)	>	173.6%	(0.0)	>	**132.8**%	(14.8)
	12	12	1568	251.2%	(15.7)	>	273.5%	(28.6)	>	199.0%	(0.0)	>	**181.3**%	(7.4)
	12	15	2398	200.6%	(12.5)	>	168.3%	(18.8)	>	**123.4**%	(0.0)	<	134.2%	(11.9)
	15	9	2116	129.5%	(11.1)	>	243.7%	(22.6)	>	139.1%	(0.0)	>	**109.6**%	(24.8)
	15	12	2075	150.8%	(11.1)	>	255.5%	(18.6)	>	150.7%	(0.0)	>	**129.0**%	(12.4)
	15	15	2864	118.2%	(13.8)	>	183.4%	(13.1)	>	123.9%	(0.0)	>	**108.9**%	(9.5)
instance p003	9	9	3245	128.9%	(17.3)	≈	130.6%	(13.4)	≈	172.4%	(0.0)	>	**123.3**%	(25.4)
	9	12	3398	164.6%	(18.0)	≈	**153.0**%	(12.3)	<	175.3%	(0.0)	≈	169.2%	(34.3)
	9	15	3294	169.9%	(25.0)	≈	171.5%	(16.4)	≈	**153.9**%	(0.0)	≈	159.2%	(36.7)
	12	9	4049	**96.8**%	(15.8)	≈	137.3%	(9.6)	>	129.7%	(0.0)	>	105.7%	(18.7)
	12	12	4330	146.7%	(16.7)	≈	**129.8**%	(13.0)	<	135.1%	(0.0)	<	147.5%	(15.9)
	12	15	4264	143.3%	(16.4)	>	136.2%	(10.4)	≈	142.3%	(0.0)	>	**129.4**%	(20.1)
	15	9	4195	120.4%	(16.6)	>	140.0%	(7.9)	>	102.6%	(0.0)	>	**89.1**%	(24.1)
	15	12	4242	200.5%	(10.3)	>	149.5%	(11.6)	≈	**110.9**%	(0.0)	<	142.0%	(15.5)
	15	15	4270	182.5%	(11.3)	>	140.1%	(11.7)	≈	142.1%	(0.0)	≈	**138.6**%	(16.7)
instance p004	9	9	1411	184.4%	(39.5)	≈	207.6%	(21.0)	>	229.6%	(0.0)	>	**182.7**%	(24.3)
	9	12	1892	221.9%	(27.3)	>	**176.9**%	(19.9)	≈	220.7%	(0.0)	>	184.5%	(26.5)
	9	15	1979	157.8%	(19.4)	>	150.9%	(20.8)	≈	171.0%	(0.0)	>	**142.7**%	(18.2)
	12	9	2037	188.6%	(23.2)	>	194.0%	(16.2)	>	152.8%	(0.0)	≈	**149.4**%	(16.7)
	12	12	2689	139.1%	(14.5)	>	143.5%	(14.1)	>	159.1%	(0.0)	>	**121.8**%	(17.1)
	12	15	2734	119.2%	(9.8)	>	122.9%	(10.0)	>	122.9%	(0.0)	>	**107.2**%	(3.4)
	15	9	2193	164.3%	(12.5)	>	206.5%	(19.1)	>	140.4%	(0.0)	>	**113.7**%	(23.9)
	15	12	2481	**135.3**%	(13.9)	≈	184.9%	(13.8)	>	145.7%	(0.0)	≈	144.3%	(24.4)
	15	15	2719	**106.2**%	(18.6)	≈	138.6%	(14.3)	>	122.8%	(0.0)	>	107.6%	(17.3)
instance p005	9	9	923	211.0%	(36.3)	≈	343.0%	(50.7)	>	245.3%	(0.0)	>	**197.3**%	(38.0)
	9	12	1293	325.6%	(40.5)	≈	**302.6**%	(34.6)	≈	333.9%	(0.0)	>	316.8%	(33.5)
	9	15	2123	264.7%	(19.8)	>	245.8%	(16.9)	≈	265.7%	(0.0)	>	**243.4**%	(27.6)
	12	9	1365	177.8%	(15.8)	>	251.8%	(28.4)	>	171.5%	(0.0)	>	**154.4**%	(17.6)
	12	12	1841	236.7%	(30.0)	>	258.7%	(21.1)	>	222.9%	(0.0)	>	**204.8**%	(26.7)
	12	15	2588	219.8%	(15.0)	≈	**204.1**%	(15.0)	<	214.5%	(0.0)	<	216.6%	(17.1)
	15	9	1317	175.6%	(21.1)	>	304.5%	(41.1)	>	193.5%	(0.0)	>	**117.0**%	(27.8)
	15	12	1634	304.6%	(21.6)	>	299.0%	(24.6)	>	260.4%	(0.0)	>	**233.5**%	(34.9)
	15	15	2460	253.3%	(14.1)	>	225.7%	(17.8)	>	**197.1**%	(0.0)	<	205.1%	(13.0)

putation times for ACO lie between approximately 100 seconds and 800 seconds. It can be concluded that although the results obtained by ACO are better in most cases, the computation times needed are significantly higher.

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

9. CONCLUSIONS

In this work, we presented several approaches based on ACO as well as VNS for reconstructing cross cut shredded text documents. All of them could significantly improve initial solutions obtained by construction heuristics. While the ACO variants needed more computation time than the VNS, the former also obtained better results on 34 of 45 instances.

Although the optimal "correct" solutions was only reached once by the proposed approaches, the results obtained are promising. For document pages with much written text on them, we were able to reconstruct large parts of the original document. A further investigation of either additional neighborhood structures to be used within VND or additional or more problem dependent construction heuristics incorporated into ACO seems to be promising.

10. REFERENCES

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

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

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

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

Table 2: Results obtained by VNS and ACO. The mean percentage gaps over 20 runs and standard deviations are presented for two independent test sets of VNS initialized using PBH and MPH as well as the mean gaps (over 20 runs) and standard deviations of 4 different ACO variants incorporating RPBH, RGMH, RRBH and all three of them, respectively. Values in columns p correspond to the results of Wilcoxon rank sum tests using a 5% error level.

				VNS-PBH			VNS-MPH			ACO			ACO-RPBH			ACO-RGMH			ACO-RRBH	
	x	y	orig.	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev	p	mean	dev
instance p001	9	9	2977	40.8%	(12.5)	>	**10.0**%	(12.7)	>	4.0%	(5.9)	>	23.8%	(8.1)	>	29.1%	(9.5)	>	**0.0**%	(0.0)
	9	12	4051	28.7%	(11.6)	>	**20.8**%	(8.8)	≈	**14.6**%	(5.1)	>	27.0%	(4.9)	>	27.8%	(3.7)	>	21.0%	(4.8)
	9	15	4215	34.7%	(8.0)	>	**28.6**%	(7.7)	≈	31.7%	(3.6)	>	34.8%	(2.8)	>	35.6%	(2.7)	>	**30.7**%	(2.1)
	12	9	4125	**26.4**%	(5.9)	≈	28.4%	(9.3)	>	27.6%	(3.3)	>	27.2%	(4.0)	>	27.0%	(3.0)	>	**25.2**%	(3.2)
	12	12	4937	26.5%	(6.1)	≈	**24.8**%	(5.9)	≈	29.6%	(2.2)	>	34.5%	(4.0)	>	30.9%	(3.0)	>	**27.5**%	(2.9)
	12	15	5147	**30.4**%	(6.6)	≈	31.0%	(10.6)	≈	34.0%	(2.7)	≈	33.1%	(2.9)	≈	34.7%	(2.8)	>	**32.7**%	(3.2)
	15	9	4099	37.5%	(11.2)	>	**30.1**%	(7.8)	≈	32.8%	(3.7)	>	32.8%	(4.7)	>	33.7%	(3.6)	>	**29.1**%	(4.1)
	15	12	4922	32.0%	(5.8)	>	**28.2**%	(7.6)	<	31.9%	(3.6)	≈	34.9%	(2.7)	>	**31.4**%	(3.1)	≈	32.5%	(2.6)
	15	15	5142	**32.3**%	(5.2)	≈	**32.3**%	(5.8)	≈	35.3%	(3.1)	>	36.5%	(2.3)	>	36.6%	(3.9)	>	**33.2**%	(3.2)
instance p002	9	9	1786	**2.1**%	(9.0)	≈	3.2%	(12.5)	≈	0.2%	(3.7)	<	**-4.1**%	(5.7)	<	-2.5%	(5.1)	<	4.8%	(4.6)
	9	12	1538	**23.0**%	(9.6)	<	34.6%	(14.1)	≈	17.0%	(3.7)	<	**16.9**%	(4.5)	<	17.6%	(3.9)	<	35.3%	(5.1)
	9	15	2462	**8.4**%	(4.4)	<	15.6%	(6.2)	<	8.0%	(2.5)	<	**6.9**%	(2.6)	<	14.0%	(2.5)	<	21.7%	(3.6)
	12	9	1757	**6.0**%	(9.7)	≈	9.6%	(11.1)	≈	9.0%	(6.3)	<	**3.3**%	(6.3)	<	11.3%	(7.1)	≈	13.4%	(4.2)
	12	12	1568	**22.6**%	(9.1)	<	29.4%	(9.8)	≈	24.8%	(4.7)	<	**21.9**%	(4.1)	<	25.7%	(4.1)	≈	27.7%	(5.0)
	12	15	2398	**9.4**%	(6.0)	<	19.0%	(10.9)	≈	14.4%	(3.1)	<	**12.4**%	(2.4)	<	20.4%	(3.1)	≈	19.7%	(3.5)
	15	9	2116	19.8%	(10.1)	>	**14.0**%	(10.7)	≈	16.3%	(3.5)	>	25.6%	(3.4)	>	16.8%	(3.6)	≈	**13.8**%	(4.0)
	15	12	2075	27.3%	(11.4)	≈	**24.6**%	(9.0)	>	17.2%	(3.4)	<	34.3%	(5.4)	>	**15.9**%	(3.8)	<	19.8%	(3.3)
	15	15	2864	16.2%	(5.0)	≈	**17.5**%	(8.1)	>	13.8%	(1.9)	>	22.2%	(2.8)	>	17.1%	(2.9)	>	**12.6**%	(1.7)
instance p003	9	9	3245	25.2%	(8.2)	>	**21.3**%	(8.0)	>	11.5%	(4.3)	>	18.9%	(3.4)	>	18.5%	(4.0)	>	**6.7**%	(4.8)
	9	12	3398	**34.1**%	(4.1)	≈	34.4%	(11.0)	>	**29.9**%	(3.6)	≈	37.4%	(3.5)	>	32.0%	(4.5)	≈	30.0%	(3.6)
	9	15	3294	25.4%	(14.3)	>	**11.6**%	(6.7)	<	21.7%	(5.3)	>	31.1%	(6.4)	>	24.3%	(2.2)	>	**18.0**%	(2.6)
	12	9	4049	23.4%	(7.3)	>	**16.4**%	(6.1)	>	14.7%	(2.0)	>	23.6%	(3.1)	>	17.4%	(2.6)	>	**11.6**%	(2.9)
	12	12	4330	25.1%	(4.6)	>	**19.9**%	(6.1)	<	24.5%	(3.3)	>	39.1%	(3.4)	>	25.7%	(3.3)	>	**23.3**%	(2.5)
	12	15	4264	**20.9**%	(6.9)	≈	23.2%	(6.2)	>	17.4%	(2.4)	>	33.0%	(4.9)	>	18.5%	(2.5)	>	**16.3**%	(2.1)
	15	9	4195	**20.6**%	(6.2)	≈	21.8%	(10.0)	>	16.1%	(3.1)	>	31.1%	(5.0)	>	24.2%	(2.2)	>	**13.6**%	(2.3)
	15	12	4242	**33.1**%	(6.4)	≈	35.7%	(8.1)	≈	37.5%	(3.2)	≈	55.7%	(5.2)	>	38.9%	(3.5)	>	**36.3**%	(2.1)
	15	15	4270	**27.3**%	(4.7)	≈	30.0%	(5.1)	>	26.3%	(2.2)	≈	35.8%	(4.0)	>	**25.7**%	(2.4)	<	27.0%	(1.4)
instance p004	9	9	1411	**28.7**%	(16.5)	≈	35.8%	(11.5)	>	21.5%	(6.4)	≈	**18.3**%	(7.8)	≈	20.7%	(9.4)	≈	18.4%	(7.2)
	9	12	1892	**21.7**%	(9.9)	<	28.0%	(10.1)	>	15.0%	(4.8)	≈	14.5%	(5.0)	≈	**14.2**%	(3.4)	≈	15.6%	(4.5)
	9	15	1979	**9.4**%	(8.5)	<	16.0%	(6.3)	>	3.9%	(3.8)	≈	**2.4**%	(3.3)	≈	5.8%	(2.8)	>	3.8%	(3.3)
	12	9	2037	**36.6**%	(10.4)	≈	38.3%	(11.0)	>	29.1%	(4.5)	>	35.7%	(6.8)	<	38.4%	(5.8)	>	**26.3**%	(5.3)
	12	12	2689	**20.8**%	(5.1)	<	28.5%	(5.0)	>	19.2%	(2.4)	<	**15.9**%	(2.6)	<	21.9%	(2.4)	≈	20.6%	(1.9)
	12	15	2734	**6.1**%	(3.6)	<	12.2%	(5.7)	>	**6.8**%	(2.8)	≈	7.1%	(1.6)	≈	10.4%	(1.7)	>	7.6%	(1.8)
	15	9	2193	41.4%	(13.4)	>	**26.6**%	(12.8)	>	18.2%	(3.5)	>	43.5%	(5.2)	>	26.0%	(3.5)	>	**12.5**%	(4.1)
	15	12	2481	**36.9**%	(10.5)	≈	38.0%	(9.7)	>	29.4%	(2.7)	>	41.7%	(5.0)	>	30.7%	(3.4)	>	**27.0**%	(2.6)
	15	15	2719	15.7%	(6.6)	≈	**14.4**%	(5.4)	>	5.1%	(2.6)	>	20.8%	(3.7)	>	6.3%	(1.8)	>	**3.9**%	(1.6)
instance p005	9	9	923	57.1%	(20.6)	≈	**54.7**%	(23.6)	>	21.1%	(8.9)	>	66.7%	(11.2)	>	36.5%	(12.0)	>	**13.6**%	(8.2)
	9	12	1293	**82.0**%	(20.3)	≈	82.1%	(21.9)	>	70.4%	(9.5)	≈	72.5%	(6.9)	≈	78.9%	(8.7)	>	**69.0**%	(7.4)
	9	15	2123	**48.4**%	(9.1)	≈	51.8%	(8.7)	>	39.3%	(6.6)	≈	45.8%	(4.6)	>	44.2%	(5.1)	>	**36.2**%	(4.5)
	12	9	1365	43.5%	(8.7)	≈	**42.4**%	(16.3)	>	27.5%	(4.2)	>	75.4%	(8.1)	>	29.7%	(3.7)	>	**23.4**%	(4.5)
	12	12	1841	**47.1**%	(14.4)	≈	51.4%	(13.6)	>	40.0%	(5.1)	>	54.3%	(7.6)	>	44.5%	(5.7)	>	**36.6**%	(3.1)
	12	15	2588	45.9%	(6.0)	≈	**44.1**%	(7.9)	>	35.9%	(3.1)	>	45.4%	(2.9)	>	37.6%	(3.6)	>	**34.5**%	(2.6)
	15	9	1317	35.5%	(29.9)	>	**12.8**%	(15.0)	>	-4.3%	(4.2)	>	43.7%	(13.2)	>	-1.2%	(3.2)	>	**-9.4**%	(1.4)
	15	12	1634	**53.9**%	(11.0)	≈	58.7%	(15.9)	>	55.4%	(4.7)	>	67.6%	(7.1)	>	59.0%	(5.2)	>	**51.5**%	(3.9)
	15	15	2460	**42.2**%	(6.4)	≈	42.9%	(8.3)	>	39.5%	(3.4)	≈	44.9%	(4.7)	>	41.0%	(3.5)	>	**38.5**%	(3.7)

[5] P. Hansen and N. Mladenović. Variable neighborhood search. In Glover and Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publisher, New York, 2003.

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

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

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

[9] P. Schüller. Reconstructing borders of manually torn paper sheets using integer linear programming. Master's thesis, Vienna University of Technology, Austria, 2008.

[10] A. Skeoch. *An Investigation into Automated Shredded Document Reconstruction using Heuristic Search Algorithms*. PhD thesis, University of Bath, UK, 2006.

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

[12] A. Ukovich, A. Zacchigna, G. Ramponi, and G. Schoier. Using clustering for document reconstruction. In E. R. Dougherty et al., editors, *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064 of *Proceedings of SPIE*, pages 168–179. International Society for Optical Engineering, 2006.