Complexity Results on a Paint Shop Problem

Th. Epping [a] W. Hochstättler [a]

[a] *Department of Mathematics, BTU Cottbus*

P. Oertel [b,1]

[b] *Ford Werke AG, Cologne*

Abstract

Motivated by an application in the automobile industry, we present results and conjectures on a new combinatorial problem: Given a word w and restricted reservoirs of colored letters, synthesize w with a minimal number of color changes.

We present a dynamic program that solves this problem and runs in polynomial time if we bound both, the number of different letters and colors. Otherwise, the problem is shown to be NP-complete. Additionally, we focus on upper bounds on the minimal number of color changes, simultaneously giving results for special instances, and posing open questions.

Key words: Dynamic Programming, NP-completeness, Paint Shop, Sequencing
1991 MSC: 90B30, 90B35, 90C39

1 Motivation

European car manufacturers are faced with a continously increasing demand for individually furnished cars. To avoid the need for large depots, cars are therefore generally built to order. This leads to a daily changing variety of car body types that have to be produced. The final stage of production planning

Email addresses: epping@math.tu-cottbus.de (Th. Epping), hochstaettler@math.tu-cottbus.de (W. Hochstättler), poertel@ford.com (P. Oertel).

[1] Supported by Alfried Krupp von Bohlen and Halbach Stiftung

typically is the sequencing stage, where a build sequence is determined for the orders to be manufactured on a given day.

Sequencing has great impact both on production quality and costs, since each order has to pass through various production phases, including a press and a body shop, a paint shop, and assembly lines. We focus on the paint shop, which is divided into several booths in which a car body is cleaned and gets its cathodic immersion painting, its prime color, and its enamel color (see [1]). Within the enamel booth, a color change occurs whenever two consecutive car bodies have to be colored in different colors. For each color change the color jets of the spray robots have to be cleaned, giving rise to non-negligible costs and water pollution. Therefore, the automotive industry has been constantly striving to reduce the number of color changes within the enamel booth.

Present technology is to apply sorting heuristics which use temporary storage systems placed in front of the paint shop to group car bodies that can be colored alike. A popular approach is the installation of a line storage system (see [2]). These systems allow sorting by spreading an input sequence on a certain number of sorting belts and merging the contents of the sorting belts to a new output sequence. However, it it reasonable to conform a car body sequence not only to the minimization of color changes within the paint shop, as preceeding and succeeding production phases may require different sequences to optimize their specific objective function. We therefore drop the usage of color storage systems and consider the car body sequence to be an external parameter. Also, there is a trend that heads for the detachment of car bodies and their features and allows more production flexibility. This concept can be realized by the use of on-line programmable micro chips attached to the car bodies and allows us to uncouple enamel colors and car bodies.

These realities yield a new type of combinatorial problem that we study from an algorithmical and a mathematical point of view.

The paper is organized as follows. In the next section, we introduce the basic mathematical model of the problem. In Section 3 we present a dynamic programming algorithm and analyze its complexity. Then we show that the problem is NP-complete if either the number of colors or the number of car body types is unbounded. Finally, we give results and conjectures on the minimal number of color changes for special instances and end with a collection of open questions.

Our notation is fairly standard as in [3].

2 Problem formulation

We will now give a formal definition of the sequencing problem as it may be encountered in production sequencing. Given a set of orders we must determine a sequence that minimizes the number of color changes during production. To simplify the definition, we assume that an arbitrary order sequence has been fixed beforehand. We associate with each car body type a letter of an alphabet Σ, and represent a sequence of car body types by a word w. A vector f of the same length as w represents a sequence of colors, with f_i denoting the color of w_i for all i. We say that we have a *color change* in f whenever $f_i \neq f_{i+1}$.

Our problem then consists in finding a permutation that minimizes the number of color changes in f while leaving the sequence of letters in w unchanged.

Problem 1 *Paint Shop Problem for Words (PPW)*

Given a finite alphabet Σ, a word $w = (w_1, \ldots, w_n) \in \Sigma^*$, a finite color set F, and a coloring $f = (f_1, \ldots, f_n)$ of w with $f_i \in F$ for $i = 1, \ldots, n$, find a permutation $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $w_{\sigma(i)} = w_i$ for $i = 1, \ldots, n$, and the number of color changes within $\sigma(f) = (f_{\sigma(1)}, \ldots, f_{\sigma(n)})$ is minimized.

We denote the *minimal number of color changes* for an instance of PPW by $\gamma(w; f)$. Note that the initial coloring of w serves only to determine the reservoirs of colored letters. We therefore can deal with these reservoirs instead of an explicit color vector (see Figure 1). We denote the *reservoir* of letter i in color j by $R(i, j)$, i. e. , $R(i, j)$ denotes the number of copies of letter i in color j.

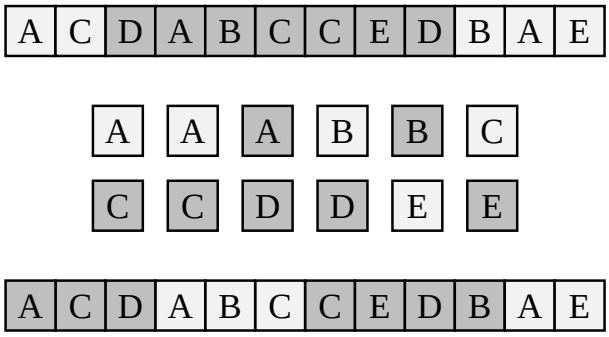

Fig. 1. An optimal resp. suboptimal colored instance with $|\Sigma| = 5$ and $|F| = 2$.

3 Solution by dynamic programming

Any instance of PPW can be solved by dynamic programming as follows. We pass through the given word w letter by letter from the left to the right. We

record each optimal coloring up to the current position, that uses a subset of the letter reservoir and ends with a specific color, in a different state.

Each state is given as

$$s_p = s_p(l_1^1, \ldots, l_{|F|}^1, l_1^2, \ldots, l_{|F|}^2, \ldots, l_1^{|\Sigma|}, \ldots, l_{|F|}^{|\Sigma|}, f) \text{ with } p = \sum l_j^i,$$

where l_j^i counts the occurences of letter i in color j, and f denotes the color that has been used to color the letter w_p. We denote by $\gamma(s_p)$ the minimal number of color changes of the partial coloring of w up to position p with the recorded colors. Note that given a state s_p the letter i that has to be colored next is determined by $i = w_{p+1}$. We then apply the dynamic program depicted in Figure 2.

Informally speaking, the pass through w yields a directed graph of feasible colorings. The nodes of the graph correspond to specific states. Edges occur between states s_{p-1} and s_p if these states have matching values of l_j^i except for one l_{j0}^{i0}, and are weighted with 0 or 1 depending on the occurence of a color change when progressing from s_{p-1} to s_p. The dynamic program searches for a shortest path within this graph.

(1) `For all` $f \in F$ `do:`
`Create` $s_0(0, \ldots, 0, f)$ `and set` $\gamma(s_0(0, \ldots, 0, f)) = 0$.
(2) `For` $p = 1, \ldots, n$ `do:`
`For all existing states` $s_{p-1}(\ldots, l_f^{w_p}, \ldots, g)$ `do:`
`For all` $f \in F$ `do:`
`If` $(R(w_p, f) - l_f^{w_p}) > 0$`, create` $s_p(\ldots, l_f^{w_p} + 1, \ldots, f)$ `and set`

$$\gamma(s_p(\ldots, l_f^{w_p} + 1, \ldots, f)) = \min_{k \neq f}\{\gamma(s_{p-1}(\ldots, l_f^{w_p}, \ldots, f)), \gamma(s_{p-1}(\ldots, l_f^{w_p}, \ldots, k)) + 1\}.$$

Fig. 2. The dynamic program for PPW.

Theorem 2 *The dynamic program depicted in Figure 2 solves an instance of PPW with a space and time complexity of* $\mathcal{O}(|F| n^{|F||\Sigma|})$.

PROOF. We proceed by induction on p and show that $\gamma(s_p)$ is the minimal number of color changes that appear if we use the color reservoir recorded in s_p for a coloring of the first p letters of w. Clearly, the initialization $\gamma(s_0) = 0$ for $p = 0$ is correct. If $p > 0$, the letter that has to be colored next for creating a state s_p is w_p. We can color w_p in color f only if there is at least one letter w_p in color f left. Thus $R(w_p, f) - l_f^{w_p}$ has to be positive. Then $\gamma(s_p)$ is the minimum of $\gamma(s_{p-1})$, where in state s_{p-1} the letter w_{p-1} has been colored with the same color as w_p, and $\gamma(s_{p-1}) + 1$, where in state s_{p-1} the letter w_{p-1}

has been colored with a different color as w_p, increased by one due to the additional color change.

As every l_j^i is bounded from above by $l_j^i \leq n$, we directly get $n^{(|F||\Sigma|)}$ as an upper bound for the number of states that can be created. The computation of $\gamma(s_p)$ requires $\mathcal{O}(|F|)$ steps. $\square$

Note that the dynamic program can be easily extended to find an optimal coloring of w. Note further that it is sufficient to record l_j^i only for $|F|-1$ colors. We can therefore improve on the complexity of the dynamic program.

Corollary 3 *The dynamic program can be implemented to run with a space and time complexity of* $\mathcal{O}(|F|n^{(|F|-1)|\Sigma|})$.

Typical values in practice are $6 \leq |\Sigma| \leq 8$ and $3 \leq |F| \leq 15$, where most colors appear with low, and only few colors appear with high frequency. The dynamic program is thus unlikely to be applicable in practice. A comparable result holds for a dynamic program that assumes the use of temporary storage systems in front of the paint shop (see [4]).

4 Complexity results

The dynamic program presented in the last section shows that Problem 1 is polynomial if we restrict the size of both, the number of colors and the size of the underlying alphabet. In this section, we will argue that this result is best possible from a complexity point of view. More precisely, we will show by reduction from 3SAT resp. pseudo-polynomial reduction from 3-PARTITION that Problem 1 is NP-complete if we bound one parameter, i. e. the size of the color set F or the size of the alphabet Σ, even if we bound it to $|F| = 2$ or $|\Sigma| = 2$.

Theorem 4 *PPW is NP-complete for* $|F| = 2$.

PROOF. We give a reduction from 3SAT (see [3]). Let $C = \{c_1, \ldots, c_m\}$ denote the set of clauses and $V = \{v_1, \ldots, v_n\}$ denote the set of variables of an instance of 3SAT. We construct an instance of PPW as follows. The word w is a sequence of n *variable blocks*. We explain the construction of a single variable block for a fixed variable v_i in detail, assuming that F contains the colors red and blue. In this proof, we use the terms *characters* and *letters* to distinguish between colored elements (the characters) and uncolored symbols of the alphabet (the letters).

We assume that v_i appears in clauses $c_{i_1}, \ldots, c_{i_k}$ as literals $l_{i_1}, \ldots, l_{i_k}$ and wlog. literals are sorted, such that $l_{i_j} = v_i$ for $j = 1, \ldots, r$ and $l_{i_j} = \bar{v}_i$ for $j = r+1, \ldots, k$. Note that we can always assume $r \geq 1$ and sort the literals independently for each variable block. The variable block of v_i then is a sequence of k *literal blocks* $b_{i_1}, \ldots, b_{i_k}$. The characters we use for building the variable block are given as follows.

- First, we provide $4k$ *variable characters* using $2k$ *variable letters* L_j^i $(j = 1, \ldots, 2k)$, each once in red and once in blue. Variable letters differ for each variable block. In the following, we drop their superscript whenever it is clear which variable we refer to.
- Second, we introduce $3m$ *satisfaction testers* using m letters T_s $(s = 1, \ldots, m)$, each once in red and twice in blue. Each letter T_s is associated with a clause c_s. We use the satisfaction testers T_{i_j} $(j = 1, \ldots, k)$, one of each, within the variable block.
- Finally, we introduce a *separator letter* Z, available only in blue. We use $k+1$ *separator characters* within the variable block considered.

We arrange the letters of the variable block as follows. For each literal block we provide four variable letters, as indicated in Figure 3. We precede the first variable letter of a literal block b_{i_j} with ZT_{i_j} for $j = 1, \ldots, r$ and precede the third variable letter of a literal block b_{i_j} with ZT_{i_j} for $j = r+1, \ldots, k$. Finally, we add a separator Z behind the last variable letter of a variable block.

$$\underbrace{ZT_{i_1}\overbrace{L_1}^{v_1}L_2\overbrace{L_2}^{\bar{v}_1}L_3}_{\text{Literal block } b_{i_1}} \ldots \underbrace{ZT_{i_r}L_{2r-1}L_{2r}L_{2r}L_{2r+1}}_{b_{i_r}}$$
$$\underbrace{\overbrace{L_{2r+1}}^{v_1}L_{2r+2}\,ZT_{i_{r+1}}\overbrace{L_{2r+2}}^{\bar{v}_1}L_{2r+3}}_{b_{i_{r+1}}} \ldots \underbrace{L_{2k-1}L_{2k}ZT_{i_k}L_{2k}L_1}_{b_{i_k}}Z$$

Fig. 3. The structure of a variable block.

As each variable letter L_j $(j = 1, \ldots, 2k)$ is available only once in each color, it is guaranteed that the two variable characters L_{2j} $(j = 1, \ldots, k)$ in each literal block b_{i_j} will have different colors. We claim that we can get by with two color changes for every literal block and prove the following:

There exists a satisfying truth assignment for $v_1, \ldots, v_n$ if and only if there exists a coloring of w with exactly $6m$ color changes.

Given a satisfying truth assignment, let $c_s = \{l_{s_1}, l_{s_2}, l_{s_3}\}$ be a clause and assume that l_{s_1} satisfies the clause. We then color the associated satisfaction tester T_s in the literal block of l_{s_1} red (and have to color the satisfaction testers in the literal blocks of l_{s_2} and l_{s_3} blue). Additionally, we color the variable letters of the variable block alternatingly (red, red, blue, blue, ...), if the

variable related to the variable block is set to TRUE in the truth assignment; if the variable is set to FALSE, we use the alternating color scheme (blue, blue, red, red, ...).

If our coloring starts with two blue variable characters and we come across a color change between two literal blocks, we always account it to the number of color changes within the preceeding one (including the color change between the last variable character and the final separator). Otherwise, if our coloring starts with two red variable characters and we come across a color change between two literal blocks, we always account it to the number of color changes within the succeeding one. In both cases, this approach results in exactly two color changes within any literal block. Thus, we end up with $6m$ color changes altogether for a coloring of w.

Now, suppose that we are given a coloring with exactly $6m$ color changes. First note that counting in a similar way as above we have at least two color changes per literal block. As we know that we have a total of $3m$ literals, we must have exactly two color changes per literal block. This is only possible if the variable letters within each variable block are colored in connected blocks of size two, resulting in a coloring sequence of either (red, red, blue, blue, ...) or (blue, blue, red, red, ...).

We then assign the value TRUE to variables whose variable block starts with two red variable characters, and FALSE otherwise and show that we get a satisfying truth assignment. Let $c_s = \{l_{s_1}, l_{s_2}, l_{s_3}\}$ be a clause and assume that the associated satisfaction tester T_s in the literal block of l_{s_1} has been colored red. In order not to give rise to more than two color changes in this literal block, the satisfaction tester must be followed by a red letter. This implies that l_{s_1} is positive if and only if its variable has been set to TRUE. Thus, the formula is satisfied. □

If we bound the size of the alphabet Σ instead of the number of colors we get a similar result.

Theorem 5 *PPW is NP-complete for* $|\Sigma| = 2$.

PROOF. We give a pseudo-polynomial reduction from 3-PARTITION (see [3], Lemma 4.1, pp. 101). Let $A = \{a_1, \ldots, a_{3m}\}$ denote the set of elements with size $s(a_1), \ldots, s(a_{3m}) \in \mathbb{Z}^+$ and $B \in \mathbb{Z}^+$ denote the bound of an instance of 3-PARTITION.

We construct an instance of PPW as follows. For each element a_i of size $s(a_i)$ we provide $s(a_i)$ times the letter L in color $f(a_i)$, where $f(a_i) \neq f(a_j)$ if $i \neq j$. Additionally, we provide $m - 1$ times the letter Z in color f_0. The word w

then consists of m *partition blocks* $b_1, \ldots, b_m$ of size B that contain the letter L and are separated by the letter Z (see Figure 4).

$$\underbrace{L \ldots L}_{B} Z \underbrace{L \ldots L}_{B} Z \ldots Z \underbrace{L \ldots L}_{B}$$

Fig. 4. The general structure of w.

Clearly, there exists a partition of A into m disjoint sets $S_1, \ldots, S_m$ such that $\sum_{a \in S_j} s(a) = B$ for $j = 1, \ldots, m$ if and only if there exists a coloring of w with $4m - 2$ color changes. □

5 k-regular instances

We now turn to upper bounds on the minimal number of color changes for an instance of PPW. It is easy to construct instances w of length n requiring $n-1$ color changes, thus in general, there is no non-trivial upper bound. Therefore, we study instances of a highly regular nature. Dealing with this challenging (and mostly unsolved) task first, may lead to new ideas for more general cases. Recall that for an instance of PPW we denote the initial reservoir of letter i in color j by $R(i, j)$ and the minimal number of color changes by $\gamma(w; f)$.

Definition 6 *Given a fixed integer $k \geq 1$, we call an instance of PPW* k-regular*, if $R(i, j) = k = \frac{n}{|\Sigma||F|}$ for all letters i and colors j (see Figure 5).*

A	C	B	B	C	C	A	C	A	B	B	A

Fig. 5. A 2-regular instance with $|\Sigma| = 3$ and $|F| = 2$.

Further, we denote the number of occurences of a letter l within a word w by $|w|_l$. We give an upper bound for $\gamma(w; f)$ in the simplest case first.

Lemma 7 *Suppose that we are given a k-regular instance of PPW with $|\Sigma| = |F| = 2$. Then $\gamma(w; f) \leq 2$ holds.*

PROOF. We suppose that $\Sigma = \{x, y\}$ and consider a sequence of consecutive letters $w^s = (w_s, w_{s+1}, \ldots, w_{s+\frac{n}{2}-1})$ of w with $|w^s| = 2k = \frac{n}{2}$, and $1 \leq s \leq \frac{n}{2} + 1$. Since $|w^1|_x + |w^{\frac{n}{2}+1}|_x = 4k$, there must exist a $1 \leq t \leq \frac{n}{2} + 1$ such that $|w^t|_x = k$. As $|\Sigma| = 2$, it follows that $|w^t|_y = k$ as well. We therefore can color all letters of w^t with one color, and all letters of $w \setminus w^t$ with the other. Note that $\gamma(w; f) = 1$, if $t = 1$ or $t = \frac{n}{2} + 1$, and $\gamma(w; f) = 2$ otherwise. □

We now consider the case of k-regular instances with bounded size of the alphabet Σ, and use Lemma 7 to prove Theorem 8 by induction on the size of the color set F.

Theorem 8 *Suppose that we are given a k-regular instance of PPW with $|\Sigma| = 2$. Then $\gamma(w; f) \leq 2(|F| - 1)$ holds.*

PROOF. Lemma 7 shows that Theorem 8 is true for $|F| = 2$. We assume that Theorem 8 is true for all values strictly smaller than $|F|$. We again suppose that $\Sigma = \{x, y\}$ and consider a sequence of consecutive letters w^s of w with $|w^s| = 2k = \frac{n}{|F|}$, and $1 \leq s \leq n - \frac{n}{|F|} - 1$. Again there must exist a t with $|w^t|_x = |w^t|_y = k$, so that we can color all letters of w^t with a single color. The remaining uncolored letters form a k-regular instance $w' = w \setminus w^t$ of length $|w'| = 2k(|F| - 1)$. By our assumption we can color the letters of w' with the remaining $|F| - 1$ colors, so that $\gamma(w'; f) \leq 2(|F| - 2)$. We therefore get $\gamma(w; f) \leq 2(|F| - 2) + 2 = 2(|F| - 1)$ for a coloring of w. $\square$

Bounding the size of the color set F instead, we were not able to show an upper bound for an optimal coloring like in Theorem 8. The examples we encountered suggest the following:

Conjecture 9 *Suppose that we are given a k-regular instance of PPW with $|F| = 2$. Then $\gamma(w; f) \leq |\Sigma|$ holds.*

A straightforward induction argument shows that Conjecture 9 is true for $k = 1$. Combining Theorem 8 and Conjecture 9 suggests the more general

Conjecture 10 *Suppose that we are given a k-regular instance of PPW. Then $\gamma(w; f) \leq |\Sigma|(|F| - 1)$ holds.*

The following example proves that the bound given in Conjecture 10 is tight if the conjecture is true.

Example 11 *Suppose that we are given a k-regular instance of PPW with a color set F and an alphabet $\Sigma = \{b_1, \ldots, b_{|\Sigma|}\}$ of the form*

$$w = (\underbrace{b_1 \ldots b_1}_{k|F|}\underbrace{b_2 \ldots b_2}_{k|F|} \ldots \underbrace{b_{|\Sigma|} \ldots b_{|\Sigma|}}_{k|F|}).$$

Then $\gamma(w; f) = |\Sigma|(|F| - 1)$ holds.

6 Concluding remarks

The minimization of color changes within a fixed sequence of colored letters is a new combinatorial problem. The complexity of the general problem seems to be sufficiently resolved. From a mathematical point of view even the structured cases of k-regular instances still seem to be surprisingly difficult. In addition to the conjectures proposed in Section 5 we pose the following problems, whose answers might provide a deeper insight into the structure of the problem.

Problem 12 *Given a k-regular instance $(w; f)$ of PPW, can one optimize or approximate $\gamma(w; f)$ in polynomial time? How about the case $k = 1$?*

We finally take a look at 1-regular instances with $|F| = 2$. This case looks quite simple, but we do not have any idea how to optimize or even approximate it within a constant factor. All we have is a formulation as an integer quadratic program (see Figure 6).

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}\sum_{i=1}^{n-1}(1 - v_i v_{i+1}) \\ \text{subject to} \quad & v_i v_j = -1 \text{ , if } w_i = w_j \text{ , and} \\ & v_i \in \{-1, 1\} \text{ for } i = 1, \ldots, n. \end{aligned}$$

Fig. 6. The integer quadratic program for 1-regular instances with $|F| = 2$.

Although this formulation is clearly inspired by the well known integer quadratic program for MAX CUT (see [5]), we were not able to deduce an approximation algorithm for our minimization problem.

Indeed, besides the dynamic program mentioned in Section 3 we do not know any efficient way to compute an optimal coloring for instances of PPW. One might expect that the natural greedy approach (when coloring w from the left to the right, keep the actual color as long as possible) should produce good results. Example 13 shows that in general this is not the case.

Example 13 *Suppose that we are given a 1-regular instance of PPW with $|F| = 2$ and $\Sigma = \{b_1, \ldots, b_{|\Sigma|}\}$ (with $|\Sigma|$ even) of the form*

$$w = (\underbrace{b_1 \ldots b_{|\Sigma|/2}}\,\underbrace{b_{|\Sigma|/2} \ldots b_{|\Sigma|}}\, b_{|\Sigma|}\,\underbrace{b_1 b_{|\Sigma|/2+1}}\,\underbrace{b_2 b_{|\Sigma|/2+2}}\ldots\underbrace{b_{|\Sigma|/2-1} b_{|\Sigma|-1}}).$$

The greedy algorithm colors w with $|\Sigma| = \frac{n}{2} = O(n)$ color changes, while the minimal number of color changes is always $\gamma(w; f) = 3$.

Likewise, an obvious improvement algorithm does not seem to exist. A natural improvement strategy based on an initial coloring could be the color exchange

between single letters or consecutive letter blocks (with regard to letter permutations). Example 14 shows that it is not sufficient to consider only consecutive letter blocks.

Example 14 *Suppose that we are given the* 1*-regular instance of PPW with* $|F| = 2$ *of the form* $w = (ABCBDDACEE)$ *and* $f = (0, 0, 1, 1, 0, 1, 1, 0, 0, 1)$. *The exchange strategy does not improve on* $\gamma(w; f)$ *unless we consider one of the non consecutive letter blocks* CE, CDE *or* BCE.

Acknowledgements

The authors would like to thank an anonymous referee for valuable comments.

References

[1] S. Spieckermann and S. Voß: *Paint Shop Simulation in the Automotive Industry.* ASIM Mitteilungen 54 (1996), pp. 367–380.

[2] Th. Epping and W. Hochstättler: *Abuse of Multiple Sequence Alignment in a Paint Shop.* Technical report zaik2001-418, Center of Applied Computer Science, 2001.

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

[4] Th. Epping and W. Hochstättler: *Storage and Retrieval of Car Bodies by the Use of Line Storage Systems.* Technical report btu-lsgdi-001.02, BTU Cottbus, 2002.

[5] M. X. Goemans and D. P. Williamson: *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming.* J. Assoc. Comput. Mach. 42 (1995), pp. 1115–1145.