OR Spektrum (1996) 18: 131–144

OR Spektrum

Heuristics for the Integer One-dimensional Cutting Stock Problem: a computational study

Gerhard Wäscher, Thomas Gau

Wirtschaftswissenschaftliche Fakultät, Betriebswirtschaftslehre – Produktion und Logistik, Martin-Luther-Universität Halle-Wittenberg, D-06099 Halle (Saale), Germany (Tel.: 0345-55-23420, e-mail: waescher@mluwiws4.wiwi.uni-halle.de)

Received: 17 January 1995 / Accepted: 3 November 1995

Abstract. In this paper the problem of generating integer solutions to the standard one-dimensional cutting stock problem is treated. In particular, we study a specific class of heuristic approaches that have been proposed in the literature, and some straightforward variants. These methods are compared with respect to solution quality and computing time. Our evaluation is based on having solved 4,000 randomly generated test problems. Not only will it be shown that two methods are clearly superior to the others but also that they solve almost any instance of the standard one-dimensional cutting stock problem to an optimum.

Zusammenfassung. In der vorliegenden Arbeit betrachten wir das Problem der Bestimmung ganzzahliger Lösungen für das Standardproblem der eindimensionalen Zuschnittplanung. Insbesondere werden eine spezielle Klasse heuristischer Lösungsverfahren, die in der Literatur beschrieben sind, sowie einige naheliegende Varianten dieser Verfahren vorgestellt. Auf der Grundlage eines numerischen Experiments, bei dem 4.000 Probleme zufällig erzeugt und gelöst wurden, werden die Verfahren miteinander verglichen und im Hinblick auf die Kriterien „Lösungsqualität" und „Rechenzeitbedarf" beurteilt. Dabei zeigt sich nicht nur, daß zwei Verfahren deutlich besser als die übrigen einzustufen sind, sondern auch, daß mit ihrer Hilfe nahezu jede Problemausprägung des klassischen eindimensionalen Zuschneideproblems optimal gelöst werden kann.

Key words: Cutting stock, integer solutions, heuristics, linear programming, column generation, numerical experiments

Schlüsselwörter: Zuschneideprobleme, Ganzzahligkeit, Heuristiken, Lineare Optimierung, Spaltengenerierung, Numerische Experimente

Correspondence to: G. Wäscher

1. Introduction

Cutting stock problems are among the earliest problems which have been studied through methods of Operational Research [22]. Since then every year there has been an increasing number of publications in the field (cf. the bibliographies by Dyckhoff and Wäscher [9], Dyckhoff and Finke [7], and Sweeney and Paternoster [32]). The majority of these publications deals with real-world applications from various industries [8]. Relatively few papers treat general methodological issues, even though there are still several interesting questions to be answered. One of these questions concerns the generation of integer solutions, which will be discussed here with respect to the one-dimensional cutting stock problem.

No exact algorithm is known that solves medium-size, practically relevant problem instances to optimality. Thus industrial integer cutting stock problems are usually dealt with through heuristics. However, many of these procedures seem to be of rather limited value. Woolsey [37], for example, reports that practitioners often come up with manually generated solutions which are better than those provided by heuristic methods. Especially in industries (steel industry, paper industry) where high-value material is being cut, such methods would not be accepted.

On the other hand, due to the complexity of the problem, there does not appear to exist an alternative to solving the integer cutting stock problem heuristicly. The purpose of this paper therefore is to describe existing heuristic procedures and to evaluate them with respect to solution quality and computing effort. We will confine ourselves to a specific class of heuristics, that starts from an initial optimal solution for the corresponding continuous relaxation of the cutting stock problem and proceeds to an integer solution in the "neighbourhood" of this solution. We concentrate on these procedures as in previous tests they have proven to be superior to other heuristic approaches [34].

The outline of the paper is as follows: In Chap. 2 a definition of the Integer One-dimensional Cutting Stock Problem is given. Existing models and algorithms for this

problem are briefly reviewed in Chap. 3. Chapter 4 contains some considerations on bounds for the optimum objective function value, that can be used as reference points for the evaluation of the solution quality of the procedures under discussion. The heuristics themselves are presented in Chap. 5. In order to evaluate the algorithms, extensive numerical experiments have been carried out. Details of the test procedure are given in Chap. 6. The results of the tests are outlined in Chap. 7, an analysis and an interpretation is presented in Chap. 8. In Chap. 9 we conclude with some final remarks.

2. Problem statement

The *Integer One-dimensional Cutting Stock Problem* (ICSP) can be defined as follows:

From a stock of material, which is available in a sufficiently large number of identical pieces of a single given length L (*standard length*), a set of m orders requiring d_i items of length (*order length*) l_i, $i = 1,\ldots, m$, has to be cut, such that the number of pieces needed to satisfy the orders is minimized.

According to the typology of cutting and packing problems introduced by Dyckhoff [6] this is a problem of type 1/V/I/R. Input and output of the cutting process are essentially identical with respect to all characteristics (quality, diameter etc.) but length. Therefore, all necessary cuts have to be performed in one single space dimension. Cutting problems of this particular type can be found in various industries, for example

- in the paper industry, where large master rolls have to be slit into sizes demanded by customers [1, 3, 20]
- in the steel industry, where steel bars have to be sheared into bars of shorter length [10, 35]
- in window production, where aluminium profiles have to be cut into sizes required as window frames [31]

etc. A comprehensive literature review of real-world one-dimensional cutting stock problems can be found in Dyckhoff and Finke [7].

An instruction that lists the items (i.e. the order lengths) to be cut from a piece of stock (i.e. the standard length) is called a *cutting pattern*. Obviously, the total length of items in a cutting pattern must not exceed the standard length L, the remaining part of the standard length represents *trim loss* and is treated as waste.

The fact that the demand for the different order sizes often is fairly high means that one can use identical cutting patterns for several standard lengths. As long as it does not increase the amount of input we allow *excess production* for any order. The pieces of length l_i that are provided beyond the required demand d_i, however, represent waste as does the trim loss. Under these conditions waste minimization and input minimization are equivalent goals [5, 33].

For both stock and orders we assume a discrete quantity measurement. In accordance with this assumption, we only allow the frequency of a cutting pattern, in which it is applied to the material stock, to be an integer number. This means that we are referring to cutting problems in which a piece of stock must either be cut completely, or, otherwise, be left unused. Neither is it permitted to cut only a fraction of it and return the rest back into stock (as is sometimes possible in the paper industry when cutting master rolls), nor are different cutting patterns feasible for one piece of stock.

Integrality constraints for the frequencies of cutting patterns are commonly found in real-world cutting problems (cf. the listings given in [7]). These constraints, on the other hand, make cutting problems difficult to solve to optimality. This is not at all surprising as the ICSP can be interpreted as a "*high multiplicity version*" of the Bin-Packing Problem [21], which is known to be NP-complete [12].

3. Models and algorithms for the ICSP

Several models for the ICSP are described in the literature [6]. Its transformation into an equivalent Bin-Packing Problem, for example, can be characterized as an *item-oriented* approach, because every item to be cut would be treated individually. However, as the number of items is equal to the total demand $\sum_{i=1}^{m} d_i$ of the ICSP and therefore may grow very large for real-world problems this is hardly a promising formulation. More appropriate is the so-called *pattern-oriented* approach. In this approach the order lengths are first combined into cutting patterns, for which – in a succeeding step – the frequencies are determined that are necessary to satisfy the demands. In particular, here it is sufficient to present the classic *Complete-Cut Model* only. For an alternative model formulation see the so-called *One-Cut Model* introduced independently by Rao [28] and Dyckhoff [4].

3.1. Complete-Cut Model

In order to present this model we introduce the following notation:

a_{ij} number of times order length l_i ($i = 1, 2,\ldots, m$) appears in cutting pattern j.

Any cutting pattern may then be characterized (apart from permutations in the arrangement of the order lengths) by a vector

$$(a_{1j}, a_{2j},\ldots, a_{mj}) \tag{1}$$

which satisfies

$$\sum_{i=1}^{m} l_i\, a_{ij} \le L, \tag{2}$$

$$a_{ij} \ge 0 \text{ and integer}, \quad i = 1, 2,\ldots, m. \tag{3}$$

To determine an optimal solution of an ICSP it is sufficient to concentrate on *effective* cutting patterns (1) only [5], i.e. patterns which – besides (2) and (3) – also satisfy the condition

$$L - \sum_{i=1}^{m} l_i\, a_{ij} < \min\{l_i \mid i = 1,\ldots, m\}. \tag{4}$$

Due to (4) an effective cutting pattern is characterized by the property that the length of the trim loss in the pattern is less than the smallest order length. Therefore, it is not possible to cut another order length from the remaining piece of standard length. That is why effective cutting patterns are also referred to as *complete cuts* [30, 33]. By introducing

n total number of complete cuts for a given ICSP;
x_j frequency of cutting pattern j ($j = 1,\ldots, n$), i.e. number of standard lengths which have to be cut according to pattern j;
x_0 total number of standard lengths to be cut,

the so-called *Complete-Cut Model* [30, 33] for the ICSP can be formulated as follows:

$$\text{Minimize} \quad x_0 = \sum_{j=1}^{n} x_j \tag{5}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}\, x_j \geq d_i, \quad i = 1,\ldots, m, \tag{6}$$

$$x_j \geq 0 \quad \text{and integer}, \quad j = 1,\ldots, n. \tag{7}$$

As the columns of the matrix $A = (a_{ij})$ simply represent all complete cuts (1), any optimal solution

$$(x_1^*, x_2^*,\ldots, x_n^*)$$

of the model (5)–(7) directly answers the question of how many times each pattern has to be applied. Constraints (6), also called *demand constraints*, guarantee that all demands are satisfied.

3.2. *Exact and heuristic algorithms*

In order to determine an optimal solution to a given ICSP one may think of generating the corresponding integer linear program (5)–(7) and apply one of today's advanced codes for *integer programming*. The authors have tested this approach on 111 problem instances (33 problems collected from the literature, 50 real-world coil-slitting problems from the steel industry and 28 real-world profile-cutting problems from window frame production). Even though single integer programs with up to 16,000 variables were solved by the software package (CPLEX 2.0) in reasonable computing time (90 seconds on a HP 9000/720 workstation), the approach turned out to be practicable for small problem instances, only (i.e. ICSP with less than 20 order lengths). It failed, for obvious reasons, when applied to medium-size and large problem instances. For example, a moderate-size instance ($m = 27$) of the ICSP, that has been published by Haessler [19] and which represents a set of real-world data, results in a model with more than 75 million variables.

Thus, for medium-size and large instances of the ICSP only heuristic approaches are applicable. Here we concentrate on a specific class of heuristics, which determine an optimal solution for the continuous relaxation of the Complete-Cut Model at first. In contrast to the generation of an integer solution this can be done in reasonable computing time, at least for the problem dimensions that are relevant to practical applications. In particular, it is not necessary to generate all columns (cutting patterns) of the Complete-Cut Model (5)–(7) in advance. Instead one may apply the *column-generation procedure* developed by Gilmore and Gomory [17]. This is a variant of the revised simplex method that has been adopted for the continuous relaxation of the ICSP. It starts from an initial set of m cutting patterns that represent a (column) basis of A (basic patterns). At each simplex iteration one of the basic patterns is replaced by a new cutting pattern that will improve the current basic solution. It can be determined by solving a knapsack problem of which the constraints are given by (2) and (3) while the coefficients of the objective function are the current values of the dual variables related to constraints (6), for details see [2].

From the optimal solution of the continuous relaxation of the ICSP, which generally is non-integer, the heuristic procedures to be discussed here proceed to an integer solution in the "neighbourhood", where they terminate. In previous numerical tests [34] even simple rounding techniques outperformed classic approaches to the ICSP like the *Repeated-Pattern-Exhaustion (RPE) Technique* [26]. Also for many problem instances it was proven that the solutions generated were optimal. Therefore the authors decided to examine this class of heuristic algorithms more thoroughly.

4. Bounds and the IRU-Property

In order to solve the ICSP in reasonable computing time, it is essential to have good bounds for the optimum objective function value available. Good bounds are also necessary as reference points for the evaluation of the solution quality of heuristics. Let

$$(\tilde{x}_1^*, \tilde{x}_2^*,\ldots, \tilde{x}_n^*) \tag{8}$$

be an optimal solution of the continuous relaxation of the Complete-Cut Model (5)–(7) and

$$\tilde{x}_0^* = \sum_{j=1}^{n} \tilde{x}_j^*$$

be the corresponding objective function value. As the optimum objective function value x_0^* of the integer program (5)–(7) is an integer number itself, a *lower bound* LB (x_0^*) for x_0^* is given by $\lceil \tilde{x}_0^* \rceil$, where $\lceil \tilde{x}_0^* \rceil$ is the smallest integer greater or equal to $\tilde{x}_0^*$:

$$\text{LB}\,(x_0^*) = \lceil \tilde{x}_0^* \rceil \tag{9}$$

An *upper bound* can be derived from the property that by rounding up the non-integer components of (8) a feasible solution for the ICSP is obtained. Therefore an upper bound UB (x_0^*) for the optimum objective function value of the ICSP is given by

$$\text{UB}\left(x_0^*\right) = \sum_{j=1}^{n} \lceil \tilde{x}_j^* \rceil$$

However, experience has shown that this upper bound can be far from the optimum objective function value [34]. In particular, practitioners involved in solving problems of

this type argued that the optimum objective function value x_0^* of any real problem instance of the ICSP appears to be given by (9). A problem instance for which this property holds, i.e.

$$x_0^* = \lceil \tilde{x}_0^* \rceil \tag{10}$$

is said to comprise the *Integer Round-Up Property* (IRU-Property). Equivalently (10) can be formulated as

$$x_0^* - \tilde{x}_0^* < 1$$

in which $x_0^* - \tilde{x}_0^*$ may be referred to as the *integrality gap*. Marcotte [23], however, has shown that the IRU-Property does not hold for all instances of the ICSP, even though, following Marcotte's line of argumentation, it seems likely that real world cutting stock problems have this property. For example, in a test of 50 real-world cutting problems and 49 problem instances published in the literature the IRU-Property has been shown to hold for all of these problems but one [36]. In fact, counterexamples to the IRU-Property hardly ever arise from randomly generated problem instances or from real-world applications of the ICSP. Therefore it is not surprising that the largest integrality gap $\left(\frac{16}{15} \approx 1.066..\right)$ found so far is very close to one (see Gau [14] for a complete description of this and some other instances of the ICSP. All instances are characterized by an integrality gap larger than the ones published by Fieldhouse [11] or Scheithauer and Terno [29]). Therefore, the lower bound (9), that we also call *IRU-Bound,* provides an excellent reference point for the evaluation of the solution quality of heuristics for the ICSP.

5. Heuristic approaches to the ICSP

The procedures to be presented here generate an integer solution from an optimal solution (8) for the relaxed Complete-Cut Model. We assume that at least one component $\tilde{x}_j^*$ of (8) is non-integer. Otherwise the respective instance of the ICSP would have been solved to optimality, already. The algorithms described in the literature for the generation of an integer solution basically proceed according to one of the principles described below and will be referred to here as the *Basic Patterns Approach* (B), the *Residual Problem Approach* (R) and the *Composite Approach* (C).

5.1. Basic Patterns Approach

In the Basic Patterns Approach the cutting patterns (*basic patterns*) that correspond to the basic variables of the optimal solution (8) for the relaxed Complete-Cut Model are identified. The number of these basic patterns is m, i.e. equal to the number of order lengths. That is due to the fact that any basic solution of (6) contains m basic variables $x_{j_1},\ldots,x_{j_m}$ $(1 \le j_1 < j_2 < \cdots < j_m \le n)$, because there are m linearily independent constraints of type (6). For simplicity of exposition we denote the index set $\{j_1,\ldots,j_m\}$ of the basic variables by $\mathcal{J}_B$, i.e. $\mathcal{J}_B = \{j_1,\ldots,j_m\}$. Only the basic patterns will be further considered for the generation of an integer solution

$$(x_1^*, x_2^*,\ldots,x_n^*), \quad x_1^*, x_2^*,\ldots,x_n^* \text{ integer} \tag{11}$$

to the corresponding Complete-Cut Model. Therefore, the non-basic variables are fixed at zero

$$x_j^* = \tilde{x}_j^* = 0 \quad \text{for} \quad j = 1, 2,\ldots,n \quad \text{with} \quad j \notin \mathcal{J}_B,$$

while the frequencies

$$(\tilde{x}_{j_1}^*, \tilde{x}_{j_2}^*,\ldots,\tilde{x}_{j_m}^*) \tag{12}$$

for the basic variables have to be adjusted such that

- the $\tilde{x}_j^*$ $(j \in \mathcal{J}_B)$ become integers x_j^* and
- the demands are still satisfied, i.e.

$$\sum_{j \in \mathcal{J}_B} a_{ij}\, x_j^* \ge d_i \quad \text{for} \quad i = 1,\ 2,\ldots,m. \tag{13}$$

For the adjustment of the frequencies $\tilde{x}_j^*$ $(j \in \mathcal{J}_B)$ several options may be considered:

▷ Procedure BRUSIM
The simplest procedure consists of simultaneously rounding up any non-integer component $\tilde{x}_j^*$ of (8) to the nearest integer, i.e.

$$x_j^* = \lceil \tilde{x}_j^* \rceil \text{ for } j \in \mathcal{J}_B.$$

The advantages of this procedure are obvious: It is extremely fast and will immediately result in a feasible solution to the Complete-Cut Model (5)–(7). This is because of the fact that by rounding up the components of (8), the demand constraints (6) cannot be violated as (8) already satisfies these constraints and therefore, by rounding up, the inequalities still remain valid. On the other hand, it has been observed that the procedure often results in integer solutions (11) far from the optimum objective function value.

▷ Procedure BRURED
Rounding up the non-integer components of (8) simultaneously often creates supplies of order lengths that exceed the respective demands by far. Then it may be feasible to reduce the frequencies of some cutting patterns to smaller integer values, again, without violating the demand constraints (6). Neumann and Morlock [25] suggest to check whether the frequencies $x_{j_1}, x_{j_2},\ldots, x_{j_m}$ can be reduced by one unit without causing a violation of the demand constraints. As soon as such a variable x_k $(k \in \mathcal{J}_B)$ has been identified, i.e. a variable for which

$$a_{ij}(x_k - 1) + \sum_{\substack{j \in \mathcal{J}_B \\ j \ne k}} a_{ij}\, x_j \ge d_i, \quad i = 1, 2,\ldots, m$$

holds, x_k is reduced by one unit.

▷ Procedure BRUSUC
As part of a more complex solution algorithm for the ICSP (that will be dealt with in detail in Chap. 5.3), Stadtler [31] suggests to determine an optimal solution for a continuous relaxation of the Complete-Cut Model of the original cutting stock problem that consists of the basic patterns only, i.e.

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}_B} \tilde{x}_j \tag{14}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{J}_B} a_{ij} \tilde{x}_j \geq d_i, \quad i = 1,\ldots,m, \tag{15}$$

$$\tilde{x}_j \geq 0, \quad j \in \mathcal{J}_B, \tag{16}$$

and of which certain variables are fixed ex ante:

– all variables of (12) which have taken integer values are fixed at these values:

$$x_j^* = \tilde{x}_j^*, \quad j \in \mathcal{J}_B, \tilde{x}_j^* \text{ integer}; \tag{17}$$

– the variable $\tilde{x}_k^*$ of (12) with the largest fractional value is rounded up to the next integer:

$$x_k^* := \lceil \tilde{x}_k^* \rceil. \tag{18}$$

Having solved (14)–(16) with the additional constraints of types (17) and (18), not necessarily all variables will turn out to be integers. In such cases the process is repeated by fixing additional variables according to (17) and (18). As at least one more variable is fixed in each iteration, the procedure terminates with an all-integer solution to (14)–(16) after m iterations at most.

This method can also be interpreted as a kind of rounding procedure. Rounding is achieved by adding constraints of type (18). In contrast to the two procedures described before, here the variables are rounded up successively, which – in connection with reoptimizing the system in each iteration – represents a more careful way of rounding.

Note that in each iteration only continuous linear programs have to be solved. However, as previously fixed variables are never released, the final integer solution may not be an optimal integer solution for the problem (14)–(16).

▷ Procedure BOPT

From the introduction of the optimization problem (14)–(16) follows another straightforward procedure for the adjustment of the frequencies (8) to integer values. As the number of variables in (14)–(16) is relatively small (equal to m, the number of order lengths) one may think of generating an optimum integer solution for this problem (*Basic Patterns Problem*) directly. In principle, this can be done by any of the solution methods (cutting plane-methods, Branch & Bound) which have been developed for all-integer linear programming problems. However, one has to keep in mind that an optimal integer solution for the problem (14)–(16) will only result in a local optimum for the original cutting stock problem, as the solution space of the latter is artificially restricted to the basic patterns of the optimal solution of the relaxed Complete-Cut Model.

BOPT and BRUSIM define the range for the objective function values that can be achieved by the various procedures from this Basic Patterns Approach. The solution generated by BOPT must be at least as good as the ones obtained by the other procedures. On the other hand, BOPT, BRURED and BRUSUC cannot generate solutions which are worse than those obtained from BRUSIM. Therefore, with respect to the quality of the solution, BOPT is clearly superior to BRUSIM, BRURED and BRUSUC, respectively.

However, as BOPT requires solving an integer program it can also be expected to consume more computing time than the other procedures. BRUSUC may also be time-consuming as it requires to solve a series of (at most m) real-value linear programs. On the other hand, with BRUSIM the components of (8) simply have to be rounded up, which hardly takes any computing time even for large cutting stock problems. BRURED uses the same rounding procedure, but additionally requires checking whether the frequencies can be reduced. Nevertheless, due to the simplicity of this reduction procedure, BRUSIM and BRURED will cause identical computing times.

In any case, a trade-off between solution quality and computing time becomes evident. Thus, especially for large cutting stock problems it will be necessary to examine whether the procedures BOPT and BRUSUC are still practicable.

5.2. *Residual Problem Approach*

The Residual Problem Approach also starts from an optimal solution (8) for the continuous relaxation of the Complete-Cut Model. However, all non-integer components of (8) are rounded down. By doing so, one obtains a vector

$$\left(\lfloor \tilde{x}_1^* \rfloor, \lfloor \tilde{x}_2^* \rfloor, \ldots, \lfloor \tilde{x}_n^* \rfloor \right) \tag{19}$$

of integer frequencies $\lfloor \tilde{x}_j^* \rfloor$, $j = 1, 2, \ldots, n$ ($\lfloor \tilde{x}_j^* \rfloor$: largest integer smaller or equal to $\tilde{x}_j^*$) and the corresponding objective function value

$$\sum_{j=1}^{n} \lfloor \tilde{x}_j^* \rfloor. \tag{20}$$

(19) is not a feasible solution for the ICSP, because– as a result of rounding down the non-integer components of (8) – it does not satisfy all the demand constraints (6) any longer. Cutting $\sum_{j=1}^{n} \lfloor \tilde{x}_j^* \rfloor$ standard lengths according to the patterns and frequencies given by (19) will provide

$$\sum_{j=1}^{n} a_{ij} \lfloor \tilde{x}_j^* \rfloor$$

units of order length l_i ($i = 1, 2, \ldots, m$). Thus

$$\max\left(0, d_i - \sum_{j=1}^{n} a_{ij} \lfloor \tilde{x}_j^* \rfloor\right), \quad i = 1, 2, \ldots, m \tag{21}$$

units of order length l_i are still missing, such that another, *residual problem* arises. The basic idea of the approach to be discussed in this chapter is to solve this residual problem separately and combine its solution with the one obtained from rounding down the solution for the relaxation of the original ICSP. The corresponding objective function value for this combined solution is simply given by the sum of (20) and the objective function value obtained for the residual problem.

If modelled explicitly as a Complete-Cut Model the residual cutting problem would result in a model that only differs from that for the original ICSP with respect to the right-hand sides. The left-hand sides are identical, because both original and residual problem contain identical sets of order lengths and, thus, result in the same set of cutting patterns (columns of the Complete-Cut Model). The original demands d_i in (6), however, are replaced by the demands (21) of the residual problem. In contrast to the original problem, the total number of items demanded of each order length in the residual problem is usually much smaller than the corresponding demand in the original problem. Because of these "low multiplicities" it seems more promising to treat the residual problem not as an ICSP but as a Bin-Packing Problem. From the various standard methods for the Bin-Packing Problem we have chosen an exact and a heuristic algorithm, defining the following two procedures:

▷ Procedure ROPT
The residual problem is solved to an optimum. However, due to having split the original ICSP into two subproblems, the combination of this solution and the one obtained by rounding down the solution of the original relaxed Complete-Cut Model does not necessarily define an optimal solution for the original ICSP (for a more detailed discussion see Gau [14]). Also the residual problem may be rather large (in terms of the number of items), in particular when the number m of order lengths is large. As the Bin-Packing Problem is NP-complete, for these problems the computing effort of this procedure may turn out to be prohibitive.

▷ Procedure RFFD
In order to reduce the computing effort instead of solving the residual problem to an optimum one may use a heuristic algorithm. From initial tests which have been performed with several simple heuristics, the *First-Fit-Decreasing (FFD)* and the *Best-Fit-Decreasing (BFD) Algorithm* have proven to be superior to others. On the other hand, there was no significant difference between the two methods in terms of the solution quality, but as FFD was faster than BFD, it has been decided to implement the FFD algorithm for the residual problem.

However, numerical tests with the FFD algorithm applied to large ICSP have shown that the objective function value obtained may be far from the optimum [34]. Gau [13, 14] therefore presented an algorithm that treats the residual problem as an ICSP, too.

▷ Procedure RSUC
In this procedure the residual problem is relaxed again with respect to the integrality constraints for the variables and solved by the column-generation technique. Rounding down the non-integer components of the optimal solution leaves another residual problem that is treated in the same way etc. The process terminates when rounding down the solution components only provides zero frequencies. In such a case usually the demand for only a few items still has to be satisfied, so that the corresponding residual problem can be solved optimally by an exact bin-packing algorithm. (For a discussion of some interesting worst-case properties of the procedure RSUC see [14]).

The procedure RSUC involves solving successively a series of real-value linear programs, arising from the residual problems. Even though these problems are characterized by a decreasing number of items, in order to speed up the procedure one may try to reduce the number of problems/linear programs to be solved. Thus RSUC has been implemented such that for any residual problem, at first, it is attempted to find an optimal (integer) solution by an exact bin-packing algorithm.

As the residual problems may still be rather large (in terms of the numbers of items), exact procedures may be very time consuming. Therefore, an implicit time limit has been set. Only if the exact bin-packing algorithm fails to solve the problem optimally within this limit, then the residual problem is treated as described above.

5.3. Composite Approach

Stadtler [31] has presented a procedure that combines ideas both from the Basic Patterns and the Residual Problem Approach. He starts with the BRUSUC procedure to generate a feasible solution of the ICSP. At this stage usually excess production occurs, which will be reduced iteratively. In order to do so, a cutting pattern is identified that contains the largest portion of order lengths in oversupply. The frequency of this pattern is reduced by one. Then this step is repeated until no more oversupply exists.

However, now some order lengths may be short in supply, giving rise to a residual problem as in the Residual Problem Approach. (Nevertheless, worst-case results similar to the ones for RSUC are no longer available [14]). Stadtler solves the residual problem heuristicly through the application of the FFD algorithm. Any other bin-packing algorithm may be used instead. From the various methods for solving the residual problem the authors have again chosen an exact algorithm. The Composite Approach therefore is represented by two procedures:

▷ CSTAOPT
BRUSUC is applied to the solution (8) of the relaxed Complete-Cut Model, following by Stadtler's reduction procedure. The residual problem is solved to an optimum by an exact algorithm.

▷ CSTAFFD
This procedure works like CSTAOPT. Instead of an exact algorithm, the FFD heuristic is applied to the residual problem.

6. Test procedure

In order to evaluate the procedures described in chapter 5, 4,000 randomly generated problem instances have been solved by these methods. The problem generator that has been used is described in [15]. Details of the test problems and implementation are given below.

6.1. *Problem parameters and classes of problem instances*

In order to generate several classes of problem instances the parameters of the ICSP have been varied in different ways.

(1) Problem size
The size of an instance of the ICSP is expressed by the number of order lengths, m. The different values to which m has been fixed in our tests were $m = 10, 20, 30, 40, 50$. According to our experience from real-world applications almost any problem of practical relevance should be covered with $m = 50$. Also all the real-world problem instances published in the literature are of smaller size [36].

(2) Standard lengths and order lengths
The standard length L has been set for all problem instances to 10,000 length units, while different cases have been distinguished with respect to the order lengths l_i. We modelled the order lengths as uniformly distributed integer random variables which were allowed to vary between one and a certain percentage b of L, i.e. $l_i \in [1, b \cdot L]$. The reason behind this was that we assumed that the relative size of the order lengths (in relation to L) would effect the performance of at least some of the approaches. The values of the length factor b that have been investigated were $b = 0.25, 0.5, 0.75$ and 1.0.

The individual demands d_i have been treated as random variables as well. They were generated in a two-step procedure. At first a constant total demand (i.e. the total number of items) T has been fixed for any problem instance of a given size m. Then – individually for any problem instance – the size T has been randomly distributed between the m orders.

(3) Total demand
The total demand has been defined as a function of m and an additional factor $\bar{d}$ ($\bar{d}$: integer):

$$T = m \cdot \bar{d}$$

$\bar{d}$ can be interpreted as average demand per order length or as "*multiplicity factor*". By varying this factor the character of the problem instances can be changed from the Bin-Packing Type ($\bar{d}$: small) to the Cutting-Stock Type ($\bar{d}$: large). The cases for which $\bar{d}$ has been investigated were $\bar{d} = 10$ and $\bar{d} = 50$.

(4) Individual demands
In order to distribute the total demand between the orders, m uniform random numbers $R_1, \ldots, R_m$ have been generated. The demands d_i have then been fixed as follows

$$d_i = \left\lfloor \frac{R_i}{R_1 + R_2 + \cdots + R_m} \right\rfloor \cdot T \quad \text{for} \quad i = 1, \ldots, m-1$$

while the remaining units were allocated to order length l_m:

$$d_m = T - \sum_{i=1}^{m-1} d_i$$

By combining the different values for the problem parameters 40 classes of problem instances are defined. In order to facilitate the characterization of these classes, later we will use a tupel (m, b, $\bar{d}$). For each class 100 problem instances have been generated so that the heuristics were tested on a total number of 4,000 problem instances.

6.2. *Implementation*

Except for BOPT all tests have been performed on an IBM-compatible 486x/66 PC (8 MB core memory) under DOS 6.0. Due to long computing times, the tests for BOPT have been run on a HP 9000/720 workstation (57 MIPS, 16 MB core memory) under HP-UX 9.0.

In order to reduce the influence on computing times that may arise from the individual style of encoding, the authors have tried to integrate standard software and other reliable, publicly available codes whenever possible. For all programs FORTRAN 77 has been used as the encoding language.

Specific features in the implementation of the different algorithms and procedures are as follows:

(1) Column generation
The implementation of the column-generation algorithm followed the procedure described in Gilmore and Gomory [17, 18]. Initial solutions were generated by means of an FFD algorithm that has been modified for the characteristics of the ICSP [2]. The Knapsack Problem that arises in each iteration has been solved with a code named MTU2, that is provided on a disk with the book by Martello and Toth [24].

(2) BRUSUC
BRUSUC requires solving a sequence of relatively small (real-value) linear programs. For this purpose the revised simplex method as described in Press et al. [27] has been implemented.

(3) BOPT
For the integer programs that have to be solved within the BOPT procedure, a professional code, CPLEX 2.0, has been chosen. CPLEX uses a Branch & Bound method for the generation of integer solutions. Initial tests with this procedure have shown that for certain problem instances it may not be possible to generate an optimal solution in reasonable computing time. Thus, the number of nodes has been limited to 50,000.

(4) ROPT
In order to solve the residual Bin-Packing Problems optimally the authors referred to the MTP algorithm which is also provided as a computer code with the book by Martello and Toth [24]. MTP can be characterized as a Branch & Bound method. It starts from a heuristicly generated solution. By backtracking then it is attempted to improve the solution. In order to avoid unacceptable computing times the number of backtrackings has been limited to 25,000.

(5) RFFD
For residual problems that had to be solved by the FFD-algorithm we used an implementation similar to that described in Chvátal [2].

(6) RSUC
Residual problems which had to be solved as linear programs were tackled using the column-generation technique as presented above. Residual problems to be treated by an exact bin-packing algorithm were solved by MTP, again. Apart from the last step of the procedure (rounding down provides zero frequencies), the number of backtrackings was limited to 2,500. For the last step 25,000 backtrackings were permitted.

(7) CSTAOPT and CSTAFFD
Residual problems were solved "optimally" by MTP (limited to 25,000 backtrackings) and heuristicly by Chvátal's modification of the FFD algorithm.

7. Results

The results of the numerical experiments will be presented in three stages. At the beginning a few remarks concerning column generation and quality of the IRU-Bound will be made. Then the relative performance of the different procedures will be discussed. Finally, an analysis is presented of how the solution quality of the best methods is affected by the different problem parameters.

7.1. Initial remarks

7.1.1. Column generation. The procedures presented for the ICSP have in common that they all start from an initial solution for the relaxation of the corresponding Complete-Cut Model. The computing effort caused by the column-generation algorithm therefore has to be regarded as a decisive factor for the practicability of these methods. Table 1 presents the computing times for different classes of problem instances.

The data is not differentiated with respect to the mulitplicity factor, as no significant differences in computing times have been observed for $\bar{d} = 10$ and $\bar{d} = 50$. It can be seen from Table 1 that computing times are determined by the relative length of the orders in the first place. Problem instances for which the size of the largest order length was limited to half of the standard length have proven to be relatively difficult. On the other hand, even for the class of problem instances (50/50/$\bar{d}$) which required the largest amount of computing time, the average computing time per problem (on a PC!) was only 32.4 s. For problems with 50 different order lengths this seems to be acceptable.

Table 1. Computing times for column generation (numbers indicate total CPU-seconds for 200 problem instances each)

Length factor b	Problem size m				
	10	20	30	40	50
0.25	85	329	829	917	1476
0.50	14	145	837	2611	6485
0.75	8	20	65	183	376
1.00	8	12	23	42	73

7.1.2. IRU-Bound. In evaluating the absolute solution quality of the different heuristics it is helpful not only to have a bound for the optimum objective function value available but also to be able to estimate how good the bound is. The IRU-Bound has proven to be an excellent reference point in our numerical experiments. For 3,996 out of 4,000 problem instances it could be shown that the IRU-Bound provided the optimum objective function value. To a large extent this result was achieved because at least one of the heuristics came up with an objective function value equal to the lower (IRU-)Bound. In most other cases in which none of the algorithms provided such a solution, the IRU-Property has been proven by the application of additional, partially time-consuming techniques (full-size implementations of Complete-Cut and One-Cut Models, enumeration techniques etc.). For the remaining four problem instances it is not known whether the IRU-Property holds. Nevertheless, the deviation of the optimum objective function value from the IRU-Bound is limited by one for all these problems.

7.2. Performance of the heuristic approaches

7.2.1. Overview. The results of our numerical experiments are summarized in Tables 2, 3, and 4. Table 2 shows the number of problem instances for which the different procedures have generated a solution with an objective function value equal to the lower bound (i.e. for which the solution has been proven to be optimal). For $m = 10,\ldots, 50$ each class contains 800 problem instances. Additionally, we have listed the computing times that were necessary to solve all problem instances in a class. In particular, we quote the total computing times (i.e. computing times for both column-generation and heuristic procedure) and, separately, the computing times for the respective heuristic procedure as such.

Table 3 informs on the relative solution quality of the different procedures. Listed are both the number of problem instances for which procedure i generated a solution superior to that of procedure j and the number of instances for which the solution was inferior.

Table 4 presents some additional information on the absolute solution quality, i.e. the average deviation from the IRU-Bound (number of stock lengths per problem; average of differences between objective function value obtained and IRU-Bound) as well as the maximum deviation.

7.2.2. Basic Patterns Approach. With respect to the quality of the solutions generated the BOPT procedure is clearly superior to any other method using the Basic Patterns Approach. For 1,472 problem instances (36.8%) an optimal solution has been found, almost twice as often as using the second-best procedure BRURED (821 problem instances; 20.5%). However, on the other hand, computing times for BOPT were extensive. With a limit of 50,000 nodes for the Branch & Bound algorithm the average computing time per problem on a HP 9000/720 workstation was 1.02 s for instances of size $m = 10$ but 148.08 s for instances of size $m = 50$ (excluding column generation). It is interesting to note that for several problem instances the

Table 2. Absolute performance (solution quality and computing times) of the heuristic procedures

Procedure	Problem size											
	m=10		m=20		m=30		m=40		m=50		All problems	
	# of problems solved optimally	computing times [s]	# of problems solved optimally	computing times [s]	# of problems solved optimally	computing times [s]	# of problems solved optimally	computing times [s]	# of problems solved optimally	computing times [s]	# of problems solved optimally	computing times [s]
BRUSIM	93	115 / 0	47	506 / 0	23	1,754 / 0	15	3,753 / 0	10	8,410 / 0	188	14,538 / 0
BRURED	325	115 / 0	193	506 / 0	129	1,754 / 0	94	3,753 / 0	80	8,410 / 0	821	14,538 / 0
BRUSUC	257	116 / 1	142	558 / 52	74	1,995 / 241	53	4,506 / 753	52	10,168 / 1,758	578	17,343 / 2,805
BOPT	428	– / 814*	321	– / 1,427*	262	– / 49,704*	251	– / 111,691*	210	– / 118,466*	1,472	– / 282,102*
ROPT	796	162 / 47	788	813 / 307	782	2,351 / 597	770	4,958 / 1,205	738	10,603 / 2,193	3,874	18,887 / 4,349
RFFD	779	115 / 0	758	506 / 0	743	1,754 / 0	726	3,753 / 0	695	8,410 / 0	3,701	14,538 / 0
RSUC	797	129 / 14	792	691 / 185	790	2,142 / 388	779	4,722 / 969	763	10,786 / 2,376	3,921	18,470 / 3,932
CSTAOPT	758	153 / 38	754	579 / 73	731	2,147 / 393	732	4,673 / 920	734	10,307 / 1,897	3,709	17,859 / 3,321
CSTAFFD	752	116 / 1	746	560 / 54	715	2,010 / 256	716	4,565 / 812	701	10,374 / 1,964	3,630	17,625 / 3,087

t_{cg} / t_h

t_{cg}: total CPU-seconds (PC 486-DX, 66 MHz; 800 problem instances) for column generation and heuristic
t_h: total CPU-seconds for heuristic only (800 problem instances), PC 486-DX, 66 MHz; *: Workstation HP 9000/720, 57 MIPS

Table 3. Relative performance (solution quality) of the heuristic approaches (all 4,000 problem instances)

Procedure i	Procedure j								
	BRUSIM	BRURED	BRUSUC	BOPT	ROPT	RFFD	RSUC	CSTAOPT	CSTAFFD
BRUSIM		0 / 3,649	0 / 3,385	0 / 3,673	0 / 3,812	0 / 3,812	0 / 3,812	0 / 3,747	0 / 3,747
BRURED			1,218 / 997	48 / 2,218	0 / 3,176	0 / 3,155	0 / 3,177	92 / 3,084	93 / 3,079
BRUSUC				31 / 2,602	0 / 3,412	2 / 3,382	0 / 3,416	0 / 3,311	1 / 3,303
BOPT					0 / 2,446	14 / 2,338	0 / 2,470	120 / 2,423	126 / 2,386
ROPT						173 / 0	1 / 48	268 / 103	307 / 63
RFFD							0 / 220	242 / 250	257 / 186
RSUC								275 / 63	328 / 37
CSTAOPT									80 / 0
CSTAFFD									

c_{ij} / c_{ji}

c_{ij}: number of problem instances for which procedure i generated a solution superior to that generated by procedure j
c_{ji}: number of problem instances for which procedure j generated a solution superior to that generated by procedure i

Table 4. Average deviation and maximum deviation from IRU-Bound (number of stock lengths per problem)

Procedure	Average deviation from IRU-Bound	Maximum deviation from IRU-Bound
BRUSIM	8.854	32
BRURED	3.056	18
BRUSUC	3.020	20
BOPT	1.950	23
ROPT	0.032	1
RFFD	0.075	1
RSUC	0.020	1
CSTAOPT	0.073	1
CSTAFFD	0.093	2

node limit did terminate the solution process before an optimal integer solution for the Basic Patterns Problem has been found. This not only explains the fact that for 31 (48) problem instances the objective function value was inferior to that from BRUSUC (BRURED) but also that the maximum deviation from the IRU-Bound for BOPT is larger than the one obtained for BRURED and BRUSUC (cf. Table 4). Due to these properties BOPT cannot be looked upon as a competitive procedure.

Amongst the remaining procedures of the Basic Patterns Approach BRURED is superior to the others. BRURED is not only characterized by the fact that its computing times are negligible but it also generates optimal solutions more frequently than BRUSUC and BRUSIM. Nevertheless, this does not mean that BRURED is superior for any problem instance. As can be seen from Table 3, BRUSUC generated better solutions for 997 problem instances, while BRURED was superior to BRUSUC in 1,218 cases. BRUSIM, finally, is clearly inferior to BRURED, as with the same amount of computing time it did not manage to solve a single problem instance better than BRURED.

However, even BRURED is unlikely to be competitive compared with procedures based on the Residual Problem Approach or the Composite Approach. For example, with the same amount of computing time RFFD has generated an optimal solution for 3,701 of the 4,000 problem instances (BRURED: 821 problem instances). There has been no problem instance for which BRURED was superior to RFFD. To conclude: The Basic Patterns Approach, in any variant, is unlikely to be a competitive approach to the ICSP.

7.2.3. Residual Problem Approach. The superiority of the Residual Problem Approach to the Basic Patterns Approach does not only become evident from the number of problems for which a better solution (and particularly an optimal solution) has been found, but also from the devi-

ation of the objective function values from the respective IRU-Bounds (cf. Table 4). Also it is worth mentioning that the largest deviation from the IRU-Bound was just one unit, i.e. each of the three procedures solved any of the 4,000 problem instances with at most one unit from the optimum objective function value. Among the three different procedures based on the Residual Problem Approach, RFFD obviously is the fastest. The computing times needed for the application of the FFD heuristic to the residual problems are so short that they can be neglected. Surprisingly though, 3,701 out of the 4,000 problem instances (92.5%) have been solved to an optimum. In order to increase this number of optimally solved problem instances one has to invest more computing time. By using another 4,349 s (for all 4,000 problem instances), i.e. 1.09 seconds on average per problem, through the application of ROPT this number is brought up to 3,874 (96.9%). By the application of RSUC we obtained 3,921 optimally solved problem instances (98.0%) for only 3,932 additional seconds of computing time (0.983 s per problem). This order of precedence is based on the solution quality of the three methods in terms of the number of problem instances solved optimally. It is also confirmed by a comparison of the procedures with respect to the average deviation from the IRU-Bound (cf. Table 4) and their relative performance. As can be seen from Table 3, for each of the 4,000 problem instances RSUC has been as least as good as RFFD, while it has only been beaten on one problem instance by ROPT. Better results for ROPT compared to the ones of RSUC, nevertheless unlikely to arise, are due to potential deviations from the optimum objective function value caused by decomposition (for details see [14]).

One may argue now that the solution quality of ROPT might improve if a larger number of backtrackings for MTP was allowed, which has been limited to 25,000 in the experiments. However, for our test problems this presumption could not be verified for realistic computing times. The authors ran additional tests with the maximum number of backtrackings set to 100,000. The computing time increased by some 300%, however, only improving the solution quality for 8 problem instances.

7.2.4. Composite Approach. Out of the two composite procedures CSTAOPT seems to be preferable. For almost the same (total) amount of computing time CSTAOPT solved 79 additional problem instances to an optimum (cf. Table 2). The superior solution quality of CSTAOPT is also reflected in the average/maximum deviation from the IRU-Bound (cf. Table 4) and by the direct comparison of the two procedures (cf. Table 3).

In relation to RSUC the solution quality of CSTAOPT is slightly worse. CSTAOPT solved 3,709 problem instances (92.7%) to an optimum, while RSUC managed to come up with an optimal solution for 3,921 cases (98.0%). The average deviation from the IRU-Bound (CSTAOPT: 0.073; RSUC: 0.020) presents a similar picture. A noteworthy aspect is that CSTAOPT is not inferior to RSUC for every problem instance (cf. Table 3). On the contrary, a relatively large number of problems (63) has been solved better by CSTAOPT. It will therefore be interesting to see whether it is possible to identify conditions under which CSTAOPT is superior.

7.3 Effects of problem parameter variations on RSUC and CSTAOPT

RSUC and CSTAOPT have been identified as the best procedures of their respective classes of approaches. We will now try to analyze how the relative performance of RSUC and CSTAOPT might be affected by different problem parameters. Particularly, we will look at solution quality, measured by the number of problem instances for which the IRU-Bound has not been attained. This is reasonable, as both procedures have solved all of the problem instances with at most one unit above the IRU-Bound.

The overall superiority of RSUC over CSTAOPT still holds for increasing problem sizes. This is an interesting result as looking at the test data for $m = 10,\ldots, 50$ only, one might conclude that the solution quality deteriorates with an increase in the problem size. However, even by additional computational tests with $m = 60$, 70, 80 this could not be verified (cf. Fig. 1).

Figure 2 presents the influence of the length factor on the solution quality for RSUC and CSTAOPT, respectively. It can be seen that especially problem instances with $b = 0.5$ turned out to be difficult for RSUC. The difficulties become evident particularly when the size of the problems increases.

For CSTAOPT Fig. 2 presents a completely different pattern. Those problem instances which turned out to be difficult for RSUC ($b = 0.5$) proved relatively easy for CSTAOPT, while easy problem instances ($b = 0.75$ and $b = 1.0$) for RSUC have been solved rather poorly by CSTAOPT. For the latter two groups a total number of 262 problem instances (13.1%) have been identified, for which the IRU-Bound was not attained by CSTAOPT. For the other two groups ($b = 0.25$ and $b = 0.5$), however, CSTAOPT failed to reach the IRU-Bound in only 29 cases.

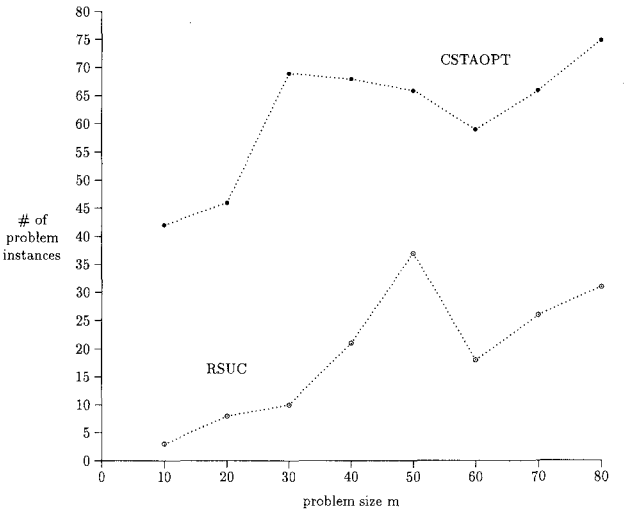

Fig. 1. Number of problem instances with objective function value above IRU-Bound – Variation of problem size

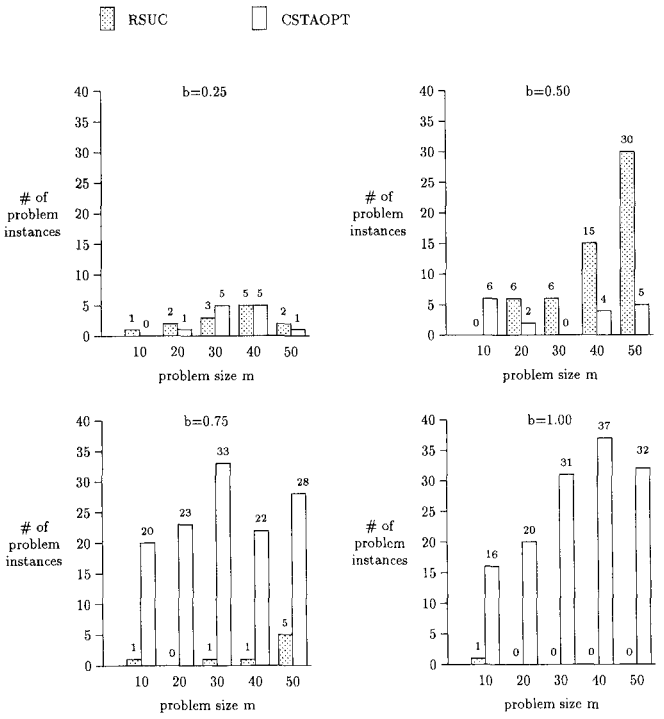

Fig. 2. Number of problem instances with objective function value above IRU-Bound – Variation of length factor

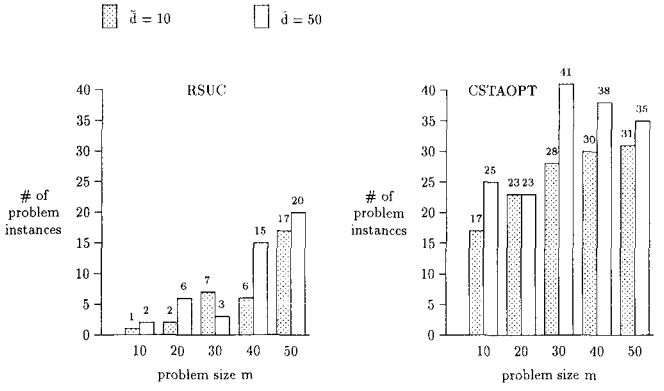

Fig. 3. Number of problem instances with objective function value above IRU-Bound – Variation of multiplicity factor

Figure 3 documents the influence of the multiplicity factor on solution quality. For both procedures RSUC and CSTAOPT there is just a slight tendency that an increasing average demand per order length will result in an increasing number of problem instances that will not be solved optimally (cf. Fig. 3).

This observation indicates that structure and size of the residual Bin-Packing Problems which have to be solved in the procedures RSUC and CSTAOPT are determined by other factors (i.e. problem size and relative length of orders) in the first place, while the influence of the multiplicity factor is relatively small.

Computing times for RSUC and CSTAOPT are hardly comparable because they mainly depend on the choice for the maximum number of backtrackings *maxb* to be performed by the Bin-Packing routine MTP. Residual problems arising from CSTAOPT are small for almost all 4,000 problem instances, and as a consequence the implemented value for *maxb* (25,000) is reached only in 13 cases. Setting *maxb* to a higher value therefore would increase computing times only because of these problems.

On the contrary, where RSUC failed to reach the IRU-Bound, for almost every problem instance (75 out of 79) the maximum number of backtrackings *maxb* (25,000) had to be performed. Additional computational tests have shown that almost all of the corresponding residual problems could not be solved exactly by MTP even when increasing *maxb* to multiples of 25,000. Computing times for RSUC therefore are much more sensitive to a proper choice for *maxb* than CSTAOPT. Altogether, looking at the computational results the implemented value for *maxb* (25,000) seems to be a good compromise for both heuristics.

8. Analysis and interpretation of results

Our results confirm that the ICSP – even though it is NP-complete – is a "well-natured" type of problem. Relatively simple heuristics provide exceptionally good and in most cases even optimal solutions. The excellent performance of the Residual Problem Approach and the Composite Approach are to a large extent due to the existence of a very good lower bound (cf. Chap. 7.1.2).

The procedures related to the Basic Patterns Approach mainly suffer from two defects. Because of the empirical validity of the IRU-Property rounding up the non-integer components of the optimal solution of the relaxed Complete-Cut Model may result in an integer solution far from the optimum, i.e. in an objective function value up to $m-1$ standard lengths above the IRU-Bound. This explains the poor performance of BRUSIM demonstrated in Tables 2 and 3. Less rigid rounding procedures such as those from BRURED, BRUSUC and BOPT perform significantly better, even though not satisfactory. This is due to the second defect of the Basic Patterns Approach, namely that no additional cutting pattern is allowed to enter the solution. As is known from general linear programming, proceeding from an optimal non-integer solution of a linear programming problem to an optimal integer solution will force additional structural columns (cutting patterns) into the basis. In the Basic Patterns Approach additional patterns are not considered, therefore the optimum is missed systematically.

Both pitfalls in the design of heuristics for the ICSP are avoided in the Residual Problem Approach. In another paper by one of the authors [14] it has been shown that if the IRU-Property holds for both the original and the residual problem, an optimal solution for the original ICSP is obtained by combining the down-rounded solution from column generation and the optimal solution of the residual problem. Our results in fact also verified the validity of the IRU-Property not only for the original but also for almost every residual problem. In other words: decomposing the ICSP according to the Residual Problem Approach results in an outstanding (empirical) worst-case perfor-

mance, which only slightly differs from that of an exact solution method (at least from a probabilistic point of view). Therefore, these heuristics may also be denoted as quasi-exact approaches [14]. Thus, theoretically, ROPT presents the most satisfying heuristic concept. Practically, it may find its limitations in the size of the residual problems, being of the Bin-Packing type and characterized by a relatively large number of items, which is strongly related to the number m of order lengths. In order to avoid unrealistic computing times, the exact algorithm for solving the residual problems had to be terminated artificially by limiting the number of backtrackings. This caused the solution quality of ROPT to deteriorate with an increase in the problem size, as can be seen from Table 2.

The fact that limitations of the Residual Problem Approach mainly arise from difficulties in solving the residual problems can also be verified for RSUC. Figure 2, for RSUC, exhibits the typical behaviour that can be observed for heuristic methods applied to the Bin-Packing Problem, i.e. that they work very well for problem instances which consist of relatively small items only ($b=0.25$) or in which the size of the items is distributed over the whole standard length ($b=0.75$ and $b=1.00$) while the solution quality is relatively poor for instances in which the size of the items does not exceed half the "bin size" resp. standard length ($b=0.5$). We would like to point out that a similar phenomenon can be observed for the column-generation algorithm as well. In terms of computing times problem instances of the ICSP characterized by $b=0.5$ also seem to be the most difficult ones (cf. Table 1).

If one accepts the hypothesis that solving the residual problems is the crucial aspect of the Residual Problem Approach, then the results presented in Fig. 3 for RSUC also fit very well into the picture. Different multiplying factors ($\bar{d}=10$, $\bar{d}=50$) will result in different magnitudes of the pattern frequencies for the non-integer solution of the continuous relaxation of the Complete-Cut Model. However, it seems unlikely that there will be a difference in size or structure of the residual problem after having down-rounded the frequencies. This is confirmed by Fig. 3, which does not present a significant difference in the solution quality for the two classes of problem instances.

From the Composite Approach we only discuss CSTAOPT here as it has proven superior to CSTAFFD. CSTAOPT incorporates a rounding procedure that satisfies a larger proportion of the original problem than does RSUC. Consequently, also the residual problems turned out to be significantly smaller. All but 13 of the 4,000 problem instances were even small enough to be solved optimally by MTP. Thus, the fact that CSTAOPT did not come up with an optimal solution can only be attributed to the specific way of how the original problem is decomposed. Although this specific way of decomposing results in easier manageable sizes of the residual problems, the worst-case characteristics are not as good as for the Residual Problem Approach by far. This is even true regardless whether the IRU-Property holds for both the original and the residual problem or not [14]. For example, CSTAOPT results in non-optimal solutions for 210 problem instances, although even a simple application of the FFD heuristic will provide optimal solutions for all these problems. On the other hand, RSUC has never been outperformed by FFD.

So far, our analysis has shown that RSUC and CSTAOPT represent two competing ways in designing heuristics for the ICSP. RSUC involves a "conservative", safe rounding procedure but sometimes results in residual problems which are difficult to solve optimally. By reducing the original problem size more rigidly, CSTAOPT, on the other hand, provides smaller residual problems, however at the expense of possibly missing the optimal solution of the original problem. The relative advantages and disadvantages of the two approaches are also reflected by Fig. 1. Generally, RSUC proves to be superior to CSTAOPT. However, with increasing problem size m the relative advantage of RSUC seems to diminish in the beginning ($m \leq 50$). This is due to the fact that with the RSUC approach an increasing number of residual problems cannot be solved optimally by MTP, while at the same time residual problems of CSTAOPT do not cause any difficulties for MTP. However, this is not true for larger problem instances ($m > 50$). Like for RSUC also for CSTAOPT now the residual problems will grow to sizes that cannot be matched by MTP any longer. The advantage of CSTAOPT over ROPT disappears. Therefore, for even larger problem instances ($m > 80$) that have not been considered here it cannot be expected that CSTAOPT becomes superior to RSUC.

The fact that the two procedures seem to be complementary (in the sense that specific problem classes that are difficult for one method may be solved better and more easily by the other) suggests to combine both heuristics. If RSUC and CSTAOPT were applied together, the number of problems for which the IRU-Bound is attained would come up to 3,984, i.e. at least 99.6% of all problem instances are solved optimally by at least one of these procedures.

9. Conclusions and outlook

The numerical experiments described in the preceeding chapters have shown that two heuristics for the ICSP are clearly superior to others: RSUC and CSTAOPT. Both methods managed to solve all of the 4,000 instances of the ICSP within one unit of the optimum objective function value. For 98.0% and 92.7%, respectively, of the problem instances it could even be shown that an optimal solution has been found. Average computing times were short, even for large problem instances. They amounted to 2.97 CPU-seconds per problem instance for RSUC and 2.37 CPU-seconds for CSTAOPT ($m=50$). In relation to column generation they absorbed 22.0% and 22.6%, respectively, of the total computing time.

For those problems which have proven to take an extraordinarily long time to solve, i.e. problems of type (50/50/$\bar{d}$), from an application-oriented point of view, there might be the need to further reduce the computing times. However, as column generation for these problem instances consumes the larger part of the total computing time (about 76.6% for RSUC and 90.3% for CSTAOPT) this should be the area for future research on code optimization.

Acknowledgement. The authors would like to express their thanks to two anonymous referees, whose detailed comments and suggestions have proven very helpful in revising the original version of the paper.

References

1. Bartmann D (1986) Entwicklung und Einsatz eines DV-Systems zur Auftragsverplanung bei einem Papierhersteller. Angewandte Informatik 12:524–532
2. Chvátal V (1980) Linear Programming. W.H. Freeman and Company, New York
3. Diegel A (1988) Cutting Paper in Richards Bay: Dynamic Local or Global Optimization in the Trim Problem. Orion 3:42–55
4. Dyckhoff H (1981) A New Linear Programming Approach to the Cutting Stock Problem. Oper Res 29:1092–1104
5. Dyckhoff H (1988) Production Theoretic Foundation of Cutting and Related Processes. In: Fandel G, Dyckhoff H, Reese J (eds) Essays on Production Theory and Planning. Springer, Berlin, pp 191–208
6. Dyckhoff H (1990) A Typology of Cutting and Packing Problems. Eur J Opnl Res 44:145–159
7. Dyckhoff H, Finke U (1992) Cutting and Packing in Production and Distribution – A Typology and Bibliography. Physica, Heidelberg
8. Dyckhoff H, Kruse H-J, Abel D, Gal T (1985) Trim Loss and Related Problems. Omega 13:59–72
9. Dyckhoff H, Wäscher G (1988) Bibliography of Cutting and Packing. Working Paper, RWTH Aachen, Germany
10. Eilon S (1960) Optimizing the Shearing of Steel Bars. J Mech Engg Sci 2:129–142
11. Fieldhouse M (1990) The Duality Gap in Trim Problems. SICUP-Bulletin, No. 5:4f
12. Garey MR, Johnson DS (1979) Computers and Intractability – A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco
13. Gau T (1993) Die Bestimmung (optimaler) ganzzahliger Lösungen eindimensionaler Zuschneideprobleme auf Grundlage eines Dekompositionsansatzes. Paper presented at the 1993 Joint DGOR/NSOR Conference, 25–27 August 1993, Amsterdam, Netherlands
14. Gau T (1994) Quasi-exakte und heuristische Verfahren zur ganzzahligen Lösung des Standardproblems eindimensionalen Zuschneidens. Working Paper, Institut für Wirtschaftswissenschaften, TU Braunschweig, Germany
15. Gau T, Wäscher G (1995) CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem. Eur J Opnl Res 84:572–579
16. Gilmore PC (1979) Cutting Stock, Linear Programming, Knapsacking, Dynamic Programming, and Integer Programming – Some Interconnections. Ann Discr Math 4:217–235
17. Gilmore PC, Gomory RE (1961) A Linear Programming Approach to the Cutting-Stock Problem. Oper Res 9:848–859
18. Gilmore PC, Gomory RE (1963) A Linear Programming Approach to the Cutting-Stock Problem – Part II. Oper Res 11: 863–888
19. Haessler RW (1975) Controlling Cutting Pattern Changes in One-Dimensional Trim Problems. Oper Res 23:483–493
20. Haessler RW (1976) Single-Machine Roll Trim Problems and Solution Procedures. TAPPI 59:145–149
21. Hochbaum DS, Shamir R (1991) Strongly Polynomial Algorithms for the High Multiplicity Scheduling Problem. Oper Res 39:648–653
22. Kantorovich LV (1960) Mathematical Methods for Organizing and Planning Production. Manag Sci 6:363–422
23. Marcotte O (1986) An Instance of the Cutting Stock Problem for Which the Rounding Property Does Not Hold. Oper Res Letts 4:239–243
24. Martello S, Toth P (1990) Knapsack Problems – Algorithms and Computer Implementations. John Wiley, Chichester
25. Neumann K, Morlock M (1993) Operations Research. Hanser, München-Wien
26. Pierce JF (1964) Some Large-Scale Production Scheduling Problems in the Paper Industry. Prentice-Hall, Englewood Cliffs
27. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical Recipes in FORTRAN – The Art of Scientific Computing. 2nd edn. Cambridge University Press, Cambridge
28. Rao MR (1976) On the Cutting Stock Problem. J Comp Soc India 7:35–39
29. Scheithauer G, Terno J (1992) About the Gap Between the Optimal Values of the Integer and Continuous Relaxation of the One-Dimensional Cutting Stock Problem. In: Gaul W et al. (eds) Operations Research Proceedings 1991. Springer, Berlin, pp 439–444
30. Stadtler H (1988) A Comparison of Two Optimization Procedures for 1- and $1^1/_2$-Dimensional Cutting Stock Problems. OR Spektrum 10:97–111
31. Stadtler H (1990) A One-dimensional Cutting Stock Problem in the Aluminium Industry and its Solution. Eur J Opnl Res 44:209–223
32. Sweeney PE, Paternoster ER (1992) Cutting and Packing Problems: A Categorized Application-Oriented Research Bibliography. J Opnl Res Soc 43:691–706
33. Wäscher G (1989) Zuschnittplanung – Probleme, Modelle, Lösungsverfahren. Unpublished Postdoctoral Thesis, University of Stuttgart, Germany
34. Wäscher G (1993) One-Dimensional Integer Cutting Stock Problems. Paper presented at the 1993 Joint DGOR/NSOR Conference, 25–27 August 1993, Amsterdam, Netherlands
35. Wäscher G, Carow P, Müller H (1985) Entwicklung eines flexiblen Verfahrens für Zuschneideprobleme in einem Kaltwalzwerk, Z Oper Res 29:B209–B230
36. Wäscher G, Gau T (1993) Test Data for the One-Dimensional Cutting Stock Problem. Working Paper, Institut für Wirtschaftswissenschaften, TU Braunschweig, Germany
37. Woolsey RED (1972) A Candle to Saint Jude or Four Real World Applications of Integer Programming. Interfaces 2:20–27