# A Node-Flow Model for 1D Stock Cutting: Robust Branch-Cut-and-Price

Gleb Belov
University of Dresden

Adam N. Letchford
Lancaster University

Eduardo Uchoa
Universidade Federal Fluminense

April 10, 2005

## 1 Introduction

*Branch-and-Cut-and-Price* (BCP) algorithms are branch-and-bound algorithms where both row generation (separation) and column generation (pricing) are performed. Following [12], we say that such an algorithm is *robust* when the separation and pricing subproblems are guaranteed to remain tractable during its execution. Robust BCP algorithms have been devised recently for a variety of problems, having been particularly successful for the *Capacitated Vehicle Routing Problem* (CVRP) [7] and the *Capacitated Minimum Spanning Tree Problem* (CMSTP) [6], for which many standard test instances have been solved for the first time.

In this paper we present a robust BCP algorithm for the *1-D Cutting Stock Problem* (CSP). In the CSP, which is strongly $\mathcal{NP}$-hard, one is given a set of item types numbered $\{1, \dots, n\}$, positive integer weights $l_1, \dots, l_n$ and order demands $b_1, \dots, b_n$ of each item type, and a bin capacity $C$. One wants to determine how to pack the items into the minimum possible number of bins. A special case of the CSP is the *Bin Packing Problem* (BPP) where the order demand of each item type is exactly one.

Two contributions make this robust BCP algorithm possible:

- A formulation suffers from *variable symmetry* when a single solution can be represented in many alternative ways by only permutating variable indices. It is not possible to construct effective BCP algorithms over such formulations, adding cuts or branching most probably only changes a fractional solution into another equivalent fractional solution. Belov and Scheithauer [2] have presented an effective BCP for the CSP over the asymmetric classic Gilmore-Gomory formulation. However, this BCP is non-robust. The pricing subproblem starts as an integer knapsack problem, with a pseudo-polynomial complexity bound of $O(nC)$. As cuts are added and branchings performed, the pricing is turned into a complex problem for which no pseudo-polynomial complexity guarantee exists.

The only class of formulations that allow pricing of pseudo-polynomial complexity after branching are *position-indexed* formulations [3], particularly the arc-flow formulation of V. de Carvalho. We present another formulation of this kind that avoids the symmetries found in many known CSP formulations other than Gilmore-Gomory [3]. It is shown that an arbitrary number of cuts or branchings over the variables from the new formulation can be used in a BCP algorithm without ever making the pricing harder than a capacitated shortest path problem having pseudo-polynomial complexity of $O(n^2C)$.

- We introduce a family of valid inequalities for the new formulation, which we call *capacity* cuts. A special kind of these cuts was already used by Vanderbeck [17]. The bound obtained by solving the continuous relaxation of the Gilmore-Golmory formulation, which we denote by $LB_{GG}$, was the best CSP bound known. It is computable in pseudo-polynomial time using the ellipsoid algorithm. Now, by solving this relaxation and adding the capacity cuts provided by heuristics, one gets a bound at least as good as $LB_{GG}$, also computable in pseudo-polynomial time. The addition of capacity cuts can be enough to close the duality gaps in some very hard instances, namely those without the *Integer Round-UP* (IRUP) property, as shown in our experiments.

Throughout the paper, we use the notation $V = \{1, \ldots, n\}$ and, for any $S \subseteq V$, $l(S) = \sum_{i \in S} l_i b_i$. Moreover, we call the quantity $l(V)/C$ the *trivial lower bound* and denote it by $LB_T$. The value obtained by rounding $LB_T$ up to the nearest integer will be called the *rounded trivial* lower bound and denoted by $LB_{RT}$. In the same way, the rounding up of $LB_{GG}$ is denoted as $LB_{RGG}$. Finally, $OPT$ denotes the number of bins used in the optimal solution.

In the next section we introduce the classic Gilmore-Gomory formulation and discuss some of its properties. Then we describe the new model and branching on its variables. Then we discuss the capacity cuts, the pricing problem, and computational results.

## 2 The Gilmore-Gomory formulation

A thorough review of formulations for the CSP has been given by de Carvalho [3]. The following formulation is usually attributed to Kantorovich. Let $U$ be an upper bound on the number of bins needed. For $i = 1, \ldots, n$ and $k = 1, \ldots, U$, define the integer variable $x_i^k$, which represents the number of times item $i$ is packed into the bin numbered $k$. Also, for $j = 1, \ldots, U$, define the binary variable $y_k$, which takes the value 1 if and only if bin $k$ is used. Then the problem is:

$$\begin{aligned}
&\text{Minimise} \quad \textstyle\sum_{k=1}^{U} y_k \\
&\text{Subject to:} \\
&\textstyle\sum_{k=1}^{U} x_i^k = b_i \qquad (i = 1, \ldots, n), && \text{(1a)}\\
&\textstyle\sum_{i=1}^{n} l_i x_i^k \le C y_k \qquad (k = 1, \ldots, U), && \text{(1b)}\\
&x_i^k \in \mathbb{Z}_+ \qquad (i = 1, \ldots, n; k = 1, \ldots, U), && \text{(1c)}\\
&y_k \in \{0, 1\}. \qquad (k = 1, \ldots, U). && \text{(1d)}
\end{aligned}$$

This formulation in itself is of little use for solving the CSP: its linear programming relaxation only yields the trivial lower bound $LB_T$ and it contains a lot of symmetry. But the Kantorovich formulation can be viewed as the starting point of a very successful way of attacking the CSP. By applying Dantzig-Wolfe decomposition [5], keeping constraints (1a) in the master and sending constraints (1b) to the subproblem, one obtains the classical Gilmore-Gomory formulation [8, 9]:

$$\begin{aligned}
&\text{Minimise} \quad \textstyle\sum_{k=1}^{q} \lambda_k \\
&\text{Subject to:} \\
&\textstyle\sum_{k=1}^{q} a_{ik} \lambda_k = b_i \qquad (i = 1, \ldots, n), && \text{(2a)}\\
&\lambda_k \in \mathbb{Z}_+ \qquad (k = 1, \ldots, q). && \text{(2b)}
\end{aligned}$$

In this formulation, a variable $\lambda_k$ is defined for every possible feasible packing of a single bin, called a *cutting pattern*. The feasible packings are indexed from 1 to $q$. The constant $a_{ik}$ represents the number of times item $i$ appears in packing $k$.

The exponential number of variables in this formulation makes column generation necessary. The pricing subproblem is a bounded integer knapsack problem. For most practical CSP instances, one can solve the linear programming relaxation of the formulation quickly, obtaining the bound $LB_{GG}$. This bound is remarkably tight. For the majority of CSP instances, $OPT - LB_{GG} < 1$, or, equivalently, $LB_{RGG} = OPT$. Indeed, for a long time it was conjectured that this was always the case, until counterexamples were found by Marcotte [11]. A modified conjecture, that $OPT - LB_{GG} < 2$ for all instances, was made by Scheithauer & Terno [16] and is still open. The worst instance known at the time of writing, due to Rietz [13], has $OPT - LB_{GG} = 6/5$. Concerning the worst-case bounds on the gap, Rietz & Scheithauer [14] proved that $OPT - LB_{GG} < \max\{2, (n+2)/4\}$ and Chan et al. [4] proved that $OPT/LB_{RGG} \leq 4/3$.

Instances for which $OPT - LB_{GG} < 1$ holds are known as *integer round-up* or IRUP instances. Many IRUP instances are fairly easy to solve, once $LB_{GG}$ has been computed, one only has to find an optimal packing by heuristic methods. When one has a non-IRUP instance, or an IRUP instance for which an optimal packing is not easily obtained by heuristics, one must do further work (e.g., branch or add cutting planes) in order to solve the problem.

It is well known that defining branches or cutting planes directly on the $\lambda$ variables destroys the knapsack structure of the subproblem, making pricing unpredictably harder. Moreover, although the model has no symmetry in terms of variables, it has some "internal" symmetry. Namely, there may be lots of similar patterns and thus lots of optimal solutions. In this case, branching on single patterns may be ineffective and heavily depend on their choice. Furthermore, the search tree becomes unbalanced. Therefore, many authors have developed branching rules and cutting planes based on different formulations, cf. [18, 3]. None of those alternative formulations is theoretically satisfactory, in the sense of completely avoiding symmetry and still guaranteeing a tractable pricing.

## 3 The node-flow formulation

We start with a formulation of the BPP, i.e. we assume the demands be equal to one. This formulation is best understood in terms of a certain directed graph, defined as follows. Introduce two dummy items 0 and $n+1$. The directed graph, denoted by $G = (V^+, A)$, has vertex set $V^+ = \{0, \dots, n+1\}$ and an arc $(i, j)$ for every $0 \le i < j \le n+1$. The weight of each item $i$ is interpreted as a weight on the vertex $i$. Then, each feasible packing of a single bin corresponds to a directed path in $G$ from 0 to $n+1$, passing through a set of vertices whose total weight does not exceed $C$. A feasible solution to the entire BPP then corresponds to a set of such paths, such that each vertex in $V$ lies in exactly one path.

This leads us to define a binary variable $x_{ij}$ for every $0 \le i < j \le n+1$, which takes the value 1 if and only if the corresponding arc is used. Note that, when $1 \le i < j \le n$, $x_{ij} = 1$ means that $i$ and $j$ are in the same bin, but no other item $k$, $i < k < j$, is also in that bin. The formulation is then:

$$\begin{aligned}
&\text{Minimise} && \textstyle\sum_{j \in V} x_{0j} && \\
&\text{Subject to:} && && \\
& && \textstyle\sum_{(i,j) \in \delta^-(j)} x_{ij} = 1 && (\forall j \in V), && \text{(3a)}\\
& && \textstyle\sum_{(i,j) \in \delta^+(i)} x_{ij} = 1 && (\forall i \in V), && \text{(3b)}\\
& && \textstyle\sum_{(i,j) \in \delta^-(S)} x_{ij} \ge l(S)/C && (\forall S \subseteq V), && \text{(3c)}\\
& && x_{ij} \in \{0, 1\} && (\forall (i,j) \in A). && \text{(3d)}
\end{aligned}$$

The above formulation is quite similar to the one usually employed on the Asymmetric Capacitated Vehicle Routing Problem. It is also an aggregated version of a formulation given by Ben Amor [1] for the BPP. The constraints (3a) and (3b) are called *in-degree* and *out-degree* equations, respectively. The constraints (3c) will be called *fractional capacity* inequalities. It should be noted that this formulation is completely asymmetric, in the sense that there is a one-to-one correspondence between feasible solutions to the BPP and feasible solutions to the integer program. We propose to call this model a *node-flow model* because both order demands are fulfilled and material is consumed by flow *into the nodes*. Compare this name to the arc-flow model of V. de Carvalho [3]: there, both processes take place in the arcs assigned to specific positions in the bin and to specific items.

We extend this formulation to the CSP, by allowing $x_{ij}$, $i < j$ to be integer and introducing additional integer variables $x_{ii}$, $i \in V$, to represent similar items occurring more than once in a bin. Then, we require constraints (3a) and (3b) to have right-hand side $b_i$, $i \in V$. Let us give an example. Consider an instance with $n = 2$, $l_1 = 3$, $l_2 = 4$, $b_1 = 5$, $b_2 = 3$ and $C = 8$. An optimal packing using four bins is: one bin with two items of type 2, one bin with one item of each type and two more bins, each one with two items of type 1. This solution is represented as $x_{01} = 3, x_{02} = 1, x_{11} = 2, x_{12} = 1, x_{13} = 2, x_{22} = 1$, and $x_{23} = 2$.

Although every packing has a unique representation in terms of $x$ variables, not every solution $x$ respecting (3a)-(3d) can be decomposed into paths corresponding to a feasible packing. Therefore, this is not yet a valid CSP formulation. This can be amended by combining it with the Gilmore-Gomory formulation. Consider this formulation on both $\lambda$ and $x$ variables:

$$\begin{aligned}
&\text{Minimise} && \textstyle\sum_{j \in V} x_{0j} && \\
&\text{Subject to:} && && \\
& && \textstyle\sum_{(i,j) \in \delta^-(j)} x_{ij} = b_i && (\forall j \in V), && \text{(4a)}\\
& && x_{ij} = \textstyle\sum_{k=1}^{q} p_{ij}^k \lambda_k && (i \in \{0, \dots, n\},\ j \in \{i, \dots, n\}), && \text{(4b)}\\
& && x_{ij} \in \mathbb{Z}_+ && (\forall (i,j) \in A), && \text{(4c)}\\
& && \lambda_k \in \mathbb{Z}_+ && (k = 1, \dots, q). && \text{(4d)}
\end{aligned}$$

where $p_{ij}^k$ is the value of the arc $(i, j)$ on the path representing packing $k$. Notice that we have not included the out-degree (3b) and fractional capacity (3c) constraints in this formulation. Indeed, it can be shown that any feasible solution to the LP relaxation of the above formulation satisfies those constraints.

The exponential number of $\lambda$ variables implies that pricing must be performed on those variables. The pricing subproblem corresponding to this formulation is a capacitated shortest-path problem over the acyclic graph $G$. Although this is slightly harder than the bounded knapsack problem, it can still be solved in pseudo-polynomial time by dynamic programming, $O(n^2C)$. Any number of cuts over $x$ variables, including branch cuts $x_{ij} \leq \lfloor \alpha \rfloor$ or $x_{ij} \geq \lceil \alpha \rceil$, can be added without ever changing the structure of this subproblem. Therefore, the pricing is guaranteed to remain tractable and the resulting BCP algorithm is robust. Moreover, if only a few cuts on $x$ are added, we can reduce the underlying graph so that the practical complexity of the shortest path solver remains close to that of the knapsack solver, namely $O(nC)$, see below.

A more compact formulation is obtained by eliminating the $x$ variables via the identities (4b). If only constraints (4a) are present, the result is exactly the traditional Gilmore-Gomory formulation. However, an additional cut of the type $\sum_{(i,j)} \pi_{ij} x_{ij} \gtreqqless r$ is transformed into $\sum_{k=1}^{q}(\sum_{(i,j)} \pi_{ij} p_{ij}^{k})\lambda_k \gtreqqless r$.

## 4 Capacity cuts

In order to obtain bounds better than $LB_{GG}$ without branching, we devise effective cuts on $x$ variables.

**Definition 1** *A* capacity *inequality is any inequality of the form:*

$$\sum_{(i,j)\in\delta^-(S)} x_{ij} \geq r(S),$$

*where $r(S)$ is any lower bound on the number of bins needed to pack the items in $S \subseteq V$.*

A similar family of cuts is known to be effective on the asymmetric CVRP. Vanderbeck [17] already used such cuts, however only with $|S| = 1$, and observed that they slightly decreased the size of the branching tree. To keep the computation of the right hand side $r(S)$ tractable, we suggest the following possible options:

- *rounded* capacity (RC) inequalities, in which $r(S) = \lceil l(S)/C \rceil$;
- *rounded Gilmore-Gomory* capacity (RGGC) inequalities, in which $r(S)$ is equal to the Gilmore-Gomory lower bound for the CSP defined only on the items in $S$, rounded up (we denote this lower bound by $LB_{RGG}(S)$).

Let us give an example of how such inequalities can be effective on the CSP. Probably the smallest known non-IRUP instance, due to M. Fieldhouse (cf. [13]), is the following: $n = 3$, $l_1 = 15$, $l_2 = 10$, $l_3 = 6$, $b_1 = 3$, $b_2 = 5$, $b_3 = 9$ and $C = 30$. For this instance, $LB_{GG} = 4.966$. By adding the RC inequality $\sum_{(i,j)\in\delta^-(\{2,3\})} x_{ij} \geq 4$ and solving the pricing again, one gets an improved bound of 5.055, enough to prove the optimality of a feasible solution of value 6. There are two other violated RC inequalities in the first fractional solution of the Fieldhouse instance. Using any one of those cuts would also close the duality gap.

The following theorem, which has no counterpart in the case of the asymmetric CVRP, is of both theoretical and practical interest:

**Theorem 1** *Let $S \subseteq V$ be a set of items and let $i$ be the index of the lowest indexed item in $S$. Then, if $r(S) = r(S \setminus \{i\})$, the capacity inequality for $S$ is dominated by the capacity inequality for $S \setminus \{i\}$ (no matter what method is used to compute $r(S)$). The same holds if $i$ is the index of the highest indexed item in $S$.*[1]

**Proof** When $i$ is the lowest index, replacing $S$ with $S \setminus \{i\}$ causes the left-hand side of the capacity inequality to decrease by $1 - \sum_{j \in S \setminus \{i\}} x_{ij} \geq 0$. If $i$ is the highest index, the corresponding decrease is $\sum_{j \in V \setminus S} x_{ji} > 0$. $\square$

This result can potentially be exploited algorithmically. Several heuristics for the separation of RC inequalities are known in the vehicle routing literature. The separation problem for RGGC inequalities appears to be more difficult. Of course, we could just add the RGGC inequality for $V$ itself to the LP, but this will not enable us to solve non-IRUP instances. However, we know that the RGGC inequality for $V$ will be dominated by the RGGC inequalities for sets $S' \subset V$ satisfying the conditions of Theorem 1. The addition of the RGGC inequality for any such $S'$ to the master will cause the linear programming lower bound to increase to *at least* $LB_{RGG}$, and possibly to even more.

Once the initial column generation phase has been completed, one simple heuristic for finding such sets $S'$ is to find sets of consecutive in-degree constraints (4a) whose dual prices sum to at most $LB_{RGG} - LB_{GG}$. It follows from simple duality theory that removing the associated items from $V$ will yield suitable sets $S'$. For stronger cuts, i.e., even smaller subsets of $S'$, it might be worthwhile re-optimizing over $S'$ by column generation, yielding a new dual solution, to see if any more in-degree equations can be dropped.

[1]This seems to imply that capacity inequalities with $r(S) = 1$ are redundant.

## 5 Column Generation

In this section we discuss the pricing problem in the Gilmore-Gomory formulation of 1D-CSP arising when we add branching constraints and/or cutting planes expressed in the CVRP variables. A straightforward way to solve the pricing problem is to compute the dual values on the arcs of the underlying graph $G$ and solve the capacitated shortest path problem. However, depending on the branching strategy, we may mainly choose nodes at small depth where only a few arcs are constrained. Then, the pricing problem is very close to the bounded knapsack problem. The question arises how to represent this simple shortest path problem by a graph with a small number of arcs.

In the root, pricing is the bounded knapsack problem:

$$\max\Big\{\sum_i d_i a_i : \sum_i l_i a_i \le C,\ a_i \le b_i,\ a_i \in \mathbb{Z}_+\ \forall i\Big\}, \tag{5}$$

where $d_i$ are simplex multipliers of the degree constraints (2a). It can be solved by dynamic programming in $O(nC)$ time [10]. With constraints on the CVRP variables, pricing becomes a capacitated shortest path problem [19]:

$$\begin{aligned}\max\Big\{\sum_{i,j} d_{ij}p_{ij} : \sum_{i,j} l_j p_{ij} \le C,\ \sum_i p_{ij} \le b_j\,\forall j,\ \sum_{i<j} p_{ij} = \sum_{k>j} p_{jk}\,\forall j, \\ p_{ij} \in \{0,1\}\,\forall i<j,\ p_{ii} \in \{0,\dots,b_i-1\}\,\forall i\Big\}\end{aligned} \tag{6}$$

and can be solved in $O(n^2C)$ time.

Ziegelmann [19] gives a way to view the binary knapsack problem as a capacitated shortest path problem using the following graph $G'$: for each node $i \in \{0,\dots,n-1\}$, introduce two arcs from $i$ to $i+1$, one arc with profit $d_{i+1}$ and capacity $l_{i+1}$, and the other arc with both profit and capacity zero. Then, the longest path from 0 to $n+1$ is the solution of the knapsack problem.

Now suppose that a constraint on some CVRP variable $x_{i_1j_1}$ is added. Then in (6) we have $d_{i_1j_1} \ne d_{ij_1}$ for $i \ne i_1$. To account for this in (5), let us modify the graph $G'$ as follows: for each $i \in \{1,\dots,n\}$, distinguish the state $\boxed{\bar{i}}$ when item $i$ is not present in the solution and the state $\boxed{i}$ when it is present at least once. This gives the graph $G''$ shown in Fig. 1.

All we need to do now is modify the set of arcs so that all feasible solutions are represented correctly even when several constraints are added.

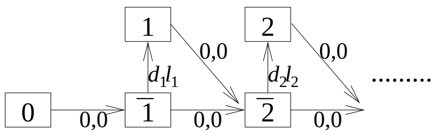

Figure 1: Graph $G''$

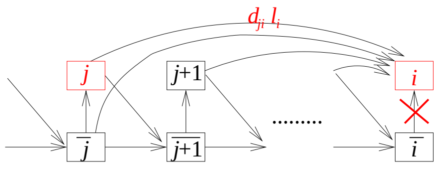

Figure 2: Graph $G''$ with new arcs

Fig. 2 shows an idea of how to do this. For simplicity, we omitted the loop arcs $(i, i)$ in the drawings.

On this graph, the complexity of pricing is $O(\kappa nC)$ where $\kappa$ is the number of nodes involved in modified arcs.

## 6 Computational experiments

We tested the capacity cuts and the branching procedure. Moreover, we discuss the issue of pseudo-polynomial complexity of column generation. All the test instances are available on *http://www.math.tu-dresden.de/˜capad/*. In addition, the hard28 set is available on the ESICUP page *http://www.apdio.pt/sicup/*.

The branch-cut-and-price framework was implemented in GNU C++ 3.3 and run on an AMD Opteron 2.2 GHz with 1 GB of memory. The LP solver was CPLEX 9.0.3.

### 6.1 Capacity Cuts

We implemented the exact separation procedure from [7]. To find a set $S$ for which the rounded capacity cut is violated by a given solution $\overline{x}_{ij}$ ($(i, j) \in A$), we solved the following mixed-integer program:

$$\text{Minimise} \quad \textstyle\sum_{i,j} \overline{x}_{ij} w_{ij}$$

Subject to:

$$w_{ij} \geq y_j - y_i \qquad (\forall (i, j) \in A), \tag{7a}$$

$$\textstyle\sum_i l_i y_i \geq MC + 1, \tag{7b}$$

$$w_{ij} \geq 0 \qquad (\forall (i, j) \in A), \tag{7c}$$

$$y_i \in \{0, 1\} \qquad (i = 1, \ldots, n), \tag{7d}$$

$$y_0 = 0 \tag{7e}$$

for all $M$ from 2 up to (current upper bound)$-1$ and stopped if the optimal objective value was smaller than $M+1-10^{-3}$. For the first three non-IRUP 1D-BPP instances from the collection "53NIRUP" with up to 40 items, we needed only 1 or 2 iterations to close the gap by inequalities with $M = 2$ or 3. It should be noted that these three instances have an integer root LP value. For other instances, where the root LP value is below integer, we found some violated cuts and sometimes raised the LP value up to the integer but not higher. Note that the MIP (7a)–(7e) is rather easy to solve for $M = 2$ and $M =$ (upper bound) $- 1$. For larger instances such as hard28 (see below), not many violated cuts with these values of $M$ were found and other values of $M$ made the MIP difficult. Thus, separation heuristics are needed as well as investigations concerning other types of cuts.

## 6.2 Branching on CVRP variables

We considered the 'hard28' set of CSP instances [15, 2] with $n \in \{140, 160, 180, 200\}$. Among those 28 instances, 5 are non-IRUP, so that current branch-and-price algorithms must perform branching in order to close the duality gap. The remaining instances are IRUP, but very hard for heuristics, so current algorithms may also have to branch repeatedly until an optimal packing is found.

We have run the proposed robust BCP algorithm using only the cuts resulting from branchings on the bounds of $x$ variables. We have not yet implemented separation heuristics for capacity cuts. A comparison with a non-robust BCP branching on $\lambda$ variables, fully described in [2], is given. Another method to compare with is the branching scheme on the variables of the arc-flow model [3]. The only change we made is that we considered only *proper* patterns $a$, i.e. with $a \leq b$. All three algorithms were executed 8 times on the whole test set, with four different starting bases and other varying parameters. For each method, we carefully selected favourable parameters, because the G&G and AFF branchings can easily need more than two hours for the 18th instance bpp716 (which is non-IRUP). In fact, both these methods heavily depend on the choice of "$=$" or "$\geq$" in the degree constraints. Furthermore, this experiment has shown that random branching variable choice is not worse than the "most infeasible" rule.

Table 1 shows the results. From the right half of the table we see that instance bpp716 is the reason why the average time for the G&G method is

Table 1: Comparison of the three branching rules on the hard28 set: average time, standard deviation, nodes, and log. average time

| | hard28 all | | | | hard28 - no bpp716 | | | |
|---|---|---|---|---|---|---|---|---|
| | $t_{\text{ave}}$ | $\Delta_t$ | nod | $t_{\text{log.ave}}$ | $t_{\text{ave}}$ | $\Delta_t$ | nod | $t_{\text{log.ave}}$ |
| G&G | 17.6 | 121.2 | 2689 | 2.3 | 2.7 | 4.1 | 212 | 2.0 |
| NFF | 10.0 | 11.9 | 163 | 6.3 | 10.3 | 11.9 | 169 | 6.7 |
| AFF | 8.3 | 11.2 | 167 | 5.1 | 8.2 | 10.9 | 168 | 5.2 |

higher. NFF branching has the smallest relative standard deviation and no instances with extremely high times.

### 6.3 Pseudo-Polynomiality of Column Generation

When branching on the variables $\lambda$ of the Gilmore-Gomory model, the only possible column generation method is branch and bound. This method performs exponentially in the worst case, but for the hard28 set it has good results. It performs very poorly on instances with very small items [2].

The capacitated shortest path solver has a pseudo-polynomial complexity, i.e. the time depends on capacity $C$. This has the following effect: for hard28, where $C = 1000$, the average time for column generation in the root is $< 1$ sec. For similar instances with $C = 10000$ it is 10 seconds (about 1000 columns are generated in both cases.)

## 7 Conclusions

A node-flow formulation for one-dimensional stock cutting was proposed. This formulation has no symmetries. Capacity cuts, based on the variables of this formulation, can raise the LP bound in the difficult non-IRUP instances enough to prove optimality of a good heuristic solution. Branching on the variables of the NFF formulation, as well as those of the arc-flow formulation, is robust. Both were compared to branching on variables of the Gilmore-Gomory formulation. NFF and AFF are faster than G&G only for instances with small number of items $n$ and/or small capacity $C$. This is explained by the pseudo-polynomial column generation complexity. The method shows high stability of running times with various algorithmic parameters.

## References

[1] H. Ben Amor (1997) Résolution du Problème de Decoupé par Génération de Colonnes. Master's thesis, École Polytechnique de Montréal, Canada.

[2] G. Belov & G. Scheithauer (2003) A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. To appear in *Eur. J. Opl Res.*

[3] V. de Carvalho (2002) LP models for bin packing and cutting stock problems. *Eur. J. Opl Res.*, 141, 253–273.

[4] L. M. A. Chan, D. Simchi-Levi & J. Bramel (1998) Worst-case analyses, linear programming and the bin-packing problem. *Math. Program.*, 83, 213–227.

[5] G. Dantzig & P. Wolfe (1960) Decomposition principle for linear programs. *Oper. Res.*, 8, 101–111.

[6] R. Fukasawa, M. Poggi de Aragão, O. Porto & E. Uchoa (2003) Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem. In Proc. of the International Network Optimization Conference, Evry, France, 231–236.

[7] R. Fukasawa, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa & R. Werneck (2004) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. In *Integer Programming and Combinatorial Optimization 10. LNCS*, 3064, 1–15.

[8] P.C. Gilmore & R.E. Gomory (1961) A linear programming approach to the cutting stock problem, Part I. *Oper. Res.*, 9, 849–859.

[9] P.C. Gilmore & R.E. Gomory (1963) A linear programming approach to the cutting stock problem, Part II. *Oper. Res.*, 11, 863–888.

[10] H. Kellerer, U. Pferschy, & D. Pisinger (2004) *Knapsack Problems.* Springer.

[11] O. Marcotte (1986) An instance of the cutting stock problem for which the rounding property does not hold. *Oper. Res. Lett.*, 4, 239–243.

[12] M. Poggi de Aragão & E. Uchoa (2003), Integer program reformulation for robust branch-and-cut-and-price. In L.Wolsey (ed.) *Annals of Mathematical Programming in Rio*, pp. 56–61.

[13] J. Rietz (2003) *Investigations of MIRUP for vector packing problems.* Ph.D. thesis, Freiberg University (in German).

[14] J. Rietz & G. Scheithauer (2002) Tighter bounds for the gap and non-IRUP constructions in the one-dimensional cutting stock problem. *Optimization*, 51, 927–963.

[15] J. E. Schoenfield (2002) Fast, exact solution of open bin packing problems without linear programming. Draft, US Army Space & Missile Defense Command, Huntsville, Alabama, USA.

[16] G. Scheithauer & J. Terno (1995) The modified integer round-up property of the one-dimensional cutting stock problem. *Eur. J. Opl Res.*, 84, 562–571.

[17] F. Vanderbeck (1999) Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Program.*, 86, 565–594.

[18] F. Vanderbeck (2000) On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.*, 48, 111-128.

[19] M. Ziegelmann (2001) *Constrained Shortest Paths and Related Problems.* Ph.D. thesis, University of Saarland.