# NEAR OPTIMAL OBJECTS PACKING THROUGH DIMENSIONAL UNFOLDING

**Emilio Bertolotti, Enrico Castaldo, Gino Giannone**

BULL HN, Italy
Via del Parlamento, 33 Borgolombardo, Mi, Italy 20098
Phone: + 39.2.6779.2054, Fax: + 39.2.6779.2515
EMail: frossill@eznet.it

**Abstract**

The efficient packing of regular shaped two dimensional objects is the core problem of several kinds of industry such as steel, thin-film and paper. This special kind of packing problem consists in cutting small rectangular stripes of different length and width from bigger coiled rectangles of raw material by combining them in such a way that trim loss is as low as possible. Previous attempts to apply "exact" discrete optimization techniques such as Simplex Partial Columns Generation, were not able to produce good cutting plans in large instances. We tackled this Roll Cutting Problem developing a new "dimension decomposition technique" that has been successfully experimented in a big steel industry first and then replicated in other steel and thin-film factories. This "unfolding" technique consists in splitting the overall search for good cutting plans in two separate graph search algorithms, one for each physical dimension of the rectangles. Searching individually in each dimension improves the overall search strategy allowing the effective pruning of useless combinations. Experimental evidence of how dimensional splitting strengthens the overall search strategy is given.

## 1. Introduction

The efficient packing of two dimensional objects affects the core business of several kinds of industry such as steel, thin film, paper and others. Normally, they have to cut small stripes of different length and width from bigger coiled rectangles of raw material by combining them in such a way that trim loss is as low as possible. This activity is crucial to their productivity, and thus highly relevant to their business. Dedicated experienced people normally apply naive methods to efficiently build sequences of cutting schema able to cover the overall set of workable commercial demands. Unfortunately, this packing problem is known to be hugely combinatorial even with tens of prototype rectangles of different sizes and the "stock cutting" industry suffers from its intractability. The "exact" computer discrete optimization techniques applied so far, such as Simplex Partial Columns Generation, were not able to deterministically produce good object packings in a reasonable response time. We tackled this Roll Cutting Problem devising a "dimensional decomposition" technique, we called 1D+1D heuristics, that has been successfully experimented in a big steel industry first (AST-ILVA) and then replicated in other factories. The technique is based on the co-operation of two separate graph search algorithms each one dedicated to a single dimension of the regularly shaped planar objects. Given $m$ possible rectangular objects to cut from $n$ available coils, then the "unfolding" of the two dimensions has the positive consequence to reduce the average time complexity of the resulting solution approach from an expected

$$\textit{Response-Time} = \textit{CONST} * (2^{n+m})$$

to a more attractive

$$\textit{Response-Time} = \textit{CONST} * (2^{n} + 2^{m})$$

where *CONST* is a common constant factor related to the speed of the machine. As a result, even if the complexity of the devised dimensional unfolding approach remains exponential, it is able to provide near optimal results in all real world cases faced so far. With the term near optimal we mean that ***optimal*** cutting plans can be ***exhaustively found in more that 95%*** of the cases within a reasonable response time (less than 120 seconds on a Pentium). This result is obtained through the dimensional unfolding which permits to avoid useless search. We practically demonstrated that the same basic 1D+1D technique applies to other kinds of cutting stock industry and is even suitable for parallel implementations whenever scale up factors require it. In fact, we applied 1D+1D to thin film industry (MANULI SpA) and also extended a critical pilot customer installation (MARCORA SpA) to exploit a parallel implementation. All these systems have been written in Common Lisp and now form a unique specialized platform named ROLL-CUTTER. Within the rest of the paper we will follow the steel industry study case to introduce the problem and the proposed solution. In section two, a precise description of the Roll Cutting Problem (RCP) together with the previously attempted Integer Linear Programming approaches are presented in some details. Section three introduces the 1D+1D decomposition and its strength in comparison with more classical 2D Implicit Enumeration Techniques. Feedbacks concerning the applications released so far are presented in section four. Section five discusses some of the implementation issues while section six gives an idea of how the ROLL-CUTTER platform is now going to be maintained and extended.

## 2. The Roll Cutting Problem

The overall objective of the Roll Cutting Problem (RCP) is to ensure the production of a sequence of *feasible* and *near optimal* cutting plans given a set of available coils and a set of workable commercial orders. To explain what this means, we will neglect any unnecessary detail concerning the chemical requirements and instead we will concentrate on the physical dimensions of the objects which constitute the core of the debated combinatorial problem. Figure 2.1 shows the basic dimensional parameters which completely describe all the involved entities: coils (*big coiled rectangles*) and commercial orders (*small rectangles or stripes*).

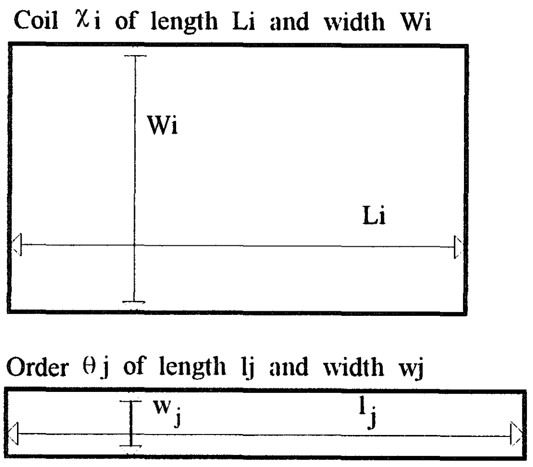
Coil χi of length Li and width Wi
Wi
Li
Order θj of length lj and width wj
wj
lj

Figure 2.1 Geometry of Coils & Stripes

Generally speaking, we have to manage two linear dimensions : Length and Width. However, to express the first linear dimension, we will use the weight $C_i$ instead of length $L_i$ since it renders the overall modelling simpler to formalize and closer to the way steel engineers describe the problem. Given the constant v expressing the unitary weight of a given raw material, then length and weight are tied by the simple relation

$$C_i = L_i W_i \; v$$

The dimensional requirements of the rectangles, either coils $\chi_i$ and orders $\theta_j$, will be thus completely defined through their weight $C_i$ ($c_j$) and their width $W_i$ ($w_j$). Conventionally, weight and width will be considered respectively as the first and the second dimension. Moreover, without loss of generality, we will always assume that all the coils have the same width $W$ :

$$W_i = W \qquad i = 1,2,3,\ldots,n$$

In this way, each coil $\chi_i$ will be completely characterized just by its weight $C_i$.

Before describing what a *feasible* cutting plan is we have to explain first how orders must be cut out from coils, given the operational constraints of the cutting machines (slitters). Figure 2.2 shows an introductory example where three orders ($\theta_1$, $\theta_2$, $\theta_3$) are cut out from two coils ($\chi_1$, $\chi_2$) of different weight.

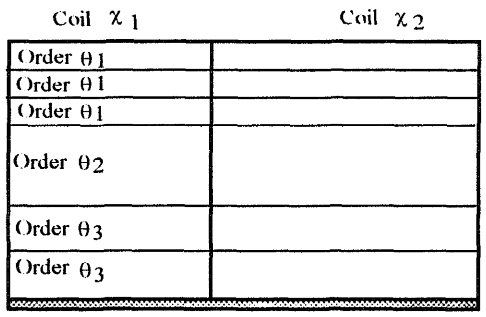

Figure 2.2 An introductory Example

Notice how coils can be bonded to reach the needed total weight while orders can be split in an arbitrary number of identical stripes to fulfil the available area *W*. This operational mode. is not limited to the steel industry. On the contrary, these "cutting rules" are quite common to all kinds of industries which must cut out planar objects of regular shape. In fact. these rules are tied to the cutting technology which is always the same no matter what the nature of the specific material to be cut is (steel. thin-film. paper, glass). Even when small sheets must be eventually obtained. this stripes cutting step is an unavoidable intermediate process Actually, there are other constraints. like the maximum weight allowed of each stripe. that discipline the feasibility of a cutting plan. However. even if these constraints are properly handled within our applications. we will not discuss them in the paper since they would inappropriately complicate the overall explanation.

Within each cutting plan the particular stripes configuration adopted is uniquely identified by the so called cutting pattern. A given cutting pattern determines the number of stripes for each order. The cutting pattern adopted in the example of figure 2.2 can be represented by the vector

$$\mathbf{Y} \equiv (y_1, y_2, y_3) \equiv (3, 1, 2)$$

By analogy, it is worth defining a coils pattern associated to a cutting plan. A coil pattern is represented through a binary vector **X** having one component for each available coil. A component's value is equal to 1 if the corresponding coil is used within the associated plan or 0 elsewhere. The coils pattern of the example is represented by the vector

$$\mathbf{X} \equiv (x_1, x_2) \equiv (1, 1)$$

Normally, a plan involves just a subset of the total available *n* coils and a subset of the total workable *m* orders. This means that some of the $x_i$ and $y_j$ will be equal to zero.

Tables 2.1, 2.2. show the weight and width values of the coils and orders used within the introductory example of figure 2.2

| Coil | Width (inches) | Weight (pounds) | Pound per inch |
|---|---|---|---|
| $\chi_1$ | 50 | 3500 | 70.0 |
| $\chi_2$ | 50 | 6000 | 120.0 |

Tab. 2.1 Coil Values of the Introductory Example

| Order | Demanded Width (inches) | Demanded Weight (pounds) | Resulting Weight (pounds) |
|---|---|---|---|
| $\theta_1$ | 5 | 2800 ±2% | 2850 |
| $\theta_2$ | 16 | 3000 ±2% | 3040 |
| $\theta_3$ | 8 | 3000 ±2% | 3040 |

Tab. 2.2 Order Values of the Introductory Example

In this simple cutting plan the trim loss is three inches *(50 - [5 * 3] + 16 + [8 * 2] = 3)*. This means 570 pounds of wasted material and the plan would probably be considered not so good. Nevertheless. since human planners are normally dealing with tens of orders and coils, the huge number of combinations would have probably rendered the search for better plans a strenuous task.

## 2.1 Formal description of RCP

We are now in the position to precisely define what a *feasible* and *good* cutting plan is. Given *n* available coils and *m* workable orders then a feasible cutting plan P is defined by the couple

$$\mathrm{P} \equiv \{\mathbf{X}, \mathbf{Y}\}$$

where $\mathbf{X} \equiv (x_1, x_2, \ldots x_n)$ and $\mathbf{Y} \equiv (y_1, y_2, \ldots, y_m)$ are respectively a coils pattern and a cutting pattern such that the following *m + 1* constraints hold

$$\frac{\sum_i^n x_i C_i}{W} y_j w_j \le c_j + \delta_j \quad (j = 1 \ldots m) \qquad (2.1)$$

$$\sum_j^m y_j w_j \le W \qquad (2.2)$$

2.1 ensures that for each order $j = 1, 2, 3, \ldots, m$ the planned weight does not violate the max. demand (*$d_j$ is the max. allowed exceeding weight for the j-th order*). 2.2 guarantees that the total width $W$ is never exceeded.

To plan the overall production demand, which comprehends $m$ different commercial orders, what we should really build is a set {P} of feasible plans able to cover exactly the entire workload. This means we should find a set of plans {P} observing the following $m$ constraints derived from (2.1) for $j = 1,2, \ldots, m$.

$$c_j \le \frac{\sum_i^n x_i C_i}{W} y_j w_j \le c_j + \delta_j \qquad (2.3)$$

However, the problem of finding the overall set of covering plans is not of much interest to the scope of the paper for a couple of reasons. First of all, it must be necessarily solved in different ways according to the particular features of each specific case (e.g. *we were asked to solve it applying an order priority rule for a steel factory*). Moreover, there is a practical reason that makes this sort of global planning quite superfluous. In fact, only the next few *good* plans are really needed since the new incoming commercial orders and the new coiled material alter the possible combinations rendering earlier produced plans soon obsolete.

Thus, we will now continue to concentrate on the solution of a single *feasible* and *good* plan which is the real core aspect of RCP. In this framework the search for "the next" *feasible* and *good* plan will start from a chosen pivot order $O_j$. This pivot is the order that the human planner wants to be certain is inserted into the next plan. As already mentioned, the choice of the pivot order could be automatically done according to a priority rule (e.g. due-dates) or through a more exhaustive strategy which tries in turn all the orders as candidate pivots. Nevertheless, all these different ways to choose the next pivot can affect the overall computational complexity at most through a linear factor (*number of orders m*). The real source of *non-polynomial* complexity ($2^{n+m}$) remains the search for a single *feasible* and *good* plan once an arbitrary pivot order has been chosen.

Given a pivot order $O_j$ several *feasible* cutting plans exist. These feasible plans are formed by all the couples {**X**,**Y**} observing the two equations (2.1) and (2.2). Before defining when a *feasible* plan P ≡ {**X**,**Y**} is judged to be a *good* plan we have to precisely define the optimization objectives used as selection criteria. These optimization objectives are the following

- Trim loss minimization :
  Min (*TL* (**Y**)).
- Maximize the usage of a given cutting pattern which is equivalent to minimize the number of knives changes :
  Max (*KU* (**X**)).

They can be formally expressed through the following relations

$$Min(TL(Y)) \equiv Min(W - \sum_j^m y_j w_j) \qquad (2.4)$$

$$Max(KU(X)) \equiv Max(\frac{\sum_i^n x_i C_i}{W}) \qquad (2.5)$$

The first objective (2.4) clearly expresses the minimization of trim loss. The second objective (2.5) is strongly tied to the minimization of the number of knife changes since maximizing the usage of the same pattern (KU(**X**)). In fact, according to the previous assumption, we are working on the search for "the next" single plan separately, so that pursuing the maximization of a plan weight, i.e. plan duration, implicitly minimize the frequency of knife changes.

We are now in the position to define when a *feasible* plan becomes a *good* plan. A candidate *feasible* plan P ≡ {**X**,**Y**} is a *good* plan if another *feasible* plan P' ≡ {**X**',**Y**'} does not dominate P through the following dominance relationship

*P' dominates P if and only if the following two conditions are jointly verified*

$$TL(\mathbf{X}') < TL(\mathbf{X}) \qquad \& \qquad (2.6)$$

$$KU(Y') > KU(Y)$$

As normally happens with the use of dominant relationships, we have just identified a set of *good* plans instead of giving a definition of *the unique absolutely optimal* plan. In fact, the presence of multiple contrasting objectives would have rendered quite unnatural the definition of "plan optimality". In other words, between a plan with lower Trim Loss and a plan with lower Knife Changes, we cannot say which one is the best. Merging the two different objectives through a linear function would be quite an arbitrary decision. Anyway, the dominance relationship is known to be a powerful tool able to drastically prune outclassed solutions (Pearl 1985). Experimental results with real world data revealed how the dominant set formed by the resulting *good* plans has no more than five or six members. There are other external reasons, like overloading or underloading conditions affecting the cutting machines workload, that can help in making a more convenient selection amongst the *good* plans.

## 2.2 Revised Simplex approaches

Generally speaking, the RCP problem is within the family of two-dimensional regular shaped cutting stock problems which are known to be NP-hard (Dyckoff et al. 1984). This kind of problem has already been tackled through ILP partial columns generation approaches. This Revised Simplex method consists in generating only the columns suggested by the proper auxiliary knapsack problem solved at each step of the Simplex iteration (Gilmore & Gomory 1966). While the method works fine for 1D cutting stock problems, there are practical limitations that makes it difficult to apply to 2D real-world instances (Dyckoff et al. 1984). In fact, the ordinarily number of coils and orders to deal with are beyond the quantities normally handled though partial column generation. Moreover, the presence of multiple objectives and very specific constraints render the problem difficult to model through ILP approaches.

A flavour concerning the size of the potential search space is given considering the following simple formula. Being $m$ the average number of available coils, $n$ the average number of candidate orders and being $\bar{k}$ the average number of stripes per order, the potential number of possible cutting plans is proportional to the power-set of the coils times the power-set of the possible stripes, i.e.:

$$2^{(n+\bar{k}m)} \qquad (2.8)$$

Typical average values for $n$, $m$ and $\bar{k}$ are respectively thirty, thirty and three.

## 3. 1D + 1D Decomposition

As already recognized (Dyckoff et al. 1984), the additional specific constraints and the particular structure found in this sort of problem may constitute a source of insuperable complexity for a general ILP modelization. By contrast, within a more tailored approach, these problem features can be exploited to render the modelization more tractable (Scutella' et al. 1995). Not surprising, a quite simple heuristic approach, based on a specialized greedy procedure, already demonstrated to produce fruitful results in a specific industrial roll cutting application (Ferreira 1990).

Pursuing the heuristic mainstream, we tackled the problem exploiting its structure to simplify the solution approach while avoiding degradation of the solution quality. More precisely, we have decomposed the overall two-dimensional cutting stock problem into two one-dimensional knapsack sub-problems we called : ***first-dimension-knapsack*** and ***second-dimension-knapsack***. A standard definition of the quite popular 0/1 knapsack problem is the following (Martello & Toth 1990).

*Given n objects each one being defined through a given cost $c_i$ and a given value $v_i$ maximize the sum of $v_i$ under the maximum capacity constraint expressed by a real value C which limits the sum of $c_i$'s.*

We will see in which way both our sub-problems can be easily mapped into the knapsack modelization and what are the benefits of this straightforward decomposition.

## 3.1 First dimension sub-problem

Given a pivot order $\theta_p$ the ***first-dimension-Knapsack*** consists in the search for the possible combinations of coils which provides a *feasible* cutting plan just for this pivot order, disregarding the computation of the cutting patterns **Y** which would normally include other available orders to minimize trim loss. This knapsack problem can be formally stated as:

$$Max\left(\frac{\sum_i^n x_i C_i}{W}\right) \qquad (3.1)$$

$$\sum_i^n \frac{x_i C_i}{W} \le \frac{c_p + \delta_p}{y_p w_p} \qquad (3.2)$$

Where 3.1, the knapsack value maximization, maps the objective of finding a coils combination **X** that maximizes the weight of the overall plan: (MAX ($KU$(**X**)) defined by 2.5.

Equation 3.2, the knapsack capacity constraint, guarantees that the coils combination **X** will never exceed the weight of the pivot order $\theta_p$. Notice that this equation must be replicated for each possible number of stripes $y_p$. Since $y_p$ ranges from 1 to ($W$ / $w_p$) we have to actually solve a number of ($W$ / $w_p$) knapsack instances. Being $\bar{k}$ the average number of stripes, to solve the first dimension sub-problem we have to solve $\bar{k}$ knapsack instances on the average (as already mentioned, the value $k$ is certainly contained within the interval [1 . 20], while its average ranges from 2 to 5).

According to 3.1 and 3.2 a solution to the first dimension sub-problem is identified by the tuple **XS** defined as

$$\mathbf{XS} \equiv (\mathbf{X}\ .\ KU(\mathbf{X})\ .\ y_p)$$

i.e. a coils pattern, its corresponding weight and a given number of stripes for the pivot order. Actually, we are interested in all the feasible patterns **X** observing the constraint 3.2. All these patterns will form the first dimension solutions set: {**XS**}. This set can be partitioned into $y_p$ different sub-sets formed through the partition relationship

$$\{\mathbf{XS}\}_r \equiv \{\mathbf{XS} := \mathbf{XS} \text{ s.t. } y_p = r\}$$

Moreover, each sub-set can be ordered by decreasing values of weight, i.e. $KU$(**X**). The possibility to establish such a partitioning and ordering is crucial within the 1D+1D framework. In fact, it will allow to integrate the overall optimization of both the $KU$(**X**) and $TL$(Y) objectives simply through an efficient linear search as explained in 3.3.

## 3.2 Second dimension sub-problem

The goal of the second dimension sub-problem is to find a cutting pattern **Y**, *compatible* with a given first dimension solution, that minimizes Trim Loss. The term *compatible* means that each candidate pattern **Y** must take into account the $y_p$ stripes of the pivot order $\theta_p$ that have already been inserted. Given a first dimension solution, identified by a tuple (**X**, $KU$(**X**), $y_p$), the ***second-dimension-knapsack*** problem can then be formally stated as:

$$Min(W - y_p w_p - \sum_j^{m-1} y_j w_j) \qquad (3.3)$$

$$y_p w_p + \sum_j^{m-1} y_j w_j \le W \qquad (3.4)$$

Similarly to the first dimension sub-problem, equation 3.3 represents the objective of finding a cutting pattern **Y** which minimize the $TL$(**Y**) objective. Equation 3.4 guarantees that the solution patterns **Y** do not exceed the total width $W$. A solution to the second dimension sub-problem is identified through the tuple

$$\mathbf{YS} \equiv (\mathbf{Y}\ .\ TL(\mathbf{Y})\ .\ y_p)$$

i.e. a cutting pattern, its corresponding trim loss and a given number of stripes of the pivot order.

Again, the set of feasible solutions {**YS**} can be partitioned into $y_p$ different sub-sets in the same manner we have done for the {**XS**} set. Moreover, the objective TL(**Y**) induces a significant partial ordering on the each solution sub-set $\{\mathbf{YS}\}_r$ .

## 3.3 1D+1D Integration

To see how the 1D+1D decomposition works on the introductory example of figure 2.2, when the pivot order is $\theta_1$ and $y_1$ equals to three, let us build the corresponding first dimension and second dimension solutions

$$XS' \equiv (X . KU(X) . y_1) \equiv ((1.1) . 190 . 3)$$

$$YS' \equiv (Y . TL(Y) . y_1) \equiv ((3.1.2) . 3 . 3)$$

Suppose now that the following two new coils will become available :

| Coil | Width (inches) | Weight (pounds) | Pound per inch |
|---|---|---|---|
| $\chi_3$ | 25 | 1750 | 70.0 |
| $\chi_4$ | 25 | 3000 | 120.0 |

then we can add the following new solutions

$$XS'' \equiv ((0. 0. 1. 1) . 190 . 3)$$

$$YS'' \equiv ((3. 0. 1) . 2 . 3)$$

The solution sets become thus formed by:

$$\{XS\}_3 \equiv \{((1. 1. 0. 0) . 190 . 3) \quad ((0. 0. 1. 1) . 190 . 3))\}$$

$$\{YS\}_3 \equiv \{((3. 1. 2) . 3 . 3) \quad ((3. 0. 1) . 2 . 3)$$

Hereafter. we will see how the availability of the separated solutions sets $\{XS\}_r$ and $\{YS\}_r$ allow to efficiently build the set of the *good* cutting plans for the original entire RCP problem defined by equations 2.1. 2.2. 2.3. 2.4.

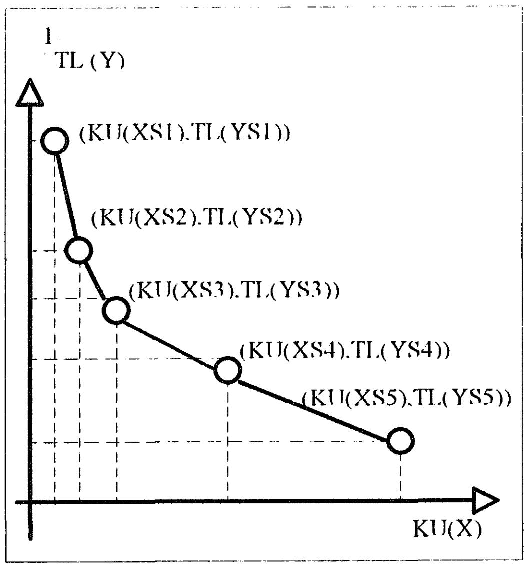

Figure 3.1 Dominance Frontier

According to the dominance relationship 2.6. to build the set of *good* cutting plans we have to find all the *dominant* cutting plans. In other words. we have to build the dominance frontier defined in the plan (*KU*(XS).*TL*(YS)) as shown in figure 3.1.

The following pseudo-procedure (***BUILD-D-***

Figure 3.1 Dominance Frontier

***FRONTIER***) can build precisely the dominance frontier in linear time. The procedure has as input the two collections of sets $\{XS\}_r$ . $\{YS\}_r$ and the maximum number of stripes $k_j$ of the given pivot order $0j$.

```
(Function BUILD-D-FRONTIER ({XS} {YS} k_j)
 ; Compute Dominance Frontier in D-frontier
  ; Init the Dominance Frontier
  (setq D-frontier nil)
  ; Cycle for each number of stripes i of
  ; the given pivot order
 (Dotimes (r  k_j)
   ; find coil pattern X with max KU
   (setq best-X
       (get-first-X  r  {XS}_r ))
   ; find  cutting pattern with min TL
   (setq bets-Y
         (fget-first-Y  r  best-X  {YS}_r  ))
   ; store the current solution
   (push (list best-X best-Y)
           D-frontier)))
```

Thanks to the partitioning and to the ordering of the partial solutions sets {XS} and {YS} the ***BUILD-D-FRONTIER*** can effectively build the dominance frontier in linear time. Being the number of steps within the ***BUILD-D-FRONTIER*** body equal to number of stripes $k_j$. its average time complexity is equal to $O(\bar{k}_j)$. To summarize. the complete set {P} formed by the *feasible* and *good* solutions to the original RCP problem will be formed by <u>*all*</u> the couples

$$P = \{Y . X\}$$

belonging to the dominance set. With the 1D+1D heuristic decomposition. we have practically reduced the complexity of the problem from an expected

$$O(2^{(n+\bar{k}m)})$$

to a more affordable

$$O(2^n + 2^{\overline{k}m})$$

In fact, the solution is composed of two sequential steps : *first-dimension-knapsack* and *second-dimension-knapsack*. Time complexity of the first step is proportional to the power-set of the coils $O(2^n)$ while for the second step the time complexity is proportional to the power-set of the possible stripes $O(2^{\overline{k}m})$.

## 4. Application payoff

Three applications based on the 1D+1D heuristic have been already released to three different customers. Other customer projects are in progress. Table 3.1 shows 15 examples produced during one of the customer acceptance tests. Notice how ROLL-CUTTER produced expert level cutting plans in less than 120 seconds compared to the 1000 seconds taken by human experts.

The payoff of the released applications can be summarized by the following two results:

1) The quality of the cutting plans produced by the applications are comparable or even better than the cutting plans produced by experienced human beings. Particularly, the dominance set always includes the cutting plans that would have been produced by the top level experts.

2) The average processing time taken by the system to produce a set of dominant cutting plans is in the order of thirty seconds. Usually, human experts need from ten to sixty minutes to produce a good cutting plan.

ROLL-CUTTER is currently installed in three different factories :

- A service center whose main activity consists in cutting small coils from bigger ones. The system is integrated with the Manufacturing Information System which supplies the information concerning the commercial demands and the coils to program. The cutting programs formulated by the system are returned to the MIS once validated by the human operator. This kind of system integration applies to the other two ROLL-CUTTER installations as well.

| Cutting Plan # | # Of orders inserted into the Plan | Total weight | # Of knives changes HUMAN EXPERT | % Trim Loss HUMAN EXPERT | TIME (SEC) HUMAN EXPERT | # Of knives changes ROLL CUTTER | % Trim Loss ROLL CUTTER | TIME (SEC) ROLL CUTTER |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 6797 | 4 | 2.64 | > 1000 | 2 | 2.64 | < 30 |
| 2 | 7 | 38924 | 6 | 0.89 | > 1000 | 4 | 0.67 | < 60 |
| 3 | 5 | 25000 | 3 | 1.33 | > 1000 | 3 | 1.33 | < 60 |
| 4 | 4 | 21000 | 3 | 2.30 | > 1000 | 3 | 1.33 | < 60 |
| 5 | 10 | 38000 | 5 | 0.76 | > 1000 | 4 | 0.83 | < 30 |
| 6 | 17 | 16300 | 6 | 0.82 | > 1000 | 5 | 1.52 | < 60 |
| 7 | 5 | 7700 | 2 | 4.79 | > 1000 | 2 | 3.00 | < 60 |
| 8 | 4 | 2600 | 2 | 2.00 | > 1000 | 2 | 1.71 | < 60 |
| 9 | 10 | 43100 | 6 | 0.76 | > 1000 | 5 | 0.96 | < 30 |
| 10 | 7 | 2280 | 2 | 1.33 | > 1000 | 2 | 1.33 | < 120 |
| 11 | 5 | 4620 | 2 | 0.75 | > 1000 | 2 | 0.75 | < 30 |
| 12 | 7 | 9000 | 3 | 3.27 | > 1000 | 3 | 3.17 | < 120 |
| 13 | 5 | 13300 | 5 | 3.62 | > 1000 | 3 | 3.99 | < 120 |
| 14 | 10 | 43100 | 6 | 0.76 | > 1000 | 6 | 0.76 | < 60 |
| 15 | 8 | 12100 | 4 | 1.15 | > 1000 | 3 | 1.00 | < 60 |

Tab. 3.1 Results taken from a Customer Acceptance Test

- A steel industry which direclty produces coils cutting them into smaller stripes. Within this context the auxiliary facilities provided by the system that allows fast selection / ordering of both coils and commercial demands have been particularly appreciated.

- The third site is concerned with thin-film production. In this case the performance of the system has been specifically stressed for a couple of reasons. First of all, the combinatorial facet of the problem was particularly relevant. Secondly, the system had to knock out a software program already used for several years from the human operators.

## 5. Implementations Issues

To implement the ROLL-CUTTER basic platform it took us two man years using Gold-Hill Common Lisp Developer running under Windows 3.11. We found the Lisp language specially helpful during the tuning of the knapsack search algorithms.

A variety of well studied exact and approximate algorithms to solve one-dimensional knapsack problems exist (Martello & Toth 1990). The two different one-dimensional knapsack problems, can be seen as two special instances of the general "value-independent-knapsack-problem" (also known as subset sum problem) where $n$ objects characterized by different cost values $c_i$ and a constant profit value $v$ must be put into a knapsack without exceeding its capacity $C$. Toth's book devoted to 0/1 knapsack problems presents the well studied MTS algorithm which is based on tree search and dynamic programming techniques. In our implementation we followed this approach augmenting the basic tree search mechanism with heuristic criteria we derived from the specific nature of the problem.

Two different depth-first tree-search algorithms (DFTS) have been implemented. The two algorithms have a common depth-first strategy which provides the implicit enumeration mechanism on top of which two specialized heuristic search policy have been imposed. The features of such DFTS heuristics are the following

◊ First-dimension-DFTS features:

  - Coils, which represents the knapsack objects, are pre-ordered by decreasing size (we verified how this ordering increases the pruning power).
  - Lower values of $y_j$ for the "pivot" order are tried first.

◊ Second-dimension-DFTS features:

  - For each candidate order $\theta_j$ the number of stripes $y'_j$ which allows to exactly match the demanded weight are tried first. This means to select first a number of stripes $y'_j$ such that

$$c_j \leq \frac{\sum_i^n x_i C_i}{W} y'_j w_j \leq c_j + \delta_j$$

  In this way the splitting of an order in different lots is avoided as much as possible.

## 6. Maintenance & Extensions

ROLL-CUTTER is in continuous evolution since its first customer installation which happened to be in January 1995. Within this evolution we are pursuing two main objectives:

- Continue to improve the performance of the basic 1D+1D decomposition technique

- Extend the application of the 1D+1D technique to other different cutting stock domains.

In the first direction we are currently developing, for a ROLL-CUTTER customer a parallel implementation of the 1D+1D algorithms. The implementation is based on a cluster of 586 PC's which co-operate during the building of the feasible solutions sets {XS}, {YS}. This improvement will allow us to tackle the production of cutting plans optimized throughout the overall orders portfolio.

Concerning the extension of the application domains we already successfully applied the 1D+1D technique to a different kind of industry : thin-film. Even if the basic technique is the same, this attempt generated some modification to the original ROLL-CUTTER platform. These needed modifications will bring us to generalize the overall ROLL-CUTTER shell. The idea is to have a unique platform providing the basic 1D+1D different decomposition mechanisms from which several more specialized platforms can be built. Other than Thin-film, another specialized platform is going to be derived for paper industry (glass industry is a good candidate too). Moreover, in the near term we intend to experiment the 1D+1D "unfolding" technique even with 3D placement problems (e.g. ships loading and unloading).

## 7 Conclusions

The 1D+1D problem decomposition allowed us to face a complex two dimensional cutting stock problem (RCP) reaching expert level performance. The method has produced exemplary results within several released applications. So far, we already demonstrated that the method is able to discover the complete set of dominant optimal solutions, 95% of the time, in a few seconds. It takes 15 minutes or more to build the "next" near optimal cutting plan to experienced human beings.

The ROLL-CUTTER platform on which all the released applications are based is now evolving in different directions. A parallel implementation of the 1D+1D technique to solve the global optimization of the overall cutting shop-floor is in progress. Moreover, we are generalizing the platform to facilitate the development of different applications addressing other industry sectors.

## References

Albano, A. and Saluppo, G. 1980. *Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods.* IEEE Transaction on Systems, Man and Cybernetics. Vol. smc-10, No. 5, pp. 242-248.

Dyckoff, H. and Kruse, H. 1984. *Trim Loss and Related Problems.* The Int. Jl of Management Science. Vol. 13, No. 1 pp. 59-72.

Ferreira, J.S. 1990. *A two-phase Roll Cutting Problem.* European Journal of Operational Research. No. 44, pp. 185-196.

Gilmore, P.C. and Gomory, R.E. 1961. *A Linear Programming Approach to the Cutting Stock Problem.* International Business Machines Corporation, Research Center, Yorktown, New York, pp. 849-859.

Gilmore, P.C. and Gomory, R.E. 1966. *The Theory and Computation of Knapsack Functions.* International Business Machines Corporation, Yorktown Heights, New York, pp.1045-1074.

Ibarra, O.H. and Kim, C., 1975. *Fast approximation Algorithms for the Knapsack and Sum of Subset Problem.* Journal of the Association for computing Machinery. Vol. 22, No. 4 pp. 463-468.

Martello, S. and Toth, P., 1990. *0/1 Knapsack Problems: Algorithms and Computer Implementation.* John Wiley & Son Ltd.

Pearl J., eds. 1985. *Heuristics, Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley.

Scutella M.G.; Bertolotti E.; Castaldo E.; Gambale M.; 1995. *Cutting Stock: Hypergraphs and Decomposition techniques.* Dept. of Informatica, Pisa Univ.. Proceedings of the Conference: Models and Algorithms for Optimization Problems" (January 1995).

Sahni S., 1975. *Approximate Algorithms for the 0/1 Knapsack Problem.* Journal of the ACM. Vol. 22 No. 1 pp. 115-124.