INTERNATIONAL DESIGN CONFERENCE - DESIGN 2000
Dubrovnik, May 23-26, 2000.

ROD CUTTING OPTIMIZATION WITH STORE UTILIZATION

Leon Kos and Jože Duhovnik

rod cutting, bin packing problem, multiple subset sum problem

Abstract: For cutting linear elements like steel rod or marble shelf from standard lengths, optimization for best utilization of raw material is frequently required to minimize waste and reduce the production costs with packing smaller elements to standard lengths. As the input material can be order dependent, store handling is limited only to good remnants from previous cuttings or over-stocking. To handle production of this type rationally, solution consist of several types of optimizations.
Putting theory to work in real production demands several aspects of application to be considered to fit the needs of designers and orders staff.

1 Introduction

In this paper, we first describe the $\mathcal{NP}$-complete optimization paradigm. Cutting from standard dimensions is in theory known as bin-packing problem with several different approaches from heuristic to genetic approximation solutions.
Real production also requires storing of good remnants for possible future use which additionally complicates this computationally intensive problem. To consider store and still give useful results, the problem is divided into knapsack optimization and bin packing optimization. With multiple knapsack problem solution we check for partial solution from store an then reduce the problem for bin packing. Bin packing is held over several standard length to optimize. Transformation of input lengths is introduced to allow cutting additions.
Structural steel is cut to preferred size from standard lengths. This way of production minimizes costs, transport logistics and enables production flexibility. Rod cutting is thus planned at the stage of design where production material is not on the stock and the preferred way is to order beams of the standard length and then cut it with minimized waste.
After dimensions are known, overview is prepared for production and orders. Detailed production plan in prepared for production and specifies cutting source from order or store. Store typically supplies non-standard length beams as a result from previous cuttings. Store is used only for a good remnants, cutting failures, and over-stocking for reserves from previous orders.
Manual combination of prescribed lengths into standard lengths can be tedious work prone to errors and non-optimal. If one must consider several standard lengths, finding best solution is also time consuming. Obvious choice of computing power leads to development of several methods to attack specific needs and optimization flexibility.

2 Background

Cutting optimization is a problem which can be divided into subproblems when we utilize store. Solution also depends on type and size of input data. Optimization process is defined as:

> Result of the algorithm is the description of the data which is somehow optimal regarding to input data.

Integer optimization is much more difficult than optimization with real numbers. Algorithm normally needs two sources for solution of the problem: *time and space*. The time is measured in the number of state changes in algorithm from start to end. Space is normally defined as maximum space needed for temporary data, which is needed over whole computation. Resource size for solving the problem with some algorithm is covered by complexity analysis which deals with *time* and *space* complexity. There is a difference between algorithm complexity and problem complexity.
Algorithm complexity describes time needed by algorithm for a solution of problem with given size. Algorithm complexity is measured as *worst case*. Problem complexity is defined as complexity of the best algorithm for the problem. This does not necessary mean that the best algorithm is known, although the problem is as hard as the best algorithm solving it is known. For many $\mathcal{NP}$-hard problem is difficult to find algorithm which will run in reasonable time on all instances and be guaranteed to find global optimum. Approximation algorithm is for such problems oftenly the only workable solution.
Rod cutting optimization consist of two $\mathcal{NP}$-complete problems (*nondeterministic polynomial*) which can attacked by several algorithms described as follows.

3 Bin Packing Problem

The bin packing problem (BPP) is defined as follows[Garrey, 1979]: Given a finite set O of numbers (the object sizes) and two constants c (bin capacity) and the m (number of the bins), is it possible to 'pack' all the objects into m bins, i.e. does there exist a partition of O into m or less subsets, such that the sum of elements in any subset does not exceed c.
Lower bound of the necessary number of bins can be calculated as

$$OPT = \frac{\sum x_i}{c}, \quad i \in \{1, \ldots, n\} \tag{1}$$

The first choice for BPP solution can be one of the approximation algorithms which pack items into bins with simple heuristics such as:

First Fit packs unassigned element into first bin which has enough space. If there is no such bin, assign element into new bin.

Last Fit packs unassigned element into last bin with enough space. Searching is similar to FF but in the reverse order of bins. If there is no such bin assign element into new bin.

Next Fit. Unassigned element is checked against last bin. If there is no space left, element is assigned to new bin.

Best Fit. Unassigned element searches for the bin which best fills the bin. If there is no space in the bins, assign element to new bin.

Exact Fit. As the speed of the above algorithms is fast enough for number of items up to 1000, one can spent more time for additional searching and improve BF algorithm with group fitting. Exact Fit heuristic[Djang and Finch, 1998] can be described with the following algorithm:

1. Take a new bin and fill it with elements until the bin is over 1/3 full.

2. Try to fill the rest of the bin exactly with single element or combination of the two or three elements.
3. If there is no such combination that will fill bin exactly, relax the problem with decreasing the bin size for one and repeat procedure 2 until success.
4. Repeat the whole procedure while there are unassigned elements available.

Initial ordering can also impact the packing process and is normally used as preprocessing stage. Random ordering is used only when the packing is continuous. Descending ordering is the strategy of choice for described algorithms as it tries to pack bigger elements first. Ordered approximation algorithms such as First Fit Descending (FFD) can be measured by quality of packing. It was shown[Garrey, 1979] that for arbitrarily large instances FFD heuristic will use as many as

$$FFD(I) = \frac{11}{9}OPT(I) \text{ bins.}$$

Meta-heuristics like simulated annealing (SA), tabu search (TS) and genetic algorithms (GA) can provide better results. Genetic algorithms have wide variety of applications and for problems like BPP perform better than SA and TS.
BPP has its specifics that impacts generation of offsprings with crossover, which is standard genetic operation. For better crossover a grouping method in chromosome is used. This kind of chromosome encoding is called *Grouping Genetic Algorithm*[Falkenauer, 1998]. GGA differs from classical GA in two different aspects:

Encoding. If standard encoding is used in BPP, crossover of two parents will not generate good offsprings as the gene would be out of sequence even if we replace only one element from one parent into another. Grouping of items is done though lookup-table. Example of lookup assignment of six items (0 to 5) assigned into bins marked with letters is

```
012345
ADBCEB
```

which means that item 0 is assigned to bin A, 1 to D, 2 and 5 to B, and 3 to C. Valid chromosome in term of GGA can be encoded as BECDA. Contents of gene (bin) can always be examined using lookup table. It is important no notice that this encoding scheme yields *variable-length* chromosome. Ordering of genes in chromosome has no impact to solution (e.g. BECDA and ABCDE represents the same solution).

Crossover. Crossover is operation which passes properties of parents to offsprings. Genes from both parents are using when combining chromosome.

For BPP crossover uses the following algorithm for generation of new solutions:

1. Choose two random sites at both parent chromosomes. For example, let us have two chromosomes:

   ```
   ABCDEF
   abcd
   ```

   which are partitioned into

   ```
   A|BCD|EF
   ab|cd|
   ```

2. Insert content between sites from second parent into first site of the first parent. Injection would produce new chromosome

```
AcdBCDEF
```

3. Some item in genes are duplicated and must be removed from solution to have a valid chromosome. Let us suppose that some items in injected bins `c` and `d` also appears in bins `C, E` and `F`. We remove the whole bins remaining the following partial chromosome

```
AcdBD
```

4. If it is possible, correct the chromosome according to bin size limits and minimize the cost function. At this stage, local optimization is performed with application of the FFD, EFD or domination criteria[Falkenauer, 1996, Martello and Tooth, 1990] to reassign unassigned items.

   For generation of second offspring the role of parents is commuted and procedure from 2. to 4. is performed.

Simple counting of the bins in not useful for cost function as the differences in packing are not shown. To promote better filled bins cost function is defined as

$$f_{BPP} = \frac{\sum_{i=1..n}(F_i/C)^k}{n} \tag{2}$$

4 Multiple Subset Sum Problem

MSSP[Capara et al., 1998] is a special case of the *Multiple Knapsack Problem* where the number of the sacks (bins) is fixed to some small number. We are considering problem where n items is packed into m sacks of different capacities $c_i, i = 1, \ldots, m$. The problem is to choose m different subsets of items in a way that every subset i fits into sack i and that the profit is maximized. Formally the MSSP can be defined as

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} w_j x_{ij} \leq c_i, \quad i = 1, \ldots, m, \\
& \sum_{i=1}^{m} x_{ij} \leq 1, \qquad j = 1, \ldots, n, \\
& x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n
\end{aligned} \tag{3}$$

For MSSP similar approximative algorithms as in BPP can be used for packing. Difficulty to the MSSP presents number of the knapsacks. For a problems where the quotient n/m is relatively large exact methods were developed[Pisinger, 1995]. Branch-and-bound techniques are mostly used in literature [Martello and Tooth, 1990, Martello et al., 1999].

5 Cutting with store utilization

When cutting beams, remnants can be significant even when several standard lengths are used in optimization. Storing this partial beams for future reuse is thus justified. Optimization must therefore also consider also items on stock. Optimization with store utilization consists of the following important steps:

1. For a certain shape of beam one must determine span of the lengths(min/max) on stock. From the order the items are selected which conforms to the store span for MSSP optimization. This removes obvious unfitted items from optimization which are left to BPP optimization.

2. Algorithm for a MSSP is selected according to the type of problem which is designated by number of items n, number of knapsacks m and type of knapsacks (capacity dense, capacity grouped, general). If the the quotient n/m is relatively large (> 10) and $m < 12$ exact method with dynamic programming is applied. Polynomial time approximation scheme (PTAS)[Caprara et al., 1999] is applied for other sizes and considering the type of knapsacks.

3. Items not assigned to the knapsacks and items not used in MSSP optimization are used in BPP optimization. Here the best standard length is not known and is selected with the scan over the range of the lengths. There is no suggestion about the method for packing and can be selected by experience on method performance during the optimization process.

4. If the standard length selected from the BPP optimization and selected length was also used in MSSP from store, the problem is relaxed by removing the knapsacks with standard length from MSSP optimization. The optimization is then repeated typically yielding better solution.

5.1 Cutting addition

When cutting beams one must consider also the width of the cut. This is important if there are several cuts and total sum exceeds beam size tolerances. Optimization with cutting addition can be performed using simple transform as shown in figure 5.1. This transform shows that every item gets cutting addition.

Figure 1: Cutting addition transform adds cut size to items (top) and to bin (bottom)

To assure that last item will still fit into the bin even if there is no cut at the end, bin size is also enlarged for cutting addition. This transform is applicable to BPP and MSSP optimization. Inverse transform is straightforward.

6 Results

Using application in industrial implementation takes optimization program to consider most input data situations. This is especially true for MSSP part of optimizations where type of optimization depends on size and type of items in store.
Application was implemented to provide the described optimization functionality. Since the optimization is numerically intensive the application was divided into engine and user interface. Engine written in C provides DLL communication to other programs (GUI). User interface in Visual Basic provides a way for navigating data flow from various input sources from/to a common company database or local ODBC connection. After initial installation, it was recognized that user needs a full control over optimization process and that tuning of the process is an important issue. Selection of items to optimize can be specified using SQL giving user a choice for complex specification of where clause. This typically includes shape, order and drawing number. Apartment model multi-threading of VB was identified as drawback when using external DLL engine because there is no way to control the state of the optimization engine. Optimization time can vary from fragment of second to several minutes, depending on input parameters and problem size.
Additional problems caused store communication as item storing must be tailored for reservation, trial runs and optimization discards. Selection of items from store can also be a critical point which oftenly requires user decision. The number of stored items can grow to the size where MSSP optimization

can choke. Among them can be numerous elements of standard lengths. There are also cases where MSSP can dominate the selection and leaving small number of items to BPP, but the packing quality is unacceptable and user is forced to drop some or whole part of MSSP propagating optimization to BPP.
Genetic algorithms for BPP performs best on typical data size. When response time is preferred for preliminary calculation, other faster algorithms can be selected. It is important no notice, that for GGA cost function (2) tends to minimize waste as it promotes better packed bins. This kind of optimization thus leaves better remnants for store.
Quality of packing was in most cases under 3%, when considering very big remnants as good items to be placed in store. On small optimizations theoretical number of bins can be achieved and thus optimal solution was found. It is also possible that optimal solution is found and the waste is significant. Leaving user to decide what to do with optimization (e.g. order nonstandard lengths). Typically remnants are so small that it cannot be considered useful for store. First consideration when developing the application was that the number of items on stock will be small. It turned out that this was optimistic thinking and new algorithms for MSSP optimization were introduced to handle large number of items on store.

7 Conclusion

Application clearly show that optimization can substantially help the design and production process for cutting material of steel structures, metal sheets, panels and other one dimensional elements. Time and cost reduction can be significant and motivation for coding better algorithms to suit the specific needs can easily be evaluated.

References

[Capara et al., 1998] Capara, A., Kepler, H., and Pferschy, U. (1998). The multiple subset sum problem. Technical report, Faculty of Economics, University of Graz.

[Caprara et al., 1999] Caprara, A., hans Kepler, and Pferschy, U. (1999). A PTAS for the multiple subset sum problem with different knapsack capacities. Technical report, University of Graz, Austria.

[Djang and Finch, 1998] Djang, P. A. and Finch, P. R. (1998). Solving one dimension bin packing problems. *submited to Journal of Heuristics*.

[Falkenauer, 1996] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(2):5–30.

[Falkenauer, 1998] Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Wiley Computer Books.

[Garrey, 1979] Garrey, Michael R. amd Johnsos, D. S. (1979). *A Guide to the Theory of NP-completness*. W.H. Freeman, San Francisco, USA.

[Martello et al., 1999] Martello, S., Pisinger, D., and Toth, P. (1999). New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*. accepted for publication.

[Martello and Tooth, 1990] Martello, S. and Tooth, P. (1990). *Knapsack Problems, Algorithms and Computer Implementations*, chapter 8, pages 221–245. John Wiley & Sons, England.

[Pisinger, 1995] Pisinger, D. (1995). *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen.

Leon Kos
University of Ljubljana, Faculty of Mechanical Engineering
CAD laboratory
Aškerčeva 6, SI-1000 Ljubljana, Slovenia
tel: +386 61 1771 436, fax: +386 61 121 4620
e-mail: leon.kos@lecad.uni-lj.si