Automatic Combination and Optimization System

Kevin Tay Nissan Pow Juti Noppornpitak Aaron Wong Eddie Lai

June 30, 2006

Contents

1 About ACOS

Automatic Combinatorics and Optimization System (ACOS) is a system that automatically combines an input file and mathematical model to generate an optimal solution. ACOS targets industrial steel companies to help them decide how much input coil they should buy, how the input coil should be cut, and how much inventory should be stored in order to satisfy the monthly demand and other constraints. Moreover, it helps the companies to estimate the optimal profit within a certain period.

ACOS is currently available on internet: http://www.csclub.uwaterloo.ca/∼jnopporn/orc/

2 Problem Description

An industrial steel company provides coils of sheet metal to manufacturing companies, building industries, and municipal governments. Let us only consider one of their products: cold rolled steel coils, 1 mm in thickness, 1280 meters long, and of various widths. They receive orders for these coils of a certain width (let us call them *output coils*) and they cut them out from wider coils (*input coils*). The company wishes to maximize the total profit, which is the total sales revenue minus the total cost of input coils.

The company buys n input coils of width IW_i, $i = 1,...,n$ from a steel mill. To purchase them at a bargain price, they need to buy in bundled bulks, where a bundle contains n_i coils of width IW_i, $i = 1,...,n$ and cost c_0 dollars. If they were to buy the input coils individually, it will cost c_i dollars for input coil i.

The company slices the input steel coils to make m different products of width OW_j, $j = 1,...,m$, which are sold at price p_j , $j = 1,...,m$. They lose a width of L each time the input coil is cut. Let us consider the problem for T months. Each month, the company's factory can process up to M output coils, regardless of their width. Expected orders for each output coil j is d_jt for month t, $t = 1,...,T$, $j = 1,...,$m. An order can be declined if the factory is over-capacity.

Output coils can be cut ahead of times and stored in inventory. They are stored according to their width separated into storage facilities for small, medium and large coils. There are varying capacity for each of these storage rooms. Input coils can also be stored in inventory for later use. It is the practice of this company to not reuse input coils after they are cut (i.e., once an input coil is cut up, the left-over is not stored in inventory). Thus, an input coil is cut up until the left-over segment cannot be cut into any of their product widths.

Our task in this contest is to model and solve the static deterministic planning problem for this company. Clearly, the real problem involves more complexity, such as lead-times for the raw materials (we are assuming just-in-time delivery of the input coils, whereas in reality, it may take several months. Often times, the steel mills may not be able to supply the requested amount, in which case the company must seek supplies from Chinese or Japanese steel mills), uncertain demand (what product to keep in inventory is heavily dependent on expected future demand. Thus, poor demand forecast can lead to short stocks and overstocks), steel price fructuation (we assume price is constant whereas this is far from reality), and several details concerning production and inventory management. However, even this simplifed model will allow the company to determine whether or not to take on new orders, plan which type of input coils to order, and determine whether they should upgrade their facility to increase production or allow more storage space for certain product coils. Also, this type of model would be run every month (or more frequently) when new orders arrive and the data updated accordingly.

3 Mathematical Model

The mathematical model is implemented in GAMS format using integer programming methology.

3.1 Variables

- $x_{i,t}$ the number of input coil i purchased in month t
- $y_{i,h,j,t}$ the number of output coil j produced by input coil h of type i in month t
- $sold_{j,t}$ the number of output coil j sold in month t
- $r_{i,t}$ the total number of unused input coil i in month t
- $Out_{j,t}$ the number of inventory for type j output coil from month t
- $In_{i,t}$ the number of inventory for type i input coil from month t
- BP_t the number of Bundles purchased in month t
- $v_{i,t}$ the number of individual input coil i purchased in month t
- z the total profit

3.2 Equations

3.2.1 Objective Function

- Profit:
$$z = \sum_t (\sum_j (P_j \times sold_{j,t}) - BP_t \times C_0 - \sum_i v_{i,t} \times C_i)$$
The total profit for T months. In each month:
profit = (total revenue) - (cost of bundles purchased) - (total cost of individual coils purchased).

3.2.2 Constraints

- Total output(t):
$$\sum_{i,j,h} y_{i,h,j,t} \leq M \qquad \forall t$$
For each month, the total number of output coils produced cannot exceed M.

- Cut(i,t):

$$(In_{i,t-1}+x_{i,t})\times IW_i \geq \sum_j \sum_h (OW_j \times y_{i,h,j,y}) + (((\sum_j \sum_h y_{i,h,j,y}\$\, (IW_i - OW_j \; gt \; L)) - \sum_h count_{i,h,t}) \times L)$$

$$+ \; r_{i,t} \times IW_i \qquad (\forall i, \forall t)$$

The cutting constraint, which determines how all the input coils are cut. For all input coils,
(total width of input coils) ≥ (total width of output coils produced) + (total width lost from cutting) + (total width of unused input coils).

- Check Waste (i,h,t):

$$IW_i \geq \sum_j (y_{i,h,j,t} \times OWj) + \sum_j (y_{i,h,j,t} - 1) \times L \qquad (\forall h, \forall t)$$

Essentially the cutting constraint for each individual input coil purchased. This constraint is used to ensure that we cannot combine waste from different input coils to produce output coils. For each input coil,
(width of input coil) ≥ (total width of output coils produced) + (total width lost from cutting).

- Storage Constraint A (j,t):

$$Out_{j,t-1} + \sum_i \sum_h (y_{i,h,j,t} - sold_{j,t}) \geq Out_{j,t} \qquad (\forall j, \forall t)$$

The total number of output coils stored in the inventory for each month cannot exceed the number of remaining output coils for that month.

- Storage Constraint B (j,t):

$$r_{i,t} \geq In_{i,t} \qquad (\forall i, \forall t)$$

The total number of input coils stored in the inventory for each month cannot exceed the number of unused input coils for that month.

- Storage Process S (small,t):

$$\sum_{a(j)} Out_{j,t} \leq O_{small} \qquad (\forall small, \forall t)$$

The total number of small-size output coils stored in the inventory cannot exceed the storage capacity for the maximum number of small-size coils.

- Storage Process M (medium,t):

$$\sum_{b(j)} Out_{j,t} \leq O_{medium} \qquad (\forall medium, \forall t)$$

The total number of medium-size output coils stored in the inventory cannot exceed the storage capacity for the maximum number of medium-size coils.

- Storage Process L (large,t):

$$\sum_{k(j)} Out_{j,t} \leq O_{large} \qquad (\forall large, \forall t)$$

The total number of large-size output coils stored in the inventory cannot exceed the storage capacity for the maximum number of large-size coils.

- Order Constraint A(j,t):

$$Out_{j,t-1} + \sum_i \sum_h y_{i,h,j,t} \geq sold_{j,t} \qquad (\forall j, \forall t)$$

The total number of output coils sold cannot exceed the total number of output coils that is available. For each month,(total output-coil inventory from last month) + (total number of output coils produced in the current month) $\geq$ (total number of output coils sold in the current month).

- Order Constraint B (j,t):

$$sold_{j,t} \leq d_{j,t} \qquad (\forall j, \forall t)$$

For each output coil type, the total number of output coils sold cannot exceed the demand for that type.

- Input Limit A (i,t):

$$In_{i,t} \leq Cap_i \qquad (\forall i, \forall t)$$

For each input coil type, the total number of input coils stored in the inventory cannot exceed the storage capacity for that type.

- Input Limit B (i,t):

$$r_{i,t} \leq x_{i,t} + In_{i,t-1} \qquad (\forall i, \forall t)$$

The total number of unused input coils in one month cannot exceed the total number of input coils available (purchased + inventory) for that month.

- Bundle (i,t):

$$BP_t \times n_i + v_{i,t} = x_{i,t} \qquad (\forall i, \forall t)$$

bundle(i,t) – Equation representing the total number of input coils available each month (obtained from purchasing bundles or bought individually). For each input coil of type i:
(number of bundles purchased $\times$ number of input coil of type i obtained in one bundle) + (number of input coils bought individually) = (total number of input coils available).

- Check Width (i,j,h,t):

$$y_{i,j,h,t}\$ \ (OW_j \ ge \ (IW_i + 1)) = 0 \qquad (\forall i, \forall j, \forall h, \forall t)$$

Ensures that the width of any output coil produced from an input coil cannot exceed the length of that input coil.

- Check Shave (i,j,t):

$$\sum_h y_{i,j,h,t}\$ \ (IW_i - OW_j le \ L) \leq In_{i,t-1} + x_{i,t} \qquad (\forall i, \forall j, \forall t)$$

For each i and j, if $(IW(i) - OW(j) \leq L)$, then we can produce at most one output coil of type j from one input coil of type i.

4 Algorithm Description

4.1 System Process Algorithm

The Client can choose to either upload a data file or gams formatted input file. In either case, the client needs to make sure to submit an input file with correct format and structure, which should be in exactly the same format as the example from the OR Contest web site. The system only accepts .txt or .csv type files. To submit the file, please go to 'Make a GAMS input file', and upload the file based on the file type. In our system, it will automatically combine the source file and the mathematical model and provide a GAMS file. Clients can see if the GAMS file is successfully generated at 'Terminal'. The client can click on the file and select 'Make an output', and write down his/her email address. After that, the system will transfer the file to the NEOS solver. After a while, the NEOS solver will send the solution directly to the client's email account.

4.2 Mathematical Algorithm

Although the industry standard for large scale cutting stock problem is using either Column Generation Algorithm or Dantzig-Wolfe decomposition, ACOS is running Linear Programming Relaxation of the Integer Program. The reason is that column generation or Dantzig-Wolfe decomposition are normally for one dimensional problem. However a cutting stock problem is a multi-dimensional problem with multi-input, multi-output, multi-period, multi-storage and multi-pricing model. In order to program an accurate and reliable system, we decided to stick with the 'not so' efficient LP Relaxation algorithm. It may not be efficient in terms of running time, but it is effective in debugging and fixing variations. The following are some specific algorithms we designed for the model.

4.3 Pattern

To cope with the problem of cutting pattern, we decided to add a subscript h to represent the input coil number and the limit of h is the maximum output we can produced plus the maximum inventory. Therefore, we would know how much output is produced from each particular input coil. This helps to determine how to cut the coil.

4.4 Cut or Shave

We designed the model that can decide whether to cut, shave, or shave after cutting. To achieve this, we implemented several conditional statements within the constraint itself. $\$(IW_i - OW_j \ gt \ L)$ means that if the input coil size minus the output coil size is greater than the loss due to cuting, the system will perform a cut. Otherwise, it will perform shave. We also implemented constraints CheckShave, CheckWidth, CheckWaste to keep track of when to shave, when to cut, and when to shave after cut. Shaving input coils reduces waste, and thus gives the company more profit.

4.5 Small, Medium and Large Storage Capacity

We decided to distinguish the size of the storage capacity when the system reads the input file instead of doing it on GAMS, as it would more efficient.

4.6 Clear Identification of Variables

In our mathematical model, we clearly identify the varibles. For the input, we separate it to number of coil we bought, number of inventory for input coil, and number of unused input coil. For the number of output, we separate it to number of output sold, number of output produced, and number of output being stored. It helps us to test the system effectively and efficiently. Also, it helps the client to easily understand the problem.

5 List of Assumptions

- There are only three cutting processes: cut, shave, and shave after cut.
- There are only three inventory sizes for output coil: small, medium and large.
- The coil widths are always positive integers.
- The prices for each input coil are always positive integers.
- The length of the coils are irrelevant.
- The demand forecast is correct and we do not have to always satisfy the demand.
- Once an input coil is cut, it cannot be stored.
- It is feasible to buy more in one month and save for another month.
- Coils do not rust in inventory.
- There is an unlimited number of input coils.