Facoltà di Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

Progetto del corso Ingegneria di Internet e del Web a.a. 2017-2018

Formulation and Solution Approaches

Andrea Graziani - matricola 0273395

27 giugno 2019

- The **Cutting-stock problem** (**CSP**) is the problem concerning cutting standard-sized pieces of stock material, called *rolls*, into pieces of specified sizes, **minimizing material wasted**.
- In order to formalize that problem as an **integer linear programming** (**ILP**), suppose that stock material width is equal to W, while m customers want n_i rolls, each of which is wide w_i, where $i = 1, ..., m$. Obliviously $w_i \leq W$.

The Cutting-stock problem
ILP Formulation

$$
\begin{aligned}
(P1) \quad & \mathit{min} \quad \sum_{k \in K} y_k \\
& s.t. \quad \sum_{k \in K} x_i^k \geq n_i, && i = 1, ..., m && \text{(demand)} \\
& \sum_{i=1}^{m} w_i x_i^k \leq W y_k, && \forall k \in K && \text{(width limitation)} \\
& x_i^k \in \mathbb{Z}_+, y_k \in \{0, 1\}
\end{aligned}
$$

- An alternative, stronger formulation, is due to Gilmore and Gomory, which main idea is **to enumerate all possible raw cutting patterns**.
- A pattern $j \in J$ is described by the vector $(a_{1j}, a_{2j}, ..., a_{mj})$, where a_{ij} represents the number of final rolls of width w_i obtained from cutting a raw roll according to pattern j.
- In this model we have an integer variable x_j for each pattern $j \in J$, indicating how many times pattern j is used; in other words it represents how many raw rolls are cut according to pattern j.

$$(P2) \quad min \quad \sum_{j=1}^{n} x_j$$

$$s.t. \quad \sum_{j=1}^{n} a_{ij}x_j \geq n_i, \qquad i = 1, ..., m \quad \text{(demand)}$$

$$x_j \in \mathbb{Z}_+, j = 1, ..., n$$

Where n represents the total number of cutting patterns satisfying following relations:

$$\sum_{i=1}^{m} w_i a_{ij} \leq W \qquad (1)$$

$$a_{ij} \in \mathbb{Z}_+$$

- **How many cutting patterns exist?**

  In general, the number of possible patterns **grows exponentially** as a function of m and it can easily run into the millions. So, it may therefore become impractical to generate and enumerate all possible cutting patterns.

  Even if we had a way of generating all existing cutting pattern, that is all columns, the **standard simplex algorithm** will need to calculate the reduced cost for each non-basic variable, which is, from a computational point of view, impossible when n is huge because is very easy for any computer to go **out of memory**.

- Fortunately, another better approach, which we have used in our project, exists: **column generation method**.

This method solves the cutting-stock problem by starting with just a few patterns and it generates additional patterns when they are needed. To be more precise, the new patterns are found by solving an auxiliary optimization problem, which, as we will see, is a knapsack problem, using dual variable information from the linear problem. Auxiliary knapsack problems can be resolved efficiently in $O(mW)$ time using dynamic programming or branch and bound method.

Grazie per l’attenzione!