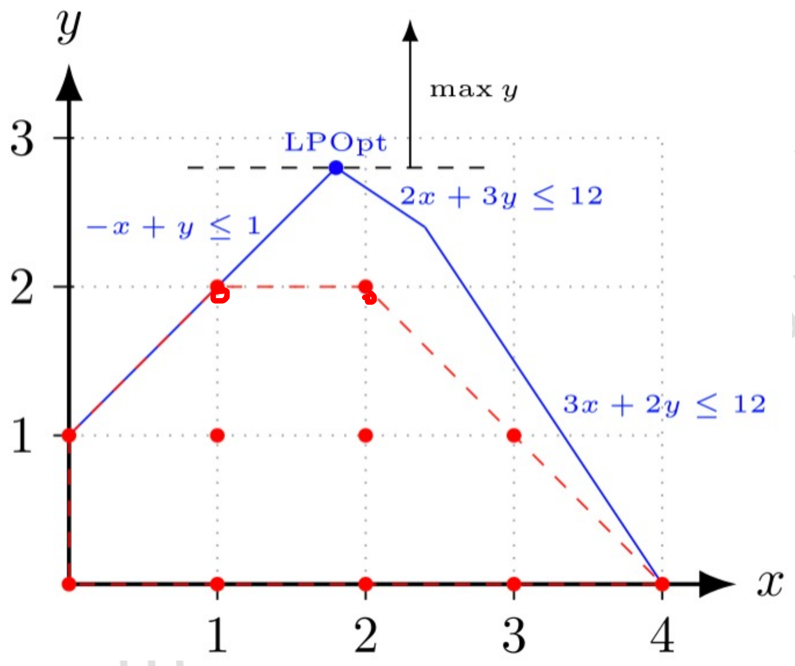
Integer programming examples

Integer programming (IP)

- An **integer programming** problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be **integers**.
- In many settings the term refers to **integer linear programming** (ILP), in which the objective function and the constraints (other than the integer constraints) are **linear**.
- Integer programming is non-convex optimization and NP-complete.

Example:

$$
\begin{aligned}
&\max y \\
-x + y &\leq 1 \\
3x + 2y &\leq 12 \\
2x + 3y &\leq 12 \\
x, y &\geq 0 \\
x, y &\in \mathbb{Z}
\end{aligned}
$$

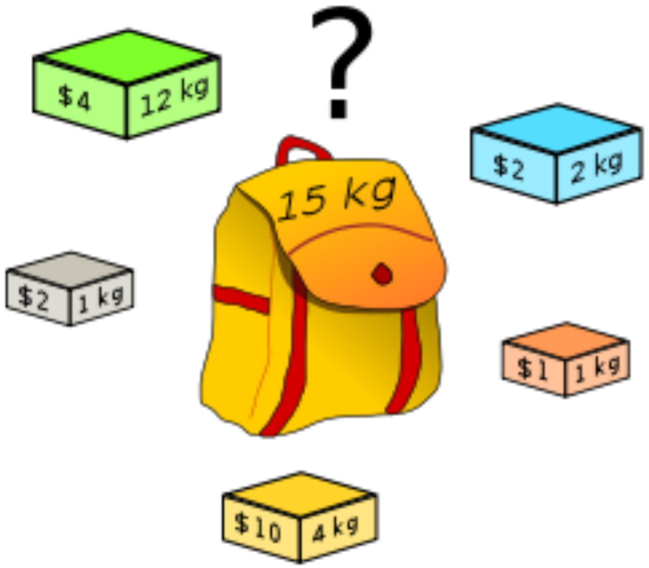


https://en.wikipedia.org/wiki/Integer_programming

Examples

- 0-1 knapsack problem
- Cutting stock problem
- Travelling salesman problem

0-1 knapsack problem



- Dynamic programming
- 0-1 integer programming

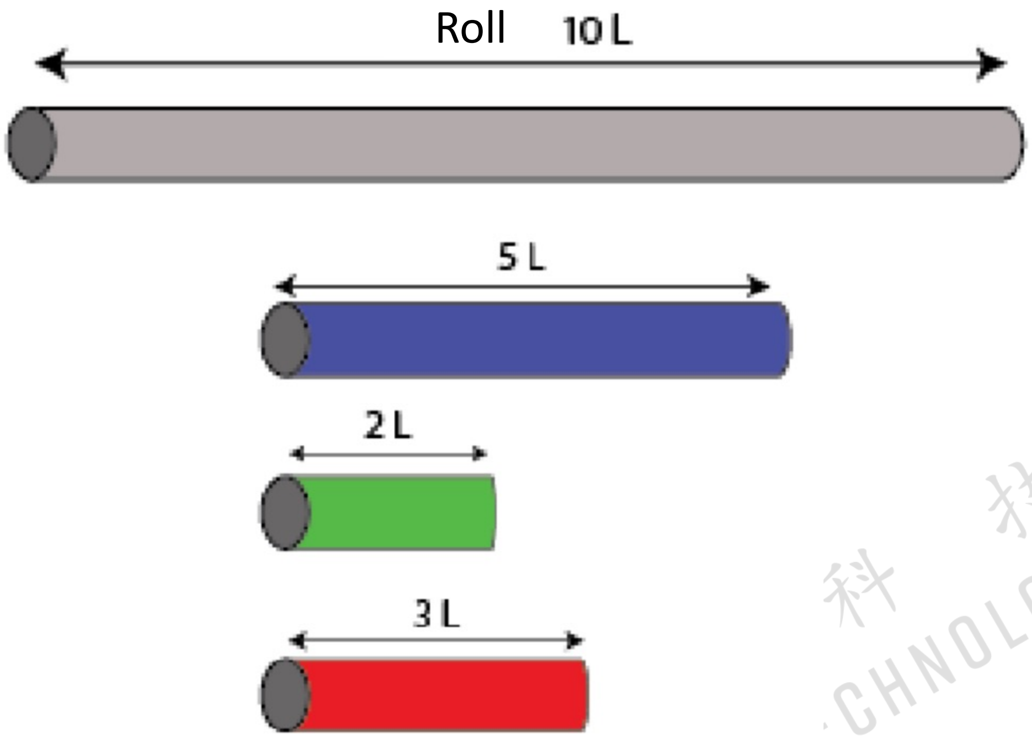
$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

https://en.wikipedia.org/wiki/Knapsack_problem
https://towardsdatascience.com/dynamic-program-vs-integer-program-which-one-is-better-for-the-knapsack-problem-759f41b9755d

Cutting stock problem

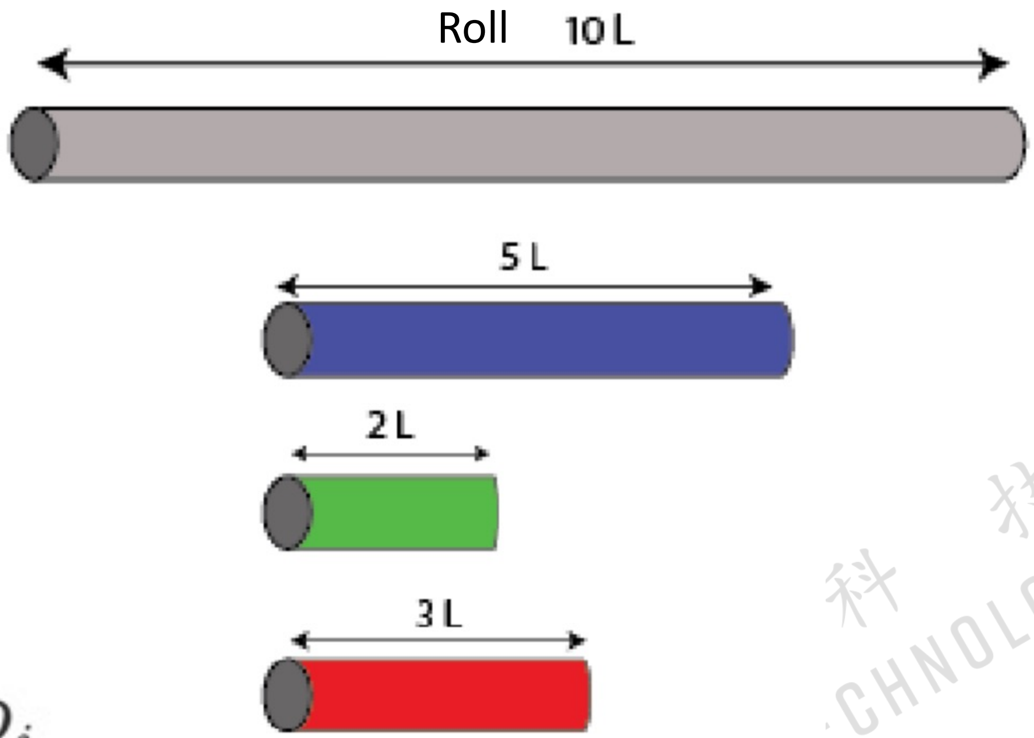
- Cutting larger-sized objects into smaller ones to meet a demand



https://himnickson.medium.com/one-dimensional-cutting-stock-problem-78b54dc839f4

Cutting stock problem

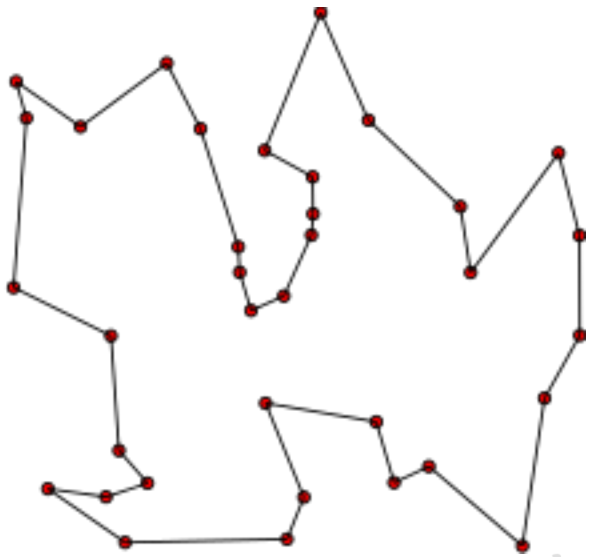
- Cutting larger-sized objects into smaller ones to meet a demand
- Y_i is a binary decision indicating if we use the big roll number *i*. A clear upper-bound for this problem is D
- X_{ij} is an integer giving the number of times we cut a small roll *j* in the big roll *i*
- Constraints
  - ✓ Demand satisfaction constraint: $$\sum_i X_{ij} \geq D_j$$
  - ✓ Roll size constraint: $$\sum_j X_{ij} \cdot W_j \leq L \cdot Y_i$$



https://matbesancon.github.io/post/2018-05-23-colgen/

Travelling salesman problem

- The travelling salesman problem (TSP) asks the following question:
  - ✓ "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"



$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j\neq i, j=1}^{n} c_{ij} x_{ij} :$$

$$x_{ij} \in \{0,1\} \qquad i,j = 1,\ldots,n;$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n;$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n;$$

https://en.wikipedia.org/wiki/Travelling_salesman_problem
https://medium.com/swlh/techniques-for-subtour-elimination-in-traveling-salesman-problem-theory-and-implementation-in-71942e0baf0c

TSP - MTZ

- Subtours elimination

$$
\begin{aligned}
\min \sum_{i=1}^{n} \sum_{j\neq i, j=1}^{n} c_{ij} x_{ij} &: \\
x_{ij} \in \{0,1\} & \qquad i,j = 1,\ldots,n; \\
u_i \in \mathbf{Z} & \qquad i = 2,\ldots,n; \\
\sum_{i=1, i\neq j}^{n} x_{ij} = 1 & \qquad j = 1,\ldots,n; \\
\sum_{j=1, j\neq i}^{n} x_{ij} = 1 & \qquad i = 1,\ldots,n; \\
u_i - u_j + n x_{ij} \le n-1 & \qquad 2 \le i \ne j \le n; \\
1 \le u_i \le n-1 & \qquad 2 \le i \le n.
\end{aligned}
$$

u_i be a dummy variable,

indicate tour ordering, such that $u_i < u_j$ implies city i is visited before city j.

https://en.wikipedia.org/wiki/Travelling_salesman_problem

Thanks

运筹学修炼日记：TSP中两种不同消除子环路的方法及callback实现（Python调用Gurobi求解，附以王者荣耀视角解读callback的工作逻辑）

原创 置顶 刘兴禄 2020-08-06 23:29:49 1934 收藏 23 版权

https://blog.csdn.net/HsinglukLiu/article/details/107848461

1 The Cutting Stock Problem

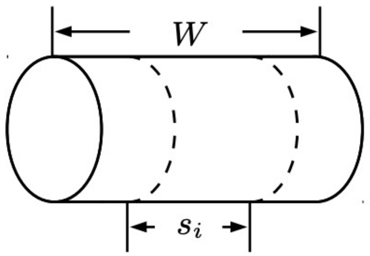


Figure 1: Raw

https://people.orie.cornell.edu/dpw/orie6300/Lectures/lec16.pdf

https://github.com/openstack-archive/deb-python-pulp/blob/master/examples/SpongeRollProblem4.py

Tutorial 10: Solving Cutting Stock Problem Using Column Generation Technique

GIAN Short Course on Optimization: Applications, Algorithms, and Computation

https://wiki.mcs.anl.gov/leyffer/images/b/bf/10a-tutorial.pdf