Project Planning : Tasks with Variable Duration

João Pedro Vasconcelos Teixeira
Computer Engineering
Informatics Center, UFPB
João Pessoa, Paraíba
Email: joaoteixeira@eng.ci.ufpb.br
Github: https://github.com/jpvt

João Wallace Lucena Lins
Computer Engineering
Informatics Center, UFPB
João Pessoa, Paraíba
Email: jwallace.lucena@gmail.com
Github: https://github.com/joallace

Itamar de Paiva Rocha Filho
Computer Engineering
Informatics Center, UFPB
João Pessoa, Paraíba
Email: itamardprf@gmail.com
GitHub: https://github.com/ItamarRocha

***Abstract*—In this article, we will discuss and expose the results of our second assignment of the Operations Research Course at the Federal University of Paraíba. In this project, we modeled and implemented a TSP-MTZ like problem involving tasks planning.**

I. Introduction

First of all, it is necessary to present the acronyms that will be used during this report.

OR	Operations research
TSP	Traveling Salesman Problem
MTZ	Miller–Tucker–Zemlin formulation of TSP

In this coursework, we had as a task to model with Integer Linear Programming a problem determined by the Professor. We implemented the model in C++ with the help of IBM's CPLEX.

II. Problem definition

Our assignment was to solve the following problem:

An engineering project consists of n tasks. When a certain task **j** can only be performed after a task **i**, we say that **i** precedes **j**. There is a table indicating the precedence that must be respected. Each task has a DN_j duration (in days) when performed normally.

However, it's possible to perform a task more quickly. In this case, its duration is reduced to DA_j, but there is an additional cost (CA_j). It is possible to further accelerate the execution of a task, reducing its duration to DMA_j, at the additional cost (CMA_j).

The project has a deadline, the **D-day**. For each day of delay, a penalty of **M** value is paid. The objective is to plan the execution of the project to minimize the total cost of penalties and tasks performed in an accelerated manner.

For example: n=6; **D**=100; **M**=2,000.

Task	DN	DA	CA	DMA	CMA
1	71	60	12000	40	24000
2	55	50	4000	44	5000
3	27	23	13000	17	17000
4	31	26	10000	20	19000
5	40	35	5000	30	7000
6	10	9	1000	8	6000

The precedences between tasks are:

Before	After
1	3
2	3
3	4
3	5
4	6
5	6

We have implemented the program to work taking any data set as an instance, as long as it follows the standards indicated in the repository's documentation. The program outputs the duration of each task, how it was performed, how much was paid in acceleration costs, penalties, and total cost.

III. Modeling

To solve our assignment, we implemented a model based on the **TSP-MTZ**, which is a formulation of the traditional **TSP**, because it works effectively for a small data set. In our modeling, we abstract the tasks as nodes, the order of execution as the arcs, the duration as the arcs distance and we kept the cost as itself.

And to make this possible we have to transform this task planning problem into a Hamiltonian cycle. In order to do so, we need to add an imaginary node with 0 cost and 0 duration that connects the "last" node to the imaginary and the imaginary to the "initial" node.

The data provided by the instances are the following:

Data

n = the total number of tasks.
D = deadline.
M = the penalty value per day.
SDN = the max duration (worst time case) in which the problem can be executed.
C = Matrix with the costs of each task i being executed in each mode k.
T = Matrix with the days taken by each task i being executed in each mode k.

Sets

J = {0, 1, 2, . . . , n}
$A = \{(i, j) : i \in J,\ j \in J,\ i \neq j\}$
P = set of arcs representing the precedences, such that $P \subset A$.
E = set of execution modes

To solve this problem, we choose to have 5 different variables type in the context. The variables will be defined below.

Variables

x_{ij} = boolean variable that defines if the arc i $\rightarrow$ j is active.
m_{ik} = boolean variable that defines if the task i used the mode k in its execution.
c_i = Integer variable that defines when did the task i finished.
α = Consists in the total time taken to execute all tasks.
β = Consists in the time that surpassed the deadline

Objective Function

$$\text{Min} \sum_{i \in J} \sum_{k \in E} C_{ik} * m_{ik} + \beta * M$$

Our Objective Function will aim to minimize the overall cost with penalties and additional payments.

Constraints

Here we will describe the constraints that we used to model our problem.

• **Linking constraints**

$$\sum_{j \in J} x_{ij} = 1\ ,\quad \forall\ i \in J$$

$$\sum_{i \in J} x_{ij} = 1\ ,\quad \forall\ j \in J$$

$$x_{ii} = 0\ ,\quad \forall\ i \in J$$

These two first constraints mean that each node should have only two connections, one being an input and the other an output. The last constraint states that no node can be connected to itself. In addition, we must connect the imaginary node to the first node and the end node to the imaginary node.

f = first node (taking into consideration the precedence)
e = last node (taking into consideration the precedence)

$$x_{0,f} = 1$$

$$x_{e,0} = 1$$

• **Cumulative finishing time**

We did the following constraint in order to assign the correct finishing time to each variable c. First we get the total days that were took to complete task j.

$$Days_j = \sum_{k \in E} m_{jk} * T_{j,k}$$

With this value in hand we can now establish the constraint.

$$c_j \geq c_i + Days_j - (1 - x_{ij}) * SDN\ ,\quad \forall\ (i, j) \in A$$

If x_{ij} is active, c_j will be lower bounded at the value of c_i + $Days_j$. Otherwise, it will be a redundant constraint, since we will have that c_j must be greater-equal than a big negative number.

• **Execution method constraint**

$$\sum_{k \in E} m_{ik} = 1\ ,\quad \forall\ i \in J$$

This constraint assures that each task will have only one execution method .

• **Precedence constraint**

$$c_j - c_i \geq 1\ ,\quad \forall\ (i, j) \in P$$

This constraint will assure that each i task is finished before a j task.

• **Alpha value**

$$\alpha = \sum_{i \in J} \sum_{k \in E} T_{ik} * m_{ik}$$

This constraint will serve to set the α value

• **Beta value**

$$\beta \geq \alpha - D$$

$$\beta \geq 0$$

These constraints will serve to set the β value

REFERENCES

[1] Ahuja R.K., Magnanti T.L., Orlin J.B. (2001) Maximum Flow Problem. In: Floudas C.A., Pardalos P.M. (eds) Encyclopedia of Optimization. Springer, Boston, MA

[2] Frederick S. Hillier, Gerald J. Lieberman - Introduction to Operations Research (2010, McGraw-Hill)