Networks and Flows on Graphs

Project Scheduling

Bashar Dudin

May 16, 2017

EPITA

What Is It About ?

A Project Scheduling Problem

We're confronted with a project scheduling (or planning) problem if, in order to achieve an *objective*, we need to go through a set of *tasks*, each having limited *ressources* and the set of which is subject to a number of *constraints*.

What Is It About ?

A Project Scheduling Problem

We're confronted with a project scheduling (or planning) problem if, in order to achieve an *objective*, we need to go through a set of *tasks*, each having limited *ressources* and the set of which is subject to a number of *constraints*.

- *Tasks* can only be defined once you know what you are dealing with, the hard part is to make full sense of how you decompose your project into tasks
- *Ressources* are generally thought of as the needed time for a task to be accomplished, it can also be funds, the number of needed employees, etc ...
- *Constraints* are grouped into three big types : *potential constraints*, *disjunctive constraints* and *cumulative constraints*.

Types of Constraints

Potential constraints

> These are *precedence constraints* and *constraints of temporal position*. The former correspond to constraints of type : “for task B to start task A has to be finished” ; these are about the order in which tasks take place. The constraints of temporal position correspond to constraints of type : “task A cannot start before January, 2”. Mathematically, they are modeled identically.

Disjunctive constraints

> These are constraints of type : “tasks A and B cannot happen simultaneously”.

Cumulative constraints

> These are relative to the evolution of resources needed to accomplish a task. They are generally hard to model.

Types of Constraints

Potential constraints

> These are *precedence constraints* and *constraints of temporal position*. The former correspond to constraints of type : “for task B to start task A has to be finished” ; these are about the order in which tasks take place. The constraints of temporal position correspond to constraints of type : “task A cannot start before January, 2”. Mathematically, they are modeled identically.

Disjunctive constraints

> These are constraints of type : “tasks A and B cannot happen simultaneously”.

Cumulative constraints

> These are relative to the evolution of resources needed to accomplish a task. They are generally hard to model.

What Are We Looking for ?

There are three things we're interested in when dealing with scheduling problems (assuming we only have potential constraints)

- earliest dates at which tasks can start

What Are We Looking for ?

There are three things we're interested in when dealing with scheduling problems (assuming we only have potential constraints)

- earliest dates at which tasks can start
- latest dates ; once the minimum date has been computed

What Are We Looking for ?

There are three things we're interested in when dealing with scheduling problems (assuming we only have potential constraints)

- earliest dates at which tasks can start
- latest dates ; once the minimum date has been computed
- margins, i.e. the difference between latest and earliest date of a task.

What Are We Looking for ?

There are three things we're interested in when dealing with scheduling problems (assuming we only have potential constraints)

- earliest dates at which tasks can start
- latest dates ; once the minimum date has been computed
- margins, i.e. the difference between latest and earliest date of a task.

We shall working two standard approaches aiming at answering these 3 related questions : the MPM method (the french method) and PERT method (the american one).

Scheduling methods

MPM (Meta Potential Method)

> In the MPM approach tasks are understood as vertices of a graph, whose arrows correspond to precedence constraints. Each arrow is valued by a number representing the time between the start of its origin till the start of its target.

Scheduling methods

MPM (Meta Potential Method)

> In the MPM approach tasks are understood as vertices of a graph, whose arrows correspond to precedence constraints. Each arrow is valued by a number representing the time between the start of its origin till the start of its target.

PERT (Program Evaluation and Review Technique)

> In the PERT case, the tasks are represented by arrows, while vertices should be understood as intermediate achievements attained once a task is completed. An arrow is valued by the time the corresponding task takes. Using a PERT diagram needs serious work on the chosen way of modeling the problem at hand. It's advantage in respect to the MPM approach is the fact it is better suited when randomness is involved.

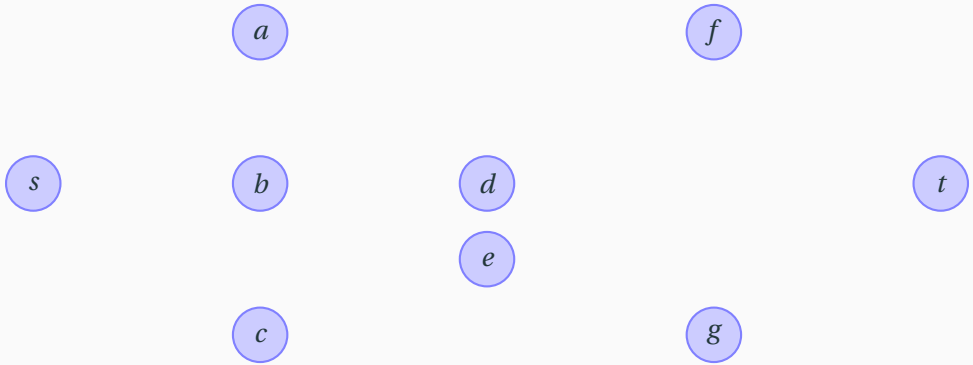
Working Example

Consider a project given by tasks a to g, under the following conditions of resource and constraints

Tasks	Needed Time	Constraints
a	6	
b	3	
c	6	
d	2	b accomplished
e	4	b accomplished
f	3	d and a accomplished
g	1	f, e, c accomplished

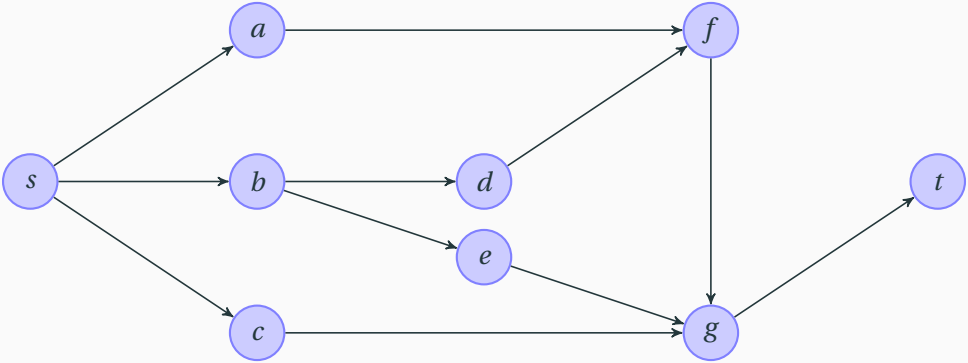
Working Example : MPM model

Each vertex corresponds to one of the tasks, s and t are start and end of project.



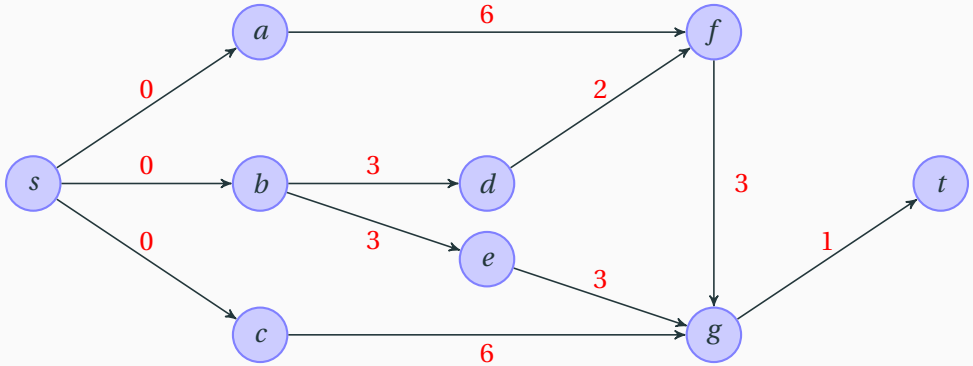
Working Example : MPM model

Arrows give a picture of the potential constraints satisfied by the tasks.



Working Example : MPM model

Valuation at arrow is the time laps between start point of origin till start point of target.



Working Example : MPM / Earliest Dates

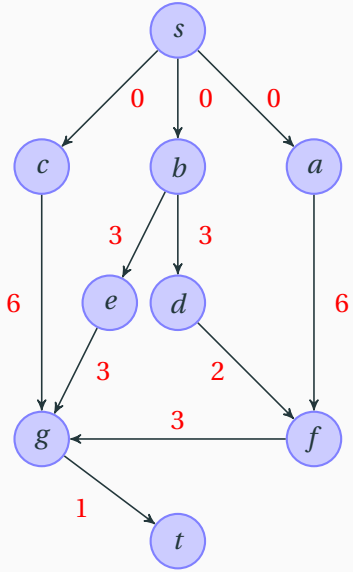
The point is to compute the earliest date at which the project ends.

This is governed by paths from source to target that take the longest time. These are called *critical paths*, the name comes from the fact any delay ε in tasks along this path would delay the whole project by ε.

Our goal can be achieved by adapting Ford's algorithms to look for longest paths in our graph.

Beware

For this strategy to work, it is important to have graphs without circuits! in our case that would mean there are contradicting tasks ...



Working Example : MPM / Earliest Dates

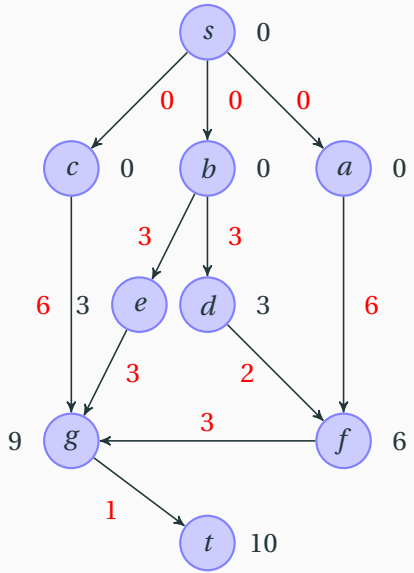
The point is to compute the earliest date at which the project ends.

This is governed by paths from source to target that take the longest time. These are called *critical paths*, the name comes from the fact any delay ε in tasks along this path would delay the whole project by ε.

Our goal can be achieved by adapting Ford's algorithms to look for longest paths in our graph.

Beware

For this strategy to work, it is important to have graphs without circuits! in our case that would mean there are contradicting tasks ...

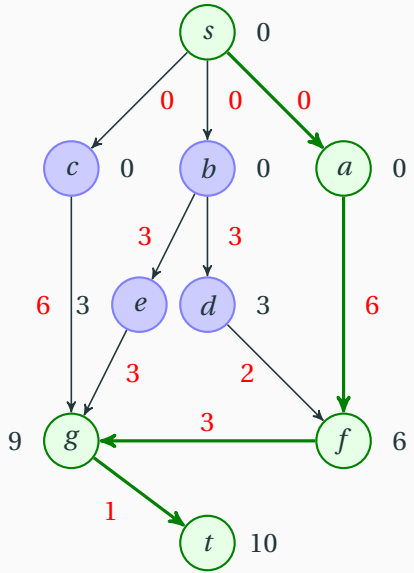


Working Example : MPM / Earliest Dates

The earliest date at which to expect our project is accomplished is 10 time units from our starting date.

More generally, we have computed the earliest date at which to expect any task to start. The start t which correspond to the end of the project starts at 10.

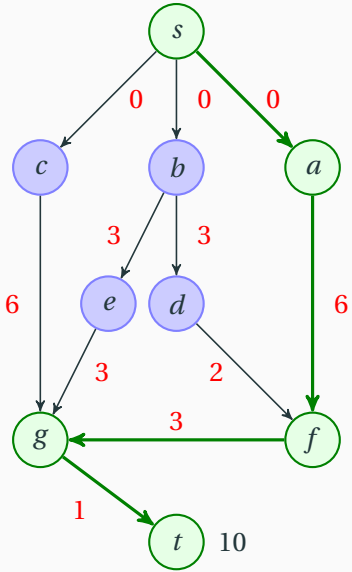
All *critical paths* from source to target are the (elementary) ones contained in the subgraph in green. There is only one here.



Working Example : MPM / Latest Dates

Knowing that the project, at the earliest, ends 10 time units after it starts, we can compute the latest dates at which non-critical tasks can start.

To do so, one looks for lengths of *shortest* paths starting at t with valuation 10.



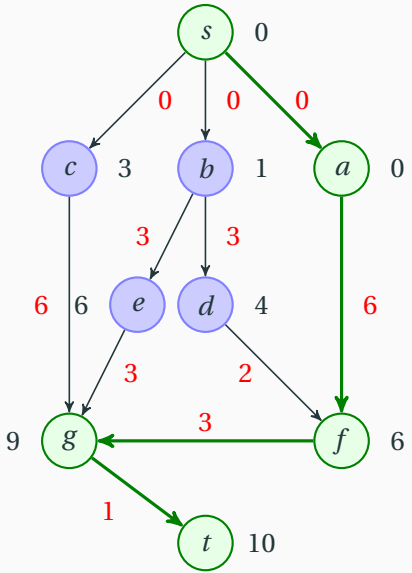
Working Example : MPM / Latest Dates

Knowing that the project, at the earliest, ends 10 time units after it starts, we can compute the latest dates at which non-critical tasks can start.

To do so, one looks for lengths of *shortest* paths starting at t with valuation 10.

We get the graph on the right. Each vertex is valued by the latest date at which it can start, keeping the project ending time at 10.

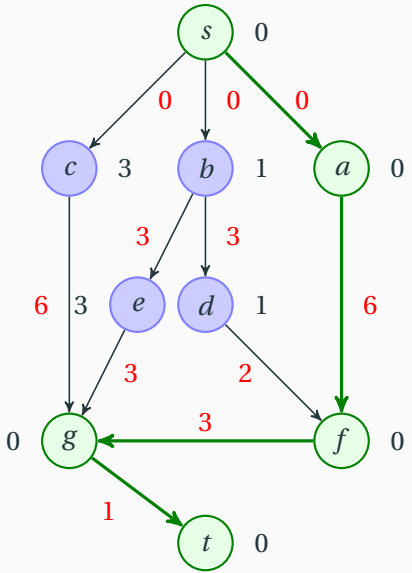
Notice that valuation of the critical path do not change.



Working Example : MPM / Margins

The margin at a vertex is simply the difference between the latest date to start and the earliest one.

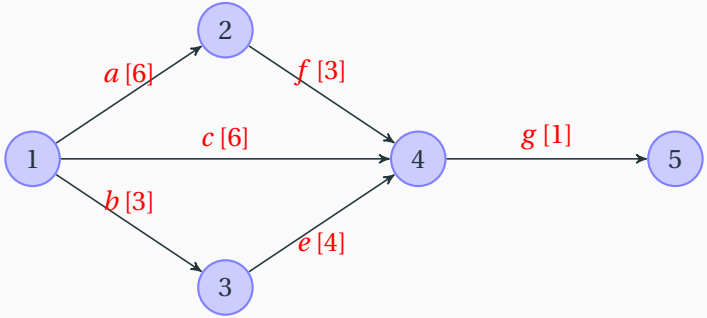
Notice that along the critical path, margins are 0.



Working Example : The PERT approach

Recall that in the PERT approach, tasks give rise to arrows valued by the time a task needs to be accomplished. Vertices correspond to intermediate objectives ; defining them comes, at first, from the concrete situation at hand.

In our case, starting from a vertex 5 corresponding to project end, we can define vertex 4 as f, c, e are accomplished etc ... This is arguably a “mathematician” example.



This is it for the course on Graphs, Networks and Flows.
Good luck with your final exams!