Basic Frontline mathematics

Work in progress, last updated on 20/10/2022 by Ding Ruiqi.

1 Simulation (Disentanglement Algorithm)

1.1 Core components and their relationships

Note:

- P6 stands for Oracle Primavera P6 EPPM.
- Only mathematically relevant properties are listed. (properties such as ID and names are not listed)

1.1.1 Tasks

Let t_i be tasks. Each task has the following properties:

property	P6 code	P6 name	type	note
S	status_code	Activity Status	user input	
t_{d0}	target_drtn_hr_cnt	Planned Duration	user input	
t_{as}	act_start_date	Actual Start	user input	
t_{ae}	act_end_date	Actual Finish	user input	
t_{dL}			constraint	lower bond for t_d
t_{dU}			constraint	upper bond for t_d
t_d	target_drtn_hr_cnt	Planned Duration	optimized	
t_s	target_start_date	Planned Start	derived	
t_e	target_end_date	Planned Finish	derived	

The status can be:

P6 code	P6 name
TK_NotStart	NOT STARTED
TK_Active	IN PROGRESS
TK_Complete	COMPLETED

- If status is NOT STARTED

  t_s is calculated based on predecessors, see Task-Task relationship below

$$t_e = t_s + t_d$$

- If status is IN PROGRESS

$$t_s = t_{as}$$

$$t_e = t_s + t_d$$

- If status is COMPLETED

$$t_s = t_{as}$$

$$t_e = t_{ae}$$

1.1.2 Task-Task relationship

Let P_{ij} be the Task-Task relationship between t_i and t_j, where t_i is the predecessor of t_j. Each relationship has the following properties:

property	P6 code	P6 name	type	note
T	pred_type	Relationship Type	user input	
t_l	lag_hr_cnt	Lag	user input	

The relationship type can be:

P6 code	**P6 name**
PR_FS	FINISH START
PR_FF	FINISH FINISH
PR_SS	START START
PR_SF	START FINISH

Note: In the following, $(t_s^j)_i$ represents the start time for task j calculated based on **one of** its predecessors t_i.

- If relationship type is FINISH START
$$(t_s^j)_i = t_e^i + t_l$$
- If relationship type is IN FINISH FINISH
$$(t_s^j)_i = t_e^i + t_l - t_d^j$$
- If relationship type is START START
$$(t_s^j)_i = t_s^i + t_l$$
- If relationship type is START FINISH
$$(t_s^j)_i = t_s^i + t_l - t_d^j$$

One task t_j can have **many different predecessors** t_i, but it must have **one unique start time**, we therefore choose the maximum of them all:

$$t_s^j = \max_i\,(t_s^j)_i$$
$$t_e^j = t_s^j + t_d^j$$

As one can see, in order to find the start/end time for task j, the start/end time for all of its predecessor tasks must be known. But these predecessors each have their own predecessors! All these relationships form a complex network of tasks.

How can we ensure that for each task we try to compute, the properties of its predecessors are already available? And how can we do this efficiently? (If we define a *simulation* as finding out all of the properties for each task, then it requires many iterations of *simulation* to achieve *optimization*.)

Luckily, we have come up with an algorithm called **disentanglement algorithm**, that helps us to "see" this messy network of tasks clearly. It answers the questions above and ensures us **correct** and **ultra-fast** simulation of the project.

1.1.3 Resources

Let r_m be resources. Each resource has the following properties:

property	**P6 code**	**P6 name**	**type**	**note**
T	rsrc_type	Resource Type	user input	
hU			constraint	upper bond for $h(t)$
$h(t)$			derived	resource histogram

The resource type can be:

P6 code	**P6 name**
RT_Labor	LABOR
RT_Mat	MATERIAL

The resource histogram for one resource is summed over tasks using this resource:

$$h(t) = \sum_{\text{tasks using this resource}} h_i(t)$$

1.1.4 Task-Resource relationship

Let R_{im} be the Task-Resource relationship between t_i and r_m, where t_i has r_m as one of its resources. Each relationship has the following properties:

property	P6 code	P6 name	type	note
q_0	target_qty_per_hr	Planned Units / Time	user input	
q	target_qty_per_hr	Planned Units / Time	derived	

$$q = \frac{t_{d0}}{t_d} q_0$$

If $\frac{t_{d0}}{t_d} = 2$, that means we want to shorten the duration by half, thus we need to double the resources: $q = 2q_0$

The resource histogram corresponding to t_i and r_m is therefore

$$h_i(t) = \begin{cases} q & \text{if } t_s < t < t_e \\ 0 & \text{otherwise} \end{cases}$$

This ensures that the **amount of labor work** and **amount of material usage** is fixed, as they depend on the nature of the task, instead of how the task is performed. By increasing the amount of resource q, we can shorten the duration of a task t_d, but we cannot magically reduce the amount of labor work / material usage needed $q \times t_d$.

$$\int_t h_i(t) = q \times t_d = \frac{t_{d0}}{t_d} q_0 \times t_d = t_{d0} q_0$$

1.2 Computational loop

- The variables we are trying to optimize are the durations of each task t_d
- The start/end dates t_s, t_e for each task are determined by t_d and the following:
  - the statuses of the tasks
  - the predecessor relationships among the tasks
  - the actual start/end dates of the tasks
- The quantity of resources q are determined by t_d
- The resource histograms $h(t)$ are determined by q and t_s, t_e
- With t_s, t_e for each task and $h(t)$ for each resource, we can define various **loss functions** for optimization
- The **loss functions** drive updates to t_d and completes the loop

Note: All of the equations above seem simple, and indeed they are, the true difficulty / complexity comes from the complex relationship among the network of tasks / resources. This is similar to **Artificial Neural Networks** that powers modern AI, where each neuron performs extremely simple computations, yet a complex network of them demonstrates almost *magical* computation ability.