Constraint Model to deal with Process Alternatives

H. Simonis

February 11, 2025

Abstract

This document describes the constraints needed to deal with a generalized scheduling model that allows for alternative processes, bill of materials, intermediate products, and process step choices.

1 Model

1.1 Notation

We use the following sets in the model.

O set of orders, may contain a single order

J set of of optional jobs to satisfy the orders

T set of all optional tasks needed to perform the jobs

T_o set of all tasks linked to order o

DM set of disjunctive resources

CM set of cumulative resources

RM set of raw materials considered

1.2 Variables

The model is expressed with *task variables*, which represent the structure of a task starting at a time point, running until its end, with a variable duration. The task may of may not be present in the solution, which is expressed with a binary `present` variable. We add a preference field which tells us what is our preference if that task is present in the solution.

z_k task variable for order k

y_{jk} task variable for job j of order k, there can be multiple, optional jobs for an order; only one will be selected.

x_{ij} task variable for task i of job j

v_{im} task variable for machineTask of task i on machine m

All task variables have fields

start start of task

end end of task

duration duration of task

present boolean flag that indicates of the task is present in schedule

preference integer preference value to use in the objective

1.3 Constraints

Our constraint model consist of a number of different types, each responsible for some aspect of the model. We discuss the available types in the following paragraphs.

In this model we assume that orders must be satisfied, we have to produce the order with one of the possible processes for the product made.

1.3.1 Constraining Order Dates

In the scenario considered, orders define a release date and a due date. The release date states that no activity of the order processing can start before this date, and the due date states that no processing for the order can be performed after the due date. The start and end of the order processing are expressed by the task variable of the order. This leads to the following inequalities.

$$\forall_{k \in O}: \quad z_k.\text{end} \leq \text{due}_k \tag{1}$$

$$\forall_{k \in O}: \quad z_k.\text{start} \geq \text{release}_k \tag{2}$$

Often, it is not possible to satisfy both release and due dates at the same time, depending on the context we may be able to relax one or the other date (but rarely both dates). The *tardiness* of an order is the amount of time that the order is late. In many cases the tardiness becomes part of the objective function.

We often also have a constraint on the *earliness* of the order. We do not want to complete the order too early, since the customer may not want to accept their order at this earlier date, and/or we have to carry the inventory cost of the product made for the earliness period. We constrain the earliness to be less than a maximal value c. Alternatively, the earliness cost may become part of the objective function.

$$\forall_{k \in O}: \quad z_k.\text{end} \geq \text{due}_k - c \tag{3}$$

This constraint can be relaxed by increasing the value of c, allowing an earlier completion time for the order.

1.3.2 Temporal Relations between Tasks

We now describe the different temporal relations between tasks that we model. We consider two tasks for the same job j, indexed with i_1 and i_2.

End Before Start The easiest and most common constraints is that a second task can only start once the first task is finished.

$$\text{endBeforeStart}(t_{i_1,j}, t_{i_2,j}) \tag{4}$$

$$t_{i_2,j}.\text{start} \geq t_{i_1,j}.\text{end} \tag{5}$$

It is very likely that this is a hard constraint, that cannot be relaxed.

Min Wait In some cases we have to wait for a specific time c after the end of the first task before we can start the second task. This might for example be due to a curing time for some paint, or a settling time in a tank to achieve the correct conditions for further processing.

$$\text{minWait}(t_{i_1,j}, t_{i_2,j}, c) \tag{6}$$

A default hard limit for c would be zero, the second task cannot start before the end of the first task.

$$t_{i_2,j}.\text{start} \geq t_{i_1,j}.\text{end} + c \tag{7}$$

We might be able to relax this constraint to some extent by shortening the waiting time c. This might incur a small quality penalty, or may require more careful handling of the item. Often, we may have a hard and a soft limit, where the soft limit is a preferred value, while that hard limit cannot be relaxed further.

Max Wait We can also constrain the maximal wait between two tasks of a given process. We impose that the start of the second task must be within a fixed time c of the end of the first task. This might be due to product properties, or may be imposed to limit the buffer space that is required between process steps.

$$\text{maxWait}(t_{i_1,j}, t_{i_2,j}, c) \tag{8}$$

$$t_{i_2,j}.\text{start} \leq t_{i_1,j}.\text{end} + c \tag{9}$$

Very often the maximal waiting time c is not a hard constraint, the time can be extended to some extend with a small quality penality.

Start Before Start A more specialized constraint states that the start of the second task must be after the start of the first task, but it is allowed that both tasks run concurrently. We encounter this constraint in one model of the SALBP (Simple Assembly Line Balancing Problem), but they are also common in project management for construction problems.

$$\text{startBeforeStart}(t_{i_1,j}, t_{i_2,j}) \tag{10}$$

$$t_{i_2,j}.\text{start} \geq t_{i_1,j}.\text{start} \tag{11}$$

It seems unlikely that this constraint can be relaxed.

NoWait In this case, the second task must start as soon as the first task finishes. This can be due to a missing buffer between the tasks, or may be due to a material constraint, e.g. the product is a food item which would spoil if it has to wait. As soon as the first task is finished, the product must vacate the first machine, and start processing on the second machine.

$$\text{noWait}(t_{i_1,j}, t_{i_2,j}) \tag{12}$$

$$t_{i_2,j}.\text{start} = t_{i_1,j}.\text{end} \tag{13}$$

Whether we can relax this constraint depend on the detail of the process, the relaxation would take the form of a `maxWait` constraint.

Blocking In this relation there is no buffer between the machines of the two tasks, but we allow the first task to stay on its machine until the machine for the second task becomes available. The end of the first task is equal to the start of the second task, but we can extend the duration of the first task to achieve this.

$$\text{blocking}(t_{i_1,j}, t_{i_2,j}) \tag{14}$$

$$t_{i_2,j}.\text{start} = t_{i_1,j}.\text{end} \tag{15}$$

In the blocking case, the duration of the first task might extend beyond the duration given by the process data, we typically need another variable to model this distinction.

1.3.3 Linking Tasks to Children

Connecting orders to multiple, alternative processes In our model we assume that each order must be satisfied within the scheduling horizon. That means that exactly one of the possible processes for the product must be selected and must be scheduled. This is expressed with an `alternative` constraint.

$$\texttt{alternative}(z_k, [y_{j,k} | \text{job } j \text{ is an alternative process for order } k]) \tag{16}$$

This constraint cannot be relaxed in the model, as long as we require that each order must be satisfied. We may have a preference value for each of the

alternative processes, the preference cost of the order is the preference value of the selected process.

Linking Jobs and their Tasks For each process have have one optional job, consisting of the tasks that are defined as elements of that process. The duration of the job is defined by the earliest and latest activities that are included. The tasks must be present if and only if the job is present. This condition is expressed with a `span` constraint. It links the `present` flags of the job to the `present` flags of the tasks, and forces the start of the job task to be the minimum of the start dates of the tasks, and the end of the job to be the maximum of the end dates of the tasks.

$$\texttt{span}(y_{jk}, [t_{ij} | \text{task } i \text{ belongs to job } j]) \tag{17}$$

This constraint typically cannot be relaxed, it is structural to the problem.

Linking Alternative Tasks to their Choices The process definition may provide alternative tasks which are defined by a number of choices. In these case the alternative task is selected if and only if exactly one of the choices is selected, the start and end of the alternative task are defined by the start and end of the selected choice.

$$\texttt{alternative}(t_{i,j}, [t_{i'j} | t' \text{ is a choice for alternative task } i]) \tag{18}$$

We typically have a preference value for each choice, and the cost of the alternative is the preference value of the selected choice.

Machine Choice As part of the basic process model we allow machines choices for specific process steps. This is the same concept as for alternative process steps, but limited only to a choice of the machine. In our model the task variables v_{im} describe the choice of machine m for task i, which is presented by task variable x_{ij}. Only at most one of the alternatives can be active, the start, end and duration of task x is determined by the active task v_{im}. If x is not active, none of the alternatives can be active.

$$\texttt{alternative}(x_{ij}, [v_{im} | m \text{ is a machine on which task } \ i \text{ can run}]) \tag{19}$$

1.3.4 Resource Constraints

Disjunctive Resources A *disjunctive* resource can handle only one task at a time, and each task is running without interruption from its start to its end, blocking the machine.

$$\forall_{m \in DM}: \quad \texttt{disjunctive}([t | \text{task } t \text{ requires machine } m \text{ and is present}]) \tag{20}$$

Cumulative Resources A *cumulative* resource can process more than one activity at a time, each task uses a certain amount of resource u_t while it is running, while the overall amount of resource available at each timepoint is given by cap_m.

$$\forall_{m \in CM}: \quad \texttt{cumulative}([t| \text{ task } t \text{ uses capacity } u_t \text{ of resource } m], \text{cap}_m) \quad (21)$$

There are many different ways to relax the cumulative constraint, for example to allow exceeding the resource limit for some period of time, or to some level. We may also decide to relax this constraint altogether, e.g. not to include manpower limitations expressed with a cumulative constraint in our relaxation of the problem.

1.3.5 Raw Material Constraints

The bill of material (BoM) for a process defines which raw materials are required in which quantity at the start of a process step. The constraint is that for every time point t the quantity in stock at the start plus all raw material deliveries up to that time minus all raw material used up to time t must be greater or equal to the defined safety stock level, i.e. we cannot use more raw materials than we have received up to a given time point.

$$\forall_{r \in RM} \forall t \in T: \quad s_r + \sum_{t' \leq t} d_{rt'} - \sum_{\{x \in T | x.\text{start} \leq t\}} x.\text{present} * u_{rx} \geq \text{safety}_r \quad (22)$$

This condition will typically be expressed as a single `cumulative` constraint. We may relax the constraint completely, ignoring the raw material, or we can relax the safety stock level to some extent.

1.3.6 Calendars

In many cases, the plant, or specific machines within a plant, do not run continuously 24/7. A typical example would be plant that runs for three shifts on each working day, starting with a morning shift on Monday 06:00, and ending the third, night shift of Friday on Saturday morning at 06:00. Some machines in the plant might be only running for two shifts, due to a lack of demand, or to save operations cost. These conditions are expressed with calendar constraints, which describe the working pattern for each resource for periods of time. The shift pattern may change over the year, e.g. extra shifts are run to deal with a demand peak before Christmas, or the factory is closed, or operates at a lower rate during holidays. A relaxation of the calendar constraint often is possible, but must be planned in advance, and/or may require a change for a longer period (a complete month or quarter of a year for example). Typically, only very specific relaxations are allowed. We may for example add a third shift, if a resource is running only two shifts, or we may add a Saturday day shift, but only if we are already running three shifts on weekdays.

1.3.7 Overall or Task Specific Preferences

The different preference values for jobs and tasks describe how satisfied we are with using one of these alternatives. As part of the user input constraints, we

may ask for specific preferences to be satisfied. This can either be a global condition, e.g. we only allow the first preference for all alternatives, or order/task specific, e.g. this specific order is important, and we should only use the best preference choices for its execution. We assume the preference values to be totally ordered, so that we can express our preference limit by inequalities

The overall constraint takes thee form:

$$\forall_{t \in T}: \quad t.\text{present} * t.\text{preference} \leq \text{limit} \tag{23}$$

For a specific order o, we can impose a preference constraint on all tasks T_o that belong to that order, and enforce an order specific preference value limit_o.

$$\forall_{t \in T(o)}: \quad t.\text{present} * t.\text{preference} \leq \text{limit}_o \tag{24}$$

Finally we can specify a constraint for selected tasks t only, enforcing a task specific preference limit_t.

$$t.\text{present} * t.\text{preference} \leq \text{limit}_t \tag{25}$$

These preference choices can be easily relaxed, if there is no solution with the preferred value.

1.4 Objective

The typical objective of our model is a weighted sum of different cost terms, typically defined by the preference values of the choices taken.

$$\begin{aligned} \min \\ & \alpha_1 \sum y_{jk}.\text{pref} * y_{jk}.\text{present} + \\ & \alpha_2 \sum x_{ij}.\text{pref} * x_{ij}.\text{present} + \\ & \alpha_3 \sum v_{im}.\text{pref} * v_{im}.\text{present} \end{aligned} \tag{26}$$