SimUSanté Instances Generator

Simon CAILLARD

15 janvier 2020

This document presents SimUProb instances (SimUSanté Instances) used in [1]. They are generated from the classic instances of the Curriculum Based-Course TimeTabling problem (CB-CTT) [2]. Part 1 of this document describes the instances of CB-CTT, SimUProb and gives the different steps to transform the first one into the second one. Part 2 presents the different criterion used to obtain various SimUSanté instances from a CB-CTT instance.

1 CB-CTT to SimUsanté instances

1.1 CB-CTT instances

SimUSanté problem differs from CB-CTT mainly in terms of soft constraints, and then this part of CB-CTT instances are not considered by our generation method. Only, resources, classes, teachers, events and time slots are used.

A CB-CTT instance is structured as follows :

— T' is a set of time slots.
— C is a set of classes, and each $c \in C$ follows a set of events Ω_c. $\bigcup_{c \in C} \Omega_c = \Omega$.
— Each event $\omega \in \Omega$ has a duration $duration_\omega$ and a preassigned teacher.
— Each event ω belongs to a course K.
— $R^{r'}$ is a finite set of rooms.
— $R^{e'}$ is a finite set of teachers. Each teacher is pre-assigned to one event.

1.2 Transformation process

From a CB-CTT instance named CB, this process aims to generate an initial SimUProb instance $CB - D_0T_0C_0A_0$. This process is divided in two phases. The first one, explained in subsection 1.2.1, splits or merges some events of CB. Indeed, in SimUSanté problem, the duration of activities are mainly between 2 and 4 time slots and then, events of CB with duration less than 2 time slots are grouped, while those with duration greater than 4 time slots are splited. These changes are made by function $splitAndMerge()$ described in algorithm 1. The second phase, as shown in subsection 1.2.2, generates $CB - D_0T_0C_0A_0$ by adding the elements specific to SimUSanté problem, like break time slots, resource types, etc.

1.2.1 First phase : function $splitAndMerge$

From a set of events Ω, $splitAndMerge()$ selects each event ω with duration greater or equal to 4 and split them in two sub events ω' and ω'' with almost the same duration, such that $duration_{\omega} = duration_{\omega'} + duration_{\omega''}$. A precedence relation is then added between ω' and ω''.

In addition, events with duration of 1 time slot are merged two by two. When there is only one remaining event ω of duration 1, this one is merged with a randomly chosen event $\omega' \neq \omega$, of duration fewer than 4. The merge is done by function $fusion()$. From a couple of events (ω, ω'), $fusion()$ returns an event ω'' whose duration $duration_{\omega''} = duration_{\omega} + duration_{\omega'}$. A pre-assigned teacher is then chosen from that of events ω and ω' to be assigned to event ω'', ditto for rooms.

Algorithm 1 : splitAndMerge

Require: Ω the set of events from a CB-CTT instance
Ensure: Ω^{new} the set of events after transformation

```
Ω^new ← ∅
Ω_1 ← ∅
for all ω ∈ Ω do
  pred_ω ← ∅
  if duration_ω > 4 then
    ω' ← ω
    ω'' ← ω
    duration_ω' ← ⌈duration_ω / 2⌉
    duration_ω'' ← duration_ω − duration_ω'
    pred_ω'' ← {ω'}
    Ω^new ← Ω^new ∪ {ω', ω''}
  else
    if duration_ω = 1 then
      Ω_1 ← Ω_1 ∪ ω
    else
      Ω^new ← Ω^new ∪ ω
    end if
  end if
end for
while Ω_1 ≠ ∅ do
  ω' ← random(Ω_1)
  Ω_1 ← Ω_1 \ {ω'}
  if Ω_1 ≠ ∅ then
    ω'' ← random(Ω_1)
    Ω_1 ← Ω_1 \ {ω"}
    Ω^new ← Ω^new ∪ fusion(ω', ω'')
  else
    ω'' ← random(Ω^new)
    Ω^new ← Ω^new \ {ω''}
    Ω^new ← Ω^new ∪ fusion(ω', ω'')
  end if
end while
return Ω^new
```

1.2.2 Second phase : Generation process

From a CB-CTT instance CB, an initial SimUProb instance $CB{-}D_0T_0C_0A_0$ is generated. This instance remains as close as possible to the original one but adds all characteristics relative to SimUProb and it is constructed as follow :

— There are $D = \left\lceil \frac{|T'|}{8} \right\rceil$ days. Each day is composed by $8+1$ time slots (one break time slot is added in our instances). Then, horizon H is a set of $|D| \times 9$ time slots.

- S is the set of sessions, then $S = C$. Each class c corresponds to a session s.
- $\forall s \in S$, $A_s = \Omega_c^{new}$ and $\bigcup_{s \in S} A_s = A$. Each event ω coincides with an activity a, then $duration_a = duration_\omega$.
- $\forall (a, a') \in A^2$ such that the corresponding events $(\omega, \omega') \in \Omega^2$ are in a same course K, we generate a precedence relation between a and a'.
- $R^r = R^{r'}$ is the set of rooms. There is only one type of rooms λ_{r_1} and $\forall a \in A$, $qtreq^a_{\lambda_{r_1}} = 1$. $\Lambda^r = \{\lambda_{r_1}\}$ represents the set of types associated to room r, and Λ^R is the set of room types.
- $R^e = R^{e'}$ is the set of employees and $\forall e \in R^e$, we define a type of employee λ_e which corresponds to the skills of employee e. $\Lambda^e = \{\lambda_e\}$ represents the set of types associated to employee e and $\Lambda^E = \sum_{e \in E} \lambda_e$ is the set of employee types. $\forall \omega \in \Omega^{new}$, if event ω was pre-assigned to employee e, then corresponding activity $a \in A$ is associated to employee type λ_e and $qtreq^a_{\lambda_e} = 1$. $dispo_e$ represents the disponibilities of employee e and $\forall e \in E$, $dispo_e = H$.

2 Criterion variation

From $CB - D_0T_0C_0A_0$, a set of SimUProb instances are generated, varying the following criteria : availability of employees (D_1), types of rooms (T_1, T_2), types of employees (C_1) and activity requirements (A_1, A_2).

These criteria are combined to provide different instances. As an illustration, $D_0T_0C_0A_0 + D_1$ provides a new instance $D_1T_0C_0A_0$ and $D_1T_0C_0A_1 + T_1$ gives $D_1T_1C_0A_1$, etc. It should be noticed that some criteria are dependent to other ones. For example, criterion T_1 adds news room types and criterion A_1 uses these types. The details of these different criteria are described in the following subsections.

In subsections 2.1, 2.2, 2.3 and 2.4, function $selection(i, S)$ returns a set of i% randomly chosen elements from a set S of elements.

2.1 Criterion C

This criteria concerns the skills (types) of employees. The aim is to add one skill to a set of 20% of randomly chosen employees denoted E_{20}. In addition, with a probability of 20%, each employee $e \in E_{20}$ has an additional skill. Algorithm 2 describes function Criterion$C$.

Algorithm 2 : Criterion$C$

Require: R^E
  $E_{20} \leftarrow selection(20, R^e)$
  for all $e \in E_{20}$ **do**
    $firstSkill \leftarrow random(\Lambda^E \setminus \Lambda^e)$
    $\Lambda^e \leftarrow \Lambda^e \cup firstSkill$
    $r \leftarrow random(0, 1)$
    if $r \geq 0,8$ **then**
      $secondSkill \leftarrow random(\Lambda^E \setminus \Lambda^e)$
      $\Lambda^e \leftarrow \Lambda^e \cup secondSkill$
    end if
  end for

2.2 Criterion D

In the basic transformed instance $D_0T_0C_0A_0$, employees are avalaible over all the horizon H (full disponibilities). Criterion D_1 aims to reduce avalaibilities of 20% of randomly selected employees. Their patterns of disponibilities are randomly replaced by one of those below :

$pattern_1^H$ **:** the first $\lceil \frac{|H|}{2} \rceil$ time slots.
$pattern_2^H$ **:** the five first time slots of each day $d \in D$ (morning).
$pattern_3^H$ **:** the last $\lceil \frac{|H|}{2} \rceil$ time slots.
$pattern_4^H$ **:** the four last time slots of each day $d \in D$ (afternoon).

Algorithm 3 gives the details of function criterion$D$.

Algorithm 3 : Criterion$D$

Require: R^e, H
  $E_{20} \leftarrow selection(20, R^e)$
  for all $e \in E_{20}$ **do**
    $dispo_e \leftarrow random(\{pattern_1^H, pattern_2^H, pattern_3^H, pattern_4^H\})$
  end for

2.3 Criterion T

In version T_1, two new room types λ_{r_2} and λ_{r_3} are added. 15% of randomly selected rooms replace their type by λ_{r_3} and 35% by λ_{r_2}. It should be noticed that each room is associated to only one type of room. Room types requirements for activities are set accordingly.

In version T_2, 10% of randomly selected rooms add a second type of room from Λ^R.

Algorithms 4 and 5 show respectively the details of function Criterion$T1$ and Criterion$T2$.

Algorithm 4 : Criterion$T1$

Require: R^r, the set of rooms, Λ^R, the set of room types.
  $\Lambda^R \leftarrow \Lambda^R \cup \{\lambda_{r_2}, \lambda_{r_3}\}$
  $R_{15} \leftarrow selection(15, R^r)$
  $R_{35} \leftarrow selection(35, R^r \setminus R_{15})$
  $A_{15} \leftarrow selection(15, A)$
  $A_{35} \leftarrow selection(35, A \setminus A_{15})$
  for all $r \in R_{15}$ **do**
    $Lambda^r \leftarrow \{\lambda_{r_3}\}$
  end for
  for all $r \in R_{30}$ **do**
    $Lambda^r \leftarrow \{\lambda_{r_2}\}$
  end for
  for all $a \in A_{15}$ **do**
    $\Lambda^a \leftarrow \Lambda^a \setminus \lambda_{r_1}$
    $qtreq^a_{\lambda_{r_1}} \leftarrow 0$
    $\Lambda^a \leftarrow \Lambda^a \cup \{\lambda_{r_3}\}$
    $qtreq^a_{\lambda_{r_3}} \leftarrow 1$
  end for
  for all $a \in A_{30}$ **do**
    $\Lambda^a \leftarrow \Lambda^a \setminus \lambda_{r_1}$
    $qtreq^a_{\lambda_{r_1}} \leftarrow 0$
    $\Lambda^a \leftarrow \Lambda^a \cup \{\lambda_{r_2}\}$
    $qtreq^a_{\lambda_{r_2}} \leftarrow 1$
  end for

Algorithm 5 : Criterion$T2$

Require: R, the set of rooms.
  $R_{10} \leftarrow selection(10, R^r)$
  for all $r \in R_{10}$ **do**
    $\Lambda^r \leftarrow Lambda^r \cup random(\Lambda^R \setminus \Lambda^r)$
  end for

2.4 Criterion A

Let A_{10} the set of 10% randomly chosen activities which will be modified by criterion A_1 and A_2. $\forall a \in A_{10}$, Λ^a represents the set of type requirements (employee and room) of activity a.

In version A_1, each activity $a \in A_{10}$ has a probability of 50% to increase by one the quantity of its actual room types requirement (i.e. $\forall \lambda_r \in \Lambda^a \cap \Lambda^R$, $qtreq^a_{\lambda_r} \leftarrow qtreq^a_{\lambda_r} + 1$), or to add a new room type requirement $\lambda_{r'}$, with $\lambda_{r'} \notin \Lambda^a$. ($qtreq^a_{\lambda_{r'}} \leftarrow 1$)

Version A_2 modifies employee types requirements of each activity of A_{10}. The modification process is the same as that of used by A_1.

We note in A_1 and A_2 that the maximal resource type quantity required by an activity can't exceed the quantity of resources associated to this type.

Algorithms 6 and 7 show respectively function CriterionA_1 and CriterionA_2.

Algorithm 6 : CriterionA_1

Require: A_{10}, a set of randomly pre-chosen activities
  for all $a \in A_{10}$ **do**
    $rand \leftarrow random(0, 1)$
    if $rand < 0.5$ **then**
      for all $\lambda_r \in \Lambda^a \cap \Lambda^R$ **do**
        $qtreq^a_{\lambda_r} \leftarrow qtreq^a_{\lambda_r} + 1$
      end for
    else
      $\lambda_{r'} \leftarrow random(\Lambda^R \setminus \Lambda^a)$
      $qtreq^a_{\lambda_{r'}} \leftarrow qtreq^a_{\lambda_{r'}} + 1$
    end if
  end for

Algorithm 7 : CriterionA_2

Require: A_{10}, a set of randomly pre-chosen activities
  for all $a \in A_{10}$ **do**
    $rand \leftarrow random(0, 1)$
    if $rand < 0.5$ **then**
      for all $\lambda_e \in \Lambda^a \cap \Lambda^E$ **do**
        $qtreq^a_{\lambda_e} \leftarrow qtreq^a_{\lambda_e} + 1$
      end for
    else
      $\lambda_{e'} \leftarrow random(\Lambda^E \setminus \Lambda^a)$
      $qtreq^a_{\lambda_{e'}} \leftarrow qtreq^a_{\lambda_{e'}} + 1$
    end if
  end for

Références

[1] S. Caillard, L. Devendeville, and C. Lucet. A Planning Problem with Resource Constraints in Health Simulation Center. In *Optimization of Complex Systems*. Springer, 2020.

[2] EEMCS DMMP Group University of Twente. High School Timetabling Project. https ://www.utwente.nl/en/eemcs/dmmp/hstt/.