Local Search Algorithm to Solve a Scheduling Problem in Healthcare Training Center*

Simon Caillard[1,2], Laure Brisoux Devendeville[1] and Corinne Lucet[1]

[1] Laboratoire MIS (EA 4290), Université de Picardie Jules Verne
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
{simon.caillard, laure.devendeville, corinne.lucet}@u-picardie.fr
[2] Health Simulation Center SimUSanté®
Amiens University Hospital, France
simon.caillard@chu-amiens.fr

Keywords: Scheduling, Local search, Healthcare training, Timetabling.

1 Introduction

SimUSanté, located in Amiens, France is one of the biggest healthcare training center in Europe. All kinds of health actors: professionals, patients, students use this center and can meet and train together by simulating medical acts in various fields of healthcare but also attending regular courses, for a total of more than 500 different formations. The problem faced by SimUSanté is a scheduling problem that consists in planning a set of training sessions respecting a set of time and resource constraints.

Scheduling problems are NP-Complete (Cooper, T.B. and Kingston, J.H. 1995). SimUSanté's problem belongs to this family of problems and specifically to the Curriculum-Based Courses Timetabling Problem (CB-CTT)(Di Gaspero L. *et. al.* 2007) which consists in finding the best weekly assignment for university lectures, available rooms and time periods for a set of classes under a set of hard and soft constraints. However, some features of SimUSanté's problem differ from CB-CTT ones, such as resources types, skills and precedence constraints required for activities, lunch break management, and objective function. Another way would be to consider CB-CTT as a variant of the Resource-Constrained Project Scheduling Problem (RCPSP)(Brucker, P. AND Knust, S. 2001). In this case, we need to add the followings constraints : some activities cannot be planned in parallel and each resource can have more than one type.

We present in this paper a local search algorithm *SimuLS*, based on dedicated neighborhood operators to solve SimUSanté's problem. We generated adequate instances[3] inspired by those used in CB-CTT. We then compared the results obtained by SimuLS with those worked out by the mathematical model implemented in CPLEX and a dedicated greedy algorithm *SimuG* (Caillard S. *et. al.* 2020).

The paper is organized as follows: in section 2, we briefly formalize the scheduling problem encountered by SimUSanté and describe how a solution is evaluated. In section 3 we present our local search algorithm *SimuLS* and give the different operators used in order to explore the search space. Section 4 provides computational results. Finally, section 5 concludes this paper and presents some perspectives.

2 Formalization and evaluation

The problem encountered by SimUSanté is to schedule a set of training sessions S over a determined period T. A training session $s \in S$ is composed by a set of activities A_s.

* This project is supported by region Hauts-de-France and Health Simulation Center SimUSanté

[3] SimUSanté instances available on: https://mis.u-picardie.fr/Benchmarks-GOC

$A = \bigcup_{s \in S} UA_s$ represents the set of all activities. Activity $a \in A$ has a specific duration and requires different types and quantities of resources. Activities can be linked by precedence constraints. In addition, there is a set of resources R which is composed by employees, rooms and materials. Each resource $r \in R$ is associated to one or more types of resources. For example a room can have both meeting room and classroom types.

Solution Sol is a set of triplets (a, t_a, R_a) where $a \in A$ is an activity, $t_a \in T$ the starting time slot of a, and $R_a \subseteq R$ the set of avalaible resources assigned to a, from t_a and for its total duration $duration_a$. The set of scheduled activities is denoted $SA = \{a | (a, t_a, R_a) \in Sol\}$, with $(SA \subseteq A)$. The set of unscheduled activities is denoted $UA = A \setminus SA$. For session $s \in S$, $Sol_s \subseteq Sol$, represents the set of triplets of the solution related to s, with $Sol_s = \{(a, t_a, R_a) \in Sol | a \in A_s\}$. $SA_s = SA \cap A_s$, is the set of scheduled activities of s, and $UA_s = A_s \setminus SA_s$, the set of unscheduled activities of s.

For a given session $s \in S$, if at least one activity has been scheduled $(SA_s \neq \emptyset)$, start date $t_{start_s} = \min\{t_a, a \in SA_s\}$, and end date $t_{end_s} = \max\{t_a + duration_a, a \in SA_s\}$ allow to compute the corresponding makespan $mk_s = t_{end_s} - t_{start_s}$ of session s. If no activity has been scheduled $(SA = \emptyset)$, then $mk_s = 0$.

The evaluation of Sol, denoted $Makespan(Sol)$, is the sum of the makespans of all sessions, plus the amount of unplanned activities, multiplied by penalty α (see equation 1). The objective is to find a valid solution with a minimum $Makespan$.

$$Makespan(Sol) = \sum_{s \in S} mk_s + |UA| \times \alpha \tag{1}$$

3 Local search algorithm: SimuLS

SimuLS is a local search algorithm that explores the solution space by applying neighborhood operators, starting from a solution provided by a greedy algorithm, *SimuG* (Caillard S. *et. al.* 2020). For a maximum preset *limitCounter* iterations, *SimuLS* relies on *saturator* operator to plan unscheduled activities and when it is not possible, it uses several operators: *intra*, *extra* and $extra^+$. Each of these operators checks possible movements and applies one. If the best solution ever met is not improved after a preset *noImprov* iterations, a part of the solution is destroyed by the *destructor* operator, in order to escape from a local minimum.

A *movement* is caracterized by a couple $< (a, t_a, R_a)\ ;\ \Upsilon >$. (a, t_a, R_a) represents a triplet that will be added to the current solution, with $a \in UA$, an unscheduled activity, $t_a \in T$, a time slot from which a could be started, and R_a, the set of resources assigned to a , that exactly matches its resources requirement. In order to plan a, we need to remove a set Υ of triplets from the solution. $\Upsilon = \{(b_1, t_{b_1}, R_{b_1}), \ldots, (b_n, t_{b_n}, R_{b_n})\}$, $n \in \{1, \ldots |Sol|\}$. The set of resources R_a can be composed by resources directly available over T, plus thoses released by canceling all activities of Υ. A *movement* respects all operational rules and resources constraints.

The choice of a movement by an operator relies on criteria such as makespan mk_s of impacted sessions, global makespan $Makespan$, the number of canceled activities, etc. The different operators present in *SimuLS* are:

$saturator_s$: This operator tends to place an unscheduled activity $a \in UA_s$ without changing the current solution. It builds a set of movements M:$\{< (a, t_a^1, R_a); \emptyset >, \ldots, < (a, t_a^k, R_a); \emptyset >\}$ so that for each movement $< (a, t_a^i, R_a)\ ;\ \emptyset > \in M$, with $i \in [1; k]$, $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[= \emptyset \quad \forall (b, t_b, R_b) \in Sol_s$.

$intra_s$: This operator removes one or more scheduled activities from session s in order to plan an unscheduled activity $a \in UA_s$. It builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol_s$, the following properties are verified:

– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[\neq \emptyset, \forall(b, t_b, R_b) \in \Upsilon$
– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[= \emptyset, \forall(b, t_b, R_b) \in \{Sol_s \setminus \Upsilon\}$

$extra_s$**:** This operator removes one or more scheduled activities from a randomly selected session $s' \neq s$, in order to plan an unscheduled activity $a \in UA_s$. It builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol_{s'}$, the following properties are verified:
– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[\neq \emptyset, \forall(b, t_b, R_b) \in \Upsilon$
– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[= \emptyset, \forall(b, t_b, R_b) \in Sol_s$

$extra_s^+$**:** This operator is an extension of $extra_s$. The canceled activities can belong to a set of sessions $\{s'_1, \ldots, s'_k\} \subseteq S$. For activity $a \in UA_s$, it builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol$, the properties below are verified :
– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[\neq \emptyset, \forall(b, t_b, R_b) \in \Upsilon$
– $[t_a^i; t_a^i + duration_a[\cap[t_b; t_b + duration_b[= \emptyset, \forall(b, t_b, R_b) \in \{Sol_s \setminus \Upsilon\}$

$destructor$**:** This operator destroys a part of current solution Sol. It builds and applies a set of movements M:$\{< \emptyset\ ;\ \{(a_1, t_{a_1}, R_{a_1}\} >, \ldots, < \emptyset\ ;\ \{(a_k, t_{a_k}, R_{a_k}\} >$ so that $\forall i \in [1; k]$, $(a_i, t_{a_i}, R_{a_i}) \in Sol$ represents the triplet that will be removed from the Sol.

Algorithm 1 : SimuLS

Input: Sol (the current solution), S (set of sessions), $\forall s \in S, UA_s$ (set of unscheduled activities for session s), $UA = \bigcup_{s \in S} UA_s$ (the set of unscheduled activities), $noImprov$, $limitCounter$
$noBest \leftarrow 0$
$counter \leftarrow 0$
$Sol \leftarrow saturator(UA)$
$bestSol \leftarrow Sol$
while $counter < limitCounter$ **do**
  if $(noBest = noImprov)$ **then**
    $Sol \leftarrow destructor(Sol)$
    $noBest \leftarrow 0$
  end if
  if $UA \neq \emptyset$ **then**
    $a \leftarrow random(UA)$
    $s \leftarrow (s/a \in UA_s)$
    $Sol \leftarrow selectOperator(\{intra, extra, extra^+\}, s, a)$
  end if
  $Sol \leftarrow saturator(UA)$
  if $Makespan(Sol) < Makespan(bestSol)$ **then**
    $noBest \leftarrow 0$
    $bestSol \leftarrow Sol$
  else
    $noBest \leftarrow noBest + 1$
  end if
  $counter \leftarrow counter + 1$
end while

In order to choose which operator to apply between $intra$, $extra$, $extra^+$, $SimuLS$ uses the $SelectOperator$ function that uses two specific counters c_{extra}^s and $c_{extra^+}^a$. The first one, c_{extra}^s, counts the number of times where $intra$ have been consecutively applied to session s. The second one counts how many times activity a remained consecutively unscheduled. By default, operator $intra$ is applied, except whenever one of these counters reaches a preset limit, $selectOperator$ then activates the operator that corresponds to the counter. In case of equality between the two counters, $extra^+$ is always used first.

4 Experimental study

A mathematical model has been implemented under CPLEX. It provides optimal results for small instances with a running time of two hours or more. Table 4 presents the comparison between CPLEX, *SimuG* and *SimuLS* on SimUSanté instances. Penalty α is set to $|T|$. *SimuLS* was implemented in Java, on an Intel i7 7500U processor. The time used to find solutions is always less than 1 second for the greedy algorithm *SimuG* and less than 1 minute for *SimuLS*. The numbers in parentheses after some if the results, represent the amount of unscheduled activities.

Table 1. Results for Brazil1 and Italy1 instances

Instance Brazil1				Instance Italy1			
Instance	**cplex**	**SimuG**	**SimuLS**	**Instance**	**cplex**	**SimuG**	**SimuLS**
$D_0T_0C_0A_0$	81	86	83	$D_0T_0C_0A_0$	101	105	102
$D_0T_0C_1A_0$	81	87	82	$D_0T_0C_1A_0$	101	104	101
$D_0T_1C_0A_1$	94	232 (4)	110	$D_0T_1C_0A_1$	107	150 (1)	116
$D_0T_1C_1A_1$	94	232 (4)	108	$D_0T_1C_1A_1$	107	187 (2)	115
$D_1T_0C_0A_0$	81	89	85	$D_1T_0C_0A_0$	101	104	104
$D_1T_0C_1A_0$	81	94	90	$D_1T_0C_1A_0$	101	104	104
$D_1T_1C_0A_1$	96	161 (2)	107	$D_1T_1C_0A_1$	107	150 (2)	114
$D_1T_1C_1A_1$	96	166 (2)	110	$D_1T_1C_1A_1$	107	180 (3)	115

Columns *cplex*, $SimUG$ and $SimULS$ represent respectively the optimums, the results of greedy algorithm and those of the local search algorithm. By the nature of a greedy algorithm, SimUG cannot scheduled all activities (see instances $D_0T_1C_0A_1$, $D_0T_1C_1A_1$, $D_1T_1C_0A_1$, $D_1T_1C_1A_1$). In this case, a penalty α is applied, and the corresponding score is rising up to 246% from optimality. In contrast, *Cplex* and $SimULG$ always schedule all activities. $SimULG$ reaches optimality for Italy$-D_0T_0C_1A_0$ instance, and always improves the results obtained by the greedy algorithm. The gap with optimality is less than 18%.

5 Conclusion

In this paper we have briefly introduced SimUSanté's problem and proposed a local search algorithm *SimuLS* to solve it. *SimuLS* is based on five neighborhood operators dedicated to SimUSanté's problem. Four of them allow to schedule activities but only one without modify solution. The last operator destroys the solution in order to escape from a local minimum. *SimuLS* is experimented on instances from CB-CTT, adapted to the SimUSanté's problem. The results obtained are compared to the optimal solutions provided by CPLEX. Contrary to *SimuG*, all activities are scheduled by *SimuLS*. It is a first step towards building an efficient metaheuristic to solve SimUSanté's problem.

References

Brucker, P. AND Knust, S., 2001, "Resource-Constrained Project Scheduling and Timetabling", *Springer*, Burke E., Erben W. (eds) Practice and Theory of Automated Timetabling III.

Caillard S., Brisoux-Devendeville L., and Lucet C., 2020, "A Planning Problem with Resource Constraints in Health Simulation Center", *Springer*, Le Thi H., Le H., Pham Dinh T. (eds) Optimization of Complex Systems: Theory, Models, Algorithms and Applications.

Cooper, T.B. and Kingston, J.H., 1995, "The complexity of timetable construction problems", *Springer*, Burke E., Ross P. (eds) Practice and Theory of Automated Timetabling.

Di Gaspero, L. and McCollum, B. and Schaerf, A., 2007, "Curriculum-based CTT - Technical Report", *The Second Int. Timetabling Competition*

Schaerf A., 1999, "A Survey of Automated Timetabling", *Artificial Intelligence Review*, Vol. 13, pp. 87-127.