An ACO Algorithm for a Scheduling Problem in Health Simulation Center

Simon Caillard, Corinne Lucet, Laure Brisoux Devendeville

An ACO algorithm for a scheduling problem in health simulation center

Simon Caillard, Corinne Lucet, and Laure Brisoux-Devendeville

Laboratoire MIS (UR 4290), Université de Picardie Jules Verne
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
{simon.caillard, corinne.lucet, laure.devendeville}@u-picardie.fr

Abstract. SimUSanté is one of the biggest European simulating and training centers, proposing training sessions for all involed in healthcare: professionals, students, patients. This paper presents the timetabling problem encountered by SimUSanté with regard to the quality objectives and the time and resource constraints. To solve it, *SimUACO-LS* which is the hybridization of the Min-Max Ants Colony Optimization algorithm *SimUACO* with the variable neighborhood search *SimULS* [3], is presented. *SimULS*, *SimUACO* and *SimUACO-LS* are compared in a set of representative instances [2], newly generated and derived from those of the Curriculum-Based Course Timetabling problem [1]. *SimUACO-LS* always improves both results of *SimULS* and *SimUACO* by respectively 3.84% and 2.97%.

Keywords: Scheduling · Healthcare training · Timetabling · Operational research · Optimization

1 Introduction

SimUSanté, located in Amiens, France, is the biggest european healthcare simulation and training center that provides more than 400 different training sessions in a wide range of healthcare areas. Everyone involved in healthcare can meet up in training sessions and learn together. A training session corresponds to a set of activities followed by a group of learners. Such teaching requires equipment and classrooms that are very specific to hospital activities and building a timetable that take into account all these resource constraints is essential for the smooth-running of the center. The aim of SimuSanté is then to schedule a maximum number of activities while maximizing the timetable compactness of each session to schedule. So, in this paper, we focus on the Curriculum-Based Course Timetabling problem (CB-CTT) [6] which is NP-Hard [5] and has similarities with the SimuSanté problem. Nevertheless, the SimUSanté problem differs in a major point: the precedence relationships between activities. In addition, 10 out of the 13 of the CB-CTT constraints are hard in the SimUSanté problem. Moreover in the literature, the solvers of CB-CTT instances, allow to violate hard constraints, whereas this is prohibited in the SimUSanté problem. Nevertheless, CB-CTT remains the academic problem closest to that of SimUSanté.

Over the last ten years, CB-CTT has been widely studied, as can be seen in many surveys [10, 13]. The best known methods for solving it are the local search methods and population-based meta-heuristics. The local search methods are based on neighborhood operators that start from an initial solution and modify a part of it to switch to another one. In addition, there are a few ways to avoid being trapped in a local minimum. The population-based methods use many solutions at the same time and combine them to obtain better ones. The best known population based methods to solve timetabling are the Ants Colony Optimization (ACO) [12] and the Genetic Algorithms (GA) [8]. Although genetic algorithms are efficient, most of the solutions obtained require an important repairing phase after crossover operator which potentially leads to remove the activities that violate hard constraints. In regards with the objectives of SimU-Santé, ACO algorithms that always produce feasible solutions seem to be better suited.

Nevertheless, such methods have the drawback of converging too early in the search process towards a local optimum. For these reasons, hybrid methods which combine both approaches are often used [9, 11]. So, in this paper, we propose *SimUACO-LS* algorithm that is a combination of $SimUACO$, a Max-Min Ant Colony Optimization (MMACO) system [14], with a Variable Neighborhood Search (VNS) $SimULS$ [3]. ACO algorithms are efficient in exploring the search space solutions, but the quality of this search relies on the quality of the heuristic information used to guide it. We have therefore developed several specialized heuristics, which are not redundant with those used in $SimULS$, to lead ants in building solutions. Although these heuristics fit the SimuSanté problem, they can be applied to similar problems, with the same hard constraints by adapting the heuristics according to the desired qualitative characteristic. *SimUACO-LS* have been tested on various instances with results close to optimum solutions.

This paper is organized as follows: in section 2, we formalize both the SimU-santé problem and the structure of the solutions. Section 3 gives the graph representation related to the formalization of the problem as an ACO problem and details our algorithm *SimUACO-LS*. In section 4, the results provided by *SimUACO-LS* are compared to optimal results given by a mathematical model implemented with CPLEX [4]. Also, the relevance of the combination of a MMACO system with a VNS algorithm is assessed by comparing *SimUACO-LS* with $SimULS$ and $SimUACO$. Finally, section 5 concludes this paper and presents some perspectives.

2 The SimUSanté problem

2.1 Data definitions

An instance is composed of a set S of training sessions to schedule over horizon H, defined by a set of working days D. Each day $d \in D$ is composed of a set T_d of 9 time slots of one hour and has a starting time slot $start_d$, an ending one end_d and a set $break_d \subset T_d$ of potential time slots for the lunch breaks.

Each session $s \in S$ is a subset of activities that must be scheduled in serie. Any scheduled session must have one lunch break per day. A session s can be partially scheduled, i.e. some of its activities are unplanned in the final solution. The makespan of a session s is the number of time slots from the start of the first activity of s to the end of the last activity of s, including lunch break.

A represents the set of all instance activities and s_a is the session to which activity a belongs. $\forall a \in A$, $duration_a$ is the number of consecutive time slots required to execute a. Some precedence constraints exist between activities and $pred_a$ denotes the subset of activities that must be completed before a starts.

The set of resources R represents both employees and rooms and Λ is the set of resource types. A resource type corresponds to a skill for employees and to a specific equipment or characteristic for rooms. $\forall \lambda \in \Lambda$, we denote R^λ the set of resources of type λ, and $qtav_\lambda^t$ its associated quantity of resources available at time slot t. To each resource $r \in R$, $\Lambda_r \subseteq \Lambda$ denotes the set of types associated to r, because employees may have several skills and rooms may have different equipments. The availability of resource r is given by $isavailable_r^t$ which is equal to 1 if r is available at time slot t and 0 otherwise. If resource r is assigned to activity a, r must be available over $duration_a$. Moreover, all employee timetables must have one lunch break per day.

In order to be scheduled, activity a needs specific resources over its entire duration. $qtreq_\lambda^a$ is the quantity of resources of type $\lambda \in \Lambda$ required by a and $\Lambda_a = \{\lambda \in \Lambda | qtreq_\lambda^a \neq 0\}$ is the set of resource types required by a. The eligibility of activity a to time slot t is given by $iseligible_a^t$ which is equal to 1 if there are enough resources for scheduling a at t, 0 otherwise.

Solving an instance of the SimuSanté problem consists in scheduling as many activities as possible of A, while minimizing the sum of the makespans of the sessions of S.

2.2 Solution and evaluation

To build a solution, it is necessary to assign a start date t_a (a time slot) and a set of resources $c_a \subseteq R$ to each scheduled activity a. Then, a solution Sol is a set of triplets $\{(a_1, t_1, c_1), \dots (a_i, t_i, c_i)\}$, where $a_i \in A$, $t_i \in H$ and $c_i \subseteq R$. A triplet (a, t, c) can be added to a solution if it satisfies all constraints of resources (c perfectly matches the resource requirements of a), of precedence and of operating rules. We denote $UA = A \setminus \{a | (a, t, c) \in Sol\}$, the set of unscheduled activities and $Sol_s = \{(a, t, c) \in Sol | a \in s\}$, the set of triplets associated to s.

Once a solution is built, its evaluation relies on two criteria: its compactness and the number of scheduled activities. The makespan $mk_s = t_{end_s} - t_{start_s}$ represents the total duration of scheduled activities of the session s, with respectively $t_{start_s} = \min_{(a,t,c) \in Sol_s}\{t\}$ and $t_{end_s} = \max_{(a,t,c) \in Sol_s}\{t + duration_a\}$, the starting time slot and the ending one of session s. We note that if $Sol_s = \emptyset$, then $mk_s = 0$.

The evaluation of solution Sol denoted $Eval(Sol)$, is established with the sum of the mk_s and the sum of penalties α given to each unplanned activity (see

equation 1). The SimUSanté problem aims to construct a valid solution Sol^* such that $Eval(Sol^*)$ is minimum.

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

3 An ant colony optimization algorithm: *SimUACO-LS*

An ant colony optimization algorithm (ACO) [7] is a bio-inspired algorithm which uses ant behavior to find food and return back to the nest. Each ant leaves pheromones on the trail from their nest to the food source. An ant moves randomly but when it detects pheromones, it follows the trail and reinforces it by leaving additional pheromones. The more ants follow a trail, the more attractive that trail becomes. Pheromones evaporate over time and therefore the least used or slower paths become the least attractive. Ants can then find the fastest trail from the nest to a source of food. A Max-Min Ant Colony Optimization (MMACO) system is derive from the ACO system. Its principle, as described in [14], is to maintain pheromones between a minimum and a maximum value in order to keep accessible a maximum of relevant paths.

We propose to solve SimUSanté problem by combining a MMACO system with a VNS algorithm [3]. Thus, the first step is to turn the problem into a path search in a graph. Because of the complexity of the SimUSanté problem, a solution will be defined as a collection of elementary paths rather than a simple path as detailed below.

3.1 Graph representation

The SimUSanté problem can be modeled by a graph $\mathscr{G} = ((A, H, R), E)$. A is the set of activities, H and R are respectively those of time slots and resources. $E = \{A \times H\} \cup \{(t, r) | t \in H, r \in R \text{ and } isavailable_r^t = 1\} \cup \{R \times R\}$. A triplet (a, t, c) of a valid solution described in section 2.2 is represented by the path $(a, t, r_1, r_2, \ldots, r_k)$ with $c = \{r_1, r_2, \ldots, r_k\} \subseteq R$. For more convenience, we also denote it (a, t, c). There are different ways of scheduling a given activity a, so there are different valid paths (a, t^i, c^i). Each ant k builds a collection $\Pi^k = ((a_1, t_1, c_1), \ldots, (a_n, t_n, c_n))$, with $n \leq |A|$, of paths corresponding to a valid solution Sol^k of the problem. Once, path (a_x, t_x, c_x) is constructed, ant k is guided by a heuristic, as explained in the section 3.2, to jump from R to A in order to choose the next $a_{x+1} \in A$.

3.2 *SimUACO-LS* principle

In *SimUACO-LS*, each ant builds a valid solution by constructing its collection of paths. During this construction process, *selectActivities*() chooses the next activity a to plan, while *selectPath*() schedules it at time slot t with resources c.

Both rely on random proportionality rules which is based only on heuristic information for *selectActivities* and based on heuristic and pheromone information for *selectPath*. Once a valid solution is built, the variable neighborhood search *SimULS* (see section 3.5) is applied to it with a probability of $\beta\%$.

The pheromone information is stored in τ, a tridimensional matrix of size $|A| \times |H| \times |R|$ such that $\tau_{a,t,r}$ is the pheromone assigned to the allocation of the resource r to activity a, at time slot t. To retrieve the pheromone information of a resource combination c assigned to an activity a at a time slot t, we simply sum the amount of pheromones of each resource involved in this combination, i.e: $\tau(a,t,c) = \sum_{r \in c} \tau_{a,t,r}$

The update pheromones phase occurs at the end of each iteration, when all ants have constructed their own solution. All the pheromones of the matrix τ are first evaporated with coefficient γ ($0 < \gamma \leq 1$): $\tau_{a,t,r} = \tau_{a,t,r} \times (1 - \gamma)$. Next all components $(a,t,c) \in Sol_{best}$ are rewarded: $\forall r \in c$, $\tau_{a,t,r} = \tau_{a,t,r} + 1/Eval(Sol_{best})$

The pheromones represent the ant learning to find better solutions by exploring the search space around the best known solution while heuristic information is an essential knowledge to guide ants to promising areas of the search space. In *SimUACO* different heuristics are used to select both activities and resources. The following section describes these heuristics and the overall process used to guide an ant to schedule activities. An ant tries to schedule each activity only once . Let UA^k be the set of not visited activities by the ant k, i.e. the unscheduled activities in solution of ant k.

3.3 Choose the next activity: *selectActivities*()

The aim of *selectActivities*() is to choose in UA^k the next activity to schedule for ant k according to a random probability rule $next_a^k$, described in equation 2. It is based on the heuristic information $poss_a$ which represents an estimation of the remaining scheduling possibilities of a over H. This estimation considers the number of times there is $duration_a$ consecutive time slots with enough resources to schedule a.

$$next_a^k = \frac{\frac{1}{poss_a}}{\sum_{a' \in UA^k} \frac{1}{poss'_a}} \tag{2}$$

The objective of this random proportionality rule is twofold: lead ants in selecting one of the more promising activities to schedule and allow a diversification of the search.

3.4 Choose time slot and resources: *selectPath*()

From a given vertex activity a, *selectPath*() chooses a path from a, through a time slot vertex t to a resource combination vertex c. This selection is made according to the random proportionality rule $p(a,t,c)$, described at equation 3, which is based on the heuristic information $\eta(a,t,c)$ and the pheromone $\tau(a,t,c)$.

$$p(a,t,c) = \frac{\tau(a,t,c) \times \eta(a,t,c)}{\sum\limits_{(a,t^i,c^i)} \tau(a,t^i,c^i) \times \eta(a,t^i,c^i)} \tag{3}$$

The higher the value of the heuristic information and the related pheromone, the more profitable a path becomes. Note that we only consider possible paths which respect all the constraints of the SimUSanté problem with regard to the current state of the solution construction process.

The heuristic information $\eta(a,t,c)$, see equation 4, is based on three indicators: the makespan variation $\Delta(a,t,c)$, the number of idle time slots $Idle(a,t,c)$ and the remaining resources from t to $t + duration_a$ and $Usage(a,t,c)$, if the path (a,t,c) is chosen.

$$\eta(a,t,c) = \frac{1}{(1+\Delta(a,t,c)) \times (1+Idle(a,t,c)) \times (1+Usage(a,t,c))} \tag{4}$$

$\Delta(a,t,c)$ computes $Nmk_{s_a} - mk_{s_a}$, the difference between the new makespan $Nmk_{s_a} = \max_{t' \in \{t+duration_a, t_{end_{s_a}}\}}\{t'\} - \min_{t'' \in \{t, t_{start_{s_a}}\}}\{t''\}$ of session s_a, if a is scheduled at time slot t with the set of resources c, and its current makespan mk_{s_a}.

$Idle(a,t,c)$ counts the number of free time slots unusable if path (a,t,c) is selected. A set of contiguous free time slots is considered as unusable if it is not possible to plan any unscheduled activities of s_a over it, because the duration of each of them exceeds the quantity of free time slots in the set.

$Usage(a,t,c)$ is computed by equation 5. Over the interval $[t; t+duration_a[$, it adds up the quantities of resources remaining in $\Lambda_c = \bigcup_{r \in c} \Lambda_r$, if the path (a,t,c) is chosen, with the aim of minimizing the use of more than one type resources. Let us note that $\Lambda_a \subseteq \Lambda_c$.

$$Usage(a,t,c) = \sum_{t' \in [t; t+duration_a[} \left(\sum_{\lambda \in \Lambda_c} \left(\sum_{r \in R^\lambda} isavailable_r^{t'} \right) - qtreq_\lambda^a \right) \tag{5}$$

3.5 A Variable Neighborhood Search : *SimULS*

The VNS algorithm *SimULS* [3] is randomly applied with a rate of $\beta\%$ to the solutions constructed by ants, with the aim of improving them, and also to diversify the population. To proceed, it relies on the operators *saturator*, *intra*, *extra*, and $extra^+$. These operators build a set of movements which individually schedules an activity by removing one or more scheduled activities of the current solution. The difference between these operators is the target session from which activities are removed. Thus, they define different neighborhoods.

Moreover, to escape from a local minimum, *SimULS* regularly uses *diversificator* to destroy a part of the current solution. The condition to use

Table 1. Result averages between $SimULS$, $SimUACO$ and $SimUACO\text{-}LS$

Instance family	$SimULS$		$SimUACO$		$SimUACO\text{-}LS$		CPLEX
	$Eval$	SD	$Eval$	SD	$Eval$	SD	
$Brazil1$	87.10	1.68	86.14	1.75	85.01	1.79	83.6
$Italy1$	109.54	1.77	108.15	1.86	106.03	1.88	102.8
$Brazil2$	179.10	1.79	179.04	4.20	172.74	4.09	165.6
$Finland1$	364.30	1.34	354.36	4.95	342.55	4.86	na
$Brazil6$	430.09	1.01	429.14	4.94	409.49	4.81	na
$Finland2$	382.08	1.42	380.5	3.91	366.58	3.94	na
$StPaul$	1427.48	1.03	1427.18	5.42	1402.14	5.20	na

$diversificator$ is reached when all activities are scheduled or when there is no improvement during a preset number of iterations. $diversificator$ removes scheduled activities according to two criteria. The first one is the position of the activity in its session. The second one is the number of resources with multiple type assigned to the activity.

4 Experimental Results

In this section, we test $SimUACO\text{-}LS$, the combination of $SimUACO$ and $SimULS$ on instances close to the SimUSanté problem. These instances are based on those of the CB-CTT problem [1] and have been modified to include characteristics inherent in the SimUSanté problem. They vary according to the following characteristics: D_1, the availabilities of employees, C_1, the skills of employees, T_1 and T_2, the types of the rooms, A_1 and A_2 the requirements of activities. For each of the followings CB-CTT instances: $Brazil1$, $Italy1$, $Brazil2$, $Brazil6$, $FinlandHighSchool$ ($Finland1$), $FinlandSecondarySchool$ ($Finland2$) and $StPaul$, we generated an instance family which represents a set of 16 instances varying the previously presented criteria [2].

$SimUACO$, $SimULS$ and $SimUACO\text{-}LS$ were written in Java and tested on an Intel i7 8700K processor with a running time limit of two hours. We implemented the mathematical model [4] with CPLEX solver (version 12.6) and set a time limit of 7200 seconds. First three columns $SimUACO$, $SimULS$ and $SimUACO\text{-}LS$ in table 1 are split into sub-columns $Eval$ and SD. $Eval$ corresponds to the average of the objective function (equation 1, with $\alpha = |H|$) computed over all instances of family and over 20 runs for each of them. SD corresponds to average of the standard deviation. On the CPLEX column are averages on optimal results when available, or na otherwise. Figure 1 is another view of these results (without CPLEX ones) where the score of $SimUACO\text{-}LS$ is set to 100%. $SimUACO\text{-}LS$ has been used with the following empirically determined parameters: $\beta = 6$ and $\gamma = 0.1$. For both algorithms $SimUACO$ and $SimUACO\text{-}LS$, the quantity of ants is set to the number of activities.

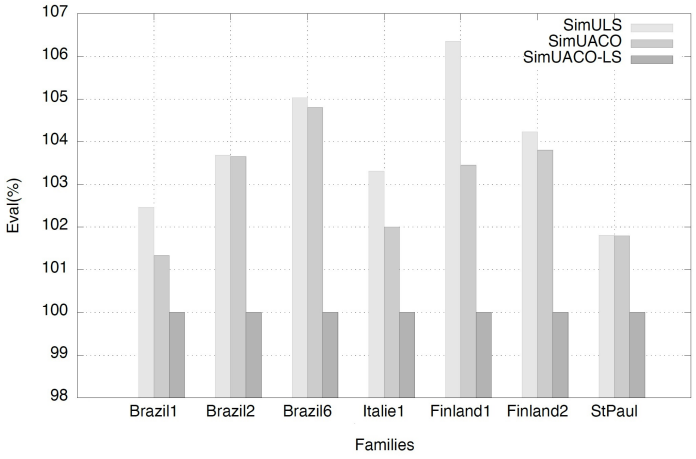

Fig. 1. Comparison between *SimuACO-LS*, *SimuACO* and *SimULS*

Our experiments show that for all methods here presented, all activities are scheduled. *SimUACO-LS* improves the results of *SimUACO* and *SimULS* by respectively 2.97 % and 3.84 %. Some optimal results can be reached on small instance families ($Brazil1$, $Brazil2$ and $Italy1$) and *SimUACO-LS* diverges on average by 3.05% from these optimal results. The standard deviation for *SimUACO-LS* remains less than 5.20 for all the tested instances with an average of 3.80. This standard deviation is lower than that of *SimUACO* (3.86) but higher than that of *SimULS* (1.43). Although the standard deviation of *SimUACO-LS* is higher than *SimULS* one, its average $Eval()$ is better which means that it better explores the solution space.

5 Conclusion & perspectives

This paper presents the planning problem of the health training and simulation center SimUSanté. The proposed method *SimUACO-LS* is a hybridization of the Max Min Ant Colony Optimization algorithm *SimUACO* with the Variable Neighborhood Search *SimULS*. The combination of both metaheuristics allows the weaknesses of each to be overcome. Build a feasible solution is a difficult problem, mainly on the largest instances. Because *SimUACO-LS* strictly respects all hard constraints, it always produces feasible solutions for all instances. Moreover, all activities are scheduled. *SimUACO-LS* always improves the solution quality of *SimUACO* and *SimULS* by respectively 2.97% and 3.84%. The results of *SimUACO-LS* have an average gap of 3.05% with optimal when known. In addition, the schedules obtained are compact and relevant for SimUSanté. The next step will be to improve the search strategy for *SimUACO-LS* in the solution space.

Acknowledgements. This project is supported by the Hauts-de-France region of France and the SimUSanté center.

References

1. Benchmarking Project for (high) School Timetabling. https://www.utwente.nl/en/eemcs/dmmp/hstt/, EEMCS FACULTY, DMMP group - High School Timetabling Project (HSTT)
2. Caillard, S., Brisoux-Devendeville, L., Lucet, C.: Health Simulation Center Simusanté®'s Problem Benchmarks. https://mis.u-picardie.fr/en/Benchmarks-GOC/
3. Caillard, S., Brisoux Devendeville, L., Lucet, C.: Local Search Algorithm to Solve a Scheduling Problem in Healthcare Training Center. In: 17th International Workshop on Project Management and Scheduling (PMS'20). Toulouse, France (Apr 2021)
4. Caillard, S., Devendeville, L.B., Lucet, C.: A Planning Problem with Resource Constraints in Health Simulation center. In: Optimization of Complex Systems: Theory, Models, Algorithms and Applications, WCGO 2019, World Congress on Global Optimization, Metz, France, 8-10 July, 2019. Advances in Intelligent Systems and Computing, vol. 991, pp. 1033–1042 (2019)
5. Cooper, T.B., Kingston, J.H.: The Complexity of Timetable Construction Problems. In: Practice and Theory of Automated Timetabling 1995. Lecture Notes in Computer Science, vol. 1153, pp. 283–295. Springer (1995)
6. Di Gaspero, L., Mccollum, B., Schaerf, A.: The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3) (01 2007)
7. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **26**(1), 29–41 (1996)
8. Gozali, A., Kurniawan, B., Weng, W., Fujimura, S.: Solving University Course Timetabling Problem Using Localized Island Model Genetic Algorithm with Dual Dynamic migration policy. IEEJ Transactions on Electrical and Electronic Engineering **15** (2020)
9. Kenekayoro, P., Zipamone, G.: Greedy Ants Colony Optimization Strategy for Solving the Curriculum Based University Course Timetabling Problem. British Journal of Mathematics & Computer Science **14(2)**, 1–10 (09 2016)
10. Lewis, R.: A Survey of Metaheuristic-Based techniques for University Timetabling problems. OR Spectrum **30**(1), 167–190 (2008)
11. Matias, J., Fajardo, A., Medina, R.: Examining Genetic Algorithm with Guided Search and Self-Adaptive Neighborhood Strategies for Curriculum-Based Course Timetable Problem. In: 2018 Fourth International Conference on Advances in Computing, Communication Automation (ICACCA). pp. 1–6 (2018)
12. Mazlan, M., Makhtar, M., Khairi, A., Mohamed, M.A.: University course timetabling model using ant colony optimization algorithm approach. Indonesian Journal of Electrical Engineering and Computer Science **13**, 72–76 (2019)
13. Mazlan, M., Makhtar, M., Khairi, A., Mohamed, M.A., Rahman, M.: A Study on Optimization Methods for Solving Course Timetabling Problem in University. International Journal of Engineering and Technology(UAE) **7**, 196–200 (2018)
14. Stützle, T., Hoos, H.H.: MAX–MIN Ant System. Future Generation Computer Systems **16**(8), 889–914 (2000)