Multi-neighborhood simulated annealing for personalized user project planning

Yinuo Li, Jin-Kao Hao

Version of Record: https://www.sciencedirect.com/science/article/pii/S1568494622000850
Manuscript_d5d78af48d77ae7988631a831309941b

Multi-neighborhood simulated annealing for personalized user project planning

Yinuo Li [a,b] and Jin-Kao Hao [a,*]

[a] *Université d'Angers, LERIA, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *GePI Conseil, Grand Maine - Rue du Grand Launay, 49000 Angers, France*

Minor Revision 16 December 2021, initial submission June 2021

Abstract

Effective decision support systems are very useful management tools in many applied domains. However, such systems are still scarce or even missing in social and medico-social establishments. This study investigates the personalized user project planning problem in French social and medico-social establishments, whose purpose is to optimize the assignment of multi-featured activities and resources to a group of residents or users subject to complex imperative constraints. We focus on the design and implementation of an innovative multi-neighborhood local optimization algorithm that serves as the key component of a decision support system for these establishments. We assess the effectiveness of the proposed approach on realistic data and show comparisons with other approaches including mathematical programming and greedy search.

Keywords: Project planning; Resource assignment; Heuristics; Simulated annealing.

1 Introduction

Decision support systems are popular management tools in many applied domains, especially in the healthcare sector [1–4]. However, such systems are still scarce or non-existent in the social and medico-social sector in France. In this study, we investigate a real-world personalized user project planning problem in French social and medico-social establishments (called structures hereafter), which is one key challenge for effective decision making in these structures.

* Corresponding author.
Email addresses: yinuo.li@etud.univ-angers.fr (Yinuo Li), jin-kao.hao@univ-angers.fr (Jin-Kao Hao).

The French law No.2002-2 ensures the right for the users of the social and medico-social structures to participate in the elaboration and implementation of their "accompanying projects". As the result, providing users with a personalized support in terms of a life project has become the main mission of all social and medico-social structures. This policy aims to improve the life quality of the aging and disabled population and lighten the financial burden of their family. However, this evolution also brings new challenges to decision-makers of these structures. First, since several years, there has been a significant decrease in staff resources in the social and medico-social sector, on the one hand, and an increase in the societal demands, on the other hand. Second, in order to improve users' independence and social reintegration capacity as much as possible, structures have to provide multiple and varied services and activities requiring multidisciplinary professionals (e.g., doctors, psychologists, educators, etc.) and resources (rooms, vehicles, medical facilities, etc.), making decision-making more complex and difficult. Third, despite the popularization of electronic health records [5], these structures are still faced with a lack of decision support and optimization tools to improve their management efficiency.

According to the guides issued by official organizations [6,7], the personalized user project is defined by a specific document that describes the professional, social and medico-social supports for the user according to his/her needs, expectations and resources. Specifically, a personalized user project is a planning of personalized activities and objectives to achieve during a period. Meanwhile, to ensure the feasibility of this planning, the assignment of different roles among various interveners (professionals, family members, etc) and resources also has to be validated. There are three main phases in a personalized user project: analysis and evaluation phase, programming phase and assessment and regulation phase.

During the analysis and evaluation phase, qualified professionals perform special tests such as personal interview and medical examination to identify user's global desires, abilities and needs. The information collected in this phase is critical to the selection of activities in the next phase. The programming phase is the main phase of a project. The decision-making team of the structure has to define the details of user's project, including the objectives to achieve, the activities to carry on and the roles of interveners. The project has to be not only feasible, but also thoughtful according to user's needs and desires collected in the first phase. Once the project plan is validated by the user, family and structure, the user can start executing his/her project. The assessment and regulation phase at the end of the project aims to assess each performed activity of the project. The evaluation results help to make better decisions in the next project.

This study focuses on the second phase, i.e., the programming phase. This

phase must provide a feasible activity planning for each user together with the required resources. In current practice, structures typically elaborate their operational planning with the help of a qualified professional or a team who must take account into many factors and unexpected changes. Given the complexity of this task, this task is typically difficult and time consuming, leading to plannings that are often far from optimal.

According to [8], the personalized user project planning problem can be considered to belong to the general class of project scheduling and management problems, which has been widely studied to fit various real-world applications such as urban project planning [9], IT-projects scheduling and assigning [10], chemical projects human resources optimization [11] and software maintenance projects selection and scheduling [12]. Most conventional project management problems are aimed at optimizing the use of resources including human resources and material resources. Therefore, the classical model called the resource constrained project scheduling problem (RCPSP) [13] has been widely extended to formulate different problems [14,15,10,16].

Generally, a project scheduling problem can be considered as a combination of several subproblems (also called planning levels [10]): project (task) selection, project (task) timetabling and resources assignment. Given the high complexity of scheduling problems with multiple planning levels, decomposition-based approaches are often applied to solve subproblems independently. Mathematical programming provides a class of popular and powerful methods for solving this kind of problems, including mixed-integer programming [10], matrix-based method [17,18], branch-and-bound [19], etc. Meanwhile, due to the high computational cost of these exact methods and the presence of complex non-linear constraints in real-world applications, heuristics and metaheuristics are also investigated to find "good enough" solutions [20–22].

The user project planning problem investigated in this study was formally defined by [8], which has a number of specific characters compared to other project scheduling problems. In particular, the problem features a number of complex constraints and three optimization objectives, as reviewed in Section 2, making it quite challenging for solution methods. [8] experimented a constructive greedy algorithm and mathematical programming. The mathematical programming approach provided near optimal solutions for small-scale instances, but its performances deteriorated greatly on middle and large-scale instances. Moreover, its high computational cost makes this approach unpractical in the real life application. The greedy approach has the advantage of being fast, but the quality of its solutions is far from satisfactory.

This study investigates the first metaheuristic approach for the user project planning problem, which will serve as the basis of a decision support system for social and medico-social structures. Our approach adopts the general simu-

lated annealing framework and introduces innovative search components with respect to the optimization objectives as well as various imperative constraints of the studied problem. We assess the approach on a set of realistic data and show its effectiveness both in terms of solution quality and computational efficiency compared to the previous approaches (integer programming and greedy search).

The rest of the paper is organized as follows. Section 2 presents the formal model of the problem in terms of constraints and objectives. Section 3 describes the new solution approach. Section 4 reports computational experiments on benchmark instances. Section 5 shows additional experiments to analyze the key components of the approach. The last section draws conclusions and identifies research perspectives.

2 Personalized User Project Planning Problem

This section recalls the mathematical formulation of the personalized user project planning problem given in [8], which serves as the basis for the current study.

2.1 Solution representation

The basic user project planning problem is to assign the users to available activities with the required human and facility resources with respect to a number of constraints and optimization objectives. Based on the notations of Table 1, we adopt a $U \times T$ matrix $\mathbf{X}$ and a $R \times T$ matrix $\mathbf{Y}$ to represent a feasible user project schedule S such that

$$x_{ut} = \begin{cases} a & \text{user } u \text{ participates in activity } a \text{ at time } t, \\ -1 & \text{user } u \text{ is not available to be arranged at time } t \ (ua_{ut} = -1), \\ 0 & \text{user } u \text{ is available at time } t \end{cases}$$

$$y_{rt} = \begin{cases} a & \text{resource } r \text{ is used by activity } a \text{ from time } t \text{ to time } t + dur_a - 1, \\ -1 & \text{resource } r \text{ is not available to be used at time } t \ (ra_{rt} = -1), \\ 0 & \text{resource } r \text{ is available at time } t \text{ if it is not used by an activity earlier} \end{cases}$$

Figure 1 illustrates a feasible solution, where users u_1 and u_3 participate in activity a_2 at timeslot t_1 and t_2 with resources $\{r_1, r_3\}$ and user u_2 takes part in activity a_1 from timeslot t_2 to t_4 with resource $\{r_2\}$.

Hereafter, we use Ω to denote the search space explored by our algorithm, which is composed of all possible feasible solutions represented by two matrices $\mathbf{X}$ and $\mathbf{Y}$.

2.2 Constraints

A feasible user project must comply with the following imperative constraints that are described using the notations of Table 1.

- C_1. *User availability.* A user is assigned to at most one activity at the same time and only when the user is available. This constraint is always satisfied by adopting the representation above.
- C_2. *Activity availability.* An activity is proposed to users only when the activity is available and does not cross more than one day.

$$\forall u \in \mathcal{U}; \forall a \in \mathcal{A}; \forall t \in \mathcal{T}; (aa_{at} = 0) \vee (x_{ut} \neq a)$$

- C_3. *Resource availability.* Only available resources are assigned and each resource is assigned to at most one activity at the same time.

$$\forall r \in \mathcal{R}; \forall t_i \in \mathcal{T}; \forall a \in \mathcal{A}; (y_{rt_i} \neq a) \vee (\bigcup_{d=i+1}^{i+dur_a-1} y_{rt_d} = \{0\})$$

- C_4. *Budget constraint.* The total cost of the activities assigned to a user must satisfy user's budget.

$$\forall u \in \mathcal{U}; \sum_{a \in \mathcal{A}} act_{ua} \times pri_a \leq bud_u$$

- C_5. *Activity capacity.* The number of users assigned to an activity cannot exceed the capacity of the activity.

$$\forall a \in \mathcal{A}; \forall t \in \mathcal{T}; num\_u_{at} \leq cap_a$$

- C_6. *Feature constraint.* An activity ready for assignment must have all required resources.

	t_1	t_2	t_3	t_4
u_1	a_2	a_2	-1	0
u_2	-1	a_1	a_1	a_1
u_3	a_2	a_2	0	0

	t_1	t_2	t_3	t_4
r_1	a_2	0	0	0
r_2	0	a_1	0	0
r_3	a_2	0	-1	0

Fig. 1. Illustrative example of solution representation with a $U \times T$ matrix $\mathbf{X}$ (left) and a $R \times T$ matrix $\mathbf{Y}$ (right).

Table 1
Notations used in the mathematical formulation [8].

Symbol	Description
$\mathcal{U}$	Set of users, $\|\mathcal{U}\| = U$, each being characterized by his/her budget, availability and preference for each activity
$\mathcal{A}$	Set of activities, $\|\mathcal{A}\| = A$, each being characterized by its price, duration (number of timeslots needed), capacity, availability and features required
$\mathcal{R}$	Set of resources, $\|\mathcal{R}\| = R$, including human and facility resources, each being characterized by its availability and features
$\mathcal{T}$	Set of timeslots, $\|\mathcal{T}\| = T$, during which users, activities and resources are to be scheduled. Typically, we take 8 timeslots per day and 5 working days per week
$\mathcal{F}$	Set of features, $\|\mathcal{F}\| = F$, including professional skills, room capacity, vehicle features, etc, required by activities and provided by resources
TS	Number of timeslots per day
ua_{ut}	Whether user u is available at timeslot t, $ua_{ut} = 0$ if available, -1 otherwise
bud_u	Budget of user u
aa_{at}	Whether activity a is available at timeslot t, $aa_{at} = 0$ if available, -1 otherwise
dur_a	Duration of activity a
pri_a	Price of activity a
cap_a	Capacity of activity a
ra_{rt}	Whether resource r is available at timeslot t, $ra_{rt} = 0$ if available, -1 otherwise
pre_{ua}	Preference of user u to activity a, pre_{u0} always equals 0
req_{af}	Whether activity a requires feature f, $req_{af} = \{0, 1\}$
fea_{rf}	Whether resource r contains feature f, $fea_{rf} = \{0, 1\}$
$num\_u_{at_k}$	Number of users starting activity a at timeslot t_k in a candidate solution S; $num\_u_{at_k} = \sum_{u \in \mathcal{U}} sta_{uat_k}$, where $sta_{uat_k} = \begin{cases} 1 & \text{if } x_{ut_k} = a \wedge (x_{ut_{k-1}} \neq a \vee k = 1), \\ 0 & \text{otherwise.} \end{cases}$
act_{ua}	Whether user u participates in activity a in a candidate solution S, $act_{ua} = 1$ if $\exists x_{ut} \in S$, $a = x_{ut}$, $act_{ua} = 0$ otherwise.
res_{at}	Set of resources used by activity a with start-time t in a candidate solution S $res_{at} = \{r \mid r \in \mathcal{R}, y_{rt} = a\}$

$$\forall a \in \mathcal{A}; \forall t \in \mathcal{T}; \forall f \in \mathcal{F}; (num\_u_{at} = 0 \vee req_{af} = 0) \vee \sum_{r \in res_{at}} fea_{rf} > 0$$

- C_7. *Preference constraint.* A user provides a preference score for each intended activity.

$$\forall u \in \mathcal{U}; \forall a \in \mathcal{A}; (act_{ua} = 0) \vee (pre_{ua} > 0)$$

2.3 Objectives

The user project planning problem aims to optimize three objectives while satisfying the constraints introduced in the last section. Using the notations of Table 1 and let S be a candidate solution, we define the objectives as follows.

Suitability maximization (f_1): This objective aims to satisfy as much as possible the activity preferences of the users. f_1 is the main problem objective.

$$f_1(S) = \sum_{u \in \mathcal{U}} \sum_{a \in \mathcal{A}} act_{ua} \times pre_{ua}$$

Structure resource minimization (f_2): This objective aims to reduce the scheduled resources to save costs for the structure. This is equivalent to maximize the available timeslots of the resources.

$$f_2(S) = R \times T - \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} dur_{y_{rt}} + \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} ra_{rt}$$

User cost minimization (f_3): This objective minimizes the cost of the scheduled activities to save expenses for the users, which is equivalent to maximize the unconsumed budget of users.

$$f_3(S) = \sum_{u \in \mathcal{U}} (bud_u - \sum_{a \in \mathcal{A}} act_{ua} \times pri_a)$$

3 Multi-Neighborhood Simulated Annealing

This section is dedicated to the presentation of the proposed multi-neighborhood simulated annealing algorithm (MNSA) for the user project planning problem.

3.1 General procedure

The MNSA algorithm follows the general SA framework [23] and features the combination of three complementary neighborhoods to explore candidate solutions. As shown in Algorithm 1, MNSA starts with an initial solution S and an initial temperature $T = T_0$. Then it performs a number of iterations to find solutions of increasing quality. During each iteration, MNSA first picks a neighborhood N_i $(i = 1, 2, 3)$ with a probability p_i where $p_1 + p_2 + p_3 = 1$,

then selects randomly a neighbor solution S' from N_i. If S' is feasible, MNSA computes the variation of the objective function between the current solution S and the sampled neighbor solution S', $\Delta f = f(S') - f(S)$ (Δf is called the move gain). If S' is better than S or as good as S ($\Delta f \geqslant 0$), then S' replaces the current solution to become the new current solution. This transition is called a move. Otherwise ($\Delta f < 0$), S' is accepted as the new current solution with probability $P(\Delta f, T) = e^{\Delta f/T}$. After performing L moves with the current temperature, where L is a controlling parameter, parameter T is decreased by a constant cooling factor $\alpha < 1$. Then MNSA starts the next round of its search with the new temperature. MNSA terminates and returns the best solution found during the search when a given stopping condition is met.

Algorithm 1 Multi-neighborhood simulated annealing (MNSA).

Input: Problem instance I, initial solution S, initial temperature T_0, cooling factor α, move counter L, probabilities p_1, p_2, p_3 for applying neighborhoods N_1, N_2, N_3
Output: The best solution found S^{best}
$T \leftarrow T_0$ /* Set initial temperature */
$S^{best} \leftarrow S$ /* Record the best solution found so far */
repeat
  $Iter \leftarrow 0$ /* Iteration counter */
  $Move \leftarrow 0$ /* Move counter */
  repeat
    Pick neighborhood N_1, N_2 or N_3 with probabilities p_1, p_2, p_3
    Pick at random a neighbor solution S' from the selected neighborhood N_i
    $Iter \leftarrow Iter + 1$
    if S' is feasible **then**
      $\Delta f \leftarrow f(S') - f(S)$
      if $\Delta f \geqslant 0$ **then**
        $S \leftarrow S'$; $Move \leftarrow Move + 1$
      else if $e^{\Delta f/T} > random(0, 1)$ **then**
        $S \leftarrow S'$; $Move \leftarrow Move + 1$
      end if
      Update S^{best} if $f(S) > f(S^{best})$
    end if
  until $Move = L$
  $T \leftarrow T * \alpha$ /* Temperature cooling down */
until a stopping condition is met
return S^{best}

It is worth noting that each iteration of MNSA (counter $Iter$) does not necessarily lead to a move (counter $Move$). After each temperature cooling, the counters $Iter$ and $Move$ will be restarted from zero. As the temperature decreases, the acceptance probability $P(\Delta f, T) = e^{\Delta f/T}$ becomes lower, leading to a lower move acceptance rate $\gamma = Move/Iter$. MNSA uses the move acceptance rate γ as one of the stopping conditions, i.e., the algorithm stops when the move acceptance rate drops to less than a given threshold (empirically fixed to $\gamma = 5\%$) for five consecutive temperature decreases. Another

stopping condition is a cutoff time limit. In accordance with the real-life situations in different structures, we fixed the time limit to be 30 minutes, which is an acceptable time for decision-makers. During the MNSA search process, only feasible solutions are evaluated and accepted, therefore the best solution found is always feasible. As we explained earlier, the user project planning problem is defined with a set of complex constraints whose feasibility verification is much computationally expensive. As such, the SA framework is quite suitable because only one candidate neighbor solution (instead of all neighbor solutions) is examined at each iteration, which possibly leads to an effective move.

3.2 Evaluation function

A candidate solution S in the search space is a feasible user projects schedule that satisfies all constraints ($C_1 - C_7$). To assess the quality of solution S, the algorithm uses the following weighted evaluation function f:

$$f(S) = \sum_{i=1}^{3} w_i \times f_i(S) \tag{1}$$

where each coefficient $w_i = \alpha_i \times \frac{1}{f_i^{max}}$ $(i = 1, 2, 3)$ is the relative weight given to the objective function f_i of the problem and α_i takes values in $[1, \ldots, 5]$ and is set by the decision-maker according to the actual importance. $f_i^{max} = max\{f_i(S) : S \in \Omega\}$ is used to normalize the objectives that are of different nature and have different scales. The normalization is based on the upper-bound function transformation method [24] (see [8] for more details).

3.3 Constructive greedy initialization procedure

The MNSA algorithm requires an initial solution to start its search, which is built with the greedy procedure presented by [8]. This procedure follows the spirit of the constructive greedy strategies [25,26] and performs several rounds of activity arrangements, giving a high priority to preferred activities. In every round, users are traversed according to the increasing order of budget, which is motivated by the solidarity consideration in real-life that users with financial difficulties are given a high priority. Specifically, for each user under consideration, the followings steps are performed to achieve an arrangement.

(1) Select the most preferred eligible activity a of the user.
(2) Iterate through timeslots from the first one until a timeslot t is found where a has been started at t and still has available places, if found, add directly the user to a and move to step 5.

(3) Iterate through timeslots from the first one until a timeslot t is found where enough available resources can be used, if found, arrange a for the user with resources and move to step 5.
(4) Move to the next preferred activity and repeat steps 2-4 until all eligible activities have been considered.
(5) Move to the next user (if the current user is the last one, move to the first user) and repeat steps 1-5.

The initialization procedure stops when all the activities of all users have been considered and no more activity can be arranged without violating constraints. During the constructive greedy procedure, the feasibility of each decision is first verified and then executed. If the decision cannot satisfy all constraints, it will not be performed. As such, the initialization procedure always provides a feasible solution satisfying all the constraints of the problem. The time complexity of this procedure is bounded by $\mathcal{O}(U * A * T * R)$ where U, A, T and R represent the number of users, activities, timeslots and resources, respectively.

3.4 Neighborhoods based on activity change, swap and exchange

It is well known that neighborhood is one key component of any SA algorithm. We present three dedicated neighborhood operators specifically designed for the user project planning problem. These neighborhood operators are inspired by real-life manual project scheduling operations used by human planners.

3.4.1 Activity cancellation and activity arrangement

The cancellation and arrangement of an activity are the most basic operations in user project planning.

The activity cancellation operation is composed of the following steps:

(1) Given the attended activity a, user u and a solution S, the start time t and the resources set used res_{at} are known;
(2) Set variable(s) $x_{ut} \dots x_{ut+dur_a-1}$ to 0;
(3) If no more user participates in activity a from time t, set variable(s) $\{y_{rt} \mid r \in res_{at}\}$ to 0.

The activity arrangement operation is composed of the following steps:

(1) Given the unattended activity a, user u and a solution S, verify constraints C_4, C_7, if feasible, continue the arrangement; else, stop the arrangement;
(2) Iterate through timeslots from the first one until a timeslot t is found

where a has been started at t and still has available places, if found, set variable(s) $x_{ut} \dots x_{ut+dur_a-1}$ to a, stop the arrangement;

(3) Pick randomly a timeslot t_i and verify other constraints until a feasible timeslot is found or tm timeslots have been considered, where tm is a controlling parameter representing the maximum number of timeslot verifications to arrange an activity. The verification of resources is achieved by iterating through the available resources from a random start and adding the resources with the required features into resources set res_{at_i} until all features are satisfied or all resources have been verified. If t_i is feasible, set variable(s) $x_{ut_i} \dots x_{ut_i+dur_a-1}$ to a and set variable(s) $\{y_{rt_i} \mid r \in res_{at_i}\}$ to a, stop the arrangement.

3.4.2 Single activity change neighborhood N_1

The single activity change affects one user and one activity, which is a basic operation in user project planning process. Figure 2 shows two moves that are eligible for single activity change: a cancellation and an arrangement. For any user and activity, the operation to achieve one move depends on whether this user has participated in this activity.

(1) Pick a user u from $\mathcal{U}$ and pick an activity a from $\mathcal{A}$;
(2) If u has already a arranged in his/her project, cancel a for u;
(3) Otherwise, if u doesn't has a arranged in his/her project, arrange a for u if it doesn't create any conflict with respect to the constraints.

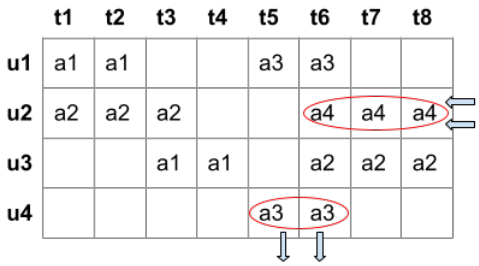


Fig. 2. Single activity changes

As explained above, the cancellation of an activity is always feasible but the arrangement of an activity may fail. The maximum number of timeslot verifications tm is a controlling parameter, the greater it is, the more probably the activity will be successfully arranged, but is more computationally expensive. Hence an appropriate tm is necessary to balance feasibility and efficiency. The single activity change will affect the number of activities in the project of the selected user and may also affect the schedule of resources. As such, it allows the solution to stay in a stable sparsity level because when the solution matrix is too sparse, which means that users don't have many activities selected in their projects, unattended activities are more probably to be picked and

then arranged. Reversely, when the solution is too dense, the cancellation of activities is more probably to be performed. In real-life scheduling, this "single activity change" operation is often used in the conception phase of users' projects to establish feasible projects for every user.

3.4.3 Activity swap neighborhood N_2

An activity swap for a user is performed by a cancellation of an attended activity and an arrangement of an unattended activity. Figure 3 shows one eligible move for activity swap.

(1) Pick a user u from $\mathcal{U}$;
(2) Swap the best activity pair (a_1, a_2): cancel activity a_1 for u, then arrange activity a_2 for u if it doesn't create any conflict.

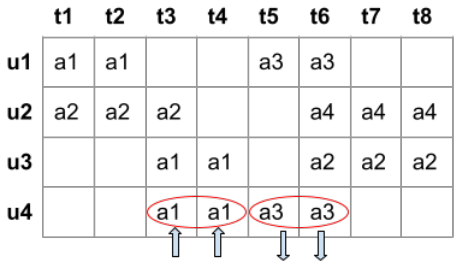


Fig. 3. Activity swap

We define the best activity pair (a_1, a_2) to swap for user u as follows:

$$(a_1, a_2) = (\underset{a \in \mathcal{A}, act_{ua}=1}{\arg\min} \; pre_{ua}, \; \underset{a \in \mathcal{A}, act_{ua}=0}{\arg\max} \; pre_{ua})$$

In other words, we replace user's least favorite activity in his/her project with his/her favorite unattended activity to maximize the main suitability objective (f_1). An activity swap concerns one user and two activities which can be seen as two particular simultaneous single activity changes. The difference is that the activity swap will never change the number of activities in the project of the selected user. In real-life, the "activity swap" operation is often used during the execution phase of the project when there exists a better activity for the user or when some selected activities become unfeasible and we want to maintain the same number of the arranged activities.

(1)

3.4.4 Activity exchange neighborhood N_3

Unlike the two neighborhoods above, an activity exchange concerns two users. It consists of swapping two different activities with the same start time of two users or simply switching the activity from one user to another. Figure 4 shows two eligible moves for activity exchange.

(1) Pick a user u_1 from $\mathcal{U}$;
(2) Take the best user-activity pair (u_2, a_1) to exchange, the start time of a_1 for u_1 is t_1;
(3) If a user u_2 starts an another activity a_2 at time t_1, exchange a_1 and a_2 of the two users ;
(4) Otherwise, if u_2 is available at t_1, cancel a_1 for u_1 and arrange a_1 for u_2 at t_1.

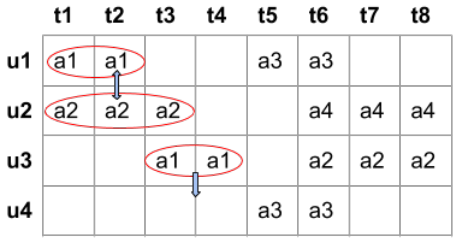


Fig. 4. Activity exchanges

We define the best user-activity pair (u_2, a_1) to exchange for user u_1 as follows:

$$(u_2, a_1) = \underset{u \in \mathcal{U}, a \in \mathcal{A}}{\arg\max}\ pre_{ua} - pre_{u_1 a} + pre_{u_1 x_{ut_1}} - pre_{u x_{ut_1}}$$

In other words, the best user-activity pair to exchange for u_1 will lead to the largest gain of the evaluation function f (Eq. (1)). The activity exchange will never change the schedule of resources because there is no activity arrangement or cancellation involved, it only changes the choice of user for an activity already arranged. Hence, it doesn't affect the structure resource objective (f_2) and the user cost objective (f_3). As such, the variation of the evaluation function f can be calculated quickly by calculating simply the variation of the suitability objective f_1. The "activity exchange" operation is often applied in real-life during the conception phase and execution phase of users' projects because many activities need typically to be changed after the evaluation phase to maximize the global user satisfaction and to ensure that every activity will be attended by users who need it the most.

3.4.5 Time complexity

(1) *Single activity change:* Given user u and activity a, checking if u has participated in a requires $O(1)$. If the operation is a cancellation, the time complexity is bound by $O(R)$ when the resources are involved. If it's an arrangement, tm timeslots will be verified where each verification requires $O(R)$. As such, one single activity change takes time $O(tmR)$, which is $O(R)$ since the timeslots verification number tm is a constant parameter.
(2) *Activity swap:* Finding the best activity pair (a_1, a_2) to swap for a user requires $O(A)$. Then, the time complexity for a cancellation and an arrangement is $O(R)$. Therefore, one activity swap takes time $O(A + R)$.
(3) *Activity exchange:* To find the best user-activity pair (u_2, a_1) to swap for a user, all user-activity pairs have to be evaluated, the variation of the evaluation function f can be calculated in $O(1)$ since only the first objective f_1 is involved. Therefore, the time complexity of one activity exchange is bound by $O(A * U)$.

3.5 Probabilistic exploration of the neighborhoods

Given the three neighborhoods presented previously, MNSA explores them in a probabilistic way. Specifically, at each iteration, MNSA selects neighborhood N_i $(i = 1, 2, 3)$ to explore with probability p_i such that $p_1 + p_2 + p_3 = 1$. As explained above, these three neighborhoods have very different efficiency and effectiveness. Compared to other neighborhood combination methods such as neighborhood union or token-ring neighborhood exploration [27], the probabilistic combination has the advantage of allowing us to adjust the importance given to each neighborhood by tuning the probability settings (p_1, p_2, p_3), which in turn influences the behavior of the MNSA algorithm. More information is provided in Section 5.1.

4 Experimentations and Results

4.1 Experimental setting

Computing platform. The proposed MNSA algorithm was implemented in C++ and compiled using the g++ compiler with the -O3 option. The experiments were carried on an Intel Xeon E5-2670 processor with 2.5 GHz and 200 MB RAM under the Linux operating system.

Table 2
Settings of MNSA's parameters.

Parameter	Section	Description	Value
T_0	3.1	initial temperature of SA	0.1
L	3.1	number of move per temperature of SA	300000
α	3.1	cooling ratio of SA	0.97
p_1, p_2, p_3	3.1	neighborhood selection probabilities	(0.8,0.1,0.1)
tm	3.4	maximum selection number of timeslots	10

We also tested the mathematical programming approach with a 0/1 programming model (see the Appendix) with the CPLEX MIP solver (version 12.8), run on an Intel Core i7-8750H 2.20 GHz processor with 32 GB RAM under the Linux operating system with a time limit of 2 hours.

Parameter settings. We used the automatic configuration tool "irace" [28] to calibrate the parameters of the MNSA algorithm. This was achieved by feeding irace with six randomly selected instances with a tuning budget of 1000. We adopted the best parameter configuration given by irace and shown in Table 2.

4.2 Test instances

Our experiments were conducted on the 20 benchmark instances introduced in [8] [1] . These instances have different sizes and characteristics. They have been created by a program generator with various tunable parameters (user number, activity number, scheduling length, etc.). These instances cover various real scenarios in different structures, bases on the responses (more than 200) received from a large-scale survey sent to more than 3600 structures in the social and medico-social sector in France. According to the number of users (from 10 to 100), these 20 instances include 3 small structure size instances, 12 medium structure size instances and 5 large structure size instances.

4.3 Computational results

Given the stochastic nature of the MNSA algorithm, we ran MNSA 20 times independently with 20 random seeds to solve each test instance. Each run was stopped when the cutoff time of 30 minutes was reached or when the move acceptance rate γ drops below the threshold of 5% for five consecutive search rounds (see Section 3.1).

[1] These instances are publicly available at: `http://www.info.univ-angers.fr/pub/hao/userprojectplanning.html`

Table 3 shows the computational results of the proposed MNSA algorithm and the compared approaches. Column 1 gives the instance identifier and Column 2 and 3 give the number of users U and the number of activities A, which indicate the size of instance. Column 4 gives the objective function weight used for each instance (see Eq. (1)). Columns 5 to 8 report the results of MNSA with its best objective value f_{best} found among 20 runs, average objective value f_{avg}, standard deviations std and the average time t_{avg} in seconds to reach the best objective value. Note f_{best} and f_{avg} correspond to lower bounds of the optimal objective value. Columns 9 to 11 (0/1P) give the lower bound (LB), upper bound (UB) and $gap\%$ ($gap\% = |UB - LB|/|LB|$) achieved by the CPLEX solver with the 0/1 programming model. Columns 12 and 13 (GIP) present the results and run time of the greedy procedure. The best values among the results of the compared approaches are highlighted in boldface. Entries with "-" mean that CPLEX failed to output a result due to one of three reasons: i) out of memory; ii) unknown solution status; iii) the pre-processing phase of CPLEX cannot be finished in two hours.

From the results of Table 3, we can make the following comments. First, compared to the powerful MIP solver CPLEX, among the 12 instances where CPLEX produced a result, MNSA found a better solution for seven instances against four cases in favor of the CPLEX. The winning cases of CPLEX concerns only some small or medium scale instances. Moreover MNSA has very small standard deviations, indicating its high robustness. Second, MNSA obviously improves on the results of the constructive greedy procedure GIP for the whole set of 20 instances. Even the average results of MNSA dominates those of GIP. The small p-value ($8.86e-5 << 0.05$) from the non-parametric Wilcoxon signed-rank test confirms the statistical significance between the results of MNSA and those of GIP. To complement this numerical comparison, Figure 5 summarizes the comparison using barchart, which clearly shows the dominance of the MNSA algorithm over the reference approaches.

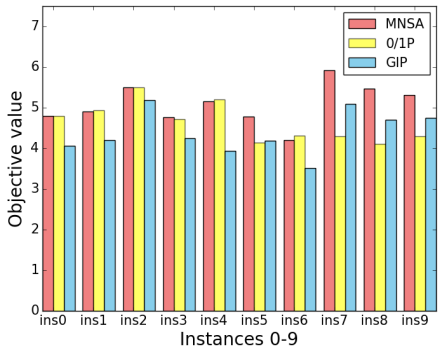
Finally, to further illustrate the the performance difference among the three compared approaches in a global manner, we show in Figure 6 their performance profiles [29]. Since the performance profile is defined with respect to a minimization criterion and our problem has a maximization objective, we first applied a transformation to convert our objective function into a minimization case. Given a set $\mathcal{I}$ of problem instances and a set $\mathcal{A}$ of algorithms, we convert the objective value $f_{i,a}$ on instance i by algorithm a to $\hat{f}_{i,a} = f_{i,best} + \Delta f_{i,a}$ where $f_{i,best}$ is the best objective value on instance i by all algorithms in $\mathcal{A}$ and $\Delta f_{i,a} = f_{i,best} - f_{i,a}$ ($\Delta f_{i,a} \geqslant 0$), which represents the gap of the objective value of approach a relative to the very best objective value. If $\Delta f_{i,a} = 0$, the approach a is the best approach in $\mathcal{A}$ on instance i, and we have $\hat{f}_{i,a} = f_{i,a} = f_{i,best}$. If $\Delta f_{i,a} > 0$, the approach a performs worse than the best approach in $\mathcal{A}$, and we have $\hat{f}_{i,a} > f_{i,best}$. This transformation method thus ensures that all transformed objective values are greater than zero and maintain

Table 3
Computational results of the proposed MNSA algorithm and comparison with the reference approaches: 0/1 mathematical programming with CPLEX (0/1P) and greedy initialization procedure (GIP).

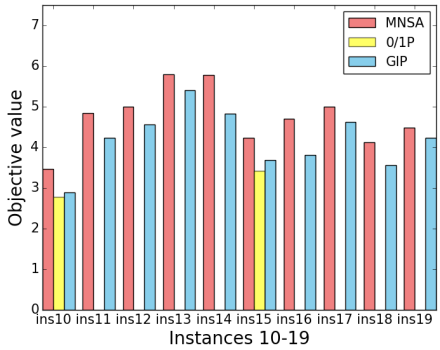
Ins	U	A	α	MNSA				0/1P			GIP	
				f_{best}	f_{avg}	std	t_{avg}	LB	UB	$gap\%$	f	t (s)
0	10	10	{5,3,0}	**4.8033**	4.8027	0.001137	158.87	**4.8033**	4.8068	0.074%	4.0623	0.0003
1	10	15	{5,2,1}	4.9113	4.8990	0.010970	171.29	**4.9300**	5.2124	5.73%	4.2060	0.0008
2	10	15	{5,1,2}	5.4940	5.4842	0.007514	100.20	**5.4966**	5.5445	0.87%	5.1804	0.0003
3	30	20	{5,1,1}	**4.7599**	4.7494	0.005576	385.44	4.7178	5.4622	15.78%	4.2480	0.0079
4	30	30	{5,3,0}	5.1557	5.1267	0.014311	241.09	**5.2055**	7.2272	38.84%	3.9318	0.0059
5	30	40	{5,1,2}	**4.7824**	4.7708	0.005865	638.44	4.1436	5.9389	43.33%	4.1916	0.0123
6	50	15	{5,2,1}	4.1999	4.1537	0.030969	203.50	**4.3155**	5.6144	30.10%	3.5143	0.0136
7	50	20	{5,3,0}	**5.9196**	5.8732	0.029707	413.04	4.3039	130.5689	2933.77%	5.0879	0.0051
8	50	30	{5,2,1}	**5.4681**	5.4393	0.015094	443.30	4.1035	119.8726	2821.22%	4.7085	0.0140
9	50	30	{5,2,1}	**5.3193**	5.2901	0.013284	401.42	4.2911	123.7569	2783.99%	4.7456	0.0103
10	50	50	{5,1,1}	**3.4648**	3.4508	0.010898	772.11	2.7762	4.0878	47.24%	2.8983	0.0110
11	50	40	{5,1,1}	**4.8414**	4.8293	0.007239	1011.47	-	-	-	4.2377	0.0879
12	50	60	{5,1,1}	**4.9955**	4.9865	0.005668	1279.76	-	-	-	4.5686	0.1007
13	50	80	{5,1,2}	**5.7975**	5.7845	0.005673	1580.86	-	-	-	5.4005	0.1621
14	50	100	{5,3,0}	**5.7826**	5.7672	0.011426	1415.78	-	-	-	4.8303	0.6189
15	70	30	{5,1,2}	**4.2270**	4.2042	0.011491	604.31	3.4143	5.3128	55.60%	3.6945	0.0197
16	70	40	{5,3,0}	**4.6985**	4.6606	0.021251	939.02	-	-	-	3.8128	0.3053
17	70	50	{5,1,2}	**5.0051**	4.9885	0.007458	1497.09	-	-	-	4.6198	0.2827
18	100	30	{5,2,1}	**4.1314**	4.1107	0.011794	775.01	-	-	-	3.5620	0.0678
19	100	50	{5,1,1}	**4.4820**	4.4716	0.005591	1789.225	-	-	-	4.2342	0.5341
p-value											8.86e-5	

the original performance quality order. Based on the transformed objective value $\hat{f}_{i,a}$, we define the performance ratio by $r_{i,a} = \hat{f}_{i,a}/min\{\hat{f}_{i,a} : a \in \mathcal{A}\}$ to represent the performance on instance i by approach a compared to the best performance by any approach on i. We set $r_{i,a}$ to $+\infty$ if approach a does not solve instance i. Then, to obtain an overall performance assessment of approach a, we define $P_a(\tau) = size\{i \in \mathcal{I} : r_{i,a} \leq \tau\}/|\mathcal{I}|$, which is the probability for approach a that its performance ratio $r_{i,a}$ is within a factor τ. The function P_a is the (cumulative) distribution function for the performance ratio. The value $P_a(1)$ is the number of instances that approach a matches or wins over other approaches. We draw the performance profiles with the software "*perprof-py*" [2]. From Figure 6, we observe that MNSA dominates the other reference approaches with the best performance profile either in terms of the best objective value or the average objective value.

[2] Available at https://ufpr-opt.github.io/perprof-py/index.html



(a) Instances 0-9



(b) Instances 10-19

Fig. 5. Comparison summary of MNSA with 0/1 mathematical programming with CPLEX (0/1P) and greedy initialization procedure (GIP).

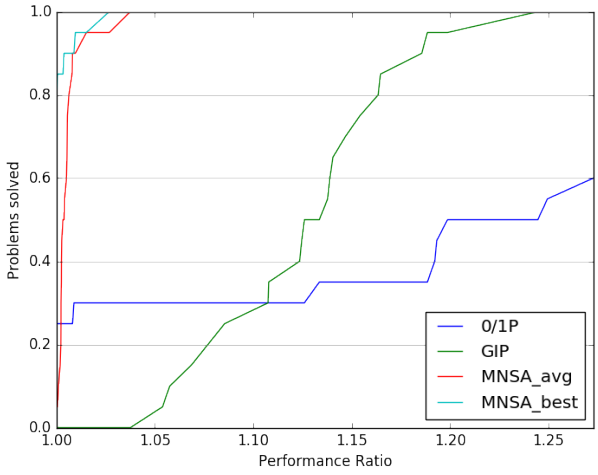


Fig. 6. Performance-profile

5 Analysis and Discussions

In this section, we present an analysis of the neighborhoods and parameters used in the proposed algorithm.

MNSA explores three neighborhoods N_1, N_2, N_3 in a probabilistic way. In order to assess the benefit of this neighborhood combination strategy, we present a comparison of MNSA with three MNSA variants where only one single neighborhood is used. We ran these variants under the same experimental conditions as for MNSA to solve the 20 instances. The computational results are reported in Table 4 and the visual presentation is shown in Figure 7. We observe that the combination of the three neighborhoods perform significantly better than each single neighborhood used alone. Moreover, among the three neighborhoods, the "single activity change" neighborhood N_1 performs much better than the other two neighborhoods. This explains why in MNSA, N_1 is given a much higher application probability ($p_1 = 0.8$ against $p_2 = p_3 = 0.1$, see Table 2), allowing thus this dominant neighborhood to play its role.

Finally, we find that the "activity swap" neighborhood N_2 alone can barely improve the results of the constructive greedy initialization. In fact, the greedy procedure typically produces a very dense solution with all possible activities and resources arranged. As the result, the activity swap can rarely find a feasible move from such a solution. Interestingly, when combined with other neighborhoods especially with N_1, the variant state of solution provides N_2 with more opportunities to find a feasible move. In conclusion, the combination of neighborhoods brings highly positive effects for the search thanks to the collective symbiosis of these complementary neighborhoods.

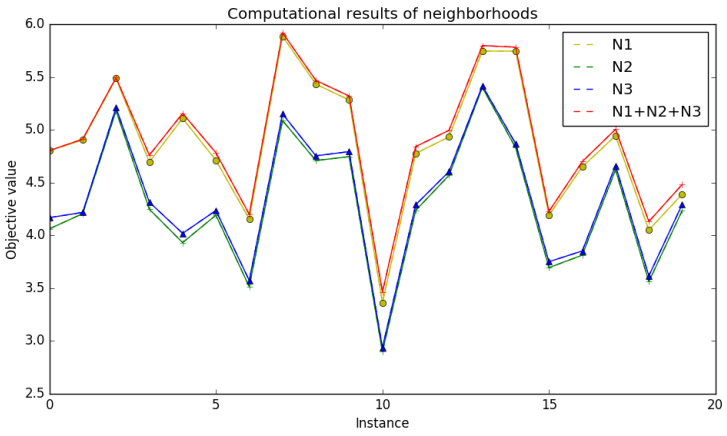


Fig. 7. Computational results of neighborhoods

Table 4
Comparative results (f_{best}) of MNSA with the combined neighborhoods ($N_1 + N_2 + N_3$) and each individual neighborhood: "single activity change" neighborhood (N_1), "activity swap" neighborhood (N_2), "activity exchange" neighborhood (N_3).

Instance	N_1	N_2	N_3	$N_1 + N_2 + N_3$
0	4.8033	4.0623	4.1682	4.8033
1	4.9050	4.2060	4.2179	4.9113
2	5.4874	5.1804	5.2096	5.4940
3	4.6947	4.2480	4.3150	4.7599
4	5.1088	3.9318	4.0195	5.1557
5	4.7107	4.1916	4.2361	4.7824
6	4.1559	3.5143	3.5731	4.1999
7	5.8859	5.0879	5.1509	5.9196
8	5.4326	4.7091	4.7537	5.4681
9	5.2823	4.7456	4.7936	5.3193
10	3.3632	2.8983	2.9389	3.4648
11	4.7769	4.2377	4.2905	4.8414
12	4.9326	4.5686	4.6048	4.9955
13	5.7463	5.4005	5.4172	5.7975
14	5.7448	4.8303	4.8695	5.7826
15	4.1891	3.6945	3.7509	4.2270
16	4.6488	3.8128	3.8526	4.6985
17	4.9438	4.6198	4.6610	5.0051
18	4.0512	3.5620	3.6170	4.1314
19	4.3904	4.2342	4.2933	4.4820
p-value	8.86e-05	8.86e-05	8.86e-05	

Table 5
Parameter levels for the 2-level full factorial experiment.

Parameters	Low level	High level
T_0	0.05	1.0
L	100000	500000
α	0.95	0.99
p_1, p_2, p_3	(0.4,0.3,0.3)	(0.8,0.1,0.1)
tm	10	50

5.2 Analysis of parameters

The proposed MNSA algorithm requires five parameters. The initial temperature T_0, the number of move per temperature L, and the cooling ratio α are required by the SA framework, while the neighborhood probabilities (p_1, p_2, p_3) and the timeslots maximum selection number tm are used to define the neighborhoods. To investigate the effects of these parameters and their interaction effects, we performed a 2-level full factorial experiment [30] with the levels of these parameters shown in Table 5.

Table 6
Results of the analysis of variance.

Source	p-value
T_0	2.615913e-297
L	0.000000e+00
α	5.014828e-213
(p_1, p_2, p_3)	3.125439e-231
tm	0.000000e+00
α*T_0	3.126968e-233
α*L	2.143320e-263
T_0*L	9.885850e-284
α*tm	2.727317e-107
T_0*tm	6.006472e-246
L*tm	4.492402e-304
α*(p_1, p_2, p_3)	1.734318e-29
T_0*(p_1, p_2, p_3)	1.622446e-04
L*(p_1, p_2, p_3)	1.310061e-144
tm*(p_1, p_2, p_3)	5.137107e-40
α*T_0*L*tm*(p_1, p_2, p_3)	1.592908e-121

We conducted the factorial experiment on six randomly selected instances and each instance was solved 20 times independently. We recorded the average value of the best objective values found on the six instances. We applied the multivariate analysis of variance on the results obtained with the verification of normality distributions and the variance homogeneity. Table 6 provides the results of the analysis with the p-values from the F-statistic test for each parameter, all the combinations of two parameters and all five parameters. For a significance level of 0.05, a p-value less than 0.05 indicates a significant relationship between each parameter (or parameter combination) and the results. We observe that each parameter has a significant effect on the algorithm performance according to the very small p-values. This is specially true for the number of moves per temperature L and the timeslots maximum selection number tm. Moreover, the interaction effects between any two parameters are also significant, which explains why we didn't apply the one-at-a-time sensitivity analysis to determine our parameters' values. In conclusion, the MNSA algorithm is sensitive to the settings of its parameters and the interaction effects between them are significant.

6 Conclusion and Perspectives

We studied the user project planning problem in social and social-medical establishments, which is a relevant and challenging application for efficient decision-makings. The proposed multi-neighborhood simulated annealing al-

gorithm is the first local optimization algorithm which explores three complementary neighborhoods in a probabilistic manner. We designed these neighborhoods according to real planning operations and demonstrated their effectiveness on realistic benchmark instances. Within a short time, the approach can provide feasible solutions of very good quality, even for the largest instances.

This work is a part of the decision support tool "MSUsager" developed by the Company GePI [3] for the social and medico-social establishments in French. "MSUsager" provides an easy-to-use user interface to collect various data such as user preferences, activity requirements, etc, which helps decision-makers to manage complex user data efficiently. This automatic user project planning functionality can greatly facilitate the work of planners in social and medico-social establishments and help them to design high quality individualized projects for the residents of these structures.

This work provides a solution for the user project conception phase. An interesting future study would be to investigate the dynamic user project planning problem for the execution phase. Indeed, even though the project planning is validated before execution, unexpected changes happen frequently such as last-minute unavailability of resources or professionals, activity interruption caused by an adverse event, additional user needs, etc. All these events would lead to the need of adjustments of users' projects. For these situations, a new feasible planning with the smallest changes as possible is needed quickly. As a result, a fast schedule adjustment mechanism could be highly relevant to enrich the decision support system. To this end, the optimization algorithm proposed in this study can serve as a basis to build such a project planning tool.

CRediT authorship contribution statement

Yinuo Li: Conceptualization, Methodology, Software, Experiments, Data curation, Writing - original draft.
Jin-Kao Hao: Supervision, Conceptualization, Methodology, Validation, Writing-review & editing.

[3] http://www.gepi-conseil.com

Declaration of competing interest

The authors declare that they have no known competing interests that could have appeared to influence the work reported in this paper.

Acknowledgment

We are grateful to our reviewers for their comments which helped us to improve the paper. This work was partially supported by the French Ministry for Research and Education through a CIFRE grant (number 2018/0225). The authors thank M. Rachid Naitali, the General Director of GePI, for his support.

References

[1] A. M. Fathollahi-Fard, A. Ahmadi, F. Goodarzian, N. Cheikhrouhou, A bi-objective home healthcare routing and scheduling problem considering patients' satisfaction in a fuzzy environment, Applied Soft Computing 93 (2020) 106385.

[2] A. Mardani, R. E. Hooker, S. Ozkul, Y. Sun, M. Nilashi, H. Z. Sabzi, G. C. Fei, Application of decision making and fuzzy sets theory to evaluate the healthcare and medical problems: A review of three decades of research with recent developments, Expert Systems with Applications 137 (2019) 202–231.

[3] F. Zandi, A bi-level interactive decision support framework to identify data mining-oriented electronic health record architectures, Applied Soft Computing 18 (2014) 136–145.

[4] S. Wang, S. Ma, Efficient methods for a bi-objective nursing home location and allocation problem: A case study, Applied Soft Computing 65 (2018) 280–291.

[5] Y. Zhang, P. Yu, J. Shen, The benefits of introducing electronic health records in residential aged care facilities: a multiple case study, International Journal of Medical Informatics 81 (10) (2012) 690–704.

[6] ANESM, Les attentes de la personne et le projet personnalisé, Tech. rep., Agence Nationale de l'Evaluation et de la qualité des Etablissements et Services Sociaux et Médico-sociaux (2008).

[7] CEDIS, Guide d'elaboration du projet d'accompagnement personnalisé, Tech. rep., Le Comité Européen pour le Développement de l'Intégration Sociale (2012).

[8] Y. Li, J.-K. Hao, B. Chabane, User project planning in social and medico-social sector: models and solution methods, Expert Systems with Applications 173 (2021) 114684.

[9] B. Khelifa, M. R. Laouar, A holonic intelligent decision support system for urban project planning by ant colony optimization algorithm, Applied Soft Computing 96 (2020) 106621.

[10] C. Heimerl, R. Kolisch, Scheduling and staffing multiple projects with a multi-skilled workforce, OR Spectrum 32 (2) (2010) 343–368.

[11] M. Bassett, Assigning projects to optimize the utilization of employees' time and expertise, Computers & Chemical Engineering 24 (2-7) (2000) 1013–1021.

[12] D. P. Ballou, G. K. Tayi, A decision aid for the selection and scheduling of software maintenance projects, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 26 (2) (1996) 203–212.

[13] P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch, Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research 112 (1) (1999) 3–41.

[14] M. Davari, E. Demeulemeester, The proactive and reactive resource-constrained project scheduling problem, Journal of Scheduling 22 (2) (2019) 211–237.

[15] S. Hartmann, D. Briskorn, A survey of variants and extensions of the resource-constrained project scheduling problem, European Journal of Operational Research 207 (1) (2010) 1–14.

[16] F. Stork, Stochastic resource-constrained project scheduling, Doctoral thesis, Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften, Berlin, Germany (2001).

[17] Z. T. Kosztyán, Exact algorithm for matrix-based project planning problems, Expert Systems with Applications 42 (9) (2015) 4460–4473.

[18] Z. T. Kosztyán, An exact algorithm for the flexible multilevel project scheduling problem, Expert Systems with Applications 158 (2020) 113485.

[19] O. Bellenguez-Morineau, E. Néron, A branch-and-bound method for solving multi-skill project scheduling problem, RAIRO-Operations Research 41 (2) (2007) 155–170.

[20] T. Ahn, S. S. Erenguc, The resource constrained project scheduling problem with multiple crashable modes: a heuristic procedure, European Journal of Operational Research 107 (2) (1998) 250–259.

[21] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, M. Denk, Competence-driven project portfolio selection, scheduling and staff assignment, Central European Journal of Operations Research 16 (3) (2008) 281–306.

[22] T. Servranckx, M. Vanhoucke, A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs, European Journal of Operational Research 273 (3) (2019) 841–860.

[23] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[24] K. Proos, G. Steven, O. Querin, Y. Xie, Multicriterion evolutionary structural optimization using the weighting and the global criterion methods, AIAA Journal 39 (10) (2001) 2006–2012.

[25] Z. Lü, J.-K. Hao, Adaptive tabu search for course timetabling, European Journal of Operational Research 200 (1) (2010) 235–244.

[26] B. Laurent, J.-K. Hao, Simultaneous vehicle and driver scheduling: A case study in a limousine rental company, Computers & Industrial Engineering 53 (3) (2007) 542–558.

[27] Z. Lü, J.-K. Hao, F. W. Glover, Neighborhood analysis: a case study on curriculum-based course timetabling, Journal of Heuristics 17 (2) (2011) 97–118.

[28] M. López-Ibánez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[29] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2) (2002) 201–213.

[30] D. C. Montgomery, Design and analysis of experiments, John Wiley & Sons, 2017.

A Appendix: 0/1 programming model for the user project planning problem

This appendix presents the 0/1 programming model introduced in [8] for the user project planning problem. This model was used to run the CPLEX MIP solver to the 20 benchmark instances whose results are reported in Section 4.3.

Decision variables

- x_{ijk} Binary variable taking value 1 if user i starts activity k at timeslot j, and 0 otherwise.
- y_{ijk} Binary variable taking value 1 if resource i starts to be used by activity k at timeslot j, and 0 otherwise.

$$Maximize \quad f(S) = w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S) \tag{A.1}$$

where

$$f_1(S) = \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} x_{uta} \times pre_{ua} \tag{A.2}$$

$$f_2(S) = \sum_{u \in \mathcal{U}} bud_u - \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} x_{uta} \times pri_a \tag{A.3}$$

$$f_3(S) = R \times T - \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} y_{rta} \times dur_a + \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} ra_{rt} \tag{A.4}$$

Subject to

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ x_{uda} = 0, if\ ua_{ut} = -1 \tag{A.5}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \sum_{a \in \mathcal{A}} x_{uta} \leq 1 \tag{A.6}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t + 1, \ldots, t + dur_a - 1\}, \\ x_{uda} = 0, if\ x_{uta} = 1 \tag{A.7}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ x_{uda} = 0, if\ aa_{at} = -1 \tag{A.8}$$

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ y_{rda} = 0, if\ aa_{at} = -1 \tag{A.9}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ (x_{uta} \times ((t \div TS) == ((t + dur_a - 1) \div TS)) > 0) \vee x_{uta} = 0 \tag{A.10}$$

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ (y_{rta} \times ((t \div TS) == ((t + dur_a - 1) \div TS)) > 0) \vee y_{rta} = 0 \tag{A.11}$$

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t - dur_a + 1, \ldots, t\}, \\ y_{rda} = 0, if\ ua_{ut} = -1 \tag{A.12}$$

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall d \in \{t + 1, \ldots, t + dur_a - 1\}, \\ y_{rda} = 0, if\ y_{rta} = 1 \tag{A.13}$$

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \sum_{a \in \mathcal{A}} y_{rta} \leq 1 \tag{A.14}$$

$$\forall u \in \mathcal{U}, \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} pri_a \times x_{uta} \leq bud_u \tag{A.15}$$

$$\forall a \in \mathcal{A}, \forall t \in \mathcal{T}, \sum_{u \in \mathcal{U}} x_{uta} \leq cap_a \tag{A.16}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, \forall f \in \mathcal{F}, x_{uta} \times req_{af} - \sum_{r \in \mathcal{R}} y_{rta} \times fea_{rf} \leq 0 \tag{A.17}$$

$$\forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \forall a \in \mathcal{A}, x_{uta} = 0 \vee x_{uta} \times pre_{ua} > 0 \tag{A.18}$$

$$\forall u \in \mathcal{U}, \forall a \in \mathcal{A}, \sum_{t \in \mathcal{T}} x_{uta} \leq 1 \tag{A.19}$$

Equation (A.1) is the objective function combining sub-objectives (A.2)-(A.4). Formulas (A.5)-(A.7) express constraint C_1, formulas (A.8)-(A.11) define constraint C_2, expressions (A.12)-(A.14) model constraint C_3. Constraint (A.19)

ensures that a user can only participate in an activity at most once. Formulas (A.15-A.18) are expression of constraints C_4, C_5, C_6, C_7, respectively.