POLITECNICO DI TORINO
Repository ISTITUZIONALE

Rescheduling rehabilitation sessions with answer set programming

Original
Rescheduling rehabilitation sessions with answer set programming / Cardellini, M., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Nisopoli, N., Porro, I.. - In: JOURNAL OF LOGIC AND COMPUTATION. - ISSN 0955-792X. - 33:4(2023), pp. 837-863. [10.1093/logcom/exad014]

Availability:
This version is available at: 11583/2977729 since: 2023-04-03T12:25:43Z

Publisher:
Oxford University Press

Published
DOI:10.1093/logcom/exad014

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Oxford University Press preprint/submitted version

This article has been accepted for publication in JOURNAL OF LOGIC AND COMPUTATION Published by Oxford University Press

(Article begins on next page)

28 November 2026

Rescheduling Rehabilitation Sessions with Answer Set Programming *

Matteo Cardellini[1,2][0000−0003−3788−9475], Carmine Dodaro[3][0000−0002−5617−5286], Giuseppe Galatà[4][0000−0002−1948−4469], Anna Giardini[5], Marco Maratea[2,3][0000−0002−9034−2527]**, Nicholas Nisopoli[4], and Ivan Porro[4][0000−0002−0601−8071]

[1] Polytechnic of Torino, Italy; matteo.cardellini@polito.it
[2] DIBRIS, University of Genova, Genova, Italy; marco.maratea@unige.it
[3] DeMaCS, University of Calabria, Rende, Italy; {maratea,dodaro}@mat.unical.it
[4] SurgiQ srl, Italy; {name.surname}@surgiq.com
[5] IT, ICS Maugeri, Italy; anna.giardini@icsmaugeri.it

Abstract. The rehabilitation scheduling process consists of planning rehabilitation physiotherapy sessions for patients, by assigning proper operators to them in a certain time slot of a given day, taking into account several requirements and optimizations, e.g., patient's preferences and operator's work balancing. Being able to efficiently solve such problem is of upmost importance, in particular as a consequence of the COVID-19 pandemic that significantly increased rehabilitation's needs. The problem has been recently successfully solved via a two-phase solution based on Answer Set Programming (ASP).
In this paper, we focus on the problem of rescheduling the rehabilitation sessions, that comes into play when the original schedule cannot be implemented, for reasons that involve the unavailability of operators and/or the absence of patients. We provide rescheduling solutions based on ASP for both phases, considering different scenarios. Results of experiments performed on real benchmarks, provided by ICS Maugeri, show that also the rescheduling problem can be solved in a satisfactory way. Finally, we present a web application that supports the usage of our solution.

1 Introduction

The rehabilitation scheduling process [32, 33, 35, 42] (RSP) consists of planning patients' physiotherapy sessions inside a rehabilitation institute. Hospitals that may profitably make a practical use of such scheduling, including those managed by ICS Maugeri[6] that will provide benchmarks and problem specifications in this paper, deal with up to hundreds of patients with a team of just few tens of physiotherapists; so, it is of paramount importance to be able to assign patients

* This is an extended and revised version of a paper presented at the 36th Italian Conference on Computational Logic (CILC 2021).
** Corresponding author.
[6] https://www.icsmaugeri.it/.

to operators efficiently. A recent article [15] found that 2.41 billion people could benefit from rehabilitation services. This finding means that almost one third of the people in the world needs rehabilitation at some point during the course of their lives for disease or injury; further, this number is predicted to trend upward given the current demographic and health shifts. In addition, there is emerging evidence that many of the people affected by the COVID-19 pandemic have long-term consequences regardless of the disease severity or length of hospitalisation, thus further increasing the demand for rehabilitation services globally. The RSP is subject to several constraints, i.e., legal, medical and ethical, that need to be taken into consideration in order to find a viable schedule. For example, the main constraints that have to be dealt with are the maximum capacity of rehabilitation gyms, the legal working time and rest periods for operators, and the minimum durations of physiotherapy sessions. Moreover, several preferences shall be considered, e.g., due to clinical and organizational reasons it is often best for the patient to be treated as often as possible by the same operator and defined slots for the rehabilitation sessions are to be preferred; also, rehabilitation professionals' work balancing needs to be taken into proper account.

In this paper, we build on a solution to the RSP based on Answer Set Programming (ASP) [8, 10, 31, 38], designed as a two-phase encoding, presented in a previous version of this paper [13]. In particular, the first phase is called *board* and deals with the problem of assigning a physiotherapist to every patient, while in the second phase, called *agenda*, a start and end time of every rehabilitation session is defined given the assignment between patients and physiotherapists found in the first phase. As already noticed in [13], our two-phase solution is not guaranteed to find the best possible overall solution, but has been designed in this way mainly because it mimics how schedules have been computed so far (in a non-automatic way) by ICS Maugeri, and gives freedom to physiotherapists' coordinators to perform any desired manual change to the board, before planning the agenda. Then, we focus on the problem of rescheduling the rehabilitation sessions, that comes into play when the original schedule cannot be implemented due to, e.g., unavailability of operators and/or absence of patients. The presented Rescheduling RSP (RRSP) encodings receive as input the previous scheduling and the unavailability of operators and/or patients, and provide as output novel solutions taking into account such unavailability. Rescheduling solutions consider a wide spectrum of scenarios involving total or partial unavailability of operators and/or patients.

Then, we tested our encoding on real benchmarks supplied by ICS Maugeri, who provided also the problem specifications, on the aforementioned scenarios, and related to the daily scheduling of neurological patients from two of their rehabilitation institutes in the North of Italy, namely Genova Nervi and Castel Goffredo. Results using the ASP solver CLINGO [27] are satisfying for the institutes considered at the moment: for the board phase, we are able to find an optimal solution in short time in the majority of the cases, while for the agenda phase, whose problem and encoding are more elaborated, by increasing the number of operators and/or patients unavailable the quality of the solution

decreases and the time increases, but still obtaining satisfying results even for our largest benchmarks. Additionally, we have also designed and implemented a web framework for managing rehabilitation problems via ASP that allows a user to insert the main parameters of the problem, solve a specific instance, and that shows results graphically in real-time.

The paper is structured as follows. Section 2 describes the problem we solve together with its main elements, while Section 3 introduces preliminaries about ASP. Then, Section 4 reviews the scheduling encodings, while Section 5 illustrates encodings for the rescheduling problem, whose results are presented in Section 6. Finally, the web application for usability of our solution is described in Section 7. The paper ends with related work discussion and conclusions in Section 8 and 9, respectively.

2 Problem description

The delivery of rehabilitation services is a complex task that involves many healthcare professions such as physicians, physiotherapists, speech therapists, psychologists, and so on. In particular, physiotherapists spend most of their time with the patients and their sessions constitute the core of the daily agenda of the patient, around which all other commitments revolve. We thus focus on scheduling and rescheduling the physiotherapy sessions in the most efficient way, optimising the overall time spent with the patients.

The agenda for the physiotherapy sessions is computed by the coordinator of the physiotherapists. This process is repeated on a daily basis in order to take into account any change in the number and type of patients to be treated, and the number of operators available; until recently, this operation has been performed manually in ICS Maugeri, without any automation.

In the following subsections, we describe in more details the main elements of the problem, namely patients, operators and sessions; the constraints and preferences entailed by the board and agenda phases; and, finally, the scheduling and rescheduling problems we deal with in this paper. These specifications have been provided by ICS Maugeri.

2.1 Main elements of the problem

In the following paragraphs we discuss patients, operators, and sessions.

Patients. Patients are characterized by their:

- type (Neurological, Orthopaedic, Alcoholic, COVID-19 Positive, COVID-19 Negative, Outpatient),
- aid needs, i.e., if they need specific care or not,
- payment status (full payer or in charge of the National Healthcare Service),
- forbidden times, i.e., the time intervals when the patient cannot be scheduled,
- ideal time, i.e., the preferred scheduled time expressed by the coordinator,

- preferred operators, i.e., the list of physiotherapists, ordered by priority, the patient can be assigned to,
- overall minimum length, i.e., the minimum amount of care time that the patient is guaranteed to be scheduled,
- sessions, i.e., the list of sessions to be scheduled.

Operators. Physiotherapists, that will be called operators from now on, are characterized by their:

- qualifications, i.e., patient's types the operator can treat,
- operating times, i.e., the part of the operator's working times devoted to the direct care of patients. The operating times are usually split in morning and afternoon shifts.

Sessions. The coordinator, in accordance with the rehabilitation program set by the physician, determines the daily activities of the patient. These activities can be performed in one or two therapy sessions, in the latter case one session will be scheduled in the morning and the other in the afternoon shift.

Each session can be delivered to patients either by individualized ("one-to-one" sessions) or supervised (one therapist supervising more patients at the same time, each patient carrying out their personal activity independently). It must be noted that, while operators are delivering one-to-one therapy to patients, they can supervise other patients but cannot deliver one-to-one therapy to another patient. When the operators are particularly overbooked, their one-to-one sessions can be partially converted to supervised ones. These mixed sessions can either start with a supervised part and then continue with the one-to-one part, or vice-versa, or even start and end with a supervised part with a middle one-to-one session. Obviously, an operator can supervise different patients only if their sessions are located at the same place. The characteristics of sessions are:

- delivery mode (one-to-one, supervised),
- minimum one-to-one length, i.e., the minimum length of the session guaranteed to be delivered in one-to-one mode,
- ideal overall length, i.e., the overall length of the session including the one-to-one and supervised parts,
- optional status, i.e., if the session can be left out of the schedule in case of overbooked operators,
- forced time, i.e., the time when the session must be scheduled; if empty, the session is placed as close as possible to the patient's preferred time,
- location, i.e., the place where the session must be delivered.

2.2 Specifications of the two phases

In this subsection we describe, in two different paragraphs, the constraints and preferences considered in the two phases of our approach.

Board. In the board phase all patients are assigned to an available operator, according to the following criteria:

- compatibility between patient and operator, depending on the patient's type and operator qualifications, the patient's forced time, if any, and the operator working times, by also checking if the operator has enough time to provide the guaranteed overall minimum length and minimum one-to-one length to each patient and session,
- forced assignments of a patient to an operator: in special cases, the coordinator can override the preferred operators list and force an assignment regardless of all other considerations,
- the patients should be fairly distributed among all available operators, taking into account their type, aid needs and payment status,
- the patients should be assigned to the operators respecting as much as possible their preferred operators list, which considers primarily the choices of the coordinator and secondarily the history of the past assignments.

Agenda. The results of the board phase can be revised and, if necessary, manually modified by the coordinator. Once the coordinator is satisfied with the board, it is possible to proceed to the agenda scheduling, using the approved board as input data. The criteria for the agenda phase are:

- compliance with the forced time of the session, if specified,
- two sessions of the same patient must be assigned in different shifts,
- compliance with the minimum one-to-one length of the session,
- no overlap between two one-to-one sessions (or their one-to-one sections if the sessions are mixed) assigned to the same operator,
- observance of the maximum capacity of the locations (1 for each room, varying for the gyms),
- respect of the overall minimum length of the patient,
- respect of the one-to-one minimum session length,
- compliance with the forbidden times of the patient,
- sessions can only be scheduled within the working times of the operator,
- the start time of each session should be as close as possible to the preferred time, either specified by the coordinator or inferred from previous schedules,
- for mixed sessions, the one-to-one part should be maximized,
- the largest possible number of optional sessions should be included,
- the overall length, including the one-to-one and supervised parts in case of mixed sessions, should be as close as possible to the ideal overall length specified by the coordinator.

2.3 RSP

In this subsection, we describe the Rehabilitation Scheduling Problem (RSP) we solve via the two phases.

The board phase deals with the problem of assigning a physiotherapist to every patient, keeping track of the total working time of the operator and the

minimum mandatory time of rehabilitation sessions. Then, in the consequential agenda phase, a start and end time of every rehabilitation session is searched given the assignments between patients and operators found in the first phase. Going more in details, in the board phase the working hours of operators are simplified by counting their total working time, in minutes, and assigning patients to each operator in order to keep the cumulative time of all the sessions in which the operators are involved underneath their total working time. In this phase, patient-operator assignment preferences, expressed by the coordinator before the start of the scheduling procedure, are taken into account and respected as far as possible. In the agenda phase, given an assignment found by the board, every patient-operator session is assigned a starting and ending time, respecting the more granular working hours of the operators and the times in which the patients are unavailable. At this stage, the location in which the rehabilitation session is performed is also considered: a location (either a gym or the room of the patient) is assigned to every session, keeping into consideration the maximum number of simultaneous sessions allowed inside the location. A macro-location is a container of equivalent locations. For example, two gyms on the same floor in Castel Goffredo are considered equivalent, and thus in the same macro-location, because any patient having a room on the same floor can be assigned to any of them indifferently but not to gyms on other floors. The choice of the gym has to be made among a subset of gyms that are located in the same macro-location in order to avoid elevators and stairs that can result in discomfort to patients and can quickly congest the hospital. In this phase, time preferences for each patient are also considered: in fact, solutions in which the sessions are performed nearer the desired time of the patients are preferred to the others.

Note that our two-phase solution is not guaranteed to find the best possible overall solution, but has been designed in this way mainly because it mimics how schedules have been computed so far (in a non-automatic way) by ICS Maugeri, and gives freedom to physiotherapists' coordinators to perform any desired manual change to the board, before planning the agenda.

2.4 Rescheduling RSP

In this subsection, we informally present the Rescheduling Rehabilitation Sessions Problem (RRSP).

Rescheduling is a procedure that may be applied to correct a previously approved schedule whenever an accident, or any unexpected event that makes the previous schedule not applicable, happens. The types of unexpected events that we have dealt with in the present work are:

- full or partial unavailability of one or more operators,
- full or partial absence of one or more patients,
- a combination of both absences.

Note that in our model the absence of an operator does not determine nor influence the absence of a patient and vice-versa, thus even when occurring in the same rescheduling problem they can be treated as independent events.

The guiding principle of rescheduling is to adjust the schedule for the unexpected events in a way that minimizes the differences with the original schedule. Like the scheduling problem, the rescheduling problem is subdivided in two phases: board and agenda.

When an operator is unavailable for a full day, their patient must be reassigned to other operators. This reassignment procedure should respect all the criteria followed in the original scheduling, such as suitable qualifications, preferences, and so on. Moreover, a new criterion is necessary: in order to not overburden the operators, the patients are reassigned preferentially to the operators with less occupied time. After the board, also the agenda has to be rescheduled. In this case, the guiding criteria are:

- the sessions of the patients who were not reassigned should keep their original times,
- the sessions of the reassigned patients should be placed in a time as close as possible to their previously scheduled ones,
- it should be possible to convert, fully or partially, an individual session belonging to a reassigned patient to a supervised one if necessary (e.g., the operator, to whom the patient was reassigned to, has no free time at all).

In case of partial unavailability of operators, their patients should preferably remain with the operator and moved in time. However, it should be possible to reassign some of them to other operators if the operating time of the partially unavailable operator is not sufficient.

In the case of full absence of patients, the rescheduling of the board is easy, since it only involves the elimination of the sessions of the missing patients. When the absence is only partial, e.g., a patient previously scheduled in the morning is now available only in the afternoon, the board should be affected only when the operator assigned to the patient is not working at the new time slot of patient availability, e.g., a part-timer who only works in the mornings. In this case, the patient should be moved to a different operator. The agenda of operators of fully absent patients should be rescheduled so that the sessions of the remaining patients are moved as close as possible to their preferred times. If a partially absent patient has to be moved to another operator, then the procedure applied is analogous to the one described below for the operator unavailability cases.

2.5 Examples of (re)scheduling

In this section, we present some examples of the scheduling of scenarios coming from real data in the hospitals of Genova Nervi, and the consequent rescheduling caused by absence of patients and/or unavailability of operators. These results will be presented for both the board and the agenda phases.

Board example. Figure 1 (top) shows a bar plot representing the distribution of patients among operators found in the board phase of a real case scenario coming from the hospital of Genova Nervi. In the scenario, 65 patients have to

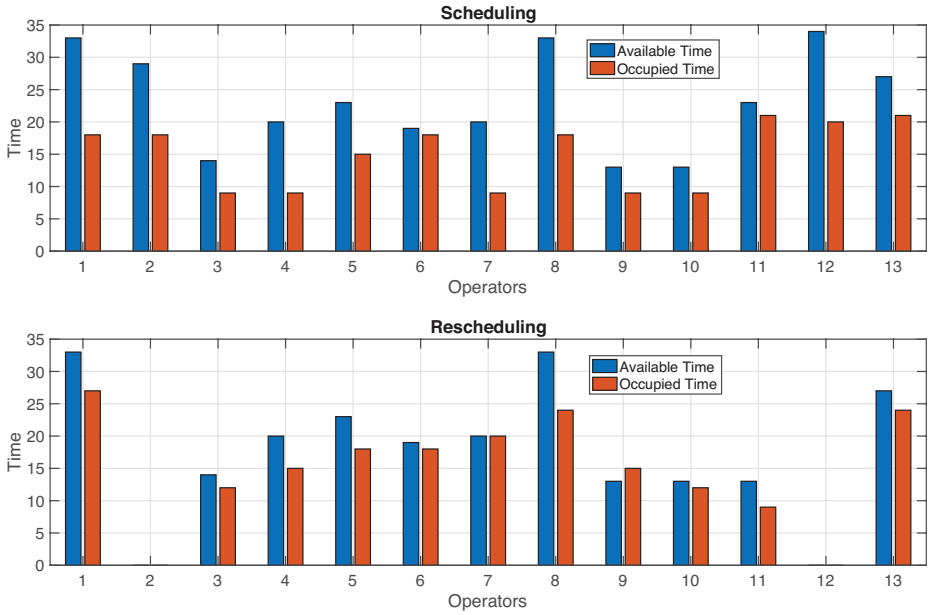


Fig. 1: Bar plots represent the distribution of patients among 13 operators in the Genova Nervi hospital (top) and the consequent rescheduling (bottom) caused by the total unavailability of operators `2` and `12`, and the partial unavailability of 10 time slots for operator `11`. Blue bars represent the amount of time slots in which the operators are available, while red bars represent the sum of the minimum individual time of all the sessions assigned to the operator.

be distributed among 13 operators. Blue bars represent the amount of time slots in which the operators are available, red bars represent the sum of the minimum individual time of all the sessions assigned to the operator. Figure 1 (bottom) shows the rescheduling caused by the total unavailability of operators `2` and `12`, and the partial unavailability for 10 time slots of operator `11`. As it can be seen, the patients have been redistributed among the remaining operators and the partial unavailability of operator `11` has been respected by removing some patients which were previously assigned to it. It can be noted that, unfortunately, now operator `9` has a number of sessions assigned in which the sum of the minimum individual time (red bar) is greater than the available time of the operator (blue bar). Nevertheless, this result is still valid because, by Maugeri's policy, if not enough time is available for operators to perform rescheduled sessions individually, then these sessions will be performed in a supervised fashion, which, although not optimal, can still be beneficial to the patient.

Agenda example. Figure 2 shows the scheduling of the agenda in a real case scenario in the hospital of Genova Nervi. Light blue squares represent time units in which the sessions will be performed in an individual fashion, and yellow squares

represent time units of sessions (or extensions of a session) in which the patient will be dealt in a supervised mode. Some sessions (e.g., session `17` of patient `13` performed in the morning by operator `4`) are performed in a complete supervised fashion since they are additional secondary sessions which are medically beneficial to the patient but not mandatory (e.g., patient `13` already performs session `16` in an individual fashion with operator `4` in the afternoon). Figure 3 shows the rescheduling caused by (*i*) the total absence of patient `61`, (*ii*) the partial unavailability of operator `10` in the morning, and (*iii*) the total unavailability of patient `12` (actually, the unavailability is set only in the afternoon, but it coincides exactly with the operator working time which works only part-time). Gray boxes delimit the slots in which operators are not available. As it can be noted by comparing Figure 2 and 3, the sessions which were, in the scheduling, inside the gray boxes are now assigned to other operators trying to keep the schedule as close as possible to the original one. Moreover, it can be seen how some reassigned sessions have a large amount of extended supervised time w.r.t. the original schedule, given the impossibility, as stated in the previous paragraph, to fit all the rescheduled sessions without reducing the sessions untouched by the unavailability of operators. Regarding the absence of patient `61`, it can be noted how session `80` – which was assigned in the schedule to operator `14` in the morning – is now not present anymore in the final reschedule, and this has given the possibility to the other sessions of the operator to fill the empty space and obtaining more individual session time.

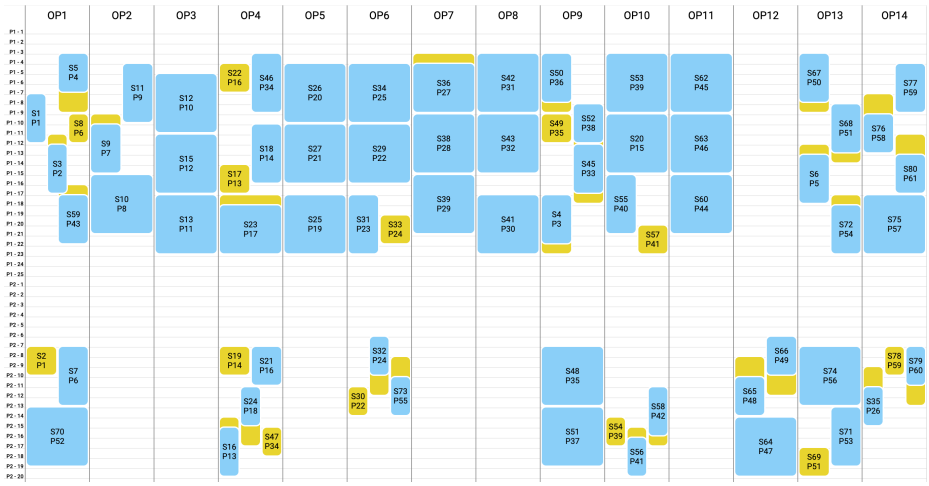


Fig. 2: Result of the scheduling of the agenda in a real case scenario in the hospital of Genova Nervi. Light blue (yellow) squares represent time units in which the sessions will be performed in an individual (supervised) fashion. The ticks on the left keep track of the period and time slot in which the session will start or end.

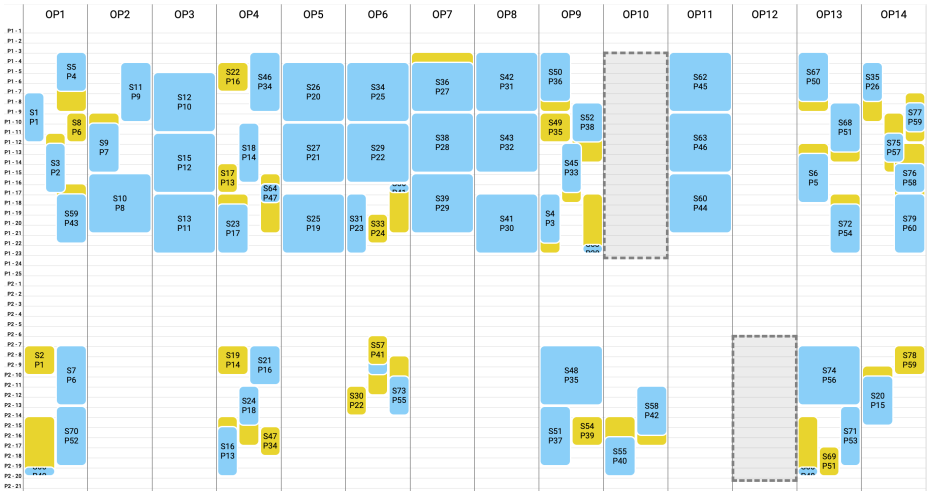


Fig. 3: Result of the rescheduling of the agenda caused by a partial unavailability of operator `10`, the total unavailability of operator `11`, and the total absence of patient `61`. Gray boxes signal the timeslots in which the operators are not available.

3 Background on ASP

Answer Set Programming (ASP) [10, 8, 31, 38] is a programming paradigm developed in the field of non-monotonic reasoning and logic programming. In this section, we overview the language of ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [12]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with an uppercase letter, and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts : conj \rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X, c), p(X); Y : b(Y, m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making

the atom $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions implemented in ASP systems are the following:

- $\#count$, number of terms;
- $\#sum$, sum of integers.

An *aggregate atom* is of the form $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{<, \leq, >, \geq, \neq, =\}$ is a comparison operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r has the following form:

$$a_1 \mid \ldots \mid a_n \text{ :- } b_1, \ldots, b_k, \mathit{not}\ b_{k+1}, \ldots, \mathit{not}\ b_m.$$

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation $\mathit{not}\ a$. The disjunction $a_1 \mid \ldots \mid a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, \mathit{not}\ b_{k+1}, \ldots, \mathit{not}\ b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules, where a rule r is *safe* if the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ; and *(ii)* each local variable of r appearing in a symbolic set $\{\mathit{Terms} : \mathit{Conj}\}$ also appears in a positive atom in Conj.

A *weak constraint* [11] ω is of the form:

$$:\sim b_1, \ldots, b_k, \mathit{not}\ b_{k+1}, \ldots, \mathit{not}\ b_m.\ [w@l, t_1, \ldots, t_z]$$

where $t_1, \ldots, t_z$ are terms, w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read "as weight w at level l", where weight is the "cost" of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). Intuitively, $t_1, \ldots, t_z$ are used to distinguish ground weak constraints between each other. An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints.

A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variables appear in it.

Semantics. Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual. The ground instantiation G_P of P is the set of all the ground instances of rules of P that can be obtained by substituting variables with constants from U_P.

An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., $\mathit{not}\ \ell$) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false (resp., true) otherwise. An

aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on the multiset S) w.r.t. I satisfies the guard; otherwise, it is false.

A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I.

A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [23] of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) [23].

Given a program with weak constraints $\Pi = \langle P, W \rangle$ and an interpretation I, the semantics of Π extends from the basic case defined above. Thus, let G_P and G_W be the instantiation of P and W, respectively. Then, let G'_W be the set $\{(w@l, t_1, \ldots, t_z) \mid :\sim b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m.\ [w@l, t_1, \ldots, t_z] \in G_W\ and\ b_1, \ldots, b_m \in I\}$. Moreover, for an integer l, $P_l^I = \sum_{(w@l,t_1,\ldots,t_z)\in G'_W} w$ if there is at least one tuple in G'_W whose level is equal to l, and 0 otherwise.

Given a program with weak constraints $\Pi = \langle P, W \rangle$, an answer set M for P is said to be dominated by an answer set M' for P, if there exists an integer l such that $P_l^{M'} < P_l^M$ and $P_{l'}^{M'} = P_{l'}^M$ for all integers $l' > l$. An answer set M for P is said to be *optimal* or *optimum* for Π if there is no other answer set M' that dominates M.

Syntactic shortcuts. In the following, we also use *choice rules* of the form $\{p\}$, where p is an atom. Choice rules can be viewed as a syntactic shortcut for the rule $p \mid p'$, where p' is a fresh new atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

4 A Two-Phase ASP Encoding for the RSP

In this section, starting from the specifications in Section 2, we present the ASP encoding of the scheduling for both phases, based on the input language of CLINGO [25], that will be the starting point for the rescheduling solutions in the next section. A subsection is devoted to each phase.

4.1 Board encoding

Data Model. The input data is specified by means of the following atoms:

- Instances of `patient(P)`, `operators(O)` and `type(T)` represent the identifiers of patients, operators, and the different types of patients that can be visited, respectively, where `P` and `O` are numbers, whereas `T` can be: *neurologic*, *neurologic-lifter*, *orthopaedic*, *orthopaedic-lifter*, *covid-19-positive*, *covid-19-negative*, or *outpatient*. Moreover, a fictitious operator with `ID` equals to -1 is included in the list of all the operators, and is needed to intercept the patients that cannot be assigned to other operators.

```
1 {assignment(OP, PAT) : operator(OP)} = 1 :- patient(PAT).
2 uniqueLocationLength(OP,PAT,DUR) :- assignment(OP,PAT), patient_session(PAT,_,LOC),
      patient_data(PAT,_,DUR), #count{ID:patient_session(ID,_,LOC), assignment(OP,ID)} < 2.
3 sameLocationLength(OP,PAT,DUR) :- assignment(OP,PAT), patient_session(PAT,DUR,LOC),
      #count{ID:patient_session(ID,_,LOC), assignment(OP,ID)} > 1.
4 :- operator_contract(OP,TIME,_), #sum{U,PAT:uniqueLocationLength(OP,PAT,U); S,
      PAT:sameLocationLength(OP,PAT,S)} > TIME.
5 :- operator_contract(OP,_,N), #count{PAT:assignment(OP,PAT)} > N.
6 :- operator_limit(OP,T,N), #count{PAT:assignment(OP,PAT), patient_data(PAT,T,_} > N.
7 :~ #sum{W, PAT:assignment(OP,PAT), patient_preference(PAT,OP,W)} = N. [N@3]
8 :~ #count{PAT: assignment(-1, PAT)} = N. [N@2]
9 :~ #sum{W, PAT:assignment(OP,PAT), history_preference(PAT,OP,W)} = N. [N@1]
```

Fig. 4: ASP encoding for the board problem.

- Instances of `operator_contract(ID,TIME,MAX)` represent the contract of the operator with identifier `ID`, and include the quantity of time (in time units) the operator works in a day (`TIME`), and the maximum number of patients the operator can visit during the day (`MAX`).
- Instances of `operator_limit(ID,T,VALUE)` represent the maximum number of patients (`VALUE`) of type `T` the operator with identifier `ID` can visit. The operator with `ID` equals to -1 has no patients limits.
- Instances of `patient_data(ID,T,MIN)` represent the data associated to the patient with the identifier `ID`, and include the type of the patient (`T`), and the minimum cumulative time of all sessions of the patient during the day (`MIN`).
- Instances of `patient_session(ID,MIN,LOC)` represent a rehabilitation session that the patient with identifier `ID` needs to perform during the day. The session is characterized by a minimum length for the session in time units (`MIN`), and the location of the session (`LOC`).
- Instances of `patient_preference(ID,OP,W)` represent the preference of the patient with identifier `ID` to be treated by the operator with identifier `OP`, where `W` specifies the weight of the preference.
- Similarly, instances of `history_preference(ID,OP,W)` represent the preference of the patient based on the history of previous sessions.

The output is an assignment represented by atoms of the form `assignment(OP, PAT)` stating that the patient `PAT` will be treated by the operator `OP`.

Encoding. The related encoding is shown in Figure 4, and is described in the following. To simplify the description, the rule appearing at line i in Figure 4 is denoted with r_i. Rule r_1 ensures that each patient is assigned to exactly one operator. Rules r_2 and r_3 are used to define if the session between a patient and an operator will be performed individually in a single location (r_2), or it will be executed in the same location of another session (r_3). Rule r_4 ensures that the time required by the patients assigned to an operator does not exceed the maximum time of her/his contract. Rule r_5 ensures that each operator does not exceed the maximum number of patients to visit during the day. Rule r_6 is

similar to the previous one, but in this case the limits are imposed according to the type of the patient.

Weak constraints from r_7 to r_9 are then used to provide preferences among different models, having levels in this order. In particular, r_7 is used to maximize the assignments that fulfil the preferences of each patient. Then, r_8 is used to minimize the number of patients that are assigned to the fictitious operator. Finally, r_9 is used to maximize the solutions that preserve assignments dictated by the history of previous sessions.

4.2 Agenda encoding

Data Model. The following atoms constitute the input data:

- Instances of `patient(ID,MIN)` represent a patient identified by `ID`, and a minimum rehabilitation session of `MIN` length in time units that the patient has to undertake during the day.
- Instances of `period(PER,OP,STA,END)` define the start (`STA`) and end (`END`) time in the period `PER` (which can be *morning* or *afternoon*), which corresponds to the shift, of the operator with identifier `OP`.
- Instances of `time(PER,OP,T)` define the time slots `T` during the period `PER` where the operator `OP` works. In particular, `T` ranges from `STA` to `END` defined for instances of `period(PER,OP,STA,END)`.
- Instances of `location(ID,CAP,PER,STA,END)` represent a location, with an identifier `ID`, a maximum capacity of `CAP`, and during the period `PER` is open from the time unit `STA` until `END`.
- Instances of `macro_location(MLOC,LOC)` define that the location `LOC` is inside the macro-location `MLOC`.
- Instances of `session(ID,PAT,OP)` represent a session between the patient `PAT` and the operator `OP`, coming from the `assignment(OP,PAT)` output of the board phase to which a unique `ID` is added (to discriminate between *morning* and *afternoon* shifts).
- Instances of `session_type(ID,OP,TYPE)` represent that the session with identifier `ID` assigned to operator `OP` is of type `TYPE` (which can be *individual* or *supervised*).
- Instances of `session_macro_location(ID,MLOC)` represent that the session with identifier `ID` has to be held in the macro-location `MLOC`.
- Instances of `session_length(ID,MIN,IDEAL)` represent that the session `ID` has a minimum length (`MIN`) that has to be performed in individual, and an ideal length (`IDEAL`) that would be beneficial to the patient, but it is not mandatory to perform.
- Instances of `mandatory_session(ID)` and `optional_session(ID)` identify sessions that are mandatory and optional, respectively.
- Instances of `forbidden(PAT,PER,STA,END)` represent an unavailability of the patient `PAT` in the period `PER` from the time unit `STA` to `END`.
- Instances of `session_preference(ID,PER,START,TYPE)` represent the preference of the patient, stating that the session should be held during the

```
1  {start(ID,PER,TS) : time(PER,OP,TS)} = 1 :- session(ID,_,OP), mandatory_session(ID).
2  {start(ID,PER,TS) : time(PER,OP,TS)} <= 1 :- session(ID,_,OP), optional_session(ID).
3  {length(ID,PER,NL) : time(PER,OP,L), NL=L-ST, TS+NL <= END, NL>= MIN, NL<= IDEAL} = 1 :-
       start(ID,PER,TS), period(PER,OP,ST,END), session(ID,_,OP),
       session_length(ID,MIN,IDEAL).
4  {session_location(ID,LOC): macro_location(MAC,LOC)} = 1 :- session_macro_location(ID,MAC).
5  {before(ID,NL): time(PER,OP,L), NL=L-ST, NL<=TS-ST} = 1 :- start(ID,PER,TS),
       period(PER,OP,ST,_), session(ID,_,OP).
6  {after(ID,NL): time(PER,OP,L), NL=L-ST, NL<=END-TS-LEN} = 1 :- start(ID,PER,TS),
       period(PER,OP,ST,END), length(ID,PER,LEN), session(ID,_,OP).
7  ext_start(ID,PER,TS-LB) :- start(ID,PER,TS), before(ID,LB).
8  ext_length(ID,PER,L+LA+LB) :- length(ID,PER,L), after(ID,LA), before(ID,LB).
9  individual_session_location(ID,LOC,OP,MIN,IDEAL) :- session_type(ID,OP,individual),
       session_location(ID,LOC), session_length(ID,MIN,IDEAL).
10 session_time(ID,OP,PL,PER,TS..TS+L-1) :- session(ID,_,OP), session_location(ID,PL),
       ext_start(ID,PER,TS), ext_length(ID,PER,L).
11 :- start(ID,PER,TS), length(ID,PER,L), session_type(ID,OP,individual), start(ID2,PER,TS2),
       session_type(ID2,OP,individual), ID!=ID2, TS2>=TS, TS2<TS+L.
12 :- session(ID1,PAT,_), session(ID2,PAT,_), start(ID1,PER,_), start(ID2,PER,_), ID1!=ID2.
13 :- individual_session_location(ID1,LOC,OP,MIN1,OPT1), length(ID1,PER,L1),
       individual_session_location(ID2,LOC,OP,MIN2,OPT2), length(ID2,PER,L2), OPT1-L1 <=
       OPT2-MIN2, OPT2-L2 <= OPT1-MIN1 , |OPT1 -L1 - OPT2 + L2| > 1.
14 :- individual_session_location(ID1,LOC,OP,MIN1,OPT1), length(ID1,PER,L1),
       individual_session_location(ID2,LOC,OP,MIN2,OPT2), length(ID2,PER,L2), OPT1-L1 >
       OPT2-MIN2, L2 > MIN2.
15 :- individual_session_location(ID1,LOC,OP,MIN1,OPT1), length(ID1,PER,L1),
       individual_session_location(ID2,LOC,OP,MIN2,OPT2), length(ID2,PER,L2), OPT1-L1 <=
       OPT2-MIN2, OPT2-L2 <= OPT1-MIN1, OPT2 < OPT1, OPT1-L1 < OPT2-L2.
16 :- session_time(ID,OP,PL,PER,T), session_time(ID2,OP,PL2,PER,T), ID != ID2, PL != PL2.
17 :- patient(PAT,MIN), #sum{LEN, ID: session(ID,PAT,_), ext_length(ID,_,LEN)} < MIN.
18 :- location(LOC,LIM,PER,ST,END), LIM>0, time(PER,_,T), T>=ST, T<END, #count{ID:
       session_time(ID,_,LOC,PER,T)} > LIM.
19 :- forbidden(PAT,PER,ST,_), session(ID,PAT,_), ext_start(ID,PER,TS), ext_length(ID,PER,L),
       ST>=TS, ST<TS+L.
20 :- forbidden(PAT,PER,_,END), session(ID,PAT,_), ext_start(ID,PER,TS), ext_length(ID,PER,L),
       END>TS, END<=TS+L.
21 :- forbidden(PAT,PER,ST,END), session(ID,PAT,_), ext_start(ID,PER,TS),
       ext_length(ID,PER,L), ST<=TS,END>TS.
22 :- time(PER,_,T), macro_location(MAC,LOC1), macro_location(MAC,LOC2),
       #sum{1,ID1:session_time(ID1,_,LOC1,PER,T); -1,ID2:session_time(ID2,_,LOC2,PER,T)} > 2.
23 :~ length(ID,_, L), session_length(ID,MIN,IDEAL), D=|L-IDEAL|. [D@6, ID]
24 :~ start(ID,PER,_), session_type(ID,_,individual), session_preference(ID,PER2,_,high),
       D=|PER-PER2|. [D@5, ID]
25 :~ start(ID,PER,TS), session_type(ID,_,individual), session_preference(ID,PER,TS2,high),
       D=|TS-TS2|. [D@4, ID]
26 :~ optional_session(ID), time(PER,_,TS), not start(ID,PER,TS). [1@3,ID]
27 :~ start(ID,PER,_), session_preference(ID,PER2,_,low), session_type(ID,_,individual),
       optional_session(ID), D=|PER-PER2|. [D@2, ID]
28 :~ start(ID,PER,TS), session_preference(ID,PER,TS2,low), session_type(ID,_,individual),
       optional_session(ID), D=|TS-TS2|. [D@1, ID]
```

Fig. 5: ASP encoding for the agenda problem.

period `PER` and it must start at the time unit `START`, where `TYPE` indicates if the preference is *high* or *low*.

The output is represented by atoms `start(ID,PER,T)`, `length(ID,PER,L)`, and `session_location(SES,LOC)`, which indicate the start, length and location of each session, respectively.

Encoding. In Figure 5 the encoding for the agenda is presented.

Rules r_1 and r_2 assign a start time to every session; for the optional session, the start atom can be unassigned. Rule r_3 defines a length for all the sessions: the session length cannot be lower than the minimum time of the session and cannot be greater than the ideal time the session should take. Rule r_4 assigns a location for each session. Rules r_5 and r_6 reserve to each session slots of time before it starts and after it ends, in which the session can be performed in a supervised fashion. These extensions cannot be longer than the difference between the maximum and the minimum length of the session.

Then, rules r_7 and r_8 define auxiliary atoms `ext_start` and `ext_length` using the slots of times reserved for the extensions. Rule r_9 defines an auxiliary atom of the form `individual_session_location(ID,LOC,OP,MIN,IDEAL)` which represents that an individual session `ID` is in the location `LOC`, is assigned to the operator `OP`, and its minimum and ideal lengths are equal to `MIN` and `IDEAL`, respectively. Rule r_{10} defines `session_time(ID,OP,PL,PER,T)` which states that during time `T` of period `PER` the session `ID` is being performed by operator `OP`.

Rule r_{11} states that two individual assignments shall not overlap. Rule r_{12} imposes that each patient is assigned to at most one session per period. Rules r_{13} through r_{15} impose that the optional individual time (i.e., the difference between the minimum length of the session and the planned length) is added fairly to all individual sessions, starting with shorter ones. Rule r_{16} imposes that for each time slot, the operator is not in two different places. Rule r_{17} states that patients must have their minimum time reserved. Rule r_{18} imposes a limit on the concurrent use of locations with limited capacity. Rules from r_{19} to r_{21} impose that a session cannot happen during a forbidden time. Rule r_{22} avoids that, during a time slot, the distribution of sessions between each pair of locations inside the same macro location is unfair (i.e., a location is at its full capacity while another is empty).

Weak constraints from r_{23} to r_{28} represent preferences, in a prioritized way. The weak constraint r_{23} states that each session duration should be as close as possible to the ideal duration. Rules r_{24} and r_{25} minimize the distance between the actual and the preferred starting time for the *high* session priority preferences. Rule r_{26} maximizes the number of optional sessions included in the scheduling. Rules r_{27} and r_{28} are similar to r_{24} and r_{25}, respectively, but for the *low* session priority preferences.

5 Rescheduling

In this section, we present RRSP encoding for both phases, with the aim of providing an alternative schedule when the original generated one is no longer viable. The need for rescheduling could arise if patients, operators, or both, are no longer available to perform physiotherapy sessions for the totality of the day, or for some specific amount of time. For this reason, the output of the RSP of the two phases (i.e., the coupling between patients and operators in the board phase and a time and place for the physiotherapy sessions to happen

in the agenda phase) is now given as input to the correspondent rescheduling encoding, together with facts representing the (total or partial) unavailability of patients and/or operators. In general, as we have seen in Section 2.4, the goal of the rescheduling is to find another schedule *as close as possible* to the original one and in which both the constraints of the problem and the unavailability are respected, and this must be done in very short time: for this reason, some constraints of the scheduling, e.g., those related to locations, have been relaxed to deal with emergency situations, and are practically dealt with in post-processing by the coordinator. In the following subsections, the RRSP encoding for both phases are presented.

5.1 Rescheduling of the board

As stated in the previous sections, in the board phase the main goal is to couple operators and patients together, respecting the qualifications and working time of operators and preferences between patients and operators. A rescheduling of the board could be needed if patients and/or operators become unavailable for some period of time (or even for the whole day). If operators become unavailable, the patients which were previously assigned to them must be reassigned to the other operators, being careful of respecting their qualifications, working times and distributing patients as fairly as possible (i.e., spreading the new patients among all the operators with available time left). If, on the other hand, some patients become unavailable, this could cause an imbalance of the distribution of patients among the operators and, for this reason, a redistribution of the patients between the operators should be carried out.

With respect to the RSP encoding presented in Section 4.1, the input data model is enriched with the following atoms:

- Instances of `assignment(OP,PAT)`, which represent the original assignment of operator `OP` to patient `PAT`, found in the scheduling.
- Instances of `unavailable(OP, H)`, which determine the unavailability of operator `OP` during its working hours for `H` time slots.
- Instances of `absent(OP, A)`, which determine the unavailability of patient `PAT` where `A` can be either `total` or `partial`.

The output of the rescheduling process is now given by the use of a new atom `reassignment(OP,PAT)` which signals that, in the rescheduling, the operator `OP` is now assigned to patient `PAT`.

In Figure 6, we present the ASP encoding of the RRSP of the board. Rule r_1 defines the auxiliary atom `unavailable(OP)` for cases in which operators have an unavailability which lasts for the totality of their shift. Then, rule r_2 is used to guess a new assignment for each patient whose previously assigned operator is no longer available for the whole working day. Note that such a reassignment is not computed for patients that are absent for the totality of the day; in fact, patients with a partial absence still need to be rescheduled in case the operator to which they were previously scheduled has become unavailable. Rules r_3 and

```
1  unavailable(OP) :- unavailable(OP,H), operator_contract(OP,T,_), H>=T.

2  {reassignment(OP,PAT): operator(OP), pref(OP,PAT), not unavailable(OP)} = 1 :-
       assignment(OPN,PAT), unavailable(OPN), not absent(PAT,total).
3  final_assignment(OP,PAT) :- reassignment(OP,PAT).
4  final_assignment(OP,PAT) :- assignment(OP,PAT), not unavailable(OP), not absent(PAT,total).

5  :- operator_limit(OP,T,MAX), reassignment(OP,PAT), patient_data(PAT,T,_), MAX < 1.

6  :~ #sum{W, PAT: reassignment(_,PAT), patient_preference(PAT,OP,W)} = N. [N@4]
7  :~ #count{PAT: reassignment(-1,PAT)} = N. [N@3]
8  :~ #sum{W, PAT: reassignment(_,PAT), history_preference(PAT,OP,W)} = N. [N@2]
9  :~ operator_contract(OP,ET,_), not unavailable(OP), #sum{T, PAT: patient_data(PAT,_,T),
       final_assignment(OP,PAT)} = N, DF = (ET-N)*(ET-N). [DF@1,OP]
10 :~ operator_contract(OP,ET,_), unavailable(OP,NP), #sum{T, PAT: patient_data(PAT,_,T),
       final_assignment(OP,PAT)} = N, DF = (ET-N-NP)*(ET-N-NP). [DF@1,OP]
```

Fig. 6: ASP encoding for the rescheduling of the board caused by total or partial unavailability of either operators or patients.

r_4 define the auxiliary atom `final_assignment(OP,PAT)` with the intention of joining the assignments which need a reassignment, caused by the unavailability of some operators, and the assignments which do not need a reassignment, since the operator coming from the scheduling is still available. Rule r_5 enforces that a patient with a type `T` (e.g., a neurological patient covered by the NHS which needs a lifter to perform the session) is not assigned to operators that cannot handle patients of that type. Then, weak constraints r_6–r_8 are the same as the scheduling but applied only to the reassignments which need to be rescheduled. Finally, r_9 and r_{10} minimize the square difference between the working time of an operator and the sum of the minimum time needed to treat each patient assigned to the operator, accounting also for the unavailability time of the operator; in this way, among the solutions having the same costs for rules r_6–r_8, the reassignments will be chosen with the aim of assigning patients to operators with the larger free time available.

5.2 Rescheduling of the agenda

As in the case of the rescheduling of the board, the rescheduling of the agenda is needed if operators and/or patients become unavailable. With respect to the scheduling encoding presented in Section 4.2, all the atoms over predicates `start`, `length`, `extstart`, and `extlength` are now part of the input of the rescheduling. Atoms of the form `session(ID,PAT,OP)`, which were part of the input of the scheduling, are also part of the input of the rescheduling, with the addition of new instances of such atoms computed during the rescheduling of the board and representing that a patient is assigned to a new operator. Moreover, additional input atoms are of the form `unavailable(OP,PER,ST,END)` (resp. `absent(PAT,PER,ST,END)`), whose instances determine the unavailability of operator `OP` (resp. absence of patient `PAT`) during the period `PER`, and during the time slots from `ST` to `END`.

```
all_un_sess(ID,OP,T,resc) | all_un_sess(ID,OP,supervised,resc) :- session(ID,PAT,OP), session_type(ID,OP,T), session(ID,PAT,OPN),
        unavailable(OPN,_,_,_), not absent(PAT,_,_,_), OP != OPN.
all_un_sess(ID,OP,T,orig) | all_un_sess(ID,OP,supervised,orig) :- session(ID,PAT,OP), session_type(ID,OP,T), not
        all_un_sess(ID,_,_,resc), unavailable(OP,_,_,_), not absent(PAT,_,_,_).
all_abs_sess(ID,OP,individual,other) :- session(ID,PAT,OP), session(_, PAT2,OP,_), PAT!=PAT2, session_type(ID,OP,individual), not
        unavailable(OP,_,_,_), not all_un_sess(ID,OP,_,_), session_length(ID,_,IL), session_preference(ID,IP,IH,_), not
        absent(PAT,_,_,_), absent(PAT2,_,_,_), #count{IDX: start(IDX,_,ST), IH!=ST; IDX: start(IDX,PER,_), IP!=PER; IDX:
        length(IDX,_,L), L!=IL} > 0.
all_abs_sess(ID,OP,T,abs) :- session(ID,PAT,OP), session_type(ID,PAT,T), absent(PAT,PER,_,_), PER != total.
fin_sess(ID,OP,T,new) :- all_un_sess(ID,OP,T,_).
fin_sess(ID,OP,T,new) :- all_abs_sess(ID,OP,T,_).
fin_sess(ID,OP,T,old) :- session(ID,PAT,OP), session_preference(ID,OP,T) not absent(PAT,_,_,_), not unavailable(OP,_,_,_), not
        fin_sess(ID,OP,_,new).
{n_start(ID,PER,TS): time(PER,OP,TS)} = 1 :- fin_sess(ID,_,OP,new), mandatory(ID).
{n_start(ID,PER,TS): time(PER,OP,TS)} :- fin_sess(ID,_,OP,new), optional(ID).
fin_start(ID,PER,TS) :- start(ID,PER,TS), fin_sess(ID,OP,_,old).
fin_start(ID,PER,TS) :- n_start(ID,PER,TS).
{n_length(ID,PER,NL): time(PER,OP,L), NL=L-ST, TS+NL<=END, NL>=1, NL<=MAX} = 1 :- n_start(ID,PER,TS), period(PER,OP,ST,END),
        all_un_sess(ID,OP,_,_), session(ID,_,OP), session_length(ID,_,MAX).
{n_length(ID,PER,NL): time(PER,OP,L), NL=L-ST, TS+NL<=END, NL>=MIN, NL<=MAX} = 1 :- n_start(ID,PER,TS), period(PER,OP,ST,END),
        all_abs_sess(ID,OP,_,_), session_length(ID,MIN,MAX).
final_length(ID,PER,NL) :- length(ID,PER,NL), fin_sess(ID,OP,_,old).
final_length(ID,PER,NL) :- n_length(ID,PER,NL).
{before(ID,NL): time(PER,OP,L), NL=L-ST, NL<=TS-ST} = 1 :- n_start(ID,PER,TS), period(PER,OP,ST,_), fin_sess(ID,OP,_,new).
{after(ID,NL): time(PER,OP,L), NL=L-ST, NL<=END-TS-LEN} = 1 :- n_start(ID,PER,TS), period(PER,OP,ST,END), n_length(ID,PER,LEN),
        fin_sess(ID,OP,_,new).
n_ext_start(ID,PER,TS-LB) :- n_start(ID,PER,TS), before(ID,LB).
n_ext_length(ID,PER,L+LA+LB) :- n_length(ID,PER,L), after(ID,LA), before(ID,LB).
fin_ext_start(ID,PER,TS) :- n_ext_start(ID,PER,TS).
fin_ext_start(ID,PER,TS) :- ext_start(ID,PER,TS), fin_sess(ID,_,_,old).
fin_ext_length(ID,PER,L) :- n_ext_length(ID,PER,L).
fin_ext_length(ID,PER,TS) :- ext_length(ID,PER,TS), fin_sess(ID,_,_,old).
:- fin_ext_length(ID,_,RL), n_ext_length(ID,_,SL), RL > SL.

:- unavailable(OP,PER,ST,END), fin_sess(ID,OP,_,new), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L), ST<=TS, END>TS.
:- unavailable(OP,PER,_,END), fin_sess(ID,OP,_,new), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L), END>TS, END<=TS+L.
:- unavailable(OP,PER,ST,_), fin_sess(ID,OP,_,new), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L), ST>=TS, ST<TS+L.
:- absent(PAT,PER,ST,END), all_abs_sess(ID,_,_,abs), session(ID,PAT,_), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L),
        ST<=TS, END>TS.
:- absent(PAT,PER,_,END), all_abs_sess(ID,_,_,abs), session(ID,PAT,_), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L),
        END>TS, END<=TS+L.
:- absent(PAT,PER,ST,_), all_abs_sess(ID,_,_,abs), session(ID,PAT,_), fin_ext_start(ID,PER,TS), fin_ext_length(ID,PER,L), ST>=TS,
        ST<TS+L.
:- fin_start(ID1,PER,TS), final_length(ID1,PER,L), fin_sess(ID1,OP,individual,_), n_start(ID2,PER,TS2),
        fin_sess(ID2,OP,individual,new), ID1!=ID2, TS2>=TS, TS2<TS+L.
:- n_start(ID1,PER,TS), n_length(ID1,PER,L), fin_sess(ID1,OP,individual,new), fin_start(ID2,PER,TS2),
        fin_sess(ID2,OP,individual,_), ID1!=ID2, TS2>=TS, TS2<TS+L.
:- fin_start(ID1,PER1,_), fin_sess(ID1,PAT,_,_), fin_start(ID2,PER2,_), fin_sess(ID2,PAT,_,_), ID1!=ID2, PER1=PER2.
:- patient(PAT,MIN), not absent(PAT,_,_,_), #sum{LEN, SES: fin_sess(SES,_,_,_), session(SES,PAT), finalExtLength(SES,_,LEN)} <
        MIN.
:~ patient(PAT,MIN), absent(PAT,_,_,_), S = #sum{LEN, SES: fin_sess(SES,_,_), session(SES,PAT,_), fin_ext_length(SES,_,LEN)}.
        [|S-MIN|@7, PAT]
:~ #count{ID: all_un_sess(ID,_,supervised,_)} = N. [N@6]
:~ n_length(ID,_,NL), fin_sess(ID,_,_,new), session_length(ID,_,IL), DF=|NL-IL|. [DF@5, ID]
:~ n_start(ID,NPER,_), all_abs_sess(ID,_,_,_), session_preference(ID,PER,_,_), DF=|PER-NPER|. [DF@4, ID]
:~ n_start(ID,PER,NTS), all_abs_sess(ID,_,_,_), session_preference(ID,PER,IH,_), DF=|IH-NTS|. [DF@3, ID]
:~ n_start(ID,NPER,_), all_un_sess(ID,_,_,_), start(ID,PER,_), DF=|NPER-PER|. [DF@2,ID]
:~ n_start(ID,PER,NTS), all_un_sess(ID,_,_,_), start(ID,PER,TS), DF=|NTS-TS|. [DF@1,ID]
```

Fig. 7: ASP encoding for the rescheduling of the agenda caused by total or partial unavailability of either operators or patients.

Rule r_1 defines atoms of the form `all_un_sess(ID,OP,T,resc)` for sessions reassigned in the board phase to another operator, to which no absence is associated. If there is no sufficient time, the session can be converted into a supervised one. Rule r_2 defines atoms of the form `all_un_sess(ID,OP,T,orig)`, representing sessions held by an operator with partial unavailability, that can be still executed by the same operator (i.e., they were not reassigned in the board phase). Rule r_3 defines `all_abs_sess(ID,OP,individual,other)`, representing sessions that are not in their ideal time (length, period or time) and that are, in the scheduling, assigned to the same operator of a patient with an absence. Thus, they include all sessions *(i)* that are not supervised; *(ii)* assigned to an op-

erator with an absent patient; or *(iii)* that were not assigned in their ideal time during the original scheduling. Intuitively, they identify all sessions that can benefit from the absence of another patient, since they can be moved closer to their ideal time. Rule r_4 defines the atom `all_abs_sess(ID,OP,individual,abs)` representing sessions associated to a partial absence. Indeed, atoms of the form `absent(PAT,total,_,_)` indicate that the patient `PAT` is absent for all day. Then, rules r_5 and r_6 are used to define the atom `fin_sess(ID,OP,T,new)`, which collects all new sessions that must be rescheduled. Sessions that must not change in the new scheduling are instead collected as instances of the atom `fin_sess(ID,OP,T,old)`, defined by rule r_7. Then, rules from r_8 to r_{23} are used to compute the preliminary starting time of the sessions (r_8–r_{11}), and their preliminary lengths (r_{12}–r_{15}), to reserve time slots before and after their starting times (r_{16} and r_{17}), and then to compute the actual starting times and lengths using the preliminary ones and the reserved time slots (r_{18}–r_{23}). Rule r_{24} imposes that supervised parts of sessions cannot be further extended in the rescheduling. Rules from r_{25} to r_{27} ensure that a session cannot be scheduled during the period in which its operator is (partially) unavailable. Rules from r_{28} to r_{30} ensure that a session cannot be scheduled during the period in which its patient is (partially) absent. Rules r_{31} and r_{32} ensure that two individual sessions cannot overlap. Rule r_{33} ensures that for each patient there is at most one session per period. Rule r_{34} states that patients with no absence must have their minimum time reserved between all the sessions of the day. Finally, weak constraints from r_{35} to r_{41} are used to express preferences among the different schedules, in a prioritized way: if partially absent patients have more than one session per day, and there is the possibility they cannot complete one of them due to the (partial) absence, the distance between the minimum cumulative time and the extended length of the remaining session must be minimized (r_{35}); the number of supervised sessions must be minimized (r_{36}); for each session the difference between the new and the maximum length must be minimized (r_{37}); original sessions must be placed as close as possible to their ideal period (r_{38}) and their ideal starting time (r_{39}); and new sessions must be placed as close as possible to their ideal period (r_{40}) and their ideal starting time (r_{41}).

6 Experimental Analysis

In this section, we present an empirical evaluation of the encodings described in Section 4 and 5. We performed our analysis on real data coming from the institutes of Genova Nervi and Castel Goffredo. The section is organized in three subsections: the first subsection describes the real data employed, then the second subsection overviews the results obtained with the RSP encoding, while the third subsection presents in details the results about the RRSP.

The analysis was performed using the ASP solver CLINGO [27], employing two different optimization methods: the first one is the default Branch&Bound (BB) optimization method [26] configured with the option `--restart-on-model` enabled; the second instead leverages the Unsatisfiable Core (USC) algorithm [3,

Institute	# Operators	# Patients	Density
Genova Nervi	[9,18]	[37,67]	[2.4,5.2]
Castel Goffredo	[11,17]	[51,78]	[3.5, 6.4]

Table 1: Main dimensions of the ICS Maugeri's institutes.

	Branch & Bound + RoM				Unsatisfiable Core			
	Genova Nervi		Castel Goffredo		Genova Nervi		Castel Goffredo	
	Board	Agenda	Board	Agenda	Board	Agenda	Board	Agenda
% Optimum	35%	0%	0%	0%	22%	45%	0%	0%
% Satisfiable	65%	100%	100%	67%	78%	55%	100%	70%
% Unknown	0%	0%	0%	33%	0%	0%	0%	30%
Avg Time for opt	1.1s	-	-	-	10s	0.01s	-	-
Avg Time Last SM	1.3s	30s	5.2s	30s	12.1s	21.3s	10.4s	30s

Table 2: Results on ICS Maugeri institutes.

4] with the options `--opt-strategy=usc,k,0,4` and `--opt-usc-shrink=bin` enabled. All the experiments were conducted on a MacBook Pro, 2.5 GHz Intel Core i7 quad-core, 16 GB of RAM and imposing a cut-off of 30s per instance.

Encodings and benchmarks used in the experiments can be found at: `http://www.star.dist.unige.it/~marco/CILC2021JLC/material.zip`.

6.1 Real data

ICS Maugeri utilizes, in its daily activity of scheduling the rehabilitation sessions of its patients, a web-based software called QRehab [41], developed by SurgiQ, which is built on top of the specified encoding; thus, the analysis can be performed on real data coming from the institutes of Genova Nervi and Castel Goffredo, which tested and used this software since mid 2020 for Genova Nervi and the beginning of 2021 for Castel Goffredo. This allowed us to access 290 RSP problem instances for Genova Nervi and 100 for Castel Goffredo. Table 1 provides an overview of the dimension of the instances in the two institutes in terms of number of physiotherapists (column # Operators), number of daily patients (column # Patients), and density of patients per operator (column Density).

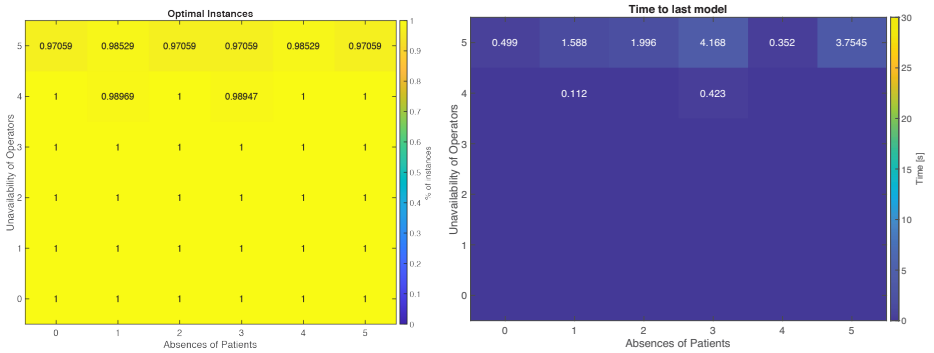
6.2 RSP results

In Table 2, the results obtained by the two RSP encodings are presented in terms of percentage of instances for which an optimal/satisfiable/no solution is computed. The last two rows report the mean time of instances solved optimally and of the last computed solution for all satisfiable instances, respectively. As a general observation, results are mixed. Indeed, the USC algorithm performs better in the agenda encoding while BB algorithm is better on the board scheduling. Moreover, 100% of the board instances are solved, while for approximately one

third of the agenda instances a solution cannot be found. Considering these are hard real instances, results are positive and highly appreciated by the personnel of the ICS Maugeri: according to data elaborated by ICS Maugeri, the introduction of the automatic scheduling system in the hospital of Genova Nervi allowed saving 74.75 minutes of the operators time per day.

6.3 RRSP results

In this section, we focus on the scalability of our approach in relation to the number of absences of patients and/or unavailability of operators. For every real world instance, absences and unavailability, as well as their periods and time slots, were randomly assigned between patients and operators. Then, in our experiment, we analyse how the encodings of the two phases performed while increasing the number of absences and unavailability. In particular, based on the feedback from the coordinators of the hospitals in Genova Nervi and Castel Goffredo, we only considered unavailability and absences of at most 5 operators and 5 patients, which are usually an upper bound of the number which could occur in a real-world hospital of that sizes daily. Starting from the scheduling results, the experiments on the board phase have been run with the Branch&Bound optimization method, while the experiments on the agenda phase have been run with the Unsatisfiable Core algorithm.

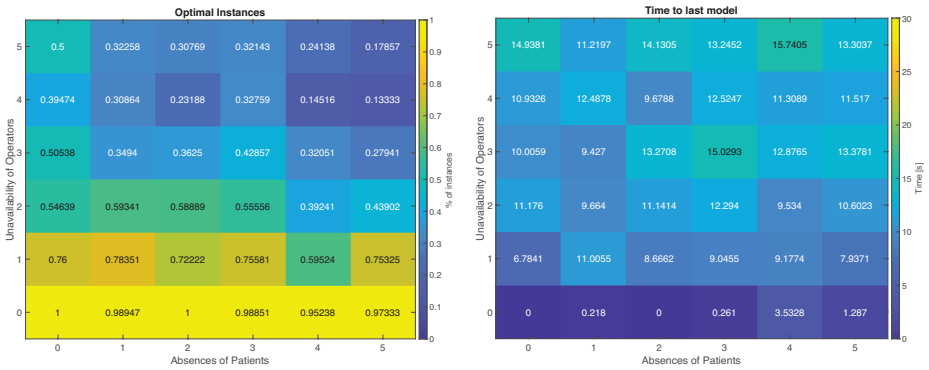


(a) Optimal result (b) Time to last model

Fig. 8: Scalability results of the board phase. On the left, the percentage of instances in which the scheduling outputted an optimal result. On the right, the average time it took to find the last model.

Scalability of the board. Figure 8 shows the results of the scalability experiments when the encoding of the board phase is considered. On the left, in Figure 8a, we show the percentage of instances where CLINGO provided an optimal solution for the rescheduling encoding. Every square inside the image represents a collection

of instances (almost 100 instances coming from real scheduling in the Genova Nervi and C. Goffredo hospitals) having a specified (randomly chosen) number of absences of patients (x-axis) and number of unavailability of operators (y-axis). On the right, in Figure 8b, we show the average running time required to find the latest stable model. As it can be seen from the figure, for the board phase, an increasing number of unavailability of operators and/or absences of patients does not cause a noticeable increase in the solving time of CLINGO which is able to always find an optimal solution. Only when dealing with 5 unavailable operators, a small percentage of instances (less than 5%) return a suboptimal solution, but, looking at the time it takes to output the last model, it can be seen that CLINGO outputs the last stable model, on average, in the first $5s$, meaning that the solution is *probably* optimal but CLINGO is not able to prove its optimality before the cut-off time.



(a) Optimal result (b) Time to last model

Fig. 9: Scalability results of the agenda phase following the same representation structure of Figure 8.

Scalability of the agenda. Figure 9 shows, in the same representation as discussed in the previous paragraph, the scalability results for the agenda encoding. Differently from the board phase, in the agenda it can be noted that increasing the number of absences of patients and unavailability of operators has a negative impact on the performance of CLINGO in terms of number of times it provides an optimal solution. Nonetheless, Figure 9b shows that even if the solutions are not marked by CLINGO as optimal, the last stable model is found on average in the first $15s$, meaning that, as in the board phase, the solution may be optimal but CLINGO is not able to prove it before the cut-off.

Finally, note that the rescheduling encodings cannot be unsatisfiable since it always exists a solution in which all the sessions belonging to unavailable operators are given to other operators which will perform them in a completely supervised fashion.

7 Web Application

In this section, we present a web application developed to be used by an end user who wants to access our solution easily, e.g., a physiotherapists' coordinator. The software is a full-stack JavaScript application with an Angular Graphical User Interface (GUI) and a Node.js as back-end which behaves as a wrapper for the ASP encoding and the CLINGO solver. The encodings and their input (expressed as ASP facts) are dynamically composed at run-time reflecting the user choices, and then the ASP solver CLINGO is executed directly on the back-end. In this way, end users do not need to install and run the solver. Moreover, the application includes (*i*) a database for storing and retrieving previous test data and new generated data, and (*ii*) a GUI to easily load pre-made scheduled scenarios, and create and customize the rescheduling. The GUI is divided into the two main parts: the board screen and the agenda screen. The board screen (reported as Figure 10) shows the assignment between operators and patients, identified by their IDs. At the top of the screen, the user can define the unavailability of operators and/or the absence of patients, which represents an input of the rescheduling. After the input is inserted, the board is updated accordingly, showing the rescheduled assignments. In particular, the reassigned patients are highlighted in green, whereas the unavailable operators and patients are highlighted in red.

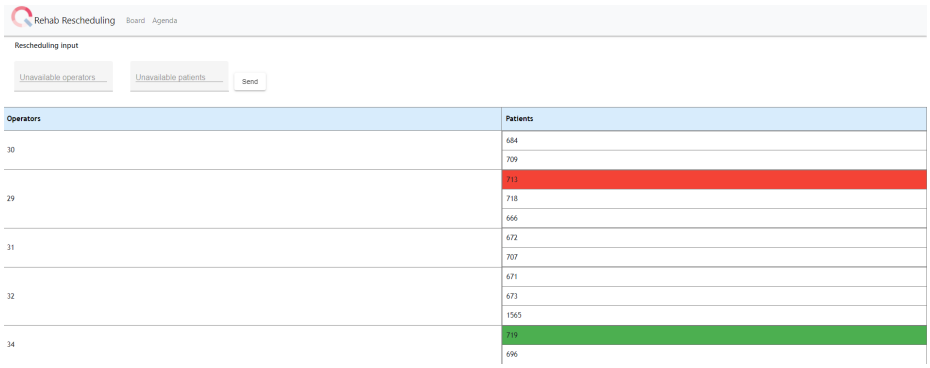


Fig. 10: Rescheduled board and input screen.

The agenda screen (reported as Figure 11) shows the agenda for each operator. In particular, rehabilitation sessions are shown in their time slots, where blue bars represent individual sessions, whereas yellow bars represent supervised ones. For each session the bar contains the patient, start and end times of the session, and a note to determine whether the session has been rescheduled or not. Moreover, the time slots when the operator does not work are coloured in grey.

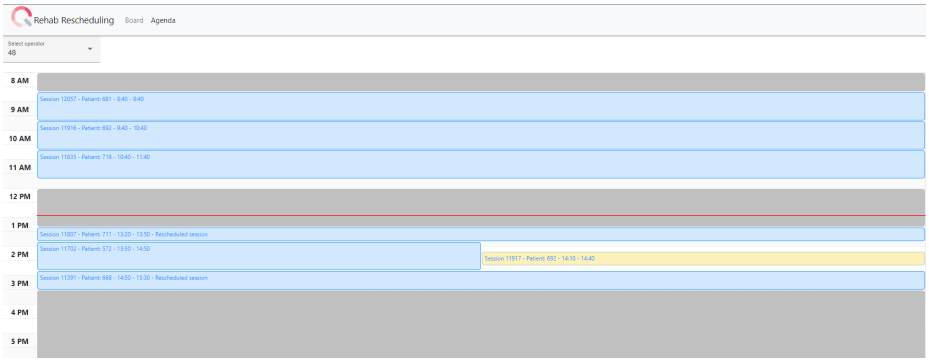


Fig. 11: Rescheduled agenda screen.

8 Related Work

This section describes, in three separate subsections, the new added material w.r.t. the conference version of the paper, the motivation for employing ASP and its usage in other scheduling problems, and the discussion of the related work.

8.1 Contribution

This paper is an extended and revised version of a paper presented at the 36th Italian Conference on Computational Logic (CILC 2021), and published in [13]. The main difference is that in the present paper we focus on rescheduling, thus all the related parts of this paper are improvements w.r.t. such work, i.e.: (*i*) the problem description in Section 2.4 and the examples in Section 2.5, (*ii*) the RRSP encoding for both phases in Section 5, (*iii*) the experimental analysis of our RRSP solution, focused on real instances, in Section 6.3, and (*iv*) the web application in Section 7. Moreover, previous work contains also an analysis on synthetic benchmarks and a study about real and synthetic instances for validating synthetic instances: the details of the synthetic benchmarks as well as the details of such analysis can be found in Section 4 of [13], and all material is available at the link provided above. Another, orthogonal direction has been followed in [14] (invited for a special issue of the best papers published in the RuleML+RR 2021 conference [13]): here, we have instead focused on strengthening and extending the scheduling solution, thus with no relation with the main contribution of the current paper which is on rescheduling.

8.2 ASP in scheduling problems

ASP has been successfully used for solving hard combinatorial and application scheduling problems in several research areas. The reasons for the success are manyfold: (*a*) ASP specifications are often appreciated even by non-experts since

they found them readable; (*b*) There are free and open source systems (like CLINGO, and WASP [1]), whose performances are often comparable (or even better) to the ones of industrial tools like, e.g. CPLEX, or to GUROBI, also thanks to a number of ASP competitions held until recently [28, 29]; and (*c*) ASP allows expressing and solving multi-level optimizations. In the Healthcare domain, the first solved problem with ASP was the *Nurse Scheduling Problem* [22, 5], where the goal is to create a scheduling for nurses working in hospital units. Then, the problem of assigning ORs to patients, denoted as *Operating Room Scheduling* [20], has been treated, and further extended to include bed management [18] and surgical teams [19]. More recent problems in which ASP is employed include the *Chemotherepy Treatment Scheduling* problem [17], in which patients are assigned a chair or a bed for their treatments. For an overview, we refer to [2].

Concerning scheduling problems beyond the healthcare domain, ASP encoding were proposed for the following problems: *Incremental Scheduling Problem* [7], where the goal is to assign jobs to devices such that their executions do not overlap one another; *Team Building Problem* [40], where the goal is to allocate the available personnel of a seaport for serving the incoming ships; the work in [30], where, in the context of routing driverless transport vehicles, the setup problem of routes such that a collection of transport tasks is accomplished in case of multiple vehicles sharing the same operation area is solved via ASP, in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH, and the recent survey paper by Falkner et al. [24], where industrial applications dealt with ASP are presented, including those involving scheduling problems.

8.3 Discussion of the related work

There have been some attempts to solve rehabilitation scheduling, since most hospitals are still doing it in a manual way. Among the automated solutions, often they are applied to real world data. However, their results are not directly comparable to ours, since their constraints and objective functions are different from the ones that emerged from our meetings with the physiotherapists and management of ICS Maugeri. In particular, to the best of our knowledge, no other solution takes into account several aspects like the preferred time for the session scheduling and devised the solution as a two-phase approach with the possibility of manual corrections in between. Huang, Zheng and Chien [32] developed a system, equipped with a GUI, which can generate the optimal schedules for rehabilitation patients to minimize waiting time, and thus enhance service quality and overall resource effectiveness of rehabilitation facilities. More recently, Huyinh, Huang and Chien [33] further refined the algorithm in order to develop a hybrid genetic algorithm (GASA) that integrates genetic algorithm (GA) and simulated annealing (SA). Recently, Li and Chen [35] designed a genetic algorithm based on a Waiting Time Priority Algorithm, which was tested on a rehabilitation department. Schimmelpfeng, Helber and Kasper [42] developed a decision support system for the scheduling process based on mixed-integer linear programs

(MILPs) to determine appointments for patients of rehabilitation hospitals, subject to numerous constraints that are often found in practice. J. Wang and Fung [44] formulated a dynamic programming (DP) model to optimize appointment scheduling taking into account patient preferences and choices. The DP model is transformed into an equivalent linear programming (LP) model. Liyang et al. [36] developed a system aiming to minimize the total waiting time between successive treatments for patients in rehabilitation hospitals. They formulated the problem as a MILP model, taking into consideration real-life requirements, and used the commercial solver Gurobi. Then, they developed an improved cuckoo search algorithm to obtain good quality solutions quickly for large-sized problems. Their model was demonstrated with data collected from a medium-sized rehabilitation hospital in China.

While we are not aware of any other rescheduling article applied to rehabilitation, the generic problem of rescheduling in case of disruption of the original schedule, has been studied in several works in the healthcare domain, especially for nurse re-rostering problems. Among them, Uhmn et al. [43] solved the rescheduling of nursing shifts with a deterministic, depth-first iterative search algorithm which minimizes the difference between the original and the rescheduled rosters. In Kitada and Morizawa [34], a recursive search is used in the case of a shortage that finds the first valid solution; in the process, a tree is constructed, whose nodes contain candidate operators to replace the missing positions; then, through a search strategy in these nodes, the best rescheduling can be found. Pato and Moz [39] instead used genetic heuristics that build on the Pareto approach, including a diversification strategy. Clark and Walker [16] used a combination of weighted sums and objective planning, using weights to prioritize; first, the method tries to minimize the number of changes from the original planning, while then it evaluates how various types of changes may affect rescheduling differently. Bäumelt et al. [45] solved nurse rescheduling by decomposing the problem and using a parallel algorithm running on the GPU. It uses an evolutionary algorithm to solve nurse rescheduling (Maenhout and Vanhoucke [37]) by minimizing additional costs and unsatisfiability, and breaking ties by minimizing shift changes from the original schedule. Temporary staffing is considered in Bard and Purnomo [9] to accommodate the absence of nurses; however, this approach may be necessary in hospitals where the scope of work cannot be met by permanent staff, which is not our case. Finally, an ASP-based approach has already been employed to solve rescheduling problems in some applications mentioned in the previous subsection: Nurse and Operating Room Rescheduling by Alviano et al. [6, 21], and Chemotherapy Treatment Re-Planning by Dodaro et al. [17].

9 Conclusion

In this paper, we have presented scheduling and rescheduling solutions for solving rehabilitation scheduling, by means of a two-phase ASP encoding. In particular, we have focused on the completely new rescheduling solution for the problem and

our solution has been tested with CLINGO on real benchmarks provided by ICS Maugeri, who provided also the specifications of the problem. We have tested our rescheduling encoding for the board and agenda by randomly generating increasing unavailability of operators and/or absence of patients. Results are satisfying for the institutes considered at the moment: for the board phase, we are always able to find an optimal solution and in short time, while for the agenda phase, whose problem and encoding are more elaborated, by increasing the number of operators and/or patients unavailable, the quality of the solution decreases and the time increases, but still obtaining satisfying results even for our larger benchmarks. A possible future development of the rescheduling techniques is their application to other types of patients whose rehabilitation scheduling operates on different principles compared to the ones described in this article, e.g., patients with cardiorespiratory issues.

References

1. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) Proceedings of the 15th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2019). Lecture Notes in Computer Science, vol. 11481, pp. 241–255. Springer (2019)
2. Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Mochi, M., Morozan, V., Porro, I., Schouten, M.: Answer set programming in healthcare: Extended overview. In: Joint Proceedings of the 8th IPS Workshop and the 27th RCRA Workshop co-located with AIxIA 2020. CEUR Workshop Proceedings, vol. 2745. CEUR-WS.org (2020)
3. Alviano, M., Dodaro, C.: Anytime answer set optimization via unsatisfiable core shrinking. Theory and Practice of Logic Programming **16**(5-6), 533–551 (2016)
4. Alviano, M., Dodaro, C.: Unsatisfiable core analysis and aggregates for optimum stable model search. Fundamenta Informaticae **176**(3-4), 271–297 (2020)
5. Alviano, M., Dodaro, C., Maratea, M.: An advanced answer set programming encoding for nurse scheduling. In: Esposito, F., Basili, R., Ferilli, S., Lisi, F.A. (eds.) Advances in Artificial Intelligence - Proceedings of the 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017). Lecture Notes in Computer Science, vol. 10640, pp. 468–482. Springer (2017)
6. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
7. Balduccini, M.: Industrial-size scheduling with ASP+CP. In: Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6645, pp. 284–296. Springer (2011). https://doi.org/10.1007/978-3-642-20895-9_33
8. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003). https://doi.org/10.1017/CBO9780511543357
9. Bard, J., Purnomo, H.: Incremental changes in the workforce to accommodate changes in demand. Health care management science **9**, 71–85 (03 2006). https://doi.org/10.1007/s10729-006-6281-y

10. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
11. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
12. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: ASP-Core-2 input language format. Theory and Practice of Logic Programming **20**(2), 294–309 (2020)
13. Cardellini, M., Nardi, P.D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Porro, I.: A two-phase ASP encoding for solving rehabilitation scheduling. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021). Lecture Notes in Computer Science, vol. 12851, pp. 111–125. Springer (2021)
14. Cardellini, M., Nardi, P.D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M., Porro, I.: Solving Rehabilitation Scheduling problems via a Two-Phase ASP approach. Submitted to Theory and Practice of Logic Programming (2022)
15. Cieza, A., Causey, K., Kamenov, K., Hanson, S.W., Chatterji, S., Vos, T.: Global estimates of the need for rehabilitation based on the Global Burden of Disease study 2019: a systematic analysis for the Global Burden of Disease Study 2019. The Lancet **396**(10267), 2006–2017 (2020), publisher: Elsevier
16. Clark, A., Walker, H.: Nurse rescheduling with shift preferences and minimal disruption. Journal of Applied Operational Research **3** (01 2011)
17. Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M., Porro, I.: An ASP-based solution to the chemotherapy treatment scheduling problem. Theory Practice of Logic Programming **21**(6), 835–851 (2021)
18. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: An ASP-based solution for operating room scheduling with beds management. In: Fodor, P., Montali, M., Calvanese, D., Roman, D. (eds.) Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019). Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
19. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: Solving operating room scheduling problems with surgical teams via answer set programming. In: Baldoni, M., Bandini, S. (eds.) AIxIA 2020 - Advances in Artificial Intelligence - Revised Selected Papers OF THE 19th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2020). Lecture Notes in Computer Science, vol. 12414, pp. 204–220. Springer (2020)
20. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: Ghidini, C., Magnini, B., Passerini, A., Traverso, P. (eds.) Advances in Artificial Intelligence - Proceedings of the 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018). Lecture Notes in Computer Science, vol. 11298, pp. 445–459. Springer (2018)
21. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: An ASP-based framework for operating room scheduling. Intelligenza Artificiale **13**(1), 63–77 (2019)
22. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: Balduccini, M., Janhunen, T. (eds.) Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2017). Lecture Notes in Computer Science, vol. 10377, pp. 301–307. Springer (2017)
23. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)

24. Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C.: Industrial applications of answer set programming. Künstliche Intelligenz **32**(2-3), 165–176 (2018)
25. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Carro, M., King, A., Saeedloei, N., Vos, M.D. (eds.) Proceedings of ICLP (Technical Communications). OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
26. Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., Schaub, T.: Progress in clasp Series 3. In: LPNMR. LNCS, vol. 9345, pp. 368–383. Springer (2015)
27. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187**, 52–89 (2012)
28. Gebser, M., Maratea, M., Ricca, F.: The design of the seventh answer set programming competition. In: Balduccini, M., Janhunen, T. (eds.) LPNMR. Lecture Notes in Computer Science, vol. 10377, pp. 3–9. Springer (2017)
29. Gebser, M., Maratea, M., Ricca, F.: The seventh answer set programming competition: Design and results. Theory and Practice of Logic Programming **20**(2), 176–204 (2020)
30. Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M., Runge, M.: Routing driverless transport vehicles in car assembly with answer set programming. Theory Practice of Logic Programming **18**(3-4), 520–534 (2018)
31. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing **9**(3/4), 365–386 (1991)
32. Huang, Y.C., Zheng, J.N., Chien, C.F.: Decision support system for rehabilitation scheduling to enhance the service quality and the effectiveness of hospital resource management. J. of the Chinese Inst. of Industrial Engineers **29**, 348 – 363 (2012)
33. Huynh, N.T., Huang, Y.C., Chien, C.F.: A hybrid genetic algorithm with 2D encoding for the scheduling of rehabilitation patients. Computers & Industrial Engineering **125**, 221–231 (2018)
34. Kitada, M., Morizawa, K.: A heuristic method in nurse rerostering following a sudden absence of nurses (01 2010)
35. Li, X., Chen, H.: Physical therapy scheduling of inpatients based on improved genetic algorithm. Journal of Physics: Conference Series **1848**(1), 012009 (apr 2021)
36. Liyang, X., Dridi, M., Hajjam, A., Lin, W., Fei, H.: A solution method for treatment scheduling in rehabilitation hospitals with real-life requirements. IMA Journal of Management Mathematics **30** (07 2018). https://doi.org/10.1093/imaman/dpy009
37. Maenhout, B., Vanhoucke, M.: An artificial immune system based approach for solving the nurse re-rostering problem. vol. 7832 (01 2011). https://doi.org/10.1007/978-3-642-37198-1_9
38. Niemelä, I.: Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. AMAI **25**(3-4), 241–273 (1999)
39. Pato, M., Moz, M.: Solving a bi-objective nurse rerostering problem by using a utopic pareto genetic heuristic. J. Heuristics **14**, 359–374 (08 2008). https://doi.org/10.1007/s10732-007-9040-4
40. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. Theory and Practice of Logic Programming **12**(3), 361–381 (2012)
41. Saverino, A., Baiardi, P., Galata, G., Pedemonte, G., Vassallo, C., Pistarini, C.: The challenge of reorganizing rehabilitation services at the time of covid-19 pandemic: A new digital and artificial intelligence platform to support team work in planning and delivering safe and high quality care. Frontiers in neurology **12**, 643251 (2021)

42. Schimmelpfeng, K., Helber, S., Kasper, S.: Decision support for rehabilitation hospital scheduling. OR Spectrum **34**(2), 461–489 (Apr 2012)
43. Uhmn, S., Ko, Y.W., Kim, J.: A deterministic approach to nurse rerostering problem. International Journal of Applied Engineering Research **12**, 14246–14250 (01 2017)
44. Wang, J., Fung, R.: Dynamic appointment scheduling with patient preferences and choices. Industrial Management & Data Systems **115**, 700–717 (2015). https://doi.org/10.1108/IMDS-12-2014-0372
45. Zdeněk, B., Dvořák, J., Sucha, P., Hanzálek, Z.: A novel approach for nurse rerostering based on a parallel algorithm. European Journal of Operational Research **251** (12 2015). https://doi.org/10.1016/j.ejor.2015.11.022