POLITECNICO DI TORINO
Repository ISTITUZIONALE

Solving rehabilitation scheduling problems via a two-phase ASP approach

Original
Solving rehabilitation scheduling problems via a two-phase ASP approach / Cardellini, M., DE NARDI, P., Dodaro, C., Galata`, G., Giardini, A., Maratea, M., Porro, I.. - In: THEORY AND PRACTICE OF LOGIC PROGRAMMING. - ISSN 1475-3081. - STAMPA. - 24:2(2024), pp. 344-367. [10.1017/S1471068423000030]

Availability:
This version is available at: 11583/2977870 since: 2023-04-17T13:17:31Z

Publisher:
Cambridge University Press

Published
DOI:10.1017/S1471068423000030

Terms of use:

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

Publisher copyright

(Article begins on next page)

28 November 2026

TPLP **24** (2): 344–367, 2024. © The Author(s), 2023. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

doi:10.1017/S1471068423000030 First published online 17 April 2023

Solving Rehabilitation Scheduling Problems via a Two-Phase ASP Approach[*]

MATTEO CARDELLINI
Polytecnic of Torino, Torino, Italy
University of Genova, Genova, Italy
(*e-mail:* matteo.cardellini@polito.it)

PAOLO DE NARDI
ICS Maugeri, Italy
(*e-mail:* paolo.denardi@icsmaugeri.it)

CARMINE DODARO
DeMaCS, University of Calabria, Rende, Italy
(*e-mail:* dodaro@mat.unical.it)

GIUSEPPE GALATÀ
SurgiQ srl, Italy
(*e-mail:* giuseppe.galata@surgiq.com)

ANNA GIARDINI
ICS Maugeri, Italy
(*e-mail:* anna.giardini@icsmaugeri.it)

MARCO MARATEA
DIBRIS, University of Genova, Genova, Italy
DeMaCS, University of Calabria, Rende, Italy
(*e-mail:* marco@dibris.unige.it)

IVAN PORRO
SurgiQ srl, Italy
(*e-mail:* ivan.porro@surgiq.com)

submitted 28 February 2022; revised 10 October 2022; accepted 16 March 2023

Abstract

A core part of 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 legal, medical, and ethical requirements and optimizations, for example, patient's preferences and operator's work balancing. Being able to efficiently solve such problem is of upmost importance, in particular after the COVID-19 pandemic that significantly increased rehabilitation's needs. In this paper, we present a two-phase solution to rehabilitation scheduling based on Answer Set Programming, which proved to be an effective tool for solving practical scheduling problems. We first present a general encoding and then add domain-specific optimizations. Results of experiments performed on both synthetic and real benchmarks, the latter provided by ICS Maugeri, show the effectiveness of our solution as well as the impact of our domain-specific optimizations.

[*] This paper is an extended and revised version of a conference paper appearing in the proceedings of the RuleML+RR 2021 conference (Cardellini *et al.* 2021).

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

KEYWORDS: answer set programming, rehabilitation scheduling, healthcare

1 Introduction

The rehabilitation scheduling process consists mainly of planning daily patients' physiotherapy sessions inside a rehabilitation institute, that hereafter we refer to as Rehabilitation Scheduling Problem (RSP) (Huang *et al.* 2012; Huynh *et al.* 2018; Li and Chen 2021; Schimmelpfeng *et al.* 2012). Hospitals that may profitably make a practical use of such scheduling, including those managed by ICS Maugeri[1], which will provide benchmarks 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 to operators, that is, physiotherapists, efficiently. A recent article by Cieza *et al.* (2020) found that 2.41 billion people could benefit from rehabilitation services. This finding means that almost one-third of the current population in the world needs rehabilitation at some point during the course of their lives due to 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 hospitalization, thus further increasing the demand for rehabilitation services globally.

The RSP is subject to several constraints, that is, 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, for example, due to clinical and organizational reasons it is often best for a patient to be treated as often as possible by the same operator and at the same time slot; also, rehabilitation professionals' work balancing needs to be taken into proper account.

In this paper, we present a solution to the RSP based on Answer Set Programming (ASP) (Gelfond and Lifschitz 1991; Niemelä 1999; Baral 2003; Brewka *et al.* 2011), which proved to be an effective tool for solving practical scheduling problems (Gebser *et al.* 2018; Ricca *et al.* 2012; Dodaro and Maratea 2017; Dodaro *et al.* 2021), thanks also to the availability of efficient ASP solvers. The solution is designed as a two-phase encoding (Section 3): the first phase, called *board*, deals with the problem of assigning a physiotherapist to every patient considering the total working time of the physiotherapist and the minimum mandatory time of rehabilitation sessions. In the second phase, called *agenda*, a start and end time of every rehabilitation session is defined given the assignment among patients and physiotherapists found in the first phase. Our two-phase solution is not guaranteed to find the best possible overall solution, but has been designed in this way because: (*i*) it simplifies the overall encoding and its practical use, and (*ii*) 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. In fact,

[1] https://www.icsmaugeri.it/.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

coordinators have specifically requested to have the possibility to manually change some patient-operator assignments (the output of the *board*) before sending it to the *agenda* phase. Even if this manual change is seldom made, it gives the coordinators a sense of control and the possibility to introduce human knowledge and expertise in the scheduling. It is important to acknowledge that this tool is not advertised and sold as a medical device, but it is a tool for supporting the decision of the coordinators. For this reason, it is legally mandatory for the coordinators to have a more granular control (and responsibility) over the decisions. We first tested (Section 4) our encoding on real scenarios from ICS Maugeri related to the daily scheduling of neurological patients in two of their rehabilitation institutes in the North of Italy, namely Genova Nervi and Castel Goffredo. In the analysis, we decided to limit the run-time of the ASP solver CLINGO (Gebser *et al.* 2012) to only 30s (while in production the cut-off is set to 5 min): This narrow time limit allows for running much more experiments and having a more significant comparison with the different optimization algorithms in CLINGO. Then, given that ICS Maugeri is planning to instrument with automated techniques other, possibly larger, institutes in addition to Genova Nervi and Castel Goffredo, we generated a wide set of synthetic benchmarks, whose parameters are inspired by real data. We made a wide experimental evaluation and statistically confronted synthetic and real data results using classification decision tree methods (Quinlan 1986), with the aim of predicting the behavior of our solution on such larger institutes. Results show that the accuracy is high, so our synthetic benchmarks appear significant to indicate a possible behavior on real data coming from other institutes with other parameters and similar characteristics. As a side effect, this analysis also outlines the features of the problems that affect the results mostly. Finally, with the aim of further improving the results, and lower the still remaining percentage of instances that could not be solved, we added domain-specific optimizations to our encoding (Section 5): results of the improved encoding show that we are now able to find a solution, even if not always optimal, to every instance within the time limit. The paper is completed by an informal description of the RSP in Section 2 and by discussing related work and presenting conclusions in Section 6 and 7, respectively.

2 Problem description

In this section, we describe the problem we face in four paragraphs. First, we present the general description of the problem, then the data that characterize the main elements of the problem, followed by the requirements of the phases. The last paragraph shows a solution schedule.

General 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 are the ones who spend most of their time with patients and their sessions constitute the core of the daily agenda of the patient, around which all other commitments revolve. For this reason, this article is focused on scheduling the physiotherapy sessions in the most efficient way, optimizing the overall time spent with the patient.

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

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

change in the number and type of patients to be treated, and the number of operators available. Up until recently, this computation has been performed manually by coordinators, without any decision support.

The usual scheduling practice entails two subsequent phases resulting in the computation of a board and an agenda, that we herewith describe. In short, the first phase, called *board*, 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. In the second phase, called *agenda*, a start and end time of every rehabilitation session is computed, given the assignment among patients and operators found in the first phase.

In more details, in the board phase, we ensure that the working hours of operators are respected by counting their total working time, in minutes, and assigning patients to each operator in such a way that the cumulative time of all their sessions remains below the operator's 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 the session, according to the clinical needs of the patient. The choice of the gym is carried out by considering the maximum number of simultaneous sessions allowed inside the gym and has to be made among a subset of gyms which are located at the same floor as the room of the patient, in order to avoid elevators and stairs that can result in discomfort to patients and slowness which can quickly congest the hospital. In this phase, time preferences for each patient are also considered: in fact, plans in which the sessions are performed closer to the desired time of the patients are to be preferred to others.

Instance description. In this paragraph, we describe the main elements of our problem in more details, namely patients, operators, and sessions, as well as the constraints and preferences entailed by the board and agenda phases.

Patients. Patients are characterized by their:

- type (Neurological, Orthopedic, COVID-19 Positive, COVID-19 Negative, Outpatient[2]),
- aid needs, that is, if they need specific care or not (e.g. if they need to be lifted),
- payment status (full payer or in charge of the National Healthcare Service),
- forbidden times, that is, the time intervals when the patient cannot be scheduled,
- ideal time, that is, the preferred scheduled timeslot in which the session should take place, expressed by the coordinator,
- preferred operators, that is, the list of physiotherapists, ordered by priority, the patient can be assigned to,

[2] A person who goes to a hospital for a daily treatment, without staying the night.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

- overall minimum length, that is, the minimum amount of care time that the patient is guaranteed to be scheduled,
- sessions, that is, the list of sessions to be scheduled.

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

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

Moreover, each operator has a limit on the number of patients of a specific type to treat.

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 one in the afternoon shift.

Each session can be delivered to patients either individualized ("one-on-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 deliver one-on-one therapy to one patient, they can supervise other patients. When the operators are particularly overbooked, their one-on-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-on-one part, or vice-versa, or even start and end with a supervised part with a middle one-on-one session. Obviously, an operator can supervise different patients only if their sessions are located at the same place. In the next paragraphs, when defining the *agenda*, Figure 1 will graphically explain the semantic of a mixed session.

The characteristics of the sessions are as follows:

- delivery mode (one-on-one, supervised),
- minimum one-on-one length, that is, the minimum length of the session guaranteed to be delivered one-on-one,
- ideal overall length, that is, the overall length of the session including the one-on-one and supervised parts,
- optional status, that is, if the session can be left out of the schedule in case of overbooked operators,
- forced time, that is, 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, that is, the place where the session must be delivered.

Constraints of the phases. The requirements that the two phases entail are reported in the following sub-paragraphs.

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

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

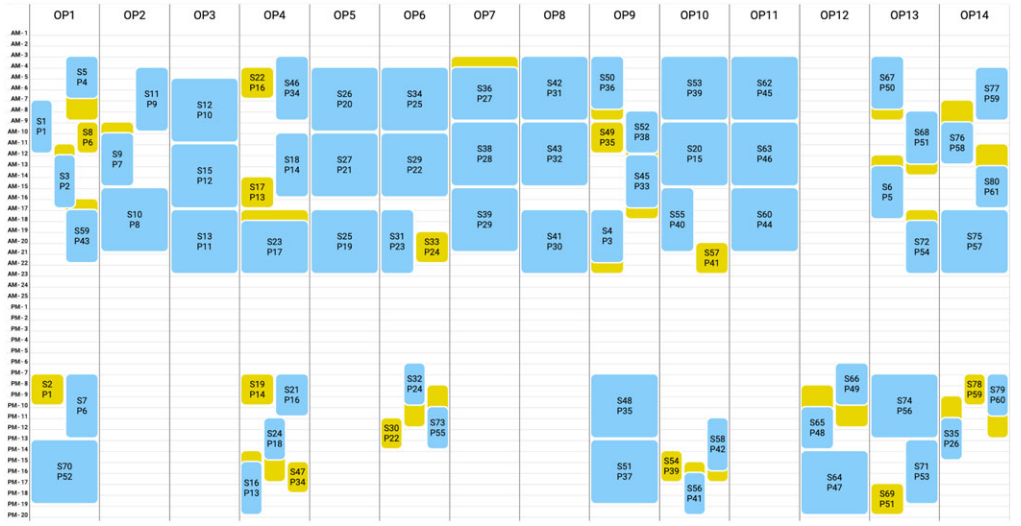

Fig. 1. Result of the scheduling of the agenda in a real case scenario in the hospital of Genova Nervi. Light blue (yellow) squares represent time units in which the sessions will be performed in an individual (supervised) fashion. The ticks on the left keep track of the period (morning or afternoon) and time slot in which the session will start or end.

- 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-on-one length to each patient and session,
- 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 (e.g. in special cases, the coordinator can override the preferred operators list and force an assignment of a patient to an operator regardless of all other considerations) 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. The criteria for the agenda phase are as follows:

- 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-on-one length of the session,
- no overlap between two one-on-one sessions (or the one-on-one part of the session if mixed) assigned to the same operator,
- observance of the maximum capacity of the locations (1 for each room, varying for the gyms),

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

- respect of the minimum cumulative time that the patient should be treated among all the sessions,
- respect of the one-on-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-on-one part should be maximized,
- the largest possible number of optional sessions should be included,
- the overall length, including the one-on-one and supervised parts in case of mixed sessions, should be as close as possible to the ideal overall length specified by the coordinator.

Scheduling example. As previously stated, the output of the *agenda* phase is a start time and duration of every session during the day. Since, in ICS Maugeri, the scheduling was already performed with a timeslot of 10 min, we decided to keep this discretization in place. For this reason, the sessions can start every 10 min in the working hours of the hospital (8AM–12AM in the morning, 1:30PM–4PM in the afternoon) and last a multiple of 10 min. Figure 1 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 in which the patient will be dealt in a supervised mode. Ticks on the left side of the figure describe the period (AM = Morning, PM = Afternoon) and the number we associate to each timeslot. As it can be seen in the first column, Operator 1 (`OP1`) deals with the first session (`S5`) as a mixed session: the session starts in individual one-to-one mode with all the attention of the operator focused on the patient (`P4`). After 4 time slots (i.e. 40 min, the session minimum time), the operator moves the attention to another patient (`P1` performing session `S1`) while the previous patient finishes the session, in the same room, on its own. In a more practical way, in the first 40 min the operator helps the patient in performing exercises which could not be performed alone in a correct way. In the remaining 20 min, which are still very beneficial to the patient, the patient performs the exercises that can be done alone while the operator is working with another patient. Being in the same room, the operator can still intervene if the supervised patient needs correction. Some sessions (e.g. session `S17` of patient `P13` performed in the morning by operator `O4`) 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 `P13` already performs session `S16` in an individual fashion with operator `O4` in the afternoon).

3 A two-phase ASP encoding for the RSP

In the following, we assume the reader is familiar with syntax and semantics of ASP. Starting from the specifications in the previous section, here we present the ASP encoding, based on the input language of CLINGO (Gebser *et al.* 2016). For details about syntax and semantics of ASP programs, we refer the reader to Calimeri *et al.* (2020).

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

3.1 Board encoding

Data Model. The input data are 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` is of the form `value-needs-status`, where `value` can be *neurologic*, *orthopedic*, *covid-19-positive*, *covid-19-negative*, or *outpatient*; `needs` can be *lifter* or *nolifter*, and `status` can be *payer* or *free*. For instance, `neurologic-lifter-payer` indicates that the patient needs a neurological treatment, must be lifted, and the treatment must be paid. Moreover, a fictitious operator with `ID` equals to -1 is included in the list of all the operators, and it is needed to intercept all patients that cannot be assigned to other operators (like a *catch-all*[3]).
- Instances of `operator_contract(ID,TIME,MAX)` represent the contract of the operator with the 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 limit.
- Instances of `patient_data(ID,T,DUR)` represent the data associated with 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 (`DUR`).
- 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 in previous days.

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

Encoding. The related encoding is shown in Figure 2 and is described in the following. To simplify the description, the rule appearing at line i in Figure 2 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), by creating two auxiliary atoms `uniqueLocationLength(OP,PAT,DUR)` and `sameLocationLength(OP,PAT,DUR)` that represent the duration `DUR` in time slots of the session between operator `OP` and patient `PAT` performed in a single or same location,

[3] This is because, for practical reasons, we always want to have a solution.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

```
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. 2. ASP Encoding for the board problem.

respectively, used in the next rule. 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 assignments. In particular, r_7 is used to maximize the assignments that fulfill 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.

3.2 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 unit 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 by the above atom, that is, `time` is defined as `time(PER,OP,STA..END):- period(PER,OP,STA,END)`.
- Instances of `location(ID,CAP,PER,STA,END)` represent a location (i.e. a gym or a room), with an identifier `ID`, a maximum capacity of `CAP`, which, 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` (i.e. a floor).
- 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*).

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

- 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 from `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 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(ID,LOC)`, which indicate the start, length, and location of each session, respectively.

Encoding. In Figure 3, 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.

Then, rules r_7 and r_8 define auxiliary atoms `extstart(ID,PER,TS)` and `extlength(ID,PER,TS)` using `TS` slots of times for the session with identifier `ID` on period `PER` reserved for the start and length extensions, respectively. 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` 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} trough 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 r_{19} through 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).

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

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

```
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  extstart(ID,PER,TS-LB) :- start(ID,PER,TS), before(ID,LB).
8  extlength(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),
       extstart(ID,PER,TS), extlength(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,_), extlength(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,_), extstart(ID,PER,TS), extlength(ID,PER,L), ST>=TS,
       ST<TS+L.
20 :- forbidden(PAT,PER,_,END), session(ID,PAT,_), extstart(ID,PER,TS), extlength(ID,PER,L), END>TS,
       END<=TS+L.
21 :- forbidden(PAT,PER,ST,END), session(ID,PAT,_), extstart(ID,PER,TS), extlength(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. 3. ASP Encoding for the agenda problem.

the preferred starting time for the sessions with *high* priority. 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 sessions having *low* priority.

4 Experimental analysis

In this section, the analyses performed on the two encodings are presented. The first part of our analysis is performed on real data coming from the institutes of Genova Nervi and Castel Goffredo; then, in order to evaluate the scalability of the approach and to analyze how our solution would behave in larger institutes having similar characteristics, an analysis is performed on synthetic instances with increasing dimensions, but considering real parameters. A comparison between the real and synthetic instances

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Table 1. *Dimensions of the ICS Maugeri's institutes.*

Institute	# Operators	# Patients	Density	# Floors	# Gyms
Genova Nervi	[9,18]	[37,67]	[2.4,5.2]	1	1
Castel Goffredo	[11,17]	[51,78]	[3.5,6.4]	2	3

Table 2. *Results on ICS Maugeri 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	1.1 s	–	–	–	10 s	0.01 s	–	–
Avg Time Last SM	1.3 s	30 s	5.2 s	30 s	12.1 s	21.3 s	10.4 s	30 s

validates the approach and demonstrates that synthetic instances can reasonably model the problem at hand. All these three parts are included, in separate paragraphs, in a first subsection, while a second subsection is devoted to a comparison to alternative logic-based formalisms. Encodings and benchmarks used in the experiments can be found at: `http://www.star.dist.unige.it/~marco/RuleMLRR2021TPLP/material.zip`.

4.1 Results of the encoding

Real data. ICS Maugeri utilizes a web-based software called QRehab (Saverino *et al.* 2021), which is built on top of the specified encoding; thus, 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 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, number of daily patients, density of patients per operator, number of floors (i.e. macro-locations), and number of total gyms (which we recall are locations in which multiple sessions can be performed in parallel). In Table 2, the results obtained by the two encodings are presented in terms of percentage of instances for which an optimal/satisfiable/no solution is computed, which also correspond to the three outcomes of interest for a practical use of our solution. The last two rows report the mean time of instances solved optimally and of the last computed solution for all satisfiable instances, respectively. The scheduling was performed using the ASP solver CLINGO (Gebser *et al.* 2012) with a cut-off of 30 s using two different optimization methods: The first is the default Branch&Bound (BB) optimization method (Gebser *et al.* 2015) with the option `--restart-on-model` enabled; the second leverages instead the Unsatisfiable Core (USC) algorithm (Andres *et al.* 2012) with the CLINGO options

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

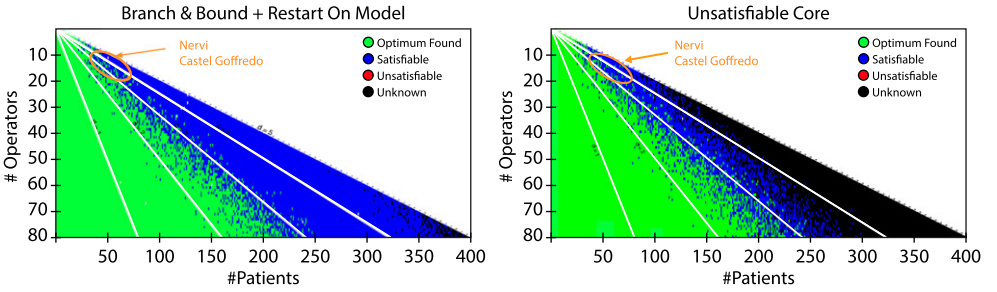

Fig. 4. Results of CLINGO using the BB optimization algorithm and the option `--restart-on-model` enabled (left) and the USC optimization algorithm (right) on synthetic benchmarks of the board.

`--opt-strategy=usc,k,0,4` and `--opt-usc-shrink=bin` enabled (which turn on the algorithm *k* (Alviano and Dodaro 2020) and the shrinking of the unsatisfiable cores (Alviano and Dodaro 2016), respectively). The cut-off of 30 *s* was chosen in order to be able to analyze a vast amount of experiments in overall reasonable time and has proven to be a sufficient amount of time to achieve meaningful results; in the software used daily by the ICS Maugeri, the cut-off is set to 300 s, as a means to solve even the hardest instances, having a limited number of instances to be run daily. As it can be seen in Table 2, results are mixed: the USC algorithm performs better in the agenda encoding while the 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 from Castel Goffredo a solution cannot be found. Considering these are hard real instances and cut-off time is limited, results are positive and highly appreciated by ICS Maugeri members.

Synthetic data. In order to understand how our solution scales to larger institutes having similar characteristics, a simulated approach is needed. For this reason, a generator able to produce random instances with features as close as possible to the ones of real hospitals was developed. Some examples of real data utilized are the percentage of individual and supervised sessions, the medium length of operator's shifts, the occurrence of forbidden time slots for patients, and the ideal length of sessions. For every new instance created, each feature was extracted from a random distribution which was modeled from the real data coming from the hospitals or from the knowledge of institute administrators and managers. In Figure 4, results of the scheduling of the board encoding, computed from the synthetic data, are presented. The x-axis defines the number of patients and the y-axis the number of operators; white lines represent points in which the density is an integer. Every pixel of the image depicts the mode of the results of 5 simulations executed with the corresponding number of patients and operators with a cutoff of 30 *s* using the BB optimization algorithm (left) and the USC optimization algorithm (right). The color of a pixel thus signals if the majority of instances with that particular number of operators and patients resulted in the following: (i) *Optimum Found*, signaling that the optimal stable model was found, (ii) *Satisfiable*, when at least one suboptimal stable model was found, but the solution is not guaranteed to be optimal, (iii) *Unknown*, if no stable model could be found before the cut-off, or (iv) *Unsatisfiable*, when no stable model exists which can satisfy all the constraints. As it can be seen from the figure,

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

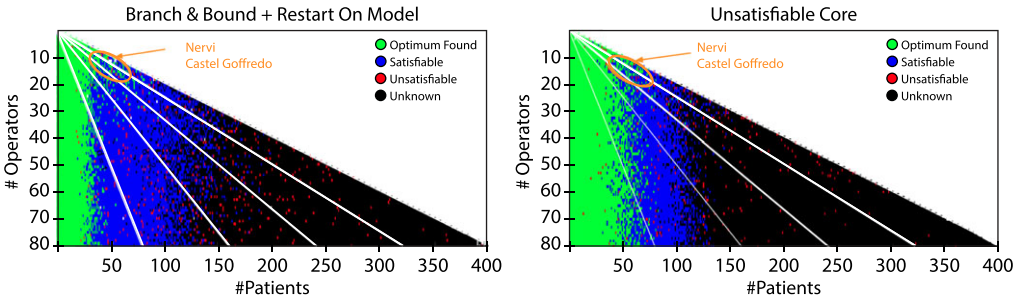

Fig. 5. Results of CLINGO using the BB optimization algorithm and the option `--restart-on-model` enabled (left) and the USC optimization algorithm (right) on synthetic benchmarks of the agenda.

the results of the scheduling are directly proportional to the density (i.e. the average number of patients for every operator), changing from *Optimum Found* to *Satisfiable* when reaching a density of approximatively 2.4 patients per operator. Notably, despite the use of random instances, no instance results *Unsatisfiable* since the fictitious operator can always catch the patients which cannot be assigned to any operator (due to all the operators reaching full capacity). The position of the hospitals of Genova Nervi and Castel Goffredo is highlighted with a circle. In this figure, it can be noted how BB gives better results than USC, by being able to find, before the cut-off, at least a suboptimal stable model for instances of higher densities, where, instead, the USC algorithm returns *Unknown*.

In Figure 5, the results of the agenda encoding, scheduled with the BB algorithm (left) and USC algorithm (right), are presented in the same format as for the previous experiment. The instances for this experiment are the same as the previous one, but are augmented with the assignment among patients and operators found by CLINGO with the board encoding and other needed parameters. As previously stated, each pixel represents 5 instances and its color represents the mode of the CLINGO results. Here two things can be noted: (i) unlike the board results, which showed a proportionality with the density, these results show a correlation only with the number of patients, and (ii) some red dots scattered in the image indicate that some instances result *Unsatisfiable*: this can happen since the random data could create some instances with features that cause an impossibility to schedule. With the BB optimization algorithm, the transition between the *Optimum Found* and *Satisfiability* results is located near 40 patients, and near 120 patients for the transition between *Satisfiability* and *Unknown*. As it can be seen in Figure 5 (right), the USC algorithm performs instead better and moves the transition between the *Optimum found* and *Satisfiable* results from 40 to 60 patients but, on the other hand, the transition between *Satisfiable* and *Unknown* slightly decreases from 120 patients to 110. The improvements on the transition between *Optimum found* and *Satisfiable* are very important in our setting, since Genova Nervi and Castel Goffredo fall into this area, confirming the improvements obtained in Table 2.

Validation of synthetic instances. In order to understand if the simulated instances correctly represent the real data and can be therefore used to predict the behavior of the system in larger institutes with similar characteristics, a validation is needed to compare the results obtained on real and synthetic instances. Intuitively, we have considered the

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

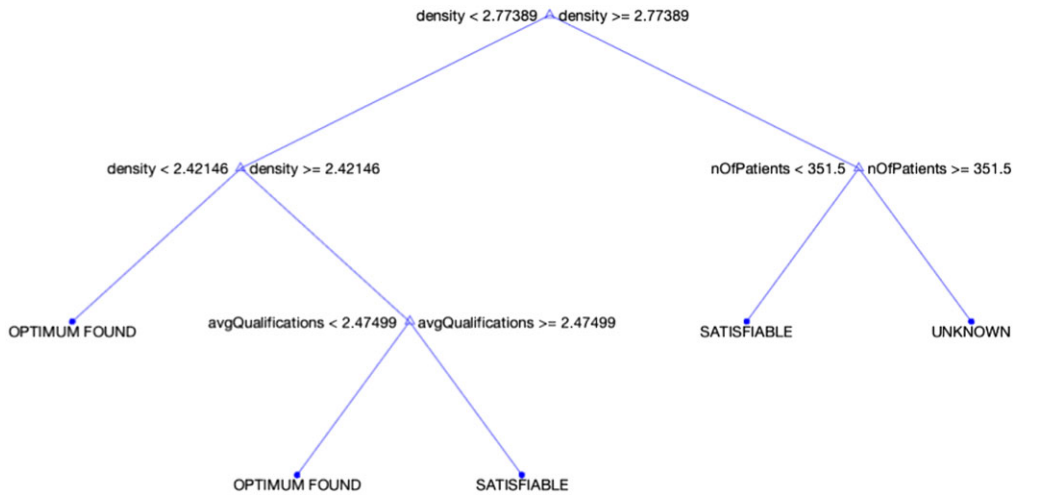

Fig. 6. Visual representation of the decision tree trained on the results found by CLINGO on real data utilizing the BB+RoM algorithm. The tree nodes represent features of the instance (density and average qualifications) and the leafs represent the result given by CLINGO (optimal found, satisfiable, unsatisfiable, unknown).

data presented in Table 2 and compared it to the results of the instances within the circles around Genova Nervi and Castel Goffredo in Figures 4 and 5, to check if they "coincide." For doing so, a decision tree was trained, taking as dataset all the features of the simulated instances, some of them listed in the previous paragraph. Then, a test dataset with features extracted from the real instances was produced and given as input to the decision tree, and the predicted result was then compared to the result given by CLINGO on the real instances. Figure 6 shows a visual representation of the decision tree trained on the results found by CLINGO on real data utilizing the BB+RoM approach. The tree nodes represent the most important features found by the decision tree approach, which allows a correct classification of the results of CLINGO, shown in the leaf nodes. The shown features are (i) the density, that is, the proportion patients/operator ratio, and (ii) the average number of qualifications, that is, the type of patients (orthopedic, neurological, covid-positive etc). Synthetic resources were then used as new data and given as input to the decision tree. The output of the decision tree was then compared with the actual result given by CLINGO. It can be seen, in fact, how the decision tree in Figure 6 is able to explain the results of the synthetic instances depicted in Figure 4 (left): the color green (representing optimality) is indeed classifiable only by the patients/operator density (i.e. the white diagonal lines) and the transition between the two results happen when the density is near 2.4, as classified, from the real data, by the decision tree. When the density is between 2.42 and 2.77 it can be noted how the average number of qualification (not explicitly shown in Figure 4) makes the difference between an optimal and suboptimal result (the more specialized the operators are, the fewer patients there are available for the planner to choose). This test showed that for the board encoding, all the results on real instances were equal to the predicted ones for both institutes; the agenda encoding produced instead the same results in 93% of the cases for Genova Nervi and in 67% of the cases for Castel Goffredo, thus showing that, overall, the synthetic data behave similarly to the real one and can be used for predicting the behavior of

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

instances in larger institutes having similar characteristics. Finally, the computed decision trees also confirm what are the most relevant features outlined above by inspecting the figures. In fact, if the height of the decision tree is increased, the accuracy of prediction does not improve that much, signaling that the features shown in Figure 6 are sufficient to explain the differences in the results of CLINGO.

4.2 Comparison to alternative logic-based formalisms

In the following, an empirical comparison of our ASP-based solution to alternative logic-based approaches is presented, obtained by applying automatic translations of our ASP encoding. In more detail, the ASP solver WASP (Alviano *et al.* 2019) was used, with the option `--pre=wbo`, which converts ground ASP instances into pseudo-Boolean instances in the wbo format (Roussel and Manquinho 2012). Then, the tool PYPBLIB (Ansótegui *et al.* 2019) was employed to encode wbo instances as MaxSAT instances. Moreover, given that other formalisms cannot handle multi-level optimizations, in order to provide a fair comparison, the ASP instances were processed using WASP with the option `--pre=lparse`, which collapses all weak constraints levels into one single level using exponential weights. With this approach, the costs found by the different approaches can be straightforwardly compared.

Three state-of-the-art MaxSAT solvers were considered, namely MAXHS (Saikko *et al.* 2016), OPEN-WBO (Martins *et al.* 2014), and RC2 (Ignatiev *et al.* 2019), as well as the industrial tool for solving optimization problems GUROBI (Gurobi Optimization, LLC 2021), which is able to process instances in the wbo format. Concerning CLINGO, the already presented BB+RoM and USC algorithms were used. The latter enables the usage of algorithm OLL (Morgado *et al.* 2014), which is the same algorithm employed by the MaxSAT solver RC2.

These experiments were run using the ASP encoding coming from the already presented real-world instances of the hospitals of Genova Nervi and Castel Goffredo. The experiments were conducted in the following way: firstly, the ASP encoding in which the weak constraints have been collapsed was transformed in the wbo and MaxSAT formats, then all the solvers were called with a cut-off of 30 *s* (the same used in all the other experiments). Then, for every formalism the following metrics were recorded: if it has found the optimum, the final cost and the time of computation. With these metrics, the formalisms can be ordered from best to worst based on their result: an optimal solution is better than one not declared optimal; if both are suboptimal then the one with the lowest cost is better; if both are optimal then the one which took less to declare optimality is better. In Table 3 the ranking among the formalisms is presented. For each of the different formalisms, the table shows the percentage of how many times it has arrived first, second, or third. *Solver TO* represents cases in which the solver was unable to find a solution before the cut-off (*Unknown*). *Pypblib TO* represents cases in which the tool PYPBLIB was unable to encode wbo instances as MaxSAT instances in a cut-off time of 60 *s*. For the board encoding both CLINGO's algorithms USC and BB+RoM are presented, since they showed comparable result; instead, in the agenda encoding only the USC algorithm is presented since it outperformed the BB+RoM in the previous tests.

The results show that for the board encoding, CLINGO is the most performant algorithm, coming first in almost all the experiments. In particular, CLINGO with the

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Table 3. *Comparison between alternative logic-based formalisms for the board and agenda phase.*

	Board						Agenda				
	BB+RoM	USC	MaxHS	OpenWBO	RC2	Gurobi	USC	MaxHS	OpenWBO	RC2	Gurobi
First	58%	41,7%	0,3%	–	–	–	77,1%	15,6%	7,3%	–	–
Second	41,7%	58,3%	1%	–	–	–	14,7%	54,2%	29,4%	–	–
Third	0,3%	–	10,3%	–	–	—	6,4%	28,4%	61,5%	–	–
Solver TO	–	–	9,3%	20,9%	20,9%	100%	1,8%	1,8%	1,8%	100%	100%
Pypblip TO	–	–	79,1%	79,1%	79,1%	–	–		–	–	–

BB+RoM optimization algorithm resulted more performant than the algorithm relying on the Unsatisfiable Core strategy, which is conformant with the experiments run on the multi-level version of the ASP encoding. For this encoding, it can be seen that for a high number of instances, around 80%, the tool PYPBLIB was unable to encode the MaxSAT instances within the cut-off. Still, in the remaining 20%, CLINGO remains the most performant algorithm. For the agenda encoding, CLINGO is still the best solver, but a more precise ranking among solvers can be noticed with MaxHS coming second and OpenWBO third. Notably, RC2 and Gurobi are, with both encoding, always unable to find a solution within the cut-off.

5 Domain-specific optimizations

Motivated by the analysis performed in the previous section, in which ASP outperformed other formalisms on translated (MaxSAT and pseudo-Boolean) formulas, we apply domain specific optimizations to our ASP encoding, with the aim of further improving the solving time and move towards solving larger instances. In Section 4, benchmarks for the board and agenda phases were presented, showing different results based on the optimization algorithm chosen (i.e. BB+RoM or USC). The domain-specific optimizations are presented to mainly decrease grounding, and consequently planning times, with the aim, as mentioned, of being able to find optimum solutions in larger instances, for example, possibly corresponding to larger hospitals. The optimizations are presented only on the agenda encoding, which is the more complicated of the two phases and has still a great margin of improvement via changes in the encoding. These changes all rely on the knowledge of the RSP domain and on the possibility to prune impossible solutions already in the grounding process, avoiding wasting time in search. The section is organized in two subsections, in which the first presents the changes and improvements done on the previous agenda encoding, while the second subsection focuses on the results.

5.1 Optimized encoding

The next two paragraphs present the specific domain optimizations introduced.

Pruning of session starts. As it can be seen in Figure 3, in rules r_1 and r_2 the start of a session is guessed between all the possible time slots in the shift of an operator, expressed via the atom `time(PER,OP,TS)`. These guess rules can be improved by reducing the number of time slots in which it is possible to start a session with the following constraints:

1. a session cannot start in a time slot near to the operator' shift's end. This is because the minimum specified time of the session would not be satisfied, given the shift's end;

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

```
29 forbiddenRange(ID,PER,XSTA,END) :- forbidden(PAT,PER,STA,END), session(ID,PAT,_),
       session_length(ID,MIN,_), XSTA = STA - MIN + 1.
30 forbiddenSlot(ID,PER,STA..END-1) :- forbiddenRange(ID,PER,STA,END).
31 allowedTime(ID,PER,T) :- time(PER,OP,T), session(ID,_,OP), session_length(ID,MIN,_),
       period(PER,OP,_,END), T <= END - MIN, not forbiddenSlot(ID,PER,T).
32 1 {start(ID,PER,TS) : allowedTime(ID,PER,TS)} 1 :- session(ID,_,OP), mandatory_session(ID).
33 0 {start(ID,PER,TS) : allowedTime(ID,PER,TS)} 1 :- session(ID,_,OP), optional_session(ID).
```

Fig. 7. Optimized encoding for pruning the session starts.

2. if a patient has a forbidden time (i.e. a time interval where the patient cannot be scheduled), the session cannot start during the forbidden time. Moreover, some timeslots before the forbidden times should be excluded beforehand since, if the session started in these timeslots, this would not allow it to end before the forbidden time starts.

In Figure 7, the ASP encoding for pruning the session starts is shown. In r_{29}, a new atom `forbiddenRange` is defined for the purpose of extending the start of forbidden times by including the time slots which would not allow the session to end before the start of the forbidden time. Rule r_{30} spreads `forbiddenRange` in all the time slots (`forbiddenSlot`) within the range. In rule r_{31}, a new atom `allowedTime` is defined as a time slot in the shift of the operator which is not a `forbiddenSlot`, and which allows the session, with its min length `MIN`, to end before the end of the shift `END`. In rules r_{32} and r_{33}, the atom `allowedTime` replaces the more broad atom `time` in the guess rule of the start of the session. In the optimized encoding, rules from r_{29} to r_{33} replace rules r_1 and r_2 of the original encoding (Figure 3).

Pruning of session extension. As stated in Section 2, the agenda encoding relies on two auxiliary atoms (`extstart` and `extlength`) as a means to reserve slots of time before it starts and after it ends to each session, in which the session can be performed in a supervised fashion. These before (after) slots of time are decided with a guess rule on the atom `before` (`after`) in rule r_5 (r_6) of Figure 3. The definition of these atoms can be improved in order to reduce the number of ground instantiations, in the following way:

1. as described in the previous paragraph, the extended part of a session cannot start during a forbidden time;
2. the `before` and `after` timeslots cannot be greater than the difference between the ideal length of a session and its minimum length. Since the weak constraints impose a minimization of the distance between the final length of the session and its ideal length, this last acts as an upper bound of the length of the session;
3. if there is already an extension before the session, the extended length of the session cannot be longer than the ideal length of the session.

Rules r_5, r_6, r_7, and r_8 of the agenda encoding presented in Figure 3 can be substituted by the encoding presented in Figure 8. Rule r_{34} states that a value for the before extension can be computed taking the difference between the start of the session and an allowed time slot distant no more than the difference between the ideal and minimum length of a session. Rule r_{35} defines the auxiliary atom `extstart` via the previously guessed `before` atom. Rule r_{36} finds the amount to reserve after the session in the same way

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Table 4. *Comparison, in terms of grounding, between the basic and the optimized encoding on real instances coming from the Maugeri's hospitals.*

	Basic		Optimized	
	Nervi	C.G.	Nervi	C.G.
Avg Grounding Time	8.3 s	11.5 s	0.7 s	0.9 s
Avg Number of Variables	323 k	587 k	51 k	59 k
Avg Number of Rules	3.8 M	11.4 M	177 k	327 k

Table 5. *Results of the optimized agenda encoding on real instances.*

	Basic+BB+RoM		Basic+USC		Opt+BB+RoM		Opt+USC	
	Nervi	C.G.	Nervi	C.G	Nervi	C.G.	Nervi	C.G.
% Optimum	0%	0%	45%	0%	0%	0%	82%	74%
% Satisfiable	100%	67%	55%	70%	100%	100%	18%	26%
% Unknown	0%	33%	0%	30%	0%	0%	0%	0%
Avg Time for opt	–	–	0.01s	–	–	–	0.01 s	4.7 s
Avg Time Last SM	30 s	30 s	21.3 s	30 s	19.6 s	20 s	20.4 s	8.0 s

```
34 1 {before(ID,NL): allowedTime(ID,PER,T), T<=TS, NL=TS-T, NL<=IDEAL-MIN} 1 :- start(ID,PER,TS),
       session(ID,_,OP), session_length(ID,MIN,IDEAL).
35 extstart(ID,PER,TS-LB) :- start(ID,PER,TS), before(ID,LB).
36 1 {after(ID,NL): time(PER,OP,T), T>=TS+LEN, NL=T-TS-LEN, NL<=IDEAL-MIN} 1 :- start(ID,PER,TS),
       length(ID,PER,LEN), session(ID,_,OP), session_length(ID,MIN,IDEAL).
37 extlength(ID,PER,LEN) :- length(ID,PER,L), after(ID,LA), before(ID,LB), session(ID,_,_),
       session_length(ID,_,IDEAL), LEN=L+LA+LB, LEN <= IDEAL.
38 :- start(ID, _, _), not extlength(ID, _, _).
```

Fig. 8. Optimized encoding for pruning of session extension.

as expressed with the `before` atom. Rule r_{37} defines the auxiliary atom `extlength` now being limited by the ideal length. Rule r_{38} imposes that a session must have an extended length (which can correspond to the individual length if no supervised time is needed); this is to avoid solutions in which the planner increases to the maximum both the before and after extensions of a session as a shortcut to falsify all instantiations of rule r_{37} by not having an extended length less than the ideal length.

5.2 Results of the optimized encoding

The next two paragraphs present the performance of the optimized encoding on real and synthetic instances, respectively.

Real Data. In Tables 4 and 5, the results of the optimized encoding on real instances of the hospitals in Genova Nervi and Castel Goffredo are presented. Table 4 presents a comparison about the grounding between the two encodings, showing the significant

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

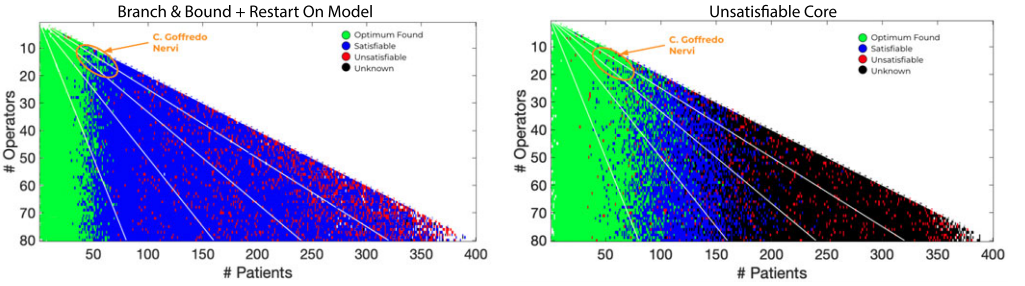

Fig. 9. Results of the synthetic benchmarks of the agenda produced by CLINGO with the optimized encoding.

reduction in terms of time, number of variables and number of rules that the optimized encoding brings. Table 5 then shows the results for the basic agenda encoding presented in Section 3.2 with the two algorithms BB+RoM and USC (this part of the table is copied from Table 2). The other half of the table shows the results for the optimized encoding presented in the previous subsection, again with the two algorithms. As it can be seen, the optimized encoding boosts the performances, especially when combined with the USC algorithm. Comparing the two encodings, the first thing that can be noticed is that the optimized encoding is able to find a solution for each instance. Moreover, it can be seen that:

- even if in Genova Nervi the percentages remain the same for the BB algorithm, the average time in which the last stable model is outputted decreases.
- with the USC algorithm, using the optimized encoding, for most of the instances both in Genova Nervi and Castel Goffredo, an optimal solution can be found.

Synthetic data. As previously explained in Section 4, testing an encoding on synthetically generated instances is important to understand how our solution could scale to larger institutes having similar characteristics. Figure 9 shows the results of the scalability test performed with the new optimized encoding, where the meaning of the colors, axes and lines has been already explained in Section 4. On the left, the results of running the optimized encoding using CLINGO with BB+RoM settings; on the right, the result are computing against the optimized encoding leveraging the USC algorithm. Comparing Figure 9 (optimized encoding, Opt) with Figure 5 (basic encoding, Basic) serious improvements can be noted:

- comparing the best combinations, that is, Basic+USC and Opt+BB+RoM, it can be noted how the transition between solution with *Optimum Found* and *Satisfiable* stays approximatively the same near 50 patients, but *Unknown* results no longer appear, meaning that in the cut-off of 30 s CLINGO can find a suboptimal solution. In fact, the aim of the optimization was to reduce to the minimum the grounding time, which has left now the largest part of the 30 s cutoff to be spent in actually solving.
- focusing on Opt, the results of Opt+BB+RoM and Opt+USC show the supremacy of the USC algorithm.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Table 6. *Comparison between alternative logic-based formalisms for the optimized agenda phase.*

	Agenda				
	USC	MaxHS	OpenWBO	RC2	Gurobi
First	94,2%	3,1%	0,0%	–	–
Second	3,1%	72,3%	21,5%	–	–
Third	0,0%	21,5%	75,5%	–	–
Solver TO	2,7%	3,1%	3,0%	100%	100%
Pypblip TO	–	–	–	–	–

Focusing on optimization algorithms, as it can be seen from the figures, the results on these benchmarks are comparable with the ones performed on real data: using the BB+RoM algorithm in fact it can be noted how the area of the graph with properties similar to the hospitals of Nervi and Castel Goffredo (the orange circle) have most of its area of a blue color (representing Satisfiable results) and only for easy examples an optimal solution can be found; using the USC approach, similarly, we can see that in the circle fall most solutions found with optimality, thus confirming the results on the real instances. In fact, comparing the results of Basic+USC and Opt+USC it can be seen a real improvement in the number of instances which can be now solved with optimality: before, with the Basic+USC approach, the transition between solutions *Optimum Found* and *Satisfiable* lied near 50 patients, while now has reached almost 90 patients.

At last, we also compared our approach with algorithm USC (that the analysis demonstrated to be the best) to the other logic-based formalisms already employed in Table 3, using the same evaluation metric and presentation. As it is clear from Table 6, the ASP approach is the best also when considering the optimized encoding.

6 Related work

This paper is an extended and revised version of Cardellini *et al.* (2021), having the following main consistent additions: (*i*) a comparison to alternative logic-based formalisms on real instances (Subsection 4.2), and (*ii*) the definition and related experimental evaluation of two domain-specific optimizations (Section 5).

There have been few attempts in the literature 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 at ICS Maugeri. In particular, to our knowledge, no other solution takes into account several aspects like the preferred time for the session scheduling and the preferences in the assignment of the patient to the operator. Huang *et al.* (2012) developed a system, equipped with a Graphical User Interface, 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, Huynh *et al.* (2018) 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 (2021) designed a genetic algorithm based on Waiting Time Priority Algorithm (WTPA), which was tested

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

on a rehabilitation department. Schimmelpfeng *et al.* (2012) 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. We already mentioned in the introduction that ASP has been already successfully used for solving application problems in several research areas (see, e.g. Gebser *et al.* (2016) for routing driverless transport vehicles, Ricca *et al.* (2012) for team scheduling, and Erdem *et al.* (2016) for a general overview) including scheduling problems in the Healthcare domain (see, e.g. Alviano *et al.* (2020) for an overview focused on them). Differently from previous papers in the Healthcare domain, the current work focuses on the rehabilitation scheduling problem, that was not addressed before using ASP, and it combines a two-phase encoding, rather than the usually employed direct encoding, with the evaluation of the solution on real benchmarks. Concerning the experimental analysis, similarly to Dodaro *et al.* (2021), in this work we compared our ASP-based solution with alternative logic-based approaches.

Finally, in Saverino *et al.* (2021) an analysis of the usage of the tool in the hospital of Genova Nervi for a period of approx. 6 months is reported. As an example, statistics about the sessions planned by our ASP encodings and their actual durations in the hospital usage are recorded. As shown in the paper, reported and planned session lengths are similar, with the ratio between these two quantities has been between 0.95 and 1.05 for the 95% of the considered time span.

7 Conclusion

In this paper, we have presented a two-phase ASP encoding for solving rehabilitation scheduling. We have started from a general solution, then extended with domain-specific optimizations. Our solution has been tested with CLINGO and both real and synthetic benchmarks, the former provided by ICS Maugeri while the latter created with real parameters and employed to understand a possible behavior of the solution on upcoming institutes where the solution will be employed. Results are satisfying for the institutes employed at the moment and give some indications about the upcoming institutes ICS Maugeri plans to instrument with this solution. Domain-specific optimizations further improve the results, by also diminishing the number of instances for which a solution cannot be found in short time. Future research includes a more fine-grained analysis of our solution by, for example, combining the strengths of the optimization algorithms, analyzing further dimensions of our encoding, for example, number of floors and gyms for synthetic benchmarks, and benchmarking the impact of the introduced domain-specific optimizations separately.

Competing interests

The authors declare none.

References

Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., and Ricca, F. Evaluation of disjunctive programs in WASP. In *LPNMR 2019* 2019, vol. 11481 of *LNCS*. Springer, 241–255.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M. K., Maratea, M., Mochi, M., Morozan, V., Porro, I. and 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* 2020, vol. 2745 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Alviano, M. and Dodaro, C. 2016. Anytime answer set optimization via unsatisfiable core shrinking. *Theory and Practice of Logic Programming*, *16*, 5–6, 533–551.

Alviano, M. and Dodaro, C. 2020. Unsatisfiable core analysis and aggregates for optimum stable model search. *Fundamenta Informaticae*, *176*, 3–4, 271–297.

Andres, B., Kaufmann, B., Matheis, O. and Schaub, T. Unsatisfiability-based optimization in clasp. In *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012* 2012, vol. 17 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 211–221.

Ansótegui, C., Pacheco, T. and Pon, J. 2019. Pypblib.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Brewka, G., Eiter, T. and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM*, *54*, 12, 92–103.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F. and Schaub, T. 2020. ASP-Core-2 input language format. *Theory and Practice of Logic Programming*, *20*, 2, 294–309.

Cardellini, M., Nardi, P. D., Dodaro, C., Galatà, G., Giardini, A., Maratea, M. and Porro, I. A two-phase ASP encoding for solving rehabilitation scheduling. In S. Moschoyiannis, R. Peñaloza, J. Vanthienen, A. Soylu and D. Roman, Eds., *Proceedings of the 5th International Joint Conference on Rules and Reasoning (RuleML+RR 2021)* 2021, volume 12851 of *Lecture Notes in Computer Science*. Springer, 111–125.

Cieza, A., Causey, K., Kamenov, K., Hanson, S. W., Chatterji, S. and Vos, T. 2020. 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.

Dodaro, C., Galatà, G., Grioni, A., Maratea, M., Mochi, M. and Porro, I. 2021. An ASP-based solution to the chemotherapy treatment scheduling problem. *Theory and Practice of Logic Programming*, *21*, 6, 835–851.

Dodaro, C. and Maratea, M. Nurse scheduling via answer set programming. In M. Balduccini and T. Janhunen, Eds., *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2017)* 2017, volume 10377 of *Lecture Notes in Computer Science*. Springer, 301–307.

Erdem, E., Gelfond, M. and Leone, N. 2016. Applications of answer set programming. *AI Magazine*, *37*, 3, 53–68.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T. and Wanko, P. Theory solving made easy with clingo 5. In M. Carro, A. King, N. Saeedloei and M. D. Vos, Eds., *Proceedings of ICLP (Technical Communications)* 2016, vol. 52 of *OASICS*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2:1–2:15.

Gebser, M., Kaminski, R., Kaufmann, B., Romero, J. and Schaub, T. Progress in clasp Series 3. In *LPNMR* 2015, vol. 9345 of *LNCS*. Springer, 368–383.

Gebser, M., Kaufmann, B. and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, *187*, 52–89.

Gebser, M., Obermeier, P., Schaub, T., Ratsch-Heitmann, M. and Runge, M. 2018. Routing driverless transport vehicles in car assembly with answer set programming. *Theory Practice of Logic Programming*, *18*, 3–4, 520–534.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, *9*, 3/4, 365–386.

Gurobi Optimization, LLC 2021. Gurobi Optimizer Reference Manual.

Huang, Y.-C., Zheng, J.-N. and Chien, C.-F. 2012. Decision support system for rehabilitation scheduling to enhance the service quality and the effectiveness of hospital resource management. *Journal of the Chinese Institute of Industrial Engineers*, *29*, 348–363.

Huynh, N.-T., Huang, Y.-C. and Chien, C.-F. 2018. A hybrid genetic algorithm with 2D encoding for the scheduling of rehabilitation patients. *Computers & Industrial Engineering*, *125*, 221–231.

Ignatiev, A., Morgado, A. and Marques-Silva, J. 2019. RC2: an efficient maxsat solver. *Journal of Satisfiability, Boolean Modeling, and Computation*, *11*, 1, 53–64.

Li, X. and Chen, H. 2021. Physical therapy scheduling of inpatients based on improved genetic algorithm. *Journal of Physics: Conference Series*, *1848*, 1, 012009.

Martins, R., Manquinho, V. M. and Lynce, I. Open-wbo: A modular maxsat solver,. In *SAT 2014* 2014, vol. 8561 of *LNCS*. Springer, 438–445.

Morgado, A., Dodaro, C., and Marques-Silva, J. Core-Guided MaxSAT with Soft Cardinality Constraints. In *Proceedings of Principles and Practice of Constraint Programming - 20th International Conference, CP 2014* 2014, Lyon, France. Springer, 564–573.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, *25*, 3-4, 241–273.

Roussel, O. and Manquinho, V. 2012. Input/Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, *1*, 1, 81–106.

Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S. and Leone, N. 2012. Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming*, *12*, 3, 361–381.

Saikko, P., Berg, J. and Järvisalo, M. LMHS: A SAT-IP hybrid maxsat solver. In *SAT 2016* 2016, vol. 9710 of *LNCS*. Springer, 539–546.

Saverino, A., Baiardi, P., Galata, G., Pedemonte, G., Vassallo, C. and Pistarini, C. 2021. 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.

Schimmelpfeng, K., Helber, S. and Kasper, S. 2012. Decision support for rehabilitation hospital scheduling. *OR Spectrum*, *34*, 2, 461–489.

https://doi.org/10.1017/S1471068423000030 Published online by Cambridge University Press