TPLP **21** (6): 835–851, 2021. © The Author(s), 2021. Published by Cambridge University Press
doi:10.1017/S1471068421000363 First published online 24 September 2021

An ASP-based Solution to the Chemotherapy Treatment Scheduling problem

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

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

ANDREA GRIONI
San Martino Hospital, Italy
(*e-mail:* andrea.grioni@hsanmartino.it)

MARCO MARATEA and MARCO MOCHI
University of Genoa, Italy
(*e-mail:* marco.mochi@unige.it)

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

submitted 05 August 2021; revised 23 August 2021; accepted 24 August 2021

Abstract

The problem of scheduling chemotherapy treatments in oncology clinics is a complex problem, given that the solution has to satisfy (as much as possible) several requirements such as the cyclic nature of chemotherapy treatment plans, maintaining a constant number of patients, and the availability of resources, for example, treatment time, nurses, and drugs. At the same time, realizing a satisfying schedule is of upmost importance for obtaining the best health outcomes. In this paper we first consider a specific instance of the problem which is employed in the San Martino Hospital in Genova, Italy, and present a solution to the problem based on Answer Set Programming (ASP). Then, we enrich the problem and the related ASP encoding considering further features often employed in other hospitals, desirable also in S. Martino, and/or considered in related papers. Results of an experimental analysis, conducted on the real data provided by the San Martino Hospital, show that ASP is an effective solving methodology also for this important scheduling problem.

KEYWORDS: healthcare, chemotherapy treatment scheduling, answer set programming

1 Introduction

The Chemotherapy Treatment Scheduling (CTS) (Hahn-Goldberg *et al*. 2014; Huang *et al*. 2017; Huggins *et al*. 2014; Sevinc *et al*. 2013) problem consists of computing a schedule for patients requiring chemotherapy treatments. The CTS problem is a complex problem for oncology clinics since it involves multiple resources and aspects, including the

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

availability of nurses, chairs, and drugs. Chemotherapy treatments have a cyclic nature, where the number and the duration of each cycle depend on the different types of cancer and the stage of the disease. Moreover, treatments may have different priorities that must be taken into account when computing a solution. A proper solution to the CTS problem is thus crucial for improving the degree of satisfaction of the main actors on the problem, that is, patients and nurses, and for a better management of the resources. Various studies, also in the context of the COVID19 emergency (Kumar and Dey 2020; Sud *et al*. 2020), have shown how delays in cancer surgeries and treatments have a significant adverse impact on patient survival. This impact varies depending on the aggressiveness of the cancer, thus stressing the importance of developing a model capable of efficiently prioritize patients.

Complex combinatorial problems, possibly involving optimizations, such as the CTS problem, are usually the target applications of AI languages such as Answer Set Programming (ASP). Indeed, ASP has been successfully employed for solving hard combinatorial problems in several research areas, and it has been also employed to solve many scheduling problems also in industrial contexts (see, e.g. the work by Erdem *et al*. (2016), Falkner *et al*. (2018), Schüller (2018), Alviano *et al*. (2020) for detailed descriptions of ASP applications).

In this paper, we apply ASP for solving the CTS problem. We first consider a specific instance of the problem which is employed in the San Martino Hospital in Genova, Italy (Section 2 and 3). The problem of the San Martino Hospital consists of assigning a chair or a bed and an hour of treatment to each patient for each fixed day. Some patient requires a bed and the solution has to meet as much request of beds as possible. As an optimization at the moment not implemented in the San Martino Hospital, but highly desired, our solution assigns the patients in such a way to have, as much as possible, the same number of patients during the blood collection performed before the treatment. Moreover, we consider a planning horizon of one week other than the daily actually employed in the hospital. Then, we enrich the problem and the related ASP encoding considering features often employed in other hospitals, considered in related papers (Turkcan *et al*. 2012; Turkcan *et al*. 2012; Heshmat and Eltawil 2021; Huggins *et al*. 2014; Dodaro *et al*. 2018), and/or desired by the S. Martino Hospital, that is, we explicitly consider the availability of nurses and drugs or a limit to the starting time of treatments (Section 3.2). Both encoding are evaluated on real data of the San Martino Hospital (Section 4): results using the state-of-the-art ASP solver CLINGO (Gebser *et al*. 2012) show that ASP is an effective solving methodology for solving all these variants of the presented CTS problem. Focusing on the encoding of the San Martino Hospital, we were able to obtain better schedule w.r.t. the ones that have been implemented in practice, and to obtain very satisfying solutions (i.e. verified by a domain expert) in short time for planning horizons of 1 week instead of one day.

Further, we investigate rescheduling solutions for the weekly real world problem for dealing with situations in which the original schedule cannot be implemented, due to patients that report unavailability (Section 5). Goal of the rescheduling is to postpone the canceled registrations, minimizing the changes to the planned schedule while of course still satisfying all the constraints that are in place. Results on some scenario with increasing number of canceled registrations and changed regimen show that rescheduling is done efficiently, in short timings in line with the need of practical applications.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

2 Problem description

In this section, we first present the CTS problem as implemented at the San Martino Hospital in Genova, and then its mathematical formulation.

2.1 CTS in the real world

The CTS problem consists of scheduling appointments to a given day for patients requiring chemotherapy treatments, and to assign to each patient a chair or a bed for the whole duration of the session if required. Patients can need different treatments and we identify four phases for each of them: (1) the registration to the Hospital reception; (2) a blood collection; (3) a medical check; and (4) the therapy. The duration of each phase can be different for each patient. Moreover, phases 1 and 4 are mandatory, whereas phases 2 and 3 are optional. However, if a patient is involved in phase 2 then also phase 3 is required. The number of phases and their duration is assigned to a patient during the registration of the appointment. Moreover, some patient might not need a bed or a chair, so for those patients the duration of phase 4 is 0.

Office hours are from 07:30 A.M. to 01:30 P.M. We consider a set of 72 time slots for each day with a duration of 5 minutes, where 07:30–07:35 is the first time slot, and 01:25–01:30 is the last one. We also identify a set of 36 available time slots which represent the possible starting time of phase 4, where 07:35–07:40 is the first time slot, 07:45-07:50 is the second time slot, and so forth. Moreover, if the duration of phase 4 for a registration exceeds a given threshold, then it must start after 11:25 A.M. (i.e. the 24th time slot). Finally, phase 4 must start after all previous phases are completed.

The input of the problem consists of registrations, where each registration includes the duration of each phase (set to 0 if a phase is not required) and the preference between chair or bed.

A solution to the problem is represented by a schedule of registrations to time slots for a given day (representing the beginning of phase 4) according to the following requirements: *(i)* each patient must be assigned to a chair or bed; *(ii)* each chair or bed can be used by only one patient for each time slot; and *(iii)* if the treatment requires more than one time slots, then the patient must always use the same chair or bed. Moreover, an optimal solution to the problem maximizes the number of patients that are assigned to the preferred resource (chair or bed). Ties are broken by minimizing the number of concurrent patients in phase 2 to have a more uniform usage of resources during the day.

Weekly CTS problem. The CTS problem aims at computing a scheduling solution for one day. Nevertheless, the definition of the problem can be slightly modified for computing a scheduling solution for one week. The main difference is that the registration includes a natural number expressing an ordering on the therapies for the patient. For instance, given a patient, there might be several registrations associated to her/him which have different duration of the phases and different preferences between chair and bed. Moreover, each patient must wait a certain number of days (depending on the therapy) from one appointment to the subsequent one.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

2.2 Formalization of the CTS problem

Definition 1 (*Weekly CTS problem*)
Let

- I be a finite set of identifiers of patients;
- $O \subset \mathbb{N}$ be a finite set of orders;
- $R \subseteq I \times O$ be a set of registrations, such that if $(i, o_1) \in R$ then for each $o_2 < o_1$, $(i, o_2) \in R$;
- D be a finite set of days;
- $ATS = \{t \mid t \in [1..72]\}$ be the set of all time slots;
- $TS = \{t \cdot 2 \mid t \in [1..36]\}$ be the set of available time slots;
- $P = \{1, 2, 3, 4\}$ be the set of phases;
- $B = \{b_1, \ldots, b_m\}$ be a set of m beds;
- $C = \{c_1, \ldots, c_n\}$ be a set of n chairs;
- $\delta : R \times P \mapsto \mathbb{N}$ be a function associating a registration and a phase to a duration such that for a registration r if $\delta(r, 2) \neq 0$ then $\delta(r, 3) \neq 0$;
- $\rho : R \mapsto \{bed, chair\}$ be a function associating the preference of a registration to beds or chairs.
- $\omega : R \mapsto \mathbb{N}^+$ be a function associating a registration to a waiting time.

Let $x : R \times D \times TS \times (B \cup C) \mapsto \{0, 1\}$ be a function such that $x(r, d, ts, s) = 1$ if the registration r is assigned to the day d and time slot ts with a bed or a chair s, and 0 otherwise. Moreover, for a given x let $A_x = \{(r, d, ts, s) \mid r \in R, d \in D, ts \in TS, s \in (B \cup C), x(r, d, ts, s) = 1\}$.

Then, given sets I, O, R, D, ATS, TS, P, B, C, and functions δ, ρ, ω, the weekly CTS problem is defined as the problem of finding a schedule x, such that

$$
\begin{aligned}
&c(1)\ |\{(d, ts) : (r, d, ts, s) \in A_x\}| = 1 && \forall r \in R;\\
&c(2)\ |\{s : (r, d, ts, s) \in A_x\}| = 1 && \forall r \in R \mid \delta(r, 4) > 0;\\
&c(3)\ x(r_2, d, ts_2, s) = 0 && \forall r_2 \in R, ts_2 \in TS, (r, d, ts, s) \in A_x \mid\\
& && \quad r \neq r_2, ts \leq ts_2 \leq ts + \delta(r, 4);\\
&c(4)\ x(r, d, ts, s) = 0 \mid ts \in \{1, \ldots, 23\} && \forall r \in R, \delta(r, 4) > 50;\\
&c(5)\ ts - \delta(r, 1) - \delta(r, 2) - \delta(r, 3) > 0 && \forall (r, d, ts, s) \in A_x;\\
&c(6)\ |\{((i, o_1), d_1, ts_1, s_1) \in A_x\}| = 1 && \forall ((i, o_2), d_2, ts_2, s_2) \in A_x \mid o_1 = o_2 + 1,\\
& && \quad d_1 = d_2 + \omega(i, o_1).
\end{aligned}
$$

Condition ($c1$) ensures that each registration must be assigned exactly once. Condition ($c2$) ensures that each patient requiring a seat must be assigned to exactly one chair or bed. Condition ($c3$) ensures that each chair or bed cannot be assigned to different patients during the same time slots. Condition ($c4$) ensures that if phase 4 is particularly long, then it must start after 11:25 A.M. (i.e. the 25th time slot). Condition ($c5$) ensures that phase 4 cannot start before other phases are concluded. Condition ($c6$) ensures that each patient waits a given number of days between one appointment and the subsequent one.

Definition 2 (*Missed preferences*)
Given a solution x, let $m^*_x = |\{r \mid (r, d, ts, s) \in A_x, s \in B, \rho(r) = chair\} \cup \{r \mid (r, d, ts, s) \in A_x, s \in C, \rho(r) = bed\}|$. Intuitively, m^*_x represents the number of registrations whose preference for beds/chairs are not fulfilled.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

Definition 3 (*Distribution of registrations*)
Given a solution x and a time slot $ts \in ATS$, let $d_x^{ts} = |\{r \mid (r, d, ts', s) \in A_x, \delta(r, 2) \neq 0, ts = ts' - \delta(r, 3) - \delta(r, 2)\}|$, that is, the number of registrations whose phase 2 starts at the time slot ts. Given a solution x and a day $d \in D$, let $g_x^d = |\{r \mid (r, d, ts, s) \in A_x\}|$, that is, the number of registrations of the day d. Given a solution x, let $d_x^* = \max(\{d_x^{ts} \mid ts \in ATS\}) - \min(\{d_x^{ts} \mid ts \in ATS\})$ and $g_x^* = \max(\{g_x^d \mid d \in D\})$.

Definition 4 (*Optimal solution*)
A solution x is said to dominate solution x' if $m_x^* < m_{x'}^*$, or if $m_x^* = m_{x'}^*$ and $d_x^* < d_{x'}^*$, or if $m_x^* = m_{x'}^*$ and $d_x^* = d_{x'}^*$ and $g_x^* < g_{x'}^*$. A solution is *optimal* if it is not dominated by any other solution.

Daily CTS problem. Finally, the daily CTS problem can be defined by setting $O = \{0\}$ and $D = \{1\}$, and, thus, Definitions 2–4 also hold for it.

3 ASP encoding

In this section we first present the ASP encoding for the weekly CTS problem, and then the main elements of the extended CTS problem, in two separate sub-sections.

3.1 ASP encoding for the (weekly) CTS problem

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 the paper by Calimeri *et al.* (2020).

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

- Instances of `reg(RID,ORDER,WDAY,PH4,PH3,PH2,PH1,S)` represent the registrations, characterized by an id (`RID`), the ordering (`ORDER`), the number of waiting days before the visit (`WDAY`), the duration of each phase (`PH4`, ..., `PH1`), and the preference for chair or bed (`S`), where `S` can be `"bed"` or `"chair"`.
- Instances of `day(DAY)` represent the available days.
- Instances of `ts(TS)` represent the available time slots for phase 4, where `TS` ranges from `1` to `36`.
- Instances of `ats(TS)` represent all time slots, where `TS` ranges from `1` to `72`.
- Instances of `chair(ID)` represent the available chairs, with its identifier `ID`.
- Instances of `bed(ID)` represent the available beds, with its identifier `ID`.

The output is an assignment represented by atoms of the form:

```
x(RID,DAY,TS,PH4,ORDER,S)
```

where the intuitive meaning is that the phase 4 of registration with id `RID` and ordering `ORD` is assigned to the day `DAY` and time slot `TS` using the chair or bed `S`.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

```
1  {x(RID,DAY,TS,PH4,0,S) : ts(TS), day(DAY)} = 1 :- reg(RID,0,_,PH4,PH3,PH2,PH1,S).
2  {x(RID,DAY+DAY2,TS,PH4,ORDER,S) : ts(TS)} = 1 :- x(RID,DAY,_,_,N,_), ORDER=N+1, day(DAY+DAY2),
       reg(RID,ORDER,DAY2,PH4,PH3,PH2,PH1,S).
3  :- x(RID,DAY,TS,PH4,_,_), PH4 > 50, TS < 24.
4  :- x(RID,DAY,TS,_,ORDER,_), reg(RID,ORDER,DAY2,PH4,PH3,PH2,PH1,S), TS-PH3-PH2-PH1<1.
5  1 {bed(ID,RID,DAY) : bed(ID); chair(ID,RID,DAY) : chair(ID)} 1:- x(RID,DAY,_,_,_,_).
6  res(RID,DAY,TS..TS+PH4-1) :- x(RID,DAY,TS,PH4,_,_), PH4 > 0.
7  chair(ID,RID,DAY,TS) :- chair(ID,RID,DAY), res(RID,DAY,TS).
8  bed(ID,RID,DAY,TS) :- bed(ID,RID,DAY), res(RID,DAY,TS).
9  :- #count{RID: chair(ID,RID,DAY,TS)} > 1, day(DAY), ts(TS), chair(ID).
10 :- #count{RID: bed(ID,RID,DAY,TS)} > 1, day(DAY), ts(TS), bed(ID).
11 support(RID,DAY,TS) :- x(RID,DAY,PH4,_,_,_), reg(RID,ORDER,_,_,PH3,PH2,_,_), PH2 > 0, TS=PH4-PH3-PH2,
       day(DAY), ats(TS).
12 numbReg(DAY,N,TS) :- N = #count{RID: support(RID,DAY,TS)}, day(DAY), ats(TS).
13 numMax(DAY,T) :- T = #max{N: numbReg(DAY,N,_)}, day(DAY).
14 numMin(DAY,T) :- T = #min{N: numbReg(DAY,N,_), N != 0}, day(DAY).
15 numbDay(DAY,N) :- N = #count{RID: support(RID,DAY,_)}, day(DAY).
16 numMaxDay(T) :- T = #max{N: numbDay(DAY, N)}.
17 :∼ x(RID,DAY,_,_,_,"bed"), chair(ID,RID,DAY,_). [1@7,RID]
18 :∼ x(RID,DAY,_,_,_,"chair"), bed(ID,RID,DAY,_). [1@7,RID]
19 :∼ numMax(DAY,T). [T@6, DAY]
20 :∼ numMax(DAY,MAX), numMin(DAY,MIN). [MAX-MIN@5,DAY]
21 :∼ numMaxDay(N). [N@4]
```

Fig. 1. ASP encoding of the real world problem.

Encoding. The related encoding is shown in Figure 1, and is described in the following. To simplify the description, we denote as r_i the rule appearing at line i of Figure 1.

Rule r_1 assigns registrations to a day and a time slot. The assignment is made only for the first registration of the patient (i.e. the one with ordering equal to 0). Rule r_2 assigns subsequent registrations (i.e. the one with ordering greater than 0) to a time slot, whereas the day is automatically computed considering the number of waiting days before the visit. Rules r_3 and r_4 encode conditions (c4) and (c5), respectively. Then, rule r_5 assigns exactly one chair/bed to each registration. Rules from r_6 to r_{10} are used to ensure that each chair and bed is assigned to at most one patient for each time slot. Rules from r_{12} to r_{16} are needed to derive auxiliary atoms that are used later on in optimization. In particular, rules from r_{12} to r_{14} compute the maximum and minimum number of patients that are simultaneously assigned to phase 2, while rules r_{15} and r_{16} compute the number of patients for each day.

Finally, weak constraints r_{17} and r_{18} are used to minimize the number of missed preferences, whereas weak constraints from r_{19} to r_{21} enforce the distribution of registrations.

Note that the encoding can be already used for the daily CTS problem by considering only atoms of the form `reg(REGID,0,WDAY,PH4,PH3,PH2,PH1,S)` and only one day, for example, `day(1)`.

3.2 Extended CTS

In this section we show how the daily CTS problem can be extended to deal with further requirements that were not considered by the solution of the San Martino Hospital of Genova. The extensions are related to *(a)* the drugs to be dispensed, and a daily limited availability for each drug; *(b)* the priority level (high, medium, low) of the registration; and *(c)* the number of nurses working in the hospital, where each nurse can assist at most k patients per time slot.

Thus, in addition to the requirements defined in Section 2, the solution scheduling has to guarantee that *(i)* each patient is assisted by exactly one nurse (Turkcan *et al.* 2012); *(ii)* each nurse can assist from 1 to k patients for each time slot (Turkcan *et al.* 2012);

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

```
reg(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S) :- reg(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S,PR,DRUG).
{nurses(ID,RID,DAY) : nurse(ID)} = 1 :- x(RID,DAY,_,_,_,_).
nurses(ID,RID,DAY,TS) :- nurses(ID,RID,DAY), res(RID,DAY,TS).
:- #count{RID: nurses(ID,RID,DAY,TS)} > K, day(DAY), ts(TS), nurse(ID), nurseLimits(K).
:- drug(DRUG,LMT,DAY), #count{RID: x(RID,DAY,_,_,ORDER,_), reg(RID,_,ORDER,_,_,_,_,_,_,DRUG)} > LMT.
:- x(RID,DAY,36,_,PH4,_,_).
:~ reg(RID,0,_,_,_,_,_,_,1,_), x(RID,_,TS,_,_,_). [TS@3,RID]
:~ reg(RID,0,_,_,_,_,_,_,2,_), x(RID,_,TS,_,_,_). [TS@2,RID]
:~ reg(RID,0,_,_,_,_,_,_,3,_), x(RID,_,TS,_,_,_). [TS@1,RID]
```

Fig. 2. ASP encoding of the added rules to the CTS extended problem.

(iii) treatments cannot exceed the maximum quantity of drugs available for each day (Heshmat and Eltawil 2021); *(iv)* treatments cannot be scheduled at the latest available time slot, since some drugs might require a long time to be prepared (Huggins *et al.* 2014); and *(v)* registrations with the highest priorities should be scheduled before other registrations (Dodaro *et al.* 2018).

Data Model. For the extended problem we start from the atoms of the CTS problem, while changing the reg atom, and add further atoms:

- Instances of `reg(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S,P,DR)` represent the registrations as in the CTS problem with the addition of a priority (`P`), and the drug (`DR`) required.
- Instances of `nurse(ID)` represent the available nurses, with its identifier `ID`.
- Instance of `nurseLimits(K)` represent the maximum number of patients (`K`) that a nurse can assist for each time slot.
- Instances of `drug(DR,LMT,DAY)` represent the amount of available drug for a given day, with its identifier `DR`, the available quantity `LMT`, and the day `DAY`.

The output is the same of the real world CTS problem.

Encoding. The encoding consists of the rules reported in Figure 1 (rules r_1–r_{21}) and the ones reported in Figure 2 (rules r_{22}–r_{30}). Rule r_{22} is needed to guarantee the backward compatibility with the previous encoding. Rule r_{23} assigns exactly one nurse to each patient (condition (i)). Rules r_{24} and r_{25} encode condition (ii), whereas r_{26} and r_{27} encode conditions (iii) and (iv), respectively. Then, rules from r_{28} to r_{30} minimize the sum of time slots assigned to patients with priority levels equal to 1, 2, and 3, respectively.

4 Experimental results

In this section we report the results of an empirical analysis of the CTS problem. Data are real world data from the San Martino Hospital for the CTS problem, while they have been randomly generated using parameters inspired by literature and real world data for the extended problem (e.g. from the work of Dodaro *et al.* (2018) for the distribution of priorities, and from the Oncology Nurses Society for the number of nurses per patient). The generation of synthetic data for the extended problem is needed since real world data from the San Martino Hospital do not cover all the features, for example, nurses and priorities, that we considered in the extended problem.

In this way we can simulate different scenarios and use them to test our encodings. The experiments were run on a AMD Ryzen 5 2600 CPU @ 3.40GHz with 15.9 GB of physical

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

RAM. The ASP system used was CLINGO (Gebser *et al.* 2016) 5.1.0, using parameters *--restart-on-model* for faster optimization and *--parallel-mode 8* for parallel execution. This setting is the result of a preliminary analysis done on the daily real world instances where we tested also other parameters, for example, `--opt-strategy=usc` for optimization. The time limit was set to 60 seconds for the daily CTS and the extended problem, whereas for the weekly CTS was set to 1200 seconds. All material used in the experiments can be found at: http://www.star.dist.unige.it/~marco/ICLP2021/material.zip.

4.1 CTS benchmarks

For the daily CTS we considered real data distributed in 22 days (two spare days of a week, Thursday and Friday, plus 4 consecutive weeks with working days Monday to Friday), each having an average number of 126 patients (12 std) with a minimum of 105 and a maximum of 148 patients per day. Data related to the patients are taken from S. Martino Hospital and differ from each other in phases duration, phase 2 need and chair or bed request. The duration of phase 4, which determines the duration of the occupation of the chair/bed, varies a lot from patient to patient. The majority of patients have a duration between 4 and 30 time slots but there are some outlier that require more than 72 time slots, i.e., these patients will end the treatment in the next session. Phase 2, that is the phase involved in the optimization, is requested by 44% of the patients while 71% of the patients require a chair and the others require a bed.

While for the real world (daily and weekly) problems we did not need to create synthetic data, for the extended one we need to assign a priority level to each patient, impose an availability for each drug used, define the number of nurses and the number of patients that a nurse can assist in the same slot. The priorities of the registrations have been generated from an uneven distribution of three possible values (with weights respectively of 0.20, 0.40, and 0.40 for registrations having priority 1, 2, and 3, respectively). In order to limit the degrees of our analysis, we decided to focus on nurses availability, and not limiting the drugs availability instead, and defined three scenario: one with low availability of nurses, one with an average availability and one having high availability. This choice is motivated by the real world scenario in the S. Martino Hospital, in which they would desire to limit and treat nurses explicitly, while quantity of drugs is not an issue. To modify the availability we set the number of nurses to 5 and then changed the number of concurrent patients that can be assigned to the same nurse to 4, 7, and 10 for the three scenario, respectively.

For the weekly problem, we considered the 4 weeks Monday to Friday and not considered the first two spare days, assigning a different regimen for the patients requiring more than one appointment. For each week there are on average 634 registrations (26 std). Each registration is linked to a treatment of a patient, and the number of patients is lower than the total number of registrations (we remind that each patient can have more registrations). On average there are 542 patients for each week and the majority requires just one treatment, while about a 10% require 2 or more treatments. For each patient, there could be up to 5 registrations, each corresponding to a day of treatment, following different regimen.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

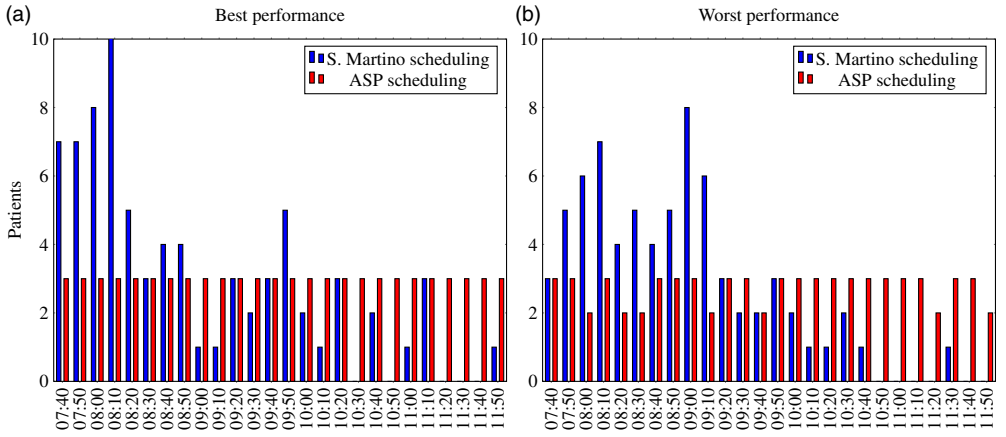


Fig. 3. Number of patients assigned to their phase 2 for each time slot in a day, comparing our scheduling with the one of the S. Martino Hospital. Figure (a) [resp. (b)] compares our best [resp. worst] result with the one of the Hospital in the same day.

4.2 Results for the real world problem

The first optimization criteria in the CTS problem is to assign the preferred resource (bed or chair) to as many patients as possible. Our solution is optimal in this respect, since it was able to accommodate *all* preferences. Moreover, we compared our results with the solution applied to the San Martino Hospital. First of all, we noted that their solution needs to resort to *virtual chairs*, meaning that some patients were not assigned to any bed nor chair. This was done to assign a time slot to all the patients, even if it implies that there might be time slots with more patients than those treatable in each time slot with the available resources. Our solution does not require such virtual chairs, that is, all the patients are assigned to an available bed or chair, and it assigns a resource (bed or chair) to 207 more patients than the one of the Hospital.

The second optimization criteria is to have a more equally distributed affluence of patients during their phase 2 for each time slot. Results are reported in Figure 3, which details the number of patients starting phase 2 in each time slot; the figure compares our best (a) and worst (b) results with those of the Hospital in the same days. From the graphs it is clear that even our worst performance produces highly balanced results. It is important to emphasize here that during the search for the optimal solution CLINGO produces several suboptimal ones, i.e., it has an *anytime behavior*. In our case, we observed that CLINGO was able to compute the optimal solution even if it could not prove its optimality within the time limit.

Concerning the weekly problem, also in this case our solution was always able to fulfill the preferences of the patients concerning bed or chairs, while for the second optimization criteria results are slightly worse than the ones for the daily problem. In particular, Figure 4 reports the number of patients for each hour and for each day starting phase 2 of one week (results of other weeks are similar). Even if in the first two days the results are worse, as expected, compared to the results on one day, they are still better than the ones implemented by S. Martino, while for days 4 and 5 they are as good as the one for the analysis on the single day. Thus, overall our solution proved to be a viable tool to schedule the all week, considering that being able to schedule the whole week in advance could lead to benefits especially from an organizational point of view.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

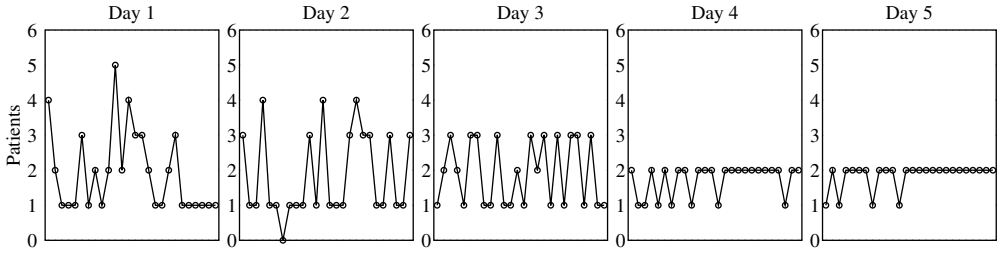


Fig. 4. Number of patients assigned to their phase 2 for each time slot and day during one working week.

Comparison to alternative logic-based formalisms. In the following, we present an empirical comparison of our ASP-based solution with alternative logic-based approaches, obtained by applying automatic translations of ASP instances. In more detail, we used the ASP solver WASP (Alviano *et al.* 2019), with the option `--pre=wbo`, which converts ground ASP instances into pseudo-Boolean instances in the wbo format (Olivier Roussel and Vasco Manquinho 2012). Then, we used the tool PYPBLIB (Anstegui *et al.* 2019) to encode wbo instances as MaxSAT instances. Moreover, in order to provide a fair comparison, we also processed our ASP instances using WASP with the option `--pre=lparse`, which collapses all weak constraints levels into one single level using exponential weights. In this way, the costs found by the different approaches can be compared.

Then, we considered three state-of-the-art MaxSAT solvers, namely MAXHS (Davies 2013), OPEN-WBO (Martins *et al.* 2014), and RC2 (Ignatiev *et al.* 2019), and the industrial tool for solving optimization problems GUROBI (Gurobi Optimization, LLC 2021), which is able to process instances in the wbo format. Concerning CLINGO, we used *(i)* its default configuration (CLINGO-DEF); *(ii)* the option `restart-on-model` (CLINGO-ROM); and `(iii)` the option `--opt-strategy=usc` (CLINGO-USC). The latter enables the usage of algorithm OLL (Morgado *et al.* 2014), which is the same algorithm employed by the MaxSAT solver RC2.

The experiment was executed on the daily instances, with a timeout of 60 seconds as in Section 4.1. Results are reported in Table 1, where for each solver and instance we report the ranking obtained by each solver. The solver is in the first position if it finds the solution with the lowest cost, a dash means that the solver outputs no solution within the time limit. As a general observation, CLINGO-ROM obtains the best performance overall, since it finds the best cost in all but two instances. The performance of CLINGO-DEF is in general slightly worse than the one of CLINGO-ROM, even if in the majority of the instances they compute the same solution. Concerning MaxSAT solvers, we observe that OPEN-WBO is the best performing solver, and it is able to produce the best solution on instances 10-15 and 10-28. MAXHS is able to print some solution within the time limit, which is however often much worse than the ones found by CLINGO-ROM, CLINGO-DEF, and OPEN-WBO. The only exception is represented by instance 10-23 where CLINGO-ROM and CLINGO-DEF find a solution with a cost equal to 116,489, and MAXHS finds a solution with a cost equal to 116,490. Concerning CLINGO-USC and RC2, we observe that both solvers are based on the same algorithm, namely OLL (Morgado *et al.* 2014), which however is able to produce only optimal solutions and none is found within the time limit. Finally, we mention that also GUROBI cannot compute a solution within the time limit.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

Table 1. *Comparison of ASP solution with alternative logic-based solutions*

Instance	CLINGO-DEF	CLINGO-ROM	CLINGO-USC	MAXHS	OPEN-WBO	RC2	GUROBI
10-01	1	1	–	4	3	–	–
10-02	1	1	–	4	3	–	–
10-05	1	2	–	4	3	–	–
10-06	1	1	–	4	3	–	–
10-07	1	1	–	4	3	–	–
10-08	1	1	–	4	3	–	–
10-09	2	1	–	4	3	–	–
10-13	1	1	–	4	3	–	–
10-14	3	1	–	4	2	–	–
10-15	2	2	–	4	1	–	–
10-16	2	1	–	4	2	–	–
10-19	1	1	–	4	3	–	–
10-20	1	1	–	4	3	–	–
10-21	1	1	–	4	3	–	–
10-22	1	1	–	4	3	–	–
10-23	1	1	–	3	4	–	–
10-26	2	1	–	4	3	–	–
10-27	1	1	–	4	3	–	–
10-28	1	1	–	4	1	–	–
10-29	1	1	–	4	3	–	–
10-30	1	1	–	4	3	–	–

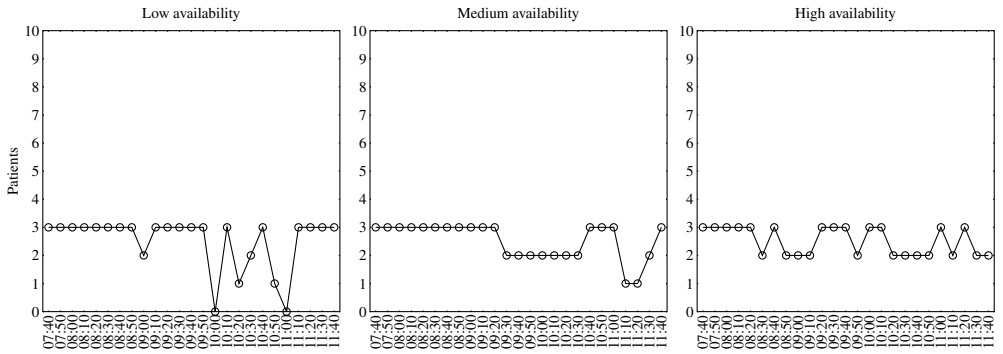


Fig. 5. Number of patients assigned to their phase 2 in the extended CTS with (left) low, (center) medium and (right) high availability of nurses.

4.3 Results for the extended problem

First, we note that also in this case our solution is able to assign the preferred resource to each patient. Regarding uniformity of patients starting phase 2, Figure 5 reports the results on our three scenario about nurses availability. Each graph is organized as Figure 3. It is possible to observe that with low availability (left) nurses are not enough to satisfy the desired property of having a (approx.) constant number of patients during the session, while with medium (center) and high (right) availability results are very positive and similar to the daily CTS problem. It is important to note that we are still able to reach an optimal result in the same amount of time with high availability of nurses even with the extended case.

The distribution of patients based on their priority in the time slots of a day for the three scenario is shown in Figure 6, in which we can see that in the low availability

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

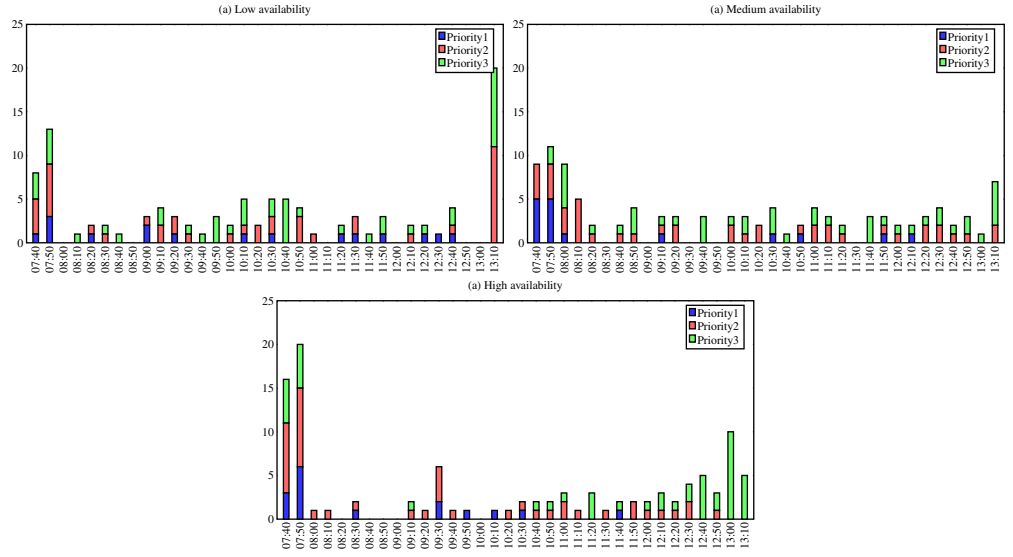


Fig. 6. Distribution of patients for the treatment and their priority in the extended problem with low (left), medium (center) and high availability (right) of nurses.

scenario the distribution does not respect the priority adequately, while with medium and high availability the results are highly satisfying, through not optimal. In the scenario with high availability of nurses, the majority of patients with high priority are scheduled in the first two available time slots, whereas in the latest time slots there are almost only patients with low priority.

It is finally interesting to note that such results, for our more involved setting, are reached despite the weak constraints related to the availability of nurses have lower level than the previous weak constraints, and the still low time limit (60 seconds).

5 Rescheduling

In this section, we show our solution to a rescheduling situation, in which the original schedule cannot be fulfilled as planned. It is important to emphasize here that rescheduling is usually independent from the quality of the schedule, and it is usually due to unpredictable events. First of all, even if every treatment has a regimen a doctor can try to personalize the treatment for a patient and try to change the regimen to have a better response. Therefore, doctors should have the possibility to modify appointments for a patient without impacting too much the original calendar. Then, a patient may be ill or unable to show up for a visit, and it is important to be able to give a new appointment as soon as possible. Having a proper solution to these situations can improve the chance of survival of patients, since several recent studies (Sud *et al.* 2020; Kumar and Dey 2020) showed that delay of treatments, due to the COVID19 emergency, has had a negative impact on the survival rates of the patients.

In our setting, the rescheduling is applied to a weekly schedule, where a number of appointments are canceled, regimen are possibly changed, and thus they need to be rescheduled. The generated output has to follow the same requirements of the weekly

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

```
1  {y(RID,DAY,TS,PH4,0,C) : ts(TS), day(DAY), not un(RID,DAY)} = 1 :- reg(RID,0,_,PH4,PH3,PH2,PH1,S), not
       regN(RID,0,_,_,_,_,_,_).
2  {y(RID,DAY,TS,PH4,0,C) : ts(TS), day(DAY), not un(RID,DAY)} = 1 :- regN(RID,0,_,PH4,PH3,PH2,PH1,S).
3  y(RID,DAY,TS,PH4,ORDER,C) :- y(RID,DAY,_,_,N,_), x(RID,DAY,_,_,N,_), x(RID,DAY,TS,PH4,ORDER,C), not
       regN(RID,ORDER,_,_,_,_,_,_), not un(RID,DAY).
4  {y(RID,DAY,TS,PH4,ORDER,C) : ts(TS), day(DAY), not un(RID,DAY)} = 1 :- y(RID,DAY1,_,_,N,_),
       reg(RID,ORDER,DAY2,PH4,PH3,PH2,PH1,S), not regN(RID,ORDER,_,_,_,_,_,_), ORDER=N+1,
       day(DAY1+DAY2).
5  {y(RID,DAY,TS,PH4,ORDER,C) : ts(TS), day(DAY), not un(RID,DAY)} = 1 :- y(RID,DAY1,_,_,N,_),
       regN(RID,ORDER,DAY2,PH4,PH3,PH2,PH1,S), ORDER=N+1, day(DAY1+DAY2).
6  y(RID,DAY,TS,PH4,ORDER,C) :- x(RID,DAY,TS,PH4,ORDER,C), un(RID,_), DAY < #min{D: un(REGID,D)}.
7  y(RID,DAY,TS,PH4,ORDER,C) :- x(RID,DAY,TS,PH4,ORDER,C), DAY < #min{D: un(_,D);D:
       regN(RID,ORDER,_,_,_,_,_,_), x(RID,D,_,_,_,ORDER,_)}.
8  :- y(RID,DAY1,_,_,ORDER,_), x(RID,DAY2,_,_,ORDER,_), not regN(RID,_,_,_,_,_,_,_), DAY1 < DAY2.
9  :~ y(REGID,DAY1,_,_,_,N,_), y(REGID,DAY2,_,_,_,_,ORDER,_), reg(RID,ORDER,DD,_,_,_,_,_), not
       regN(RID,ORDER,_,_,_,_,_,_), ORDER=N+1. [|DD - (DAY2 -DAY1|@8,RID,ORDER]
10 :~ y(REGID,DAY1,_,_,_,N,_), y(REGID,DAY2,_,_,_,_,ORDER,_), not regN(RID,ORDER,ORDER,_,_,_,_,_),
       ORDER=N+1. [|DD - (DAY2 -DAY1|@8,RID,ORDER]
11 :~ y(RID, DAY1,_,_,_,0,_), x(RID, DAY2,_,_,_,0,_). [|DAY1 - DAY2|@7,RID]
```

Fig. 7. ASP encoding of the rescheduling problem.

scheduling plus some additional requirements: *(i)* changed appointments can only be postponed, i.e., appointments cannot be moved before the original day of the appointment because patients could be not ready; *(ii)* no appointment can be changed before the first day in which a patient requires a modification to her/his registration or there is an unavailable patient; this is required to limit the changes to the calendar; *(iii)* a patient that has already completed the first appointment but has neither unavailable days nor modified registrations cannot be postponed, to ensure compliance with the regimen; and *(iv)* when the first appointment of a patient is postponed, then all subsequent appointments of the same patient should be rescheduled to try to follow the regimen.

Moreover, an optimal solution has to (a) minimize the sum of the distance between the days required by a regimen and the actual day of the appointment, (b) minimize the distance between the new day for the first appointment and the old one for the patients with a postponed appointment, while (c) maximizing the number of patients that are assigned to the preferred resource.

Data Model. The input data is the same of the weekly problem plus the following atoms:

- Instances of `x(RID,DAY,TS,DUR,ORDER,S)` represent the previous scheduling, i.e., the output of the encodings described in Section 3.
- Instances of `un(RID, DAY)` represent the day (`DAY`) in which the patient characterized by an id (`RID`) is unavailable.
- Instances of `regN(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S)` represent the new registrations due to changes of regimens decided by doctors, characterized by an id (`REGID`), the ordering (`ORD`), the number of waiting days before the visit (`WDAY`), the duration of each phase (`PH4`, ..., `PH1`), and the preference for chair or bed (`C`), where `C` can be `"bed"` or `"chair"`.

The output is similar to the encodings described in Section 3, where the schedule is represented by atoms of the form `y(RID,DAY,TS,DUR,ORDER,S)`.

ASP Encoding. The encoding for the rescheduling problem is made of the rules in Figure 7, where again to simplify the description we denote as r_i the rule appearing at the line i, plus rules from r_3 to r_{10} and weak constraints r_{17} and r_{18} from Figure 1 where atom y replaces atom x.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

Table 2. *Summary of the rescheduling results*

Scenario	Unavailable patients	Changed regimen	Unnecessary registration moved	Time
I	15	–	0	196 s
II	20	–	0	218 s
III	25	–	0	229 s
IV	15	1	0	190 s
IV	15	2	0	203 s
IV	15	3	0	205 s

Rule r_1 and r_2 assign registrations to a day and a time slot, where r_1 assigns a day and a time slot to patient without a new registration, while r_2 assigns a day and a time slot using new registrations. The assignment is made only for the first registration of the patient (i.e., the one where ordering is equal to 0). Rule r_3 ensures that assignment of the previous scheduling is preserved for patients that need no modification. Rule r_4 assigns a day and a time slot to patients without a new registration, while r_5 assigns a day and a time slot using new registrations, for the subsequent registrations following the first one. Rule r_6 ensures that the assignments before the first day of unavailability of that patient are not changed. Rule r_7 ensures that all the assignments are not changed before the first day in which a patient is unavailable or is required a new registration. Then, rule r_8 ensures that all the changed assignments are not anticipated. Finally, weak constraints r_9 and r_{10} are used to minimize the sum of the distance between the days requested by a regimen and the actual day of the appointment assigned to a patient, considering old registrations in r_9 and the new registrations in r_{10}, whereas weak constraint r_{11} minimizes the distance between the day assigned to the first assignment in the old and new schedules.

Experiments. We tested our solution to the rescheduling problem in a number of different scenarios. Scenario I, II, and III consider a sudden unavailability of 15, 20, and 25 patients, respectively, whereas Scenario IV,V, and VI considers a sudden unavailability of 15 patients and 1, 2, and 3 regimens, respectively, are modified. Results are summarized in Table 2. First of all, it is important to emphasize that all obtained solutions are optimal and that all of them were computed in less than 4 minutes. Indeed, the ASP solution is able to reschedule the patients without changing any unnecessary registration, i.e., no patient has the first appointment changed (apart from the ones with unavailability). We also mention that our solutions always assign the preferred resource (bed or chair) to all the patients. We finally report that further scenario with 20 unavailable patients and changed regimen, not shown in the table, confirm the results.

6 Related work

Sevinc *et al.* (2013) addressed the CTS problem through a two-phase approach. In the first one an adaptive negative-feedback scheduling algorithm is adopted to control the load on the system, while in the second phase two heuristics based on the Multiple Knapsack Problem have been evaluated to assign patients to specific infusion seats. The overall

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

design has been tested at a local chemotherapy center and has yielded good results for patient waiting times, orderly execution of chemotherapy regimen and utilization of infusion chairs. Huang *et al.* (2017) developed and implemented a model to optimize safety and efficiency in terms of staffing resource violations measured by nurse-to-patient ratios throughout the workday and at key points during treatment to decide when to schedule patients according to their visit duration. The optimization model was built using Excel Solver. Hahn-Goldberg *et al.* (2014) addressed in particular dynamic uncertainty that arises from requests for appointments that arrive in real time and uncertainty due to last minute scheduling changes through a proactive template of an expected day in the chemotherapy center using a deterministic optimization model updated, to accommodate last minute additions and cancelations to the schedule, by a shuffling algorithm. Huggins *et al.* (2014) presented a mixed-integer programming optimization model developed with the objective of maximizing resource utilization, while balancing human workload, in particular taking into account variability in length of treatment, increased patient demand, and resource limitations. Compared to the papers mentioned above, our solution tackles further aspects. For the scheduling problem, in addition to taking into account the starting time of treatments, we also manage the previous phases (blood collection and visit). We also minimize the number of patients who request blood collection at the same time. Considering these previous phases is important because, as mentioned by Lam *et al.* (2016), usually the CTS problem involves many departments and knowing when a resource will be used is vital for an efficient planning. While in our work we focused on rescheduling as a tool that allows to face unexpected events, the rescheduling proposed by Hahn-Goldberg *et al.* (2014) is implemented as a way to reach better results when new patients are added. Another difference is that Hahn-Goldberg *et al.* (2014) apply the rescheduling to a daily problem, while we consider a 5 days horizon and more registrations for each patient.

ASP has been already used as a tool for solving scheduling problems (Erdem *et al.* 2016; Gebser *et al.* 2018), also in the healthcare domains (Alviano *et al.* 2020). In this paper, we focus on a novel problem whose requirements, differently from previous work, were completely specified by the S. Martino Hospital, and we employed real world data for the empirical analysis.

Finally, we mention that this paper extends and revises a paper appearing in the informal CEUR Proceedings (Dodaro *et al.* 2020), with the following additions: *(i)* a mathematical formulation of the problem, *(ii)* encodings for a real world problem, *(iii)* a rescheduling solution, and *(iv)* all analysis performed on real data.

7 Conclusions and current work

In this paper, we have employed ASP for solving some variants of the CTS problem. We started from the problem addressed by the S. Martino Hospital in Genova, then considered longer planning horizons, and desired features not taken at the moment into account by the hospital. Results on real data show that our solution is able, for several of such variants, to balancing in a highly satisfiable way the number of patients during the days and/or the slots of a day, and in short time. We also presented a rescheduling solution that shows flexibility in dealing with unavailability and changed regimen, minimizing the number of changes to be done on the original schedule. Future work includes investigat-

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

ing the possibility to use extensions of ASP, such as ASP with preferences (e.g., using ASPRIN (Brewka *et al.* 2015)), since it can be effective in solving multi-objective version of CTS.

Acknowledgments

The work is carried out in the framework of a partnership with Janssen-Cilag SpA which partially sponsored the research. Authors are also grateful to the staff working at the San Martino Hospital for their support and for the data used in the paper.

Competing interests: Ivan Porro and Giuseppe Galatà have business interest in SurgiQ.

References

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

Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Khan, M. K., Maratea, M., Mochi, M., Morozan, V., Porro, I. and Schouten, M. 2020. Answer set programming in healthcare: Extended overview. In *IPS and RCRA 2020*. CEUR Workshop Proceedings, vol. 2745. CEUR-WS.org.

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

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. 2015. asprin: Customizing answer set preferences without a headache. In *AAAI 2015*. AAAI Press, 1467–1474.

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.

Davies, J. 2013. Solving maxsat by decoupling optimization and satisfaction. Ph.D. thesis, University of Toronto.

Dodaro, C., Galatà, G., Maratea, M., Mochi, M. and Porro, I. 2020. Chemotherapy treatment scheduling via answer set programming. In *CILC 2020*. CEUR Workshop Proceedings, vol. 2710. CEUR-WS.org, 342–356.

Dodaro, C., Galatà, G., Maratea, M. and Porro, I. 2018. Operating room scheduling via answer set programming. In *AI*IA*. LNCS, vol. 11298. Springer, 445–459.

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

Falkner, A. A., Friedrich, G., Schekotihin, K., Taupe, R. and Teppan, E. C. 2018. Industrial applications of answer set programming. *Knstliche Intelligenz 32,* 2–3, 165–176.

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

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 and Practice of Logic Programming 18,* 3-4, 520–534.

Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press

Hahn-Goldberg, S., Carter, M. W., Beck, J. C., Trudeau, M., Sousa, P. and Beattie, K. 2014. Dynamic optimization of chemotherapy outpatient scheduling with uncertainty. *Health Care Management Science 17,* 4, 379–392.

Heshmat, M. and Eltawil, A. 2021. Solving operational problems in outpatient chemotherapy clinics using mathematical programming and simulation. *Annals of Operations Research 298*, 1–18.

Huang, Y.-L., Bryce, A. H., Culbertson, T., Connor, S. L. and Looker, S. A. e. a. 2017. Alternative Outpatient Chemotherapy Scheduling Method to Improve Patient Service Quality and Nurse Satisfaction. *Journal of Oncology Practice 14,* 2, 82–91.

Huggins, A., Claudio, D. and Prez, E. 2014. Improving resource utilization in a cancer clinic: An optimization model. In *IIE Annual Conference and Expo 2014*.

Ignatiev, A., Morgado, A. and Marques-Silva, J. 2019. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput. 11,* 1, 53–64.

Kumar, D. and Dey, T. 2020. Treatment delays in oncology patients during COVID-19 pandemic: A perspective. *Journal of Global Health 10,* 1. International Society of Global Health.

Lam, G., Jouini, O. and Cardinal, J. 2016. Outpatient chemotherapy planning: a literature review with insights from a case study. *IIE Transactions on Healthcare Systems Engineering 6,* 3, 127–139.

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

Morgado, A., Dodaro, C. and Marques-Silva, J. 2014. Core-Guided MaxSAT with Soft Cardinality Constraints. In *CP 2014*. Springer, Lyon, France, 564–573.

Olivier Roussel and Vasco Manquinho. 2012. Input/Output Format and Solver Requirements for the Competitions of Pseudo-Boolean Solvers.

Schüller, P. 2018. Answer set programming in linguistics. *Künstliche Intelligence 32,* 2–3, 151–155.

Sevinc, S., Sanli, U. A. and Goker, E. 2013. Algorithms for scheduling of chemotherapy plans. *Computers in Biology and Medicine 43,* 12, 2103–2109.

Sud, A., Jones, M. E., Broggio, J., Loveday, C. and Torr, B. e. a. 2020. Collateral damage: the impact on outcomes from cancer surgery of the COVID-19 pandemic. *Annals of Oncology 31,* 8, 1065–1074. Elsevier.

Turkcan, A., Zeng, B. and Lawley, M. 2012. Chemotherapy operations planning and scheduling. *IIE Transactions on Healthcare Systems Engineering 2,* 1, 31–49.

https://doi.org/10.1017/S1471068421000363 Published online by Cambridge University Press