Chemotherapy Treatment Scheduling via Answer Set Programming*

Carmine Dodaro[1], Giuseppe Galatà[2], Marco Maratea[3]**, Marco Mochi[2,3], and Ivan Porro[2]

[1] DEMACS, University of Calabria, Rende, Italy,
dodaro@mat.unical.it
[2] SurgiQ srl, Italy
E-mail: {name.surname}@surgiq.com
[3] DIBRIS, University of Genova, Genova, Italy
marco.mochi@edu.unige.it,marco.maratea@unige.it

Abstract. The problem of planning and 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, and the availability of resources, e.g. treatment time, nurses, and pharmacy quantities. At the same time, realizing a satisfying schedule is of upmost importance for obtaining the best health outcomes.
In this paper we present a solution to the problem based on Answer Set Programming (ASP), that recently proved to be a consistent methodology for solving complex scheduling problems involving optimization. Results of an experimental analysis, conducted on benchmarks with realistic sizes and parameters, show that ASP is a suitable solving methodology also for this important scheduling problem.

1 Introduction

The Chemotherapy Treatment Scheduling (CTS) [29–31, 35] 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 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 for preparing a solution. A proper solution to the CTS problem is thus crucial for improving the degree of satisfaction of patients and nurses, and for a better management of resources. Various studies, also in the context of the COVID19 emergency [32, 36], have shown how delays in cancer surgery

** Corresponding author.

and treatment 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 and tools such as Answer Set Programming (ASP). As a matter of fact, ASP has been successfully employed for solving hard combinatorial problems in several research areas, including Artificial Intelligence [8, 9, 18], Bioinformatics [20], Hydroinformatics [22], and it has been also employed to solve many scheduling problems [14, 26, 34, 1, 15, 16, 5, 6, 19, 8], and in industrial applications (see, e.g., [2, 17]). The success of ASP is due to different factors, including a simple but rich syntax [12], which includes optimization statements as well as powerful database-inspired constructs like aggregates, an intuitive semantics [10], and the availability of efficient solvers (see, e.g., [4, 23, 33, 25, 3]).

In this paper, we propose the first ASP encoding for solving the CTS problem, and then we tested our solution by experimenting with several instances simulating real-world scenarios. Results obtained using the state-of-the-art ASP solver CLINGO [24] show that ASP is a suitable solving methodology for the CTS problem.

To summarize, the main contributions of this paper are the following:

- We provide an ASP encoding for solving the complete CTS problem (Section 4).
- We generated several instances simulating real-world scenario and conducted an experimental analysis assessing the good performance of our solution (Section 5).
- We analyze related literature (Section 6).

The paper is completed by Section 2, which contains needed preliminaries about ASP, by an informal description of the CTS problem in Section 3, and by conclusions and possible topics for future research in Section 7.

2 Background on ASP

Answer Set Programming (ASP) [10] is a programming paradigm developed in the field of nonmonotonic reasoning and logic programming. In this section we overview the language of ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [10, 13]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with uppercase letter and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants.

A *ground set* is a set of pairs of the form $\langle consts : conj \rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X, c), p(X); Y : b(Y, m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions implemented in ASP systems are the following:

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

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

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

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

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

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

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

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

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

An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., *not* ℓ) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false (resp., true) otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on the multiset S) with respect to I satisfies the guard; otherwise, it is false.

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

A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [21] of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) [21].

Given a program with weak constraints $\Pi = \langle P, W \rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W \rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by an interpretation I if all the literals in ω are true w.r.t. I. An *optimum answer set* for Π is an answer set of G_P that minimizes the sum of the weights of the violated weak constraints in G_W in a prioritized way.

3 Problem Description

In this section, we provide an informal description of the CTS problem and its requirements.

The CTS problem consists of computing a schedule for chemotherapy patients. Chemotherapy treatment plans have a cyclic nature, following a schema that depends on the required treatment and each different treatment session requires different drugs to be dispensed. The input of the problem is a list of registrations, corresponding to treatment sessions for the patients, where each registration includes:

- the drugs to be dispensed, with a maximum of 3 drugs per session;
- the priority level of the registration, where 1, 2, and 3 correspond to registrations with high, medium, and low priority, respectively; and
- the day before which the treatment must start, if the registration corresponds to the first session of the patient, or the number of waiting days before the subsequent session, otherwise.

Registrations range over a period of time of 14 days, where each day is composed by 8 time slots. Then, each hospital has c available chairs, each chair can be assigned to at most *ntreat* treatments, n nurses working in the hospital, k patients that a nurse can visit per time slot, and a maximum quantity of

available drugs for each day. In our setting, c, $ntreat$, n, and k are fixed and set to 15, 10, 5, and 4, respectively.

The output of the problem is a schedule of registrations to time slots according to the following requirements:

- the first session treatment must be scheduled before the date reported in the registration;
- the subsequent sessions must be scheduled exactly after the number of waiting days specified in the registration;
- each chair can be used by only one patient for each time slot;
- if the treatment requires more than one time slots, then the patients must always use the same chair;
- each nurse can assist from 1 to k patients for each time slot;
- each chair can be assigned to at most $ntreat$ treatments;
- treatments cannot exceed the maximum quantity of drugs available for each day;
- treatments must be scheduled as soon as possible, therefore it is not possible that a day has no scheduled registration and subsequent days have scheduled registrations;
- since some drugs might require a long time to be prepared, treatments cannot be scheduled at the latest available time slot.

Moreover, as a further requirement, registrations with the highest priorities should be scheduled before other registrations.

4 ASP Encoding for the CTS problem

Starting from the specifications in the previous section, here we present the ASP encoding, based on the input language of CLINGO [23], for the scheduling problem.

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

- Instances of `reg(REGID,PRIOR,M,DUEDATE,TID1,TID2,TID3)` represent the registrations, characterized by an id (`REGID`), a priority score (`PRIOR`, we recall that 1 is the highest priority and 3 is the lowest priority), a value indicating an internal order of treatments (`M`, where 0 indicates the first treatment, 1 the second treatment, etc.), the date by which the treatments must be carried out (`DUEDATE`), the ids of treatments (`TID1`, `TID2`, `TID3`) that must be carried out.
- Instances of `mss(DAY,TS)` represent the available time slots for each day, e.g., `mss(1,1)`, ..., `mss(1,8)` denote that 8 slots are available for the day 1, where each slot has a fixed duration (30 minutes or 1 hour) depending on the scenario.

```
1  {x(RID,DAY,TS,TID1,TID2,TID3,PRIOR,0) : mss(DAY,TS), DAY <= DUEDATE} = 1 :-
       reg(RID,PRIOR,0,DUEDATE,TID1,TID2,TID3).
2  {x(RID,DAY1+DAY2,TS,TID1,TID2,TID3,0,M) : mss(DAY1+DAY2,TS)} = 1 :- x(RID,DAY1,_,_,_,_,_,N),
       reg(RID,_,M,DAY2,TID1,TID2,TID3), M=N+1, mss(DAY1+DAY2,_).
3  res(RID,DAY,TS..TS+D1-1,NCHAIR,NNURSE) :- x(RID,DAY,TS,TID1,_,_,_,_),
       type(TID1,_,NCHAIR,NNURSE,D1).
4  res(RID,DAY,(TS+D1)..(TS+D1+D2-1),NCHAIR,NNURSE) :- x(RID,DAY,TS,TID1,TID2,_,_,_),
       type(TID1,_,_,_, D1), type(TID2,_,NCHAIR,NNURSE,D2).
5  res(RID,DAY,(TS+D1+D2)..(TS+D1+D2+D3-1),NCHAIR,NNURSE) :- x(RID,DAY,TS,TID1,TID2,TID3,_,_),
       type(TID1,_,_,_,D1), type(TID2,_,_,_,D2), type(TID3,_,NCHAIR,NNURSE,D3).
6  {chair(ID,RID,DAY,TS) : chair(ID)} = NCHAIR :- res(RID,DAY,TS,NCHAIR,_).
7  {nurses(ID,RID,DAY,TS) : nurse(DAY,ID)} = NNURSES :- res(RID,DAY,TS,_,NNURSES).
8  :- #count{RID : chair(ID,RID,DAY,TS)} > 1, chair(ID), mss(DAY,TS).
9  :- #count{RID : nurses(ID,RID,DAY,TS)} > k, nurse(DAY,ID), mss(DAY,TS).
10 :- chair(ID1,RID,DAY,TS), chair(ID2,RID,DAY,TS+1), ID1 < ID2.
11 :- #sum{NCHAIR,RID : res(RID,DAY,TS,NCHAIR,_)} > c, mss(DAY,TS).
12 :- T1 = #max{TS1 : res(_,DAY,TS1,_,_)}, T2 = #max{TS2: mss(DAY,TS2)}, T1 > T2.
13 :- #count{RID: chair(ID,RID,DAY,_)} > ntreat, mss(DAY,_), chair(ID).
14 :- T = #max{TS: mss(DAY,TS)}, x(_,DAY,T,_,_,_,_,_).
15 :- not x(_,DAY,_,_,_,_,_,_), x(_,DAY+1,_,_,_,_,_,_), mss(DAY,_).
16 :- #count{RID:x(RID,DAY,_,TID,_,_,_,_); RID:x(RID,DAY,_,_,TID,_,_,_);
       RID:x(RID,DAY,_,_,_,TID,_,_)} > MAX/Q, mss(DAY,_), type(TID,Q,_,_,_), Q != 0,
       drug(TID,MAX).
17 :~ x(RID,DAY,_,_,_,_,1,0). [DAY@4,RID]
18 :~ x(RID,DAY,_,_,_,_,2,0). [DAY@3,RID]
19 :~ x(RID,DAY,_,_,_,_,3,0). [DAY@2,RID]
20 :~ x(_,DAY,_,_,_,_,_,0). [DAY@1]
```

Fig. 1. ASP encoding of the CTS problem

- Instances of `type(TID, QUANT, NCHAIRS, NNURSES, D)` represent for each treatment, denoted by its identifier `TID`, the amount of drugs (`QUANT`), the number of chairs (`NCHAIRS`), the number of nurses (`NNURSES`), and the duration expressed (`D`) required by the treatment.
- Instances of `drug(TID,MAX)` represent for each treatment, denoted by its identifier `TID`, the maximum availability of the required drug for each day (`MAX`).
- Instances of `chair(ID)` represent the available chairs, with its identifier `ID`.
- Instances of `nurse(ID,D)` represent the nurses available in a specific day, where `ID` is the identifier of the nurse and `D` is the day.

Moreover, we also take advantage of three constants, namely `c`, `k`, and `ntreat`, corresponding to the ones described in the previous section.

The output is an assignment represented by atoms of the form

`x(RID,DAY,TS,TID1,TID2,TID3,PRIOR,M)`

where the intuitive meaning is that the registration with id `RID` is assigned to the day `DAY` and its starting time slot is `TS`, whereas the terms `TID1`, `TID2`, `TID3`, `PRIOR`, and `M` are the ones of `reg(REGID,PRIOR,M,DUEDATE,TID1,TID2,TID3)`, described above.

Table 1. Drugs availability for each groups of patients in each day in scenario α.

Number of patients	Drug A	Drug B	Drug C
60	400	2000	600
80	500	2500	800
100	700	3200	1150

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 the line i of Figure 1.

Rules r_1 and r_2 guess an assignment for the registrations to a day `DAY` and a time slot `TS`, where r_1 is used to guess the first day of the treatment and r_2 the subsequent days. Rules r_3, r_4, and r_5 are auxiliary rules which are used for deriving atoms of the form `res(RID,DAY,H,NCHAIR,NNURSE)` starting from the assignment derived in rules r_1 and r_2. Basically, those atoms include, for each registration of a given day, all the time slots where the registration is assigned, and the number of chairs and nurses required by the registration. Then, rules r_6 and r_7 guess the chairs and the nurses that must be assigned to each selected registration. Subsequent rules, from r_8 to r_{16}, are used to check that the schedule fulfills all the requirements. In particular, rules r_8 and r_9 ensure that each chair is assigned to at most one patient and each nurse can visit at most k patients for each time slot, respectively. Rule r_{10} enforces that a patient has always the same chair until the treatment is not finished. Rule r_{11} is used to guarantee that the number of assigned chairs does not exceed the number of available chairs, denoted with the constant c. Rules r_{12} and r_{13} ensure that each treatment does not exceed the allotted time expressed by instances of `mss` and the number of treatments assigned to a chair does not exceed the maximum number of treatments (denoted with the constant *ntreat*), respectively. Rule r_{14} guarantees that treatments start before the last available slot of `mss`. Rule r_{15} ensures that if a day has at least one scheduled registration, then all previous days must also have at least one scheduled registration, whereas rule r_{16} is used to enforce that the maximum availability of the drugs is not exceeded for each day. Then, weak constraints from r_{17} to r_{19} are used to optimize the schedule of the registrations according to their priority. In particular, registrations with the highest priority must be scheduled before other registrations. Note that this optimization is considered for the first day of treatment only. Finally, weak constraint r_{20} is used to schedule the treatments as soon as possible. Note that r_{20} is somehow subsumed by weak constraints from r_{17} to r_{19}, however we find out that adding this weak constraints slightly improves the overall performance in our experiments.

5 Experimental Results

In this section we report the results of an empirical analysis of the CTS problem. Data have been randomly generated using parameters inspired by real-world data. In this way we can simulate different scenarios and use them to test our

Table 2. Drugs availability for each groups of patients in each day in scenario β.

Number of patients	Drug A	Drug B	Drug C
60	350	1850	550
80	400	2250	700
100	550	3000	900

Table 3. Schema of the treatment followed by every patient, inspired by a sample chemotherapy regimen [37]

Day	Drugs	Dose	Duration
1	A-B-C	20 mg/m^2, 100 mg/m^2, 30 units	1, 2, 1 time slot
2-5	A-B	20 mg/m^2, 100 mg/m^2	1, 2 time slot
6-7	Rest		
8	C	30 units	1 time slot

encoding. The experiments were run on a AMD Ryzen 5 2600 CPU @ 3.40GHz with 7.6 GB of physical RAM. The ASP system used was CLINGO [23], using arguments *–restart-on-model* for a faster optimization and *–parallel-mode 12* for parallel execution. The time limit was set to 300 seconds.

5.1 CTS benchmarks

The generated benchmarks vary for the number of patients and drug availability but they all consider a 14-days calendar. Two different scenarios were considered. The first one (scenario α) is characterized by an amount of drugs that allow the system to use the available chairs in a high percentage. For the second one (scenario β), we severely reduced the number of available drugs, to test the encoding in a situation in which the drugs become a limitation for the system and then the usage of the chairs is reduced. Each scenario was tested with 10 different randomly generated inputs for each of the different groups of patients: 60, 80, and 100. The characteristics of the tests are the following:

- 2 different benchmarks, comprising a planning period of 14 working days, and different numbers of available drugs, as reported in Table 1 and in Table 2, for each group of patients;
- 3 different types of drugs that are assigned to the patients following the schema reported in Table 3;
- For each patient, there are 6 different registrations, each corresponding to a day of treatment, following the schema reported in Table 3, with the first one having a randomly generated priority and a due date of the treatment with a value inside a range of days based on the priority. In this way, we simulate the common situation where a manager takes a list of patients with different priorities and tries to schedule every patient as soon as possible, taking into account the priority.

The priorities of the first registration have been generated from uneven distribution of three possible values (with weights respectively of 0.20, 0.40, and 0.40 for

Table 4. Parameters for the random generation of the scheduler input.

Patients	Scenario	Num. of Priority 1 mean (std)	Num. of Priority 2 mean (std)	Num. of Priority 3 mean (std)
60	α	12.9 (2.5)	24 (4.82)	23 (4.65)
80	α	17.3 (1.48)	32.6 (3.41)	30.1 (3.69)
100	α	19.5 (3.26)	38 (3.68)	42.5 (4.67)
60	β	11.9 (2.80)	24.4 (4.60)	23.7 (4.60)
80	β	16.8 (1.66)	32.6 (3.23)	30.6 (3.69)
100	β	19.5 (3.26)	38 (3.68)	42.5 (4.67)

Table 5. Average chair occupation (in % over the total available) for the scenario α.

	Day													
Patients	0	1	2	3	4	5	6	7	8	9	10	11	12	13
60	66	64	62	61	61	57	38	59	57	57	41	36	49	29
80	83	82	81	80	80	77	52	75	68	67	46	43	68	50
100	92	94	93	95	97	86	67	89	83	89	71	76	90	72

registrations having priority 1, 2, and 3, respectively). Depending on the priority the due date of the treatment is randomly assigned from three different ranges: [1,6) for priority 1, [6,11) for priority 2, and [11,15) for priority 3, respectively.

The parameters of the test have been summed up in Table 4. In particular, for each group of patients (60, 80 and, 100), we reported the mean and the standard deviation of the number of patients with priority 1, 2 and 3, respectively.

5.2 Results

The encoding was tested on each scenario (α, i.e. drugs abundance, and β, i.e. drugs scarcity) and with each number of patients (60, 80 or 100). We summarized our results in Tables 5 and 7 for scenario α and Tables 6 and 8 for scenario β, respectively. In each of these tables we report the average for each day, calculated over 10 tests with randomly generated input, of the infusion chairs occupation and the usage of each treatment drug as a percentage over the maximum quantity that could be produced in that day. As a general observation, these results show that our solution is capable to reach a good level of chairs occupation and drugs usage, especially in the first half of the planning period. In the second half, the efficiency decreases for the simple reason that many patients have either finished their treatments or are in their later stages, which are less time and drug consuming (see Table 3). In a real-world application, a new schedule with new patients would actually be planned such that the second half of the first schedule would overlap with first half of the second schedule, thus having some slots pre-occupied and filling all spaces left empty.

Finally, we present some more detailed results achieved on one instance of scenario α. In particular, we present in Fig. 2 the occupation of a chair during the planning period, while in Fig. 3 we show the drug usage. Fig. 4 reports

Table 6. Average chair occupation (in % over the total available) for the scenario β.

	Day													
Patients	0	1	2	3	4	5	6	7	8	9	10	11	12	13
60	56	55	55	56	56	53	39	53	48	50	45	41	51	46
80	67	65	70	70	71	63	54	68	68	64	61	63	69	68
100	90	91	92	92	93	87	64	89	80	81	73	75	89	78

Table 7. Average drug usage (in % over the available quantity) for the scenario α.

		Day												
Patients	Drug	0	1	2	3	4	5	6	7	8	9	10	11	12	13
60	A	99	97	92	92	91	84	61	93	91	91	63	56	74	50
	B	99	95	93	91	91	85	56	87	83	84	60	53	77	41
	C	99	96	93	94	94	88	57	86	84	86	64	54	67	45
80	A	100	98	96	95	94	91	66	90	83	81	54	50	83	62
	B	100	97	95	94	94	92	61	92	78	80	58	54	80	60
	C	93	95	97	96	96	90	59	81	80	75	50	45	77	56
100	A	83	83	85	87	89	77	57	75	71	77	64	68	77	59
	B	86	86	83	84	86	78	66	86	80	86	67	70	85	67
	C	72	75	77	79	81	69	48	66	61	66	53	57	70	58

the day the first session of each treatment, subdivided by patient priority, was scheduled: as we can see priority 1 patients begin their treatment at the first day available, then priority 2 are obviously favoured over priority 3 patients. In Fig. 5 we show the aggregated number of patients treated per day: note that this number can significantly vary because the duration of the sessions can be very different depending on the phase of the treatment.

6 Related Work

In this section we review related literature devoted to acknowledging some of the most interesting works published in the latest years which dealt with the CTS problem.

Sevinc et al. [35] 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 design has been put to test 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. [30] 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 durations. The optimization model was built using Excel Solver. Hahn-Goldberg et al. [29] addressed in partic-

Table 8. Average drug usage (in % over the available quantity) for the scenario β.

		Day													
Patients	Drug	0	1	2	3	4	5	6	7	8	9	10	11	12	13
60	A	96	90	90	92	93	90	70	89	87	94	83	76	90	81
	B	91	90	90	91	91	86	66	85	76	78	71	67	82	76
	C	92	93	94	93	93	85	63	91	79	84	75	64	84	70
80	A	100	96	97	98	100	91	84	97	97	97	95	94	99	98
	B	89	84	95	94	95	86	70	92	92	84	81	84	91	92
	C	86	88	95	94	95	80	72	91	91	87	76	81	95	87
100	A	98	95	94	96	98	96	75	95	91	92	82	85	92	85
	B	90	92	94	93	94	87	64	90	81	82	71	73	91	78
	C	90	94	93	94	95	85	60	88	76	77	73	74	90	80

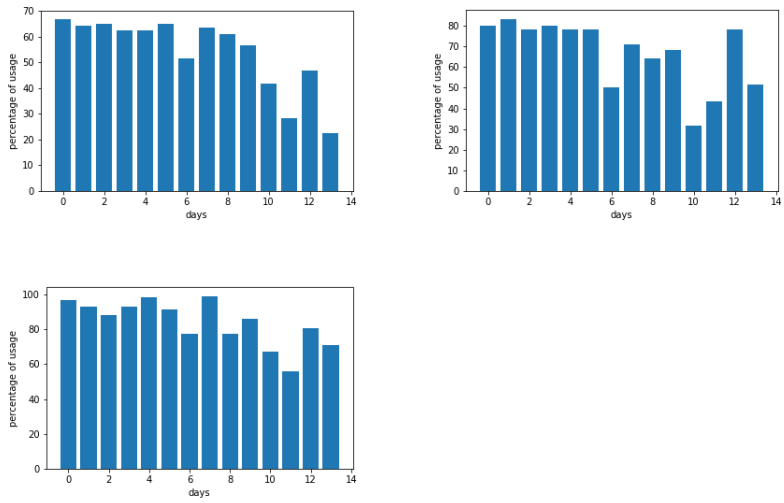

Fig. 2. Chair occupation for each day of the planning period (in % over the total available time) for scenario α schedules with 60 (top left), 80 (top right) or 100 patients (bottom left).

ular 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 centre using a deterministic optimization model updated, to accommodate last minute additions and cancellations to the schedule, by a shuffling algorithm. Huggins et al. [31] 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.

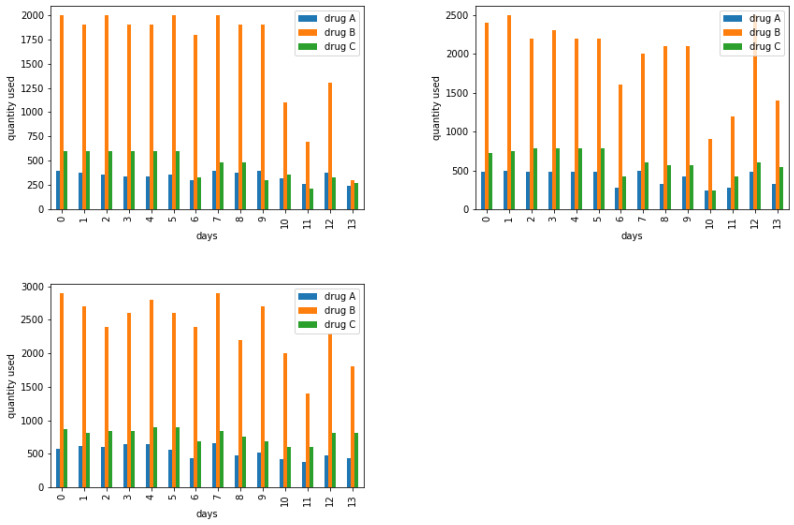

Fig. 3. Drugs usage for scenario α schedules with 60 (top left), 80 (top right) and 100 (bottom left) patients.

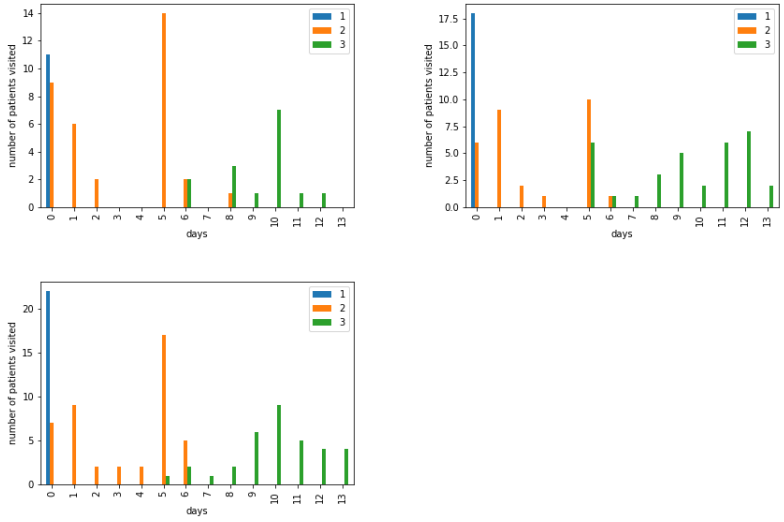

Fig. 4. Distribution of the first session of each treatment for scenario α with 60 (top left), 80 (top right) and 100 patients (bottom left). The blue bar indicates the priority 1 patients while the orange and green ones the priority 2 and 3 patients, respectively.

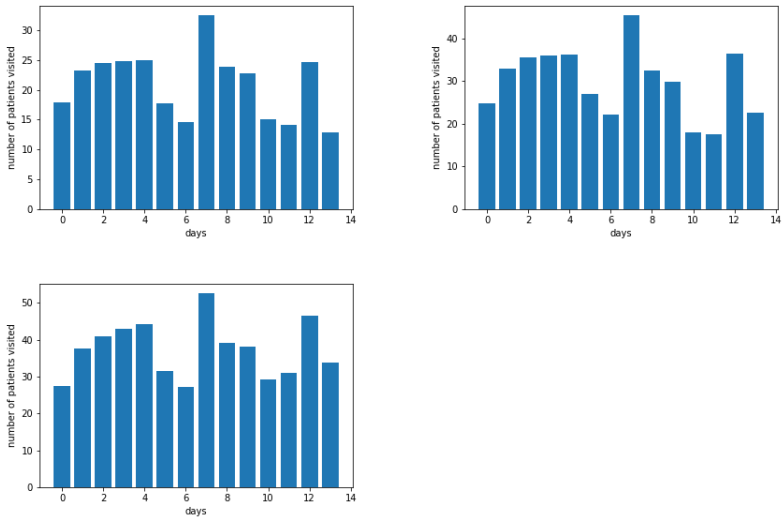

Fig. 5. Total number of patients treated in each day of the planning period for scenario α with 60 (top left), 80 (top right) and 100 (bottom left) patients (bottom left).

7 Conclusions and Current Work

In this paper we have employed ASP for solving the CTS problem, and we then presented the results of an experimental analysis on instances generated in order to simulate real-world scenarios. The proposed solution and the good results confirm that ASP is a viable AI tool for solving hard scheduling problems, mainly due to the available modeling rules and constructs, and availability of efficient solvers.

Concerning future work, we are currently improving the analysis by investigating with other parameters, e.g. with $k = 7$, given that a range between 4 and 7 for k is often employed in papers, or with 20 and 40 patients. We also plan to to include the design, encoding and analysis of a re-scheduling solution, in case the off-line solution, as proposed in this paper, cannot be fully implemented for circumstances such as canceled registrations, and the evaluation of heuristics and optimization techniques (see, e.g., [7, 27, 28]) for further improving the effectiveness of our solution. Finally, we plan to experimentally confront to the alternative solutions mentioned in the related work section, assuming such solutions are publicly available and the comparison is significant, and to analyse with real data when they become available.

References

1. Abels, D., Jordi, J., Ostrowski, M., Schaub, T., Toletti, A., Wanko, P.: Train scheduling with hybrid ASP. In: LPNMR. Lecture Notes in Computer Science, vol. 11481, pp. 3–17. Springer (2019)
2. Abseher, M., Gebser, M., Musliu, N., Schaub, T., Woltran, S.: Shift design with answer set programming. Fundamenta Informaticae **147**(1), 1–25 (2016)
3. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) LPNMR. LNCS, vol. 11481, pp. 241–255. Springer (2019)
4. Alviano, M., Dodaro, C.: Anytime answer set optimization via unsatisfiable core shrinking. Theory and Practice of Logic Programming **16**(5-6), 533–551 (2016)
5. Alviano, M., Dodaro, C., Maratea, M.: An advanced answer set programming encoding for nurse scheduling. In: AI*IA. LNCS, vol. 10640, pp. 468–482. Springer (2017)
6. Alviano, M., Dodaro, C., Maratea, M.: Nurse (re)scheduling via answer set programming. Intelligenza Artificiale **12**(2), 109–124 (2018)
7. Alviano, M., Dodaro, C., Marques-Silva, J., Ricca, F.: Optimum stable model search: Algorithms and implementation. J. Log. Comput. (2020). https://doi.org/10.1093/logcom/exv061, `http://dx.doi.org/10.1093/logcom/exv061`
8. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
9. Balduccini, M., Gelfond, M., Watson, R., Nogueira, M.: The USA-Advisor: A case study in answer set planning. In: LPNMR. LNCS, vol. 2173, pp. 439–442. Springer (2001)
10. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
11. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
12. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: Asp-core-2 input language format. Theory and Practice of Logic Programming **20**(2), 294–309 (2020)
13. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2 Input Language Format (2013), `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.01c.pdf`
14. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artificial Intelligence **231**, 151–181 (2016)
15. Dodaro, C., Galatà, G., Khan, M.K., Maratea, M., Porro, I.: An ASP-based solution for operating room scheduling with beds management. In: RuleML+RR. Lecture Notes in Computer Science, vol. 11784, pp. 67–81. Springer (2019)
16. Dodaro, C., Galatà, G., Maratea, M., Porro, I.: Operating room scheduling via answer set programming. In: AI*IA. LNCS, vol. 11298, pp. 445–459. Springer (2018)
17. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Schekotihin, K.: Combining answer set programming and domain heuristics for solving hard industrial problems (application paper). Theory and Practice of Logic Programming **16**(5-6), 653–669 (2016)
18. Dodaro, C., Leone, N., Nardi, B., Ricca, F.: Allotment problem in travel industry: A solution based on ASP. In: RR. LNCS, vol. 9209, pp. 77–92. Springer (2015)

19. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR. LNCS, vol. 10377, pp. 301–307. Springer (2017)
20. Erdem, E., Öztok, U.: Generating explanations for biomedical queries. Theory and Practice of Logic Programming **15**(1), 35–78 (2015)
21. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
22. Gavanelli, M., Nonato, M., Peano, A.: An ASP approach for the valves positioning optimization in a water distribution system. Journal of Logic and Computation **25**(6), 1351–1369 (2015)
23. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: ICLP (Technical Communications). OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
24. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. **187**, 52–89 (2012)
25. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: Lang, J. (ed.) IJCAI. pp. 5450–5456. ijcai.org (2018)
26. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research **60**, 41–95 (2017)
27. Giunchiglia, E., Maratea, M., Tacchella, A.: Dependent and independent variables in propositional satisfiability. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA. Lecture Notes in Computer Science, vol. 2424, pp. 296–307. Springer (2002)
28. Giunchiglia, E., Maratea, M., Tacchella, A.: (In)Effectiveness of look-ahead techniques in a modern SAT solver. In: Rossi, F. (ed.) CP. Lecture Notes in Computer Science, vol. 2833, pp. 842–846. Springer (2003)
29. Hahn-Goldberg, S., Carter, M.W., Beck, J.C., Trudeau, M., Sousa, P., Beattie, K.: Dynamic optimization of chemotherapy outpatient scheduling with uncertainty. Health Care Manag Sci **17**(4), 379–392 (Dec 2014)
30. Huang, Y.L., Bryce, A.H., Culbertson, T., Connor, S.L., Looker, S.A.e.a.: Alternative Outpatient Chemotherapy Scheduling Method to Improve Patient Service Quality and Nurse Satisfaction. Journal of Oncology Practice (Dec 2017)
31. Huggins, A., Claudio, D., Pérez, E.: Improving resource utilization in a cancer clinic: An optimization model. In: IIE Annual Conference and Expo 2014 (01 2014)
32. Kumar, D., Dey, T.: Treatment delays in oncology patients during COVID-19 pandemic: A perspective. J Glob Health **10**(1), 010367–010367 (Jun 2020), publisher: International Society of Global Health
33. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. Theory and Practice of Logic Programming **14**(6), 841–868 (2014)
34. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the Gioia-Tauro seaport. Theory and Practice of Logic Programming **12**(3), 361–381 (2012)
35. Sevinc, S., Sanli, U.A., Goker, E.: Algorithms for scheduling of chemotherapy plans. Computers in Biology and Medicine **43**(12), 2103–2109 (Dec 2013)
36. Sud, A., Jones, M.E., Broggio, J., Loveday, C., Torr, B.e.a.: Collateral damage: the impact on outcomes from cancer surgery of the COVID-19 pandemic. Annals of Oncology **31**(8), 1065–1074 (Aug 2020), publisher: Elsevier
37. Turkcan, A., Zeng, B., Lawley, M.: Chemotherapy operations planning and scheduling. IIE Thansactions on Healthcare Systems Engineering **2** (03 2010). https://doi.org/10.1080/19488300.2012.665155