An LBBD approach for Solving the Chemotherapy Treatment Scheduling Problem

Simone Caruso[1], Carmine Dodaro[2], Marco Maratea[2,*], Cinzia Marte[2] and Marco Mochi[3]

[1]DIBRIS, University of Genova, Italy
[2]DeMaCS, University of Calabria, Rende, Italy
[3]SurgiQ srl, Genova, Italy

Abstract

The Chemotherapy Treatments Scheduling (CTS) problem 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, e.g., treatment time, nurses, and drugs. Simultaneously, achieving an effective schedule is crucial for ensuring optimal health outcomes. A solution to this problem based on a direct encoding to Answer Set Programming (ASP) has been presented. In this paper, we propose an alternative solution that employs a Logic-based Bender Decomposition (LBBD) approach implemented through the usage of multi-shot solving. Results on real data show advantages for the LBBD approach. [1]

Keywords

Answer Set Programming, Logic Programming, Digital Health

1. Introduction

The Chemotherapy Treatment Scheduling (CTS) [1, 2, 3, 4, 5] 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 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, i.e., patients and nurses, and for a better management of the resources. Various studies 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., [6, 7, 8, 9]. A solution to this problem based on a direct ASP encoding has been already presented in [5], considering specification and data from the San Martino Hospital in Genova, Italy.

In this paper we present a second, alternative solution which, instead, employs a Logic-based Bender Decomposition (LBBD) approach taken from the field of Operation Research and already successfully

[1]This paper should not be considered as original since it contains part of the material of a more extended framework which is currently under evaluation at the 41st International Conference on Logic Programming (ICLP 2025).

CILC'25: 40th Italian Conference on Computational Logic, June 25-27 Alghero, Italy

*Corresponding author.
†These authors contributed equally.
simone.caruso@edu.unige.it (S. Caruso); carmine.dodaro@unical.it (C. Dodaro); marco.maratea@unical.it (M. Maratea); cinzia.marte@unical.it (C. Marte); marco.mochi@edu.unige.it (M. Mochi)
0000-0002-5617-5286 (C. Dodaro); 0000-0002-9034-25276 (M. Maratea); 0000-0003-3920-8186 (C. Marte); 0000-0002-5849-3667 (M. Mochi)

employed in ASP-based scheduling solutions in Healthcare [10]. It consists of defining a master problem and (one or several) subproblem(s) that interact with it by passing constraints that "cut" the search space and help converging to an (optimal) solution. The LBBD approach is implemented through the usage of the multi-shot solving variant of CLINGO [11]. Finally, we conducted an empirical evaluation on real data and specification from the San Martino Hospital in Italy showing that the LBBD approach brings computational advantages over the direct approach.

The paper is structured as follows: Section 2 provides the necessary background on the key concepts; Section 3 the CTS problem in detail; Section 4 reviews the direct ASP encoding of the problem; Section 5 details our proposed LBBD-based encoding; Section 6 discusses the experimental results and performance evaluation; finally, Sections 7 and 8 conclude the paper by discussing LBBD-related works, and by drawing conclusion and possible directions for future research.

2. Background

In this section, we introduce the necessary preliminaries on ASP and the LBBD-based approach, organized into two separate subsections.

2.1. Background on ASP

Answer Set Programming (ASP) [12] is a programming paradigm developed in the field of non-monotonic reasoning and logic programming. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [12, 13]. Hereafter, we assume the reader is familiar with logic programming conventions.

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

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 operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r has the following form:

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

where $a_1, \dots, a_n$ are standard atoms, $b_1, \dots, b_k$ are atoms, $b_{k+1}, \dots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation $not\ \ a$. The disjunction $a_1 \mid \dots \mid a_n$ is the *head* of r, while the conjunction $b_1, \dots, b_k, not\ b_{k+1}, \dots, not\ b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in the 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* [14] ω is of the form:

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

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read as "weight w at level l"). 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 the rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [15] 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) [15].

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 by levels.

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

2.2. Background on LBBD

The LBBD approach [16] is an optimization technique that extends the classical Benders Decomposition (BD) approach [17, 18, 19], offering greater flexibility and applicability to a wider range of complex problems. The central aspect of the approach lies in decomposing the problem into a *master problem* (MP) and one or more *subproblems* (SPs). The MP typically focuses on high-level decisions, whereas the SPs deal with more detailed and often logically complex constraints. The significant difference between classical BD and LBBD lies in the nature of the SPs, in the technique used to generate Benders cuts, and the properties of the resulting solution [20]. The main steps of the LBBD approach are presented in Algorithm 1. Consider a problem aimed at maximizing a function $f(x, y)$, subject to a set of constraints $C(x, y)$, $C_x(x)$, and $C_y(y)$, where x and y are vectors of variables that belong to the respective domains D_x and D_y. The LBBD approach begins by decomposing the problem P into a MP P_M and a set of SPs $\mathcal{P}_S = \{P_S^1, \dots, P_S^n\}$ (line 1). The algorithm proceeds by solving P_M to obtain an optimal solution x^* (line 3), which maximizes $f(x, y)$ for a fixed $y \in D_y$, subject to the constraints $C_x(x)$. If P_M is unsatisfiable (line 4), the procedure terminates (line 5); otherwise, the solution x^* is passed to each SP in $\mathcal{P}_S$ (line 7) in order to solve it. Each $\mathcal{P}_S$ can lead to one of the two possible outcomes: (*i*) admits an optimal solution y^* (line 8), in which case the solution is stored in a set, initially empty (line 6), that accumulates all the optimal solutions from the SPs (line 12); or (*ii*) is unsatisfiable (line 9), meaning that the solution x^* cannot be extended to the SP P_S. In this case, a new constraint C^* on x^* is derived from P_S and added to P_M (line 10). This constraint serves to refine P_M based on the last solution found, which led to the

Algorithm 1: Algorithm to solve a problem P with the LBBD methodology.

Input: A problem P, a set of constraints $C(x, y)$, $C_x(x)$, and $C_y(y)$
Output: A solution to P

1 split P in P_M and $\mathcal{P}_S = \{P_S^1, \dots, P_S^n\}$;
2 **while** *True* **do**
3 $x^* = solve(P_M, C_x(x))$;
4 **if** P_M *is* UNSAT **then**
5 **return** UNSAT
6 $Y = \emptyset$;
7 **for** each $P_S^i \in \mathcal{P}_S$ **do**
8 $y^* = solve(P_S^i(x^*), C(x^*, y), C_y(y))$; // $P_S^i(x^*)$ denotes that P_S^i keeps the input values x^*
9 **if** P_S^i *is* UNSAT **then**
10 $C_x(x) = C_x(x) \cup C^*$;
11 **else**
12 $Y = Y \cup \{y^*\}$;
13 **if** *all* P_S^i *is* SAT **then**
14 **return** (x^*, Y)

unsatisfiable problem. The algorithm then iterates (line 2), incorporating the added constraint C^*. This iterative process continues until either P_M is proven unsatisfiable (which would imply that the entire problem is unsatisfiable) or the optimal solution is found, confirming that all subproblems are satisfiable (lines 13 and 14).

3. Problem Description

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

```
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]
```

Figure 1: ASP direct encoding

concurrent patients in phase 2 to have a more uniform usage of resources during the day.

4. Direct ASP Encoding

In this section, we review the direct ASP encoding in [5], based on the input language of CLINGO [21].

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`.

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 ensures that if phase 4 is particularly long, then it must start after 11:25 A.M. (i.e., the 25th time slot). Rule r_4 ensures that phase 4 cannot

start before other phases are concluded. 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.

5. LBBD Encoding

In this section, we present our ASP-based solution to the CTS problem, exploiting the LBBD approach and solving it via a single SP. We begin by introducing the encoding of the MP P_M, which deals with the assignment of a day and a (preferred) resource to each registration (either a bed or a chair). Next, we describe the encoding of the SP P_S, which determines the specific time slots within the assigned day and ensures a balanced distribution of patients requiring phase p_2 across the working week.

5.1. Master problem

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

- `registration(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S)` which represents a patient registration 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 the resource (`S`) which can assume the value "bed" or "chair".
- `day(DAY)` which represents the available days.
- `chair(ID)` and `bed(ID)` which, respectively, represent the available chairs and beds, with their identifier `ID`.

The output of P_M consists of a scheduling containing atoms of the form:

- `x(REGID,DAY,PH4,ORD,S)`, denoting the registration of a patient `REGID` linked to a day `DAY`, with a duration of the fourth phase `PH4` in terms of time slots, the ordering value `ORD` and the patient's preference for chair or bed `S`;
- `bed(ID,REGID,DAY)` and `chair(ID,REGID,DAY)`, representing the association of a patient `REGID` to a bed or a chair with identifier `ID` in a day `DAY`, respectively.

Encoding. Figure 2 reports the encoding for the master problem of the Chemotherapy Treatment Scheduling problem. For simplicity of the description, we denote as r_i the rule appearing at line i of the figure. Rule r_1 assigns a day to patients with ordering equal to 0, i.e. those for whom we are scheduling their first appointment, while rule r_2 assigns a resource, a chair or a bed, to patients. Rules r_3 and r_4 are a relaxed version of two subproblem constraints added to speed up the convergence of the problem to the optimal solution. Specifically, they prevent assigning treatments on the same day with a total duration that exceeds the available time for each chair and bed, respectively. The first two levels of optimization are introduced by the weak constraints r_5 and r_6, which minimize the missed preferences. Rule r_7 defines a support atom that is used to compute the total number of patients scheduled in a day, whereas rule r_8 computes the maximum number of patients in a day which is minimized by the weak constraint r_{11}. Rule r_{11} is optimized by the bound defined in the rules r_9 and r_{10}, the bound is the optimal distribution of the patients equally on all the days computed as the number of patients divided by the number of days.

```
1  {x(REGID,DAY,PH4,0,S): day(DAY)} = 1 :- registration(REGID,0,_,PH4,PH3,PH2,PH1,S).
2  1 {bed(ID,REGID,DAY): bed(ID); chair(ID,REGID,DAY): chair(ID) } 1 :- x(REGID,DAY,_,_,_).
3  :- #sum{PH4, REGID: x(REGID,DAY,PH4,_,_), chair(ID,REGID,DAY)} > chair_total_time,
       day(DAY), chair(ID).
4  :- #sum{PH4, REGID: x(REGID,DAY,PH4,_,_), bed(ID,REGID,DAY)} > bed_total_time, day(DAY),
       bed(ID).
5  :~ x(REGID,DAY,_,_,1), chair(_,REGID,DAY). [1@4,REGID]
6  :~ x(REGID,DAY,_,_,0), bed(_,REGID,DAY). [1@4,REGID]
7  numbReg(DAY,TOT) :- TOT = #count{REGID: x(REGID,DAY,_,_,_)}, day(DAY).
8  numMax(DAY,MAX) :- MAX = #max{N: numbReg(DAY,N)}, day(DAY).
9  bound(N_PAT/N_DAYS) :- N_PAT = #count{REGID: reg(REGID,_,_,_,_,_,_,_)}, N_DAYS =
       #count{DAY: day(DAY)}, N_PAT \ N_DAYS = 0.
10 bound(N_PAT/N_DAYS+1) :- N_PAT = #count{REGID: reg(REGID,_,_,_,_,_,_,_)}, N_DAYS =
       #count{DAY: day(DAY)}, N_PAT \ N_DAYS != 0.
11 :~ numMax(DAY,M), bound(B), M-B > 0. [M-B@3,DAY]
```

Figure 2: CTS master problem

5.2. Subproblem

According to the instruction on line 8 of Algorithm 1, the solution of the MP P_M is given as input to the SP P_S, Additionally, P_S takes in input the following atoms:

- `registration(REGID,ORD,WDAY,PH4,PH3,PH2,PH1,S)`, representing a registration `REGID`, the ordering value `ORD`, the number of waiting days before the visit `WDAY`, the duration of each phase `PH4`, ..., `PH1`, and the preference for resource `S`;
- `day(DAY)`, denoting the available days `DAY`;
- `ts(TS)`, indicating the available time slots for phase 4;
- `ats(TS)`, representing all available time slots;
- `chair(ID)` and `bed(ID)`, denoting the available chairs and beds, with their identifier `ID`, respectively.

Based on the solution of P_M, P_S generates the atom `y(REGID, DAY, TS, PH4, ORD, S)`, representing the registration of a patient `REGID`, scheduled for a specific day `DAY` and time slot `TS`. It also indicates the duration of the fourth phase `PH4` in terms of time slots, the ordering value `ORD`, and the patient's preference for either a chair or a bed, denoted by `S`.

The full encoding of P_S is presented in Figure 3. For simplicity, we denote the rule on line i of the figure as r_i. Rule r_1 assigns a time slot to patients with an order equal to 0, i.e. those attending their first appointment, while rule r_2 assigns a time slot to appointments with an order greater than 0. Rules r_3 and r_4 ensure that if phase f_4 is particularly long, then it must start after 11:25 A.M. (i.e. the 25th time slot) and f_4 cannot start before other phases are concluded, respectively. Rules from r_5 to r_9 ensure that at most one patient is assigned to each resource at each time slot. Finally, from rule r_{10} to r_{13} it is computed the maximum and minimum number of patients that are accessing phase f_2 simultaneously and they are used in the weak constraints r_{14} and r_{15} that enforce the distribution of registrations.

If P_S is unsatisfiable, a no-good cut is generated and added to the MP, as specified in lines 9 and 10 of Algorithm 1. The cut has the following form:

```
nogood_bed(NGID) :- bed(_,REGID,DAY) : unfeasable_bed(REGID,DAY,NGID); nogood(NGID).
nogood_chair(NGID) :- chair(_,REGID,DAY) : unfeasable_chair(REGID,DAY,NGID); nogood(NGID).
:- nogood_bed(NGID), nogood_chair(NGID).
```

This cut excludes the assignment determined by P_M. Specifically, after P_S returns unsat, a unique `ID` (`NGID`) is generated to represent the current iteration. For each patient, an `unfeasible_chair(REGID,DAY,NGID)` and an `unfeasible_bed(REGID,DAY,NGID)` atoms are created.

```
{y(REGID,DAY,TS,PH4,0,S): ts(TS) } = 1 :- x(REGID,DAY,PH4,0,S).
{y(REGID,DAY+DAY2,TS,PH4,ORD,S): ts(TS)} = 1 :- x(REGID,DAY,_,N,_),
    registration(REGID,ORD,DAY2,PH4,PH3,PH2,PH1,S),ORD=N+1,day(DAY+DAY2).
:- y(REGID,DAY,TS,PH4,_,_), PH4 > 50, TS < 24.
:- y(REGID,DAY,TS,_,ORD,_), registration(REGID,ORD,_,_,PH3,PH2,PH1,S), TS-PH3-PH2-PH1<1.
res(REGID,DAY,TS..TS+PH4-1) :- y(REGID,DAY,TS,PH4,_,_), PH4>0.
chair(ID,REGID,DAY,TS) :- chair(ID,REGID,DAY), res(REGID,DAY,TS).
:-#count{REGID: chair(ID,REGID,DAY,TS)} > 1, chair(ID), day(DAY), ts(TS).
bed(ID,REGID,DAY,TS) :- bed(ID,REGID,DAY), res(REGID,DAY,TS).
:-#count{REGID: bed(ID,REGID,DAY,TS)} > 1, bed(ID), day(DAY), ts(TS).
support(REGID,DAY,TS) :- y(REGID,DAY,PH4,_,_,_), reg(REGID,ORD,_,_,PH3,PH2,_,_), PH2>0,
    PH4-PH3-PH2=TS, ats(TS).
numbReg(DAY,N,TS) :- N = #count{REGID: support(REGID,DAY,TS)}, day(DAY), ats(TS).
numMax(DAY,T) :- T = #max{N: numbReg(DAY,N,_)}, day(DAY).
numMin(DAY,T) :- T = #min{N: numbReg(DAY,N,_), N != 0}, day(DAY).
:~ numMax(DAY,M). [M@3,DAY]
:~ numMax(DAY,M), numMin(DAY,N), M-N > 0. [M-N@2]
```

Figure 3: CTS subproblem.

These atoms indicate that the patient's assignment to the corresponding resource is not feasible and all such atoms are grouped together under the generated id. At this point, this set is excluded as a possible solution, and a support atom `nogood_chair(NGID)` is generated. This atom is true only if the set of patients assigned to chairs in the current iteration `NGID` are re-assigned indeed to chairs. A similar process applies for beds with the atom `nogood_bed(NGID)`. Finally, the constraint ensures that both `nogood_bed` and `nogood_chair` cannot be true simultaneously, thereby preventing the P_M from producing an output that includes the same set of atoms, `bed(ID,REGID,DAY)` and `chair(ID,REGID,DAY)`, which has already been shown to result in an unsatisfiable P_S. As a result, the assignment is excluded from the solutions of P_M.

6. Experimental Results

In this section, we report the results of an empirical analysis on real-world data of the performance obtained with the LBBD approach and with the original, direct encoding for the CTS problem. The experiments were run on an AMD Ryzen 5 3600 CPU @ 3.60GHz with 16 GB of physical RAM. As ASP system, we used CLINGO version 5.4.1. Moreover, for the LBBD approach implemented in our work, we used the multi-shot feature of CLINGO [11] which allows to iteratively solve a problem without restarting from scratch at each iteration, so that we could add the new rules derived from the subproblems, i.e. the cuts, into the MP without restarting the process at each iteration. For the direct encoding, CLINGO was configured with the option *–restart-on-model* for faster optimization and a time limit of 200 seconds, while for the LBBD approach, we maintained the general timeout of 200 seconds but we also imposed a timeout of 30 seconds to each call to the subproblem. Data includes 4 weeks of scheduling, each week from Monday to Friday and with an average number of registrations equal to 596 (with standard deviation equal to 25). Each registration corresponds to a day of treatment, following a different regimen and up to five for a single patient.

Figure 4 shows the time required by CLINGO to solve the instances, with respect to the number of patients, using the direct encoding and the alternative LBBD approach. While the direct encoding reaches the timeout in all the instances without finding any solution (except for one where it obtained a low-quality solution), our approach solved all instances. About the quality of the solutions returned, although the new approach did not manage to prove the optimality of the solutions, the results in Figure 5 show that the solution found is very close to the optimum. Figure 5 provides the daily distribution of patients over a week, showing the number of patients accessing the second phase during each time slot. The patients are well distributed among the time slots with a maximum of 3 patients. The other optimization criterion is to assign patients to their preferred resource which is always fulfilled in the

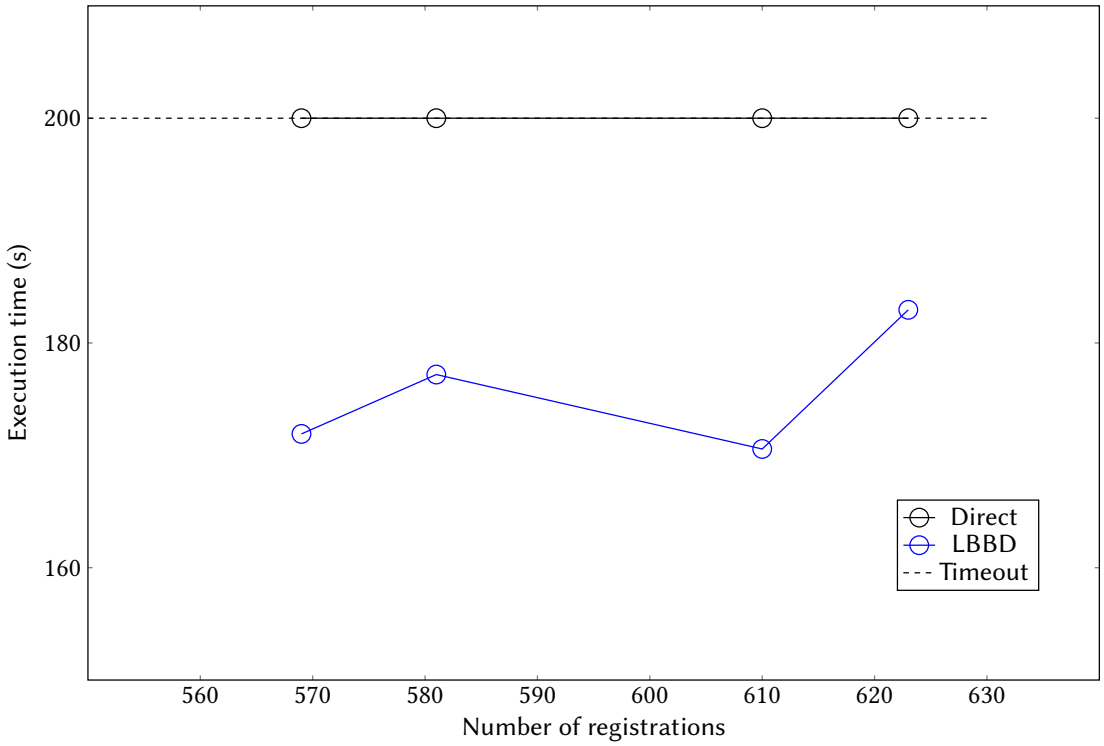
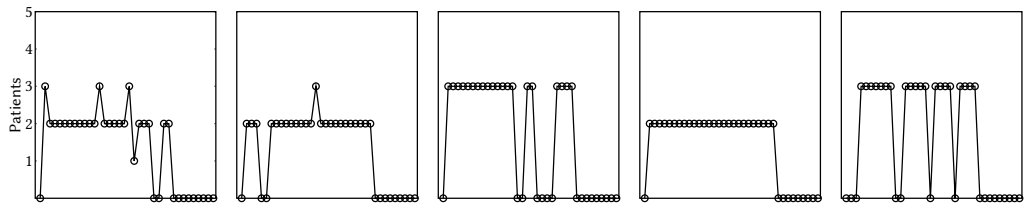


Figure 4: Time comparison of the performances of the direct encoding and our solution for the CTS problem.



(a) Day 1 (b) Day 2 (c) Day 3 (d) Day 4 (e) Day 5

Figure 5: Distribution of the patients accessing their second phase at the same time in a working week for the CTS problem.

results reported. Thus, the results prove that the LBBD approach achieves significant performance improvements over the direct approach on the CTS real-world data.

7. Related Work

The LBBD approach, which inspired our work, has already been used in combinatorial optimization problems, especially in scenarios where the problem can be divided into multiple parts. Its versatility and effectiveness have been demonstrated across various domains, most notably in scheduling problems: In this domain, [22] showcased the applicability of LBBD across different scheduling scenarios, underscoring its capability to manage complex, large-scale problems while enhancing both solution quality and computational efficiency.

Focusing on scheduling problems in the healthcare domain, there are some works in which it has been employed. [23] proposed a heuristic adaptation of LBBD to the home healthcare problem, focusing on designing efficient routes and schedules for workers who visit patient homes. Then, [24] applied the LBBD method to the problem of home hospice care staffing and scheduling using mixed integer programming. The objective here is to effectively match hospice care aides with patients and schedule visits to their homes. The authors leverage the versatility of LBBD to recompute staff assignments and

visitation schedules as needed efficiently. A similar problem was analyzed by [25] that focused on the home healthcare delivery problem. Specifically, the goal of their problem was to match patients with healthcare aides and schedule multiple home visits over a given time horizon, to maximize the number of patients served. The recursive LBBD approach was explored by [26]. In particular, they considered a class of multi-mode appointment scheduling problems involving variable resource availability and setup times. Moreover, they showed the efficiency of this recursive LBBD method using real-life data from a gastroenterology clinic at the University Hospital of Northern Norway. [27] developed three novel LBBD methods and a cut propagation mechanism to tackle the distributed ORS problem. In the same domain, [28] considered a stochastic variant of the problem, incorporating uncertain surgery durations by exploiting the LBBD cut to maintain efficiency and robustness. Recently, [10] presented the first LBBD approach for a specific scheduling problem expressed in ASP, involving chronic outpatients with non-communicable diseases who require ongoing hospital services, thus a problem similar to the ones considered by [24] and by [25]. In particular, they showed that LBBD can expand the applicability of ASP to larger instances without sacrifice the quality of the solutions. In this paper, which is somewhat more comprehensive, we achieved a comparable result for both scheduling problems analyzed.

8. Conclusion

In this paper, we have presented an analysis of the Chemotherapy Treatment Scheduling (CTS) problem, modeled and solved with ASP. Specifically, we proposed an LBBD approach implemented through the usage of multi-shot solving, and compared its results to a direct encoding on real data. Results have shown a performance improvement over the classical approach, in which all concepts and specifications are codified in a single encoding. Future works include the definition and analysis of other cuts, and the definition and implementation of additional solving strategies, e.g., of domain-dependent heuristics.

References

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

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

[3] A. Huggins, D. Claudio, E. Pérez, Improving resource utilization in a cancer clinic: An optimization model, in: IIE Annual Conference and Expo 2014, 2014.

[4] S. Sevinc, U. A. Sanli, E. Goker, Algorithms for scheduling of chemotherapy plans, Computers in Biology and Medicine 43 (2013) 2103–2109.

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

[6] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Magazine 37 (2016) 53–68.

[7] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intelligenz 32 (2018) 165–176.

[8] P. Schüller, Answer set programming in linguistics, Künstliche Intelligence 32 (2018) 151–155. URL: https://doi.org/10.1007/s13218-018-0542-z. doi:10.1007/s13218-018-0542-z.

[9] M. Alviano, R. Bertolucci, M. Cardellini, C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, M. Mochi, V. Morozan, I. Porro, M. Schouten, Answer set programming in healthcare: Extended overview, in: IPS and RCRA 2020, volume 2745 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2745/paper7.pdf.

[10] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Logic-based Benders decomposition in

answer set programming for chronic outpatients scheduling, Theory and Practice of Logic Programming 23 (2023) 848–864. URL: https://doi.org/10.1017/s147106842300025x. doi:10.1017/S147106842300025X.
[11] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82.
[12] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103.
[13] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, Theory and Practice of Logic Programming 20 (2020) 294–309.
[14] F. Buccafurri, N. Leone, P. Rullo, Enhancing Disjunctive Datalog by Constraints, IEEE Transactions on Knowledge and Data Engineering 12 (2000) 845–860.
[15] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artificial Intelligence 175 (2011) 278–298.
[16] J. Hooker, A hybrid method for planning and scheduling, in: Proc. of CP 2004, volume 3258 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 305–316.
[17] J. Benders, Partitioning procedures for solving mixed-variables programming problems., Computational Management Science 2 (2005).
[18] A. M. Geoffrion, Generalized benders decomposition, Journal of optimization theory and applications 10 (1972) 237–260.
[19] R. Rahmaniani, T. G. Crainic, M. Gendreau, W. Rei, The benders decomposition algorithm: A literature review, European Journal of Operational Research 259 (2017) 801–817.
[20] J. Hooker, G. Ottosson, Logic-based benders decomposition, Mathematical Programming 96 (2003) 33–60.
[21] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: Proc. of ICLP (Technical Communications), volume 52 of *OASICS*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15.
[22] J. Hooker, Logic-based benders decomposition: Theory and applications, Springer Cham, 2024. URL: https://doi.org/10.1007/978-3-031-45039-6. doi:10.1007/978-3-031-45039-6.
[23] A. Ciré, J. N. Hooker, A heuristic logic-based benders method for the home health care problem, in: Manuscript, presented at Matheuristics, 2012.
[24] A. Heching, J. Hooker, Scheduling home hospice care with logic-based benders decomposition, in: Proc. of CPAIOR 2016, volume 9676 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 187–197.
[25] A. Heching, J. N. Hooker, R. Kimura, A logic-based benders approach to home healthcare delivery, Transp. Sci. 53 (2019) 510–522. URL: https://doi.org/10.1287/trsc.2018.0830. doi:10.1287/TRSC.2018.0830.
[26] A. Riise, C. Mannino, L. Lamorgese, Recursive logic-based benders' decomposition for multi-mode outpatient scheduling, European Journal of Operational Research 255 (2016) 719–728.
[27] V. Roshanaei, C. Luong, D. M. Aleman, D. Urbach, Propagating logic-based benders' decomposition approaches for distributed operating room scheduling, European Journal of Operational Research 257 (2017).
[28] C. Guo, M. Bodur, D. M. Aleman, D. R. Urbach, Logic-based benders decomposition and binary decision diagram based approaches for stochastic distributed operating room scheduling, INFORMS Journal on Computing 33 (2021) 1551–1569.