Decomposition approaches for scheduling chronic outpatients' clinical pathways in Answer Set Programming*

Paola Cappanera[1 [0000−0003−3674−3896]], Marco Gavanelli[2 [0000−0001−7433−5899]], Maddalena Nonato[2 [0000−0001−8708−120X]], and Marco Roma[1 [0000−0002−0925−212X]]

[1] DINFO, Università degli Studi di Firenze, Italy, {paola.cappanera,marco.roma}@unifi.it
[2] DE, Università degli Studi di Ferrara, Italy, {marco.gavanelli,nntmdl}@unife.it

Abstract. Chronic patients suffering from non-communicable diseases are often enrolled into a diagnostic and therapeutic care program featuring a personalized care plan. Healthcare is mostly provided at the patient's home, but those examinations and treatments that must be delivered at the hospital have to be explicitly booked. Booking is not trivial due to, on the one hand, the several time constraints which become particularly tight in the case of comorbidity, on the other hand, the limited availability of both staff and equipment at the hospital care units. This suggests that the scheduling of the clinical pathways for enrolled outpatients should be managed in a centralized manner, taking advantage of the fact that demand for services is known well in advance. The aim is to serve as many requests as possible (unattended requests are supplied by contracted private health facilities) in a timely manner, taking patients priority into account. Booking involves setting a date and a time for each selected health service, which is rather complex. In this work, we provide a declarative approach by encoding the problem in Answer Set Programming (ASP). In order to improve the scalability of the ASP approach, we present and compare two heuristic approaches, respectively based on service demand and time decomposition. All approaches are tested on instances of increasing size to assess scalability with respect to time horizon and number of requests.

Keywords: clinical pathways; outpatient appointment scheduling; Answer Set Programming; decomposition approaches

1 Introduction

Increasing life expectancy in most of the countries world wide comes at the price of an ever increasing share of the elderly population. A large percentage of these people suffers from so-called Non-communicable Chronic Diseases (NCDs), such as diabetes, hypertension, cirrhosis, obesity, just to mention a few of them. For most NCDs, medical guidelines, built along standardized protocols and evidence-based medicine, are nowadays well-assessed. Beside drug therapies, such guidelines involve health services such as treatments and medical examinations to be repeated

*This work was partially supported by GNCS-INdAM.

on a regular basis. Once enrolled in a diagnostic and therapeutic care program, an NCD patient is assigned a personalized care plan, known as *Clinical Pathway* (CP in the following). A CP merges the medical guidelines regarding diagnosed NCDs and customizes them to the specific patient features [1]. As NCDs are chronic diseases, patients are expected to abide by their individual care plan for their entire life. Therefore, besides periodic reassessments, the health services included in a given CP are known a priori over a long time horizon, which allows for well in advance planning.

Less critical NCD patients live at home and are attended at their own domicile. Indeed, permanent hospitalization is postponed as long as possible to improve lifestyle quality and for the sake of budget containment. Nevertheless, such patients have to access hospital premises on a regular basis as outpatients in order to comply with their CPs and receive at the hospital those health services which cannot be delivered at home - think of dialysis for patients with renal failure. To add to this complexity, elderly patients often suffer from comorbidities, i.e., a same patient is subject to more than one NCD, which leads to more complex CPs and more frequent trips to the hospital. Care effectiveness relies on delivering assistance in a timely manner, according to each specific CP. Therefore the appointment schedule should strictly abide by the CP timetable.

Scheduling such CPs means to i) assign a date and ii) set a starting time for each health service a patient must receive at the hospital, that is i) build the master plan and ii) set the daily agenda for each day of the planning horizon. Such process can be quite challenging when limited resources do not allow to satisfy all requests (residual ones are outsourced to private health clinics). On the one hand, request selection should prioritize the most critical patients, such as minimizing the number of residual requests of highest priority as first, and so on. On the other hand, regarding selected services, schedule feasibility can be quite challenging as it requires to deal with several features: ensure regular frequency of periodic services, ensure that interfering services are not too close in time, ensure that when a service can be delivered only provided that another one is also delivered in a given time window, this requirement is satisfied. Patients, in particular if elder and fragile, are not able to take care of such complexity by themselves. Because of that, in Italy, following the guidelines given by the European Community `https://www.euro.who.int/__data/assets/pdf_file/0004/53860/E92341.pdf` the *Family Health Nurse* role has been recently introduced to act as the case manager, i.e., someone who is in charge of (beside other duties) scheduling an appointment for each health service included in the CP at each upcoming period (see `https://www.uslnordovest.toscana.it/attachments/article/7380/Progetto%20IFC%20Dipartimento%20Inf.%20co%20Asl%20TNO_all._Delibera.pdf`). However, at present, the Family Health Nurse interacts with the booking systems of the different care units one at a time, when booking one service at a time, trying to cope with the complexity of the time requirements in a trial and error strategy. Indeed, Family Health Nurses operate in a greedy manner, in the sense that they book the appointments one after the other at the different care units, unaware of the requests of other patients. Since the availability of resources at hospital care units - both in terms of staff and equipment - is limited, patients compete for access, and fairness (intended as giving top priority to the most critical patients) may not be ensured. This situation has been further exacerbated by the recent pandemic situation. Thus, appointment scheduling is a very critical and time consuming task for the Family Health Nurse,

putting at stake the timely delivery of care [10]. Indeed, coordination is crucial. Consider for example, a patient p_1 booking a service on day d and saturating the service capacity for that day. Then, another patient, say p_2, with higher priority than p_1, asks for service. If d is the only possible date for p_2, p_2 will fail to book an appointment, while other dates could have been likely possible for p_1. Therefore, (i) coordination and (ii) patient prioritization are essential in building feasible schedules when available resources do not allow to serve every request.

We argue that a centralized management of the CPs of all the enrolled patients would allow to i) optimize resource availability, ii) account for fairness by potentially considering patient priority, iii) ensure timely delivery of the care plans, iv) exploit the potential advantages of synchronized activities, and v) help the National Health System (NHS) save money by limiting the involvement of contracted private suppliers that come into play whenever the public system is not able to meet the requests on time. This yields a very challenging problem that will be called the Non-communicable Chronic Diseases Agenda problem (NCD Agenda). Despite of a rich literature on outpatient multi appointment scheduling - see [9] for a recent review - to our knowledge this particular problem has never been addressed.

This work aims at providing an automated tool to solve the NCD Agenda, thus releasing Family Health Nurses from this task, so that i) scarce resources are optimized and fairly managed, ii) clinical pathways are timely implemented, iii) as many requests as possible are satisfied by the NHS, prioritizing patient requests according to the patient status.

This paper is an extended version of [3], which is here expanded in several directions as summarized hereafter. Recall that in [3] only the master plan was addressed, that is the problem of assigning a day to as many requests as possible. In that problem, neither a start time nor an operator is assigned to each scheduled service. As a first contribution, this paper addresses the whole problem by providing an Answer Set Programming (ASP) approach for the daily agenda problem. Second, in order to improve the scalability of the ASP approach, we propose two ways of decomposing the problem, named Time Granularity Based and Patient Priority Based decomposition, respectively. Each decomposition could be thought of as a greedy heuristic where each step is an invocation of the ASP solver, in a sense similar to [16]. Thanks to decomposition, we are able to solve large size instances and, as a further plus, the Time Granularity Based approach may yield an upper bound that helps to evaluate the quality of the heuristic solutions.

All approaches are compared based on the results of an extensive experimental campaign, ranging from small to large instances.

The paper is organized as follows: Fundamentals of ASP are recalled in Section 2. Section 3 briefly reviews ASP based approaches devoted to optimization problems in the healthcare domain in order to shortly summarize the state of the art and highlight the paper novelties. Section 4 provides an abstract description of the NCD Agenda problem, introducing the decisions, the objective function, and the constraints specific to this problem in a general way. An ASP formalization for the abstract problem is presented in Section 5, while Section 6 presents the decomposition based solution approaches. Finally, computational results are discussed in Section 7 while conclusions are drawn in Section 8, where on going works are also sketched.

2 Preliminaries

Logic programming is one of the four programming paradigms [8]; a logic program consists of a set of clauses, in the form of implications $Head \leftarrow Body$, where $Head$ is an atom (or a disjunction) and $Body$ is a conjunction of literals (atoms, possibly negated). Atoms can have parameters that can be variables or constants. An atom (or a clause) is ground if it does not contain variables. A Declarative Semantics provides a formal meaning to a logic program by assigning a truth value to each ground atom in such a way to satisfy all the clauses. ASP is a logic programming language relying on the Stable Models semantics [5]; a logic program can have zero, one, or more than one stable models. In ASP, each solution to a combinatorial problem is associated with a stable model. There exist several solvers, based on different technologies; however, the best-performing ones are based on a two-phase solution scheme: a grounding phase, that generates a ground program equivalent to the original one, followed by a solving phase, in which a stable model of the ground program is computed. The ground program can be built by substituting to the variables in each clause all the possible constants appearing in the program (although modern grounders may avoid generating useless clauses), and, in the worst case, it is exponentially larger than the original one. One of the best ASP solvers is Clasp [4], which finds stable models by using technologies developed in SAT solvers such as conflict graphs, conflict-directed clause learning, and restarts. These technologies are very efficient in proving satisfiability/unsatisfiability of a problem; optimization problems are usually transformed into a sequence of satisfiability problems. ASP solvers can solve problems up to Σ_2^P. As a rule of thumb, to get an efficient solving algorithm one should produce programs whose grounding is not too large; for example, if some parameter of an atom can take a large variety of different values, the ground program can be very large.

Beside the basic syntax for clauses, recent ASP solvers accept several extensions [2]. A clause of the format $L \leq \{Head\} \leq U \leftarrow Body$ states that, in case the $Body$ is true, then the solver can choose the truth value of the atom in $Head$; this is the usual syntax for defining decision variables in optimization problems. The bounds L and U are optional; if they are present, then the number of true atoms matching with $Head$ in the stable model must be between L and U. If a clause is without head, $\leftarrow Body$, then the head is intended as the literal *false*, it is called an Integrity Constraint (IC) and in all stable models the $Body$ must be false.

3 Related papers

In the last few years, several decision problems in the health care domain have been tackled by means of ASP based approaches. A short review covering the latest on this topic can be found in [15]. In the following, the most notable contributions are recalled, emphasizing decomposition when present.

The study in [16] deals with scheduling outpatients due to take pre-operative exams before surgery. The problem is decomposed into two steps: First, high level scheduling decisions are taken, i.e., exam areas are staffed and patients are given an appointment day. Second, exams starting times are set, complying with first level decisions, so that served patients are maximized and waiting time minimized.

Since the two phases are solved in pipe, in a greedy way, demand is over estimated in phase one to ensure feasibility in phase two.
The study in [7] schedules multi appointments for rheumatic outpatients at a Hospital Day Service where multidisciplinary diagnostic tests and therapies are delivered. The problem is encoded into ASP. To face large computing times, patients are partitioned into three classes with decreasing priority, and the problem is solved three times, starting from the highest priority class. Residual capacity is updated at the end of each iteration.
In [13] the nurse (re)scheduling problem is addressed. Nurse scheduling consists of determining a shift assignment for each nurse for a given planning horizon such that working hours, shift mix, and rest days comply with hospital rules. Rescheduling is due in case of nurse temporary absences, and consists of feasibly scheduling vacant duties, while minimizing deviations between the new and the original schedule. This work improves on the representation of hospital and work balance constraints presented by [14]. In particular, the new encoding avoids parameters combinations that do not lead to feasible schedules.
Appointment scheduling for patients in need for chemotherapy treatments ([19]) must deal with the availability of special equipment, either a chair or a bed, that are assigned to a patient for the whole session. Patients have a preference for one of the two. A treatment encompasses up to 4 subsequent steps, some of which are optional, whose duration is patient dependant and known. In case of multiple treatments, a treatment frequency is given. While resource assignment is a bottleneck, evenly spreading patients requiring blood collection is a target. The weekly problem is solved, as well as the rescheduling one. The ASP program is further generalized to include patient priority and staff and drugs availability. Results show an improvement with respect to real schedules.
Rehabilitation sessions for inpatients are scheduled in [18]. There, two types of resources are present: gyms, whose access is restricted to patients on the same floor and have a given capacity, and operators, who can supervise a maximum number of patients at the same time. Solution quality criteria and constraints include: on the patient side, continuity of care and preferred time slots; on the operator side workload balancing and abiding by the working rules. The daily problem is decomposed into two subsequent decision phases, namely the board and the agenda, which are solved in pipe: to build the board, patients are assigned to operators with an eye to care continuity and by keeping the cumulative working time below the limits; to fill the agenda according to the board, a starting and ending time is set for each session, considering gym capacity, with an eye to slot preferences. There is no guarantee that a feasible agenda exists which is board-compatible and complies with the rules. Therefore, some rules are overridden, i.e., potential overlapping are admitted: some sessions are partially turned from one-to-one care to supervision (one operator supervises a few patients at a time). Experiments run on real and realistic data assess efficacy and efficiency of the approach.
Finally, ASP has been proved effective in tackling a very challenging problem in health care management: the (re)scheduling of operating rooms. Difficulties arise since it is deeply intertwined with the management of staff and resources. Indeed, a planned surgery requires a free bed at the specialty ward or at intensive care units, starting from surgery date for the predicted length of stay ([20]), and a bed at the post anaesthetic care unit for temporary post-surgery staying ([17]). Since a surgical team is made of surgeons, anesthesiologists, and nurses, the whole surgery

slot must be fully contained into the current working shift of each team member. Based on its specialty, priority, special needs, and expected duration, a request is assigned a day and a time during the operating rooms time blocks reserved to its specialty. The proposed ASP encoding proved able to tackle realistic instances and to scale well with time.
In conclusion, we observe, as a general statement, that ASP proved able to capture and easily represent the complex features of several challenging scheduling problems arising in the health care domain. Decomposition schemes are often implemented, motivated by the need for solving large instances in a reasonable time. Usually, such decomposition patterns consist of splitting the decision set into several subsets according to different strategies, either hierarchical (with an increasing detail level) or based on the value of some attributes of the input; then, the resulting sub-problems are solved in pipe. Thus, not only optimality is not guaranteed, but in some cases even feasibility may not be ensured without resorting to heuristic patches.
This paper builds upon [3] and extends it in several ways. In that paper, the problem was introduced for the first time and the focus was on scheduling the master plan. Here, we address the whole problem and discuss different decomposition schemes to face the computational burden that arises when solving realistic instances.

4 The NCDs Agenda problem

Let us formally introduce the problem components, i.e., the decisions to be taken, the objective function that measures the quality of a solution, and the constraints that define the conditions under which a schedule is feasible. These constraints range from those concerning individual patients up to those affecting different patients who share the same care unit on the same day.
Table 1 summarizes the formal notation. Consider a planning horizon (i.e., a set of days) denoted as $Horizon$. Each pathway $cp(p)$ consists of a set of packets Π_p, where a packet $\pi \in \Pi_p$ is made of a set of services $S(\pi)$. In the following, $s \in S(\pi)$ will be written as $s \in \pi$ to simplify the notation. Each $s \in \pi$ has a given duration dur^s and can be delivered by any operator of a given care unit $cu(s)$. All $s \in \pi$ must either be all delivered on the same date or rejected. In the second case we speak of residual packets and residual services. Ideally, the delivery date of packet π is $d^*(\pi) \in Horizon$, with a tolerance of $\pm\rho(\pi)$ days.A schedule is described in terms of i) delivered packets $\Pi^* = \bigcup_d \Pi(SP(d))$ and residual ones $\overline{\Pi}$; ii) for each delivered packet π, the delivery date, denoted as $\tau(\pi)$ (as the one of all services in that packet); iii) the starting time of each delivered service s, denoted as $t(s)$, and iv) the operator who will deliver each scheduled service s, denoted as $Op(s)$. At the end of the solution process, $\Pi = \Pi^* \cup \overline{\Pi}$.

4.1 A patient's pathway feasibility

The schedule of a pathway must comply with three types of time-constraints, i.e., *Frequency*, *Interdiction*, and *Necessity*, called FIN constraints in the following.
Frequency ensures the timely delivery of care and it is satisfied once the date of π belongs to $[d^*(\pi) - \rho(\pi), d^*(\pi) + \rho(\pi)]$.
Regarding *interdiction* and *necessity*, consider two health services s_i, s_j in different packets of the same pathway. The following constraints may be given:

Table 1: Mathematical notation

Symbol	Description
Input data	
$Horizon = \{1, .., n_H\}$	A planning horizon, i.e., a set of days.
$S = \{s_1, s_2, \dots\}$	A set of health services.
$P = \{p_1, p_2, \dots\}$	A set of patients.
$Priority(p)$	Priority of patient p.
P^I, P^{II}, P^{III}	Patient sets with decreasing priority.
$cp(p)$	The clinical pathway of patient p.
Π_p	The set of packets in $cp(p)$.
$\Pi = \bigcup_p \Pi_p$	The set of packets.
$\pi = \langle S(\pi), d^*(\pi), \rho(\pi) \rangle$	A packet, defined as a triplet.
$S(\pi) \subseteq S$	The services of packet π, to be delivered on the same day even if at different care units.
$d^*(\pi)$	Ideal delivery date of π.
$\rho(\pi)$	Tolerance with respect to the ideal delivery date.
dur^s	Duration of service s.
$cu(s)$	The care unit entitled to deliver service s.
$ShiftStart^d(o)$	Starting time of the working shift of operator o on day d.
$ShiftDur^d(o)$	duration of the shift of operator o on day d.
$Shift^d(o)$	the working shift of operator o on day d $\left[ShiftStart^d(o), ShiftStart^d(o) + ShiftDur^d(o)\right]$
$O(cu)$	Operators at the same care unit cu.
$CU = \{cu_1, cu_2, \dots\}$	A set of care units.
Decision variables	
$\tau(\pi)$, $\tau(s)$	Date of packet π, date of service s.
$t(s)$	Starting time of service s.
$Op(s)$	Operator delivering service s.
$\Pi(MP)$	Set of packets selected in the master plan.
$\Pi(SP(d))$	Packet set scheduled in the agenda of day d.
Π^*	Set of delivered packets in a solution.
$\overline{\Pi}$	Set of residual packets. In particular, $\overline{\Pi}(MP)$ denotes those discarded by the master problem and $\overline{\Pi}(SP(d))$ those discarded by $SP(d)$.

Forward (Backward) Interdiction:
$\tau(s_i) = d \Rightarrow \tau(s_j) \notin [d, .., d + \delta]$ ($[d - \delta, .., d]$).
Forward (Backward) Necessity:
$\tau(s_i) = d \Rightarrow \tau(s_j) \in [d + \delta + 1, .., d + \delta + \Delta]$ ($[d - \delta - \Delta, .., d - \delta - 1]$).
Note that interdiction constraints are satisfied even if s_j is never scheduled, while necessity constraints require s_j to be given a date that must be next but not too close to the date of s_i. Examples regarding interdiction include the case of one service affecting the results or the effectiveness of another service when too close in time, i.e., at least δ days must elapse between the two dates. Necessity typically concerns the timing of a primary activity and a secondary one which is functional to the first one only if sufficiently but not too close in time.
Any time constraint set on $\tau(s)$ propagates on all the other services in the same packet. Therefore, the available options for feasible dates may become rather limited, despite of tolerance. For a given staffing level, an incremental booking process - which to our knowledge is common practice in most cases - may fail to find time-feasible dates with enough residual capacity to accommodate all requests, since previous booking has been done without knowledge of incoming demand.
The FIN constraints so far presented concern the schedule of individual pathways, with no interaction among patients, and characterize the master plan. The set of packets selected in the master plan, i.e., all π s.t. $\tau(\pi) = d,\ d \in Horizon$ will be denoted by $\Pi(MP)$. However, patients who receive service from the same care unit on the same day compete for operators' time and other limited resources. At this stage of the project, operators in the same unit cu, $O(cu)$, have identical skills but potentially different shifts, which may also vary on a daily basis, denoted as $\left[ShiftStart^d(o), ShiftStart^d(o) + ShiftDur^d(o)\right]$. Therefore additional constraints are also necessary to ensure feasibility, as discussed hereafter.

4.2 The daily agenda problem

For each day d, the daily agenda sub-problem $SP(d)$ receives from the master plan the list of packets that the master problem has scheduled on day d, formally the set $\{\pi \in \Pi(MP) \text{ s. t. } \tau(\pi) = d\}$. Each sub-problem $SP(d)$ consists of computing a feasible schedule for as many such packets as possible. In detail, select the maximum number of packets (according to patients priority) such that for each of their services s it is possible to set a starting time $t(s)$ and an operator $Op(s)$ so that: i) a patient does not receive more than one service at a time; ii) operator $Op(s)$ will serve at most one patient at a time, and iii) operator o can deliver each service that has been assigned to him/her (namely, the set of services $\{s : Op(s) = o\}$) without preemption and within her/his working shift defined as $\left[ShiftStart^d(o), ShiftStart^d(o) + ShiftDur^d(o)\right]$.
In particular, given two different service s_i and s_j, with $i \neq j$,
i) requires that either $t(s_i) + dur^{s_i} \leq t(s_j)$ or $t(s_j) + dur^{s_j} \leq t(s_i)$ whenever both $s_i, s_j \in cp(p)$ (that is, whenever the two services belong to the pathway of the same patient, the time intervals in which the services are delivered do not overlap);
ii) requires that the same non overlapping condition holds when $Op(s_i) = Op(s_j)$ (the two services s_i and s_j are delivered by the same operator);
iii) means that $ShiftStart^d(Op(s_i)) \leq t(s_i)$, that is, any service must start not

earlier than the beginning of the shift, and $t(s_i) + dur^{s_i} \leq ShiftStart^d(Op(s_i)) + ShiftDur^d(Op(s_i))$ stating that any service must end no later than the end of the shift.
We refer to these constraints as the *daily agenda* constraints.

4.3 A minimization problem

Patients are classified according to their health conditions and a priority is given accordingly, yielding N classes of patients, being the first class the highest priority one, namely P^I, P^{II}, $\cdots$, P^N. The aim is to serve as many requests as possible according to priority. A hierarchical objective function based on patients priority is then proposed. This means that, for each $p \in P^I$ and each $\pi \in cp(p)$, either schedule π (and set the values of $\tau(\pi), t(s)$ and $Op(s)$) or discard it, so that the number of residual packets of highest priority patients is minimized. Then, and in case of ties, the number of residual packets of patients in P^{II} is minimized, and so on, until the least priority class is processed.
Summing up, given the clinical pathways of all patients in P and the resource availability for the current planning horizon described in terms of the working time windows of the operators of each care unit, the NCDs Agenda problem consists of taking decisions at three levels: 1) for each $p \in P$ and each $\pi \in cp(p)$, either select π or discard it, in order to minimize the hierarchical objective function; 2) for all selected π assign a date $\tau(s)$ to each $s \in \pi$ so that FIN constraints are satisfied; 3) for all d, and for all s such that $\tau(s) = d$ set a time $t(s)$ so that the daily agenda constraints are satisfied.

5 An ASP approach

The ASP formulation takes as input the description of the instance by means of a set of predicates. Predicate `patient(P,Pri)` is true for each patient `P` with priority `Pri`; his/her packets are listed with facts `packet(P,Pkt)`. Predicate `service_in_packet(S,Pck)` indicates that service `S` belongs to packet `Pck`. The set of services is provided by predicate `srvType(S,CU,Dur)` where `S` identifies the service, `CU` is the resource type (e.g., the care unit) and `Dur` the service duration. Predicate `tot_capacity(D,CU,TotDur)` provides the total duration of available service in the care unit `CU` in day `D`. An interdiction of `D` days between two services `S1` and `S2` is declared through the predicate `interdiction(S1,S2,D)`, while `necessity(S1,S2,(Min,Max))` states that service `S1` requires `S2` to be scheduled between `[Min,Max]` days in advance. Predicate `day(D)` is true for each available day `D` in the horizon. Predicate `ideal_date(P,Pck,ID)` provides, for each packet `Pck` for a patient `P`, the ideal date `ID` for scheduling the packet, while `within_tolerance(Pck, D, ID)` is true if day `D` is within the given tolerance from the ideal date `ID` for packet `Pck`.

5.1 The Master Problem

The main decision Master Problem (MP) takes is the schedule of each packet π for each patient; the next clause declares that given a packet `Pck`, it can be scheduled in a day within the tolerance from the ideal date:

```
0 <= {schedule(P, Pck, D) :
       day(D), within_tolerance(Pck,D,ID)
      } <= 1
     :- packet(P, Pck), ideal_date(P, Pck, ID).
```

Note that the number of `schedule` atoms for such packet is required to lie in $\{0,1\}$: in fact, a packet could be not scheduled, and the number of scheduled packets should be maximized (see the objective function).
Interdiction between services is implemented by the following IC:

```
:- schedule(P, Pck1, D1), service_in_packet(S1,Pck1),
   schedule(P, Pck2, D2), service_in_packet(S2,Pck2),
   interdiction(S1,S2,Delta),  D1<D2, D2 <= D1+Delta.
```

stating that if service `S1` interdicts `S2` for `Delta` days, the two services are scheduled for the same patient, and the day `D2` in which `S2` is scheduled is between `D1` and `D1+Delta`, then such schedule is inconsistent.
Necessity is dealt with by the following IC; if a necessity is not satisfied within the planning horizon nor it can be postponed beyond it, then the schedule is inconsistent

```
:- schedule(P, Pck1, D1), service_in_packet(S1,Pck1),
   necessity(S1, S2, _),
   not satisfies_necessity(P, Pck1, S1, S2),
   not necessity_beyond_horizon(P, Pck1, S1, S2).
```

where the two predicates used in the IC are defined as

```
satisfies_necessity(P, Pck1, S1, S2) :-
  necessity(S1,S2,(Min,Max)),
  schedule(P, Pck1, D1), service_in_packet(S1,Pck1),
  schedule(P, Pck2, D2), service_in_packet(S2,Pck2),
  D1+Min <= D2, D2 <= D1+Max.
necessity_beyond_horizon(P, Pck1, S1, S2) :-
  D1+Max>n_H, necessity(S1, S2, (Min, Max)),
  schedule(P, Pck1, D1), service_in_packet(S1,Pck1).
```

Let `OpenDur` be the total available hours in day `D` for care unit `CU`; if the sum of the durations of services attended by `CU` and scheduled in day `D` exceeds `OpenDur`, then the schedule is inconsistent:

```
:- tot_capacity(D, CU, OpenDur),
#sum{Dur,P,S:schedule(P,Pck,D),service_in_packet(S,Pck),
     srvType(S,CU,Dur)}>OpenDur.
```

The objective is defined through the so-called *weak constraints*, i.e., integrity constraints that can be relaxed, with the objective to maximize the number of the satisfied ones. Weak constraints are identified syntactically by using $:\sim$ as implication symbol. Each weak constraint can have a priority and a weight: the ASP solver searches the solution maximizing the weighted sum of the highest-priority weak constraints; among those solutions, the weight of satisfied second-highest-priority weak constraints is maximized, and so on.
The objective function defined in Section 4.3 is simply implemented through the weak constraint

```
:~packet(P, Pck), not schedule(P, Pck, _),
 patient(P,Pri). [1@Pri,P,Pck]
```

maximizing the number of scheduled packets for the patients at the highest priority class and, in case of ties, those at the second-highest priority, and so on.

5.2 The Daily agenda

Note that even if the ASP program showed in Section 5.1 provides an assignment of packets to days, it might still be the case that for some day there exists no feasible assignment of times of the day for each service the master plan has scheduled on that day. For example, for some day one patient could be required to be serviced by two different care units at the same time. In order to obtain a feasible solution of the whole problem, we now extend the ASP formulation [3] to address also the daily agenda problem.
For each operator `Op` in a care unit `CU` and for each day `D`, predicate `op_shift(D, CU , Op, Start, Duration)` provides the `Start` time and the `Duration` of the shift.
The decisions the ASP program should take consist of the start time and the operator assigned to each service that was scheduled on a day. Predicate `start_time` assigns a start time to each service belonging to a packet scheduled in the day.

```
1<={start_time(P, S, Day, Start):time(Start)}<=1 :-
    schedule(P,Pck,Day), service_in_packet(S,Pck).
```

For each scheduled service `S`, the assignment to the operator `Op` that provides it is encoded in predicate `provides(CU,Op,P,S,D)`, where `P` is the patient and `D` is the day.

```
1<={provides(CU,Op,P,S,D) :
   op_shift(D,CU,Op,_,_),srvType(S,CU,_)}<=1                (1)
   :- schedule(P,Pck,D),  service_in_packet (S,Pck).
```

The constraints on the daily agenda state that two services involving the same patient or the same operator cannot be in parallel. We first define the concept of precedence: a service `S1` for patient `P1` precedes service `S2` for patient `P2` if the end time of `S1` is before (or in the same instant as) the start time of `S2`:

```
precedes(P1,S1,P2,S2,Day) :- srvType(S1,_,D1),
 start_time(P1,S1,Day,Start1), Start1+D1<=Start2,
 start_time(P2,S2,Day,Start2).
```

We can now state that if two services are provided to the same patient or by the same operator, then one of the two services precedes the other:

```
:- not precedes(P,S1,P,S2,Day),
   not precedes(P,S2,P,S1,Day), S1 != S2,
   provides(_,_,P1,S1,Day), provides(_,_,P2,S2,Day),
   schedule(P,Pck1,Day), service_in_packet(S1,Pck1),
   schedule(P,Pck2,Day), service_in_packet(S2,Pck2).
:- not precedes(P1,S1,P2,S2,Day),
   not precedes(P2,S2,P1,S1,Day), P1!=P2,
   provides(CU,Op,P1,S1,Day), provides(CU,Op,P2,S2,Day),
   schedule(P1,Pck1,Day), service_in_packet(S1,Pck1),
   schedule(P2,Pck2,Day), service_in_packet(S2,Pck2).
```

Finally, each service should be scheduled within the working shift of the operator that provides it:

```
:- provides(CU,Op,P,S,Day), start_time(P,S,Day,Start),
   srvType(S,CU,Dur), Start+Dur > ShiftStart+ShiftDur,
   op_shift(Day,CU,Op,ShiftStart,ShiftDur).
:- provides(CU,Op,P,S,Day), start_time(P,S,Day,Start),
   op_shift(Day,CU,Op,ShiftStart,ShiftDur),
   Start<ShiftStart.
```

6 Decomposition approaches

The ASP formulation described in Sections 5.1 and 5.2 is a declarative approach able to find the optimal solution given enough computational resources. On the other hand, in real life applications addressing NP-hard problems, the optimal solution is often not achievable in reasonable time, and obtaining good solutions in practical time is often enough. One solution strategy often reported in the literature (see Section3) is to decompose the whole problem into smaller sub-problems and solve them independently. The final solution is obtained by recombining the subproblems solutions and enforcing coherence among them in case it is not guaranteed. We propose two features along which decomposition may be carried out.

The first one relies on decomposing service demand, according to which the same problem is solved several times on a different input. In this case, we adopt patient priority as the decomposition criterion, in line with the objective function. We call this approach the Patient Priority Based decomposition.

The second approach is based on decomposing the time decisions according to their granularity, and it consists of solving in pipe two different problems once: the master problem first and the daily agenda problem second. We call this approach the Time Granularity Based decomposition.

Typically, decomposition schemes suffer from the limit intrinsic in the greediness of the approach as decisions taken in the first ASP program are typically never backtracked upon. We propose some adjustments in each of our approaches to edge against this drawback: in the Patient Priority Based greedy approach, the starting time of the scheduled services are computed at each iteration as part of the process of building a feasible solution; however, at each new iteration, they can be adjusted in order to accommodate additional services. The values in the final solution are the ones computed at the last iteration. On the other hand, in the Time Granularity Based greedy approach, we enrich the master problem by adding a relaxed version of the constraints of the daily agenda problem, aiming at increasing the number of services selected by the master plan that also admit a feasible starting time. A pictorial view of the two approaches is provided in Fig.1 and Fig.2.

6.1 Decomposition by time granularity

This two level approach exploits different time granularity and fixes the decisions in a greedy way. At the highest granularity the time unit is the day. Indeed, the master problem addressed by the ASP module described in Section 5.1 is solved

and it returns a master plan. The target is to minimize the weight of the unsatisfied requests, according to the hierarchical objective function introduced in Section 4. To improve the chances that the master plan can be feasibly implemented at the daily level in the following step, a partial awareness of the resource availability is included into the master problem, i.e., beside FIN constraints, a relaxed version of the resource constraints are added. Namely, the duration of the services requiring an operator of care unit cu scheduled on a certain day must be not larger than $\sum_{o \in O(cu)} End_o - Start_o$ where $[Start_o, End_o]$ is the time window of operator o on that day. Nevertheless, solution optimality is not guaranteed because of the partial awareness (other daily agenda constraints are overlooked) so that some services selected in the master plan cannot be feasibly scheduled during the second step. Therefore, while the whole approach is greedy-like, the decisions present in the master plan may have to be reversed (discard an accepted service) depending on the results of the next step.

As for the implementation, the model returned by the ASP approach contains the `schedule` predicate which assigns a day (corresponding to $\tau(\pi)$) to all delivered packets.

In the second step, for a given master plan, for each day d of the planning horizon $Horizon$, the daily agenda problem is solved. In this case, we again solve the ASP program in Section 5.2, with the following modifications in order to solve an optimization problem: 1) not all services scheduled by the MP are actually provided by an operator, i.e. in Eq 1 for each scheduled packet there can be between 0 and 1 atoms `provides`/5 instead of between 1 and 1:

```
0<={provides(CU,Op,P,S,D) :
   op_shift(D,CU,Op,_,_),srvType(S,CU,_)}<=1                    (2)
   :- schedule(P,Pck,D),  service_in_packet (S,Pck).
```

2) we maximize the number of packets (considering the patient's priority) that can be feasibly delivered, where a packet is delivered only provided that each of its services is assigned an operator

```
#maximize { 1@Pri,Pat,Pkt : sat_pkt(Pat, Pkt),
                            priority(Pat, Pri) }.
sat_pkt(Pat, Pkt) :-
  provides(_,_,Pat,Srv) : service_in_packet(Srv,Pkt);
  schedule(Pat, Pkt, day).
```

that is, a subproblem $SP(d)$ having the same objective function as the MP, whose input are i) $\Pi(d)$, i.e., all the packets π such that $\tau(\pi) = d$, ii) the resource availability of each care unit in terms of the starting and ending time of each operator.

Note that the subproblems $SP(d)$ can be solved in parallel.

It is worth noting that the final solution obtained by the union of the models returned by the MP and by the sub-problems could still be inconsistent with respect to necessity constraints. E.g., suppose the MP assigns a service s_1 to day d_1, and s_2 to d_2 (together with several other packets), where s_1 requires s_2. Suppose that, while the sub-problem $SP(d_1)$ is satisfiable, that for $SP(d_2)$ is not (due to the various constraints within the packets of day d_2), and its optimal solution does not assign an operator to s_2. Joining all the models, we have that s_1 is delivered while s_2 is not, despite the necessity constraint.

For this reason, a posteriori we check the validity of each necessity constraint and, in case one is not satisfied, we (recursively) remove from the solution the packet requiring the non existing service. This reduces the quality of the objective function, but provides a solution satisfying all constraints.

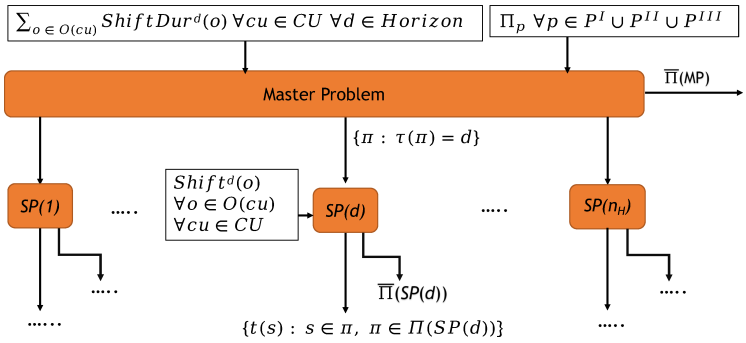

Fig. 1: Decision flow for Time Granularity Based in case of three priority classes and n_H days

6.2 Decomposition by priorities

An alternative decomposition criterion operates on the input of the instance. It subdivides the patient set into a few classes, as many as the different patient priority levels - 3 in our case, P^I, P^{II}, and P^{III}. The highest priority ones P^I are processed first, by solving the full problem (containing both the ASP modules in Section 5.1 and 5.2). In this way we maximize the chances of high priority patients requests to be served. This phase returns the value of $\tau(\pi)$ and $t(s)$ of all scheduled requests. The idea is to fix these decisions, update resource availability accordingly, and solve the whole problem again with respect to P^{II}. Finally, iterate on P^{III}. However, to retain as much flexibility as possible, the different time granularity of the decisions are exploited: only the values of $\tau(\pi)$ are fixed and passed to the next step, while the starting times of scheduled services will be recomputed in the best possible way during the next iteration. The values of the starting time computed at the last iteration are those returned as part of the solution. We trade computational time (as we potentially recompute the value of the starting time of a selected service more than once) for solution improvement. In this sense, this is a smart variant of a classical decomposition approach that has been applied to many real problems, such as in [7] as mentioned in Section 3.
In conclusion, each criterion our decomposition is based on in the two approaches, i.e., input decomposition and time decomposition, are not new and have been somehow proposed in other studies, such as [7] for the first and [18] for the second. However, we added original features to the general schemes to tailor them to this specific problem. In our case, indeed, in the Time Granularity Based approach, the master problem is enriched with some knowledge of the constraints of the subproblems.In the Patient Priority Based approach we propose an enhanced

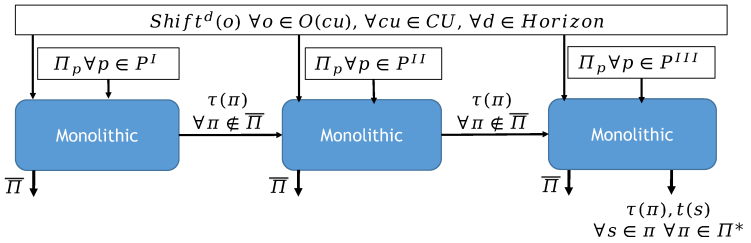

Fig. 2: Decision flow for Patient Priority Based in case of three priority classes

variant which computes the starting time of a service as a proof of feasibility of the current solution but may later backtrack on such a decision in favour of a better solution. Moreover, to our knowledge, the two decomposition schemes have never been compared on the same problem.

7 Computational results

Computational results have been obtained on randomly generated pathways, inspired by well assessed and publicly available medical guidelines for the most common NCDs (such as [11] and [12] for diabetes). A brief description of the instance generator follows. First, the generator allocates resources on a weekly basis. Specifically, 5 CUs are considered, and on each day of the week, a number of operators drawn with uniform probability between 1 and 4 are activated for each CU. The total capacity of each CU for each day is a value drawn with uniform probability between 24 and 60 time slots, allocated among the operators. The weekly resource allocation is then replicated in each week of the given time horizon. Second, CPs are generated, each of them consisting of service packets. Each packet is assigned a frequency, an ideal initial date and a tolerance. Each service has a duration ranging from 6 to 15 slots and it refers to a CU. Each packet consists of at most 4 services. In terms of service demand, given the desired number of patients, each of them is associated with a priority class (low, medium, and high) and a number of CPs between 1 and 4. The probability of assigning a patient to a class is inversely proportional to the priority. The probability of assigning a certain number of CPs to a given patient is inversely proportional to the number of CPs itself. The average number of slots requested daily (total number of slots requested by services divided by the length of the time horizon) remains nearly constant for all the instances as the length of the time horizon varies, and is directly proportional to the number of patients.

For each combination of number of patients in $\{10, 20, 40\}$ and length of the planning horizon in $\{30, 60\}$, 20 instances are generated, summing up to 120 instances. Detailed information on the average numbers - across the 20 instances, of packets and services, as well as the minimum and maximum number of services among the 20 instances are given in Table 2 separately for each group of instances.

Table 2: Instance features

Instance group	Avg # packets	Avg # services	Min # services	Max # services
10-30	56.5	69.6	32	193
20-30	118.0	147.6	69	389
40-30	229.4	286.0	125	687
10-60	112.7	138.8	65	374
20-60	234.9	294.0	140	734
40-60	456.4	568.4	263	1340

Experiments[3] have been run on a Linux Mint 19.1 Tessa OS, Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz machine with a 64GB System Memory. The solver version is Clingo 5.6.2.

In the following, we report computational results both in terms of efficiency and efficacy for the three approaches proposed, i.e., monolithic (the program composed of the two modules described in Sections 5.1 and 5.2), Patient Priority Based, and Time Granularity Based.

Table 3 offers an overview of the results reporting separately, for each of the three approaches and for each instance group, information about average computational times (efficiency) and number of instances successfully solved (efficacy). Specifically, for what concerns efficiency, we report both the (average) total computational time and the (average) time spent in each phase, i.e., (average) solving and grounding times. For what concerns efficacy, we report the number of instances in which the algorithm terminated within the time limit of 3600 seconds, the number of instances that reached the time limit, and the number of instances for which a memory error occurred - # errors. For each approach and for each group of instances, these three values sum up to the number of instances, i.e., 20. Interestingly, we observe the following facts: the average time spent in the grounding phase is very limited and stable across the instance groups for the Time Granularity Based approach. Contrarily, for the monolithic approach, the percentage of time spent in the grounding phase with respect to the total time ranges from a minimum of 23.41% (on the 40-60 instances) to a maximum of 50.19% (on the 10-60 instances), with an average of 37.45% across all the 120 instances. For the Patient Priority Based approach, the figures are: minimum of 31.61% (on the 40-30 instances), maximum of 83.84% (on the 20-60 instances), average of 53.21%. In addition, the Time Granularity Based method turns out to be the best option of the three approaches in terms of number of instances successfully solved in each group. This ability does not seem to depend on the length of the time horizon. Instead, as the number of patients increases, the effectiveness tends to decrease, especially when 40 patients are considered. Yet, its efficacy remains far better than that obtained by the other two methods. Moreover, Time Granularity Based is the only method among the three ones for which there are no memory errors.

As a further overview, Figure 3 shows the number of instances for which each of the three methods terminated within the time plotted on the x-axis. As noted above, the Time Granularity Based method performs the best in terms of both

[3]The code of the project, as well as the instances, are contained in the GitHub repository: `https://github.com/MarcoRoma96/NCD-agenda-outpatients.git`

Average computational time and memory errors							
		10-30	20-30	40-30	10-60	20-60	40-60
Avg. time	monolithic	580.1	1403.1	1146.8	664.4	1161.0	2584.2
	decomp. by patients	529.4	1504.8	2163.2	871.9	1400.2	2362.4
	decomp. by time	17.6	540.4	1980.4	183.7	563.4	1981.8
Avg. solving time	monolithic	383.7	1021.3	683.9	331.0	584.4	1979.2
	decomp. by patients	199.8	898.5	1479.5	292.5	226.3	1540.4
	decomp. by time	17.4	540.1	1980.0	183.4	563.0	1981.0
Avg. grounding time	monolithic	196.4	381.8	462.9	333.5	576.6	605.0
	decomp. by patients	329.6	606.4	683.7	579.3	1173.9	822.0
	decomp. by time	0.2	0.3	0.4	0.3	0.5	0.8
#solved before timelimit (3600s)	monolithic	18	11	5	17	9	1
	decomp. by patients	19	12	5	17	10	2
	decomp. by time	20	17	9	19	17	9
#reached timelimit	monolithic	2	5	0	1	1	1
	decomp. by patients	1	5	4	1	1	1
	decomp. by time	0	3	11	1	3	11
#errors	monolithic	0	4	15	2	10	18
	decomp. by patients	0	3	11	2	9	17
	decomp. by time	0	0	0	0	0	0

Table 3: For each group of instances (columns) and for each approach (rows) the following information are given: average (across the instances) total computational time -Avg time, average time spent in the solving phase - Avg solving time, average time spent in grounding phase - Avg grounding time, numbers of instances for which a memory error occurred - # errors, number of instances solved within the time limit fixed to 3600 seconds.

the number of instances solved and the computation time required. The other two methods show comparable performance between them.

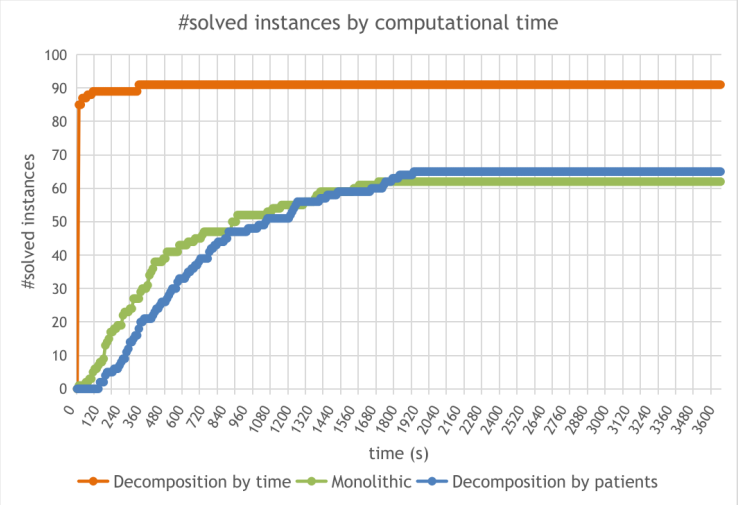

Fig. 3: For each approach, the y-axis shows the number of instances for which the algorithm terminates within the time limit in the x-axis.

A more detailed view of the distribution of computational times over the 6 groups of instances is shown in Figures 4, 5, 6 for the monolithic, Patient Priority Based, and Time Granularity Based approaches, respectively. The computational results seem to show that the impact of the number of patients on the computational time is greater than the impact of the time horizon length, at least for the monolithic and the Patient Priority Based approaches.

Finally, Figure 7 shows an overview of the performance of the three approaches in terms of both computational efficiency and quality of the solution obtained. Specifically, the quality of the solution is measured in terms of the percentage of unscheduled services in each priority class by giving a gradually decreasing weight to the three classes (from high to low priority). In fact, since the choice of the optimum solution is driven by a hierarchical objective function that first minimizes the number of unscheduled high priority services, then the number of medium priority services, and finally the number of low priority services, these three percentages can be transformed into a single numerical value, where the three least significant digits represent the percentage of unscheduled low priority services, and so on. The figure shows a rather sharp separation between the Time Granularity Based approach on the one hand and the other two approaches (monolithic and Patient Priority Based) on the other hand. Further investigation is needed to understand whether the clouds shown in the figure correspond to instances with specific characteristics for which one method is preferable to the other.

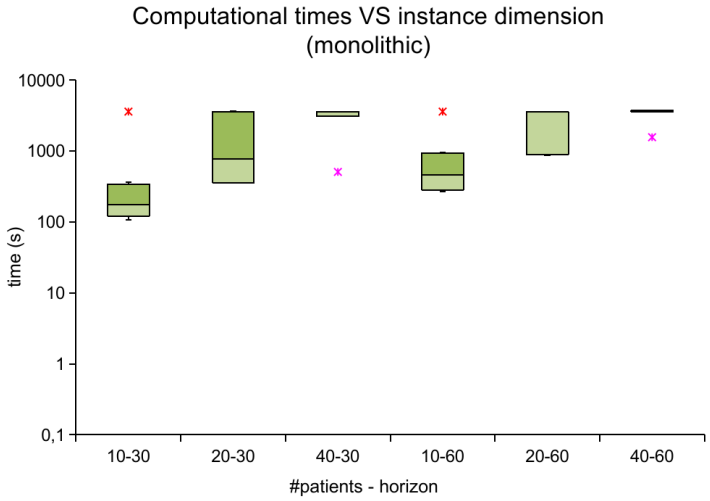

Fig. 4: Distribution of computation times for each group of instances - monolithic approach

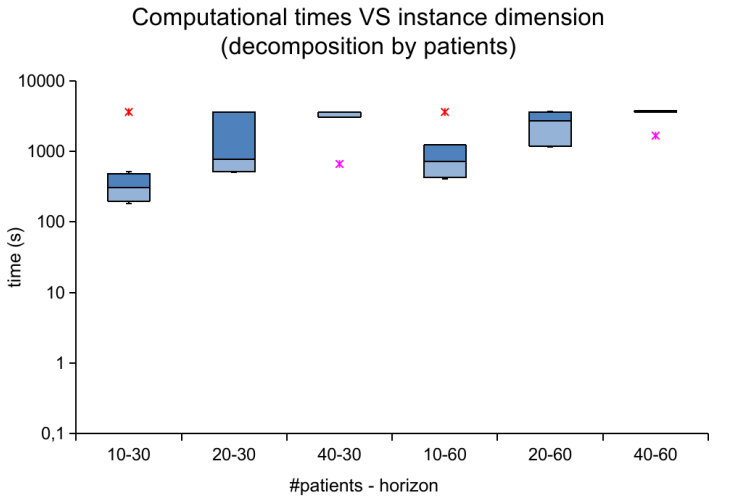

Fig. 5: Distribution of computational times for each group of instances - Patient Priority Based approach

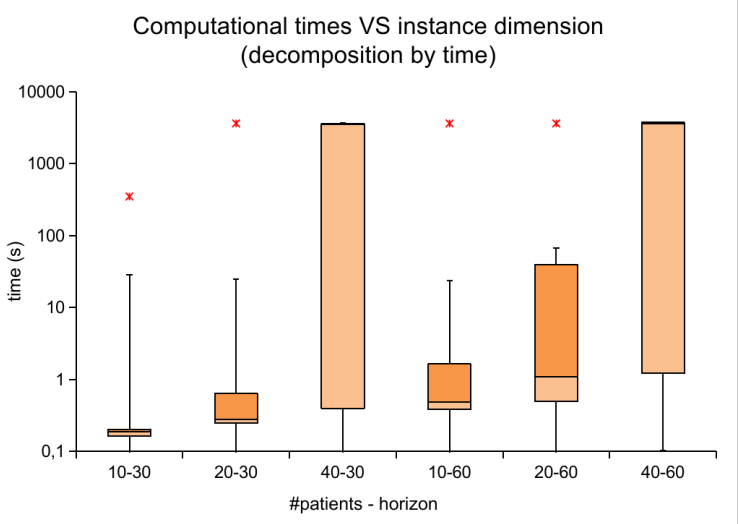

Fig. 6: Distribution of computational times for each group of instances - Time Granularity Based approach

8 Conclusions and on going work

In a historical moment such as the one we are living, exploiting in the best way the limited resources available at public health care facilities is of paramount importance. It is equally important to ensure uniformity of care across the country and among patients. It therefore becomes essential to deploy their clinical paths in order to pursue these objectives. To this aim, the current project could be easily extended to encompass the arrival of new patients who get enrolled in a care pathway at a time when the agenda has already been planned. To this purpose, a similar problem to the one here discussed can be solved, in which, on the one hand, the appointment date of the scheduled services is allowed to vary within a very small time window, such as a ± 1 variation, with respect to the previously scheduled date; on the other hand, the number of scheduled packets of the new patients are maximized and the number of packets affected by the date shift is minimized. The trade off between these two criteria should be carefully calibrated in order to devise the appropriate weight to be used in a weighted objective function. We expect such rescheduling problem to be simpler to solve with respect to the whole NCD Agenda, as each appointment date can only be shifted of a small time window, so the rescheduling problem might be approachable even without decomposition (although there could be stricter requirements of quick responsiveness). We will delve into this subject in future work.

A similar approach allows to handle periodical rescheduling, as time goes by, in a rolling horizon framework, considering a partial overlap of successive periods. Towards the end of one planning period, the requests of the upcoming period get known. These are scheduled, together with the ones in the overlapping part of the current planning period, allowing for minor date shifts of the latter ones.

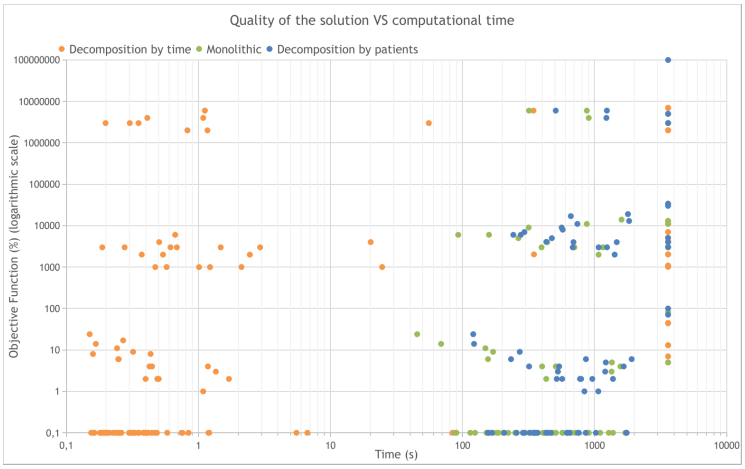

Fig. 7: Each dot corresponds to an instance. The value reported on the x-axis represents the computational time required to solve the instance, while on the y-axis the quality of the solution is given in terms of percentage of unscheduled health services in each patient priority class - high, medium and low. Specifically, the three percentages are converted in a 9-digit number, three digits for each priority class: starting from the less significant digits, the three blocks - of three digits each, represent respectively the percentage of unscheduled low-priority, medium-priority, and high-priority services. The conversion reflects the hierarchical objective function that minimizes the unscheduled services separately for each priority class. For example, if the 3 percentages were respectively 100 for the highest priority, 74 for the medium, and 82 for the lowest one, the value shown on the y-axis would be 100,074,082.

A further extension concerns taking into consideration fairness among patients. To this aim, one possibility would be to minimize, for each priority class, the maximum number of unscheduled packets of each patient, in order to avoid to penalize specific individuals for the benefit of the others. On the other hand, such an approach could suffer from well know effects (such as after the maximum is minimized, the unscheduled packets of the other patients do not affect the objective function any longer, and all solutions having the same maximum are considered equally good) and could slow down the solution process. Therefore, other kinds of fairness representation should be investigated. For example, in the Constraint Programming research area one idea is minimizing the (linear or quadratic) deviations from the average. This will be the subject of a further study.
In this paper, we tailored classical ideas about decomposition to the specific features of the Agenda problem. In particular, in the Time Granularity Based approach, the master problem is enriched with a relaxed version of the subproblems. In the Patient Priority Based approach, we propose an enhanced variant of the well known input based decomposition, which computes the starting time of a service as a proof of feasibility of the current master plan, but may later backtrack on such a decision in favour of a better solution.
The experimental results described in this paper show that decomposing the problem proves crucial in order to respond to these needs in a short computation time up to about 600 health services. This seems particularly true for the Time Granularity Based decomposition. However, as the problem size increases, even for the Time Granularity Based approach, the number of instances terminated within the time limit tends to decrease rapidly. Thus, interesting lines of future research emerge with the goal of solving problems of even larger size. Specifically, one possibility might be to hybridize the two proposed decomposition approaches, namely Patient Priority Based and Time Granularity Based, so that the strengths of each one can be exploited. Another promising line of research involves enriching the information that the subproblems send back to the master problem in a logic-based Benders decomposition framework. The Time Granularity Based in particular, paves the way towards this approach, because of the hierarchical relationship between the two sets of variables involved in Time Granularity Based, i.e., the day a packet is delivered, if any, and the time of the day. Finally, it is extremely challenging to be able, for example through machine learning techniques, to understand what are the peculiarities of the instances that make one approach preferable to another. This will put in the hands of decision makers a set of tools from which to choose the one that best fits the particular setting in which the provider operates.

References

1. Aspland, E., Gartner, D., Harper, P.: Clinical pathway modelling: a literature review. Health Systems **10**(1): 1-23 (2021)
2. 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 Pract. Log. Program. **20**(2), 294–309 (2020)
3. Cappanera, P., Gavanelli, M., Nonato, M., Roma, M.: A Decomposition Approach to the Clinical Pathway Deployment for Chronic Outpatients with Comorbidities. AIRO Springer Series **8**, 213–226 (2022)

4. Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T.: Multi-shot ASP solving with clingo. Theory Pract. Log. Program. **19**(1): 27-82 (2019)
5. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In Int. Conf. Logic Programming, pp. 1070-1080, (1998)
6. Jauden, T., Kaminski, R., Ostrowski, Schellhorn, S., Wanko, P., Schaub, T.: Clingo goes linear constraints over reals and integers. Theory Pract. Log. Program. **17**(5-6), 872–888 (2017)
7. Guido, R., Ielpa, G., Conforti, D.: Scheduling outpatient day service operations for rheumatology diseases. Flex. Serv. Manuf. J. **32**: 102-128 (2020)
8. Lloyd, J.W.: Foundations of Logic Programming. Springer (1987)
9. Marynissen, J., Demeulemeester, E.: Literature review on multi-appointment scheduling problems in hospitals. Eur. J. Oper. Res. **272**(2): 407-419 (2019)
10. Yu, S., Kulkarni, V.G., Deshpande, V.: Appointment Scheduling for a Health Care Facility with Series Patients. Prod Oper Manag **29**(2): 388-409 (2020)
11. https://salute.regione.emilia-romagna.it/cure-primarie/diabete/gestione-integrata-del-diabete-mellito-di-tipo-2-2017 [last accessed 7.7.2021]
12. https://www.regione.toscana.it/documents/10180/23793180/ALL+A+23-2019+PDTA-Diabete.pdf/f1e8ea87-145f-08c4-6c3d-16b69f5f43c2?t=1578658143393 [last accessed 7.7.2021]
13. Alviano, M., Dodaro, C., & Maratea, M. Nurse (re) scheduling via answer set programming. *Intelligenza Artificiale 12*, 2, 109–124, (2018).
14. Dodaro, C., & Maratea, M. Nurse scheduling via answer set programming, *International Conference on Logic Programming and Nonmonotonic Reasoning*, 301–307, Springer (2017)
15. Alviano, M., Bertolucci, R., Cardellini, M., Dodaro, C., Galatà, G., Kamran Khan, M., Maratea, M., Mochi, M., Morozan, V., Porro, I., Schouten, M. (2020). Answer Set Programming in Healthcare: Extended Overview. In *IPS-RCRA@ AI* IA 2020*
16. Caruso, S., Galatà, G., Maratea, M., Mochi, M., Porro, I. Scheduling Pre-Operative Assessment Clinic via Answer Set Programming. CEUR Workshop Proceedings, Vol. 3065, pp. 26 - 36, 2021, IPS 2021 and RCRA 2021
17. Galatà, G., Maratea, M., Mochi, M., Morozan, V., Porro, I. An ASP-based solution to the Operating Room Scheduling with care units. CEUR Workshop Proceedings Vol 3065, RCRA 2021
18. Cardellini, M., De Nardi, P., Dodaro, C., Galatà, G., Giardini, A., Maratea, M. Porro, I. A Two-Phase ASP Encoding for Solving Rehabilitation Scheduling, in Proceedings of International Joint Conference on Rules and Reasoning, pp. 111–125, LNCS Vol. 12851, Springer (2021)
19. Dodaro, C., Galatà, G., Maratea, M., Mochi, M. I. Porro. Chemotherapy Treatment Scheduling via Answer Set Programming, CEUR Workshop CILC 2020
20. Dodaro, C., Galatà, G., Kamran Khan, M., Maratea, M. Porro, I. Operating Room (Re) Scheduling with Bed Management via ASP. Theory and Practice of Logic Programming, 22(2), 229-253 (2021)