UNIVERSITÀ DEGLI STUDI DI GENOVA
SCUOLA POLITECNICA
DIBRIS
DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA INFORMATICA - ARTIFICIAL INTELLIGENCE AND HUMAN-CENTERED COMPUTING

Anno Accademico 2019/2020

Tecniche di Intelligenza Artificiale per la risoluzione del problema di schedulazione dei trattamenti chemioterapici

Candidato
Marco Mochi

Relatore
Prof. Marco Maratea

Correlatore
Dott. Giuseppe Galatà

Sommario

La pianificazione delle sedute chemioterapiche nelle cliniche oncologiche è un problema complesso per via dei vincoli che la soluzione al problema deve rispettare come: la natura ciclica dei trattamenti, la disponibilità delle risorse come farmaci e infermieri. Allo stesso tempo ottenere una soluzione soddisfacente è di massima importanza per poter assicurare delle buone cure ai pazienti.
Un altro problema importante da affrontare é quello della ripianificazione, che deve essere effettuata per poter modificare le cure ai pazienti che stanno rispondendo male ad un determinato trattamento o per ovviare ad indisponibilità di un paziente.
Nella tesi presentiamo una soluzione ai problemi di pianificazione e ripianificazione utilizzando uno strumento di Intelligenza Artificiale basato sulla programmazione dichiarativa, l'Answer Set Programming (ASP).
In seguito alla presentazione della nostra soluzione analizziamo i risultati ottenuti prendendo in considerazione diverse quantità di pazienti e diverse disponibilità di risorse.
I risultati ottenuti ci permettono di dire che la nostra soluzione si è rivelata buona e possiamo quindi dire che l'ASP è uno strumento efficace per affrontare il problema della pianificazione e ripianificazione delle sedute chemioterapiche.

Abstract

The problem of planning and scheduling chemotherapy treatments in oncology clinics is a complex problem, given that the solution has to satisfy several requirements such as the cyclic nature of chemotherapy treatment plans, and the availability of resources, e.g. nurses and pharmacy quantities. At the same time, realizing a satisfying schedule is of upmost importance for obtaining the best health outcomes.

To allow to shift an appointment of a patient on a different day due to poor response to a treatment or because of the availability of the patient is important to study the problem of rescheduling too.

In this thesis, we present a solution to the problem of scheduling and rescheduling using an Artificial Intelligence tool based on declarative programming, the Answer Set Programming (ASP).

Having presented our solution we analyze the results obtained by testing it on different benchmarks, with vary number of patients and drug availability.

The results of the experimental analysis, show that ASP is a suitable solving methodology for this scheduling and rescheduling problem.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo presentiamo le motivazioni che rendono il problema della pianificazione delle sedute chemioterapiche un problema interessante e reso ancora più di attualità dalle problematiche legate al COVID19.
Successivamente presentiamo brevemente lo stato dell'arte, che sarà poi affrontato in maniera più esaustiva in un capitolo successivo, elencando in particolare le varie metodologie utilizzate per affrontare il problema da noi trattato. Infine, presentiamo gli obbiettivi della tesi e la sua struttura.

1.1 Contesto e motivazioni

La pianificazione delle sedute riguardanti i trattamenti oncologici é un problema complesso per via della presenza di molte risorse e aspetti da prendere in considerazione, tra cui la disponibilità di infermieri, medicine e sedie.
I trattamenti chemioterapici sono ciclici e le tempistiche e il numero dei cicli dipende dal tipo di cancro, dallo stadio della malattia e dalla risposta del paziente. Inoltre i vari trattamenti priorità diverse, che vengono prese in considerazione quando si effettua la pianificazione.
Visto tutto ciò risulta di vitale importanza per una clinica riuscire ad ottenere una pianificazione ottimale, per aumentare il livello di soddisfazione dei pazienti e degli infermieri, e sfruttare al meglio le risorse a disposizione. Inoltre le tempistiche sono molto importanti anche per le cure dei pazienti: sono molti gli studi che hanno infatti evidenziato come i ritardi nei trattamenti abbiano un effetto malevolo. Questo effetto varia a seconda dell'aggressività e del tipo di cancro; da qui l'importanza di avere una soluzione che riesca a gestire le diverse priorità dei trattamenti.

Inoltre le problematiche relative al COVID19 rendono ancora più importante lo sfruttamento ottimale delle risorse a disposizione e la diminuzione dei tempi di attesa e tempo di permanenza nei vari reparti dell'ospedale.

1.2 Panoramica stato dell'arte

Il problema della pianificazione, sia in generale che in particolare nel contesto delle sedute chemioterapiche, è un problema che è stato affrontato utilizzando diverse tecniche.
Nella letteratura sono presenti articoli che affrontano il problema tramite programmazione a numeri interi, come il lavoro di Huggins et al. (2014) che hanno affrontato il problema cercando di ottimizzare l'utilizzo delle risorse, altri invece, come nel lavoro di Turkcan et al. (2010) utilizzano metodi euristici ed altri ancora invece utilizzano tecniche a 2 fasi per risolvere il problema, come Sevinc et al. (2013).
Sono invece poco sfruttate soluzioni che utilizzino tecniche di Intelligenza Artificiale, ad esempio soluzioni basate sulla programmazione dichiarativa e in generale metodi di rappresentazione della conoscenza, dove l'utente specifica, utilizzando linguaggi formali, le proprietà che deve avere la soluzione, e solutori software si occupano di computare tale soluzione. Queste soluzioni sono particolarmente indicate per affrontare problemi combinatori complessi, che possono comprendere anche l'ottimizzazione, come quello da noi affrontato.
Uno degli strumenti utilizzabili è l'Answer Set Programming (ASP) (Gelfond and Lifschitz (1991), Faber et al. (2011)), che è stato già utilizzato in diversi campi di ricerca.

1.3 Obbiettivi della tesi

Gli obbiettivi della tesi sono i seguenti:

- studiare il problema della pianificazione delle sedute dei trattamenti oncologici
- cercare di trovare una soluzione al problema, utilizzando l'ASP
- creare diversi test, rappresentati diversi scenari possibili
- svolgere le analisi sperimentali per analizzare il comportamento della nostra soluzione al variare degli scenari
- studiare il problema della ripianificazione nell'ambito dei trattamenti oncologici

- cercare di trovare una soluzione al problema
- effettuare le analisi dei risultati tenendo in considerazione le caratteristiche, come le tempistiche ridotte, che deve avere una buona soluzione a questo problema

1.4 Struttura della tesi

Nella tesi dopo questo capitolo introduttivo presenteremo in generale il problema e l'Input che lo caratterizza. In particolare definiremo i requisiti del modello per la pianificazione del problema, questi requisiti sono stati presi in parte da articoli inerenti la pianificazione e le sedute oncologiche e in parte sono stati assunti da noi.
Dopo aver presentato il problema più nel dettaglio presentiamo il linguaggio ASP, che utilizziamo per sviluppare la nostra soluzione, e la traduzione in questo linguaggio dell'Input e dei requisiti precedentemente mostrati.
Avendo mostrato la soluzione per il problema della pianificiazione presentiamo successivamente l'Input e i requisiti per la ripianificazione. In seguito mostriamo la traduzione in ASP dei requisiti e il modello per la ripianificazione.
Nel capitolo 5 eseguiamo un'analisi dei risultati ottenuti, mostrando l'utilizzo delle risorse e la pianificazione dei pazienti.
Per meglio presentare il problema nell'arco dei vari capitoli includiamo degli esempi che si riferiscono tutti allo stesso scenario, presentiamo quindi un Input di esempio, ed il suo corrispettivo Output e in conclusione il calendario delle sedute generato partendo da esso.

Capitolo 2

Presentazione problema

Il problema da trattare consiste nell'ottimizzare il calendario delle sedute dei pazienti oncologici, dove le sedute di questi pazienti si protraggono nel tempo seguendo uno schema che varia a seconda del trattamento necessario e prevedono somministrazioni di medicine diverse a seconda della seduta.
In questo capitolo presentiamo l'Input del problema, introducendo la tipologia di pazienti ed il numero.
Inoltre presentiamo i requisiti che si devono rispettare. Per utilizzare dei requisiti che rappresentassero più precisamente possibile le problematiche e le restrizioni delle cliniche abbiamo aggiunto dei requisiti nuovi e studiato i requisiti presenti in altri papers come: Sevinc et al. (2013), Turkcan et al. (2010), Hahn-Goldberg et al. (2014), Alviano et al. (2017), Dodaro and Maratea (2017), Alviano et al. (2018), Aringhieri et al. (2015), Heydari and Soudi (2015).
Nei capitoli successivi presenteremo invece le caratteristiche degli Input usati per effettuare i test in maniera più dettagliata.
Infine, presentiamo l'Output generato dalla nostra soluzione ed una possibile rappresentazione in un sistema che la implementi, per meglio spiegare il comportamento e i risultati ottenuti.

2.1 Input

Le istanze di Input variano per numero di pazienti e medicine ma prevedono tutte un arco temporale di 14 giorni. La soluzione permette di modificare il numero di sedie e infermieri che sono a disposizione della clinica e permette anche di cambiare il numero massimo di pazienti visitati da un infermiere in un'ora.
I trattamenti dei pazienti oncologici sono tipicamente dei trattamenti che si ripetono nel tempo secondo uno schema fissato, che può essere modificato solo dal medico a seconda del caso e dalla risposta del paziente, e che prevedono per ogni seduta un determinato quantitativo di medicinale da somministrare. Per rendere i nostri test realistici abbiamo utilizzato uno schema di trattamento reale e lo abbiamo assegnato ad ogni paziente.
Lo schema del trattamento consiste nel somministrare nella prima seduta tre diversi medicinali e successivamente per 4 giorni di fila somministrare due dei tre medicinali precedentemente somministrati, dopo queste 4 sedute consecutive vi sono 2 giorni di pausa ed infine si somministra un solo medicinale nell'ultima giornata.

Per meglio definire lo schema del trattamento presentiamo la tabella 2.1. Nella tabella sono indicate le medicine A, B e C che sono tre diversi medicinali utilizzati in un ciclo di trattamenti per pazienti oncologici. Come indicato dallo schema i tre medicinali sono somministrati rispettivamente in dosi da 20 e 100 mg/m^2 per le medicine A e B mentre in 30 unità per C. Le tre somministrazioni hanno una durata espressa da noi in finestre temporali, nel nostro caso abbiamo che le medicine A e C richiedono il tempo di una finestra temporale mentre B ne richiede 2.
Facendo un esempio un paziente che inizi ad eseguire i trattamenti il giorno 4 dovrà ricevere quel giorno tutte e tre le medicine e continuare ad andare nella clinica i giorni 5,6,7 ed 8 e ricevere in quei giorni le medicine A e B. Avrebbe successivamente due giorni di riposo per poi tornare il giorno 11 e ricevere la medicina C.

Tabella 2.1 Schema seguito da ogni paziente, ispirato da uno schema presentato da Turkcan et al. (2010)

Giorno	Medicine	Dose	Durata
1	A-B-C	20 mg/m^2, 100 mg/m^2, 30 unità	1, 2, 1 finestre temporali
2-5	A-B	20 mg/m^2, 100 mg/m^2	1, 2 finestre temporali
6-7	Riposo		
8	C	30 unità	1 finestra temporale

2.2 Requisiti

Per effettuare la pianificazione bisogna prendere in considerazione alcuni vincoli e requisiti tipici del problema della creazione del calendario dei trattamenti di pazienti oncologici e alcuni vincoli più generici riferiti alle problematiche delle clinche e l'utilizzo delle risorse:

- La prima seduta può essere pianificata ogni giorno ma rispettando la data limite indicata nella registrazione della seduta
- Le sedute successive devono essere pianificate precisamente dopo i giorni indicati nella registrazione.
- Le sedie devono essere utilizzate da una sola persona ogni slot di tempo e i pazienti che rimangono per più ore devono rimanere nella stessa sedia.
- Gli infermieri possono supportare da 1 ad n pazienti ogni ora a seconda di quale rapporto pazienti: infermiere si vuole mantenere.
- Ogni giorno non si deve superare una quantità fissata (la scorta) di ogni medicinale.
- Visto che alcuni medicinali possono prevedere una preparazione che può richiedere un pò di tempo abbiamo previsto un vincolo che limita l'inizio di una visita un'ora prima rispetto all'ultimo slot disponibile.
- Si prevede un limite di 10 sedute ogni giorno per ogni sedia.
- Visto che si devono pianificare tutti i pazienti il prima possibile, non è possibile che dopo un giorno senza registrazioni si visiti un paziente.

2.3 Output

Come Output si vuole ottenere il calendario delle sedute, con indicato l'orario di inizio visita per ogni paziente e la sedia che occupa. Inoltre si indica quale infermiere dovrà occuparsi del paziente in una determinata ora.
Oltre ad avere indicazioni riguardo all'organizzazione delle sedute, l'Output ci fornisce pure le informazioni riguardanti l'utilizzo delle varie risorse, così da permetterci di fare analisi sull'utilizzo di esse.
Presentiamo un calendario delle sedute nella figura 2.1 costruito partendo dal risultato ottenuto con un problema con 20 pazienti.

Figura 2.1 Calendario rappresentante i pazienti pianificati nei vari giorni, blu: Paz. con priorità 1, arancione: Paz. con priorità 2, verde: Paz. con priorità 3.

Il calendario è stato creato con un tool online (https://plan.toggl.com/), per far vedere come effettivamente sarebbero distribuiti i pazienti.
Da questo esempio si vede come in un utilizzo reale la soluzione del nostro pianificatore tenda ad effettuare tutte le visite i primi giorni (come sarebbe auspicato) e man mano che i giorni avanzano le visite si fanno sempre più sporadiche.
Questo comportamento è accentuato maggiormente in questo caso in cui ci sono 20 pazienti, dove gli ultimi giorni non ci sono proprio visite, ma è presente anche con piú pazienti. Da questo si capisce che in un utilizzo reale della nostra soluzione si dovrebbero pinificare i gruppi di pazienti sempre su 14 giorni ma con i primi 7 giorni già parzialmente pianificati precedentemente. L'Output ottenuto dovrá quindi essere utilizzato successivamente come Input e deve quindi essere strutturato in modo adeguato.

Capitolo 3

Pianificazione

In questa sezione si presenta la nostra soluzione, sviluppata in linguaggio ASP (Gelfond and Lifschitz (1988), Gelfond and Lifschitz (1991)), che permette di assegnare le date delle varie sedute ai pazienti. Per ottenere dei risultati soddisfacenti, ma al tempo stesso utilizzabili in un contesto reale, si è deciso di limitare la ricerca dell'ottimo per un tempo di 300 secondi.
Il modello assegna una data e un orario ad ogni paziente per ogni seduta, indicandone anche l'ID della sedia che dovrà utilizzare e l'infermiere che dovrà provvedere alla somministrazione dei farmaci.
La nostra soluzione è abbastanza flessibile da poter permettere la modifica del numero di infermieri e sedie, così da poter essere utilizzato da cliniche di varie dimensioni, e permette inoltre di modificare il numero di pazienti che un infermiere puó visitare ogni ora, cosí da poter assegnare il carico di lavoro deciso dalla clinica.
Prima di presentare in dettaglio le regole del modello e la struttura dell'Input presentiamo il linguaggio ASP e la sua sintassi e semantica.
Successivamente presentiamo l'Input, presentando come sono strutturati gli atomi in Input e analizzando i valori reali utilizzati per ottenere il calendario delle sedute mostrato nel capitolo 2, infine presentiamo le regole del nostro modello.

3.1 Linguaggio ASP

Answer Set Programming (ASP) è un paradigma di programmazione svillupato nel campo della programmazione logica e del reasoning. Nel fare questa panoramica assumiamo che le convenzioni della programmazione logica siano note, più informazioni riguardo ad ASP possono trovarsi a https://potassco.org/.

Sintassi. La sintassi di ASP è simile a quella di Prolog. Le variabili sono stringhe con la prima lettera maiuscola e le costanti sono interi non negativi o stringhe con la prima lettera minuscola. Un termine è o una variabile o una costante.
Un *termine* è una variabile o una costante. Un *atomo standard* è un'espressione $p(t_1,\ldots,t_n)$, dove p è un *predicato* con n argomenti e $t_1,\ldots,t_n$ sono i termini. Un atomo $p(t_1,\ldots,t_n)$ è ground se $t_1,\ldots,t_n$ sono costanti. Un *set ground* è un set di coppie nella forma $\langle consts : conj\rangle$, dove $consts$ è una lista di costanti e $conj$ è una congiunzione di atomi ground. Un *set di simboli* è un set definito sintatticamente come $\{Terms_1 : Conj_1;\cdots;Terms_t : Conj_t\}$, dove $t > 0$, e per tutte le $i \in [1,t]$, ogni $Terms_i$ è una lista di termini tale che $|Terms_i| = k > 0$, e ogni $Conj_i$ è una congiunzione di atomi standard. Un *set di termini* è o un ground set o un set di simboli. Intuitivamente, un set di termini $\{X : a(X,c), p(X); Y : b(Y,m)\}$ rappresenta l'unione di due set: il primo contiene i valori X che rendono la congiunzione $a(X,c), p(X)$ vera, e il secondo contiene i valori Y che rendono la congiunzione $b(Y,m)$ vera. Una *funzione aggregato* è della forma $f(S)$, dove S è un set di termini, e f è un *simbolo di una funzione aggregato*.
Le funzioni aggregato mappano vari set di costanti su una costante. Le più comuni funzioni implementante nei sistemi ASP sono le seguenti:

- *#count*, numero di termini.
- *#sum*, somma di interi.
- *#max*, valore massimo tra interi.

Un *atomo aggregato* è della forma $f(S) \prec T$, dove $f(S)$ è una funzione aggregato, $\prec \in \{<, \leq, >, \geq, \neq, =\}$ è un operatore di confronto, e T è un termine chimato guardia. Un atomo aggregato $f(S) \prec T$ è ground se T è una costante e S è un set ground. Un *atomo* è un atomo standard o un atomo aggregato. Una *regola r* ha la seguente struttura:

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

dove $a_1,\ldots,a_n$ sono atomi standard, $b_1,\ldots,b_k$ sono atomi, $b_{k+1},\ldots,b_m$ sono atomi standard, e $n,k,m \geq 0$. Un literal è o un atomo standard a o la sua negazione *not* a. La disgiunzione $a_1 \vee \ldots \vee a_n$ è la *testa* di r, mentre la congiunzione $b_1,\ldots,b_k,not\ b_{k+1},\ \ldots,not\ b_m$ è il suo *corpo*. Regole senza corpo sono dette *fatti*. Regole senza testa sono dette *constraints*. Una variabile che compare solo nei termini di una regola r è detta *locale* in r, altrimenti è una variabile *globale* di r. Un programma ASP è un insieme di regole *safe*, dove una regola r è *safe* se rispetta le seguenti condizioni: *(i)* per ogni variabile globale X di r c'è un atomo standard ℓ nel corpo di r tale che X appare in ℓ; e *(ii)* ogni variabile locale di r che appare in un set di simboli $\{Terms : Conj\}$ appare anche in un atomo positivo in $Conj$.
Un *weak constraint* (Buccafurri et al. (2000)) ω è nella forma:

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

dove w e l sono rispettivamente il peso e il livello ω. (Intuitivamente, $[w@l]$ si legge "come peso w al livello l", dove il peso è il "costo" pagato al violare la condizione nel corpo di w, mentre il livello puó essere usato per indicare una diversa priorità). Un programma ASP con weak costraints è $\Pi = \langle P,W\rangle$, dove P è un programma e W è un set di weak costraints.
Un atomo standard, un literal, una regola, un programma o un weak costraints è *ground* se non vi appaiono variabili in esso.

Esempio di regola e constraint
Per presentare un esempio di una regola immaginiamo di dover creare un modello che ci permetta di asserire chi è padre tra un gruppo di persone:
$padre(P)$:- $uomo(P), figlio(M,P)$. Dove figlio(M,P) è vero quando M è figlio della persona P.
In questo modo l'atomo padre(P) sarà vero per tutte le persone che compaiono nell'atomo figlio e che rendono vero l'atomo uomo.
Per presentare i contraints immaginiamo di dover creare un modello riguardante gli scacchi e dover esprimere che non vi siano due pedine nella stessa casella X, Y:
:- $posizione(X,Y,N), posizione(X,Y,M), N! = M$. Dove posizione(X,Y,T) è vera quando la pedina generica T è nella casella identificata da X, Y. Con il constraint imponiamo che almeno uno dei tre atomi sia falso. Quindi nel nostro esempio o almeno una tra le pedine N ed M non sono in X, Y o N ed M hanno lo stesso valore e rappresentano quindi la stessa pedina.

Esempio di funzioni aggregato

Consideriamo di voler calcolare il numero di padri presenti tra un gruppo di persone, dopo aver utilizzato la regola presentata nell'esempio precedente:
$padre(P)$:- $uomo(P), figlio(M,P)$.
Per ottenere il numero di padri presenti possiamo usare la funzione aggregato #count:
$numPadri(N)$:- $N = \#count\{P : padre(P)\}$. In questo modo N sarebbe uguale al numero di atomi padre veri.
Volessimo invece contare la somma dei crediti di alcuni esami, espressi con l'atomo esame(id, crediti), potremmo usare #sum:
$totCrediti(N)$:- $N = \#sum\{N,I : esame(I,N)\}$. Così si sommano tutti i valori della N presenti negli atomi esame. Avessimo invece un gruppo di persone con atomo persona(nome, età) e volessimo trovare l'età della persona più anziana potremmo usare la funzione aggregato #max:
etàMaggiore(N) :- $N = \#max\{E : persona(M,E)\}$. In questo modo N sarebbe uguale all'età maggiore, perchè max controlla tutti gli atomi persona e prende il valore della E maggiore.

Semantica. Sia P un programma ASP. L' *Herbrand universe* U_P e la *Herbrand base* B_P di P sono definite come sempre. L'istanza ground G_P di P è l'insieme di tutte le istanze ground delle regole di P che possono essere ottenute sostituendo le variabili con le costanti da U_P.
Un' *interpretazione* I per P è il sottoinsieme I di B_P. Un literal ground ℓ (risp., *not* ℓ) è vero rispetto ad I se $\ell \in I$ (risp., $\ell \notin I$), e falso (risp., vero) altrimenti. Un atomo aggregato è vero rispetto ad I se il valore della funzione aggregato (i.e., il risultato dell'applicazione di f negli insiemi S) rispetto ad I rispetta la guardia; altrimenti è falsa.
Una regola ground r è *soddisfatta* da I se almeno un atomo nella testa è vero rispetto ad I mentre tutte le congiunzioni del corpo di r sono vere rispetto ad I.
Un modello è un'interpretazione che risolve tutte le regole di un programma. Dato un programma ground G_P e un'interpretazione I, la *ridotta* Faber et al. (2011) di G_P rispetto ad I è il sottoinsieme G_P^I di G_P ottenuto eliminando da G_P le regole in cui un literl nel corpo è falso rispetto ad I. Un'interpretazione I per P è un *answer set* (o modello stabile) per P se I è un modello minimo di G_P^I (i.e., I è un modello minimo G_P^I) Faber et al. (2011).
Dato un programma con weak costraints $\Pi = \langle P, W \rangle$, la semantica di Π amplia quella del caso base spiegato precedentemente. Sia $G_\Pi = \langle G_P, G_W \rangle$ l'istanza di Π; un constraint $\omega \in G_W$ è violato da un'intepretazione di I se tutti i literal in ω sono veri rispetto ad I. Un *optimum*

answer set per Π è un answer set di G_P che minimizza la somma dei pesi dei weak constraints violati in G_W con ordine prioritario.

Abbreviazione sintattica. Useremo successivamente anche le *choice rules* nella forma $\{p\}$, dove p è un atomo. Le choice rules possono essere interpretate come un'abbreviazione sintattica per la regola $p \vee p'$, dove p' è un atomo nuovo che non appare in altre parti del programma, questo significa che p può essere scelto come vero.
Nel nostro modello utilizziamo #minimize$\{p : x(p)\}$, la utilizziamo per via della sua maggiore leggibilità e serve per minimizzare la somma di tutti i valori p che rendono vera x(p), è quindi interpretabile come un weak constraint. Come le weak costraints si può assegnare a #minimize dei livelli diversi, ottenendo per esempio, volendo assegnare un livello pari a 2: #minimize$\{p@2 : x(p)\}$

Esempio Guess and Check e choice rules.
Consideriamo di avere un grafo e doverne colorare i nodi in modo tale da non assegnare a due nodi collegati tra loro lo stesso colore ed avendo a disposizione 3 colori. Per assegnare i colori ai vari nodi possiamo fare:
$col(X,rosso)|col(X,giallo)|col(X,verde)$:-$nodo(X)$. che rende vero solo uno dei tre atomi col e quindi questa è la fase del guess.
:- $collegati(X,Y),col(X,C),col(Y,C)$. Così si controlla che due nodi X ed Y collegati tra loro non abbiano lo stesso colore, quesa è al fase del check.
Nel nostro modello utilizziamo le choice rules che ci permettono di riscrivere in maniera alternativa il guess and check:
$1\{col(X,C) : colore(C)\}1$:- $nodo(X)$ In questo modo per ogni nodo X assegna un colore C tra quelli possibili. Poi bisogna scrivere la regola per il check.

Esempio weak constraint e minimize.
Consideriamo di dover organizzare l'orario degli esami di alcuni corsi universitari aventi studenti in comune e avendo 3 diversi slot di tempo. Il nostro obbiettivo è quello di minimizzare il numero totale di studenti in comune tra esami nello stesso orario. Per assegnare gli orari degli esami possiamo sempre usare la choice rule:
$1\{assign(X,S) : time(S)\}1$:- $course(X)$. Si assegna un orario tra quelli possibili.
Per minimizzare la somma degli studenti comuni possiamo usare la weak constraint:
: $assign(X,S),asign(Y,S),studentiComuni(X,Y,N),N > 0.[N@0]$Gli studenti in comune sono indicati dal valore N, che viene sommato per tutte le coppie di corsi X ed Y, e viene mi-

nimizzato grazie al weak constraint. Se fosse un vincolo normale questo vincolo imporrebbe che non ci siano studenti comuni e non restituirebbe una soluzione nel caso in cui questo non fosse possbile, mentre il weak constraint, nel caso in cui questo vincolo non potesse essere rispettato, permette di ottenere una soluzione che minimizza la somma di N fino a portarlo a 0 nel caso in cui il vincolo fosse rispettato.
In alternativa ai weak constraints si possono utilizzare le funzioni minimize e maximize. Nel nostro modello utilizziamo la funzione minimize, che serve a minimizzare la somma di un valore ed è più intuitiva nel funzionamento rispetto ai weak constraints.
Nel caso precedente avremmo potuto minimizzare la somma di N cosí:
$\#minimize\{N@1 : studentiComuni(X,Y,N)\}$

3.2 Codifica ASP

In questa parte del capitolo presentiamo la struttura degli atomi che formano l'Input con alcuni esempi per vedere come si presentano in ASP e la strutttura dell'Ouput che si genera. Oltre a presentare l'Input aggiungiamo alcune immagini dell'Input reale del nostro problema, ed in particolare l'Input del problema che genera l'Output presentato con il calendario nel capitolo 2, così da presentare tutte le componenti che formano il problema.
Infine, presentiamo le regole presenti nel nostro modello che servono a rispettare i requisiti del problema, presentati nel capitolo 2, e ad assegnare le date del calendario ai vari pazienti. Per le regole sfruttiamo alcune funzioni presentate nel paragrafo di presentazione del linguaggio ASP come le funzioni aggregate e le choice rules.

3.2.1 Input

In questa sezione presentiamo gli atomi che formano l'Input:

- Istanze di reg(R,P,OT,DD,T1,T2,T3) rappresentano le registrazioni caratterizzate da un id registrazione del paziente(R), una priorità(P), un numero che ha un valore compreso tra 0 a 5(OT) che identifica l'ordine da seguire tra le sedute della stessa persona, un numero (DD) che se OT è uguale a 0 rappresenta la data massima entro la quale effettuare i trattamenti mentre se OT ha un valore tra 1 e 5 rappresenta la distanza (DD) in giorni che deve esserci dalla seduta precedente e gli id dei trattamenti richiesti (T1, T2, T3). Sotto mostriamo le registrazione delle sedute del paziente 0, l'Input sarà formata dall'insieme di tutte le sedute di tutti i pazienti.

```
reg(0,2,0,7,3,1,2). reg(0,0,1,1,3,1,0). reg(0,0,2,1,3,1,0).
   reg(0,0,3,1,3,1,0). reg(0,0,4,1,3,1,0). reg(0,0,5,3,3,0,0).
```

- Istanze di type(T,DOSE,NCH,NN,D) rappresentano le diverse tipologie di trattamento avente ognuna un id(T), una dose richiesta da somministrare(DOSE), il numero di sedie(NCH) e infermiere(NN) necessario e la durata del trattamento(D).

```
type(1,20,1,1,1). type(2,100,1,1,2). type(3,30,1,1,1). type(0,0,0,0,0).
```

- Istanze di mss(D,H) rappresentano i vari slot di tempo disponibili nei vari giorni(D) e nelle ore(H). L'atomo presente nel nostro Input crea tanti atomi mss con tutte le possibili coppie di numeri formati da un numero tra 1 e 14 e uno tra 1 ed 8.

```
mss(1..14,1..8).
```

- Istanze chair(ID) e nurse(D,ID) che rappresentano tutte le sedie e gli infermieri disponibili. Nell'Input ci sono due atomi che creano le risorse di sedie ed infermieri con id da 1 a c e da 1 ad n rispettivamente per sedie ed infermieri.

```
chair(1..c). nurse(1..n).
```

3.2.2 Output

- Gli appuntamenti sono registrati in x(R,D,H,T1,T2,T3,P,OT) e per ognuna di queste si associa res(R,D,H,NCH,NN) per ogni ora in cui il paziente ID richiede le risorse NC e NN. Le sedie e le infermiere sono identificate da chair(ID,R,D,H) con id della sedia(ID) e da nurses(ID,R,D,H) con id infermiere(ID).

Come detto nel Capitolo 2 l'Output ottenuto non deve essere solo utilizzato per avere le date e gli orari delle sedute dei pazienti, ma anche per essere utilizzato come Input per la pianificazione successiva.
Infatti, in un contesto reale si dovrebbe effettuare la pianificazione su 14 giorni ma questi 14 giorni condividerebbero la prima settimana con la seconda settimanda della pianificazione precedente. Questo perchè andando avanti nei giorni finiscono i pazienti da visitare e i trattamenti stessi dei pazienti si fanno via via meno frequenti e richiedono meno risorse,

questo comporta un utilizzo scarso delle seconde settimane, che rimangono con vari slot disponibili.

3.2.3 Encoding

Adesso presentiamo le regole in linguaggio ASP con una piccola spiegazione della regola.

Per tutte le registrazioni delle prime sedute si assegna uno slot tra quelli esistenti tramite la metodologia del Guess and Check, prova ad assegnare un giorno ed un orario:

```
1 1 {x(R,D,H,T1,T2,T3,P,0) : mss(D,H), D <= DD} 1 :- reg(R, P, 0, DD, T1, T2,
    T3).
```

Questa regola viene utilizzata solo per le prime sedute ed associa per ogni reg, che fornisce le informazione riguardante il codice di un paziente e le medicine che deve prendere, un giorno e un orario tra quelli permessi, che sono rappresentati da mss(D,H). Una volta assegnato un giorno ed un orario viene formato l'atomo x con le informazioni presenti in reg ed in più le informazioni riguardanti il giorno e l'ora assegnate.

Per ogni prima seduta si pianificano le sedute successive avendo giorno fisso ed uguale alla somma tra la giornata della seduta precedente più la distanza che è indicata nella registrazione della seduta successiva:

```
2 1 {x(R,D1+D2,H,T1,T2,T3,0,M) : mss(D1+D2,H)} 1 :-
    x(R,D1,_,_,_,_,_,N),reg(R,_,M,D2,T1,T2,T3),M=N+1,mss(D1+D2,_).
```

Successivamente ad aver assegnato una seduta si procede, per ogni registrazione che fa riferimento alla seduta successiva dello stesso paziente, ad assegnare la seduta successiva, questa volta però la giornata non è libera, ma deve rispettare le tempistiche indicate dalla registrazione, per questo vi è mss(D1 + D2, H), dove D1 + D2 rappresentano il giorno in cui è stata fatta la visita e D2 il tempo in giorni che deve trascorrere prima della seduta successiva che si sta pianificando. L'orario rimane invece scelto liberamente.

Salva per ogni ora necessaria le risorse richieste per il primo trattamento:

```
3 res(R,D,H..H+D1-1,NCH,NN):- x(R,D,H,T1,_,_,_,_),type(T1,_,NCH,NN,D1).
```

Una volta assegnato il giorno e l'ora ed aver creato l'atomo x, si procede a creare l'atomo res. Res serve per tener traccia del tempo necessario per effettuare tutti i trattamenti del paziente e del numero di infermieri e sedie di cui ha bisogno. In questo caso si registra in res il tempo e le richieste del primo medicinale. Le risorse richieste dal primo medicinale vengono prese da type, che viene identificata con T1, e fornisce la durata del trattamento, il numero di sedie richieste e il numero di infermieri richiesti.

Salva per ogni ora necessaria le risorse richieste per il secondo trattamento:

```
4 res(R,D,H+D1..H+D1+D2-1,NCH,NN):-
    x(R,D,H,T1,T2,_,_,_),type(T1,_,_,_,D1),type(T2,_,NCH,NN,D2).
```

In questo caso registriamo il tempo e le richieste del secondo medicinale, infatti in res, come orario iniziale di queste richieste, è indicata l'ora di inizio dei trattamenti più il tempo richiesto al primo medicinale, così che sia indicata ad ogni ora la richiesta corretta di risorse del paziente.

Salva per ogni ora necessaria le risorse richieste per il terzo trattamento:

```
5 res(R,D,H+D1+D2..H+D1+D2+D3-1,NCH,NN):-x(R,D,H,T1,T2,_,_,_),
    type(T1,_,_,_,D1),type(T2,_,_,_,D2),type(T3,_,NCH,NN,D3).
```

In questo caso registriamo il tempo e le richieste del terzo medicinale. Così come è stato fatto nella regola precedente l'orario di inizio delle richieste associate al terzo trattamento è uguale all'ora in cui inizia il trattamento più la durata del primo e del secondo trattamento. Nel caso in cui il terzo trattamento non fosse richiesto questo non sarebbe un problema perchè T3 sarebbe uguale a 0 e troverebbe il type corrispondente che ha richieste di risorse e durata nulli.

Assegna le sedie e gli infermieri per ogni ora, questi vincoli mantengono la soluzione flessibile permettendo di gestire richieste di più infermieri per un paziente:

```
6 {chair(ID,R,D,H) : chair(ID)} = NCH :- res(R,D,H,NCH,_).
7 {nurses(ID,R,D,H) : nurse(ID)} = NN :- res(R,D,H,_,NN).
```

Una volta finiti di creare gli atomi res essi vengono utilizzati per assegnare ad ogni paziente, per ogni ora e giorno, la giusta quantità di sedie ed infermieri. Sia le sedie che gli infermieri sono identificati da un ID e vengono scelti casualmente tra quelli disponibili.

Ogni sedia ad una sola persona e ogni infermiere non può fare più di r persone:

```
8 :- #count{R: chair(ID,R,D,H)} > 1,chair(ID),mss(D,H).
9 :- #count{R: nurses(ID,R,D,H)} > r,nurse(D,ID),mss(D,H).
```

Siccome gli infermieri e le sedie vengono assegnate ai vari pazienti in modo casuale dalla regola precedente, per rispettare il requisito che prevede che non ci siano più pazienti per una sedia ogni ora e che un infermiere non visiti più di r pazienti, queste due regole si assicurano che per ogni giorno ed ora la somma di pazienti che sono assegnate ad una data sedia e un dato infermiere non siano superiori rispettivamente ad 1 e ad r. Lo fa mettendo la disuguaglianza col segno maggiore, perchè essendo un vincolo per essere rispettato la disuguaglianza deve essere falsa.

Si mantiene sempre la stessa sedia per persone che rimangono più ore, quindi si controlla che l'id della sedia assegnata allo stesso paziente per due ore consecutive sia lo stesso:

```
10 :- chair(ID1,R,D,H),chair(ID2,R,D,H + 1), ID1 < ID2.
```

Visto che non vogliamo che un paziente debba muoversi tra una sedia e l'altra durante i trattamenti si confrontano due atomi chair che si riferiscono alla stessa persona e allo stesso giorno ed in due ore consecutive. Gli ID delle sedie indicati da questi due atomi devono essere diversi uguali,quindi diciamo che ID1 deve essere minore di ID2, questo è un modo alternativo di dire che devono essere diversi e lo utilizziamo perchè rende il nostro modello più veloce.

Si controlla che il numero delle sedie utilizzate ogni ora non sia superiore a quello disponibie:

```
11 :- #sum{NCH,R: res(R,D,H,NCH,_)} > c,mss(D,H).
```

Queste due regole controllano che il numero di sedie ogni ora non siano superiori a quelli a disposizione. Per farlo si utilizza la funzione sum, che somma i valori di un dato valore di un atomo indicato, in questo caso somma tutti gli NCH presenti in res.

Trattamenti non possono andare oltre l'orario consentito, visto che si assegna solo l'ora di inizio della visita si deve controllare che il proseguio della seduta non vada oltre all'orario disponibile:

```
12 :- H1 = #max{HOUR1 : res(_,D,HOUR1,_,_)}, H2 = #max{HOUR2 : mss(D,HOUR2)},H1
      > H2.
```

Siccome ogni paziente deve rimanere un tempo variabile a seconda dei medicinali richiesti, si deve controllare che l'ora di arrivo del paziente più la durata totale della sua seduta non vadano oltre l'orario di chiusura della clinica. Per farlo otteniamo con le funzioni aggregato #max l'ora maggiore indicata in res in un giorno e l'ora massima tra quelle disponibili nel mss, una volta ottenute H1 e H2 le imponiamo una maggiore dell'altra, che come nella regola precedente ci serve a dire che devono essere diverse.

Non più di n trattamenti per ogni sedia

```
13 :- #count{R: chair(ID,R,D,_)} > ntreat,mss(D,_),chair(ID).
```

Per ogni sedia identificata da un ID e per ogni giorno tra quelli che consideriamo, contiamo quanti pazienti usano quella sedia e imponiamo che sia maggiore di un valore modificabile a seconda della clinica.

Inizio visite non oltre un certo orario:

```
14 :- H = #max{HOUR : mss(D,HOUR)},x(_,D,H,_,_,_,_,_).
```

Si prende il valore massimo dell'orario disponibile nel mss e si vede se esiste un atomo x con quel valore, per quello che è il funzionamento dei vincoli nel caso in cui esistesse la soluzione trovata non sarebbe valida. In questo modo imponiamo la non esistenza di una seduta che inizi nell'ultima ora disponibile.

Non si devono lasciare giorni inutilizzati se non quando si finiscono i pazienti:

```
15 :- not x(_,DAY,_,_,_,_,_,_),x(_,DAY + 1,_,_,_,_,_,_),mss(DAY,_).
```

Per ogni giorno presente nel mss controlliamo che se c'è una seduta il giorno successivo ci sia anche quel giorno stesso, lo facciamo imponendo che per ogni giorno entrambi gli atomi x siano veri.

Controllo disponibilità medicine:

```
16 :- #count{R: x(R,D,_,T,_,_,_,_); R: x(R,D,_,_,T,_,_,_); R:
     x(R,D,_,_,_,T,_,_)} > M/DOSE,mss(D,_),type(T,DOSE,_,_,_),drug(T,M)
```

Per ogni medicina T sommiamo, con la funzione count, il numero di pazienti che la richiedono e si impone che sia maggiore della quantità M che si ha nel magazzino, diviso la dose che si sommistra ad ogni seduta. Così facendo per ogni medicina perchè venga rispettato il vincolo la diseguaglianza deve essere falsa.

3.2.4 Ottimizzazione

Si minimizzano con tre pesi differenti la somma dei giorni utilizzati per le prime sedute per i pazienti con priorità 1,2,3. Si ottimizzano solo le prime perchè le successive sedute sono conseguenza della prima:

```
17 #minimize{DAY@4,REGID: x(REGID,DAY,_,_,_,_,1,0)}.
18 #minimize{DAY@3,REGID: x(REGID,DAY,_,_,_,_,2,0)}.
19 #minimize{DAY@2,REGID: x(REGID,DAY,_,_,_,_,3,0)}.
```

Per ognuna delle tre funzioni di minimizzazione si somma il valore DAY di ogni atomo x avente un REGID diverso e avente come valore degli ultimi due campi quelli indicati. In questo modo il modello modifica gli atomi x, quindi gli assegnamenti delle sedute, rispettando le altre regole, in modo tale da minimizzare queste somme.

Si minimizza la somma dei giorni utilizzati per tutte le prime sedute:

```
20 #minimize{DAY@1: x(_,DAY,_,_,_,_,_,0)}.
```

Per ogni valore DAY negli atomi x aventi come ultimo campo il valore 0, quindi sono tutte le prime sedute, viene fatta la somma e il modello cerca di minimizzarla cambiano di valori delle x. Questa regola non sarebbe necessaria per il funzionamento del pianificatore ma abbiamo visto che ci permette di ottenere risutlati migliori.

Capitolo 4

Ripianificazione

L'obbiettivo di questo capitolo è presentare l'Input e la soluzione sviluppata per il problema della ripianificazione, ma prima di presentarli spighiamo perchè questo è un problema molto importante nell'ambito ospedaliero e oncologico.
Al giorno d'oggi sono sempre di più le cure che vengono personalizzate per ogni paziente e si è orientati verso un'individualizzazione dei trattamenti, soprattuto in ambiti come quello oncologico dove a fronte di un calendario delle sedute standard il medico insieme al paziente può decidere di modificare lo schema o cambiare i dosaggi delle medicine in seguito alla risposta del paziente.
Questa modifica delle cure serve sia per poter reagire a delle cure che stanno avendo poco successo sia per rispondere ad imprevisti ed impegni che possono impossibilitare un paziente ad effettuare una seduta o la clinica ad effettuare la visita, questo perchè i pazienti, possono avere difficoltà a rispettare un appuntamento preso, sia perchè la clinica può dover sospendere l'utilizzo di alcune sedie o avere problemi improvvisi con il personale. Una soluzione efficace a queste situazioni risulta essere molto importante, non solo per aumentare il livello di utilizzo delle risorse a dispozione della clinica, ma anche e soprattutto per poter aumentare la probabilità di una guarigione dei pazienti. Molti studi (Kumar and Dey (2020); Sud et al. (2020)) infatti hanno constatato che, anche nel periodo di massima emergenza di COVID19, ritardi alle cure, sia sottoforma di interventi che di cicli di medicinali, hanno un forte impatto negativo sui tassi di sopravvivenza dei pazienti. Questo impatto dipende dalla tipologia e aggressività del tumore, da qui l'importanza di implementare un modello capace di gestire vari livelli di priorità.
Nel nostro caso il problema della ripianificazione consiste nel partire da un calendario di sedute fissato per i pazienti, avendo quindi le informazioni riguardanti giorno, ora e sedia che i vari pazienti dovranno utilizzare, e dover spostare una o più sedute di qualche giorno in avanti

o cambiando le medicine da somministrare o il loro dosaggio. Per rispettare l'importanza del mantenimento dei tempi corretti una volta iniziati i cicli delle sedute, il nostro modello permette di spostare solo le prime sedute dei pazienti per poter permettere ad altri pazienti di avere la loro seduta ripianificata e modificata.

4.1 Input

Per poter modificare il calendario delle sedute la nostra soluzione dovrà ricevere come Input il risultato ottenuto precedentemente e le informazioni riguardanti la clinica e i medicinali, quindi Input in comune con il pianificatore, più alcune informazioni necessarie per la ripianificazione:

- Il calendario delle sedute per tutti i pazienti presenti nei giorni interessati.
- Le registrazioni dei pazienti a partire dal primo giorno che dovrà essere modificato.
- Informazioni riguardo ad eventuali indisponibilità di un paziente un dato giorno.
- Una nuova registrazione per per le sedute dei pazienti che vanno modificate.

Il modello riceve quindi in ingresso tutte le registrazioni a partire dal giorno della prima seduta modificata o da spostare.
Tra le registrazioni possono essercene di due tipi nuovi: una uguale a quella del calendario originale, che quindi non presenta modifiche, ed una che modifica un trattamento, sia per dosi di medicinale e/o per giorni di attesa tra una seduta e l'altra. Inoltre si possono indicare eventuali giorni di indisponibilità di un paziente, in modo da spostare eventuali sedute assegnate in quel giorno.

4.2 Requisiti

Oltre ai requisiti in comune con il problema della pianificazione, la ripianificazione prevede nuovi requisiti:

- Le sedute successive alla prima dei pazienti che non hanno problemi di disponibilità e non hanno modifiche alla prima seduta non vanno modificate, quindi per poter far spazio ai pazienti in seguito alle modifiche possono solo essere spostate le prime sedute.
- Le sedute possono essere posticipate ma non anticipate.
- Le sedute precedenti al primo giorno di indisponibilità di un paziente non vanno spostate.
- Il calendario delle sedute non deve essere modificato prima del primo giorno di indisponibilità o modifica di una seduta.

Questi requisiti ci servono per mantenere il nuovo calendario il più simile possibile a quello originale e a rispettare i vari livelli di priorità.
Abbiamo infatti detto quanto siano problematici i ritardi dei trattamenti una volta iniziati i cicli di cure, per questo si permette al sistema di spostare, oltre ai pazienti che richiedono modifiche, solo le prime sedute dei pazienti, così da non compromettere lo svolgimento delle cure così come descritto dallo schema che devono seguire.

4.3 Output

Come Output si vuole ottenere il nuovo calendario delle sedute con le sedie e le infermiere assegnate ai vari pazienti. Per ottenere il calendario nuovo si devono rispettare i requisiti detti precedentemente e cercare di spostare i pazienti con livelli di priorità più bassi.
Nel capitolo in cui abbiamo presentato la pianificazione abbiamo anche mostrato il risultato del modello sottoforma di calendario delle sedute, adesso continuando quell'esempio consideriamo come cambierebbe l'Output in seguito alla ripianificazione.

4.4 Codifica ASP

In questa sezione del capitolo presentiamo l'Input, di cui presentiamo anche la forma in ASP, corrispondente in particolare all'istanza del problema presentato nei precedenti capitoli, da cui siamo precedentemente partiti per formare il calendario delle sedute mostrato nel capitolo 2, e presentiamo le regole, che formano il nostro modello, in particolare, utilizzando il linguaggio ASP, traduciamo i requisiti che abbiamo presentato precedentemente utilizzando in parte alcune regole già utilizzate per la pianificazione, altre modificate leggermente e altre nuove e specifiche del problema della ripianificazione.
Per tutte le regole che sono state modificate rispetto al problema della pianificazione, diamo una spiegazione che permetta di comprendere il funzionamente della regola e motivi la sua presenza nel modello. Le regole presenti, oltre ai requisiti, contengono anche le regole utilizzate per effettuare l'ottimizzazione.

4.4.1 Input in comune con la pianificazione

- Istanze di reg(R,P,OT,DD,T1,T2,T3) rappresentano le registrazioni caratterizzate da un id registrazione del paziente (R), una priorità (P), un numero che ha volore uguale a da 0 a 5 (OT) che identifica l'ordine da seguire tra le sedute della stessa persona, un numero (DD) che se OT è uguale a 0 rappresenta la data massima entro la quale effettuare i trattamenti mentre se OT ha un valore tra 1 e 5 rappresenta la distanza (DD) in giorni che deve esserci dalla seduta precedente e gli id dei trattamenti richiesti (T1, T2, T3). Includiamo sotto un esempio di alcune registrazioni corrispondenti all'istanza che stiamo usando già dai capitoli precedenti. In questo caso abbiamo queste ed altre registrazioni, ma che sono tutte successive al primo giorno da cui partire per effettuare delle modifiche nel calendario.

```
reg(0,0,4,1,3,1,0). reg(0,0,5,3,3,0,0). reg(1,0,4,1,2,1,0).
   reg(1,0,5,3,2,0,0). reg(2,0,4,1,2,3,0).
```

- Istanze di type(T,DOSE,NCH,NN,D) rappresentano le diverse tipologie di trattamento avente ognuna un id (T), una dose richiesta da somministrare (DOSE), il numero di sedie (NCH) e infermiere (NN) necessario e la durata del trattamento (D). Gli atomi type rappresentano quindi la quantità di risorse richieste e la durata necessaria per somministrare una particolare medicina.

```
type(1,20,1,1,1). type(2,100,1,1,2). type(1,30,1,1,1). type(0,0,0,0,0).
```

- Istanze di mss(D,H) rappresentano i vari slot di tempo disponibili nei vari giorni (D) e nelle ore (H). Mostriamo l'atomo presente nel nostro Input e che crea tanti atomi mss con tutte le possibili coppie di numeri formati da un numero tra 1 e 14 e uno tra 1 ed 8.

```
mss(1..14,1..8).
```

- Istanze chair(ID) e nurse(D,ID) che rappresentano tutte le sedie e gli infermieri disponibili. Con questi atomi si creano le risorse di sedie ed infermieri con id da 1 a c e da 1 ad n rispettivamente per sedie ed infermieri.

```
chair(1..c). nurse(1..n).
```

4.4.2 Input per il ripianificazione

- Istanze di x(R,D,H,T1,T2,T3,P,M) rappresentano il calendario prodotto dal modello. Mostriamo alcuni degli atomi presenti nell'Input di esempio che rappresentano ognuno una seduta diversa.

```
x(2,10,1,1,3,2,3,0). x(3,11,1,1,3,2,3,0). x(7,13,1,3,2,1,3,0).
   x(8,13,1,1,3,2,3,0).
```

- Istanze di un(R,D) che indicano l'indisponibilità del paziente (R) il giorno (D).

```
un(40,3). un(97,2). un(58,5). un(2,12). un(66,10).
```

- Istanze di regN(R,P,OT,DD,T1,T2,T3) che indicano le nuove registrazioni per modificare una seduta di un paziente. Questi sono gli atomi che indicano le nuove sedute, che andranno a modificare le registrazioni corrispondenti.

```
regN(98,2,0,10,1,3,2). regN(57,0,3,1,2,2,0). regN(49,0,2,3,3,3,3).
```

4.4.3 Output

- Gli appuntamenti sono registrati in y(R,D,H,T1,T2,T3,P,OT) e per ognuna di queste si associa res(R,D,H,NCH,NN) per ogni ora in cui il paziente ID richiede le risorse NC e NN. Le sedie e le infermiere sono identificate da chair(ID,R,D,H) con id della sedia (ID) e da nurses(ID,R,D,H) con id infermiere (ID).

4.4.4 Codifica

4.4.5 Regole in comune con pianificazione

Alcune regole presenti nel modello per la ripianificazione sono uguali a quelle per la pianificazione.
In particolare le regole 6-7-8-9-10-11 del pianificatore sono in comune e servono per assegnare il numero corretto di sedie ed infermieri ai vari pazienti (regole 6 e 7) e a controllare che le risorse non vengano assegnate a più persone per quanto riguarda le sedie e a non più di r persone per quanto riguarda gli infermieri (regole 8 e 9). La regola 10 serve per assegnare la stessa sedia ad un paziente che rimane per più ore nella clinica, così che non si debba spostare da una sedia all'altra, mentre le regole 11 controllano che le sedie utilizzate ogni ora non superino quelle a disposizione della clinica.
Infine la regola 12 controlla che le visite non vadino oltre l'orario consentito dalla clinica e la visita 13 serve per vincolare ad un valore fissato il numero di sedute massime che possono esserci in un giorno in una data sedia.

4.4.6 Regole adattate per la ripianificazione

Le regole presentate adesso sono regole che rimangono concettualmente invariate rispetto alla pianificazione ma che cambiano per via dell'atomo x. Infatti nella pianificazione le sedute non vengono più assegnate tramite l'atomo x, ma tramite l'atomo y.
In particolare le regole che subiscono questa modifica sono le regole 3-4-5, che servono ad assegnare ad ogni ora e ad ogni paziente il corretto numero di risorse.
Infine anche la regola 16, che controlla la quantità di medicinale utilizzato ogni giorno, viene modificato sostituendo all'atomo x l'atomo y.

4.4.7 Regole nuove per la ripianificazione

Si assegnano tutte le sedute precedenti al primo giorno in cui si modifica una seduta:

```
21 y(R,D,H,T1,T2,T3,P,M) :- x(R,D,H,T1,T2,T3,P,M), D < #min{DAY: un(_,DAY); D:
    regN(R,_,N,_,_,_,_),x(R,D,_,_,_,_,_,N)}.
```

Questa regola assegna in y i valori di x se il giorno della seduta x che si sta trattando è minore del giorno più piccolo tra il primo giorno di indisponibilità di un paziente e la modifica della seduta di un paziente. Per selezionare il giorno più piccolo tra quelli in cui viene fatta una modifica o viene segnalata un'indisponibilità si utilizza la funzione min.

Per tutte le registrazioni delle prime sedute che non sono modificate si assegna il paziente in uno slot tra quelli esistenti in un giorno in cui il paziente è disponibile:

```
22 1 {y(R,D,H,T1,T2,T3,P,0) : mss(D,H),not un(R,D),D <= DD} 1 :-
    reg(R,P,0,DD,T1,T2,T3),not regN(R,_,0,_,_,_,_).
```

Per ogni atomo reg che non ha un suo corrispettivo regN, si crea l'atomo y scegliendo tra i giorni presenti nel mss un giorno che però non sia presente nell'atomo un con lo stesso ID del paziente di cui si sta pianificando la seduta, e che sia prima della data limite di inizio delle sedute. Per questa regola abbiamo usato una choice rule con alcuni vincoli per rispettare i requisiti del problema.

Per tutte le registrazioni delle prime sedute che devono essere modificate si assegna il paziente in uno slot tra quelli esistenti in un giorno in cui il paziente è disponibile:

```
23 1 {y(R,D,H,T1,T2,T3,P,0) : mss(D,H),not un(R,D),D <= DD } 1
    :-regN(R,P,0,DD,T1,T2,T3).
```

Per ogni atomo regN associato ad una prima seduta, si crea l'atomo y. L'atomo viene creato assegnando le informazioni presenti in regN ed assegnando un giorno D tra quelli presenti nel mss e che non sia presente nell'atomo un con lo stesso ID del paziente di cui si sta

pianificando la seduta, e che sia prima della data limite di inizio delle sedute.

Per tutte le sedute che non sono state modificate rispetto al vecchio calendario assegna la seduta successiva nel calendario, se il paziente non ha problemi di disponibilità o sedute da modificare:

```
24 y(R,D,H,T1,T2,T3,0,M) :-
    y(R,D,_,_,_,_,_,M-1),x(R,D,_,_,_,_,_,M-1),x(R,D,H,T1,T2,T3,0,M),not
    regN(R,_,0,_,_,_,_),not un(R,_) .
```

Per ogni atomo y rappresentante una seduta e gli atomi x rappresentanti la stessa seduta e la seduta successiva dello stesso paziente, col paziente che non abbia una modifica nella prima seduta e che non sia indisponibile in un giorno, assegna l'atomo x rappresentante la seduta successiva in un nuovo atomo y.

Per tutte le sedute successive alle sedute che sono state pianificate un giorno diverso rispetto al calendario originale, si aggiunge una nuova seduta, se non hanno una nuova registrazione:

```
25 1 { y(R,D,H,T1,T2,T3,0,N+1) : mss(D,H),not un(R,D),D > D1 } 1 :-
    y(R,D1,_,_,_,_,_,N),reg(R,_,N+1,D2,T1,T2,T3),not
    regN(R,_,N+1,_,_,_,_),mss(D1 + D2,_).
```

Per ogni atomo y, rappresentante una seduta che è stata spostata rispetto al calendario originale, e che non ha una nuova registrazione regN, si assegna la nuova seduta in un giorno che rispetti le condizioni di non essere un giorno in cui il paziente è indisponibile e sia un giorno successivo rispetto a quello pianificato nel calendario originario.

Per tutte le sedute successive alle sedute che sono state pianificate un giorno diverso rispetto al calendario originale, si aggiunge una nuova seduta, se hanno una nuova registrazione:

```
26 1 {y(R,D,H,T1,T2,T3,0,N+1) : mss(D,H),not un(R,D),D > D1 1} :-
     y(R,D1,_,_,_,_,_,N),regN(R,_,N+1,D2,T1,T2,T3),mss(D1 + D2,_).
```

Per ogni atomo y, avente una corrispettiva nuova registrazione, che modifica quindi la seduta successiva, si registra la nuova seduta seguendo le indicazioni della nuova registrazione. Il giorno che si assegna rispetta sempre le condizioni di non essere un giorno in cui il paziente è indisponibile e sia un giorno successivo rispetto a quello pianificato nel calendario originario.

Assegniamo tutte le sedute precedenti alla data di indisponibiltà dei pazienti che sono indisponibili un giorno:

```
27 y(R,D,H,T1,T2,T3,0,N+1) :- x(R,D,H,T1,T2,T3,0,N+1),un(R,_),D < min{D:
     un(R,D).}
```

Se un paziente è indisponibile un giorno, si utilizza la funzione min per trovare il suo primo giorno di indisponibilità e assegniamo tutte le sue sedute precedenti a quel giorno.

Non si possono spostare le sedute precedentemente a quando le avevano originariamente:

```
28 :- y(R,D1,_,_,_,_,_,M),x(R,D2,_,_,_,_,_,M),D1 < D2. }
```

Si utilizza un constraint per fare in modo che due sedute corrispondeti x ed y se vere, abbiano il giorno della seduta y, quindi quella del nuovo calendario, lo stesso giorno o dei giorni successivi rispetto al giorno originario.

4.4.8 Ottimizzazione

Minimizza la differenza tra la distanza tra le sedute ottimale e quella reale:

```
29 #minimize{|DD - (D2 - D1)|@4,R,M:
       y(R,D1,_,_,_,_,_,M),y(R,D2,_,_,_,_,_,M+1),reg(R,_,M+1,DD,_,_,_),not
       regN(R,_,M+1,_,_,_,_)}.
```

Per tutte le sedute indicate da y e le sedute successive, che non hanno subito modifiche con nuove registrazioni, si calcola il modulo della differenza tra la distanza reale in giorni tra le due sedute, e la distanza ottimale. Questa minimizzazione ha peso massimo e quindi ha la priorità maggiore. In questo modo il nostro modello cercherà di assegnare le giuste distanze tra tutte le sedute, anche quelle che hanno la prima seduta spostata rispetto al calendario originario.

Minimizza la differenza tra la distanza tra le sedute ottimale e quella reale per le sedute che aevavno una nuova registrazione.

```
30 #minimize{|DD - (D2 - D1)|@4,R,M:
       y(R,D1,_,_,_,_,_,M),y(R,D2,_,_,_,_,_,M+1),regN(R,_,M+1,DD,_,_,_)}.
```

Per tutte le sedute indicate da y e le sedute successive, che hanno subito modifiche con nuove registrazioni, si calcola il modulo della differenza tra la distanza reale in giorni tra le due sedute, e la distanza ottimale. Questa minimizzazione ha peso massimo e quindi ha la priorità maggiore. In questo modo il nostro modello cercherà di assegnare le giuste distanze tra tutte le sedute, anche quelle che hanno la prima seduta spostata rispetto al calendario originario.

Si minimizza la distanza tra le prime sedute nel calendario originale e quello nuovo con peso decrescente a seconda della priorità

```
31 minimize{|D2 - D1|@3,R: y(R,D1,_,_,_,_,1,0),x(R,D2,_,_,_,_,1,0)}.
32 minimize{|D2 - D1|@2,R: y(R,D1,_,_,_,_,1,0),x(R,D2,_,_,_,_,2,0)}.
33 minimize{|D2 - D1|@1,R: y(R,D1,_,_,_,_,1,0),x(R,D2,_,_,_,_,3,0)}.
```

Per tutte le nuove prime sedute e le prime sedute originarie si calcola il modulo della differenza tra il nuovo giorno e quello originario e se ne minimizza la somma, in questo

modo cerchiamo di spostare il meno possibile le prime sedute dei pazienti.

Capitolo 5

Risultati analisi sperimenatali

In questo capitolo presentiamo inizialmente il setting in cui abbiamo eseguito i test e gli scenari che abbiamo testato, indicando come sono state generate le varie istanze di Input. Successivamente analizziamo i risultati ottenuti dalla nostra soluzione sia per quanto riguarda la pianificazione sia che per la ripianificazione. I risultati ottenuti, che saranno trattati nello specifico nei prossimi paragrafi, ci permettono di dire che ASP risulta essere uno strumento utile per risolvere in tempi ragionevoli problemi complessi come quello da noi trattato.

5.1 Setting sperimentale e benchmarks

Prima di presentare gli scenari e le istanze che abbiamo generato presentiamo il setting sperimentale, quindi il computer che abbiamo utilizzato e i paramentri utilizzati con clingo. Per quanto riguarda il computer abbiamo eseguito i test su un computer con a disposizione 8 gb di RAM e un processore AMD Ryzen 5 2600 avente 6 core e 12 thread e una frequenza di 3.4 Ghz.
Per ottenere i risultati che abbiamo ottenuto, nella pianificazione abbiamo usato come parametri per clingo: time out a 300 secondi, che veniva utilizzato da tutte le istanze con più di 20 pazienti. Con 20 pazienti invece la soluzione è ottima e trovata in pochi secondi. Inoltre abbiamo usato l'impostazione parallel-mode 12, che permette di sfruttare 12 thread durante la ricerca dell'ottimo e l'opzione restart-on-model, che indica al solver di ripartire da capo con la ricerca dell'ottimo una volta trovata una soluzione.
Per quanto riguarda la ripianificazione invece il sistema non ha un limite di tempo perchè raggiunge l'ottimo in un tempo ragionevole (con l'istanza più complessa poco più di un minuto), abbiamo mantenuto l'opzione parallel-mode 12 ma non usiamo più restart-on-model, che è

più indicata nella ricerca di una buona soluzione in tempi brevi, piuttosto che nella ricerca dell'ottimo, ma visto che in questo caso otteniamo la soluzione in poco tempo utilizziamo –opt-strategy=usc e –opt-usc-shrink=bin, che si sono rivelati molto efficaci nel trovare una soluzione ottima finita in poco tempo.

Prima di analizzare i risultati presentiamo le caratteristiche sperimentali dell'Input.

I test sono stati eseguiti prendendo in considerazione due diversi scenari. Il primo (scenario α) è caratterizzato da un quantitativo di medicine che permette al sistema di sfruttare le risorse della clinica simulata in maniera ottimale. Per il secondo (scenario β), abbiamo ridotto di molto le medicine per poter testare la nostra soluzione in una situazione in cui le medicine diventano la risorsa limitante del sistema, e di conseguenza, le altre risorse, come per esempio le sedie vengono utilizzate minormente.

I risultati per la pianificazione sono ottenuti utilizzando 15 sedie ed impostando a 4 e a 7 il numero massimo di pazienti visitati in un'ora dagli infermieri, che sono 5, abbiamo eseguito i test su 10 diverse istanze per ogni gruppo di 20, 40, 60, 80 e 100 pazienti per ognuno degli scenari creati.
Le caratteristiche delle istanze per ogni gruppo di pazienti sono le seguenti:

- 2 diversi scenari, comprendenti un periodo di 14 giorni lavorativi, e diversi numeri di medicine disponibili, come disposto nella tabella 5.2 e nella tabella 5.3, per ogni gruppo di pazienti;
- 3 diversi tipi di medicine che sono assegnati ai pazienti seguendo lo schema riportato nella tabella 5.1;
- Per ogni paziente, ci sono 6 diverse registrazioni, ognuna corrispondente ad un giorno di trattamento, seguendo lo schema riportato nella tabella 5.1, con la prima avente un livello di priorità generato casualmente e una due date del primo trattamento che ha un valore compreso in un certo range, a seconda della priorità.
  In questo modo, simuliamo la situazione in cui un manager prenda una lista di pazienti con diverse priorità e cerchi di assegnarli tutti il prima possibile, prendendo in considerazione le diverse priorità.

Le priorità assegnate alla prima registrazione sono generate partendo da tre possibili valori aventi peso differente (rispettivamente 0.20, 0.40 e 0.40 per le registrazioni con priorità 1,2 e 3). I range delle date corrispondenti al due date di ogni priorità sono i seguenti:

Tabella 5.1 Schema seguito da ogni paziente, ispirato da uno schema presentato da Turkcan et al. (2010)

Giorno	Medicine	Dose	Durata
1	A-B-C	20 mg/m^2, 100 mg/m^2, 30 unità	1, 2, 1 finestre temporali
2-5	A-B	20 mg/m^2, 100 mg/m^2	1, 2 finestre temporali
6-7	Riposo		
8	C	30 unità	1 finestra temporale

- [1,6) per la priorità 1,
- [6,11) per la priorità 2 e
- [11,15) per la priorità 3.

Tabella 5.2 Disponibilità medicine per ogni gruppo di pazienti nello scenario α.

Numero di pazienti	Medicina A	Medicina B	Medicina C
20	300	1600	450
40	400	1600	550
60	400	2000	600
80	500	2500	800
100	700	3200	1150

Tabella 5.3 Disponibilità medicine per ogni gruppo di pazienti nello scenario β.

Numero di pazienti	Medicina A	Medicina B	Medicina C
20	200	1100	350
40	300	1200	400
60	350	1850	550
80	400	2250	700
100	550	3000	900

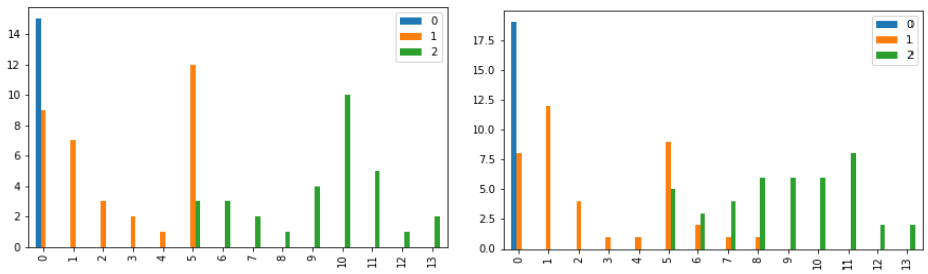
I parametri dei test sono riassunti nella tabella 5.4. In particolare, per ogni gruppo di pazienti, presentiamo la media e la deviazione standard del numero di pazienti con priorità 1,2 e 3. Come detto precedentemente, i vari livelli di priorità comportano una diversa data limite entro la quale deve essere effettuata la prima seduta, le sedute successive, essendo lo schema dei trattamenti fissi, sono definite di conseguenza nei giorni successivi.

Tabella 5.4 Parametri per gli Input generati casualmente. Con std indicante la deviazione standard.

Pazienti	Scenario	Num. priorità 1 media (std)	Num. priorità 2 media (std)	Num. priorità 3 media (std)
20	α	3.7 (1.1)	8.8 (2.44)	7.5 (2.01)
40	α	7.6 (1.9)	15.5 (1.9)	16.9 (3.17)
60	α	12.9 (2.5)	24 (4.82)	23 (4.65)
80	α	17.3 (1.48)	32.6 (3.41)	30.1 (3.69)
100	α	19.5 (3.26)	38 (3.68)	42.5 (4.67)
20	β	3.1 (1.9)	9.2 (2.12)	8 (1.37)
40	β	8.1 (1.7)	14.5 (1.85)	7.2 (2.55)
60	β	11.9 (2.80)	24.4 (4.60)	23.7 (4.60)
80	β	16.8 (1.66)	32.6 (3.23)	30.6 (3.69)
100	β	19.5 (3.26)	38 (3.68)	42.5 (4.67)

5.2 Risultati pianificazione

Per dimostrare la bontà dei risultati, analizzeremo l'utilizzo delle risorse, quindi la percentuale di utilizzo delle sedie e degli infermieri e la percentuale di utilizzo delle medicine.
Prima di analizzare nel dettaglio i risultati ottenuti nei vari scenari presentiamo due grafici dei risultati ottenuti con 80 e 100 pazienti nello scenario α per analizzare e spiegare una caratteristica comune ai risultati ottenuti.



(a) 80 pazienti. (b) 100 pazienti.

Figura 5.1 Distribuzione prime sedute in 14 giorni per 80 e 100 pazienti.

Come si può notare nelle immagini in 5.1 vi è un pattern particolare nella distribuzione delle sedute, infatti, oltre ad un picco i primi due giorni vi è un picco il giorno 5 e successivamente tra il giorno 10 e il giorno 11. Questo comportamento che potrebbe sembrare anomalo è invece giustificato dalla presenza di un solo schema di sedute, che in particolare prevede 4

giorni di fila più una pausa di 3 giorni prima dell'ultima seduta.
Nel grafico, il giorno 5, corrisponde proprio al primo giorno in cui i pazienti del giorno 1 non hanno visite, e liberano quindi le risorse per nuovi pazienti. Come già detto nei capitoli precedenti questa situazione nella realtà non comporta problemi, infatti in uno scenario reale i pazienti avranno più schemi diversi ed in più la pianificiazione sarà eseguita su 2 settimane ma con la seconda in comune con la pianificazione eseguita precedentemente, così da sfruttare a pieno tutti i giorni.

Per verificare la veridicità della nostra ipotesi abbiamo testato il modello assegnando ai pazienti 3 diversi schemi di sedute ed abbiamo ottenuto il grafico 5.2.

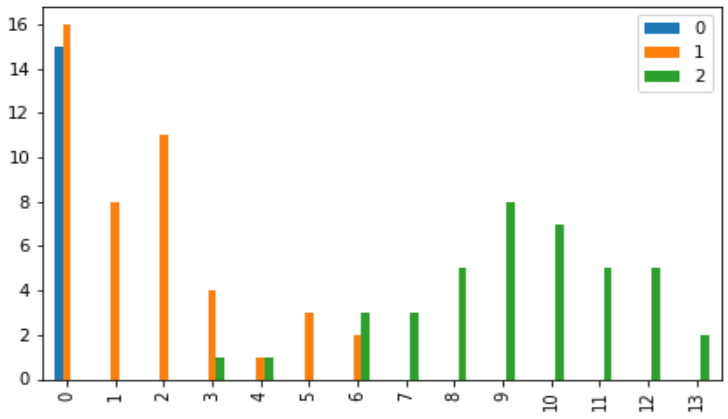
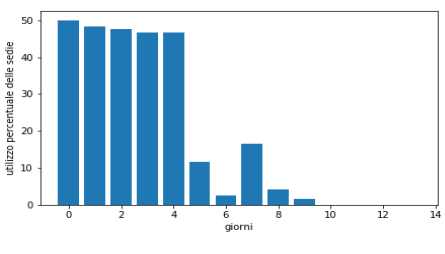


Figura 5.2 Distribuzione prime sedute in 14 giorni per 100 pazienti con calendario sedute diversi.

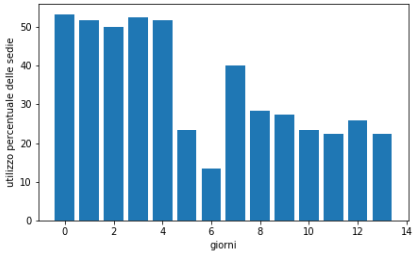
Nel grafico 5.2, si può notare che non ci sono più i picchi ottenuti con un solo schema di trattamento, questo conferma la nostra ipotesi. Tuttavia vediamo che anche con 3 schemi si ha un andamento a discesa prima di un picco successivo, ma questo è dovuto alla presenza delle sedute successive più che ad avere un singolo schema, averne più di 1 diminuisce la presenza del pattern, ma non può eliminarla completamente. Questo risolve quindi il problema dei picchi ma non risolve il problema della sporadicità dei trattamenti nella seconda settimana, che sarebbe mitigato in un caso reale da un utilizzo continuativo del pianificatore, in particolare si potrebbero pianificare le sedute dei pazienti per 14 giorni e successivamente pianificare quelle di un nuovo gruppo di pazienti dopo 7 giorni, per fare in modo che la seconda settimana del primo pianificatore combaci con la prima settimana del secondo pianificatore, così da sfruttare a pieno tutti i giorni.

Analizzato l'andamento particolare delle sedute nel calendario passiamo ora ad un'analisi più specifica sulle risorse utilizzate nello scenario α. I grafici mostrati successivamente sono tutti grafici ottenuti impostando il rateo di pazienti visitabili in un'ora da un infermiere a 4, le considerazioni fatte su questi grafici sono valide anche per i risultati ottenuti con il rateo pari a 7. Non mostriamo quei grafici perchè sono estremamente simili a quelli con rateo uguale a 4, infatti, come si vedrà nei paragrafi successivi, le risorse sono sfruttate quasi completamente e quindi aumentare il numero di pazienti visitabili in un'ora non comporta vantaggi significativi.
Inoltre ricordiamo che mentre i risultati ottenuti con 20 pazienti rappresentano la soluzione ottima, ottenuta in pochi secondi, i risultati ottenuti con gli altri gruppi di pazienti sono soluzioni che non rappresentano l'ottimo ma una sua approssimazione. Il nostro modello infatti cerca la soluzione ottima per al massimo 300 secondi, raggiunti i quali smette di ricercare l'ottimo e restituisce il miglior risultato ottenuto in quel lasso di tempo.
Per quanto riguarda l'utilizzo medio delle varie sedie in un'istanza con 20, 40, 60, 80 e 100 pazienti i risultati ottenuti sono presentati nella figura 5.3.
Dai grafici nella figura 5.3 possiamo fare due analisi diverse per i risultati ottenuti con 20 e 40 pazienti e con 60, 80 e 100 pazienti.
Nel caso di 20 e 40 pazienti notiamo che l'utilizzo delle sedie sia molto basso rispetto a quello ottenuto con più pazienti, questo comportamento è dovuto ad una minor disponibilità di medicinali in proporzione alle sedie. Questo limite imposto dai medicinali sarà più evidente nei grafici successivi, dove analizzeremo nel dettaglio l'utilizzo dei medicinali.
Per quanto riguardo i risultati ottenuti con 60, 80 e 100 pazienti invece, si può vedere come, soprattutto i primi giorni l'utilizzo delle sedie sia molto alto, raggiungendo picchi vicino al 100% di utilizzo, e rimane sopra l'80% per quasi la totalità dei giorni. I giorni in cui si ha un calo corrispondono al giorno successivo al picco di nuovi pazienti, questo perchè con questo schema di sedute si "perdono" i pazienti dei primi 2 giorni, ma contemporaneamente non si possono aggiungere molti altri pazienti in quanto poi i pazienti dei primi 2 giorni avranno ancora una seduta nei giorni successivi. Per verificare quanto detto dobbiamo analizzare l'utilizzo della altre risorse e successivamente potremo confermare quanto detto ora.

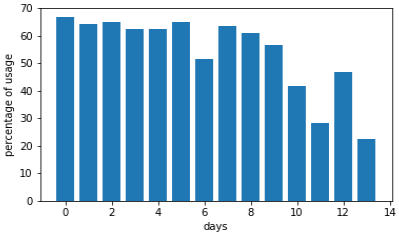
Passiamo ora ad analizzare nello specifico una sedia tra quelle utilizzate e vediamo l'utilizzo percentuale ottenuto in un'istanza per ogni gruppo di pazienti nei grafici nella figura 5.4.
Non essendoci un requisito che renda uniforme l'utilizzo delle sedie è più importante analizzare la media complessiva piuttosto che una singola sedia. Detto questo l'utilizzo nei vari giorni della sedia analizzata anche se utilizzata in maniera minore non si discosta troppo



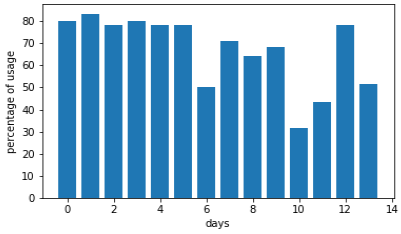
(a) 20 pazienti.



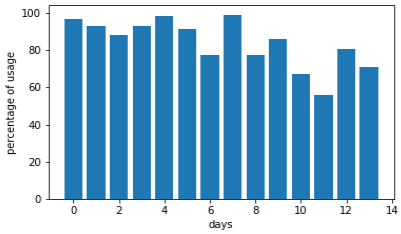
(b) 40 pazienti.



(c) 60 pazienti.



(d) 80 pazienti.

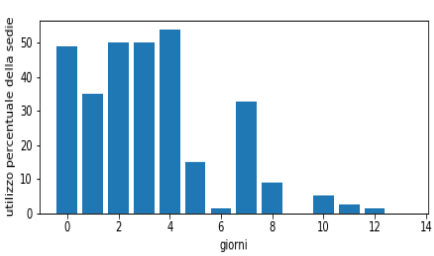


(e) 100 pazienti.

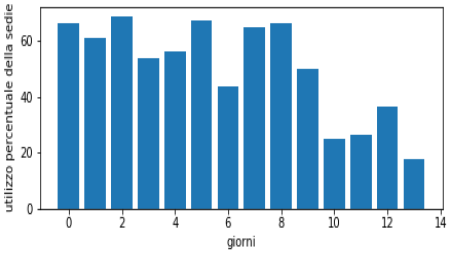
Figura 5.3 Utilizzo medio delle sedie nelle istanze con 20, 40, 60, 80 e 100 pazienti.

dalla media globale delle sedie nelle istanze con più pazienti. La differenza tra la singola sedia e la media globale sarà più marcata nelle situazioni di sotto utilizzo globale delle sedie, quindi nel nostro caso con 40 e ancora di più con 20 pazienti, si nota una grande differenza tra l'utilizzo medio delle sedie e l'utilizzo della sedia singola, mentre in una clinica in cui si utilizzino le sedie costantemente sopra all'80% la differenza tra le singole sedie sarà poco significativa.

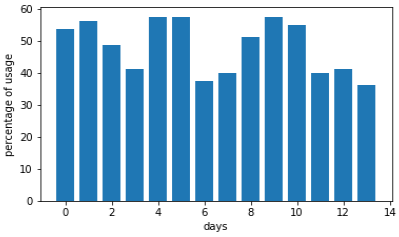
Essendo l'obbiettivo della nostra soluzione ottimizzare il numero di pazienti visitati e quindi in maniera indiretta ottimizzare le risorse utilizzate, ci aspettiamo che si ottengano percentuali di utilizzo delle sedie molte alte e di conseguenza la discrepanza tra le singole sedie non sarebbe un problema, può diventare un problema nel momento in cui una clinica abbia delle risorse in quantità eccessiva o scarsa rispetto alle altre, in quel caso la gestione delle singole



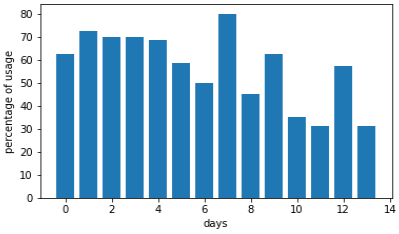
(a) 20 pazienti.



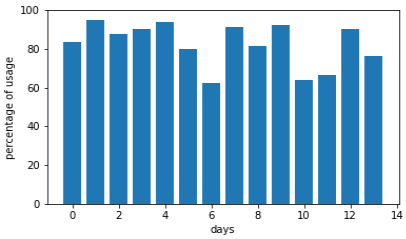
(b) 40 pazienti.



(c) 60 pazienti.



(d) 80 pazienti.



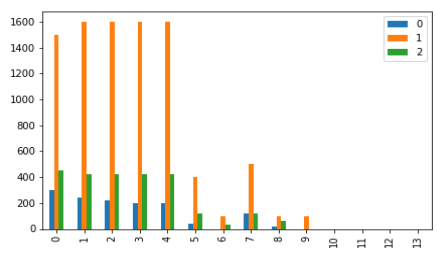
(e) 100 pazienti.

Figura 5.4 Utilizzo nei vari giorni di una sedia nelle istanze con 20, 40, 60, 80 e 100 pazienti.

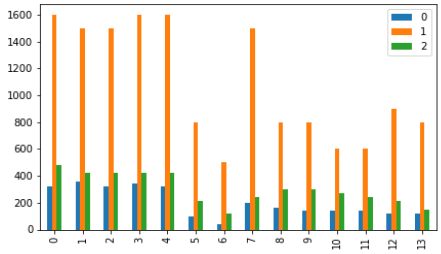
sedie andrebbe controllata aggiungendo ulteriori vincoli.

Così come nella media anche nel grafico delle sedie singole si nota un andamento simile a quello delle visite nel calendario. Per giustificare questo andamento analizziamo i grafici dei medicinali.

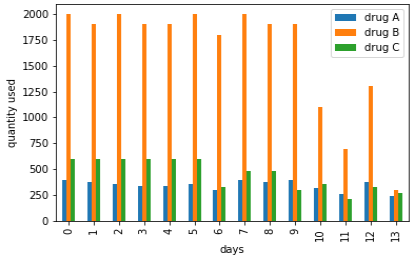
Nei grafici nella figura 5.5 possiamo notare che l'utilizzo delle medicine sia sempre molto vicino ai valori indicati nella tabella 5.2, soprattutto nei grafici con 20 e 40 pazienti possiamo notare come essendo sempre così vicino al limite dei medicinali sia effettivamente questa la causa del sottoutilizzo delle sedie, e non una cattiva pianificazione del nostro modello.
Come abbiamo detto precedentemente, i giorni in cui si ha un calo di utilizzo delle sedie



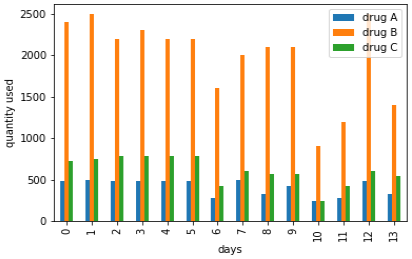
(a) 20 pazienti.



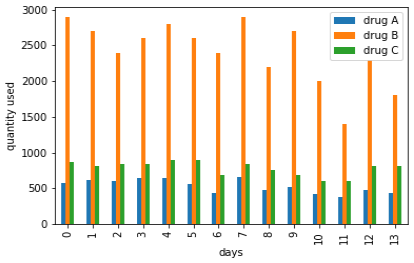
(b) 40 pazienti.



(c) 60 pazienti.



(d) 80 pazienti.

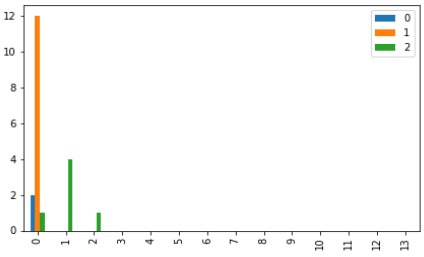


(e) 100 pazienti.

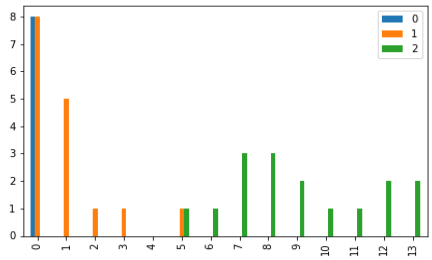
Figura 5.5 Utilizzo medio nei vari giorni dei medicinali nelle istanze con 20, 40, 60, 80 e 100 pazienti.

corrispondono ai giorni successivi al picco di nuovi pazienti nel giorno 5, questo perchè con questo schema di sedute i pazienti dei primi 2 giorni sono nei loro giorni di riposo, ma per completare il loro ciclo di sedute dovranno tornare, utilizzando le risorse a loro necessarie, quindi in quei giorni non si possono aggiungere molti altri pazienti in quanto i pazienti dei primi 2 giorni hanno già bloccato molte risorse nei giorni successi. Questo comportamento si può notare analizzando il grafico dell'utilizzo dei medicinali, che presenta due picchi di utilizzo proprio il giorno dopo al calo dell'utilizzo delle sedie, questo ci fa capire che anche il poco utilizzo di risorse di quel giorno è portato al limite o quasi visto che i giorni successivi si saturano i medicinali.

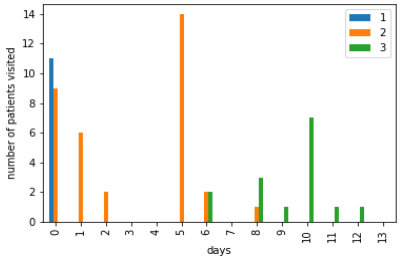
Per quanto riguarda la pianificazione dei pazienti in sè dopo aver verificato che le risorse sono utilizzate in modo quasi ottimo (comunque soddisfacente considerando che sono stati ottenuti lasciando il modello lavorare per 300 secondi) vediamo nella figura 5.6 come sono organizzati i pazienti e se i vari livelli di priorità sono rispettati.



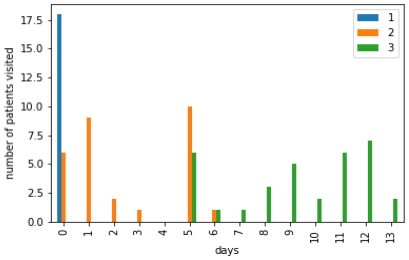
(a) 20 pazienti.



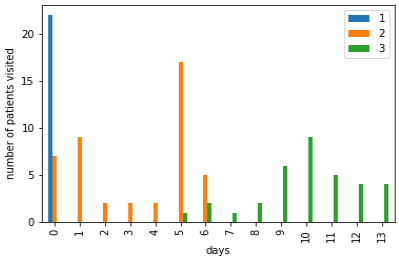
(b) 40 pazienti.



(c) 60 pazienti.



(d) 80 pazienti.



(e) 100 pazienti.

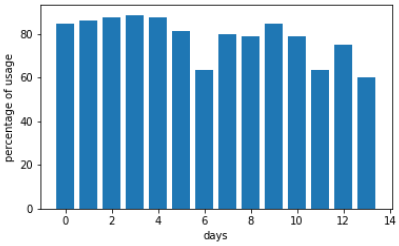
Figura 5.6 Distribuzione delle prime sedute dei pazienti differenziati per priorità.

Possiamo vedere che con tutti i diversi gruppi di pazienti quelli con priorità massima, la priorità 1 corrispondente al blu nei grafici, iniziano il ciclo di sedute il primo giorno successivo. I pazienti con priortià media sono tutti visitati i giorni successivi, considerando per ognuno di essi la data limite entro la quale iniziare le sedute. I pazienti a priorità minore sono visitati quando i pazienti a priorità media sono terminati o stanno terminando. Può capitare che vi sia un certo numero di pazienti a priorità minore che vengano visitati prima di pazienti

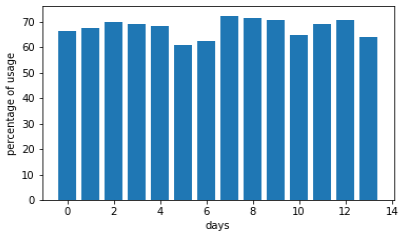
con priorità più alta, questo avviene nei grafici presentati nei giorni 5 con 80 e 100 pazienti, questa situazione non è ottimale e la otteniamo perchè come detto 300 secondi non sono sufficienti per ottenere l'ottimo. Tuttavia per quanto riguarda la distribuzione dei pazienti a seconda delle priorità il risultato è molto vicino a quello ottimo, essendo pochi i pazienti a priorità minore che vengono visitati prima e quando succede vi è la differenza di un giorno.

Avendo analizzato lo scenario α vediamo ora lo scenario β per vedere come la diminuzione della disponibilità dei medicinali influenza i risultati.
Nello scenario β consideriamo solo i risultati per 60, 80 e 100 pazienti perchè, come mostrato precedentemente le istanze con 20 e 40 pazienti erano già limitati dai medicinali.

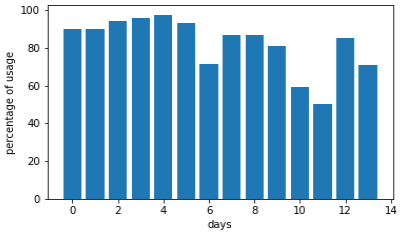
Partiamo dal grafico nella figura 5.7, che rappresenta l'utilizzo medio percentuale delle varie sedie. In questo grafico si può notare un calo del 20% nell'utilizzo delle sedie: questo è causato dalla scarsità dei medicinali, che si esauriscono prima che possano essere sfruttate pienamente le sedie. Questo comportamente era prevedibile visto come abbiamo strutturato lo scenario β.



(a) 60 pazienti.



(b) 80 pazienti.

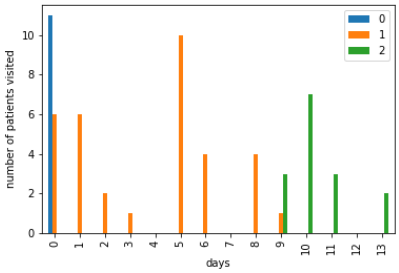


(c) 100 pazienti.

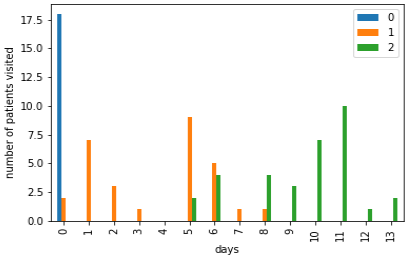
Figura 5.7 Utilizzo medio delle sedie nelle istanze con 20, 40, 60, 80 e 100 pazienti nello scenario β.

Oltre ad un calo dell'utilizzo delle risorse vi è anche un naturale peggioramento dei risultati

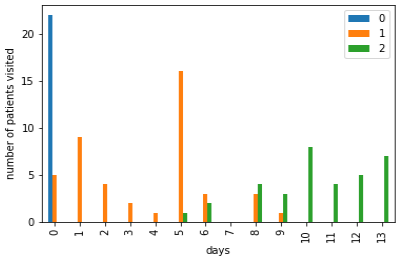
ottenuti nel calendarizzare i pazienti, come si vede nella figura 5.8. In particolare si nota che i pazienti con priorità 2 vengono calendarizzati più avanti nel calendario e la nostra soluzione fa più fatica a separare gli ultimi pazienti con priorità media e quelli con priorità più bassa.



(a) 60 pazienti.



(b) 80 pazienti.



(c) 100 pazienti.

Figura 5.8 Distribuzione delle prime sedute dei pazienti differenziati per priorità nello scenario β.

Presentiamo la tabella 5.5 e la tabella 5.6 che riassumono l'utilizzo delle sedie mentre le tabelle 5.7 e 5.8 riassumono l'utilizzo percentuale dei medicinali utilizzati in entrambi gli scenari, i risultati rappresentano la media ottenuta testando per ogni scenario e gruppo di pazienti 10 istanze differenti.

Nelle tabelle 5.5 e 5.6 e nelle tabelle 5.7 e 5.8 si conferma quello che si poteva notare nei grafici, più andiamo avanti nei giorni e quindi più si esauriscono i pazienti a cui pianificare la seduta, più le risorse vengono meno utilizzate, sia per quanto riguarda le sedie, che per quanto riguarda i medicinali.

Questo, come detto precedentemente non è da inputare ad una cattiva pianificazione, ma anzi, vuol dire che il pianificatore riesce a pianificare tutte le sedute i primi giorni. Si nota anche una differenza nei due scenari, infatti, grazie alle maggiori risorse disponibili, lo scenario α riesce ad eseguire più sedute nei primi giorni ed ad arrivare nelle ultime giornate come pochi

Tabella 5.5 Utilizzo medio delle sedie (in % sul totale disponibile) per lo scenario α.

	Giorno													
Pazienti	0	1	2	3	4	5	6	7	8	9	10	11	12	13
20	50	44	43	43	43	10	3	20	5	3	1	1	0	0
40	53	51	50	50	50	42	30	45	39	37	18	15	23	11
60	66	64	62	61	61	57	38	59	57	57	41	36	49	29
80	83	82	81	80	80	77	52	75	68	67	46	43	68	50
100	92	94	93	95	97	86	67	89	83	89	71	76	90	72

Tabella 5.6 Utilizzo medio delle sedie (in % sul totale disponibile) per lo scenario β.

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

pazienti, in questo scenario aggiungere un nuovo gruppo di pazienti ogni 7 giorni sembra essere una soluzione ottimale per permettere di sfruttare a pieno le risorse.

Tabella 5.7 Utilizzo medio dei medicinali (in % sul totale disponibile) per lo scenario α.

		Giorno													
Pazienti	Medicina	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	A	100	87	84	80	80	17	3	38	11	5	2	2	1	0
20	B	93	88	85	85	85	19	5	38	8	5	3	2	1	0
	C	100	85	85	84	84	23	6	38	14	6	3	2	0	0
	A	80	83	78	78	78	61	42	71	51	53	25	19	32	15
40	B	100	92	93	90	90	80	54	84	76	71	36	31	48	23
	C	87	81	82	82	82	73	50	74	65	62	31	27	31	17
	A	97	93	93	92	93	80	61	91	80	89	65	58	78	57
60	B	97	92	93	92	92	80	57	84	78	82	59	54	74	54
	C	97	93	94	94	96	83	57	81	72	79	63	54	71	55
	A	100	98	96	95	94	91	66	90	83	81	54	50	83	62
80	B	100	97	95	94	94	92	61	92	78	80	58	54	80	60
	C	93	95	97	96	96	90	59	81	80	75	50	45	77	56
	A	83	83	85	87	89	77	57	75	71	77	64	68	77	59
100	B	86	86	83	84	86	78	66	86	80	86	67	70	85	67
	C	72	75	77	79	81	69	48	66	61	66	53	57	70	58

Tabella 5.8 Utilizzo medio dei medicinali (in % sul totale disponibile) per lo scenario β.

		Giorno													
Pazienti	Medicina	0	1	2	3	4	5	6	7	8	9	10	11	12	13
60	A	96	92	93	93	91	89	69	92	87	94	87	81	92	83
	B	91	89	94	93	92	88	66	91	84	86	75	63	75	68
	C	92	93	94	95	93	86	68	90	79	86	74	63	83	68
80	A	100	96	97	98	100	91	84	97	97	97	95	94	99	98
	B	89	84	95	94	95	86	70	92	92	84	81	84	91	92
	C	86	88	95	94	95	80	72	91	91	87	76	81	95	87
100	A	98	95	94	96	98	96	75	95	91	92	82	85	92	85
	B	90	92	94	93	94	87	64	90	81	82	71	73	91	78
	C	90	94	93	94	95	85	60	88	76	77	73	74	90	80

5.3 Risultati ripianificazione

Sempre di più la medicina e ancora con più forza la medicina oncologica vanno verso un trattamento che viene definito di 'precisione', risulta quindi di vitale importanza affiancare ad un programma di pianificazione anche uno per la ripianificazione, vista la frequente modifica dei trattamenti assegnati ai pazienti in seguito alla risposta ai farmaci prescritti.
Vista l'importanza della ripianificazione consideriamo il problema di avere un calendario di sedute già programmato e doverlo modificare a seguito di alcune situazioni, che possono essere: modifiche al trattamento che deve seguire un paziente, giornata di indisponibilità di un paziente che non potrà esserci un determinato giorno o anche cause legate alla clinica e quindi eventuali orari all'interno di una giornata in cui non si potranno effettuare i trattamenti.
Per ottenere il risultato ottenuto la soluzione cerca di minimizzare i giorni di 'spostamento' dal trattamento indicato dall'Input nel caso in cui il paziente non potesse recarsi nella clinica un giorno, così da minimizzare la differenza dal trattamento assegnato al paziente, mentre nel caso in cui fosse stato modificato il trattamento, e in particolare il numero di giorni tra una seduta e l'altra di un paziente, il modello rispetterà il nuovo programma cercando di modificare il meno possibile il calendario degli altri pazienti, e spostando al più le persone che devono eseguire la prima seduta (cercando di spostare pazienti con priorità minore).

A differenza della pianificazione il problema della ripianificazione permette di ottenere un risultato ottimo in tempo brevi, quindi non viene impostato un tempo limite, anche per le istanze con più pazienti.

Per verificare il funzionamento della ripianificazione consideriamo varie situazioni arrivando fino a considerare l'impossibilità di effettuare 20 diverse sedute in un determinato giorno e modificando 3 sedute in un calendario organizzato con 100 pazienti, che è una situazione molto estrema. Questo scenario ottiene un risultato ottimo in 65 secondi.

Per la ripianificazione abbiamo provato 10 situazioni diverse, con un numero cresente di pazienti e un numero crescente di indisponibilità e modifiche alle sedute, in particolare vediamo un riassunto delle istanze nella tabella 5.9.
I risultati ottenuti sono riassunti nella tabella 5.10, come si vede nella tabella il ripianificatore riesce a sfruttare molto spesso i buchi nel calendario per riassegnare le sedute senza spostare gli altri pazienti, quando non riesce comunque il numero di sedute spostate è limitato e analizzando i risultati si vede che vengono spostate al più di un giorno.

Tabella 5.9 Riassunto istanze per la ripianificazione.

Istanza numero	Numero di pazienti	Pazienti con giorni indisponibili	sedute da modificare
1	20	6	0
2	60	1	0
3	60	0	1
4	60	1	1
5	60	5	5
6	100	1	0
7	100	5	0
8	100	10	2
9	100	20	0
10	100	20	5

Tabella 5.10 Riassunto risultati della ripianificazione.

Risultato dall'istanza	Pazienti priorità 1 spostati	Pazienti priorità 2 spostati	Pazienti priorità 3 spostati
1	0	0	0
2	0	0	2
3	0	0	0
4	0	0	2
5	0	0	2
6	0	0	1
7	0	0	0
8	0	0	0
9	0	0	0
10	0	1	0

Dopo aver mostrato i risultati delle istanze vediamo come si è giunti ad ottenere il risultato ottenuto con l'istanza più piccola, quindi con 20 pazienti con le sedute già pianificate e 6 pazienti che hanno un'indisponibilità 1 giorno. L'istanza analizzata corrisponde a quella mostrata nei precedenti capitoli, dove abbiamo mostrato esempi di Input e il calendario ottenuto. Adesso portiamo avanti l'esempio così da mostrare in maniera precisa un possibile caso in tutte le sue fasi, vedendo sia come è composto l'Input e il risultato che si ottiene con la pianificazione e come questo può essere modificato dalla ripianificazione.
Le giornate di indisponibilità sono presentate sotto con gli atomi un, mentre il calendario che si era ottenuto indicava per quelle giornate le sedute indicate con gli atomi x.

```
un(4,5). un(4,6). un(9,5). un(3,8). un(0,8). un(2,8).
```

```
x(4,5,6,1,3,0,0,3). x(4,6,2,1,3,0,0,4). x(9,5,2,3,2,0,0,4).
   x(3,8,7,2,0,0,0,5). x(0,8,3,3,0,0,0,5). x(2,8,2,2,0,0,0,5).
```

Come detto prima la nostra soluzione cerca di riassegnare queste sedute il prima possibile, cercando di spostare meno persone possibili, e se proprio non ci riesce, sposta le prime sedute dando peso alle priorità dei pazienti che dovrebbe spostare, quindi prima cerca di fare posto spostando le sedute dei pazienti con priorità 3, poi con quelli a priorità 2 ed infine, se non riesce altrimenti, prova con quelli a priorità 1.

Per vedere se sia possibile spostare le sedute dei pazienti nei giorni successivi dobbiamo analizzare la quantità di medicinale richiesta da ogni paziente e le ore che deve starci, infine la disponibilità nei giorni successivi.

Partiamo dal paziente con l'identificativo 4, che non può andare nella clinica i giorni 5 e 6 e che in quei due giorni a due visite, entrambe richiedenti i medicinali 1 e 3, perchè ricordiamo che l'atomo x è composto dai campi: ID paziente, giorno seduta, ora seduta, medicinale 1,2 e 3, priorità e numero della seduta.
I medicinali 1 e 3 richiedono rispettivamente di 20 e 30 e occupano uno slot di tempo a testa e richiedono un'infermiere ed una sedia. Quindi dobbiamo cercare un giorno con un buco di due ore e che abbia la disponibilità di medicinale.

L'utilizzo di medicinale nei vari giorni nel calendario originario è raffigurato nella tabella 5.11 e la quantità a disposizione ogni giorno è rispettivamente: 300, 1600 e 450.

Come si vede dalla tabella sia il giorno 7 che il giorno 8 potremmo mettere le due sedute da spostare, per poter assegnare le sedute in quei giorni bisogna ora controllare la disponibilità di sedie ed infermieri.
Per farlo si possono vedere i dati relativi all'utilizzo o i grafici, in questo esempio, avendo solo 20 pazienti, nelle giornate 7 ed 8 vi sono poche persone, come si può vedere nel grafico dell'utilizzo delle sedie 5.9. Quindi le giornate 7 ed 8 sono valide per avere assegnate le due sedute da spostare.

Questo esempio riassume la modalità con cui la nostra soluzione controlla che ci sia la possibilità o meno di aggiungere una seduta in un dato giorno, avendo noi analizzato un caso

Tabella 5.11 Utilizzo medicinali nei vari giorni.

Giorno	Quantità medicina 1 usata	Quantità medicina 2 usata	Quantità medicina 3 usata
1	300	1500	450
2	240	1600	420
3	220	1600	420
4	200	1600	420
5	200	1600	420
6	40	400	120
7	0	100	30
8	120	500	120
9	20	100	60
10	0	100	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0

con solo 20 pazienti, per rendere l'esempio meno complesso, è stato molto semplice trovare un giorno libero e controllare che vi fossero disponibilità di sedie ed infermieri, con un calendario più fitto di sedute, quindi con più pazienti, trovare un buco nelle giornate diventa più complicato, e può diventare necessario spostare le sedute di uno o più pazienti.

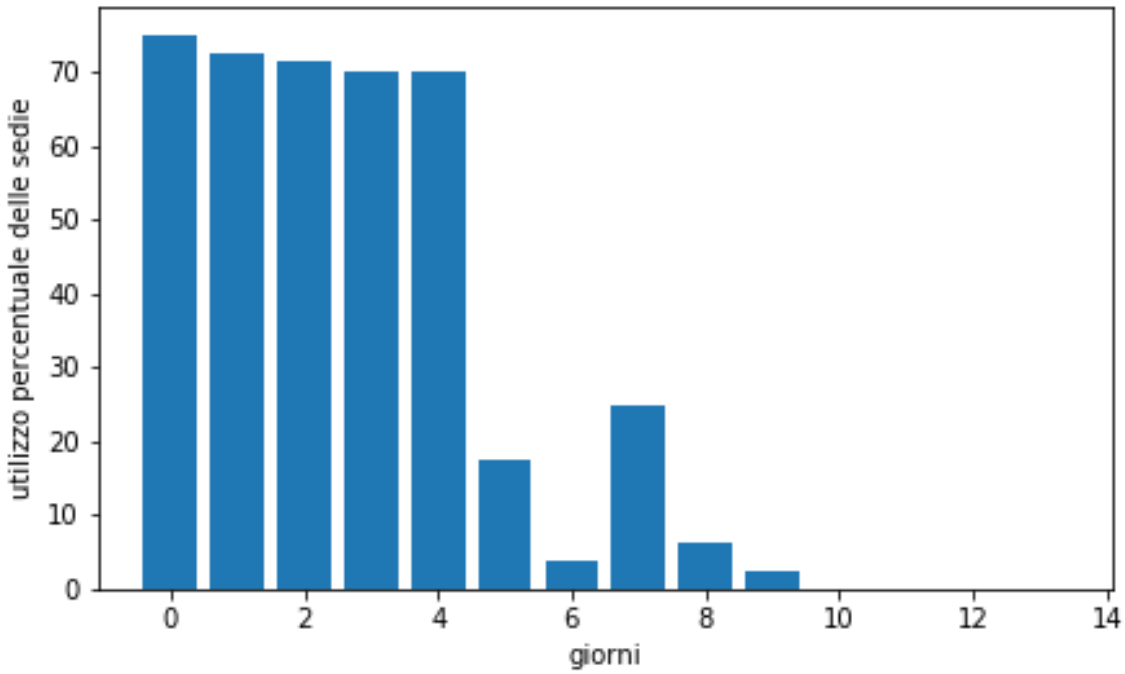


Figura 5.9 Giornate programmate nei giorni in cui i pazienti sono indisponibili

Capitolo 6

Stato dell'arte e conclusioni

In questa sezione analizziamo lo stato dell'arte, separando il tutto in tre paragrafi. Nel primo paragrafo presentiamo alcuni lavori pubblicati negli ultimi anni riguardanti il problema della pianificazione in generale e studi riguardanti le tecniche che abbiamo utilizzato per il nostro modello, nel secondo presentiamo gli articoli che hanno trattato il problema della pianificazione delle sedute oncologiche con strumenti diversi rispetto al linguaggio ASP, mentre nel terzo presentiamo altri problemi di pianificazione, non riguardanti i trattamenti dei pazienti oncologici, sviluppati utilizzando ASP.
Infine proviamo a trarre alcune conclusioni sul lavoro presentato e alcuni possibili sviluppi futuri, sia per quanto riguarda aspetti più operativi sia che per possibili nuovi approfondimenti di carattere più accademico.

6.1 Risoluzione problemi di pianificazione

Aringhieri et al. (2015) tratta il problema della pianificazione delle sedute chirurgiche sia dei pazienti con degenza sia per i day surgery. Affronta il problema utilizzando un metodo a due fasi. Ha l'obbiettivo sia di minimizzare i tempi di attesa dei pazienti che massimizzare l'utlizzo delle risorse dell'ospedale.

Heydari and Soudi (2015) propone una soluzione all'incertezza generata dalla pianificazione in ambito ospedaliero. Oltre a pianificare gli interventi dei pazienti già registrati cerca di pianificare le sedute dei pazienti in modo tale da poter assorbire l'arrivo di nuovi eventuali pazienti, dando quindi valore alle pianificazioni 'robuste'.

Kumar and Dey (2020); Sud et al. (2020) hanno indagato gli effetti del coronavirus sui pazienti oncologici, in particolare come il coronavirus, e i conseguenti ritardi nelle sedute possano influenzare negativamente la riuscita delle cure e il tasso di sopravvivenza dei pazienti.

6.2 Pianificazione di sedute oncologiche

Sevinc et al. (2013) affronta il problema tramite un approccio a due-fasi. Nella prima fase un algoritmo di pianificazione adattivo a feedback-negativo é adottato per controllare il peso nel sistema, mentre nella seconda fase vengono utilizzate due euristiche basate sul 'Problema dello zaino multiplo' per assegnare ai pazienti le sedie di infusione. Il sistema é stato testato in un centro di chemeoterapia locale e ha restituito tempi di attesa e utilizzo delle sedie di infusione buoni.

Huang et al. (2017) sviluppa e implementa un modello per ottimizzare la sicurezza e l'efficienza in termini di violazione delle risorse dello staff, misurate con il rapporto infermiere-pazienti durante una giornata e in specifici durante un trattamento per decidere quando schedulare i pazienti secondo il loro tempo di visita. Il modello ottimizza il risultato utilizzando Excel Solver.

Hahn-Goldberg et al. (2014) utili in situazioni di incertezza che si verificano da richieste di appuntamenti che arrivano in tempo reale e da modifiche a sedute già pianificate utilizzando un template creato a partire da una giornata attesa. Nel caso una nuova registrazione non fosse soddisfatta dal template precedentemente creato questo verrebbe aggiornato.

Turkcan et al. (2010) pianificano le sedute dei pazienti oncologici considerando vincoli sulle risorse minimizzando il tempo di idle time degli infermieri e minimizzando i ritardi sui trattamenti dei vari pazienti.

Huggins et al. (2014) presentano un problema di ottimizzazione mixed-integer sviluppato con l'obbiettivo di massimizzare l'utilizzo delle risorse a disposizione, bilanciando il carico di lavoro delle risorse umane, in particolare tenedo in considerazione la diversa durata dei trattamenti, aumento delle richieste di pazienti e vincoli nella disponibilità delle risorse.

6.3 ASP nei problemi di pianificazione

Calimeri et al. (2014); Gebser et al. (2017, 2020) negli ultimi anni la diffusione e l'efficacia dei solver per problemi ASP è incrementata anche grazie alle diverse competizioni che sono state organizzate negli anni.

Calimeri et al. (2016); Gebser et al. (2017) assegna dei job ai vari device in modo tale da non creare sovrapposizione durante l'esecuzione di questi job.

Ricca et al. (2012) si pone l'obbiettivo di organizzare il personale di un porto per affrontare il lavoro richiesto dalle navi in arrivo tenendo in considerazione alcuni vincoli come: distribuzione dei lavori bilanciata, turnover nei lavori più pericolosi e lavoratori assegnati a lavori che richiedono skills adeguate alle loro.

Alviano et al. (2017, 2018); Dodaro and Maratea (2017) dove si pone l'obbiettivo di creare una pianificazione riguardante i turni degli infermieri tenendo in considerazione alcuni vincoli tramite. Oltre al problema della pianificazione viene affrontato anche il problema della ripianificazione in modo da poter gestire cambiamenti inaspettati nella turnazione degli infermieri.

Amendola (2018a) propone una soluzione per il problema dell'ISG (Interdependet Scheduling Games), nel quale ogni giocatore controlla alcuni dei servizi e può attivarli indipendentemente. I servizi diventano profittevoli nel momento in cui gli altri servizi da cui è dipendente sono già stati attivati. L'obbiettivo del problema è trovare una pianificazione che massimizzi i profitti.

Amendola et al. (2016) si pone l'obbiettivo di risolvere il problema del CPAP (Conference Paper Assignment Problem), che consiste nell'assegnare i reviewer ai vari articoli che vengono proposti all'interno di una conferenza. La soluzione proposta è stata testata su dati reali provenienti che ne attestano i buoni risultati.

Amendola (2018b) propone una soluzione in ASP del problema del SRP (Stable Roommates Problem), che é una versione alternativa al noto problema del Stable Marriage Problem, che consiste nel cercare di accoppiare un set di 2n persone tenendo in considerazione l'ordine di preferenza che le persone hanno datto alle altre.

Alviano et al. (2020) analizza diversi problemi in ambito healthcare, partendo da problemi standard come pianificazione di sale operatorie e di nurse scheduling e presenta soluzioni che sono state sviluppate con l'ASP per migliorare le soluzioni giá esistenti e ulteriori problemi che sono stati affrontati con l'ASP.

6.4 Conclusioni

Dopo aver studiato e iniziato a formulare i vincoli caratteristici del problema della pianificazione, abbiamo generato diversi scenari da utilizzare come Input per le analisi sperimentali. Abbiamo deciso di efettuare le analisi su diversi gruppi di pazienti, differenziati dal numero complessivo, e due diversi scenari: uno con abbondanza di medicinali e uno con scarsità di medicinali.
La soluzione che abbiamo presentato è stata scritta utilizzando l'Answer Set Programming, cercando di ottenere una soluzione che potesse essere al contempo buona e ottenuta in tempi ragionevoli. Abbiamo quindi effettuato diverse analisi dei risultati.
Le analisi dei risultati sono state fatte prendendo in considerazione tutti gli scenari che abbiamo generato. Per ognuno dei risultati abbiamo analizzato l'utilizzo delle risorse, quindi: sedie e medicinali e abbiamo analizzato la pianificazione finale, controllando che i pazienti a priorità maggiori iniziassero i trattamenti prima rispetto agli altri pazienti.
Successivamente siamo passati allo studio della ripianificazione e alla generazione di alcuni Input da testare.
Abbiamo analizzato la ripianificazione controllando che gli spostamenti dei pazienti avvenissero effettivamente nel primo slot temporale libero rispettando comunque lo schema delle sedute.
Dopo queste analisi possiamo trarre le conclusioni riguardo ai risultati ottenuti.
Il modello da noi presentato permette di rispettare i requisiti richiesti ed è abbastanza flessibile da poter affrontare problemi dell'ultima ora tramite la ripianificazione.
Il modello ottiene ottimi risultati soprattutto per quanto riguarda lo sfruttamento delle risorse, e di conseguenza riesce a massimizzare in modo soddisfacente anche la pianificazione dei pazienti.
Per quanto ottenuto nei risultati possiamo quindi dire che ASP è uno strumento di AI valido per affrontare problemi di pianificazione complessi, grazie alla possibilità di modellare le regole e i vincoli in modo da affrontare le problematiche reali e grazie alla presenza di solver efficienti.

Perchè il nostro modello possa essere usato efficacemente da una clinica deve prima di tutto essere di facile utilizzo.
Al momento il modello può essere eseguito solo tramite linea di comandi e le istanze utilizzate, così come i risultati, sono salvati sotto forma di file tetuali. Successivamente per i risultati ottenuti abbiamo costruito un parser che ci ha permesso di ottenere delle tabelle csv contenenti i risultati che rendono più facile la visualizzazione dello sfruttamento delle risorse e la distribuzione dei pazienti.
In un contesto reale la soluzione dovrebbe poter essere utilizzabile tramite un'interfaccia, web o meno, che permetta di modificare agilmente anche alcuni dei parametri da noi presentati: il numero di infermieri, il numero di sedie, le quantità di medicinale e il numero massimo di pazienti visitabili in un'ora da ogni infermiere. Permetteremo inoltre la possibilità di utilizzare istanze di input direttamente da un database, che potrebbe rappresentare la lista di attesa di una clinica.
Per quanto riguarda invece altri sviluppi possibili per ampliare lo studio del problema vi sono alcune possibili strade.
Possiamo studiare il problema utilizzando diversi vincoli, che possono emergere a seconda delle diverse cliniche; si possono utilizzare diversi schemi di trattamento per meglio rappresentare uno scenario reale ed infine si possono considerare scenari diversi da quelli da noi considerati, quindi una maggior disponibilità di risorse, o una diversa quantità di pazienti o anche una pianificazione effettuata su un arco temporale diverso rispetto a quello da noi preso in considerazione.
Per quanto riguarda l'utilizzo di ASP invece si potrebbe provare ad utilizzare un solver diverso per ottenere la soluzione, in particolare si potrebbe utilizzare WASP, già testato in articoli come Alviano et al. (2019), che potrebbe permettere di ottenere una soluzione migliore in tempi minori o potremmo convertire la nostra soluzione in formati che possano essere poi utilizzati con SAT solvers (ad esempio Alviano et al. (2020)) sfruttando la stretta relazione che intercorre tra ASP e SAT (Giunchiglia et al. (2008); Giunchiglia and Maratea (2005)).

Bibliografia

Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., and Ricca, F. (2019). Evaluation of disjunctive programs in WASP. In Balduccini, M., Lierler, Y., and Woltran, S., editors, *LPNMR*, volume 11481 of *LNCS*, pages 241–255. Springer.

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

Alviano, M., Dodaro, C., and Maratea, M. (2017). An advanced answer set programming encoding for nurse scheduling. In *AI*IA*, volume 10640 of *LNCS*, pages 468–482. Springer.

Alviano, M., Dodaro, C., and Maratea, M. (2018). Nurse (re)scheduling via answer set programming. *Intelligenza Artificiale*, 12(2):109–124.

Amendola, G. (2018a). Preliminary results on modeling interdependent scheduling games via answer set programming. In *RiCeRcA@AI*IA*, volume 2272 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Amendola, G. (2018b). Solving the stable roommates problem using incoherent answer set programs. In *RiCeRcA@AI*IA*, volume 2272 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Amendola, G., Dodaro, C., Leone, N., and Ricca, F. (2016). On the application of answer set programming to the conference paper assignment problem. In *AI*IA*, volume 10037 of *Lecture Notes in Computer Science*, pages 164–178. Springer.

Aringhieri, R., Landa, P., Soriano, P., Tanfani, E., and Testi, A. (2015). A two level metaheuristic for the operating room scheduling and assignment problem. *Computers and Operations Research*, 54:21–34.

Buccafurri, F., Leone, N., and Rullo, P. (2000). Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860.

Calimeri, F., Gebser, M., Maratea, M., and Ricca, F. (2014). The design of the fifth answer set programming competition. *CoRR*, abs/1405.3710.

Calimeri, F., Gebser, M., Maratea, M., and Ricca, F. (2016). Design and results of the Fifth Answer Set Programming Competition. *Artificial Intelligence*, 231:151–181.

Dodaro, C. and Maratea, M. (2017). Nurse scheduling via answer set programming. In *LPNMR*, volume 10377 of *LNCS*, pages 301–307. Springer.

Faber, W., Pfeifer, G., and Leone, N. (2011). Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298.

Gebser, M., Maratea, M., and Ricca, F. (2017). The sixth answer set programming competition. *Journal of Artificial Intelligence Research*, 60:41–95.

Gebser, M., Maratea, M., and Ricca, F. (2020). The seventh answer set programming competition: Design and results. *Theory Pract. Log. Program.*, 20(2):176–204.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium , Seattle, Washington, August 15-19, 1988 (2 Volumes)*, pages 1070–1080. MIT Press.

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

Giunchiglia, E., Leone, N., and Maratea, M. (2008). On the relation among answer set solvers. *Ann. Math. Artif. Intell.*, 53(1-4):169–204.

Giunchiglia, E. and Maratea, M. (2005). On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels). In *ICLP*, volume 3668 of *LNCS*, pages 37–51. Springer.

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

Heydari, M. and Soudi, A. (2015). Predictive / reactive planning and scheduling of a surgical suite with emergency patient arrival. *Journal of medical systems*, 40:30.

Huang, Y.-L., Bryce, A. H., Culbertson, T., Connor, S. L., Looker, S. A., Altman, K. M., Collins, J. G., Stellner, W., McWilliams, R. R., Moreno-Aspitia, A., Ailawadhi, S., and Mesa, R. A. (2017). Alternative Outpatient Chemotherapy Scheduling Method to Improve Patient Service Quality and Nurse Satisfaction. *Journal of Oncology Practice*.

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

Kumar, D. and Dey, T. (2020). Treatment delays in oncology patients during COVID-19 pandemic: A perspective. *J Glob Health*, 10(1):010367–010367. Publisher: International Society of Global Health.

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

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

Sud, A., Jones, M. E., Broggio, J., Loveday, C., Torr, B., Garrett, A., Nicol, D. L., Jhanji, S., Boyce, S. A., Gronthoud, F., Ward, P., Handy, J. M., Yousaf, N., Larkin, J., Suh, Y.-E., Scott, S., Pharoah, P. D. P., Swanton, C., Abbosh, C., Williams, M., Lyratzopoulos, G., Houlston, R., and Turnbull, C. (2020). Collateral damage: the impact on outcomes from cancer surgery of the COVID-19 pandemic. *Annals of Oncology*, 31(8):1065–1074. Publisher: Elsevier.

Turkcan, A., Zeng, B., and Lawley, M. (2010). Chemotherapy operations planning and scheduling. *IIE Thansactions on Healthcare Systems Engineering*, 2.

Ringraziamenti

Alla fine della tesi vorrei ringraziare il professore Marco Maratea, che mi ha seguito dall'inizio del lavoro della tesi e sulla sua scrittura.
Vorrei inoltre ringraziare anche il Dott. Galatà e il Dott. Dodaro che mi hanno aiutato dandomi consigli sullo sviluppo della tesi.
Ringrazio i miei famigliari per il supporto e la mia fidanzata Francesca per il sostegno datomi; così come tutti i miei amici in università, sia quelli conosciuti durante gli anni della triennale che i nuovi amici della magistrale, che hanno reso questi anni di studio più semplici.
Infine ringrazio tutti i miei amici al di fuori dell'università che da sempre mi accompagnano.