

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



**Università
di Genova**

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

Anno Accademico 2020/2021

**Tecniche di Intelligenza Artificiale e strumenti applicativi
per la ripianificazione di sessioni riabilitative ospedaliere**

Candidato

Nicholas Nisopoli

Relatore

Prof. Marco Maratea

Correlatore

Dott. Giuseppe Galatà

Indice

1	Introduzione	1
1.1	Contesto e motivazioni	1
1.2	Stato dell'arte	2
1.3	Contributi della tesi	3
1.4	Struttura della tesi	4
2	Definizione del problema	5
2.1	Presentazione	5
2.2	Input e output	7
2.3	Scenari	9
3	Answer Set Programming	11
3.1	Sintassi	11
3.2	Semantica	13
3.3	Abbreviazioni sintattiche	14
3.3.1	Variabili anonime	14
3.3.2	Choice Rules	14
4	Codifica della pianificazione	15
4.1	Codifica dell'assegnamento	15
4.2	Codifica dell'agenda	17
5	Ripianificazione	19
5.1	Requisiti	19
5.1.1	Requisiti per il riassegnamento	19
5.1.2	Requisiti per la ripianificazione dell'agenda	20
5.2	Codifica del riassegnamento	20
5.2.1	Input in comune con l'assegnamento	20
5.2.2	Input per il riassegnamento	21
5.2.3	Output	21

5.2.4	Operatore non disponibile	21
5.2.5	Operatore non disponibile e paziente non presente	23
5.2.6	Riassegnamento parziale	23
5.2.7	Input modificato	24
5.3	Codifica della ripianificazione dell'agenda	25
5.3.1	Input in comune con la pianificazione	25
5.3.2	Input per la ripianificazione	25
5.3.3	Output	26
5.3.4	Operatore non disponibile	26
5.3.5	Paziente non presente	29
5.3.6	Operatore non disponibile e paziente non presente	31
5.3.7	Indisponibilità e assenza con orari parziali	33
5.3.8	Input modificato	33
6	Analisi sperimentali	36
6.1	Riassegnamento dei pazienti	36
6.1.1	Altri approcci	39
6.1.2	Riassegnamento parziale	39
6.2	Ripianificazione dell'agenda	41
6.2.1	Scenario 1: operatore non disponibile	41
6.2.2	Scenario 2: paziente non presente	43
6.2.3	Scenario 3: operatore non disponibile e paziente non presente . . .	45
6.2.4	Scenario 4: indisponibilità e assenza parziali	48
7	Applicazione Web	50
7.1	Graphical User Interface	51
7.2	Database	53
7.3	API RESTful	55
8	Lavori correlati	57
8.1	Problemi di ripianificazione	57
8.2	Ripianificazione di sessioni riabilitative	58
8.3	ASP nei problemi di ripianificazione	59
8.4	GUI per il supporto dell'ASP	59
9	Conclusioni e lavori futuri	60
9.1	Conclusioni	60
9.2	Lavori futuri	61
	Elenco delle figure	iv

Elenco delle tabelle	v
Bibliografia	x

Capitolo 1

Introduzione

1.1 Contesto e motivazioni

La riabilitazione costituisce il terzo pilastro del sistema sanitario, accanto alla prevenzione e alla cura, per il completamento delle attività volte a tutelare la salute dei cittadini.

La riabilitazione è un processo nel quale il paziente viene rieducato in modo che possa raggiungere nuovamente una certa autonomia in seguito a traumi, patologie, o interventi chirurgici. Le categorie dei pazienti trattati possono essere di tipo neurologico, ortopedico, o ambulatoriale. A tutto ciò al giorno d'oggi si aggiungono le problematiche legate al Covid-19. Esso può comportare conseguenze durature, rispetto alla gravità della malattia, oltre a mettere in secondo piano altri problemi sanitari; inoltre, i pazienti post-coronavirus potrebbero dover necessitare del trattamento riabilitativo, poichè il covid lascia anche delle conseguenze a livello neurologico. Un recente articolo [Cieza et al., 2020] stima che 2.45 miliardi di persone (una ogni tre al mondo) potrebbero avere bisogno della riabilitazione in un determinato momento, nel corso della loro vita; inoltre, i cambiamenti attuali di carattere demografico e sanitario potrebbero portare questi numeri a salire: ciò indica quindi un'urgenza nell'attribuire priorità ai servizi di riabilitazione. Il riconoscimento precoce del bisogno di riabilitazione e l'inizio di quest'ultima possono ridurre i costi sanitari e la durata dell'ospedalizzazione, oltre a prevenire la disabilità [Stucki et al., 2005]. Da questo si comprende quanto sia importante anche la realizzazione di una pianificazione efficiente del processo di riabilitazione e che essa possa rispondere al verificarsi di eventi imprevisti, così che i pazienti possano beneficiarne senza ritardi e nel modo più efficace.

I problemi a cui una struttura ospedaliera può andare incontro normalmente sono la mancanza improvvisa di personale medico, o problematiche relative agli orari in cui è possibile essere visitati per i pazienti. Per affrontare queste situazioni è necessario assegnare nuovamente i pazienti agli operatori, o modificare gli orari delle sedute all'interno dell'agenda ospedaliera, tenendo conto di tutte le variabili e i vincoli in gioco. Attualmente questa serie

di operazioni viene eseguita spesso manualmente: se una pianificazione potesse far fronte a questo tipo di eventi in modo automatico si assicurerebbe la soddisfazione, sia dei pazienti che degli operatori sanitari, garantendo una migliore assistenza medica. Gli ospedali come quelli gestiti da ICS Maugeri¹, che trattano centinaia di pazienti con una squadra di solo una decina di operatori sanitari, beneficerebbero nell'uso di questo tipo di ripianificazione. Sarebbe inoltre utile ai coordinatori delle strutture uno strumento, quale un'applicazione web, che permetta l'utilizzo della ripianificazione in modo semplice e automatico; tramite una rappresentazione grafica, anche i meno esperti in campo informatico potrebbero fare uso delle metodologie implementate.

Per concludere, anche se nella tecnologia può essere visto il rischio di una disumanizzazione delle cure e dei rapporti tra medico e paziente, essa può portare solo verso una migliore sanità, se vengono considerate le realtà culturali attuali e vengono soddisfatti i nuovi bisogni dei pazienti [Meskó B, 2017]. Il nuovo fenomeno, conosciuto come *digital health*, ha ormai modificato la pratica della medicina e le modalità con cui vengono forniti i servizi e le cure: con il costante sviluppo tecnologico e i cambiamenti culturali in atto, questo fenomeno è destinato solo a crescere. Un ulteriore impulso a tutto ciò potrà arrivare dal PNRR, che attribuirà un cospicuo finanziamento a sanità e digitalizzazione.

1.2 Stato dell'arte

Il problema della ripianificazione in ambito medico è già stato affrontato con l'utilizzo di diverse tecniche. Nella letteratura si possono trovare diversi esempi, come quello di [Salbert, 2015] che ripianifica gli interventi chirurgici in modo che il tempo per eseguire i restanti interventi sia minimizzato, tramite un'estensione dell'algoritmo del *Longest Processing Time* (LPT) (tempo di processazione più lungo); un altro esempio è [Clark and Walker, 2011] che utilizza una combinazione di somme pesate e programmazione obiettivo.

Altre soluzioni proposte utilizzano algoritmi euristici [Pato and Moz, 2008] [Kitada and Morizawa, 2010], algoritmi paralleli [Zdeněk et al., 2015] e sistemi immunitari artificiali [Maenhout and Vanhoucke, 2013]. Nello specifico della riabilitazione ospedaliera è possibile trovare un esempio di pianificazione nel lavoro di [Schimmelpfeng et al., 2010] che utilizza la programmazione lineare mista a interi (MILPs), ma non implementano nessun tipo di ripianificazione.

È già presente, inoltre, una soluzione [Cardellini et al., 2021] per la pianificazione di sessioni riabilitative che fa uso di tecniche di IA, ma non è ancora stata risolta il problema della ripianificazione. Si può quindi vedere come non siano molte le soluzioni che implementano la ripianificazione e come l'utilizzo di tecniche moderne di Intelligenza Artificiale

¹<https://www.icsmaugeri.it> - ICS Maugeri gestisce Istituti Scientifici e Unità di Riabilitazione all'interno di strutture sanitarie pubbliche e Centri di Ricerca e Prevenzione sul territorio nazionale.

sia limitato.

Se si passa alla ricerca di sistemi supportati anche da un'interfaccia grafica, un esempio è [Huang et al., 2012], che ha sviluppato un sistema, equipaggiato da una GUI, che genera una pianificazione ottima per i pazienti di riabilitazione, minimizzando i tempi di attesa.

Manca, però, nello stato dell'arte, un'applicazione web che implementi tecniche di IA per risolvere questo tipo di problema.

Nel Capitolo 8 verrà approfondita l'analisi dello stato dell'arte.

1.3 Contributi della tesi

La tesi si pone l'obiettivo di risolvere il problema della ripianificazione delle sessioni di riabilitazione, in modo da supportare la struttura ospedaliera tramite uno strumento che possa fornire la soluzione in modo ottimo e automatico.

I principali punti affrontati nella tesi sono i seguenti:

- Per modellare il problema è stato deciso di utilizzare l'Answer Set Programming (ASP) [Gelfond and Lifschitz, 1991, Brewka et al., 2011], una forma di programmazione logica di tipo dichiarativo, in grado di risolvere problemi di ricerca complessi. La scelta di questo particolare linguaggio è dovuta alla sua capacità di risolvere problemi combinatori, che possono includere ottimizzazioni, tramite una rappresentazione naturale e intuitiva della conoscenza ed efficaci strumenti di risoluzione, come *CLINGO* [Gebser et al., 2016]. Inoltre i particolari costrutti del linguaggio facilitano la modellizzazione dei problemi, permettendone l'utilizzo con applicazioni del mondo reale.

Prendendo come punto di partenza la pianificazione, è stato quindi realizzato un modello ASP per la risoluzione del problema della ripianificazione.

- La soluzione è stata testata utilizzando dei dati reali provenienti da ICS Maugeri, in particolare dagli istituti di Genova Nervi e Castel Goffredo. Sono stati presi in considerazione diversi scenari, osservando il comportamento della soluzione al variare di essi; è stato così possibile comprendere le capacità e i limiti del risultato ottenuto. Vengono analizzati principalmente i tempi di esecuzione, che devono essere necessariamente brevi, come è stato già anticipato, per poter provvedere immediatamente ad eventuali imprevisti, disponendo immediatamente di una nuova agenda. Viene poi studiata la qualità del risultato ottenuto, per comprendere come ha agito la soluzione e se essa è conforme ai requisiti.
- Infine, per poter permettere ai coordinatori delle strutture ospedaliere di eseguire la ripianificazione, è stata sviluppata un'applicazione web che grazie al suo facile utilizzo permette anche agli utenti non esperti di sfruttare lo strumento. La soluzione ASP

e il risolutore del linguaggio, CLINGO, sono stati integrati in un'architettura Node.js, mentre è stata creata un'interfaccia grafica tramite Angular. I coordinatori sono quindi in grado di inserire gli input per effettuare la ripianificazione e di visualizzare graficamente, ed in modo intuitivo, la nuova organizzazione delle sessioni.

1.4 Struttura della tesi

Nel Capitolo 2 viene formalizzato e descritto nel dettaglio il problema che sarà affrontato nel corso della tesi. Vengono descritte le caratteristiche delle strutture ospedaliere su cui deve essere modellata la soluzione e viene spiegato l'approccio che viene utilizzato per i diversi scenari delineati.

Nel Capitolo 3 viene presentato il linguaggio ASP, utilizzato per sviluppare la soluzione. Ne vengono studiate le principali caratteristiche, quali la sintassi e la semantica, con particolare risalto ai costrutti che vengono usati in questa tesi.

Nel Capitolo 4 viene mostrata la codifica per eseguire la pianificazione standard, sul cui output deve basarsi la ripianificazione.

Nel Capitolo 5 viene presentata la ripianificazione, con i suoi requisiti e i suoi vincoli, e viene illustrata la codifica ASP realizzata per ottenere la soluzione.

Nel Capitolo 6 è presentata l'analisi delle soluzioni ottenute, mostrando i risultati al variare degli scenari. Vengono analizzati i tempi di esecuzione, rispetto al variare dell'input, e la qualità del risultato.

Nel Capitolo 7 viene presentata l'applicazione web sviluppata per supportare la soluzione ASP realizzata. Vengono poi presentate nel dettaglio le sue principali componenti.

Nel Capitolo 8 vengono introdotti altri sistemi dello stato dell'arte e lavori correlati, confrontandoli con l'approccio usato in questa tesi.

Infine nel Capitolo 9 si giunge alle conclusioni e vengono fatte considerazioni sui possibili lavori futuri.

Capitolo 2

Definizione del problema

In questo capitolo viene definito il problema della ripianificazione, mostrando le principali caratteristiche delle strutture ospedaliere su cui si opera e le problematiche a cui bisogna far fronte. Vengono presentati gli input e gli output del problema e i requisiti delineati. Infine il problema viene scomposto e analizzato in tutti i suoi scenari.

2.1 Presentazione

L'agenda delle sessioni di riabilitazione viene realizzata dai coordinatori delle strutture, su base giornaliera. Nel caso avvenissero delle complicazioni tali per cui fosse necessario modificare l'agenda, questo processo dovrebbe essere ripetuto, tenendo conto delle nuove variabili in gioco; fino ad ora ciò è stato eseguito manualmente, senza nessuna automazione.

Sono stati identificati due eventi che portano alla necessità di eseguire la ripianificazione:

- L'indisponibilità di uno o più operatori
- L'assenza di uno o più pazienti

Studiando questi due eventi si delineano degli scenari, sui quali è stata costruita la soluzione.

Per risolvere il problema devono inoltre essere presi in considerazione molti fattori, quali le caratteristiche dei pazienti e degli operatori, e le tipologie di visite che devono essere portate a termine. Di seguito vengono illustrate le caratteristiche dei tre elementi principali del problema, pazienti, operatori e sessioni, riscontrate all'interno delle strutture ospedaliere trattate:

Pazienti. I pazienti sono caratterizzati da:

- Tipologia (Neurologici, Ortopedici, Covid-19 Positivi, Covid-19 Negativi, Ambulatoriale).
- Necessità di supporto (i.e. necessitano di trattamenti specifici).
- Stato di pagamento (paganti o a carico del Servizio di Sanità Nazionale).
- Orari vietati; intervalli di tempo in cui il paziente non può essere visitato.
- Orari preferiti; preferenza espressa dal coordinatore.
- Operatori preferiti; una lista di operatori a cui il paziente può essere assegnato.
- Sessioni; una lista di sessioni.

Operatori. Gli operatori sono caratterizzati da:

- Qualifica; il tipo di pazienti che può trattare.
- Orari di lavoro; gli orari che può dedicare alle visite dei pazienti. Questi orari sono divisi nei turni di mattina e pomeriggio.

Sessioni. Il coordinatore determina le attività giornaliere del paziente. Queste attività possono essere svolte in una o due sessioni; nel secondo caso queste dovranno essere svolte in due turni differenti. Ogni sessione può essere svolta individualmente, o in supervisione (un operatore supervisiona più pazienti allo stesso tempo). Un operatore può tenere solo una sessione individuale alla volta, ma nel mentre può svolgere più sessioni in supervisione. Alcune sessioni possono essere svolte in parte in individuale e in parte in supervisione, se l'operatore non ha abbastanza tempo per svolgerle completamente in individuale.

Nel caso di indisponibilità di un operatore, le sessioni di quest'ultimo vengono inserite, parzialmente o totalmente, nell'agenda di un altro operatore; nel caso fosse sovraccarico, le sessioni possono essere svolte completamente in supervisione. Nel caso un paziente fosse assente, la sua sessione viene eliminata dall'agenda, o spostata nel caso di assenza parziale. Le rimanenti sessioni in agenda possono essere spostate in accordo alle preferenze del coordinatore.

Le sessioni sono caratterizzate da:

- Modalità (individuale, supervisione).
- Durata minima; la durata minima garantita alle sessioni individuali (non viene considerata nel caso di imprevisti).
- Durata ideale; la durata totale che include anche le parti in supervisione.

L'esecuzione della ripianificazione implica due fasi, così come la pianificazione su cui si basa, che verrà descritta più nel dettaglio nel Capitolo 4.

Le due fasi sono costituite dal tabellone e dall'agenda:

- **Tabellone.** Nella prima fase si affronta il problema di riassegnare i pazienti agli operatori; data l'indisponibilità di un operatore potrebbe essere necessario che dei pazienti debbano essere assegnati ad altri operatori. Si considerano, inoltre, le preferenze espresse e le qualifiche degli operatori. Ci sono poi delle preferenze da rispettare: evitare il sovraccarico del singolo operatore e rispettare la priorità delle preferenze. I requisiti principali del tabellone sono:
 - Deve essere presente compatibilità tra paziente ed operatore, rispetto a tipologia del paziente e specializzazione dell'operatore.
 - Gli operatori non devono essere sovraccaricati.
 - Le preferenze tra pazienti e operatori devono essere rispettate.
- **Agenda.** Nella seconda fase viene assegnato un orario di inizio e di fine per ogni nuova sessione risultante dagli assegnamenti della prima fase. Per fare ciò devono essere rispettate le caratteristiche delle sessioni. Ciò che si deve cercare di ottenere è che l'agenda non venga modificata eccessivamente; allo stesso tempo, però, se alcune sessioni ne avessero la possibilità, possono essere avvicinate ai loro orari preferiti. I requisiti principali dell'agenda sono:
 - Le sessioni devono essere ripianificate rispettando gli orari di lavoro dell'operatore.
 - Per ogni sessione è preferito lo svolgimento in individuale.
 - La nuova agenda deve essere il più simile possibile all'originale.

Le due fasi verranno analizzate più nel dettaglio per ogni scenario delineato, nella Sezione 2.3.

2.2 Input e output

Il problema si basa su una pianificazione già generata, che viene presa come input, con l'obiettivo di realizzare una nuova pianificazione, l'output, in modo che siano rispettati tutti i vincoli e le preferenze. Si osservi in dettaglio gli input e gli output per ognuna delle due fasi descritte:

l'input del riassegnamento si basa sulle caratteristiche di pazienti ed operatori, sugli assegnamenti originali e sui dati di indisponibilità. Nella prima fase, quindi, vengono utilizzati

questi dati per generare i nuovi assegnamenti.

L'output è l'insieme di tutti i nuovi assegnamenti.

Si può osservare in Figura 2.1 un esempio in cui vengono mostrati gli assegnamenti tra operatori e pazienti, appartenenti ad una pianificazione che non può più essere utilizzata, poichè gli operatori 61 e 69 non sono più presenti; vediamo quindi in (b) l'output, che mostra i nuovi assegnamenti dei pazienti, prima assegnati agli operatore 61 e 69, in seguito all'esecuzione di una ripianificazione.

Come già detto, per effettuare la ripianificazione bisogna che vengano rispettati tutti i vin-

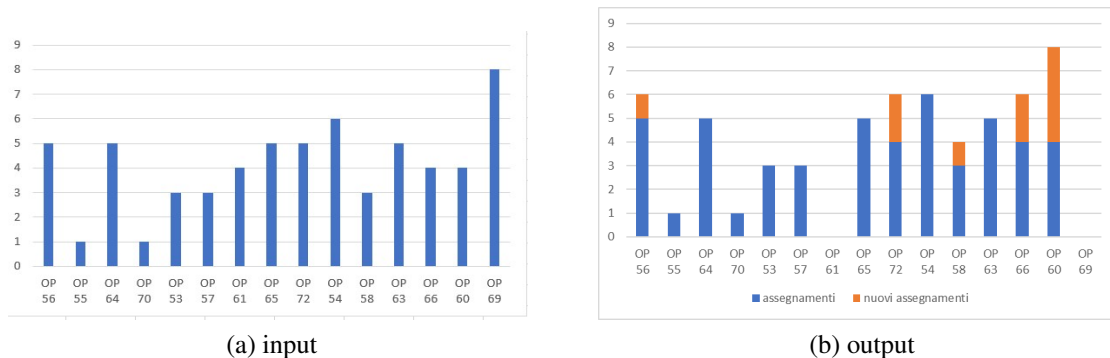


Figura 2.1: Esempio di input e output della fase di riassegnamento

coli, come il rispetto delle qualifiche degli operatori, anch'esse parte dell'input. Si osservi in Tabella 2.1 un esempio: per ogni tipologia di paziente è segnato il numero di operatori che hanno la qualifica per visitarli; ogni operatore può avere più di una qualifica. Inoltre, per ogni tipologia è segnato il numero di pazienti che richiede quel tipo di trattamento. Attraverso questi dati si può intuire la difficoltà che può richiedere realizzare una nuova pianificazione: ad esempio, nel caso mancasse un operatore che tratta la tipologia Ambulatoriale, data la presenza di un alto numero di pazienti di quel tipo; bisogna poi sapere come assegnare i pazienti, nel caso mancasse l'unico operatore qualificato per trattare il Covid-19.

Rispetto ai nuovi assegnamenti generati dall'output del tabellone, vengono generate delle

#	Neurologico	Ortopedico	Covid-19 ++	Ambulatoriale
Operatori	11	7	1	5
Pazienti	21	17	9	24

Tabella 2.1: Numero di operatori per qualifica e numero di pazienti per tipologia a confronto

nuove sessioni che devono essere inserite nell'agenda.

L'input dell'agenda si basa sulle caratteristiche di pazienti ed operatori, sulle sessioni originali, sui dati di indisponibilità e sull'output generato dalla fase precedente.

L'output della ripianificazione è costituito dagli orari e dalle durate delle sessioni e viene usato per generare la nuova agenda.

2.3 Scenari

Come detto in precedenza, rispetto all'evento da affrontare, si inserisce il problema in uno scenario differente. Gli scenari sono i seguenti:

- Operatore non disponibile
- Paziente non presente
- Operatore non disponibile e paziente non presente: la combinazione dei due scenari precedenti
- Indisponibilità e assenze parziali: la combinazione degli scenari precedenti, ma con orari parziali

Ognuno di essi deve essere affrontato in maniera diversa. Analizzandoli in dettaglio: nel primo scenario, nella fase del tabellone vengono riassegnati i pazienti prima assegnati ad operatori non più disponibili, tenendo conto della specializzazione degli operatori e della quantità di ore che già devono svolgere, così da non sovraccaricare il singolo operatore; inoltre, le preferenze paziente-operatore espresse anticipatamente alla pianificazione cercano di essere rispettate. Nella fase dell'agenda si assegna un orario di inizio e di fine alle nuove sessioni risultanti dalla fase precedente, rispettando gli orari di lavoro dell'operatore e gli orari delle altre sessioni già presenti nell'agenda; se nella nuova agenda non ci fosse abbastanza spazio, le nuove sessioni possono essere trasformate in sessioni supervisionate, ovvero possono essere svolte in parallelo ad altre sessioni; il vincolo principale da rispettare è che la nuova agenda sia il più simile possibile a quella originale.

Nel secondo scenario non va eseguita la prima fase, ma solo l'agenda; qui per ogni paziente non presente non vanno più considerate le loro sessioni; inoltre le sessioni nella stessa agenda di una sessione cancellata potrebbero avere la possibilità di essere spostate, data la presenza di nuovo spazio libero, ed essere così avvicinate al loro orario preferito, se già non lo rispettano.

Nel terzo scenario vengono combinati i due scenari precedenti; inoltre nella prima fase viene tenuto conto anche dell'assenza di pazienti per determinare la quantità di ore lavorative di un operatore.

Nell'ultimo scenario le ore di indisponibilità o di assenza possono venire considerate anche parziali: per cui se un operatore non è presente solo per alcune ore di una giornata, solo

alcuni suoi pazienti potrebbero essere riassegnati e nella sua agenda le sessioni devono essere spostate, se si svolgevano nell'orario in cui non è più presente; per un paziente, se non fosse disponibile in certi orari della giornata, la sua sessione deve essere spostata al di fuori di questi orari.

Capitolo 3

Answer Set Programming

Answer Set Programming (ASP) è un paradigma di programmazione sviluppato nel campo della programmazione logica e del reasoning. Esso fa parte della categoria dei cosiddetti linguaggi di programmazione a vincoli. La sintassi assomiglia a quella di Prolog, ma i meccanismi computazionali usati nell'ASP sono differenti. L'idea dell'ASP è di rappresentare un problema computazionale attraverso un programma, il cui set di risposte corrisponde alla soluzione, quindi di usare il solutore per generare il set di risposte del programma: la prima fase, chiamata *grounding*, genera il programma proposizionale, la seconda, *solving*, valuta il programma e genera un *answer set*.

Dal punto di vista della rappresentazione della conoscenza, un insieme di atomi può essere considerato come la descrizione di uno stato di conoscenza completo. Per cui, gli atomi che non fanno parte di questo insieme sono considerati falsi. Un programma ASP tratta quindi una conoscenza incompleta attraverso la *negazione standard*, includendo nell'input del modello solo le informazioni necessarie. Le applicazioni del mondo reale hanno spesso a che fare con conoscenze incomplete e per questo l'ASP è adatto per questo tipo di problemi.

È importante anche studiare la presenza di solutori efficienti [Alviano et al., 2019a] [Gebser et al., 2012], che rendono il linguaggio ideale per affrontare problemi come quello presentato in questa tesi, come è testimoniato dalla serie di competizioni ASP [Gebser et al., 2017, Gebser et al., 2020, Calimeri et al., 2014].

In questo capitolo vengono presentati i principali concetti di questo linguaggio.

3.1 Sintassi

Le variabili sono stringhe che iniziano con la lettera maiuscola, mentre le costanti iniziano con la lettera minuscola. Un *termine* è sia o una variabile o una costante. Un *atomo* è un'espressione nella forma $p(t_1, \dots, t_n)$, dove p è un predicato e t_1, \dots, t_n sono termini.

Un atomo è ground se tutti i termini sono costanti. Un *letterale* è o un atomo p (letterale positivo), o la sua negazione $\text{not } p$ (letterale negativo). Dato un letterale l , $\text{not } l$ rappresenta il suo letterale complementare. Un insieme di letterali L è detto *consistente* se, per ogni letterale $l \in L$, il suo letterale complementare non è contenuto in L .

Le *funzioni aggregato* mappano degli insiemi di tuple di termini ground su termini ground. Le funzioni più comuni implementate in ASP sono:

- *count*, numero di termini
- *sum*, somma di interi
- *max*, valore massimo tra interi
- *min*, valore minimo tra interi

Un *atomo aggregato* ha la forma:

$$\#aggr\{e_1, \dots, e_n\} < u$$

dove e_1, \dots, e_n sono elementi aggregati, della forma $t_1, \dots, t_m : l_1, \dots, l_n$, per $n \geq 0$, $\#aggr$ è una funzione aggregato, $< \in \{<, \leq, >, \geq, =, \neq\}$ è una relazione aggregato e u è un termine, chiamato guardia.

Una *regola disgiuntiva* (in breve, *regola*) r è un costrutto:

$$a_1 \vee \dots \vee a_n :- b_1, \dots, b_k, b_{k+1}, \dots, b_m.$$

dove $a_1, \dots, a_n, b_1, \dots, b_n$ sono letterali e $n \geq 0$, $m \geq k \geq 0$. La disgiunzione $a_1 \vee \dots \vee a_n$ è chiamata la *testa* di r , mentre la congiunzione $b_1, \dots, b_k, b_{k+1}, \dots, b_m$ è il suo *corpo*. Una regola senza letterali nella testa (i.e. $n = 0$) è un *vincolo*. Una regola con precisamente un letterale nella testa (i.e. $n = 1$) è detta *regola normale*. Se il corpo è vuoto (i.e. $m = k = 0$) la regola è un *fatto* e normalmente il simbolo $:-$ è omissso.

Un programma ASP è un insieme finito di regole, le quali è necessario, normalmente, che siano *sicure*. Una regola è sicura se ogni sua variabile appare almeno in un letterale positivo del corpo. Un programma ASP è sicuro se ognuna delle sue regole è sicura.

I problemi di ottimizzazione sono modellati in ASP tramite i *weak constraints* [Buccafurri et al., 2000]. Un weak constraint ω è della forma:

$$:\sim b_1, \dots, b_k, b_{k+1}, \dots, b_m. [w@l]$$

dove w e l sono rispettivamente il peso e il livello di ω . In dettaglio il peso equivale al "costo" pagato nel violare la condizione espressa dal corpo, mentre il livello può essere usato per definire una priorità, nel caso di più weak constraint; il livello è opzionale e può essere omesso. Nel caso di più livelli, vengono considerati come answer sets quelli che minimizzano la somma dei pesi dei vincoli deboli violati con la priorità più alta e, tra questi, vengono scelti quelli che minimizzano la somma dei pesi dei vincoli violati del livello inferiore, e così via. Un programma ASP con weak constraints è $\Pi = \langle P, W \rangle$, dove P è un programma e W è un insieme di weak constraints.

3.2 Semantica

Herbrand Universe e Literal Base. Per ogni programma P , l'Herbrand Universe, indicato da U_p , è l'insieme di tutte le costanti presenti in P . Se non sono presenti costanti in P , U_p consiste in una costante arbitraria. L'Herbrand literal base B_p è l'insieme di tutti i letterali ground derivabili dai simboli predicati in P e le costanti in U_p .

Istanziamento Ground. Per ogni regola r , $Ground(r)$ indica l'insieme di regole ottenute scambiando ogni variabile in r con le costanti contenute in U_p in ogni combinazione possibile. Per un programma P , la sua istanziamento ground è l'insieme $grnd(P) = \bigcup_{r \in P} Ground(r)$.

Interpretazione. Un'interpretazione I è un insieme consistente di letterali ground $I \subseteq B_p$ rispetto a P . Un'interpretazione consistente $X \subseteq B_p$ è detta *chiusa sotto* P se, per ogni $r \in grnd(P)$, $a_1, \dots, a_n \cap X \neq \emptyset$ quando $B_p \subseteq X$.

Modello. È un'interpretazione I che è chiusa sotto P , ovvero che soddisfa tutte le regole di un programma. Un modello è *minimo* se ogni $I^I \subset I$ non è un modello di P .

Answer Sets. Un'interpretazione I per P è un answer set (o modello stabile) di P se I è un modello minimo tra tutte le interpretazioni consistenti che sono chiuse sotto P .

Answer Set Ottimo. Dato un programma con weak constraints $\Pi = \langle P, W \rangle$, la semantica di Π viene estesa dal caso base definito sopra. Sia $G_\Pi = \langle G_P, G_W \rangle$ l'istanziamento di Π ; un vincolo $\omega \in G_W$ è violato da un'interpretazione I se tutti i letterali in ω sono veri rispetto a I . Un answer set ottimo di ω è un answer set di G_P che minimizza la somma dei pesi dei weak constraints violati in G_W con ordine prioritario.

3.3 Abbreviazioni sintattiche

Questa sezione specifica addizionali costrutti del linguaggio che vengono usati all'interno della tesi.

3.3.1 Variabili anonime

Una variabile anonima in una qualunque regola è indicata da "_". Ogni occorrenza del simbolo indica una variabile inutilizzata all'interno del contesto (i.e. le presenze di più variabili anonime rappresentano variabili distinte).

3.3.2 Choice Rules

Un *atomo di scelta* è nella forma:

$$\{e_1; \dots; e_m\} < u$$

dove $e_1; \dots; e_m$ sono elementi di scelta per $m \geq 0$, $<$ è una relazione aggregato e u è un termine. La parte $< u$ può anche essere omessa se $<$ sta per \geq e $u = 0$.

Una choice rule ha la forma:

$$\{e_1; \dots; e_m\} < u :- b_1, \dots, b_n.$$

dove b_1, \dots, b_n sono letterali per $n \geq 0$. Una choice rule implica che, se il corpo della regola è vero, un sottoinsieme arbitrario di $e_1; \dots; e_m$ può essere scelto come vero, così che possa essere soddisfatta la relazione aggregata con u .

La choice rule può essere espressa con la regola:

$$a_i | \hat{a}_i :- b_1, \dots, b_n, l_1, \dots, l_k.$$

per ogni $1 \leq i \leq m$, dove \hat{a}_i rappresenta un atomo ausiliario a cui vengono assegnati gli elementi nel caso a_i sia falso, insieme al vincolo:

$$:- b_1, \dots, b_n, not \#count\{\hat{a}_i : a_1, l_1, \dots, l_k; \hat{a}_m : a_m, l_{1m}, \dots, l_{km}\} < u.$$

La prima regola esprime che l'atomo a_i può essere scelto come vero (o falso) se b_1, \dots, b_n e l_1, \dots, l_m sono veri. Questo genera il sottoinsieme degli elementi di scelta. Il vincolo invece fa sì che venga rispettata la relazione con u , se b_1, \dots, b_n sono veri.

Capitolo 4

Codifica della pianificazione

In questo capitolo viene presentata la codifica ASP con cui viene realizzata la pianificazione delle sessioni di riabilitazione. La codifica per la ripianificazione, presentata nel prossimo capitolo, è basata su di essa.

4.1 Codifica dell'assegnamento

I dati in input in sono descritti dai seguenti atomi:

- *operator(OP,ET,OT,MA,PNPL,NPL,NP,PNP,POPL,OPL,OP,POP,PC,NC,COM)*, che rappresenta un operatore di riabilitazione, caratterizzato da un identificativo (OP), il tempo che può utilizzare per le visite (ET), il tipo di operatore, part-time o no (OT), un numero di pazienti massimo che può visitare (MA), un numero massimo di pazienti paganti di tipo neurologico con necessità di ausilio (PNLP), un numero massimo di pazienti non paganti di tipo neurologico con necessità di ausilio (NLP), un numero massimo di pazienti non paganti di tipo neurologico senza necessità di ausilio (NP), un numero massimo di pazienti paganti di tipo neurologico senza necessità di ausilio (PNP), un numero massimo di pazienti paganti di tipo ortopedico con necessità di ausilio (POPL), un numero massimo di pazienti non paganti di tipo ortopedico con necessità di ausilio (OPL), un numero massimo di pazienti non paganti di tipo ortopedico senza necessità di ausilio (OP), un numero massimo di pazienti paganti di tipo ortopedico senza necessità di ausilio (POP), un numero massimo di pazienti risultati Positivi al Covid-19 (PC), un numero massimo di pazienti risultati Negativi al Covid-19 (NC), un numero massimo di pazienti con macroattività ambulatoriale complessa (COM).
- *patient(ID,TP,STAT,LN,MIN)*, che rappresenta un paziente, caratterizzato da un identificativo (ID), la tipologia di paziente (neurologico, ortopedico, covid+, covid-) (TP),

lo stato, ovvero se è pagante o meno (STAT), l'indicazione del bisogno dell'ausilio del sollevatore o meno (LN), la durata minima della visita di cui ha bisogno (MIN).

- $pref(OP, PAT, W, I)$, che rappresenta la preferenza dell'operatore verso il paziente e viceversa (può essere ereditato dalle pianificazioni passate), caratterizzata da l'identificativo dell'operatore (OP), l'identificativo del paziente (PAT), la priorità della preferenza (W), se è stata ereditata o meno (I).
- $operator(-1, 100, 0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100)$, che rappresenta un "operatore falso", usato per assegnare i pazienti che non sono stati potuti assegnare ad altri operatori.

L'output è composto dagli assegnamenti, rappresentati dagli atomi $assignment(OP, PAT, W, PREF)$, caratterizzati da paziente (PAT), operatore (OP), priorità di assegnamento (W), e un indicazione del rispetto o meno della preferenza (PREF).

La relativa codifica è mostrata nella figura sottostante. La regola 1 si assicura che ogni paziente sia assegnato esattamente ad un operatore. La regola 2 si assicura che il tempo richiesto da un paziente assegnato ad un operatore non superi il tempo massimo stabilito dal suo contratto. La regola 3 si assicura che ogni operatore non superi il numero massimo di pazienti che può visitare durante la giornata. Le regole dalla 4 alla 14 sono simili alla precedente, ma i limiti sono imposti rispetto alla tipologia di paziente.

I vincoli di ottimizzazione dal 15 al 17 sono usati per stabilire le preferenze tra gli assegnamenti. In particolare, il 15 ottimizza gli assegnamenti che rispettano le preferenze di ogni paziente. Il 16 minimizza gli assegnamenti all'operatore fittizio. Infine il 17 ottimizza gli assegnamenti dei pazienti senza preferenze.

```

1 1 {assignment(OP, PAT, W, PREF): operator(OP,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_) , pref(OP, PAT, W
, PREF)} 1 :- patient(PAT,_,_,_,_).
2 :- operator(OP,MAX,_,_,_,_,_,_,_,_,_,_,_,_,_) , #sum {T, PAT: patient(PAT,_,_,_,T) ,
assignment(OP,PAT,_,_)} > MAX.
3 :- operator(OP,_,_,MAX,_,_,_,_,_,_,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_)} >
MAX.
4 :- operator(OP,_,_,_,MAX,_,_,_,_,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 1, 1, 1, _)} > MAX.
5 :- operator(OP,_,_,_,_,MAX,_,_,_,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 1, 2, 1, _)} > MAX.
6 :- operator(OP,_,_,_,_,_,MAX,_,_,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 1, 2, 0, _)} > MAX.
7 :- operator(OP,_,_,_,_,_,_,MAX,_,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 1, 1, 0, _)} > MAX.
8 :- operator(OP,_,_,_,_,_,_,_,MAX,_,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 2, 1, 1, _)} > MAX.
9 :- operator(OP,_,_,_,_,_,_,_,_,MAX,_,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 2, 2, 1, _)} > MAX.
10 :- operator(OP,_,_,_,_,_,_,_,_,_,MAX,_,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 2, 1, 0, _)} > MAX.
11 :- operator(OP,_,_,_,_,_,_,_,_,_,_,MAX,_,_) , #count {PAT: assignment(OP, PAT, _,_),
patient(PAT, 2, 2, 0, _)} > MAX.
```

```

12 :- operator(OP,_,_,_,_,_,_,_,_,_,_,MAX,_,_), #count {PAT: assignment(OP, PAT, _, _),
    patient(PAT, 4, _, _, _)} > MAX.
13 :- operator(OP,_,_,_,_,_,_,_,_,_,_,MAX,_,_), #count {PAT: assignment(OP, PAT, _, _),
    patient(PAT, 5, _, _, _)} > MAX.
14 :- operator(OP,_,_,_,_,_,_,_,_,_,_,MAX,_,_), #count {PAT: assignment(OP, PAT, _, _),
    patient(PAT, 6, _, _, _)} > MAX.
15 :~ #sum{W, PAT: assignment(_, PAT, W, 1)} = N. [N@3]
16 :~ #count{PAT: assignment(-1, PAT, _, _)} = N. [N@2]
17 :~ #sum{W, PAT: assignment(_, PAT, W, 0)} = N. [N@1]

```

4.2 Codifica dell'agenda

I dati in input sono descritti dai seguenti atomi:

- *patient(PAT,AUT,MIN)*, che rappresenta un paziente, caratterizzato da un identificativo (ID), l'indicazione se è autonomo o meno (AUT), la durata minima della sua sessione (MIN).
- *session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP)*, che rappresenta la sessione di riabilitazione, caratterizzata da un identificativo (ID), l'identificativo del paziente (PAT), l'identificativo dell'operatore (OP), l'identificativo del luogo di visita (LOC), la tipologia del paziente, ovvero se necessita di una sessione individuale, supervisionata, o di gruppo. (Una sessione supervisionata può essere portata avanti simultaneamente a una sessione individuale, o ad altre sessioni supervisionate) (T), la durata minima della sessione (MIN), la durata ideale per la sessione (IL), il periodo ideale per la sessione (IP), l'orario ideale per la sessione (IH), l'indicazione per una sessione opzionale o obbligatoria (O), l'indicazione della priorità con cui deve essere considerato l'orario ideale (TP).
- *period(PER,OP,ST,END)*, che modella i turni degli operatori, caratterizzato da il periodo in cui l'operatore svolge il suo turno (PER), l'identificativo dell'operatore (OP), l'orario in cui inizia il turno (ST), l'orario in cui termina il turno (END).
- *time(PER,OP,L)*, che modella i turni degli operatori, caratterizzato da il periodo in cui l'operatore svolge il suo turno (PER), l'identificativo dell'operatore (OP), la durata del turno (L).
- *forbidden(PAT,PER,ST,END)*, che rappresenta gli intervalli di tempo in cui un paziente non è disponibile, caratterizzato da l'identificativo del paziente (PAT), il periodo a cui appartengono gli intervalli temporali (PER), l'orario di inizio di indisponibilità (ST), l'orario di fine di indisponibilità (END).

L'output è rappresentato dagli atomi *start(ID,PER,T)* e *length(ID,PER,L)*, che indicano rispettivamente l'orario di inizio e la durata di ogni sessione.

La relativa codifica è mostrata nella figura sottostante. Le regole 1 e 2 assegnano un tempo di inizio ad ogni sessione; per le sessioni opzionali può non essere assegnato. La regola 3 definisce una durata per ogni sessione; essa non può essere minore del tempo minimo definito e non può essere maggiore della durata ideale. Le regole 4 e 5 riservano per ogni sessione degli slot temporali prima dell'inizio e dopo la fine. Quindi, le regole 6 e 7 definiscono due atomi che usano gli slot riservati per estendere la durata delle sessioni. La regola 8 impone che due sessioni individuali non si sovrappongano. La 9 impone che ogni paziente abbia al massimo una sessione per periodo. La 10 impone che i venga riservato il tempo minimo dei pazienti. Le regole dalla 11 alla 12 impongono che una sessione non abbia luogo durante gli orari proibiti.

Il vincolo di ottimizzazione 13 fa in modo che la durata delle sessioni sia il più vicino possibile a quella ideale. Le regole 14 e 15 minimizzano la distanza tra gli orari scelti e preferiti per le sessioni prioritarie. Le regole 16 e 17 agiscono nello stesso modo di 14 e 15, ma per sessioni non prioritarie.

```

1 1 {start(ID, PER, TS): time(PER, OP, TS)} 1 :- session(ID, _, OP, _, _, _, _, _, 0, _).
2 0 {start(ID, PER, TS): time(PER, OP, TS)} 1 :- session(ID, _, OP, _, _, _, _, _, 1, _).
3 1 {length(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>= MIN, NL<= MAX}
   1 :- start(ID, PER, TS), period(PER, OP, ST, END), session(ID, _, OP, _, _, MIN, MAX,
   _, _, _, _).
4 1 {before(ID, NL): time(PER, OP, L), NL=L-ST, NL<=TS-ST} 1 :- start(ID, PER, TS), period(PER, OP,
   ST, _), session(ID, _, OP, _, _, _, _, _, _).
5 1 {after(ID, NL): time(PER, OP, L), NL=L-ST, NL<=END-TS-LEN} 1 :- start(ID, PER, TS), period(
   PER, OP, ST, END), length(ID, PER, LEN), session(ID, _, OP, _, _, _, _, _, _).
6 extstart(ID, PER, TS - LB) :- start(ID, PER, TS), before(ID, LB).
7 extlength(ID, PER, L + LA + LB) :- length(ID, PER, L), after(ID, LA), before(ID, LB).
8 :- start(ID, PER, TS), length(ID, PER, L), session(ID, _, OP, _, 0, _, _, _, _, _),
   start(ID2, PER, TS2), session(ID2, _, OP, _, 0, _, _, _, _, _), ID!=ID2, TS2>=TS,
   TS2<TS+L.
9 :- start(ID1, PER1, _), session(ID1, PAT, _, _, _, _, _, _, _), start(ID2, PER2, _),
   session(ID2, PAT, _, _, _, _, _, _, _), ID1!=ID2, PER1=PER2.
10 :- patient(ID, _, MIN), #sum{LEN, SES: session(SES, ID, _, _, _, _, _, _), extlength(SES, _,
   LEN)} < MIN.
11 :- forbidden(PAT, PER, ST, END), session(ID, PAT, _, _, _, _, _, _, _), extstart(ID,
   PER, TS), extlength(ID, PER, L), ST<=TS, END>TS.
12 :- forbidden(PAT, PER, _, END), session(ID, PAT, _, _, _, _, _, _, _), extstart(ID,
   PER, TS), extlength(ID, PER, L), END>TS, END<=TS+L.
13 :~ length(ID, _, L), session(ID, _, _, _, _, OPT, _, _, _), D = |L - OPT|. [D@5, ID
   ]
14 :~ start(ID, PER, _), session(ID, _, _, _, 0, _, _, OPT, _, 1), D = |PER - OPT|. [D@4,
   ID]
15 :~ start(ID, PER, TS), session(ID, _, _, _, 0, _, _, PER, OPT, 1), D = |TS - OPT|. [D@3,
   ID]
16 :~ start(ID, PER, _), session(ID, _, _, _, 0, _, _, OPT, _, 0), D = |PER - OPT|. [D@2,
   ID]
17 :~ start(ID, PER, TS), session(ID, _, _, _, 0, _, _, PER, OPT, 0), D = |TS - OPT|. [D@1,
   ID]

```

Capitolo 5

Ripianificazione

In questo capitolo viene presentata la soluzione, sviluppata in linguaggio ASP, della ripianificazione delle sessioni riabilitative. Vengono illustrati i requisiti, gli atomi in input che modellano il problema e il relativo output. Infine, vengono descritte le regole del modello per ogni scenario.

Questa soluzione viene applicata su una pianificazione già generata, che per qualche motivo non può essere più utilizzata appieno.

5.1 Requisiti

5.1.1 Requisiti per il riassegnamento

Il riassegnamento deve rispettare i seguenti requisiti:

Vincoli

- Ogni paziente prima assegnato ad un operatore non più disponibile, deve essere riassegnato ad un singolo operatore disponibile.
- Gli assegnamenti devono considerare le specializzazioni degli operatori.
- La preferenza di un operatore da parte di un paziente e viceversa deve essere rispettata.

Preferenze

- I pazienti devono essere distribuiti in modo da non sovraccaricare il singolo operatore.
- L'assegnazione rispetto alle preferenze deve seguire la priorità.

5.1.2 Requisiti per la ripianificazione dell'agenda

La ripianificazione deve rispettare i seguenti requisiti:

Vincoli

- Ad ogni nuova sessione deve essere assegnato un orario di inizio ed una durata.
- Se una sessione non può essere inserita come individuale, deve essere inserita come supervisionata.
- Le sessioni individuali non possono essere sovrapposte tra loro.
- Le sessioni di gruppo non possono essere sovrapposte a sessioni di altro tipo.
- Due sessioni dello stesso paziente possono essere previste solo in mezze giornate diverse.
- Il tempo minimo di visita di un paziente (individuale + supervisione, contando tutte le sessioni) deve essere rispettato.
- Gli orari ed i turni dell'operatore devono essere rispettati.
- Gli orari vietati per i pazienti devono essere rispettati.
- Le sessioni cancellate dal paziente non devono più essere considerate.

Preferenze

- Per ogni nuova sessione individuale è preferito l'inserimento come individuale.
- Gli orari delle nuove sessioni devono essere il più simile possibile a quelli delle sessioni corrispondenti nell'agenda originale.
- Se una sessione non era nel suo orario preferito, bisogna provare ad avvicinarla a questo orario, se sono state eliminate delle sessioni dall'agenda di appartenenza.

5.2 Codifica del riassegnamento

5.2.1 Input in comune con l'assegnamento

I dati in input in comune con la codifica di assegnamento sono descritti dagli atomi presentati nel Capitolo 4, Sezione 4.1.

5.2.2 Input per il riassetto

I dati in input per il riassegnamento sono descritti dai seguenti atomi:

- *assignment(OP,PAT,W,PREF)*, che rappresenta l'assegnamento originale di un paziente ad un operatore, caratterizzato da l'identificativo dell'operatore (OP), l'identificativo del paziente (PAT), la priorità dell'assegnamento (W), il rispetto o meno della preferenza (PREF). Questi atomi sono ottenuti dall'output dell'assegnamento.
- *unavailable(OP)*, che rappresenta l'indisponibilità di un operatore, caratterizzata dall'identificativo dell'operatore (OP).
- *absent(PAT)*, che rappresenta l'assenza di un paziente, caratterizzata dall'identificativo del paziente (PAT).

5.2.3 Output

L'output è composto dagli atomi nella forma $newAssignment(OP, PAT, W, PREF)$, che rappresentano i nuovi assegnamenti dei pazienti (PAT) agli operatori (OP), con una priorità di assegnamento (W), e un indicazione del rispetto o meno della preferenza (PREF). I pazienti, prima assegnati ad operatori non più disponibili, saranno quindi stati assegnati a nuovi operatori.

5.2.4 Operatore non disponibile

Di seguito vengono presentate le regole per effettuare il riassegnamento nello scenario in cui sia presente solo l'indisponibilità dell'operatore.

Regole per il riassetto

Tramite la regola I vengono create le nuove istanze $newAssignment(OP, PAT, W, PREF)$, che costituiranno l'output. Per ogni paziente che era stato assegnato in precedenza ad un operatore ora non disponibile, ovvero per cui esiste un'istanza dell'atomo *unavailable*, viene effettuato un singolo assegnamento ad un operatore disponibile.

```

1 1 {newAssignment(OP, PAT, W, PREF): operator(OP,_,_,_,_,_,_,_,_,_,_,_),
    pref(OP, PAT, W, PREF), not unavailable(OP)} 1 :- patient(PAT,_,_,_,_),
    assignment(OPN,PAT,_,_), unavailable(OPN).

```

Con le seguenti due regole viene creato il nuovo atomo *finalAssignment* il quale raggruppa tutti gli assegnamenti validi, ovvero quelli originali ancora validi e quelli nuovi realizzati da questa codifica. Questi atomi verranno usati per i controlli di ottimizzazione.

```

2  finalAssignment(OP,PAT,W,PREF) :- assignment(OP,PAT,W,PREF), not unavailable(OP)
.
3  finalAssignment(OP,PAT,W,PREF) :- newAssignment(OP,PAT,W,PREF).

```

```

4 :- operator(OP,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 1, 1, 1, _), SP < 1.
5 :- operator(OP,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 1, 2, 1, _), SP < 1.
6 :- operator(OP,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 1, 2, 0, _), SP < 1.
7 :- operator(OP,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 1, 1, 0, _), SP < 1.
8 :- operator(OP,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 2, 1, 1, _), SP < 1.
9 :- operator(OP,_,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 2, 2, 1, _), SP < 1.
10 :- operator(OP,_,_,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 2, 1, 0, _), SP < 1.
11 :- operator(OP,_,_,_,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 2, 2, 0, _), SP < 1.
12 :- operator(OP,_,_,_,_,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 4, _, _, _), SP < 1.
13 :- operator(OP,_,_,_,_,_,_,_,_,_,_,_,_,SP,_,_,_,_,_,_,_), newAssignment(OP, PAT, _, _),
    patient(PAT, 5, _, _, _), SP < 1.
14 :- operator(OP,_,_,_,_,_,_,_,_,_,_,_,_,_,SP), newAssignment(OP, PAT, _, _),
    patient(PAT, 6, _, _, _), SP < 1.

```

Dapprima è stato tentato l'uso di questo vincolo di ottimizzazione, con priorità minore, che fa in modo che venga minimizzato il numero di nuovi assegnamenti ad un singolo operatore. Ciò permette di effettuare una distribuzione equa dei pazienti tra gli operatori possibili.

In seguito si è deciso di affrontare uno scenario più realistico, tenendo conto dei tempi di lavoro degli operatori e del numero dei pazienti già assegnato ad ognuno di essi. Con questo nuovo vincolo si minimizza la differenza tra il tempo massimo di lavoro di un operatore e la somma dei tempi minimi necessari a trattare ogni paziente assegnato al dato operatore. In questo modo si cercherà di assegnare i pazienti agli operatori con più tempo libero a disposizione, sempre tenendo conto della loro specializzazione.

```
:~ operator(OP,ET,_,_,_,_,_,_,_,_,_,_,_,_), #sum{T, PAT: patient(PAT,_,_,_,T)
, finalAssignment(OP,PAT,_,_) = N, DF = |ET - N|. [DF@1]
```

Ulteriori regole di ottimizzazione sono utilizzate per gestire gli assegnamenti; in ordine di priorità, dalla più bassa alla più alta: la *16* ottimizza gli assegnamenti per pazienti senza preferenze di operatore; la *17* minimizza il numero di assegnamenti all'operatore fittizio; la *17* ottimizza gli assegnamenti per pazienti con preferenze di operatore.

```

:~ #sum{W, PAT: newAssignment(_, PAT, W, 0)} = N. [N@2]
:~ #count{PAT: newAssignment(-1, PAT, _, _)} = N. [N@3]
:~ #sum{W, PAT: newAssignment(_, PAT, W, 1)} = N. [N@4]

```

5.2.5 Operatore non disponibile e paziente non presente

Di seguito sono presentate le regole di riassegnamento per lo scenario in cui allo stesso tempo non siano presenti operatore e paziente.

Regole per il riassegnamento

Tramite la regola 1 vengono riassegnati i pazienti sempre secondo la regola dello scenario precedente, ma in questo caso viene controllato che il paziente non sia assente; nel qual caso, non verrà effettuato nessun riassegnamento. La regola 2 non considera più gli assegnamenti relativi a pazienti che non sono più presenti nella giornata di visite; in questo modo, nei vincoli di ottimizzazione non verranno più conteggiati questi pazienti nel calcolo delle ore occupate degli operatori.

```

1 {newAssignment(OP, PAT, W, PREF): operator(OP,_,_,_,_,_,_,_,_,_,_,_,_,_) ,
    pref(OP, PAT, W, PREF), not unavailable(OP)} 1 :- patient(PAT,_,_,_,_) ,
    assignment(OPN,PAT,_,_) , unavailable(OPN) , not absent(PAT).
finalAssignment(OP,PAT,W,PREF) :- assignment(OP,PAT,W,PREF) , not unavailable(OP)
    , not absent(PAT).

```

Le restanti regole sono le stesse descritte per lo scenario precedente.

5.2.6 Riassegnamento parziale

Di seguito sono riportate le regole per effettuare il riassegnamento nel caso un operatore e/o un paziente esprimano un'indisponibilità parziale.

5.3 Codifica della ripianificazione dell'agenda

5.3.1 Input in comune con la pianificazione

I dati in input in comune con la codifica di pianificazione sono descritti dagli atomi presentati nel Capitolo 4, Sezione 4.2.

5.3.2 Input per la ripianificazione

I dati in input per la ripianificazione sono descritti dai seguenti atomi:

- *session*(*ID*,*PAT*,*OP*,*LOC*,*T*,*MIN*,*IL*,*IP*,*IH*,*O*,*TP*), che rappresenta la sessione di riabilitazione, caratterizzata da un identificativo (*ID*), l'identificativo del paziente (*PAT*), l'identificativo dell'operatore (*OP*), l'identificativo del luogo di visita (*LOC*), la tipologia del paziente, ovvero se necessita di una sessione individuale, supervisionata, o di gruppo. (Una sessione supervisionata può essere portata avanti simultaneamente a una sessione individuale, o ad altre sessioni supervisionate) (*T*), la durata minima della sessione (*MIN*), la durata ideale per la sessione (*IL*), il periodo ideale per la sessione (*IP*), l'orario ideale per la sessione (*IH*), l'indicazione per una sessione opzionale o obbligatoria (*O*), l'indicazione della priorità con cui deve essere considerato l'orario ideale (*TP*). In questo caso rappresentano le nuove sessioni, generate dalla fase del riassegnamento.
- *start*(*ID*,*PER*,*H*), che rappresenta l'orario di inizio di una sessione nell'agenda originale, caratterizzato da l'identificativo della sessione (*ID*), il periodo in cui avviene la sessione (*PER*), l'orario di inizio della sessione (*H*).
- *length*(*ID*,*PER*,*L*), che rappresenta la durata di una sessione nell'agenda originale, caratterizzato da l'identificativo della sessione (*ID*), il periodo in cui avviene la sessione (*PER*), la durata della sessione (*L*).
- *extstart*(*ID*,*PER*,*TS*), che rappresenta l'orario di inizio esteso di una sessione nell'agenda originale, caratterizzato da l'identificativo della sessione (*ID*), il periodo in cui avviene la sessione (*PER*), l'orario d'inizio esteso (*TS*).
- *extlength*(*ID*,*PER*,*L*), che rappresenta la durata estesa di una sessione nell'agenda originale, caratterizzato da l'identificativo della sessione (*ID*), il periodo in cui avviene la sessione (*PER*), la durata estesa (*L*).
- *unavailable*(*OP*), che rappresenta l'indisponibilità di un operatore, caratterizzato dall'identificativo dell'operatore (*OP*).

- *absent(PAT)*, che rappresenta la cancellazione di una sessione da parte di un paziente, caratterizzato dall'identificativo del paziente (PAT).

5.3.3 Output

L'output è composto dagli atomi *newStart(ID,PER,H)*, che sono usati per assegnare un nuovo orario di inizio alle sessioni, cercando, però, di renderlo il più simile possibile a quello dell'agenda originale. E dagli atomi *newLength(ID,PER,L)*, usati per assegnare una nuova durata alle sessioni. Inoltre è composto dagli atomi *newextstart(ID,PER,H)* e *newextlength(ID,PER,L)*, che rappresentano rispettivamente l'orario di inizio e la durata estesi, per conoscere la durata totale di ogni sessione/

5.3.4 Operatore non disponibile

Di seguito vengono riportate le regole per la soluzione allo scenario dell'operatore non disponibile.

Regole nuove per la ripianificazione

Tramite la regola 1 vengono creati dei nuovi atomi nella forma *newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP)* per tutte le nuove sessioni per le quali esiste una sessione originale tenuta da un operatore non più disponibile, per il quale esiste un atomo nella forma *unavailable(OP)*. Nel caso una sessione non possa essere inserita nell'agenda come individuale, essa verrà trasformata, modificando forzatamente il suo tipo in modo che sia completamente supervisionata; in questo modo potrà essere svolta in contemporanea ad altre sessioni.

```
1 newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) | newSession(ID,PAT,OP,LOC,1,MIN,
    IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP), session(ID,
    PAT,OPN,_,_,_,_,_,_,_,_), unavailable(OPN), OP != OPN.
```

Tramite le seguenti regole, dalla 2 alla 9, vengono raggruppati tutti gli elementi validi al fine della ripianificazione: sessioni, orari di inizio e durate, anche estesi. Per far ciò si considerano gli atomi in input, escludendo da essi quelli che appartenevano a sessioni tenute da operatori non più disponibili; con la regola successiva viene aggiunto allo stesso gruppo l'elemento associato alla nuova sessione descritta dalla regola 1. Questi atomi saranno utili per i confronti successivi e per avere una visione complessiva dell'agenda finale.

```
2 finalSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,
    IL,IP,IH,O,TP), not unavailable(OP), not newSession(ID,PAT,_,_,_,_,_,_,_,_,
    _,_,_).
```

```

3  finalSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- newSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP) .
4  finalStart(ID,PER,TS) :- start(ID,PER,TS), session(ID,_,OP,_,_,_,_,_,_,_,
    _), not unavailable(OP), not newSession(ID,_,_,_,_,_,_,_,_,_,_) .
5  finalStart(ID,PER,TS) :- newStart(ID,PER,TS) .
6  finalLength(ID,PER,NL) :- length(ID,PER,NL), session(ID,_,OP,_,_,_,_,_,_,_,
    _), not unavailable(OP), not newSession(ID,_,_,_,_,_,_,_,_,_,_) .
7  finalLength(ID,PER,NL) :- newLength(ID,PER,NL) .
8  newextstart(ID,PER,TS) :- extstart(ID,PER,TS), session(ID,_,OP,_,_,_,_,_,_,_,
    _,_,_), not unavailable(OP), not newSession(ID,_,_,_,_,_,_,_,_,_,_)
    .
9  newextlength(ID,PER,TS) :- extlength(ID,PER,TS), session(ID,_,OP,_,_,_,_,_,_,_,
    _,_,_), not unavailable(OP), not newSession(ID,_,_,_,_,_,_,_,_,_,_)
    .

```

La regola *10* si assicura che le sessioni individuali non si sovrappongano, mentre la *11* e la *12* si assicurano che le sessioni di gruppo non siano sovrapposte a sessioni di altro tipo; la *13* impone che ogni paziente abbia una sola sessione per periodo (mattina o pomeriggio).

```

10 :- finalStart(ID1, PER, TS), finalLength(ID1, PER, L), finalSession(ID1, _, OP,
    _, 0, _, _, _, _, _), finalStart(ID2, PER, TS2), finalSession(ID2, _, OP,
    _, 0, _, _, _, _, _), ID1!=ID2, TS2>=TS, TS2<TS+L.
11 :- finalStart(ID, PER, TS), finalLength(ID, PER, L), finalSession(ID, _, OP, _,
    TY, _, _, _, _, _), finalStart(ID2, PER, TS2), finalSession(ID2, _, OP, _,
    2, _, _, _, _, _), TY != 2, TS2>=TS, TS2<TS+L.
12 :- finalStart(ID, PER, TS), finalLength(ID, PER, L), finalSession(ID, _, OP, _,
    2, _, _, _, _, _), finalStart(ID2, PER, TS2), finalSession(ID2, _, OP, _,
    TY, _, _, _, _, _), TY != 2, TS2>=TS, TS2<TS+L.
13 :- finalStart(ID1, PER1, _), finalSession(ID1, PAT, _, _, _, _, _, _, _),
    newStart(ID2, PER2, _), newSession(ID2, PAT, _, _, _, _, _, _, _), ID1
    !=ID2, PER1=PER2.

```

Regole adattate per la ripianificazione

Le seguenti regole vengono riprese dalla pianificazione, ma vengono in parte modificate ed agiscono, non più sull'atomo *session*, ma su *newSession*.

La regola *14* attribuisce un orario di inizio ad ogni nuova sessione obbligatoria, rispettando gli orari di lavoro dell'operatore. La *15* cerca, invece, di assegnare un orario al massimo numero di sessioni opzionali. La regola *16* attribuisce una durata ad ogni nuova sessione, rispettando gli orari dell'operatore e la durata massima stabilita per tale sessione.

Per le sessioni individuali questi orari rappresentano solo lo svolgimento della parte in individuale; infatti le sessioni possono essere portate avanti in parte in individuale e in parte in supervisione. Nel caso in cui nessun orario fosse disponibile per lo svolgimento in individuale, la sessione sarà svolta completamente in supervisione: in questo caso gli orari rappresentano lo svolgimento completo.

```

14 1 {newStart(ID, PER, TS): time(PER, OP, TS)} 1 :- newSession(ID, _, OP, _, _, _,
    _, _, _, 0, _).
15 0 {newStart(ID, PER, TS): time(PER, OP, TS)} 1 :- newSession(ID, _, OP, _, _, _,
    _, _, _, 1, _).
16 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL >=
    1, NL <= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END),
    newSession(ID, _, OP, _, _, _, MAX, _, _, _, _).

```

Le regole 17 e 18 riservano del tempo prima e dopo le sessioni. La 19 e la 20 definiscono dei nuovi atomi per creare un nuovo orario di inizio e una nuova durata: essi vengono usati per estendere la durata di una sessione individuale, qualora servisse più tempo al suo svolgimento; questa verrà svolta in parte come supervisionata per poter essere sovrapposta ad altre sessioni. La 21 si assicura che ad ogni paziente venga riservato il tempo minimo stabilito per le sue visite.

```

17 1 {before(ID, NL): time(PER, OP, L), NL=L-ST, NL<=TS-ST} 1 :- newStart(ID, PER, TS),
    period(PER, OP, ST, _), newSession(ID, _, OP, _, _, _, _, _, _, _).
18 1 {after(ID, NL): time(PER, OP, L), NL=L-ST, NL<=END-TS-LEN} 1 :- newStart(ID, PER,
    TS), period(PER, OP, ST, END), newLength(ID, PER, LEN), newSession(ID, _, OP, _,
    _, _, _, _, _, _).
19 newextstart(ID, PER, TS - LB) :- newStart(ID, PER, TS), before(ID, LB).
20 newextlength(ID, PER, L + LA + LB) :- newLength(ID, PER, L), after(ID, LA),
    before(ID, LB).
21 :- patient(ID, _, MIN), #sum{LEN, SES: finalSession(SES, ID, _, _, _, _, _, _, _),
    newextlength(SES, _, LEN)} < MIN.

```

Infine, le regole 14 e 15, che riprendono le regole 11 e 12 della pianificazione, assicurano che la sessione non venga inserita in determinati orari vietati per il paziente, per il quale esiste un atomo della forma *forbidden*.

Ottimizzazione

Con questo primo vincolo di ottimizzazione, con priorità maggiore, viene minimizzato il numero di nuove sessioni completamente in supervisione. In questo modo si cercherà prima di inserire nell'agenda ogni sessione individuale come individuale e solo nel caso esse non riuscissero a rispettare i vincoli imposti, verranno trasformate in supervisionate.

```

22 :- #count{ID: newSession(ID, _, _, _, 1, _, _, _, _, _)} = N. [N@4]

```

Questi vincoli di ottimizzazione permettono di rispettare il vincolo che la nuova agenda realizzata dalla ripianificazione sia il più simile possibile a quella originale. Dalla priorità più bassa a quella più alta: il 23 è usato per avere l'orario di inizio di una nuova sessione il più simile possibile all'orario originale di quella sessione, se ne viene rispettato il periodo; il 24 è usato per avere il periodo di una nuova sessione il più simile possibile al periodo della sessione originale. L'ultimo, infine, minimizza per ogni sessione la differenza tra la

durata della nuova sessione e la durata massima per la sessione, in modo che essa possa durare il più possibile in individuale.

```

23 :~ newStart(ID,PER,NTS), newSession(ID,_,_,_,_,_,_,_,_,_,_), start(ID
    ,PER,TS), DF = |NTS - TS|. [DF@1, ID]
24 :~ newStart(ID,NPER,_,_), newSession(ID,_,_,_,_,_,_,_,_,_,_), start(ID,
    PER,_), DF = |NPER - PER|. [DF@2, ID]
25 :~ newLength(ID,_,NL), newSession(ID,_,_,_,_,_,_,IL,_,_,_,_), DF = |NL
    - IL|. [DF@3, ID]

```

Inoltre sono stati aggiunti ulteriori vincoli di ottimizzazione, con priorità minore dei precedenti, per fare in modo che le nuove sessioni cerchino di ripetere gli orari ideali dichiarati dalla sessione stessa. Quindi come priorità massima si cercherà di avvicinare il più possibile gli orari a quelli originali e con priorità minore a quelli ideali. Dalla priorità più bassa a quella più alta: la 26 cerca di avvicinare l'orario di inizio di una sessione non prioritaria il più possibile all'orario ideale; la 27 cerca di avvicinare il periodo di una sessione non prioritaria il più possibile all'orario ideale; la 28 cerca di avvicinare la durata di ogni sessione alla durata ideale.

Tuttavia questi vincoli, con la priorità più bassa, influiscono in maniera minore sulla soluzione, tranne che in situazioni particolari in cui ci siano più posizioni utili all'inserimento e un orario preferito dal paziente. Per cui essi non verranno considerati all'interno della soluzione finale.

```

26 :~ newStart(ID, PER, TS), newSession(ID,_,_,_,_,_,_,_,PER,OPT,_,_), D
    =|TS - OPT|. [D@1, ID]
27 :~ newStart(ID, PER, _), newSession(ID,_,_,_,_,_,_,_,OPT,_,_,_), D =|
    PER - OPT|. [D@2, ID]
28 :~ newLength(ID,_, L), newSession(ID,_,_,_,_,_,_,OPT,_,_,_,_), D = |L
    - OPT|. [D@3, ID]

```

Tramite questi livelli di priorità, si cercherà di far rispettare principalmente il vincolo con priorità più alta, così via fino al vincolo con priorità più bassa. Si può comprendere, grazie ad essi, l'importanza attribuita ad ogni vincolo all'interno del problema.

5.3.5 Paziente non presente

Di seguito vengono riportate le regole per la soluzione allo scenario del paziente non presente.

Regole nuove per la ripianificazione

Con la prima regola vengono create delle nuove sessioni che rappresentano delle sessioni già presenti all'interno dell'agenda per le quali potrebbe esserci la possibilità di essere

spostare più vicino al loro orario preferito. Si considerano quindi tra le sessioni individuali quelle che non siano state cancellate, tramite l'utilizzo dell'atomo *absent(PAT)*, e che non non inizino già al loro orario preferito: questo viene controllato tramite il conto dei parametri orario di inizio, periodo, durata che non vengono rispettati; anche se solo uno di questi non viene rispettato, la sessione viene contata. Inoltre queste sessioni devono essere tenute da un operatore che teneva in origine una sessione cancellata; infatti solo in questo caso potrebbe esserci uno spazio libero per permettere lo spostamento di altre sessioni.

```

1 newSession(ID,PAT,OP,LOC,0,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,0,MIN,IL,
    IP,IH,O,TP), not absent(PAT), absent(PAT2), session(_, PAT2, OP, _, _, _, _,
    _, _, _, _), PAT!=PAT2, #count{ID: start(ID,_,ST), IH!=ST; ID: start(ID,PER
    ,_), IP!=PER; ID: length(ID,_,L), L!=IL} > 0.

```

Come per lo scenario precedente, vengono raggruppati tutti gli elementi validi. In questo caso si escludono le sessioni cancellate, ovvero per cui è presente l'atomo *absent(PAT)*.

```

2 finalSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,
    IL,IP,IH,O,TP), not absent(PAT).
3 finalSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- newSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP).
4 finalStart(ID,PER,TS) :- start(ID,PER,TS), session(ID,PAT,_,_,_,_,_,_,_,_,
    ,_), not absent(PAT), not newSession(ID,_,_,_,_,_,_,_,_,_,_,_).
5 finalStart(ID,PER,TS) :- newStart(ID,PER,TS).
6 finalLength(ID,PER,NL) :- length(ID,PER,NL), session(ID,PAT,_,_,_,_,_,_,_,_,
    ,_), not absent(PAT), not newSession(ID,_,_,_,_,_,_,_,_,_,_,_).
7 finalLength(ID,PER,NL) :- newLength(ID,PER,NL).
8 newextstart(ID,PER,TS) :- extstart(ID,PER,TS), session(ID,PAT,_,_,_,_,_,_,_,
    ,_,_,_), not absent(PAT), not newSession(ID,_,_,_,_,_,_,_,_,_,_,_).
9 newextlength(ID,PER,TS) :- extlength(ID,PER,TS), session(ID,PAT,_,_,_,_,_,_,
    ,_,_,_,_), not absent(PAT), not newSession(ID,_,_,_,_,_,_,_,_,_,_,_).

```

Regole adattate per la ripianificazione

In questo scenario una sessione può essere spostata solo se la sua durata può continuare a rispettare un minimo, oltre che un massimo. A differenza dello scenario precedente, la durata assegnata è vincolata anche da un minimo.

```

10 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>=
    MIN, NL<= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END),
    newSession(ID, _, OP, _, _, MIN, MAX, _, _, _, _).

```

La regola 11 impone che ogni ad paziente, che non sia assente, venga dedicato un tempo minimo per le sue visite.

```

11 :- patient(ID,_,MIN), not absent(ID), #sum{LEN, SES: finalSession(SES,ID,_,_,_,_,
    ,_,_,_,_,_), newextlength(SES,_,LEN)} < MIN.

```

Le restanti regole sono le stesse descritte per lo scenario precedente, tra le regole adattate

Ottimizzazione

La prima regola di ottimizzazione cerca di avvicinare il nuovo orario di inizio a quello preferito, se viene rispettato il periodo preferito; la seconda cerca di far rispettare alla sessione il periodo preferito. Il terzo vincolo, con priorità maggiore, cerca invece di massimizzare la durata della sessione.

```

12 :~ newStart(ID,PER,NTS), newSession(ID,_,_,_,_,_,_,_,PER,IH,_,_), DF =
    |IH - NTS|. [DF@1, ID]
13 :~ newStart(ID,NPER,_), newSession(ID,_,_,_,_,_,_,_,PER,_,_,_), DF = |
    PER - NPER|. [DF@2, ID]
14 :~ newLength(ID,_,NL), newSession(ID,_,_,_,_,_,_,IL,_,_,_,_), DF = |NL
    - IL|. [DF@3, ID]

```

5.3.6 Operatore non disponibile e paziente non presente

Di seguito vengono riportate le regole per la soluzione allo scenario dove sono presenti sia operatori non disponibili che pazienti non presenti.

Regole nuove per la ripianificazione

Con queste regole è necessario separare dalle sessioni originali le diverse sessioni che vengono prese in considerazione negli scenari precedenti. Con la regola 1 vengono considerate le sessioni nuove che sono state riassegnate a nuovi operatori; viene inoltre osservato che la sessione non sia stata cancellata. La seconda regola considera invece le sessioni originali che possono essere spostate, data la cancellazione di un'altra sessione; si osserva che queste sessioni non siano tra quelle nuove. In questo modo si creano due gruppi mutualmente esclusivi, così che possano essere trattati diversamente da certi vincoli necessari solo a un certo tipo di sessione.

```

1 unSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) | unSession(ID,PAT,OP,LOC,1,MIN,IL,
    IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP), session(ID,PAT,
    OPN,_,_,_,_,_,_,_,_,_), unavailable(OPN), not absent(PAT), OP != OPN.
2 newSession(ID,PAT,OP,LOC,0,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,0,MIN,IL,
    IP,IH,O,TP), not unavailable(OP), not absent(PAT), absent(PAT2), session(_,
    PAT2,OP,_,_,_,_,_,_,_,_,_), not unSession(ID,_,_,_,_,_,_,_,_,_,_),
    PAT!=PAT2, #count{ID: start(ID,_,ST), IH!=ST; ID: start(ID,PER,_), IP
    !=PER; ID: length(ID,_,L), L!=IL} > 0.

```

I due gruppi di sessioni si uniscono poi nel singolo gruppo definito dall'atomo *newSession*, per poter usare quest'ultimo all'interno dei vincoli comuni a tutte le sessioni. (Per esempio l'assegnazione di un nuovo orario di inizio)

```

3 newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- unSession(ID,PAT,OP,LOC,T,MIN,
    IL,IP,IH,O,TP) .
4 newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- absSession(ID,PAT,OP,LOC,T,MIN,
    IL,IP,IH,O,TP) .

```

Regole adattate per la ripianificazione

Si assegnano a queste sessioni, come per gli scenari precedenti, una durata: per quelle nuove la durata non è vincolata da un valore minimo (oltre a > 0) e può essere anche solo di uno slot temporale, mentre per quelle già presenti deve essere rispetta una durata minima.

```

5 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>=
    1, NL<= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END), unSession
    (ID, _, OP, _, _, _, MAX, _, _, _).
6 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>=
    MIN, NL<= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END),
    absSession(ID, _, OP, _, _, MIN, MAX, _, _, _).

```

Le restanti regole sono le stesse descritte per lo scenario precedente, tra le regole adattate.

Ottimizzazione

Il vincolo di ottimizzazione con priorità più alta minimizza il numero di sessioni trasformate in *completamente in supervisione* tra le nuove sessioni. Il vincolo successivo cerca di massimizzare la durata di tutte le sessioni prese in considerazione.

```

7 :~ #count{ID: unSession(ID, _, _, _, 1, _, _, _, _, _)} = N. [N@6]
8 :~ newLength(ID,_,NL), newSession(ID, _, _, _, _, _, IL, _, _, _), DF = |NL
    - IL|. [DF@5, ID]

```

Con priorità minore, i seguenti vincoli cercano di far rispettare per le sessioni già presenti nell'agenda il loro periodo preferito; se questo viene rispettato si cerca di avvicinare il più possibile l'orario di inizio a quello preferito.

```

9 :~ newStart(ID,NPER,_), absSession(ID, _, _, _, _, _, PER, _, _), DF = |
    PER - NPER|. [DF@4, ID]
10 :~ newStart(ID,PER,NTS), absSession(ID, _, _, _, _, _, PER, IH, _), DF =
    |IH - NTS|. [DF@3, ID]

```

Infine con la priorità più bassa si prova a far rispettare alle nuove sessioni il loro periodo originale; se questo viene rispettato si cerca di avvicinare il più possibile l'orario di inizio a quello originale.

```

11 :~ newStart(ID,NPER,_), unSession(ID,_,_,_,_,_,_,_,_,_,_), start(ID,
    PER,_), DF = |NPER - PER|. [DF@2, ID]
12 :~ newStart(ID,PER,NTS), unSession(ID,_,_,_,_,_,_,_,_,_,_), start(ID,
    PER,TS), DF = |NTS - TS|. [DF@1, ID]

```

L'ordine di priorità tra i vincoli per il rispetto di orario di inizio e periodo delle sessioni nuove e già presenti può essere invertito, rispetto alle preferenze che si hanno nel privilegiare un tipo di sessione rispetto all'altro.

5.3.7 Indisponibilità e assenza con orari parziali

Di seguito vengono riportate le regole per la soluzione allo scenario in cui gli le indisponibilità e le assenze possono non essere considerate assolute, ma ristrette a certe fasce orarie indicate in input.

5.3.8 Input modificato

Per poter trattare gli orari come parziali devono essere modificati alcuni degli atomi in input. Per indicare l'indisponibilità degli operatori l'atomo viene espresso come:

- *unavailable(OP,PER,ST,END)*

dove OP indica l'identificativo dell'operatore, PER il periodo in cui è indisponibile, ST l'orario di inizio e END l'orario di fine dell'indisponibilità.

Per indicare, invece, l'assenza dei pazienti in certe fasce d'orario viene usato l'atomo:

- *absent(PAT,PER,ST,END)*

dove PAT indica l'identificativo del paziente, PER il periodo in cui è assente, ST l'orario di inizio e END l'orario di fine dell'assenza. Nel caso l'assenza fosse totale, viene indicata con l'inserimento di uno zero nei campi PER, ST, END.

Regole nuove per la ripianificazione

Con queste due prime regole vengono suddivise le sessioni inerenti all'indisponibilità di operatori. Con la prima vengono raggruppate le nuove sessioni di pazienti riassegnati nella prima fase, che inoltre non hanno associata alcun tipo di assenza. Con la seconda sono considerate le sessioni tenute da un operatore con un'indisponibilità parziale (se quest'ultima fosse totale tutte le sessioni prima assegnate a questo operatore sarebbero presenti come atomi *unSession*. Come nelle codifiche precedenti, queste sessioni possono essere trasformate in "solo in supervisione".

-
- 1 `unSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) | unSession(ID,PAT,OP,LOC,1,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP), session(ID,PAT,OPN,_,_,_,_,_,_,_,_), unavailable(OPN,_,_,_), not absent(PAT,_,_,_) OP != OPN.`
 - 2 `origSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) | origSession(ID,PAT,OP,LOC,1,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP), not unSession(ID,_,_,_,_,_,_,_,_,_,_), unavailable(OP,_,_,_), not absent(PAT,_,_,_).`
-

Le seguenti due regole suddividono le sessioni relative all'assenza di pazienti. La prima raggruppa le sessioni già presenti nell'agenda in cui si trova una sessione a cui è legato un atomo del tipo *absent*, che non si trovano ancora nel loro orario preferito. L'altra regola raggruppa a parte le sessioni per le quali il paziente ha espresso un'assenza parziale.

-
- 3 `otherSession(ID,PAT,OP,LOC,0,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,0,MIN,IL,IP,IH,O,TP), not unavailable(OP,_,_,_), not allUnSession(ID,_,_,_,_,_,_,_,_,_,_), not absent(PAT,_,_,_), absent(PAT2,_,_,_), session(_,PAT2,OP,_,_,_,_,_,_,_,_,_), PAT!=PAT2, #count{ID: start(ID,_,ST), IH !=ST; ID: start(ID,PER,_) , IP!=PER; ID: length(ID,_,L), L!=IL} > 0`
 - 4 `absSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- session(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP), absent(PAT,PER,_,_), PER != 0.`
-

Le seguenti regole fanno in modo che le sessioni tenute da un operatore con un'indisponibilità parziale non vengano inserite negli orari in cui l'operatore non sia presente.

-
- 5 `:- unavailable(OP,PER,ST,END), newSession(ID,_,OP,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), ST<=TS, END>TS.`
 - 6 `:- unavailable(OP,PER,_,END), newSession(ID,_,OP,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), END>TS, END<=TS+L.`
 - 7 `:- unavailable(OP,PER,ST,_), newSession(ID,_,OP,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), ST>=TS, ST<TS+L.`
-

Le seguenti regole fanno in modo che le sessioni per cui un paziente abbia espresso un'assenza parziale (definite dalla regola 4) non vengano inserite negli orari relativi all'assenza.

-
- 8 `:- absent(PAT,PER,ST,END), absSession(ID,PAT,_,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), ST<=TS, END>TS.`
 - 9 `:- absent(PAT,PER,_,END), absSession(ID,PAT,_,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), END>TS, END<=TS+L.`
 - 10 `:- absent(PAT,PER,ST,_), absSession(ID,PAT,_,_,_,_,_,_,_,_,_), newextstart(ID,PER,TS), newextlength(ID,PER,L), ST>=TS, ST<TS+L.`
-

Queste regole raggruppano gli atomi definiti in precedenza, in modo da poter applicare certi vincoli comuni al gruppo di sessioni. Le regole 11 e 12 raggruppano le regole definite in 1 e 2; le regole 13 e 14 raggruppano la 3 e la 4; infine la 15 e la 16 raggruppano i due insiemi appena definiti nell'atomo *newSession*, a cui vengono applicati i vincoli più generali.

```

11 allUnSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- unSession(ID,PAT,OP,LOC,T,MIN
    ,IL,IP,IH,O,TP) .
12 allUnSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- origSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP) .
13 allAbsSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- otherSession(ID,PAT,OP,LOC,T
    ,MIN,IL,IP,IH,O,TP) .
14 allAbsSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- absSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP) .
15 newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- allUnSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP) .
16 newSession(ID,PAT,OP,LOC,T,MIN,IL,IP,IH,O,TP) :- allAbsSession(ID,PAT,OP,LOC,T,
    MIN,IL,IP,IH,O,TP) .

```

Regole adattate per la ripianificazione

Le regole 17 e 18 assegnano una durata alle sessioni definite dai gruppi espressi in precedenza, secondo i vincoli definiti anche per gli altri scenari.

```

17 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>=
    1, NL<= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END),
    allUnSession(ID, _, OP, _, _, MAX, _, _, _).
18 1 {newLength(ID, PER, NL): time(PER, OP, L), NL = L - ST, TS + NL <= END, NL>=
    MIN, NL<= MAX} 1 :- newStart(ID, PER, TS), period(PER, OP, ST, END),
    allAbsSession(ID, _, OP, _, MIN, MAX, _, _, _).

```

Le regole rimanenti sono le stesse espresse per gli scenari precedenti, dove l'atomo *newSession* qui è definito secondo le regole 15 e 16.

Ottimizzazione

Le regole di ottimizzazione sono le stesse espresse nello scenario precedente, ma vengono applicate ai raggruppamenti effettuati dalle regole prima definite.

```

19 :~ #count{ID: allUnSession(ID, _, _, 1, _, _, _, _)} = N. [N@6]
20 :~ newStart(ID,PER,NTS), allUnSession(ID, _, _, _, _, _, _, _), start(
    ID,PER,TS), DF = |NTS - TS|. [DF@1, ID]
21 :~ newStart(ID,NPER,_), allUnSession(ID, _, _, _, _, _, _, _), start(
    ID,PER,_), DF = |NPER - PER|. [DF@2, ID]
22 :~ newLength(ID,_,NL), newSession(ID, _, _, _, _, IL, _, _, _), DF = |NL
    - IL|. [DF@5, ID]
23 :~ newStart(ID,PER,NTS), allAbsSession(ID, _, _, _, _, PER, IH, _, _), DF
    = |IH - NTS|. [DF@3, ID]
24 :~ newStart(ID,NPER,_), allAbsSession(ID, _, _, _, PER, _, _, _), DF =
    |PER - NPER|. [DF@4, ID]

```

Capitolo 6

Analisi sperimentali

In questo capitolo vengono riportate i risultati dell'analisi della soluzione ottenuta, prendendo in considerazione gli scenari che possono occorrere, dalle situazioni comuni e realistiche ad alcuni casi più particolari, che sono stati presi comunque in considerazione. Poi, per valutare la scalabilità dell'approccio, la soluzione è stata testata a fronte di un numero sempre crescente di indisponibilità.

I dati usati sono reali e sono stati forniti dagli Istituti Clinici Scientifici Maugeri, in particolare dagli istituti di Genova Nervi e Castel Goffredo. La Tabella 6.1 fornisce una sintesi delle dimensioni delle istanze dei due istituti in termini di numero di operatori sanitari, numero di pazienti e densità di pazienti per operatore.

Istituto	Operatori	Pazienti	Densità
Genova Nervi	[9,18]	[37,67]	[2.4,5.2]
Castel Goffredo	[11,17]	[51,78]	[3.5,6.4]

Tabella 6.1: Dimensioni degli istituti di ICS Maugeri

I test sono stati eseguiti su un computer con processore Intel Core i7-1065G7 CPU @ 3.2GHz con 16 GB di RAM, 4 core e 8 thread. Il sistema ASP usato è stato *CLINGO*, versione 5.4.0.

6.1 Riassegnamento dei pazienti

Consideriamo qui lo scenario in cui non siano più disponibili uno o più operatori: in questo caso i pazienti originariamente assegnati ad essi devono essere riassegnati. Sono stati eseguiti vari test, prendendo in considerazione un numero sempre maggiore di operatori non

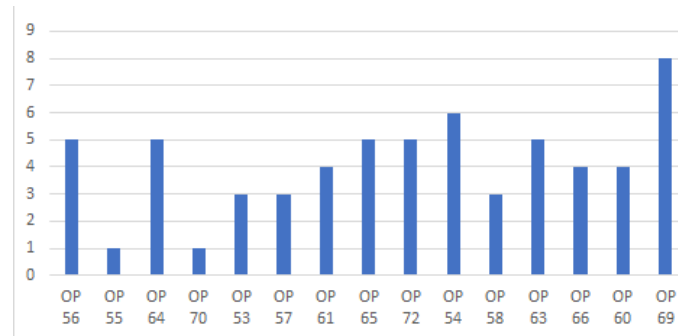


Figura 6.1: Il numero di pazienti assegnato ad ogni operatore

disponibili e relativamente un certo numero maggiore di pazienti da riassegnare. I dati a disposizione sono composti da:

- 15 operatori.
- 62 pazienti.

Dati provenienti dall'istituto di Castel Goffredo.

L'obiettivo è riassegnare i pazienti agli operatori con più tempo libero a disposizione. Si consideri innanzitutto il tempo massimo a disposizione di ogni operatore all'interno del proprio turno. Si confronti, quindi, questo con il tempo minimo utile a svolgere le visite dei pazienti assegnati in origine. È possibile osservare questo confronto in Figura 6.2: nel grafico di sinistra per ogni operatore sono messi a confronto tempi massimi (ET) e tempi occupati dalle visite dei pazienti già assegnati ad essi (OT). A destra, invece, per maggior chiarezza è indicato la quantità di tempo libero per operatore.

Sono stati effettuati le analisi su esempi di indisponibilità di 1, 2, 5 e 8 operatori e sul

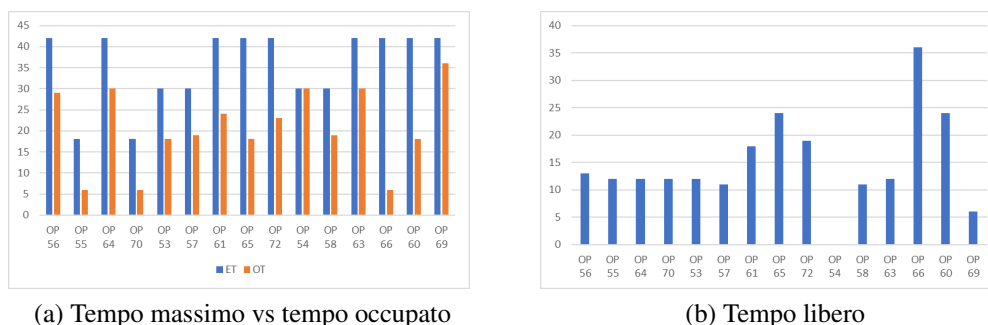


Figura 6.2: Tempo massimo e tempo occupato a confronto per ogni operatore

conseguente riassegnamento di un numero sempre maggiore di pazienti: 4, 10, 24, 31. Si possono vedere i risultati del riassegnamento in Figura 6.3. Bisogna ricordare, inoltre, che in questa distribuzione vengono rispettati i bisogni dei pazienti, relativamente alle specializzazioni degli operatori. Infatti, osservando il grafico precedente, si nota che l'operatore con maggior tempo libero, e perciò più indicato per il riassegnamento, è l'operatore 66. Eppure nel grafico (a) nessuno dei 4 pazienti viene riassegnato a questo operatore: il motivo è che tale operatore tratta solamente i pazienti di categoria MAC, alla quale nessuno di questi pazienti appartiene. Si nota comunque il tentativo di assegnamento ad operatori meno occupati, anche all'aumentare di operatori non disponibili. Nel caso di molti pazienti da riassegnare, gli operatori verranno riempiti fino al loro massimo e alcuni pazienti dovranno essere lasciati da parte per un ulteriore assegnamento, data la mancanza di operatori adatti al loro trattamento, come nel caso del grafico (d).

Nel grafico di Figura 6.4 è presente l'andamento del tempo di esecuzione all'aumentare dei pazienti da riassegnare. Vediamo come, anche con un numero di pazienti sempre maggiore, la soluzione sia computata sempre in meno di 0.2 secondi.

Si prenda ora in considerazione la Tabella 6.2, che mette a confronto, per ogni tipo-

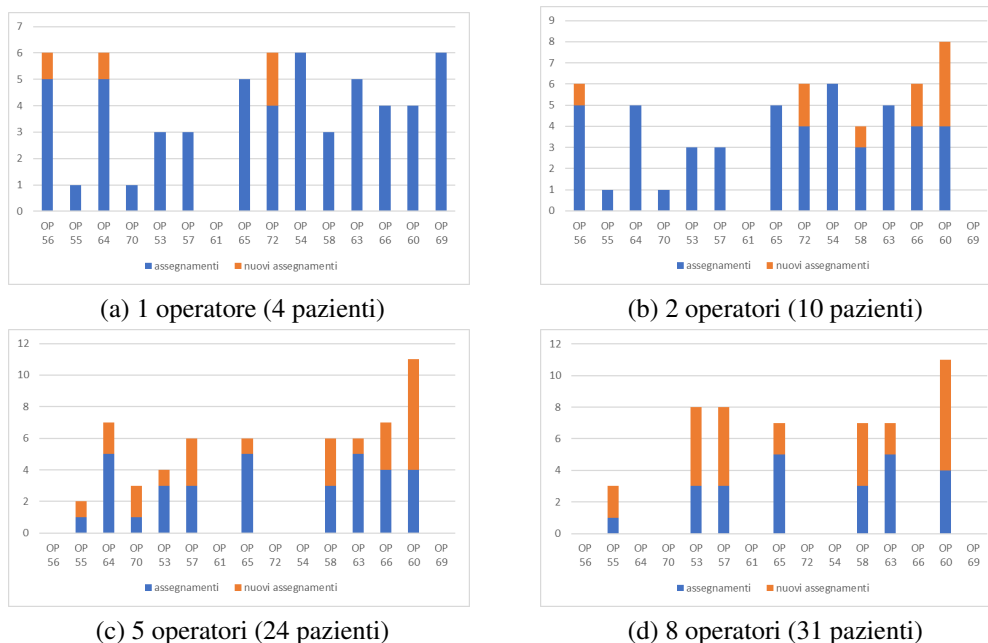


Figura 6.3: Riassegnamento dei pazienti con indisponibilità di 1, 2, 5, 8 operatori

gia di paziente, il numero di operatori che hanno la qualifica per visitarli e il numero di pazienti di quella categoria. Poichè la compatibilità tra qualifica e tipologia deve essere rispettata durante gli assegnamenti, si può comprendere meglio la scelta di date azioni. Ad

esempio, l'operatore 71 può visitare pazienti, sia neurologici che ortopedici, e può supportare una più vasta categoria di pazienti. Se consideriamo il riassegnamento (b) di Figura 6.3 si osserva come molti pazienti siano assegnati all'operatore 60, in seguito all'indisponibilità dell'operatore 69: questo è dato dal fatto che entrambi possiedono solo la qualifica Ambulatoriale e dalla scarsità di operatori con questa qualifica. Tramite questi dati si comprende anche la necessità dell'operatore fittizio introdotto nella soluzione: se non fosse più presente l'operatore che tratta il Covid-19, i suoi pazienti non potrebbero essere riassegnati agli operatori presenti; vengono così assegnati all'operatore fittizio, per essere assegnati in seguito.

#	Neurologico	Ortopedico	Covid-19 ++	Ambulatoriale
Operatori	11	7	1	5
Pazienti	21	17	9	24

Tabella 6.2: Numero di operatori per qualifica e numero di pazienti per tipologia a confronto

6.1.1 Altri approcci

Approccio di distribuzione uniforme

Oltre alla soluzione appena descritta è stato provato un ulteriore approccio al problema: in questo caso i pazienti vengono distribuiti in maniera uniforme agli operatori in maniera casuale, sempre tenendo conto delle specializzazioni. Ogni operatore dovrà quindi avere il minor numero di pazienti assegnato. In Figura 6.5 si nota come i 4 pazienti siano stati riassegnati a 4 operatori differenti, a differenza della soluzione mostrata nel grafico (a) di Figura 6.3.

Nello specifico del nostro problema è stato deciso di tenere conto anche degli orari di lavoro degli operatori, come mostrato in precedenza, ma si è voluto analizzare anche questo tipo di approccio.

6.1.2 Riassegnamento parziale

Con l'introduzione degli orari parziali, viene modificata, come già visto, la codifica riassegnamento e per questo anche la soluzione ottenuta potrebbe essere diversa. Se un operatore non è presente per un certo numero di ore minore delle sue ore massime lavorative, il riassegnamento dei suoi pazienti potrebbe non occorrere, o non essere totale: le ore lavorative rimaste potrebbero essere ancora sufficienti.

In Figura 6.6 è mostrato un esempio in cui l'operatore 61 ha un'indisponibilità di 30 ore su

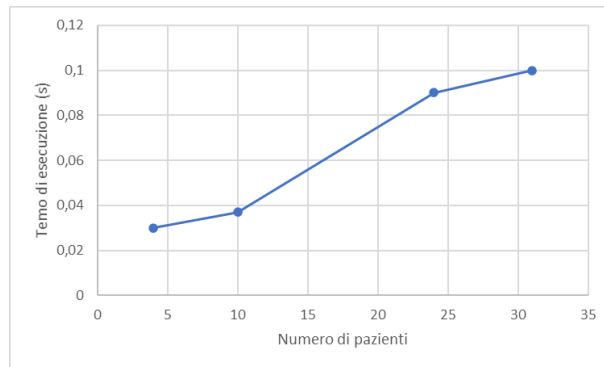


Figura 6.4: Tempo di esecuzione (s) all'aumentare dei pazienti da riassegnare

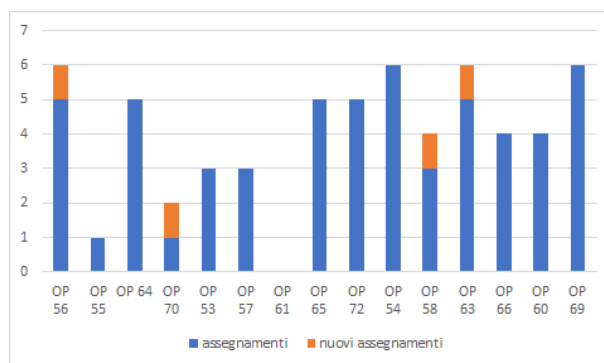


Figura 6.5: Riassegnamento uniforme con indisponibilità di 1 operatore

un totale di 42. In questo caso avviene un riassegnamento parziale: infatti solo tre sessioni vengono assegnate ad altri operatori, mentre l'operatore 61 ha ancora un numero di ore sufficiente a svolgere una sessione.

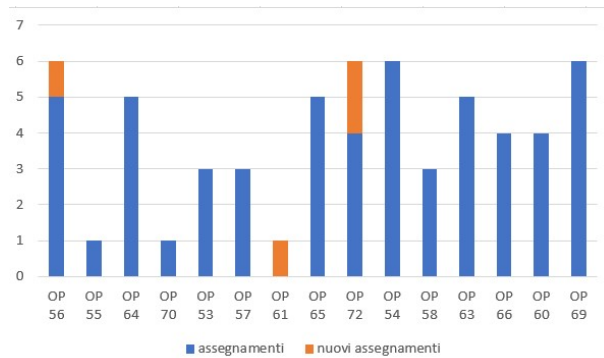


Figura 6.6: Riassegnamento dei pazienti con indisponibilità parziale di 1 operatore

6.2 Ripianificazione dell'agenda

L'agenda delle sessioni di riabilitazione dovrà essere modificata a seconda dello scenario in cui ci si trova. Analizzeremo qui di seguito tutti gli scenari considerati.

I dati a disposizione sono composti da:

- 56 pazienti.
- 68 sessioni di riabilitazione.
- 15 operatori.

Dati provenienti dall'istituto di Genova Nervi.

6.2.1 Scenario 1: operatore non disponibile

In questo scenario, data l'indisponibilità di uno o più operatori, certi pazienti sono stati riassegnati a nuovi operatori, con una conseguente creazione di nuove sessioni che devono essere inserite all'interno dell'agenda. L'inserimento della nuova sessione tiene conto di ogni slot temporale disponibile, senza imporre nessun limite minimo di tempo, favorendo così l'inserimento in individuale; il resto della sessione potrà essere terminato in supervisione. Altrimenti, in mancanza di spazio nell'agenda, la sessione può essere svolta completamente in supervisione. A seguire gli orari delle sessioni devono rimanere il più possibile uguali a quelli originali

Possiamo analizzare un primo esempio, mostrato in Figura 6.7, dove avviene la ripianificazione di due sessioni, entrambe assegnate all'operatore 44. La sessione 8 viene inserita negli unici due slot temporali disponibili, mentre il resto è eseguito in supervisione. La sessione 9, eseguita nel pomeriggio, viene invece inserita come individuale in un singolo slot, poichè quest ultimo occupato soltanto da un'altra sessione completamente in supervisione.

Si osservi in Figura 6.8 un caso più particolare, con l’inserimento di tre nuove sessioni. In questo caso i due slot temporali liberi consecutivi vengono suddivisi tra due sessioni, così che ognuna di esse abbia uno spazio in individuale. La terza sessione, terminati gli spazi liberi, viene inserita completamente in supervisione.

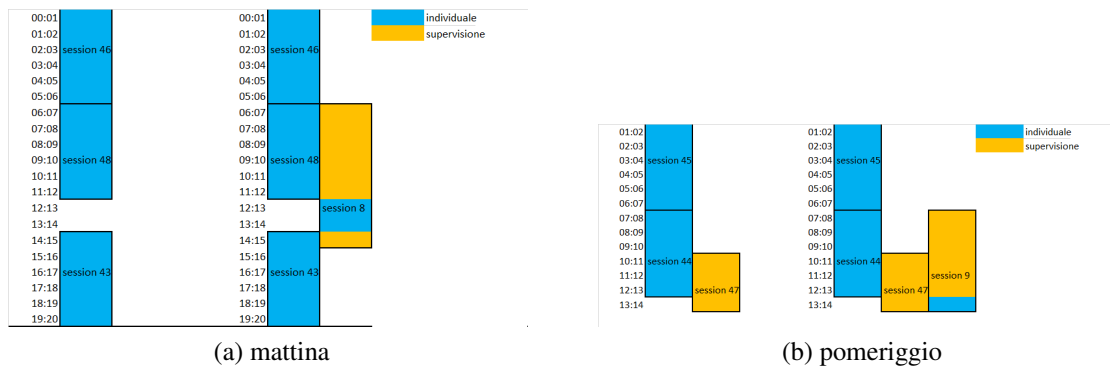


Figura 6.7: Ripianificazione dell'agenda con 2 nuove sessioni (operatore 44)

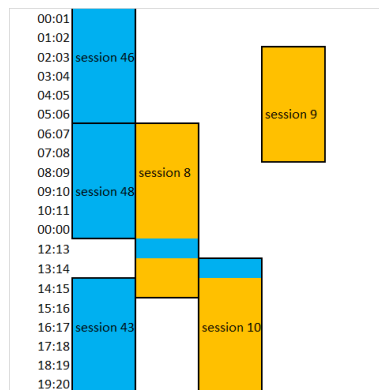


Figura 6.8: Ripianificazione dell'agenda con 3 nuove sessioni (operatore 44, mattina)

Caso particolare

Osserviamo ora questo caso particolare, per capire se sia utile considerare degli orari preferiti da rispettare oltre a quelli originali. Come si vede in Figura 6.9, in questa agenda è presente una sola sessione e ne deve essere inserita una nuova, il cui orario originale, slot 6, è occupato; rispettando primariamente la lunghezza, i due orari possibili sarebbero lo slot 0 e 12, trovandosi alla stessa distanza dallo slot 6. Senza nessuna preferenza espressa il sistema sceglie da solo e in questo caso la scelta ricade sullo slot 12. Ovviamente

se volessimo che una preferenza di orario per questa nuova sessione venisse rispettata, in questo caso non succederebbe: se la preferenza fosse per uno slot più vicino allo slot 0, questa non verrebbe rispettata. Con l'aggiunta di vincoli di ottimizzazione questo obiettivo viene raggiunto, ma con un aumento dei tempi di esecuzione. In seguito ad un'analisi su un numero diverso di dati e in presenza di questa situazione si è osservato un aumento del 15% dei tempi tramite l'utilizzo di questi ulteriori vincoli. Anche se l'aumento dei tempi non è considerevole, si è deciso di proseguire senza prendere in considerazione eventuali orari ideali.

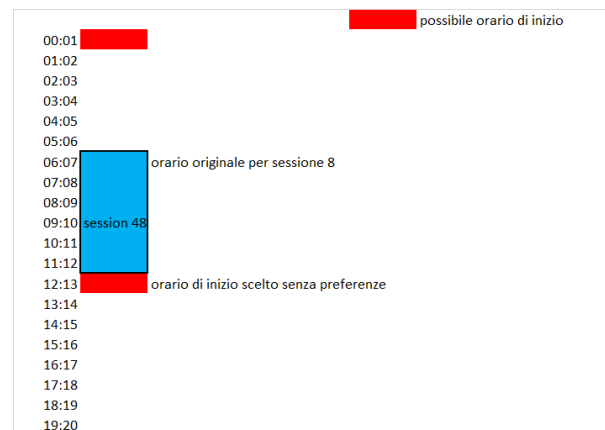


Figura 6.9: Ripianificazione dell'agenda: scelta dell'orario

6.2.2 Scenario 2: paziente non presente

In questo scenario, data l'assenza di uno o più pazienti e la successiva cancellazione delle sessioni relative, l'agenda deve essere aggiornata. Inoltre certe sessioni hanno la possibilità di essere spostate più vicine al loro orario preferito, se già non lo rispettavano, grazie alla possibilità di spazio temporale liberato dalla cancellazione di una sessione.

Se tutte le sessioni rispettano già il loro orario preferito, la nuova agenda risulterà uguale all'agenda originale, tranne che per la mancanza delle sessioni cancellate. Altrimenti, se esiste la possibilità, le sessioni tenute dallo stesso operatore che teneva la sessione cancellata verranno spostate in modo da avvicinarsi il più possibile all'orario preferito. Come si vede nell'esempio di Figura 6.10 nell'agenda divisa negli orari di mattina e pomeriggio viene cancellata la sessione 53. In seguito alla ripianificazione vengono spostate 3 sessioni, i cui orari preferiti possono essere visti tra parentesi a fianco delle relative sessioni nell'immagine. La sessione 49 viene anticipata, in modo che la 55 possa essere spostata al mattino, vicino ai suoi orari preferiti. La 52 viene lasciata al pomeriggio, con la possibilità di raggiungere la propria durata massima; da notare che ora tutte le sessioni di questo operatore vengono svolte nella loro durata massima in individuale.

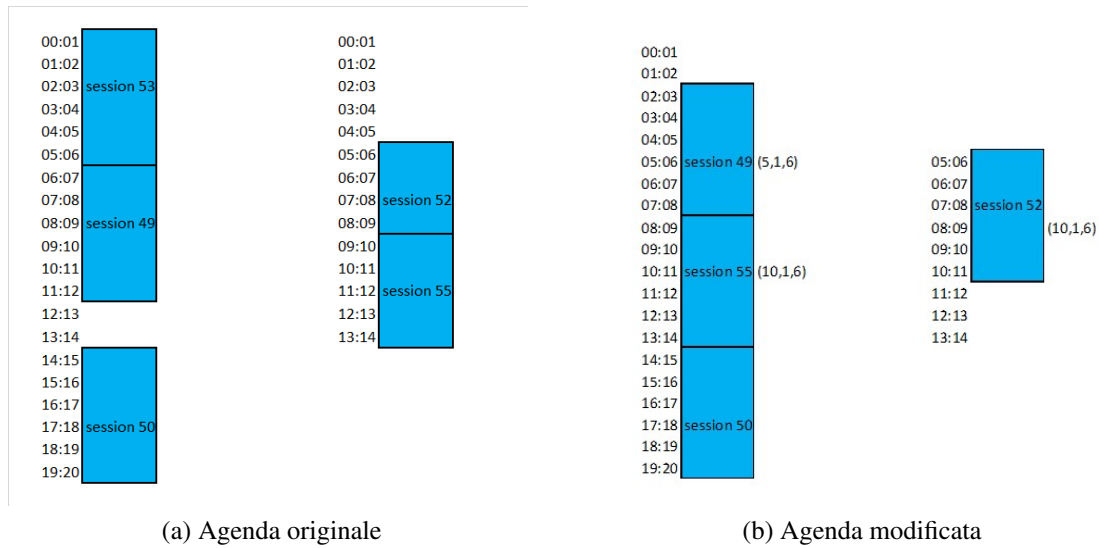


Figura 6.10: Ripianificazione dell'agenda con 1 paziente assente

Le prove sono state eseguite su dati con un numero sempre crescente di pazienti assenti. Su ogni prova un certo numero di sessioni viene preso in considerazione perchè esse vengano spostate e alcune di esse vengono effettivamente modificate temporalmente. Come si osserva anche dalla Tabella 6.3, all'aumentare del numero dei pazienti assenti, aumenta anche il numero di sessioni considerate e relativamente delle sessioni spostate; infatti più sessioni vengono eliminate dall'agenda, più spazio libero è a disposizione di altre sessioni. Tuttavia, al di sopra di un certo numero di pazienti, questo numero viene a calare: ciò perchè un numero molto elevato di sessioni è stato oramai cancellato e le sessioni che hanno la possibilità di essere modificate sono diminuite. In Figura 6.11 sono mostrati i tempi di esecuzione che rispecchiano questi risultati: al di sopra di un certo numero di pazienti assenti le sessioni che vengono spostate scendono di numero e di conseguenza diminuiscono i tempi di computazione. Si osservi come i tempi siano sempre minori di 0.6 secondi.

In Tabella 6.4 si può invece vedere l'effettivo risultato della ripianificazione; viene mostra-

Numero di pazienti	Sessioni considerate	Sessioni modificate
1	4	3
10	15	10
20	23	16
30	15	11

Tabella 6.3: Numero di sessioni spostate rispetto al numero di pazienti assenti

to il numero di sessioni che rispetta gli orari preferiti, divisi in orario di inizio, periodo e

durata, prima e dopo la ripianificazione, tenendo conto delle sessioni cancellate. Anche per questa analisi è stato tenuto conto del numero crescente di pazienti assenti: vediamo come nella maggior parte dei casi avvenga un miglioramento, ovvero un maggior numero di sessioni rispetto i propri orari preferiti, in seguito allo spostamento. Si noti come dall'assenza di un numero più elevato di pazienti, come dal caso di 20, si abbia un netto miglioramento nel rispetto degli orari di inizio, data la maggiore possibilità di spostamento delle sessioni all'interno dell'agenda. Infine, a conferma della soluzione, le sessioni rispettano in maggior numero la durata, a scendere il periodo e per ultimo l'orario di inizio, seguendo la priorità descritta nei vincoli del problema.

Numero di pazienti	Orario di inizio	Periodo	Durata	Sessioni totali
1	32 - 32	54 - 55	54 - 56	66
10	25 - 26	44 - 45	46 - 50	55
20	17 - 18	35 - 38	37 - 42	44
30	13 - 16	25 - 27	27 - 29	33

Tabella 6.4: Numero di sessioni che rispettano gli orari preferiti prima e dopo la ripianificazione rispetto al numero di pazienti assenti

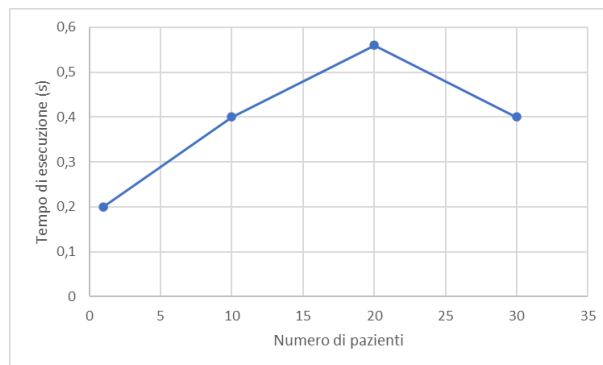


Figura 6.11: Tempo di esecuzione (s) all'aumentare dei pazienti assenti

6.2.3 Scenario 3: operatore non disponibile e paziente non presente

In questo scenario vengono considerati gli scenari precedenti allo stesso tempo. Come già esposto nel capitolo precedente, viene assegnata una priorità diversa all'inserimento di un tipo di sessione rispetto all'altro. L'ordine di priorità può essere scambiato, rispetto alle necessità della struttura. La priorità agisce solamente nel caso in cui i due tipi di sessione da inserire si trovino nella stessa agenda.

Osserviamo quindi l'esempio in Figura 6.12 in cui si verifica l'assenza di un paziente (sessione 49) e l'inserimento di tre nuove sessioni riassegnate a questa agenda. In questo caso la priorità più alta è stata assegnata alle sessioni già presenti nell'agenda, che possono essere spostate per rispettare i propri orari preferiti. Nella nuova agenda vediamo come le sessioni 52 e 55, prima inserite al pomeriggio vengano spostate al mattino, loro periodo preferito, mentre le nuove sessioni vengano inserite o al pomeriggio, o nel restante spazio al mattino. È possibile vedere tra parentesi accanto alle sessioni, gli orari preferiti per le sessioni già presente e gli orari originali per le nuove sessioni. Si ricorda che gli orari mostrati in figura indicano solo quelli in individuale, ma queste sessioni vengono svolte in parte anche in supervisione.

Si osservi invece in Figura 6.13 l'esempio con priorità alle nuove sessioni. In questo

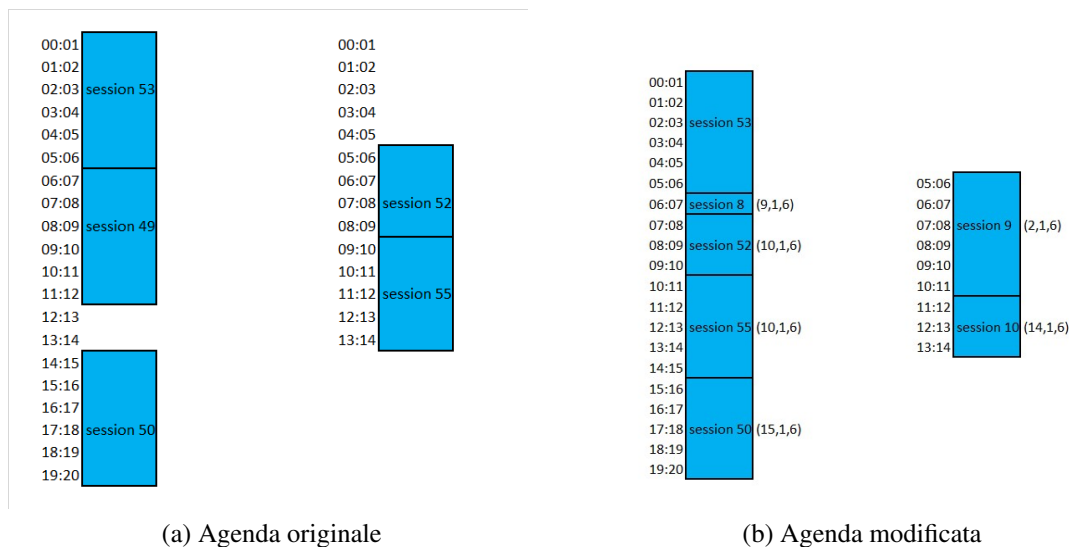


Figura 6.12: Ripianificazione dell'agenda con 1 paziente assente e 3 nuove sessioni con priorità alle sessioni originali (solo orari individuali)

caso tutte e tre le nuove sessioni vengono inserite al mattino, rispettando il proprio periodo originale, mentre il resto dell'agenda rimane invariato rispetto all'originale; solo la durata della sessione 50 viene accorciata di uno slot per permettere l'inserimento in individuale a tutte le nuove sessioni.

È interessante notare i tempi per ottenere una soluzione attraverso la codifica qui considerata (Tabella 6.5). Sono state considerate le varie tipologie che possono occorrere e per ognuna sono stati osservati i tempi medi per raggiungere la soluzione ottima e il numero di sessioni che sono state considerate dalla codifica. Si osservi come nei casi base, che si

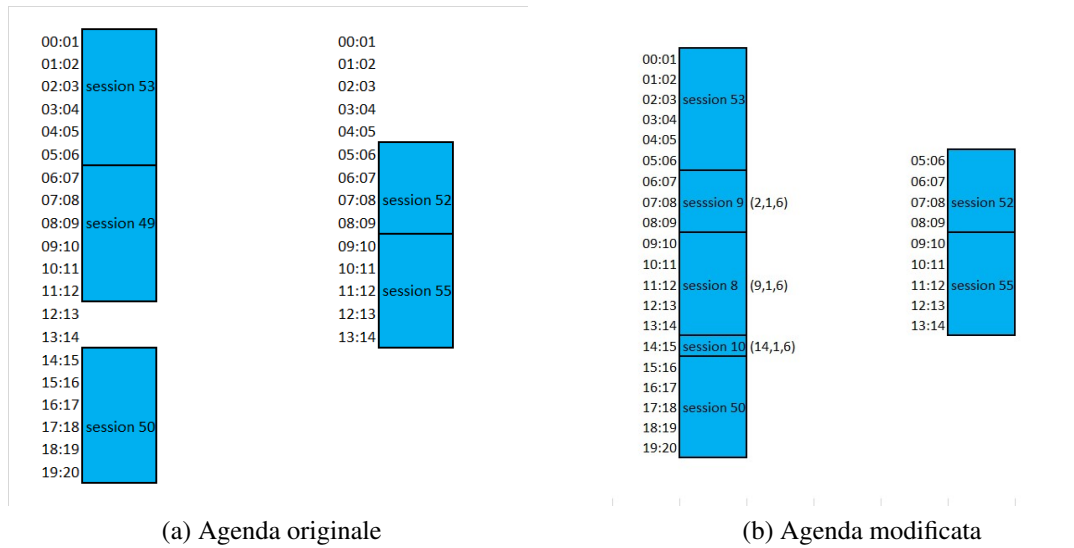


Figura 6.13: Ripianificazione dell'agenda con 1 paziente assente e 3 nuove sessioni con priorità alle nuove sessioni (solo orari individuali)

riducono a quelli analizzati in precedenza, i tempi di esecuzione rimangano sempre bassi, conformi a quelli ottenuti nei rispettivi scenari. Nel caso in cui la codifica consideri allo stesso tempo nuove sessioni e assenze in agende separate, ovvero le nuove sessioni non vengono inserite in un'agenda dove si sia verificata un'assenza, l'aumento dei tempi non è notevole. Nel caso, invece, in cui le nuove sessioni vengano inserite nella stessa agenda in cui si sia verificata un'assenza, in tempi aumentano in modo ragguardevole. Ciò è dovuto alla presenza dei vincoli di ottimizzazione, che devono agire sui due differenti tipi di sessione all'interno della stessa agenda.

Tipologia scenario	Tempo medio (s)	Sessioni considerate
Solo indisponibilità	0.4	6
Solo assenza	0.2	8
Nuove sessioni e assenze in agende separate	0.5	11
Nuove sessioni e assenze nella stessa agenda	15	6

Tabella 6.5: Tempi medi di esecuzione (s) rispetto alla tipologia di scenario all'interno della codifica con scenari combinati

6.2.4 Scenario 4: indisponibilità e assenza parziali

In questo scenario le indisponibilità e le assenze non sono più totali, ovvero che coprono l'intera giornata, ma sono definite da degli orari, che potrebbero coprire solo una parte della giornata lavorativa.

In Figura 6.14 è mostrato un esempio: l'operatore 34 ha espresso un'indisponibilità parziale, per cui tutte le sessioni, tranne la sessione 9, sono state riassegnate (in (a) è mostrata solo la posizione della sessione rimasta in agenda). Inoltre il paziente della sessione 9 ha espresso un'assenza, per cui non potrà essere presente per le visite in quei determinati slot temporali. È possibile vedere in (b) l'indisponibilità colorata in grigio e l'assenza in rosso. All'interno della parte in grigio non potrà essere svolta nessuna sessione di quella agenda, mentre nella parte rossa non potrà essere svolta solamente la sessione che ha espresso tale orari. In conseguenza a questo, la sessione è stata spostata subito dopo agli orari ad essa proibiti, cercando di rispettare i propri orari preferiti.

In un altro esempio, in Figura 6.15, viene analizzata la soluzione a fronte della sola assenza

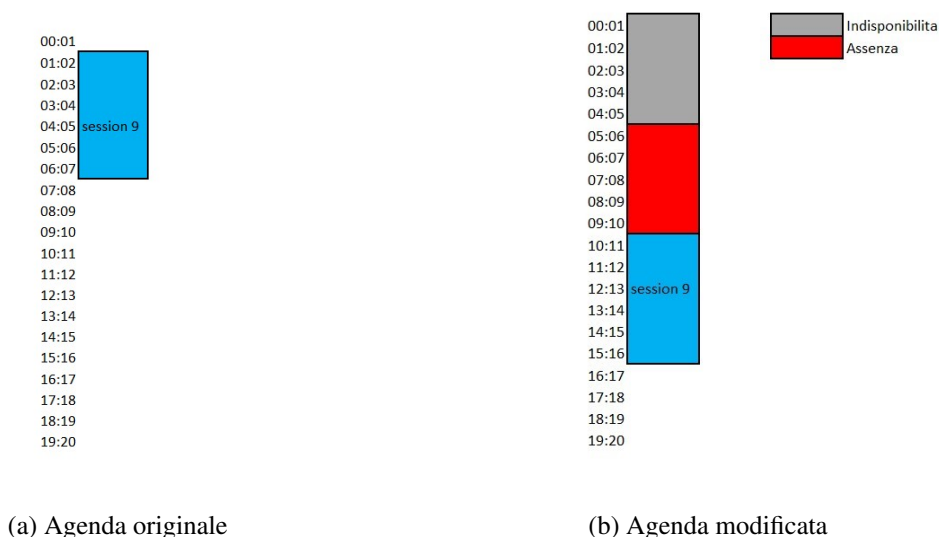
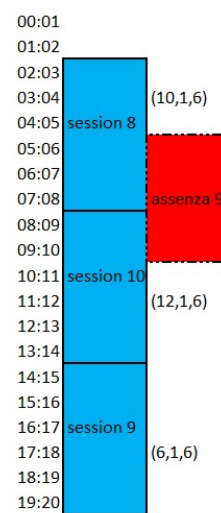


Figura 6.14: Ripianificazione dell'agenda con l'indisponibilità parziale dell'operatore e un'assenza parziale

parziale: l'assenza è la stessa espressa nell'esempio precedente; vediamo come la sessione 9 debba essere spostata in fondo all'agenda e di conseguenza debbano essere spostate anche le altre sessioni, perchè ognuna di loro venga svolta nella sua durata massima. Ognuna cerca inoltre di rispettare il proprio orario preferito.



(a) Agenda originale



(b) Agenda modificata

Figura 6.15: Ripianificazione dell'agenda con un'assenza parziale

Capitolo 7

Applicazione Web

La soluzione sviluppata fino a questo momento deve poter essere usata dagli operatori medici all'interno degli ospedali. Per questo motivo è stata realizzata un'applicazione web che permette l'utilizzo delle codifiche ASP e del risolutore CLINGO. Il programma è un'applicazione full-stack JavaScript con un'interfaccia grafica (GUI) sviluppata in Angular e un backend in Node.js.

L'architettura del sistema è mostrata in Figura 7.1: la parte centrale del sistema, defi-

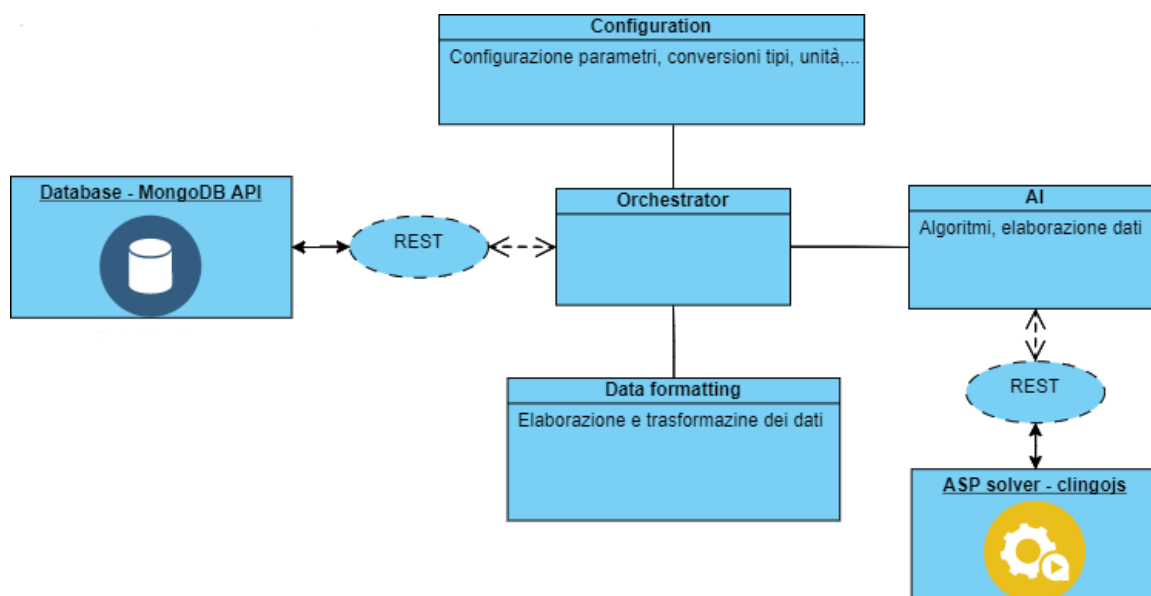


Figura 7.1: Architettura del sistema

nito come *Orchestrator*, collega tra loro tutte le parti dell'applicazione, gestendo il flusso

di dati tra di esse. Inoltre, funge da collegamento con l'interfaccia grafica, che può mostrare così i dati che le vengono inviati. Tutti i dati sono salvati all'interno di un database MongoDB. Sono presenti le sezioni di configurazione e formattazione dei dati, in modo da convertire i dati provenienti e diretti al database, così da permetterne l'utilizzo in ogni parte dell'applicazione. La sezione di AI contiene i codici inerenti alle codifiche ASP e il collegamento al risolutore ASP. I servizi del database e del risolutore vengono utilizzati tramite il servizio di un API RESTful. Per poter far uso del risolutore è stato utilizzato *clingojs*, un modulo per Node.js che implementa le funzionalità di CLINGO.

Osserviamo ora più nel dettaglio il flusso dei dati (Figura 7.2), per capire meglio il funzionamento dell'applicazione:

1. L'utente inserisce un input attraverso la GUI.
2. I dati vengono letti e rispetto ad essi vengono estratte dal database le informazioni necessarie.
3. I dati estratti vengono analizzati e trasformati: il formato letto è in JSON, e le informazioni devono essere tradotte in formato ASP, in modo che possano essere utilizzate successivamente con la nostra codifica. Altre informazioni aggiuntive potrebbero essere create, rispetto ai dati ricevuti.
4. Verificando i dati ricevuti, la parte di AI sceglie la codifica appropriata e invia i dati all'API di CLINGO.
5. Il risolutore restituisce il modello ottimo per la soluzione.
6. I dati ricevuti vengono nuovamente trasformati, da ASP a JSON.
7. I dati presenti nel database vengono aggiornati con quelli appena ricevuti.
8. La GUI si aggiorna di conseguenza.

7.1 Graphical User Interface

La GUI è divisa in due schermate: una per visualizzare il tabellone e l'altra per visualizzare l'agenda per ogni operatore. Nella schermata del tabellone, Figura 7.3, viene visualizzata una tabella che mostra per ogni operatore i pazienti assegnati ad esso. La visualizzazione riflette i dati presenti nel database. Nella parte alta è presente l'input dove possono essere inseriti i parametri relativi all'indisponibilità degli operatori e all'assenza di pazienti, per poter eseguire la ripianificazione. Dopo avere inviato i dati, all'ottenimento della soluzione,

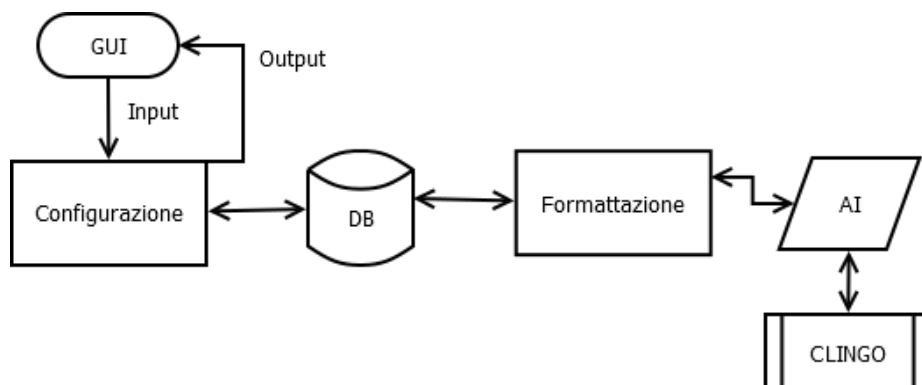


Figura 7.2: Flusso dei dati nell'applicazione

il tabellone, così come l'agenda, viene modificato secondo la nuova pianificazione. Inoltre vengono resi con chiarezza i cambiamenti avvenuti rispetto alla pianificazione originale (Figura 7.4): i pazienti riassegnati vengono colorati in verde; i pazienti assenti (assenza totale) vengono colorati in rosso; gli operatori non disponibili (indisponibilità totale) vengono colorati in rosso, oltre a non avere più nessun paziente assegnato.

Nella schermata dell'agenda (Figura 7.5), è presente un menù a tendina dal quale è pos-


<div>  Rehab Rescheduling Tabellone Agenda </div>	
Input per ripianificazione	
<div> <input type="text" value="Operatori non disponibili"/> <input type="text" value="Pazienti assenti"/> <input type="button" value="Invia"/> </div>	
Operatori	Pazienti
29	713
	718
	666
31	672
	707
32	671
	673
	1565
30	684
	709
	714

Figura 7.3: Schermata degli assegnamenti con input per la ripianificazione

sibile scegliere l'operatore di cui si vuole vedere l'agenda. Le sessioni sono visualizzate nei rispettivi orari e sono distinte dal colore: in blu sono le sessioni individuali, mentre in giallo quelle in supervisione. Per ogni sessione è specificato, il proprio identificativo, il

48	719
	692
	626
50	668
	711
49	684

Figura 7.4: Schermata degli assegnamenti ripianificata

paziente trattato e gli orari di inizio e fine; inoltre se la sessione è stata ripianificata, questo viene specificato. Infine, in grigio vengono identificati gli orari in cui l'operatore non lavora. Sono resi così con chiarezza la distinzione tra i turni e la presenza o meno di slot temporali liberi per lo svolgimento di ulteriori sessioni.


 Rehab Rescheduling Tabellone Agenda	
Selezione operatore	48
8 AM	
9 AM	Sessione 12057 - Paziente: 681 - Orario: 8:40 - 9:40
10 AM	Sessione 11916 - Paziente: 692 - Orario: 9:40 - 10:40
11 AM	Sessione 11835 - Paziente: 719 - Orario: 10:40 - 11:40
12 PM	
1 PM	Sessione 11807 - Paziente: 711 - Orario: 13:20 - 13:50 - Sessione ripianificata
2 PM	Sessione 11702 - Paziente: 572 - Orario: 13:50 - 14:50 Sessione 11917 - Paziente: 692 - Orario: 14:10 - 14:40
3 PM	Sessione 11391 - Paziente: 668 - Orario: 14:50 - 15:30 - Sessione ripianificata
4 PM	

Figura 7.5: Schermata dell'agenda ripianificata

7.2 Database

Per la gestione dei dati è stato scelto di usare MongoDB, un DBMS non relazionale, orientato ai documenti. È classificato come NoSQL, poichè non si basa sulla tradizionale struttura dei database relazionali, ma memorizza i dati in documenti in stile JSON con schema dinamico, denominati BSON (documenti JSON binari). I documenti sono raggruppati in collezioni, che possono essere anche eterogenee.

Le caratteristiche principali di MongoDB sono:

- Consente servizi di alta disponibilità.
- Garantisce la scalabilità automatica, in modo da supportare grandi quantità di dati senza influire eccessivamente sulle performance.

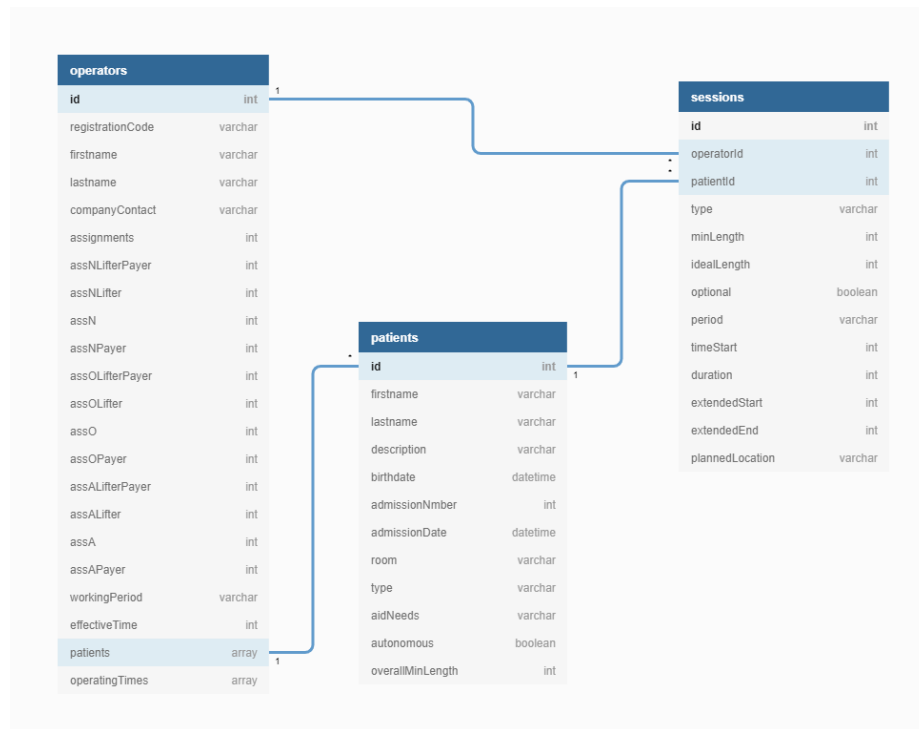


Figura 7.6: Diagramma delle relazioni del database

In Figura 7.6 vengono mostrate le collezioni salvate all'interno del database e i relativi campi. *operators* contiene tutte le informazioni relative al singolo operatore, sia personali che relative all'attività lavorativa (orari di lavoro, specializzazioni, ...). *patients* contiene le informazioni del paziente: l'identificativo di ogni paziente sarà presente al massimo all'interno di un solo documento degli operatori (campo *patients*). *sessions* contiene le informazioni sullo svolgimento delle sessioni; ogni documento di sessione è identificato univocamente anche dalla coppia di campi identificativo dell'operatore e identificativo del paziente. Nella figura sono mostrate anche le relazioni tra le collezioni, che, anche se non sono esplicitate all'interno del database, sono valide e considerate all'interno dell'applicazione per il recupero e l'utilizzo dei dati.

Poichè i dati vengono salvati in formato BSON, il database è protetto dai tradizionali attacchi di *SQL injection*, dato il tipo di codifica per i caratteri speciali. Inoltre tutti i dati

inseriti vengono prima validati, così che non sia possibile l'inserimento di dati inesatti, o inaffidabili.

7.3 API RESTful

Per poter accedere ai servizi del database e del solutore CLINGO, è stata creata un API REST.

Una API (Application Programming Interface, ovvero Interfaccia di programmazione delle applicazioni) è un insieme di procedure che facilitano la comunicazione con un sistema per il recupero di informazioni o l'esecuzione di una funzione. REST (REpresentational State Transfer) è un insieme dei seguenti vincoli architetturali:

- Interfaccia uniforme per i componenti, in modo che le informazioni vengano trasferite in modo standard.
- La comunicazione client-server deve essere *stateless*, ovvero non vengono memorizzate le informazioni del client; ogni richiesta è distinta.
- Le risorse possono essere memorizzate nelle cache per ottimizzare le interazioni.
- Client e server sono completamente indipendenti tra loro. L'unica informazione che il client possiede è l'URI della risorsa richiesta, che è l'unico modo per interagire con essa.
- Il sistema è organizzato su più livelli.

Una API REST comunica tramite richieste HTTP in modo da eseguire le funzionalità CRUD (Create, Read, Update e Delete). Quando una richiesta viene inviata, l'API trasferisce al client uno stato rappresentativo della risorsa, in questo caso in formato JSON. L'insieme delle seguenti operazioni compone questa API:

- POST (fornisce dati)
- GET (recupera un indice di risorse o una singola risorsa)
- PUT (crea o sostituisce una risorsa)

Grazie a questo insieme di operazioni è possibile inviare richieste al database, in modo da: richiedere un insieme di risorse o una singola risorsa rispetto al valore di un certo campo; o aggiornare il database, modificando particolari campi.

Per interrogare il risolutore CLINGO, invece, viene eseguita un'operazione POST tramite

la quale vengono inviati i dati che vengono trasformati in input per la codifica ASP. All'interno dell'API avviene la risoluzione della codifica ASP; il risolutore è impostato per restituire tutti i modelli, con l'ultimo che sarà perciò quello ottimo, visto il particolare algoritmo utilizzato che crea soluzioni sempre di migliore qualità [Rosa et al., 2008]. I dati contenuti all'interno di questo modello vengono estratti e restituiti al client. Essi compongono la soluzione della ripianificazione. Una volta ottenuti questi dati, il database viene aggiornato e di conseguenza si aggiorna anche la visualizzazione della GUI.

Capitolo 8

Lavori correlati

In questo capitolo vengono studiati altri lavori di ricerca sulla ripianificazione, sia in ambito ospedaliero generico, sia mirata alla riabilitazione, che hanno usato strumenti diversi rispetto a quelli di questa tesi. Poi vengono studiati altri problemi di ripianificazione risolti con l'uso dell'ASP. Infine si analizzano gli strumenti già realizzati che supportano l'ASP in un ambiente grafico.

8.1 Problemi di ripianificazione

Spesso gli ospedali vanno incontro a problemi che costringono a dover abbandonare la pianificazione già presente. La realizzazione di una ripianificazione è un problema presente negli ospedali di tutto il mondo [Mutingi and Mbohwa, 2017] ed è stato affrontato con diversi approcci nel corso del tempo, molti dei quali cercano di minimizzare le differenze con la pianificazione originale.

[Uhm et al., 2017] risolve la ripianificazione dei turni infermieristici con un algoritmo deterministico, una ricerca iterativa di tipo *depth-first*, che ottimizza sul tempo, sullo spazio usato e sul costo della soluzione; data la limitazione del problema su tempo e profondità la soluzione può non essere trovata e va approssimata con algoritmi euristici.

Vengono usate anche soluzioni euristiche, come quella di [Kitada and Morizawa, 2010]: in caso di mancanza di personale viene utilizzato una ricerca ricorsiva che trova la prima soluzione valida; nel processo viene costruito un albero, i cui nodi contengono gli operatori candidati a sostituire i posti mancanti; tramite una strategia di ricerca in questi nodi è possibile trovare la migliore ripianificazione. [Pato and Moz, 2008] invece usa un'euristica genetica che si basa sull'approccio Pareto, includendo una strategia di diversificazione utopica.

[Clark and Walker, 2011] usa una combinazione di somme pesate e programmazione obiet-

tivo, usando i pesi per definire la priorità; in un primo approccio cerca di minimizzare il numero di cambiamenti rispetto alla pianificazione originale, mentre nel secondo valuta come vari tipi di cambiamento possano influenzare in maniera diversa la ripianificazione. [Salbert, 2015] usa l'algoritmo del *Longest Processing Time*, adattandolo all'ambiente delle sale operatorie.

[Bäumelt et al., 2016] risolve la ripianificazione degli infermieri decomponendo il problema e utilizzando un algoritmo parallelo eseguito sulla GPU.

Per risolvere la ripianificazione degli infermieri [Maenhout and Vanhoucke, 2013] usa un algoritmo evolutivo, minimizzando i costi aggiuntivi e le insoddisfaccibilità e, in secondo piano, minimizzando i cambiamenti dei turni rispetto alla pianificazione originale.

In [Bard and Purnomo, 2006] viene considerata la possibilità di assumere personale temporaneo, per far fronte all'assenza di infermieri; questo approccio però può essere necessario in ospedali dove la portata del lavoro non può essere soddisfatta dal personale permanente, che non è il caso qui trattato.

8.2 Ripianificazione di sessioni riabilitative

[Bikker et al., 2020] propone un approccio per evitare di dover ricorrere a una ripianificazione, nel caso dell'arrivo di un paziente urgente che costringerebbe allo spostamento di un altro paziente o a tempi di attesa più lunghi; il problema è formulato come un processo decisionale di Markov che tiene conto della pianificazione dei pazienti e degli arrivi futuri.

[Zheng et al., 2020] si concentra sul problema attuale del Covid-19 e sviluppa un modello di rischio per predire la durata della riabilitazione dovuta a casi di Covid-19: il modello è stato realizzato tramite regressione lineare, basandosi su vari fattori di rischio e popolazioni a differente rischio.

[Cardellini et al., 2021] risolve il problema della pianificazione delle sessioni utilizzando l'ASP ed è il punto di partenza per lo sviluppo della ripianificazione all'interno di questa tesi.

Altre ricerche propongono varie soluzioni per la pianificazione di sessioni riabilitative, ma non includono alcuna soluzione alla ripianificazione: [Schimmelpfeng et al., 2010] utilizza la programmazione lineare mista a interi (MILPs); [Chien et al., 2008] formalizza il problema come un problema *job-shop* ibrido; [Chien et al., 2009] combina un algoritmo genetico al Data Mining per risolvere la pianificazione e ridurre i tempi di attesa dei pazienti.

8.3 ASP nei problemi di ripianificazione

[Alviano et al., 2019b, Dodaro and Maratea, 2017, Alviano et al., 2017] affrontano il problema della ripianificazione degli infermieri, che si verifica quando uno o più infermieri si rendono indisponibili; l'obiettivo è trovare una nuova pianificazione che copra quei turni, minimizzando la differenza con la pianificazione originale.

[Dodaro et al., 2019] realizza la ripianificazione per l'assegnazione dei pazienti alle sale operatorie; può accadere che un intervento chirurgico duri più del dovuto o che un paziente cancelli la propria registrazione e che quindi alcune registrazioni vadano spostate.

[Dodaro et al., 2021] realizza sempre la ripianificazione per l'assegnazione dei pazienti alle sale operatorie, tenendo conto anche dell'assegnazione dei letti.

[Eiter et al., 2021] usa l'ASP per gestire la ripianificazione di macchine parallele con tempi di attrezzaggio e date di rilascio dipendenti dalla sequenza, in cui i *jobs* possono essere elaborati solo dalle macchine specializzate.

[Garcia-Mata et al., 2015] si pone l'obiettivo di studiare le possibilità di ripianificazione, specialmente all'interno di fabbriche di semiconduttori, tramite l'utilizzo di nuove metodologie e tecnologie, tra cui l'ASP.

8.4 GUI per il supporto dell'ASP

Negli ultimi anni sono stati proposti vari strumenti per migliorare il design e facilitare l'utilizzo di applicazioni ASP. Ad esempio, *Integrated Development Environment* (IDEs), come ASPIDE [Febbraro et al., 2011] e SEALION [Busoniu et al., 2013] che supportano l'utente durante lo sviluppo tramite strumenti per lo sviluppo, il test e il debug di programmi ASP.

JDLV è un plug-in per la piattaforma Eclipse che integra la tecnologia ASP con l'ambiente di sviluppo Java. JDLV è basato su *JASP* [Febbraro et al., 2012], un linguaggio ibrido che permette la comunicazione tra ASP e Java.

In [Dasseville and Janssens, 2015] è presentato un ambiente di programmazione online per il sistema IDP, dotato di strumenti di supporto per lo sviluppo e il debug di programmi logici.

In [Ricca et al., 2011] viene risolto il problema del *Team Building* tramite una codifica ASP e viene presentata una GUI sviluppata in Java per permettere l'interazione con gli utenti.

Altre interfacce grafiche sono state presentate per lo sviluppo nei campi di *E-Tourism* [Ricca et al., 2010] e di *E-Learning* [Garro et al., 2006, Ianni et al., 2005].

Capitolo 9

Conclusioni e lavori futuri

9.1 Conclusioni

In questa tesi è stato risolto il problema della ripianificazione di sessioni riabilitative, necessaria nel caso non fosse più possibile utilizzare la pianificazione originale (e.g. indisponibilità di operatori, assenza di pazienti). Le specifiche del problema sono state espresse come regole ASP ed è stato usato il risolutore CLINGO. Le codifiche sono state realizzate a partire dai casi base più semplici, per poi combinarle ed ottenere la soluzione finale. Poi sono stati presentati i risultati dell'analisi sperimentale e di scalabilità, ottenuti tramite l'utilizzo di dati reali forniti da ICS Maugeri; sono stati analizzati tutti gli scenari, prendendo in considerazione anche i casi più particolari e le problematiche che possono derivarne.

La soluzione ottenuta rispetta tutti i requisiti richiesti e si rivela molto flessibile ad affrontare e risolvere ogni tipo di problema considerato: riesce a trattare il problema per ognuno degli scenari analizzati, dai casi base fino a quello più complesso, con orari parziali. Inoltre riesce ad ottenere soluzioni ottime in tempi molto brevi, sia per il riassegnamento dei pazienti, sia per lo spostamento delle sessioni, osservando, quindi, la necessità dell'applicazione di rispondere tempestivamente ad eventuali problemi e di poter riorganizzare la struttura di conseguenza.

Infine è stata sviluppata un'applicazione web che supporta l'esecuzione della soluzione di ripianificazione. Essa permette l'inserimento delle possibili indisponibilità e assenze e rende graficamente gli assegnamenti e le agende, con particolare attenzione alle modifiche: osservando l'applicazione è subito possibile comprendere i cambiamenti che offre la soluzione, così da poterli attuare con rapidità. Potrebbe quindi essere già implementata e utilizzata all'interno della struttura ospedaliera, così che gli operatori sanitari possano utilizzare la soluzione sviluppata.

Osservando i risultati ottenuti, è possibile dire che l'ASP è un valido strumento di intelligenza artificiale per risolvere problemi di ripianificazione, ciò dovuto anche al fatto di

poter modellare problemi complessi e reali tramite vincoli e regole, e alla presenza di solutori efficienti (e.g. CLINGO [Gebser et al., 2012] e WASP [Alviano et al., 2019a]).

In conclusione, dopo avere studiato i lavori di ricerca correlati, si osserva che non sono molti i lavori che risolvono il problema della ripianificazione delle sessioni riabilitative ospedaliere, e, a quanto risulta, la soluzione ottenuta in questa tesi è l'unica che per farlo utilizza tecniche di Intelligenza Artificiale.

9.2 Lavori futuri

In questa sezione vengono introdotti alcuni problemi che non sono stati affrontati, la cui risoluzione, però, potrebbe migliorare e rendere più realistica la soluzione.

Ottimizzazione. Come è stato mostrato nell'analisi, i tempi di risoluzione sono molto brevi, ma questo potrebbe essere dovuto, in parte, alle piccole dimensioni delle strutture ospedaliere considerate, con un basso numero di operatori e pazienti. Per ottenere una diminuzione dei tempi potrebbe essere possibile modificare le regole, in modo da restringere maggiormente l'insieme dei risultati. Altrimenti potrebbe essere utile modificare le impostazioni di CLINGO, cambiando così le modalità di risoluzione.

Ritardo nella ripianificazione. In questa tesi il problema è stato considerato come se gli avvenimenti avvenissero all'inizio della giornata lavorativa. Se, però, un imprevisto accadesse durante il suo svolgimento, la ripianificazione dovrebbe tenere conto delle sessioni che hanno già avuto luogo. Per far ciò si dovrebbero aggiungere dei vincoli che effettuino la ripianificazione solo negli orari dell'agenda seguenti all'ora in cui si è generata la nuova soluzione. Così facendo si restringe il campo delle soluzioni, e le sessioni avrebbero meno slot temporali a disposizione per lo spostamento.

Nuovo paziente. Si può ragionare sul verificarsi di un terzo avvenimento: l'aggiunta di un paziente inaspettato. La soluzione qui è relativamente semplice: con l'aggiunta dei dati del paziente nel sistema, la codifica lo assegnerebbe ad un operatore e aggiungerebbe la nuova sessione creata all'agenda, tenendo conto delle sessioni già presenti. I vincoli di ottimizzazione, in questo caso, terrebbero conto degli orari preferiti per l'inserimento della sessione.

Estensione alla multi-palestra. Un ulteriore miglioramento sarebbe l'estensione alla multi-palestra. L'ambiente trattato finora è composto da una singola palestra, dove si svolgono tutte le sessioni. Certe strutture possiedono però più palestre dove diverse sessioni si svolgono in contemporanea. Bisognerebbe quindi modificare la codifica così che possa supportare lo svolgimento di più sessioni anche in luoghi diversi, vincolando gli orari degli

operatori, in modo che non possano tenere più sessioni allo stesso tempo in luoghi diversi. Andrebbe inoltre modificato l'input, a cui si aggiungerebbero le diverse palestre, associate alla struttura a cui appartengono.

Explainability. Per migliorare la soluzione si potrebbe implementare l'*Explainability*, un supporto agli algoritmi di intelligenza artificiale che permette agli umani di comprenderne la soluzione. In questa tesi sono stati realizzati strumenti e tecniche che supportano gli operatori nella ripianificazione; perchè essi possano comprendere la soluzione generata e si affidino al sistema, è necessario che venga in qualche modo espresso il ragionamento intrapreso dal sistema per ottenere una data soluzione. Inoltre è possibile che non possa essere trovata una soluzione valida, che soddisfi tutti i vincoli. Sarebbe quindi utile una spiegazione sulle possibili cause del fallimento e su come agire per trovare una soluzione.

Elenco delle figure

2.1	Esempio di input e output della fase di riassegnamento	8
6.1	Il numero di pazienti assegnato ad ogni operatore	37
6.2	Tempo massimo e tempo occupato a confronto per ogni operatore	37
6.3	Riassegnamento dei pazienti con indisponibilità di 1, 2, 5, 8 operatori	38
6.4	Tempo di esecuzione (s) all'aumentare dei pazienti da riassegnare	40
6.5	Riassegnamento uniforme con indisponibilità di 1 operatore	40
6.6	Riassegnamento dei pazienti con indisponibilità parziale di 1 operatore . . .	41
6.7	Ripianificazione dell'agenda con 2 nuove sessioni (operatore 44)	42
6.8	Ripianificazione dell'agenda con 3 nuove sessioni (operatore 44, mattina) .	42
6.9	Ripianificazione dell'agenda: scelta dell'orario	43
6.10	Ripianificazione dell'agenda con 1 paziente assente	44
6.11	Tempo di esecuzione (s) all'aumentare dei pazienti assenti	45
6.12	Ripianificazione dell'agenda con 1 paziente assente e 3 nuove sessioni con priorità alle sessioni originali (solo orari individuali)	46
6.13	Ripianificazione dell'agenda con 1 paziente assente e 3 nuove sessioni con priorità alle nuove sessioni (solo orari individuali)	47
6.14	Ripianificazione dell'agenda con l'indisponibilità parziale dell'operatore e un'assenza parziale	48
6.15	Ripianificazione dell'agenda con un'assenza parziale	49
7.1	Architettura del sistema	50
7.2	Flusso dei dati nell'applicazione	52
7.3	Schermata degli assegnamenti con input per la ripianificazione	52
7.4	Schermata degli assegnamenti ripianificata	53
7.5	Schermata dell'agenda ripianificata	53
7.6	Diagramma delle relazioni del database	54

Elenco delle tabelle

2.1	Numero di operatori per qualifica e numero di pazienti per tipologia a confronto	8
6.1	Dimensioni degli istituti di ICS Maugeri	36
6.2	Numero di operatori per qualifica e numero di pazienti per tipologia a confronto	39
6.3	Numero di sessioni spostate rispetto al numero di pazienti assenti	44
6.4	Numero di sessioni che rispettano gli orari preferiti prima e dopo la ripianificazione rispetto al numero di pazienti assenti	45
6.5	Tempi medi di esecuzione (s) rispetto alla tipologia di scenario all'interno della codifica con scenari combinati	47

Bibliografia

- [Alviano et al., 2019a] Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., and Ricca, F. (2019a). 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 et al., 2017] 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 et al., 2019b] Alviano, M., Dodaro, C., and Maratea, M. (2019b). Nurse (re)scheduling via answer set programming1. *Intelligenza Artificiale*, 12:109–124.
- [Bard and Purnomo, 2006] Bard, J. and Purnomo, H. (2006). Incremental changes in the workforce to accommodate changes in demand. *Health care management science*, 9:71–85.
- [Bäumelt et al., 2016] Bäumelt, Z., Dvořák, J., Sucha, P., and Hanzálek, Z. (2016). A novel approach for nurse rostering based on a parallel algorithm. *Eur. J. Oper. Res.*, 251:624–639.
- [Bikker et al., 2020] Bikker, I. A., Mes, M. R., Sauré, A., and Boucherie, R. J. (2020). Online capacity planning for rehabilitation treatments: An approximate dynamic programming approach. *Probability in the Engineering and Informational Sciences*, 34(3):381–405.
- [Brewka et al., 2011] Brewka, G., Eiter, T., and Truszczyński, M. (2011). Answer set programming at a glance. *Commun. ACM*, 54:92–103.
- [Buccafurri et al., 2000] Buccafurri, F., Leone, N., and Rullo, P. (2000). Enhancing disjunctive datalog by constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 12:845 – 860.

- [Busoniu et al., 2013] Busoniu, P.-A., Oetsch, J., Puhrrer, J., Skocovsky, P., and Tompitis, H. (2013). Sealion: An eclipse-based ide for answer-set programming with advanced debugging support. *Theory and Practice of Logic Programming*, 13(4-5):657–673.
- [Calimeri et al., 2014] Calimeri, F., Gebser, M., Maratea, M., and Ricca, F. (2014). The design of the fifth answer set programming competition. *CoRR*, abs/1405.3710.
- [Cardellini et al., 2021] Cardellini, P. D. N., C. Dodaro, G. G., A. Giardini, M. M., and Porro, I. (2021). A two-phase asp encoding for solving rehabilitation scheduling. In *Proceedings of the 5th International Joint Conference on Rules and Reasoning*.
- [Chien et al., 2009] Chien, C.-F., Huang, Y.-C., and Hu, C.-H. (2009). A hybrid approach of data mining and genetic algorithms for rehabilitation scheduling. *IJMTM*, 16:76–100.
- [Chien et al., 2008] Chien, C.-F., Tseng, F.-P., and Chen, C.-H. (2008). An evolutionary approach to rehabilitation patient scheduling: A case study. *European Journal of Operational Research*, 189(3):1234–1253.
- [Cieza et al., 2020] Cieza, A., Causey, K., Kamenov, K., Hanson, S., Chatterji, S., and Vos, T. (2020). Global estimates of the need for rehabilitation based on the global burden of disease study 2019: a systematic analysis for the global burden of disease study 2019. *The Lancet*, 396.
- [Clark and Walker, 2011] Clark, A. and Walker, H. (2011). Nurse rescheduling with shift preferences and minimal disruption. *Journal of Applied Operational Research*, 3.
- [Dasseville and Janssens, 2015] Dasseville, I. and Janssens, G. (2015). A web-based IDE for IDP. *CoRR*, abs/1511.00920.
- [Dodaro et al., 2021] Dodaro, C., Galatà, G., Khan, M. K., Maratea, M., and Porro, I. (2021). Operating room (re)scheduling with bed management via ASP. *CoRR*, abs/2105.02283.
- [Dodaro et al., 2019] Dodaro, C., Galatà, G., Maratea, M., and Porro, I. (2019). An asp-based framework for operating room scheduling. *Intelligenza Artificiale*, 13(1):63–77.
- [Dodaro and Maratea, 2017] Dodaro, C. and Maratea, M. (2017). Nurse scheduling via answer set programming. In *LPNMR*, volume 10377 of *LNCS*, pages 301–307. Springer.
- [Eiter et al., 2021] Eiter, T., Geibinger, T., Musliu, N., Oetsch, J., Skocovsky, P., and Stepanova, D. (2021). Answer-set programming for lexicographical makespan optimisation in parallel machine scheduling.

- [Febbraro et al., 2012] Febbraro, O., Leone, N., Grasso, G., and Ricca, F. (2012). Jasp: A framework for integrating answer set programming with java. In *KR*.
- [Febbraro et al., 2011] Febbraro, O., Reale, K., and Ricca, F. (2011). Aspide: Integrated development environment for answer set programming. In Delgrande, J. P. and Faber, W., editors, *Logic Programming and Nonmonotonic Reasoning*, pages 317–330, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Garcia-Mata et al., 2015] Garcia-Mata, C., Marquez, P., and Burtseva, L. (2015). Rescheduling in industrial environments: Emerging technologies and forthcoming trends. *International Journal of Combinatorial Optimization Problems and Informatics*, 6:34–48.
- [Garro et al., 2006] Garro, A., Palopoli, L., and Ricca, F. (2006). Exploiting agents in e-learning and skills management context. *AI Commun.*, 19:137–154.
- [Gebser et al., 2016] Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Wanko, P. (2016). Theory solving made easy with clingo 5. In *ICLP*.
- [Gebser et al., 2012] Gebser, M., Kaufmann, B., and Schaub, T. (2012). Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187:52–89.
- [Gebser et al., 2017] Gebser, M., Maratea, M., and Ricca, F. (2017). The design of the seventh answer set programming competition. In Balduccini, M. and Janhunen, T., editors, *LPNMR*, volume 10377 of *Lecture Notes in Computer Science*, pages 3–9. Springer.
- [Gebser et al., 2020] Gebser, M., Maratea, M., and Ricca, F. (2020). The seventh answer set programming competition: Design and results. *Theory and Practice of Logic Programming*, 20(2):176–204.
- [Gelfond and Lifschitz, 1991] Gelfond, M. and Lifschitz, V. (1991). Lifschitz, v.: Classical negation in logic programs and disjunctive databases. new generation computing 9, 365–385. *New Generation Computing*, 9:365–385.
- [Huang et al., 2012] Huang, Y.-C., Zheng, J.-N., and Chien, C.-F. (2012). Decision support system for rehabilitation scheduling to enhance the service quality and the effectiveness of hospital resource management. *Journal of The Chinese Institute of Industrial Engineers*, 29:348–363.
- [Ianni et al., 2005] Ianni, G., Panetta, C., and Ricca, F. (2005). Specification of assessment-test criteria through asp specifications. volume 142.
- [Kitada and Morizawa, 2010] Kitada, M. and Morizawa, K. (2010). A heuristic method in nurse rostering following a sudden absence of nurses.

- [Maenhout and Vanhoucke, 2013] Maenhout, B. and Vanhoucke, M. (2013). An artificial immune system based approach for solving the nurse re-rostering problem. In Middendorf, M. and Blum, C., editors, *Evolutionary Computation in Combinatorial Optimization*, pages 97–108, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Meskó B, 2017] Meskó B, Drobni Z, B. G. B. G. Z. (2017). Digital health is a cultural transformation of traditional healthcare. *mHealth*, 3 38.
- [Mutingi and Mbohwa, 2017] Mutingi, M. and Mbohwa, C. (2017). The nurse rerostering problem : An explorative study.
- [Pato and Moz, 2008] Pato, M. and Moz, M. (2008). Solving a bi-objective nurse rerostering problem by using a utopic pareto genetic heuristic. *J. Heuristics*, 14:359–374.
- [Ricca et al., 2010] Ricca, F., Dimasi, A., Grasso, G., Ielpa, S. M., Iiritano, S., Manna, M., and Leone, N. (2010). A logic-based system for e-tourism. *Fundam. Informaticae*, 105:35–55.
- [Ricca et al., 2011] Ricca, F., Grasso, G., Alviano, M., Manna, M. L., Lio, V., Iiritano, S., and Leone, N. (2011). Team-building with answer set programming in the gioia-tauro seaport. *Theory and Practice of Logic Programming*, 12:361 – 381.
- [Rosa et al., 2008] Rosa, E. D., Giunchiglia, E., and Maratea, M. (2008). A new approach for solving satisfiability problems with qualitative preferences. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N. M., editors, *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 510–514. IOS Press.
- [Salbert, 2015] Salbert, A. S. (2015). Operating room rescheduler.
- [Schimmelpfeng et al., 2010] Schimmelpfeng, K., Helber, S., and Kasper, S. (2010). Decision support for rehabilitation hospital scheduling. *Operations Research-Spektrum*, 34.
- [Stucki et al., 2005] Stucki, P. D. m. G., Stier-Jarmer, M., Grill, E., and Melvin, J. (2005). Rationale and principles of early rehabilitation after an acute injury or illness. *Disability and rehabilitation*, 27:353–9.
- [Uhm et al., 2017] Uhm, S., Ko, Y.-W., and Kim, J. (2017). A deterministic approach to nurse rerostering problem. *International Journal of Applied Engineering Research*, 12:14246–14250.

- [Zdeněk et al., 2015] Zdeněk, B., Dvořák, J., Sucha, P., and Hanzálek, Z. (2015). A novel approach for nurse rostering based on a parallel algorithm. *European Journal of Operational Research*, 251.
- [Zheng et al., 2020] Zheng, Q.-N., Xu, M.-Y., Zheng, Y.-L., Wang, X.-Y., and Zhao, H. (2020). Prediction of the rehabilitation duration and risk management for mild-moderate covid-19. *Disaster Medicine and Public Health Preparedness*, 14(5):652–657.

Ringraziamenti

Ringrazio il Prof. Marco Maratea, che mi ha sempre dato buoni consigli, e mi ha seguito nello svolgimento e nella scrittura della tesi.

Ringrazio inoltre il Dott. Giuseppe Galatà per l'aiuto datomi nello sviluppo della tesi.

Ringrazio i miei genitori per il sostegno e per avermi accompagnato fino a questo giorno, e la mia fidanzata Valeria per il suo supporto e per essermi stata sempre accanto.