Università degli Studi di Milano
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Tecnologie dell'Informazione

Progetto di un software per ottimizzare i turni degli operatori nelle case di riposo

Relatore: Prof. ROBERTO CORDONE
Correlatore: Dott. MATTEO SALANI

Tesi di MICHELE MILESI
Matricola nr. 566535

15 Dicembre 2004

INDICE

SOMMARIO

La cooperativa FT Service di Brescia fornisce parte del personale alla casa di riposo di Ghedi. A tal fine, le assegna 16 operatori in pianta stabile, mentre altri operatori stagionali coprono, in periodi critici, le eventuali emergenze generate da una concentrazione di ferie o malattie.

Le richieste, variabili nel tempo, della casa di riposo e la momentanea indisponibilità degli operatori, per malattia o ferie, rendono difficile la gestione dei turni del personale. Ogni mese, infatti, il responsabile dei turni spende almeno quattro ore per pianificare il periodo e, giornalmente, dedica un'ora per gestire gli imprevisti. Oltre a questa gestione *normale* si verificano dei picchi di lavoro che richiedono un alto numero di straordinari o, addirittura, l'utilizzo di lavoratori esterni, con conseguente aumento dei costi.

I responsabili di FT Service si trovano di fronte a tre esigenze. La prima esigenza consiste nel ridurre il tempo speso, da parte del responsabile del servizio, per determinare i turni. La seconda esigenza chiede di diminuire, se non azzerare, il costo degli straordinari e l'utilizzo di operatori esterni. A questi due problemi, di carattere prettamente economico, si aggiunge la necessità di rispettare i vincoli normativi che regolano il diritto del lavoro. Non è una cosa semplice: il numero limitato di operatori può portare a stendere soluzioni che non rispettino interamente la normativa, pur di coprire tutte le richieste. La violazione delle leggi può portare a sanzioni amministrative e, per casi gravi, alla reclusione.

FT Service ha contattato la piccola cooperativa Crema Ricerche Optisoft per realizzare un software per la gestione dei turni.

Questo lavoro ha portato a quattro progetti principali:

1. analizzare in modo completo il problema, per rendere l'applicazione piú aderente alla realtà del mondo del lavoro italiano. Definendo un modello aderente al caso specifico ma possibilmente, tanto flessibile da potersi estendere ad altre realtà, e verificandone l'applicabilità tramite un prototipo

2. fornire in tempi rapidi una soluzione ammissibile di buona qualità, se non ottima. Per raggiungere questo scopo si utilizza un solutore di programmazione lineare e diverse procedure euristiche.

3. progettare la versione definitiva del software, cercando di definire un framework che possa rendere l'applicazione scalabile e altamente adattabile ad altre realtà.

4. sperimentare il software, e le procedure che lo caratterizzano, direttamente in una realtà lavorativa

L'uso di solutore di Programmazione Lineare applicato direttamente al modello matematico, tradotto in un opportuno linguaggio di modellazione, consente di: realizzare un test di inammissibilità, di ottenere informazioni utili per le procedure euristiche e una stima per difetto del costo della soluzione ottima. Le procedure euristiche, a loro volta, permettono di definire una soluzione ammissibile o di migliorarne una ottenuta in precedenza. Il software consente di definire delle interfacce di facile utilizzo, permettendo cosi, anche a persone non esperte di modellazione o programmazione, di utilizzare le tecniche descritte in precedenza. Una attenta progettazione del software permette di ridurre il lavoro necessario ad adattare l'applicazione ad altre realtà. La sperimentazione permette di valutare quantitativamente l'effettiva risoluzione dei problemi della cooperativa e, quindi, nell'ordine:

1. la riduzione dei tempi di calcolo;
2. la diminuzione del ricorso a straordinari operatori extra;
3. il rispetto delle regole, rigido per alcune o con violazioni ridotte al minimo per altre.

Questa valutazione ha permesso di individuare limiti pratici dei vari strumenti impiegati, rispetto sia alla dimensione che sia alla tipologia delle istanze del problema.

Attualmente è in corso la realizzazione dell'applicazione finale per FT Service, ed è in fase di studio la personalizzazione dell'applicazione per la catena di negozi d'abbigliamento US Fashion Store. A breve potrà partire la fase di sperimentazione pratica in ben due campi d'applicazione differenti.

1. DESCRIZIONE DEL PROBLEMA

> "Le lettere lunari sono rune, ma non le si può vedere," disse Elrond "non quando le si guarda direttamente. Si può vederle soltanto quando la luna brilla dietro di esse, ma ciò che conta di piú, anzi il punto fondamentale, è che la luna deve trovarsi nella stessa fase e nella stessa stagione di quando le lettere furono scritte."
>
> *Lo Hobbit* - J. R. R. Tolkien

Sommario

In questo capitolo si descrive il problema della definizione dei turni dal punto di vista dell'organizzazione del lavoro. Il problema è regolamentato sia dalle leggi nazionali che dal Contratto Collettivo Nazionale del Lavoro UNEBA. Altri vincoli che caratterizzano il problema derivano dall'organizzazione della casa di riposo. In ultimo è necessario tener conto dei vincoli di natura sociale, cioè della soddisfazione del personale, che è fondamentale nella realizzazione pratica delle soluzioni proposte.

1.1 Riferimenti normativi

In Italia il rapporto di lavoro è regolamentato da un ampio corpus legislativo. A partire da alcuni regi decreti del 1923, queste norme sono state via via modificate secondo le necessità del periodo storico. Le ultime normative in materia sono definite dal Decreto Legislativo 66/2003 [Ita03], che recepisce alcune norme di carattere comunitario, successivamente modificato ed integrato dal Decreto Legislativo 213/2004 [Ita04]. Queste norme definiscono, in modo generale:

1. i diritti e doveri del dipendente;
2. i diritti e doveri del datore di lavoro;
3. le tipologie di lavoro (a tempo pieno o parziale, a tempo indeterminato o determinato, ...);
4. l'orario lavorativo e gli straordinari;
5. ferie, riposi e malattie;
6. sanzioni disciplinari, nei confronti del dipendente;
7. sanzioni amministrative e penali, nei confronti del datore di lavoro;

8. rapporti con il mondo sindacale.

Alla legislazione si affiancano i Contratti Collettivi Nazionali del Lavoro o CCNL. Questi contratti, definiti di comune accordo dalle parti sociali, per i lavoratori, dai rappresentati di categoria, per i datori di lavoro, e dal governo, hanno il compito di adattare le norme nazionali ai singoli ambiti produttivi. Ai contratti collettivi è consentito, con alcune restrizioni, modificare alcuni parametri, individuare delle eccezioni o aggiungere delle limitazioni ancora piú stringenti di quelle definite dal legislatore. Spetta ai contratti collettivi specificare i parametri di remunerazione dei lavoratori. Alcuni aspetti particolari dei contratti collettivi, quali ad esempio la retribuzione, possono essere ridefiniti in sede locale, sia essa regionale o provinciale. Ogni Contratto Collettivo deve poi essere concordato, a livello individuale, da ogni singola azienda rispetto ai propri dipendenti. Questa ultima trattativa, detta anche contratto individuale, tende, normalmente, a specificare i punti lasciati aperti dal CCNL, quali ad esempio l'orario giornaliero, ed ad introdurre altri benefici. I Contratti Collettivi sono ridiscussi periodicamente, tipicamente ogni due o tre anni.

Ne risulta che, per poter determinare una turnazione del personale in modo corretto, è necessario considerare fino a quattro fonti normative, oltre alle esigenze produttive, di organizzazione del lavoro interna all'azienda e alle preferenze dei lavoratori.

Il Contratto Collettivo Nazionale di riferimento, per le case di riposo, è il contratto UNEBA[1]. Il CCNL UNEBA regola il rapporto di lavoro del personale dipendente degli istituti operanti nel campo del sociale, per attività educative, di assistenza e di beneficenza, nonché di culto o religione dipendenti dall'autorità ecclesiastica. Il contratto può essere esteso anche al personale dipendente di altre istituzioni che dichiarino di accettarne completamente la disciplina nel contratto individuale di lavoro.

Le istituzioni per il quale è applicabile il Contratto Collettivo sono, ad esempio:

- colonie marine e montane, case per ferie, accoglienza pellegrini, pensionati e/o patronati per studenti;
- case per esercizi spirituali;
- istituti che perseguono, a norma delle costituzioni o dello statuto, finalità di culto, religione, assistenza e beneficenza, quali servizi per soggetti in stato di disagio sociale, comunque denominati (Comunità di accoglienza, centri di assistenza, ...);
- istituzioni preposte alla gestione di attività artistico-culturali, quali, ad esempio, catacombe, musei, biblioteche, ecc.;
- istituzioni che gestiscono servizi di tipo socio-assistenziale, previsti dalle attuali disposizioni legislative regionali, quali ad esempio:

[1] Unione Nazionale Istituzioni e Iniziative di Assistenza Sociale

- istituti di assistenza domiciliare;
- case albergo;
- case protette;
- case di riposo con reparto protetto;
- residenza sanitaria assistenziale (RSA);
- servizi per minori comunque denominati (Istituti educativo - assistenziali, comunità alloggio);
- centri di aggregazione giovanile;
- servizi di animazione;
- comunità terapeutiche per tossicodipendenti ed alcooldipendenti;
- servizi per portatori di handicap, comunque denominati (istituti assistenziali, centri di riabilitazione, istituti psico-medico-pedagogici, comunità alloggio);
- centri di psicoterapia dell'età evolutiva;
- centri culturali, ricreativi e sportivi;
- uffici o centri retti da Curie, Diocesi, Parrocchie o Associazioni ecclesiali;
- istituzioni rette da persone fisiche appartenenti al clero secolare o regolare;
- Onlus che gestiscono attività private sociali rispondenti alla matrice culturale cattolica.

Questa lista, non esaustiva, dà un'idea del campo d'impiego per un modello e il conseguente software, che si occupi della gestione dei turni del personale. In particolare, le case di riposo cadono nella categoria delle residenze sanitarie assistenziali. La normativa presente nel Contratto Collettivo deve essere applicata a tutto il personale assunto a tempo indeterminato e, se possibile, anche a quello con rapporto di lavoro a tempo indeterminato [ACCU03].

Il CCNL UNEBA, almeno fino ad oggi, viene definito a livello nazionale. A livello regionale possono essere apportate delle integrazioni per lo piú di carattere economico [UCCU01].

1.2 Norme che influenzano l'organizzazione del lavoro

1.2.1 Orario di lavoro

L'orario lavorativo determina i limiti quantitativi delle prestazioni richieste al lavoratore. L'orario lavorativo determina anche la distribuzione dei giorni lavorativi nella settimana e l'inizio e la fine di ogni giorno lavorativo. Per legge l'orario lavorativo è fissato a 40 ore settimanali. I singoli contratti collettivi di categoria possono diminuire questo limite. Il lavoratore ha diritto ad un riposo completo di 24 ore nell'arco della settimana.

Orario	Lun	Mar	Mer	Gio	Ven	Sab	Dom
Settimana da 5 giorni							
40 h	8h	8h	8h	8h	8h	-	-
38 h	7h e 36'	7h 36'	7h 36'	7h 36'	7h 36'	-	-
38 h	8h	8h	8h	8h	6h	-	-
Settimana da 6 giorni							
40 h	6h 40'	6h 40'	6h 40'	6h 40'	6h 40'	6h 40'	-
40 h	6h	6h 30'	6h 30'	7h	7h	7h	-
38 h	6h 20'	6h 20'	6h 20'	6h 20'	6h 20'	6h 20'	-

Tab. 1.1: Esempi di distribuzione dell'orario lavorativo settimanale.

L'orario settimanale può essere distribuito su cinque o sei giorni. In una settimana da cinque giorni il sabato è il giorno di riposo mentre la domenica viene considerata una giornata festiva. Per le settimane da sei giorni il riposo coincide con la domenica. L'orario lavorativo ed il numero di giornate lavorative in una settimana determinano la durata di una giornata lavorativa. In tabella 1.1 sono riportati alcuni esempi di distribuzioni. Come si nota dalla tabella, le giornate di una settimana lavorativa possono anche non avere tutte la medesima durata. La definizione precisa della distribuzione dell'orario, unitamente agli orari di inizio e fine lavoro, spetta ai singoli datori di lavoro. In ogni caso al lavoratore deve essere garantito un riposo di almeno 11 ore consecutive tra una giornata lavorativa e la successiva.

Straordinari e lavoro supplementare

L'orario lavorativo determina un limite che, in caso di necessità, può essere superato. Le ore extra lavorate possono essere definite come lavoro supplementare o straordinari. Si indica come lavoro supplementare il numero di ore che eccedono l'orario base, ma non quello di legge. Tutte le ore che superano il limite legale sono definite ore straordinarie. Generalmente le ore straordinarie sono pagate con una maggiorazione. Il numero massimo di ore straordinarie che un lavoratore può effettuare è pari a 250 ore all'anno. In ogni caso, di norma, non è possibile effettuare piú di 48 ore in una settimana. In casi eccezionali è possibile superare questo limite, previa comunicazione alla Direzione Provinciale del Lavoro.

Lavoratori a tempo parziale

Nel caso dei lavoratori a tempo parziale, o part time, l'orario lavorativo viene ridotto secondo un'accordo tra il lavoratore ed il datore di lavoro. La distribuzione delle ore e dei giorni lavorativi può essere differente rispetto a quella relativa all'orario lavorativo standard. Gli straordinari sono pagati con una maggiorazione ulteriore fino al raggiungimento dell'orario lavorativo standard, oltre questo valore sono pagati come per un lavoratore a tempo completo. Nel caso che il CCNL introduca il lavoro supplementare, è compito del Contratto

Collettivo definire quanta parte del lavoro eccedente l'orario base debba essere considerato come lavoro supplementare. Per tutti gli altri aspetti un lavoratore part time è equiparato ad uno a tempo completo.

Lavoro notturno

Generalmente viene definito turno notturno qualsiasi turno che abbia almeno un'ora nel periodo che va dalle 24 alle 6. Ogni contratto collettivo può allargare questo periodo. Non possono essere adibiti al lavoro notturno i minorenni, le donne in stato interessante e le madri con figli di età inferiore ai 2 anni. Nessun lavoratore può effettuare piú di 8 ore di lavoro notturno nell'arco di 24 ore.

1.2.2 Ferie, permessi e malattie

Ferie

Ogni lavoratore ha diritto ad un periodo di ferie di almeno 4 settimane. Di queste almeno due devono essere consumate durante l'anno, mentre le restanti devono essere consumate nei 18 mesi successivi. Le ferie sono concordate tra il lavoratore ed il datore di lavoro. Può quindi esistere un limite alle ferie che il dipendente può godere in modo continuativo, e le ferie possono essere richieste dal dipendente o determinate, a seconda delle necessità, dal datore di lavoro. Non è possibile ottenere un'indennità sostitutiva per le ferie non godute, a meno di una risoluzione del rapporto.

Permessi retribuiti

I contratti collettivi possono introdurre una limitata riduzione dell'orario su base annua. Queste ore di riduzione vanno a formare un monte permessi retribuiti, dal quale il lavoratore può attingere sia in misura oraria che giornaliera.

Il monte ore matura col progredire dell'anno lavorativo. Ad esempio, se il CCNL prevede 24 ore di riduzione, ogni mese lavorato il monte permessi viene incrementato di 2 ore (24 ore / 12 mesi = 2 ore / mese). Alla fine dell'anno i permessi non goduti danno luogo ad un'indennità sostitutiva.

Malattie

In caso di necessità il servizio sanitario nazionale può assegnare al dipendente alcune giornate di riposo. In questo caso il dipendente è giustificato nel non presentarsi al lavoro, ma non può prolungare l'assenza oltre il periodo certificato da un medico. Se il dipendente è in ferie le giornate di malattia non sono conteggiate nel novero delle ferie consumate. Per periodi inferiori a tre giorni non è richiesto il certificato medico.

Lavoro notturno	15%
Lavoro festivo	15%
Lavoro festivo notturno	30%
Lavoro diurno straordinario	35%
Lavoro notturno straordinario	40%
Lavoro festivo diurno straordinario	50%
Lavoro festivo notturno straordinario	60%

Tab. 1.2: Maggiorazioni retributive per straordinari, lavoro notturno e festivo

1.3 Contratto Collettivo UNEBA

Orario di lavoro

Il CCNL UNEBA stabilisce l'orario di lavoro in 38 ore settimanali. L'articolazione degli orari viene definita tenendo conto delle necessità organizzative dell'istituto. L'orario di lavoro è distribuito, nell'arco della settimana, in modo da garantire un giorno di riposo, normalmente coincidente con la domenica. Al lavoratore, al quale è richiesto di effettuare una prestazione lavorativa durante il riposo settimanale, spetta un riposo compensativo.

Riduzione d'orario

Al dipendente viene garantito un monte ore permessi pari a 24 ore. Questo monte ore matura nella misura di 2 ore ogni mese lavorato.

Straordinari

Le ore straordinarie sono misurate, ogni mese, rispetto al totale delle ore lavorate nella media dei sei mesi precedenti, scontando le ore già pagate, durante quei mesi, come straordinarie. Questo comporta che è possibile gestire una banca ore nella quale far confluire gli straordinari. Le ore presenti in banca possono essere utilizzate per compensare ore eventualmente non lavorate. Questa banca ore può essere gestita anche in modo opposto, facendovi confluire le ore non lavorate, le quali copriranno successive ore straordinarie. Le ore straordinarie possono quindi essere compensate da periodi d'astensione dal lavoro. Nel caso le ore non siano compensate, queste devono essere pagate con una maggiorazione, come riportato in tabella 1.2. Un lavoratore può effettuare al massimo 120 ore di lavoro straordinario nell'arco di un anno.

Lavoratore part time

Un lavoratore part time può effettuare un numero di ore di lavoro supplementare pari al 15% del proprio orario di lavoro. Ad esempio un lavoratore part time assunto a 19 ore può effettuare 2 ore e 51 minuti di lavoro supplementare alla settimana. Queste ore sono pagate normalmente. Le ore che eccedono questo valore sono considerate come lavoro straordinario.

Lavoro notturno e festivo

Viene considerato lavoro notturno tutto il lavoro svolto dalle ore 22 alle 6. Ai lavoratori che effettuano lavoro notturno, o festivo, spetta una maggiorazione del compenso, come riportato in tabella 1.2.

Festività Sono considerate festività le domeniche e le seguenti date:

- Capodanno;
- Epifania;
- Anniversario della Liberazione (25 aprile);
- Lunedí di Pasqua;
- Festa del lavoro (1° maggio);
- Festa della Repubblica (2 giugno);
- Assunzione della Madonna;
- Festa di tutti i Santi;
- Immacolata Concezione;
- Santo Natale;
- Santo Stefano;
- Santo Patrono.

Nel caso che la festività cada nel giorno di riposo, al dipendente spetta un compenso pari al normale importo di una giornata lavorativa, normalmente un ventiseiesimo della paga mensile.

Ferie

Al lavoratore spettano 33 giorni di ferie calcolati in un settimana di sei giorni, per un totale di cinque settimane e mezzo effettive. Il dipendente non può richiedere piú di tre settimane di ferie consecutive. La Direzione dell'Istituto deve determinare, possibilmente concordandole con il dipendente, due settimane di ferie nel periodo che va dal 1 giugno al 30 settembre. Le restanti ferie sono a discrezione del dipendente, previo accordo con la Direzione dell'Istituto.

Turno	Orario	Verde	Giallo	Blu	Rosa	Richiesta
Notte	0:00 - 7:00	1	1	1	1	4
Mattina	6:00 - 13:00	4	4	4	4	16
Pomeriggio	13:00 - 20:00	1	1	1	1	4
Pomeriggio 2	14:00 - 21:00	1	1	1	1	4
Sera	17:00 - 24:00	1	1	1	1	4
3,5V	7:00 - 10:30	1				1
3,5b	6:00 - 9:30					1
3,5m	9:00 - 12:30					1
mb	7:00 - 14:00					1
b/m	6:00 - 13:00	Saltuario				

Tab. 1.3: Turni presenti alla casa di riposo di Ghedi

1.4 Vincoli organizzativi

Spesso ogni istituto è una realtà a sè, caratterizzata da particolari consuetudini e necessità. A seconda di queste i singoli istituti si possono dotare di una propria organizzazione del lavoro, la quale non può però contrastare con le normative vigenti. In particolare la casa di riposo di Ghedi è strutturata in nuclei, organizza il proprio lavoro tramite sequenze di tre turni lavorativi ed un riposo (denominate 3+1) e utilizza un pattern ciclico per assegnare i turni alla maggior parte degli operatori. Nel seguito descriviamo in dettaglio questi aspetti.

1.4.1 I Nuclei

La casa di riposo è strutturata in reparti o nuclei. In particolare sono presenti quattro reparti, identificati da un colore: nucleo Verde, nucleo Giallo, nucleo Blu e nucleo Rosa. Alcuni operatori operano principalmente in un nucleo solo. Questo permette la conoscenza sia tra gli operatori che tra gli operatori e gli ospiti di un nucleo.

I Turni

La tabella 1.3 riporta i vari turni, con le loro caratteristiche. Generalmente per ogni nucleo sono definiti i seguenti turni: Notte, Mattina, Pomeriggio, Pomeriggio 2 e Sera. Per il nucleo verde è definito un ulteriore turno (3,5V). A questi turni se ne affiancano altri che non sono associati ad un nucleo. Generalmente ogni turno deve essere effettuato da un solo operatore per nucleo, ma per alcuni turni la richiesta può essere maggiore (ad esempio, il turno della Mattina prevede 4 operatori per nucleo). I turni 3,5b e 3,5m possono essere raggruppati e sostituiti da un terzo turno b/m nelle giornate, note a priori, in cui mancano entrambi gli operatori dedicati a questi turni. Come vedremo successivamente, la cooperativa FT Service ha il compito di gestire il nucleo verde.

1.4.2 Turnazione 3+1

L'istituto, in particolar modo per gli operatori a tempo pieno, definisce una turnazione che prevede una successione di tre giornate consecutive, con il medesimo turno, ed una giornata di riposo. Questa turnazione viene chiamata del 3+1. Tramite il 3+1 è possibile suddividere gli operatori in due gruppi. Il primo, al quale appartiene chi aderisce al 3+1, ha il compito di garantire la copertura dei turni dei vari reparti. I lavoratori del secondo gruppo, spesso a tempo parziale, tappano i buchi alla copertura generati dalle assenze per ferie e malattie degli operatori sia del primo gruppo che sia degli altri nuclei . La turnazione 3+1 ha due vantaggi: permette di valutare facilmente il numero di operatori necessari e di forzare gli operatori a consumare consumare, durante l'anno, parte delle ferie e dei permessi. In questo modo è possibile evitare di pagare, a fine anno, i permessi non goduti e si rispettano le norme che prevedono che un certo numero di giorni di ferie siano effettuate entro la fine dell'anno.

Dimensionamento del personale di un nucleo

Ogni nucleo richiede la copertura di 8 turni ogni giorno (4 Mattine, 1 Pomeriggio, 1 Pomeriggio 2, 1 Sera e 1 Notte). In una turnazione del 3+1 è disponibile, ogni giorno, il 75% degli operatori. Sono quindi necessari 11 operatori a tempo pieno: 8 è infatti il 75% di 10,67 che, arrotondato all'intero superiore, dà 11. Ferie e permessi coprono in media il 9% dell'anno, per cui serve un altro operatore; 10,67 è il 91% di 11,72 che arrotondato da 12. Con 12 operatori è possibile coprire tutti i turni del nucleo e se il tasso di malattia si mantiene sotto il 2,8%, non è necessario nessun altro operatore: infatti la differenza tra i 12 operatori effettivi e gli 11,72 necessari è di 0,33 operatori che è circa il 2,8% del totale.

Recupero dei riposi come ferie e permessi

In media un mese è costituito da 26 giornate lavorative. Un anno composto da 365 giorni è suddivisibile in 12 mesi mediamente di 30,42 giorni. Il che, ricondotto alla settimana da 6 giorni, genera un mese composto da 26,07 giornate di lavoro effettivo. Lavorando in media, secondo l'orario di lavoro da 38 ore, 6 ore e 33 minuti al giorno si ottiene che il carico medio delle ore lavorate, nel mese, deve essere pari a 165 ore, per essere precisi 165,13 ore. La turnazione del 3+1 garantisce 23 giornate lavorative[2] per un totale, essendo il turno medio di 7 ore, di 161 ore lavorative. Le 4 ore mancanti possono essere recuperate come ferie e/o permessi. In questo modo si consumano i permessi, i quali andrebbero retribuiti al termine dell'anno, e le ferie che devono essere completamente godute nell'arco di 30 mesi dall'inizio dell'anno[3].

[2] Anche in questo caso è piú corretto dire 22,82 giornate.

[3] Piú precisamente, 2 settimane nell'anno solare ed il rimanente entro 18 mesi dalla fine dell'anno.

Giorno	1	2	3	4	5	6	7	8	9	10	11	12
Turno	N	N	N	R	S	S	S	R	P2	P2	P2	R
Giorno	13	14	15	16	17	18	19	20	21	22	23	24
Turno	M	M	M	R	P	P	P	R	M	M	M	R

Tab. 1.4: Sequenza dei turni

1.4.3 Pattern ciclici di turni

La casa di riposo di Ghedi, come molte strutture ospedaliere, utilizza una sequenza ciclica di turni per assegnare i compiti di alcuni operatori. Questa sequenza, se rispettata, presenta due vantaggi: l'operatore memorizza piú facilmente i turni e alla lunga gli operatori con orari simili formano gruppi affiatati.

Conoscenza del turno

L'operatore memorizza con estrema facilità il suo orario e può, quindi, regolare i suoi impegni esterni piú semplicemente. Questo permette all'operatore, e alla sua famiglia, di godere maggiormente del tempo libero a disposizione.

Conoscenza tra gli operatori

Se il ciclo viene sempre rispettato, alla lunga, gli operatori che effettueranno i medesimi turni, anche tra nuclei diversi, tendono ad essere sempre gli stessi. In questo modo è possibile creare dei gruppi di lavoro affiatati tra di loro, i quali possono sostenersi a vicenda in caso di necessità.

La sequenza impiegata a Ghedi, è caratterizzata da di sei gruppi, composti da tre turni identici ed un riposo (tabella 1.4). Il primo operatore effettua la sequenza partendo dal primo giorno. Il secondo operatore inizia la sequenza tre giorni piú tardi mentre il terzo inizia sei giorni piú tardi, e cosí via. Il fatto che la sequenza abbia un periodo di 24 giorni e che ogni operatore sia sfasato di 3 giorni rispetto al precedente, determina che sono necessari 8 operatori per tornare al punto di partenza e coprire in modo uniforme la sequenza completa. La sequenza applicata agli 8 operatori è riportata in tabella 1.5 nelle righe etichettate da 1 a 8. Come si vede, in ogni sequenza si eseguono tutte le tipologie di turno. Ogni giorno sei operatori coprono Notte, Sera, Pomeriggio, Pomeriggio 2 e due Mattine e gli altri due operatori sono in riposo. Quindi, sono necessari almeno altri due operatori per completare i rimanenti turni mattutini non assegnati.

A questo pattern se ne può affiancare un altro, sempre presentato in tabella 1.5, che permette di gestire anche i rimanenti 4 operatori, in modo da ricondurci ai 12 operatori minimi necessari come descritto in sezione 1.4.2.

Va sottolineato il fatto che l'introduzione dei pattern, cosi organizzato, presenta alcuni problemi, quali: la distribuzione non equa dei turni e il sotto utilizzo di alcuni operatori.

Op.	Giorni											
	1	2	3	4	5	6	7	8	9	10	11	12
1	N	N	N		S	S	S		P2	P2	P2	
2	M	M		N	N	N		S	S	S		P2
3	P		M	M	M		N	N	N		S	S
4		P	P	P		M	M	M		N	N	N
5	M	M	M		P	P	P		M	M	M	
6	P2	P2		M	M	M		P	P	P		M
7	S		P2	P2	P2		M	M	M		P	P
8		S	S	S		P2	P2	P2		M	M	M
9	M	M			M	M			M	M		
10		M	M			M	M			M	M	
11			M	M			M	M			M	M
12	M			M	M			M	M			M

Op.	Giorni											
	13	14	15	16	17	18	19	20	21	22	23	24
1	M	M	M		P	P	P		M	M	M	
2	P2	P2		M	M	M		P	P	P		M
3	S		P2	P2	P2		M	M	M		P	P
4		S	S	S		P2	P2	P2		M	M	M
5	N	N	N		S	S	S		P2	P2	P2	
6	M	M		N	N	N		S	S	S		P2
7	P		M	M	M		N	N	N		S	S
8	R	P	P	P	R	M	M	M	R	N	N	N
9	M	M			M	M			M	M		
10		M	M			M	M			M	M	
11			M	M			M	M			M	M
12	M			M	M			M	M			M

Tab. 1.5: Sequenza dei turni applicata a tutti gli operatori. Lo spazio bianco implica un riposo.

Distribuzione non equa dei turni Di fatto, i turni meno appetibili, in particolare la notte, sono sempre assegnati ai primi otto operatori, mentre agli altri operatori sono assegnati principalmente turni mattutini, essi inoltre godono di piú riposi rispetto ai loro colleghi.

Sotto utilizzo di alcuni operatori Gli operatori che non effettuano i turni secondo il pattern lavorano solo quindici giorni su trenta, molto meno dei ventisei necessari a coprire l'orario base.

Entrambi i problemi sono vincoli di tipo sociale, discussi meglio nella sezione 1.5. Va però sottolineato che la gestione quotidiana bilancia almeno in parte il secondo problema. Infatti i quattro operatori extra corrono il rischio di vedersi cancellati uno o piú riposi per coprire le assenze degli altri operatori, assenze sia ordinarie (ferie e permessi) che straordinarie (malattie).

1.4.4 Numero minimo di riposi

L'organizzazione della casa di riposo prevede che, ogni mese, siano garantiti almeno 5 riposi ad ogni operatore. Questo vincolo è una clausola del contratto singolare, la quale migliora una condizione definita da quello collettivo. Questo infatti prevede un riposo ogni sei giorni lavorati.

1.5 Vincoli di tipo sociale

Organizzando il lavoro di un gruppo di persone, è in genere necessario cercare di tener conto anche delle preferenze dei singoli, oltre che cercare di trattare tutti in modo equo. Definiamo queste esigenze *vincoli sociali* poichè sono generati da eventi esterni all'istituto o hanno una ripercussione sulle attività sociali degli operatori.

È difficile dare un elenco esaustivo degli aspetti di cui sarebbe necessario tener conto. Spesso la sensibilità, o la situazione personale, dei singoli operatori varia il peso delle singole situazioni. Ad esempio per qualcuno può essere un peso effettuare straordinari, mentre per altri è uno spreco il non poterli effettuare; o, ancora, qualcuno potrebbe preferire i turni serali e notturni che invece dispiaciono ai piú. Vi sono però alcuni aspetti generali:

- *Equa distribuzione dei turni:* questo aspetto si presta a varie interpretazioni, da variare il piú possibile le assegnazioni dei turni, ad assegnare a tutti un turno notturno al mese.
- *Equa distribuzione dei carichi:* si richiede che nessun operatore sia costretto ad effettuare delle ore di straordinario mentre altri non raggiungono l'orario minimo.
- *Richiesta di lavorare in determinati orari:* un operatore potrebbe richiedere di lavorare, o meno, in una determinata fascia oraria.

- *Ferie e riposi:* per legge le ferie dovrebbero essere concordate tra il datore di lavoro ed il dipendente. Purtroppo non è sempre possibile accontentare tutte le richieste.

1.6 Il compito della cooperativa FT Service

La Cooperativa FT Service ha il compito di gestire completamente il personale del nucleo verde, per il quale il carico è fisso, e parzialmente quello del nucleo giallo, per il quale la richiesta può variare di giorno in giorno. Ha, inoltre, il compito di coprire le assenze in tutti gli altri nuclei. I turni che FT Service deve effettuare al di fuori del nucleo verde sono, in ogni caso, noti a priori. A tal fine dedica 7 operatori a tempo pieno al reparto verde e 2 a quello giallo. I rimanenti 7 operatori, tra i quali anche qualcuno a tempo parziale, coprono i turni mancanti dei nuclei verde e giallo, nonché le assenze negli altri reparti. Gli operatori assegnati in modo definitivo al nucleo verde aderiscono alla turnazione 3+1.

2. IL MODELLO MATEMATICO

Troppo celebrare per capire che si può star bene
senza complicare il pane
Giudizi Universali - Samuele Bersani

Sommario

In questo capitolo si definisce un modello matematico per il problema di definire i turni del personale della casa di riposo. In particolare si riconduce il problema a quello che, in letteratura, è noto come *Nurse Rostering Problem*. Si definiscono i vincoli e le funzioni obiettivo in modo da poter risolvere il problema sia con un approccio di tipo *Constraint Satisfaction Problem* che di tipo *Multi Goals Optimization Problem*.

2.1 Nurse Rostering Problem

In letteratura il problema di definire i turni di servizio per gli operatori di una casa di riposo, e piú in generale di una struttura ospedaliera, è identificato come *Nurse Rostering Problem* (NRP) o *Nurse Scheduling Problem* (NSP). NRP è un argomento molto interessante nel campo, piú generale, dello *staff scheduling and rostering*. Ne è un esempio l'elevato numero di lavori presentati alle varie edizioni della conferenza internazionale *Practice and Theory of Automated Timetabling* (PATAT) (si vedano gli atti dei convegni [BCPR95], [BC97], [BE00], [BC02] e [BT04]). L'interesse per questo problema consiste nel fatto che, sebbene sia regolamentato in modo differente a seconda della nazione, è alla fine riconducibile a poche famiglie di vincoli che possono presentarsi o meno nei singoli casi. Ad esempio, [CLLR03] individua sedici famiglie di vincoli che caratterizzano il problema. Il caso in questione, ne presenta ben quattordici. Un altra particolarità consiste nell'omogeneità di molti approcci tradizionali, con carta e penna: essi sono pressochè identici, sebbene concepiti per strutture e nazioni differenti. Ad esempio la casa di riposo di Ghedi fa ricorso a sequenze cicliche di turni, tecnica risolutiva presente in molti casi citati in letteratura (si veda ad esempio [BCB01] e [Pla01]). Queste sequenze cicliche sono progettate per coprire tutti i turni richiesti utilizzando al meglio gli operatori disponibili. In caso di carico fisso ci si limita ad assegnare ad ogni operatore un preciso pattern. Nel caso si debbano coprire delle assenze, per ferie o malattia, si possono utilizzare gli operatori ad hoc.

Un altro motivo che rende NRP interessante consiste nel fatto d'essere di difficile soluzione. Tipicamente, infatti, è un problema con vincoli molto stretti, tanto da risultare spesso insoddisfacibile, quando d'altra parte è necessario

determinare una soluzione ad ogni costo. Spesso è necessario definire i carichi di lavoro per un alto numero di operatori in un periodo che va da un paio di giorni ad uno o piú mesi, con un notevole impatto sociale. Il tutto deve avvenire nel minor tempo possibile, visto che bisogna anche poter gestire le emergenze. Ad esempio quando, a causa di un imprevisto, una soluzione prevista diventa irrealizzabile è necessario determinarne una nuova nel giro di pochi minuti per tamponare la situazione. In ogni caso, come evidenziato in letteratura, la soluzione automatica si è sempre dimostrata migliore rispetto a quella manuale sia in termini di tempo che di correttezza.

2.1.1 Rassegna bibliografica

Per risolvere NRP si sono provate varie strade. Tra queste spiccano:

- approcci di tipo modellistico [JSV98] risolti anche mediante solutori di programmazione lineare [MK92];
- l'utilizzo di metodi di programmazione matematica [MWG76];
- l'uso di algoritmi euristici, che ricalcano in parte le procedure manuali ([Hun95] e [JK92]), tra i quali:
  - metodi greedy [SW77];
  - tabu search [GL97];
  - algoritmi genetici [AD01];
- la tendenza a combinare le varie tecniche tra loro [AW04].

In molti lavori si evidenzia come, per riuscire ad ottenere una soluzione, sia necessario rilassare uno o piú vincoli [CB02]; in questi casi riveste un ruolo importante la scelta del vincolo da rilassare e, nelle applicazioni pratiche, spesso si lascia che sia il decisore, o schedulatore, a decidere a quali condizioni rinunciare.

È normale definire NRP come un problema di ottimizzazione [JSV98] oppure come un problema di decisione [aH01], ottenendo quindi un problema di ottimizzazione (spesso multi obiettivo) oppure un problema di soddisfacibilità dei vincoli. Questi due approcci non sono tra loro esclusivi, e possono essere combinati per cercare una soluzione del problema.

Per una bibliografia piú dettagliata si può far riferimento a [EJK+04, §3.7] ed a [CLLR03].

2.1.2 NRP come problema di ottimizzazione

Definendo NRP come un problema di ottimizzazione (OP) si cerca di minimizzare, o massimizzare, una funzione obiettivo, rispettando tutti i vincoli della formulazione. Si tratta, storicamente, del primo approccio al problema, nel quale si cerca di risolvere il modello tramite le tecniche proprie della programmazione matematica, tramite programmazione lineare, lineare intera o multi obiettivi (*Goal Programming*) appoggiandosi a strutture quali grafi o reti.

Sebbene l'approccio matematico porti, in teoria, alla migliore soluzione possibile, in molti casi non risulta applicabile in pratica. Infatti l'alto numero di vincoli, spesso difficilmente descrivibili, la presenza di piú funzioni obiettivo, la necessità di ottenere una soluzione in tempi brevi e la possibilità di avere istanze che non ammettono soluzione riducono il campo d'applicazione della programmazione matematica. Risulta invece piú produttivo utilizzare metodi euristici o meta-euristici.

2.1.3 NRP come problema di decisione

Tipicamente NRP prevede vincoli molto stretti, perché le risorse impiegate sono molto costose, e vengono quindi ridotte al minimo indispensabile. Diventa, quindi, difficile ottenere non solo la soluzione ottima, ma anche una soluzione ammissibile. In questo caso è piú semplice, e sufficiente per scopi pratici, gestire NRP come un problema di soddisfacibilità dei vincoli o *Constraint Satisfaction Problem* (CSP). Ultimamente, questo approccio ha avuto una larga diffusione, grazie anche alla relativamente recente introduzione della programmazione per vincoli o *Constraint Programming* (CP), seguita da una varietà di linguaggi per la programmazione logica per vincoli, i quali rendono molto semplice descrivere anche un'insieme complesso di vincoli.

CP è un paradigma di programmazione specifico per le tecniche di *Constraint Satisfaction*. CP permette di modellare un problema come un insieme di variabili, che ammettono un insieme finito di valori. Queste variabili sono associate tra di loro tramite dei vincoli o requisiti (*constraint*). Ad ogni vincolo è associato un algoritmo di propagazione che ha il compito di eliminare, dal dominio delle variabili, tutti quei valori che non portano ad una soluzione ammissibile. Ogni qual volta un vincolo elimina un valore l'evento viene propagato a tutti gli altri vincoli. Il processo di propagazione termina quando non esistono piú valori che violano un vincolo.

Alla CP sono state affiancate, in ogni caso, anche altre tecniche euristiche o meta-euristiche quali *tabu search*, algoritmi genetici e *simulated annealing*.

2.1.4 Combinare i due punti di vista

I due approcci non sono tra loro incompatibili. Ambedue si basano sul medesimo modello matematico: una soluzione ottima per OP è anche una soluzione ammissibile per CSP e, viceversa, una soluzione ammissibile per CSP è ammissibile per OP e, nel caso che OP permetta la violazione di qualche vincolo e l'obiettivo sia di minimizzarne la violazione, è anche ottima. Nel capitolo 3, si descriverà l'utilizzo di un solutore di Programmazione Lineare Commerciale, sia per risolvere la formulazione OP che quella CSP. Nel capitolo 4 verrà descritto un metodo costruttivo che, genera una soluzione valida per il CSP e un algoritmo di TS che la migliora rispetto a una o piú funzioni obiettivo proprie di una formulazione OP.

2.2 Un modello per la soddisfacibilità dei vincoli

In questa sezione definiamo i vincoli che caratterizzano le soluzioni ammissibili di NRP per la specifica applicazione all casa di riposo di Ghedi, ottenendo cosi una formulazione di tipo CSP. Nella sezione successiva introdurremo alcune funzioni obiettivo e dei rilassamenti sia per poter ricondurre la formulazione da CSP a OP che per poter aumentare il numero di istanze ammissibili.

2.2.1 Rappresentazione della soluzione

Le soluzioni NRP vengono spesso rappresentate da una tabella che riporta, per ogni coppia operatore-giorno, il compito assegnato (vedi tabella 2.1). questa descrizione viene chiamata operatore-giorno o *nurse-day*.

Op.	G1	G2	G3	G4	G5	G6	G7
1	T1	T1	T1	R	T2	T2	T2
2	R	T2	T2	T2	R	T1	T1
3	T2	R	R	T1	T1	R	R

Tab. 2.1: Schema operatore-giorno per un esempio di assegnazione dei turni

Un altra descrizione comune consiste in un insieme di griglie che riportano, giorno per giorno, quale operatore effettua un determinato turno (vedi tabella 2.2). Questa visione viene chiamata operatore-compito o *nurse-task*.

	G1			G7		
Op.	T1	T2	R	T1	T2	R
1	X				X	
2			X	X		
3		X				X

Tab. 2.2: Schema operatore-compito per un esempio di assegnazione dei turni

Esiste un terzo modo per descrivere le soluzioni, che consiste nel rappresentare gli assegnamenti in una tabella turno-giorno. Sebbene risulti essere pratico per il responsabile del servizio, il quale può sapere direttamente che operatore sta eseguendo un determinato servizio, risulta essere poco utilizzato poichè:

1. è di difficile consultazione per gli operatori;
2. non è adatto per i turni che non hanno un numero fisso di richieste, quali ad esempio le ferie.

La descrizione che adottiamo è quella operatore-giorno.

2.2.2 Dati e variabili

Orizzonte temporale È dato un insieme L di giorni (orizzonte temporale) che corrisponde a m mesi consecutivi. Si definisce il sottoinsieme $M_k \subseteq L$ come l'insieme dei giorni che costituiscono il mese $k \in \{1..m\}$. Inoltre l'orizzonte temporale è anche suddiviso in s settimane consecutive: il sottoinsieme $S_t \subseteq L$ è l'insieme dei giorni che appartengono alla settimana $t \in \{1..s\}$. Otteniamo quindi che:

$$L = \bigcup_{k=1}^{m} M_k = \bigcup_{t=1}^{s} S_t$$

Orario contrattuale, ferie e malattie È dato un insieme N di operatori. Per ogni operatore $i \in N$ si definisce $h_i \geq 0$ come il numero medio di ore che egli deve effettuare in una settimana e $H_{ki} \geq 0$ come il massimo numero di ore che deve effettuare durante il mese k.

Le ore in eccesso rispetto al valore H_{ki} sono accumulate, mese per mese, come straordinari. Per ogni operatore è noto il numero di ore di straordinario f_i che ha accumulato dall'inizio dell'anno fino al giorno che precede l'inizio dell'orizzonte temporale. L'operatore può anche richiedere una o piú giornate di ferie che formano l'insieme $L_i^F \subseteq L$. Similmente può ottenere, dal servizio sanitario, una o piú giornate malattia che definiscono l'insieme $L_i^I \subseteq L$. Per taluni operatori il contratto prevede, nel limite del possibile, un riposo ogni tre giorni lavorativi consecutivi. Questo tipo di turnazione ciclica, detta del *3+1*, definisce l'insieme degli operatori $N_3 \subseteq N$.

Turni e carico di lavoro È dato l'insieme T di turni. I turni possono essere classificati come lavorativi ($W \subseteq T$), di riposo ($R \subseteq T$), di ferie ($F \subseteq T$) o di malattia ($I \subseteq T$). I turni lavorativi sono quelli nei quali l'operatore si presenta alla casa di riposo per svolgere un compito che gli è stato assegnato. I turni di riposo, malattia e ferie sono turni nei quali l'operatore non si presenta alla casa di riposo e quindi non può svolgere nessun compito. Le quattro categorie dei turni sono tra di loro disgiunte e sono una partizione dell'insieme T dei turni. Si ha quindi che:

$$T \equiv W \cup F \cup I \cup R$$

$$W \cap F = W \cap I = W \cap R = F \cap I = F \cap R = I \cap R = \emptyset$$

Tutti i turni, a parte quelli di riposo, concorrono al calcolo del monte ore lavorate dall'operatore. In questo caso si parla anche di giorni lavorativi. I turni lavorativi hanno una durata fissa d_j, espressa in ore, mentre per i turni di malattia e riposo la durata v_i dipende dal tipo di contratto d'assunzione dell'operatore. Ad esempio, per un operatore inquadrato con un contratto da 38 ore per sei giorni alla settimana, si ha che $v_i =$ 6h 33'. Tutti i turni lavorativi hanno un orario d'inizio b_j e di fine e_j per i quali valgono le relazioni $0 \leq b_j < e_j \leq 24$ e $e_j - b_j \geq d_j$ La durata può risultare inferiore al periodo che intercorre tra l'inizio e la fine del turno nel caso, piú generale, che sia possibile effettuare una

pausa. Va sottolineato che non tutti i turni possono essere svolti da un operatore. Infatti, può darsi che un operatore non sia abilitato ad eseguire un certo turno, che il turno non sia di sua competenza o si trovi in condizioni tali da non poter effettuare un certo tipo di mansione. In questo caso si definisce l'insieme $W_{il} \subseteq W$ come l'insieme dei turni che l'operatore i può svolgere il giorno l. Infine, si definisce come carico di lavoro $c_{jl} \geq 0$ il numero di operatori che devono svolgere il turno $j \in W$ nel giorno $l \in L$.

Reparti La casa di riposo è suddivisa organizzativamente in un insieme D di reparti. È possibile associare ai singoli reparti i vari turni lavorativi, indicando con $T_r \subseteq W$ l'insieme dei turni che fanno riferimento ad un particolare reparto $r \in D$. Alcuni operatori sono associati ad un reparto: $N_r \subseteq N$ è l'insieme degli operatori che afferiscono al reparto r.

Operatore con turno preferenziale In alcuni casi è possibile che, ad un operatore sia assegnato preferibilmente sempre un determinato turno. Si definisce quindi l'insieme $W^*_{il} \subseteq W_{il}$ come l'insieme dei turni che si preferisce l'operatore i esegua il giorno l.

Operatori Jolly In casi eccezionali è possibile utilizzare degli operatori di riserva, o jolly, al fine di riuscire a completare il carico di lavoro. Si definisce quindi l'insieme $N^J \subseteq N$ degli operatori jolly.

Le decisioni prese nelle schedulazioni precedenti possono avere una ricaduta sulle decisioni che possono essere effettuate durante la schedulazione del periodo L. In particolare sappiamo il numero di giorni consecutivi lavorati $a_i \in \{0..6\}$ prima del primo giorno del periodo L, l'ultimo turno $y_i \in T$ effettuato dall'operatore i e, il numero di ore g_i già lavorate nella prima settimana, se questa non inizia di lunedí.

Frontiera Le decisioni che devono essere prese per l'orizzonte temporale L, sono influenzate dalle assegnazioni dei turni nel periodo immediatamente precedente. In particolare, le decisioni precedenti influiscono solo sui primi giorni del orizzonte temporale. Definiremo come *frontiera* questo periodo iniziale che subisce l'influenza delle decisioni prese nel passato.

Variabili decisionali

Il nostro scopo consiste nel definire per ogni operatore $i \in N$ che turno $j \in T$ svolgerà il giorno $l \in L$. La soluzione si presta ad essere modellata tramite le variabili decisionali $x_{ijl} \in \{0, 1\}$. Con $x_{ijl} = 1$ indichiamo che l'operatore i deve effettuare il turno j nel giorno l, in caso contrario è $x_{ijl} = 0$.

2.2.3 Definizione dei vincoli

È possibile introdurre una classificazione per raggruppare i vincoli, a seconda della loro provenienza. Possiamo individuare:

- vincoli introdotti dalla tipologia del servizio svolto;
- vincoli definiti dalla normativa del lavoro vigente in Italia;
- vincoli introdotti dal Contratto Collettivo Nazionale del Lavoro UNEBA;
- vincoli caratteristici dell'organizzazione del lavoro presso la casa di riposo di Ghedi;
- vincoli di tipo sociale.

Il poter determinare l'origine del vincolo permette di estendere un algoritmo anche a contesti differenti, rispetto a quello per il quale è stato concepito, modificando solamente i vincoli non comuni ai due contesti.

2.2.4 Vincoli di servizio

Questi vincoli sono introdotti dalla natura del servizio offerto dalla casa di riposo. Il primo vincolo definisce la necessità di effettuare un certo numero di turni in una determinata giornata.

$$\sum_{i \in N} x_{ijl} = c_{jl}, \quad j \in W, l \in L \tag{2.1}$$

Questo è un vincolo rigido. Altrettanto rigido è il vincolo che limita il numero di turni che un operatore può svolgere in una giornata.

$$\sum_{j \in T} x_{ijl} = 1, \quad i \in N, l \in L \tag{2.2}$$

All'operatore non devono essere assegnati dei turni che non può svolgere.

$$x_{ijl} = 0, \quad j \notin W_{il}, i \in N, l \in L \tag{2.3}$$

Questo vincolo può talvolta ricadere tra quelli definiti dalla normativa del lavoro italiana. Questi casi sono, però, piú un eccezione piuttosto che la norma[1]. Sono fortemente caratterizzati dal tipo di servizio offerto[2]. A seconda del contesto si può definire questo vincolo come rigido o flessibile. Esso, infatti, racchiude le preferenze, le abilitazioni o gli impedimenti dell'operatore. Una possibile soluzione del problema, come suggerito in [CB02], consiste nel suddividere il presente vincolo in due vincoli, relativi ad abilità e preferenze, definendo le prime come vincolo rigido mentre il secondo come flessibile.

[1] Si pensi alle donne incinta, a quelle in maternità e ai lavoratori minorenni
[2] Ad esempio il concetto d'orario notturno è generalmente estraneo ad un lavoro d'ufficio

2.2.5 Vincoli determinati dalla normativa del lavoro

Tutti i vincoli introdotti dalla normativa che regola il lavoro sono, per definizione, rigidi in quanto definiti da leggi[3]. La normativa italiana prevede che al lavoratore sia garantito almeno un riposo di 24 ore ogni sei giorni lavorati.

$$\sum_{t=0}^{6} \sum_{j \in W \cup I \cup F} x_{ij(l+t)} \leq 6, \quad i \in N, l \in 2 \dots |L| - 6 \tag{2.4}$$

Per tenere conto del periodo precedente l'orizzonte temporale, è necessario modificare il vincolo nei giorni della frontiera, definendo

$$\sum_{t=0}^{6-a_i} \sum_{j \in W \cup I \cup F} x_{ij(t+1)} \leq 6 - a_i, \quad i \in N \tag{2.5}$$

Tra due turni di lavoro successivi, al lavoratore deve essere garantito un riposo di almeno undici ore.

$$x_{ij_1 l-1} + x_{ij_2 l} \leq 1, \quad i \in N, l \in L, j_1, j_2 \in W : 24 - e_{j_1} + b_{j_2} < 11 \tag{2.6}$$

Anche in questo caso il vincolo va modificato per i giorni appartenenti alla frontiera.

$$x_{ij1} = 0, \quad i \in N, j \in W : y_i \in W \wedge 24 - e_{y_i} + b_j < 11 \tag{2.7}$$

Va sottolineato il fatto che, per il vincolo (2.4), le giornate di malattia o ferie sono considerate lavorative, mentre non lo sono ai fini del vincolo (2.6). Bisogna pure garantire che l'operatore si possa godere le ferie concertate con il datore di lavoro, o come ferie o come permessi.

$$\sum_{j \in F \cup R} x_{ijl} = 1 \quad l \in L_i^F, i \in N \tag{2.8}$$

Similmente, l'operatore non deve essere chiamato in servizio durante un congedo per malattia.

$$\begin{cases} \sum_{j \in I \cup R} x_{ijl} = 1 & \text{se } l \in L_i^I \\ \sum_{j \in I} x_{ijl} = 0 & \text{se } l \notin L_i^I \end{cases}, \quad i \in N, l \in L \tag{2.9}$$

I vincoli (2.8) ed (2.9), sebbene siano concettualmente simili gli insiemi L_i^F eL_i^I, differiscono per il fatto che, mentre è possibile assegnare dei giorni di ferie extra al lavoratore, lo stesso non vale per le malattie. Il fatto che si considerino anche i giorni di riposo come ferie o malattie, serve a non violare il vincolo (2.4). Il numero di ore di lavoro in una settimana e in un mese è limitato per ogni lavoratore.

$$\sum_{l \in M_k} \Big(\sum_{j \in W} d_j x_{ijl} + \sum_{j \in I \cup F} v_i x_{ijl} \Big) \leq H_{ki}, \quad i \in N, k \in \{1..m\} \tag{2.10}$$

[3] L'innoservanza di questi vincoli può avere risvolti civili o addirittura penali

$$\sum_{l \in S_t} \Big(\sum_{j \in W} d_j x_{ijl} + \sum_{j \in I \cup F} v_i x_{ijl} \Big) \leq 48, \quad i \in N,\, t \in \{2..s\} \tag{2.11}$$

$$\sum_{l \in S_1} \Big(\sum_{j \in W} d_j x_{ijl} + \sum_{j \in I \cup F} v_i x_{ijl} \Big) + g_i \leq 48, \quad i \in N \tag{2.12}$$

Il vincolo (2.12) è la versione di (2.11) da introdurre se il primo giorno del nostro orizzonte temporale non corrisponde con il primo giorno della settimana. Il vincolo (2.10) è l'unico vincolo flessibile di questo insieme. La sua violazione introduce il concetto di straordinario.

2.2.6 Vincoli introdotti dal CCNL UNEBA

Il contratto UNEBA definisce solo due vincoli rigidi. Il primo indica il numero minimo di riposi che deve godere un operatore nell'arco di un mese.

$$\sum_{l \in M_k} \sum_{j \in R} x_{ijl} \geq 5, \quad k \in \{1..m\},\, i \in N \tag{2.13}$$

Il secondo vincolo, che sarà descritto nella sezione 2.3, limita il numero di straordinari che un operatore può effettuare in un anno.

2.2.7 Vincoli operativi ed organizzativi

I vincoli di questo gruppo rappresentano gli accordi tra gli operatori e la cooperativa. Non sono imposizioni definite da un contratto nazionale, nè derivano da accordi commerciali tra la cooperativa e la casa di riposo. Sono tipicamente vincoli flessibili con una bassa penalità, quindi di norma i primi ad essere violati.

Il primo vincolo (2.14) definisce la turnazione 3+1. Questo vincolo si applica solamente agli operatori dell'insieme N_3.

$$\sum_{t=0}^{3} \sum_{j \notin R} x_{ij(l+t)} \leq 3, \quad i \in N_3,\, l \in 2 \ldots |L| - 3 \tag{2.14}$$

Anche questo vincolo deve essere riformulato nei pressi della frontiera.

$$\sum_{t=0}^{3-a_i} \sum_{j \notin R} x_{ij(t+1)} \leq 3 - a_i, \quad i \in N_3 \tag{2.15}$$

Per gli operatori che appartengono all'insieme N_3, questi vincoli dominano i vincoli (2.4) e (2.5). Il secondo vincolo determina la permanenza di un operatore in un reparto.

$$\sum_{i \in N_r} \sum_{j \in \{T_r \cup F \cup I \cup R\}} x_{ijl} \geq \min\{ \sum_{j \in T_r} c_{jl}, |N_r| \}, \quad r \in D, l \in L \tag{2.16}$$

Un operatore può essere assegnato al di fuori del proprio reparto solamente se il carico richiesto è inferiore al numero di operatori in carico al reparto. In ogni caso il vincolo (2.1) garantisce che tutti i turni di un nucleo siano coperti anche se il numero di operatori disponibili nel nucleo è inferiore al carico complessivo.

Un operatore deve effettuare sempre un turno preferenziale

$$\sum_{j \in R \cup I \cup F \cup W^*_{il}} x_{ijl} = 1, i \in N, l \in L : \sum_{j \in W^*_{il}} c_{jl} > 0 \tag{2.17}$$

Un operatore al quale sia assegnato un turno preferenziale, in un giorno che è richiesto il turno, o sta effettuando un turno, o sta effettuando un altro dei suoi turni preferenziali o non è al lavoro.

L'introduzione degli operatori jolly richiede di modificare il vincolo (2.2) introducendo i vincoli (2.18) e (2.19).

$$\sum_{j \in T} x_{ijl} = 1, \, i \in N/N^J, \, l \in L \tag{2.18}$$

$$\sum_{j \in W} x_{ijl} \leq 1, \, i \in N^J, \, l \in L \tag{2.19}$$

In questo modo il vincolo per gli operatori *normali* non viene modificato e si introduce soltanto la possibilità di utilizzare anche gli operatori jolly per i turni lavorativi.

2.3 Funzioni obiettivo

Le soluzioni ammissibili e non ammissibili si distinguono definendo un insieme di vincoli. I vincoli possono essere suddivisi in due classi: vincoli rigidi o *hard* e vincoli flessibili o *soft* [CLLR03]. I primi sono vincoli che non possono essere mai violati. I secondi sono invece vincoli che possono essere violati a patto di pagare una pesante penalità.

I vincoli definiti nella sezione 2.2.3 costituiscono un modello CSP. Una soluzione del modello CSP è un assegnamento di turni agli operatori che rispetta tutti i vincoli. Non è detto che sia una soluzione soddisfacente. Per poter distinguere se una soluzione è piú o meno soddisfacente tra tutte quelle ammissibili si possono definire una o piú funzioni obiettivo, la cui valutazione permetterà di confrontare le soluzioni stesse. In questo modo, ci riconduciamo ad un modello di tipo OP. Un'altra motivazione per introdurre delle funzioni obiettivo consiste nel fatto che molte istanze non hanno una soluzione ammissibile. Poichè stiamo modellando un caso reale la risposta "*non esistono soluzioni*" non è accettabile. Si rende quindi necessario rilassare alcuni vincoli, in particolar modo quelli flessibili, ed introdurre opportune funzioni obiettivo per scegliere la soluzione che viola il meno possibile l'insieme dei vincoli rilassati.

Le funzioni obiettivo, prese in considerazione nel seguito, sono:

1. minimizzare il costo della soluzione;

2. minimizzare la violazione di un vincolo flessibile;

3. tener presente un vincolo di tipo sociale.

Ci troviamo, quindi, di fronte ad un modello OP multi-obiettivo. Nella sezione 2.3.6 discuteremo in che modo combinare i diversi obiettivi, in modo da trovare una formulazione che li consideri tutti.

2.3.1 Minimizzare il costo

Utilizzo degli operatori jolly

L'utilizzo degli operatori jolly dovrebbe essere un evento raro, tipico di condizioni critiche. Se divenisse la norma, significherebbe che la casa di riposo è sotto organico. L'obiettivo consiste, quindi, nel minimizzare l'utilizzo degli operatori jolly, o piú precisamente il numero di ore da loro effettuato.

$$\min \sum_{i \in N_J} \sum_{l \in L} \sum_{j \in W} d_j x_{ijl} \tag{2.20}$$

2.3.2 Ore straordinarie

Uno dei principali motivi per cui un'istanza non ammette una soluzione ammissibile, consiste nell'impossibilità di coprire tutti i turni richiesti, facendo effettuare all'operatore solamente le ore definite contrattualmente. L'introduzione di ore straordinarie permette, al contrario, di gestire anche gli eventuali carichi aggiuntivi temporanei senza richiedere l'introduzione di altri operatori. Definiamo la quantità $o_{ki} \geq 0$ come il numero di ore straordinarie lavorate dall'operatore i il mese k. Possiamo riscrivere il vincolo (2.10) come:

$$\sum_{l \in M_k} \big(\sum_{j \in W} d_j x_{ijl} + \sum_{j \in I \cup F} v_i x_{ijl} \big) \leq H_{ki} + o_{ki}, \quad i \in N,\, k \in \{1..m\} \tag{2.10'}$$

Lo straordinario è, regolamentato e penalizzato, principalmente per tutelare il dipendente dallo sfruttamento. Sia la normativa che il contratto collettivo UNEBA fissano un tetto massimo al numero di ore straordinari che è possibile effettuare in un anno. Per il contratto UNEBA il limite è pari 120 ore contro le 250 ore definite dalla normativa.

$$f_i + \sum_{k=1}^{m} o_{ki} \leq 120, \quad i \in N \tag{2.21}$$

A questo si aggiunge il fatto che lo straordinario deve essere corrisposto con un compenso maggiorato.

Questo da luogo ad un'ulteriore funzione obiettivo:

$$\min \sum_{i \in N} \sum_{k=1}^{m} o_{ki} \tag{2.22}$$

Ore non lavorate

Un problema opposto a quello delle ore straordinarie si presenta quando un operatore lavora, in un mese, meno del monte ore stabilito contrattualmente. La variabile $u_{ki} \geq 0$ indica quante ore l'operatore i ha svolto in meno rispetto al limite contrattuale per il mese k. Questa definizione modifica il vincolo (2.10′) nel modo seguente:

$$\sum_{l \in M_k} \Big(\sum_{j \in W} d_j x_{ijl} + \sum_{j \in I \cup F} v_i x_{ijl} \Big) + u_{ki} = H_{ki} + o_{ki}, \quad i \in N,\, k \in \{1..m\} \quad (2.10'')$$

Le due quantità o_{ki} e u_{ki} svolgono rispettivamente il ruolo di variabile di surplus e di slack per il vincolo (2.10).

Ne risulta la funzione obiettivo:

$$\min \sum_{i \in N} \sum_{k=1}^{m} u_{ki} \tag{2.23}$$

Ricondurre le ore non lavorate a giornate In condizioni normali, ovvero non in presenza di straordinari, la somma delle ore non lavorate è un valore costante. Non ha quindi senso cercare di minimizzare (2.23). Tuttavia le ore non lavorate possono essere compensate, in generale, come ferie o permessi. La casa di riposo di Ghedi cerca, dove possibile, di compensare le ore non lavorate tramite un giornata completa di ferie. È quindi necessario che questo numero di ore sia per ciascun operatore il piú vicino possibile ad un multiplo della durata v_i di una giornata lavorativa media. La trasformazione delle ore non lavorate in ferie avviene ridistribuendo tali ore tra gli operatori, ma senza introdurre nuove ore di straordinario. Se, ad esempio, un operatore, con $v_i = 7$ deve colmare 13 ore non lavorate è meglio assegnargli un solo giorno di ferie e sprecare 6 ore piuttosto che assegnargli due giorni di ferie e pagargli un ora di straordinario. Possiamo descrivere questa necessità tramite la seguente funzione obiettivo non lineare

$$\min \sum_{i \in N} \Big[\big(\sum_{k=1}^{m} u_{ki} \big) \bmod v_i \Big] \tag{2.24}$$

2.3.3 Violazione della turnazione 3+1

Se si ammette di poter negare il riposo ogni tre giorni lavorativi agli operatori dell'insieme N_3 è necessario sostituire i vincoli (2.14) e (2.15) rispettivamente con

$$\sum_{t=0}^{3} \sum_{j \notin R} x_{ij(l+t)} \leq 3 + p_{il}, \quad i \in N_3,\, l \in 2 \ldots |L| - 3 \tag{2.14′}$$

e con

$$\sum_{t=0}^{3-a_i} \sum_{j \notin R} x_{ij(t+1)} \leq 3 - a_i + p_{i1}, \quad i \in N_3 \tag{2.15′}$$

Dove $p_{il} = 0$ indica che si sta rispettando la turnazione 3+1 mentre $p_{il} = 1$ indica che per l'operatore i non è previsto un riposo nel periodo di quattro giorni che va dal giorno l al giorno $l + 3$, e quindi si viola il vincolo (2.14). Il vincolo (2.14) domina il vincolo (2.4) ma questo non è piú vero se si rilassa il primo. Infatti se $p_{il} = 1$ per quattro giorni consecutivi si viola anche il vincolo (2.4). Una nuova funzione obiettivo consiste, quindi, nel minimizzare il numero di attivazioni della variabile p_{il}:

$$\min \sum_{i \in N_3} \sum_{l=1}^{|L|-3} p_{il} \tag{2.25}$$

Oppure, se si vuole distribuire la violazione in modo equo tra gli operatori:

$$\min \pi \tag{2.26}$$

con il vincolo

$$\sum_{l=1}^{|L|-3} p_{il} \leq \pi, \quad i \in N_3 \tag{2.27}$$

2.3.4 Assegnazione di un operatore al di fuori del reparto

Alcuni operatori sono assegnati in modo permanente ad un reparto. Questi operatori possono operare al di fuori del proprio reparto solamente se non esistono compiti da svolgere nel reparto. Questa condizione è garantita dal vincolo (2.16). Si può scegliere di permettere che un operatore svolga un compito al di fuori del proprio reparto, anche se esistono ancora compiti da svolgere in reparto. Per conseguenza dobbiamo introdurre una nuova variabile $q_{rl} \geq 0$ ad indicare il numero di operatori che, nella giornata l, svolgono un compito al di fuori del proprio reparto r sebbene esistano ancora attività da svolgere in reparto. Va, quindi, modificato il vincolo (2.16) nel modo seguente

$$\sum_{i \in N_r} \sum_{j \in \{T_r \cup F \cup I \cup R\}} x_{ijl} + q_{rl} \geq \min\{\sum_{j \in T_r} c_{jl}, |N_r|\}, \quad r \in D, l \in L \tag{2.16'}$$

introducendo, come funzione obiettivo,

$$\min \sum_{l \in L} \sum_{r \in D} q_{rl} \tag{2.28}$$

che consiste nel minimizzare, giorno per giorno, il numero di operatori che svolgono un turno al di fuori del proprio reparto.

Turni preferiti

Per i turni preferiti valgono le medesime considerazioni fatte per i reparti di competenza. La variabile $\hat{q_{il}} \in \{0, 1\}$ indica che l'operatore i non ha effettuato

il suo turno preferito il giorno l ($\hat{q_{il}} = 1$); in caso contrario vale $\hat{q_{il}} = 0$. Il vincolo (2.17) può essere riscritto come

$$\sum_{j \in \{R \cup I \cup F \cup W^*_{il}\}} x_{ijl} = 1 - \hat{q_{il}}, i \in N, l \in L : \sum_{j \in W^*_{il}} c_{jl} > 0 \qquad (2.17')$$

Introducendo la funzione obiettivo

$$\min \sum_{l \in L} \sum_{i \in N} \hat{q_{il}} \qquad (2.29)$$

Le quantità q_{il} e $\hat{q_{il}}$ indicano il numero di operatori che non stanno effettuando un turno di loro competenza.

2.3.5 Introduzione dei vincoli sociali

Tra i principali vincoli sociali ricordiamo:

Periodicità dei turni Una successione di turni caotica può non essere gradita agli operatori. L'adozione di uno schema ciclico di turni permette di mitigare questo problema. Purtroppo, il carico di lavoro non costante nel tempo non può garantire che lo schema sia sempre rispettato.

Successione di turni simili Riuscire a garantire ad un operatore di svolgere una sequenza di turni simili, come orario, può rendere piú sopportabile il lavoro.

Equa distribuzione dei carichi lavorativi Un altro motivo di scontento tra i lavoratori si manifesta quando, ad alcuni operatori, vengono richieste prestazioni straordinarie, mentre ad altri non si assegnano compiti sufficienti a coprire il numero di ore lavorative del mese. Questo vicolo, relativamente agli straordinari, è difficile da descrivere per il fatto che ne esistono due interpretazioni, da parte degli operatori. Mentre, da una parte, c'è chi non vuole effettuare straordinari, c'è anche chi si lamenta se l'azienda non ne permette l'effettuazione[4].

Equa distribuzione dei turni notturni

Alcuni turni si svolgono in orario notturno. Per un operatore può non essere piacevole l'essere continuamente assegnato ad un turno notturno, mentre altri operatori non ne effettuano nemmeno uno. Si tratta quindi di cercare di bilanciare la distribuzione dei turni tra gli operatori. Questo concetto può essere esteso a tutti i turni. Ricordando che contratto UNEBA definisce l'orario notturno come l'intervallo che va dalle 22 alle 6 del mattino possiamo definire la seguente funzione obiettivo:

$$\min \nu \qquad (2.30)$$

[4] Si ricordi che gli straordinari sono pagati con una maggiorazione rispetto alle ore normali

con

$$\sum_{l \in L} \sum_{j \in W_N} x_{ijl} \leq \nu, \quad i \in N \tag{2.31}$$

dove $\sum_{jl} x_{ijl}$ è il numero di turni notturni che l'operatore i effettua e ν è il massimo di tali valori.

Periodicità dei turni

Per garantire una certa periodicità dei turni, è necessario definire un pattern di turni. Va sottolineato il fatto che esistono insiemi di turni che hanno il medesimo orario e variano solamente a causa del compito o del reparto in cui sono svolti. Possiamo, quindi, definire un pattern con le funzioni

$$\omega : N \times L \to (0, 24) \tag{2.32a}$$

$$\Omega : N \times L \to (0, 24) \tag{2.32b}$$

che associano, ad ogni coppia operatore giorno, rispettivamente l'ora d'inizio e di fine del turno, rispetto al pattern prestabilito. In questo modo è possibile definire la distanza di un turno da un pattern come

$$\Psi(i, j, l) = |\omega(i, l) - b_j| + |\Omega(i, l) - e_j|, \quad j \in W \tag{2.33}$$

La funzione obiettivo sarà, quindi

$$\min \sum_{i \in N} \sum_{j \in W} \sum_{l \in L} x_{ijl} \Psi(i, j, l) \tag{2.34}$$

che consiste nel cercare di minimizzare la distanza dal pattern proposto

Successione di turni simili

Si vuole garantire che gli operatori dell'insieme N_3 eseguano tre turni identici in successione. A tal fine, va introdotta la variabile $w_{il} \in \mathbb{B}$ definita come

$$w_{il} = \begin{cases} 0 & \text{se } \exists j \in W \cup F \cup I : x_{ijl} = x_{ij(l+1)} = x_{ij(l+2)} = 1 \\ 1 & \text{altrimenti} \end{cases} \tag{2.35}$$

indicante, quindi, che l'operatore i effettua tre lavorativi turni identici nei tre giorni consecutivi l, $l + 1$ e $l + 2$. Questa variabile è definita solamente per i giorni che seguono immediatamente un giorno di riposo nel pattern ideale. Cioè il turno predefinito per l'operatore i nel giorno $l - 1$ deve essere un giorno di riposo. Sia L_i^R questo sottoinsieme di giorni. L'obiettivo diventa minimizzare il numero di variabili w_{il} attive:

$$\min \sum_{i \in N_3} \sum_{l \in L_i^R} w_{il} \tag{2.36}$$

Funzione	Definizione	Variabile	Unità Misura
Uso Jolly	(2.20)	$d_j x_{ijl}$	Ore
Ore straordinarie	(2.22)	o_{ik}	Ore
Ore non lavorate	(2.23)	u_{ik}	Ore
Violazioni 3+1	(2.25)	p_{il}	N° Violazioni
Violazioni del reparto	(2.28)	q_{il}	N° Violazioni
Violazione dei turno preferenziale	(2.29)	$\hat{q_{il}}$	Violazioni
Massimo numero di turni notturni	(2.30)	ν	N° Turni
Violazioni pattern	(2.34)	$x_{ijl}\Phi(i,j,l)$	Ore
Sequenze simili	(2.36)	w_{il}	N° Violazioni
Carico massimo	(2.38)	ρ	Numero puro

Tab. 2.3: Unità di misura delle funzioni obiettivo

Equa distribuzione dei carichi lavorativi

L'operatore i nel mese k, lavora $H_{ik}+o_{ik}-u_{ik}$ ore, mentre l'accordo contrattuale ne prevede H_{ik}. Il suo carico relativo è definito come il rapporto tra queste due quantità:

$$\frac{H_{ik}+o_{ik}-u_{ik}}{H_{ik}} = 1 + \frac{o_{ik}-u_{ik}}{H_{ik}} \tag{2.37}$$

Tralasciando il valore costante 1, possiamo scrivere una funzione obiettivo del tipo:

$$\min \rho \tag{2.38}$$

con

$$\frac{o_{ik}-u_{ik}}{H_{ik}} \leq \rho \tag{2.39}$$

che cerca di livellare il carico lavorativo relativo tra gli operatori. L'intento è cercare una soluzione nella quale tutti fanno ore straordinarie o tutti lavorano meno del previsto.

2.3.6 Riunire le varie funzioni obiettivo

Il solutore lineare opera con una sola funzione obiettivo. Anche gli algoritmi euristici tendono ad ottimizzare una sola funzione obiettivo, anche se possono tenerne presenti varie. Questi comportamenti determinano la necessità di raggruppare in qualche modo tutte le funzioni obiettivo descritte sinora.

In tabella 2.3.6 sono riportate le varie funzioni obiettivo indicando per ognuna la sua unità di misura. Si nota immediatamente che esistono due principali gruppi omogenei, quali ad esempio le funzioni che trattano ore e definiscono il costo naturale della soluzione ((2.20), (2.22) e (2.23)) e le funzioni che hanno a che fare con le violazioni dei vincoli flessibili ((2.25), (2.28), (2.29), (2.30)

e (2.36)). La funzione obiettivo (2.34), sebbene sia valorizzata in ore, rappresenta la violazione di un vincolo flessibile e quindi non appartiene al primo gruppo, ma necessita di un opportuno fattore di trasformazione per essere aggiunto al secondo gruppo. L'ultima funzione rimanente ((2.38)) tratta un numero puro che rappresenta un rapporto tra ore e può essere considerato come una violazione di un vincolo flessibile.

In realtà, però, anche funzioni con la stessa unità di misura non si possono semplicemente sommare: ad esempio 1 ora di straordinario costa molto meno di un ora effettuata da un operatore jolly.

Introducendo opportuni parametri di confronto possiamo riunire le funzioni obiettivo in un unica funzione:

$$\begin{aligned}\min & \sum_{i\in N}\sum_{k=1}^{m}(\alpha_o o_{ki}+\alpha_u u_{ki})+\sum_{i\in N_j}\sum_{l\in L}\sum_{j\in W}\alpha_d d_j x_{ijl}+\\ &+\sum_{l\in L}\Big(\alpha_\Psi\sum_{i\in N}\sum_{j\in W}\kappa x_{ijl}\Psi(i,j,l)+\alpha_q\big(\sum_{r\in D}q_{rl}+\sum_{i\in N}\hat{q_{il}}\big)\Big)+\\ &+\sum_{i\in N_3}\Big(\alpha_w\sum_{l\in L_i^R}w_{il}+\alpha_p\sum_{l=1}^{|L|-3}p_{il}\Big)+\alpha_\nu\nu+\alpha_\rho|\rho|\end{aligned} \tag{2.40}$$

Dove α_o, α_u, α_d, α_p, α_Ψ, α_q, α_ν, α_w e α_ρ sono i parametri di confronto, mentre κ è il fattore di trasformazione necessario per ricondurre (2.34) ad una violazione di vincolo.

Nasce il problema di dare un peso corretto ai vari parametri di confronto e di definire in modo adeguato κ.

Definire i pesi delle funzioni obiettivo

La definizione precisa dei pesi richiede una o piú l'interviste agli esperti del problema, tipicamente i responsabili del servizio. In queste interviste si effettua una simulazione del comportamento della funzione obiettivo al variare dei pesi. All'esperto sono sottoposte varie combinazione di pesi e le soluzioni relative. L'esperto esprime un indice di gradimento per ogni coppia pesi-soluzione. Tramite opportune operazioni su matrici, da questi dati è possibile definire i valori dei pesi che massimizzano la soddisfazione dell'esperto.

Di seguito riportiamo una prima stima per la combinazione dei pesi. Piú che altro si definiscono alcuni principi da considerare durante la definizione di questi. È in programma la fase di correzione della stima tramite colloqui con il responsabile di FT Service.

Come ricordato in precedenza:

- è necessario definire un fattore di trasformazione per poter sommare valori di natura differente;
- a parità di unità di misura, non tutte variabili hanno lo stesso peso o significato.

  - un ora di operatore jolly costa piú di un ora di straordinari;
  - un ora di straordinari costa piú sia di un ora non lavorata che di una violazione di un vincolo flessibile;
  - sotto certe condizioni le ore non lavorate potrebbero maggiorare sempre un valore costante;
  - essendo ore a costo zero le ore non lavorate dovrebbero non introdurre violazioni ad un vincolo flessibile.

- le violazioni flessibili hanno pressochè il medesimo costo

Possiamo quindi definire che in generale deve valere:

1. $\alpha_d > \alpha_o > \alpha_u$
2. $\alpha_o > \alpha_p \approx \alpha_q \approx \alpha_\Psi$

Riferendoci al caso particolare della casa di riposo di Ghedi dove vale che:

- $|L| \leq 31$
- $|N_r| = 8$
- $\Psi(i,j,l) \mapsto [0,34]$ e $< E, \Psi(i,j,l) >= 13.44$
- il numero di turni notturni, in un nucleo, è pari a $1/8$ del totale dei turni da effettuare

proviamo a definire qualche piccola regola per calibrare i nostri pesi:

- $\alpha_o > \max_{i \in N}\{v_i\}\alpha_u$, per evitare che la riduzione delle ore non lavorate introduca straordinari;
- la violazione di un vincolo flessibile al giorno per operatore vale in media 1;
- $\alpha_o \gg 1 \gg \alpha_u$;
- per ricondurre la violazione del pattern ad 1, si ha che $\kappa < E, \Psi(i,j,l) > \approx 1$ da cui $\kappa \approx 1/13.44 = 0.74$
- se si vuole equiparare (2.30), (2.36) e (2.38) a violazioni di vincoli giornalieri è necessario che:
  - $\alpha_\nu * 8 \approx 1$ per ricondursi al massimo ad una violazione al giorno;
  - $\alpha_w/4 \approx 1$ equivalente al massimo ad una violazione per ogni periodo di quattro giorni;
  - $\alpha_\rho/31 \approx 1$ per ricondursi al massimo ad una violazione al giorno.

In tabella2.4, sono suggeriti i valori di prima approssimazione per i pesi della funzione obiettivo (2.40) .

Parametro	**Aspetto**	**Valore**
α_d	utilizzo di un jolly	100
α_o	costo dello straordinario	10
α_u	penalità per le ore non lavorate	0,1
$\alpha_p, \alpha_q, \alpha_\Psi$	violazione di un vincolo	1
κ	fattore di trasformazione per la violazione del pattern	0,74
α_ν	notti non bilanciate	0,13
α_w	sequenze identiche non rispettate	4
α_ρ	carico mal distribuito	31

Tab. 2.4: Elenco dei pesi assegnati alla funzione obiettivo (2.40)

3. UTILIZZO DI UN SOLUTORE DI PROGRAMMAZIONE LINEARE INTERA

> "Il computer che avete davanti" disse "è il piú grande computer di questo tipo che esista al mondo. Contiene cinque milioni e trecentomila criotroni e può tener conto simultaneamente di centomila variabili."
>
> *Lenny* - Isaac Asimov

Sommario

In questo capitolo si descrive l'utilizzo di un solutore di Programmazione Lineare Intera open source, GLPK, per la risoluzione del problema tramite un approccio prettamente modellistico. Un solutore lineare può essere utilizzato sia per determinare la schedulazione ottimale, sia per verificare l'ammissibilità di un'istanza. Purtroppo, l'utilizzo di un solutore lineare è limitato, dalla complessità del problema e dal tempo a disposizione, ad istanze di piccola dimensione. Si introdurranno, quindi, alcune tecniche per ridurre la complessità del problema, rafforzando la formulazione, o per costruire una soluzione intera, arrotondando la soluzione ottima del rilassamento lineare del modello. Per meglio definire i limiti d'applicabilità e descrivere il comportamento del solutore, si riportano i risultati delle prove effettuate su un insieme di test generato a partire da un'istanza reale. In ultimo si descrive la libreria Java CROS-GLPK creata per gestire i risultati generati da GLPK.

3.1 Strumenti general purpose per la risoluzione di problemi di Programmazione Lineare Intera

Una volta che un problema è stato modellato come un problema di Programmazione Lineare Intera (PLI), per determinarne una soluzione, si può implementare un algoritmo risolutivo ex-novo oppure fare affidamento a strumenti preesistenti. Questi strumenti, definiti *general purpose*, perché non sono legati ad un unico dominio applicativo o problema, permettono teoricamente di risolvere qualsiasi problema per il quale sia stato costruito un modello matematico di PLI. In pratica, però, la soluzione può richiedere un tempo che, sebbene finito, è assolutamente non praticabile.

Tra gli strumenti a disposizione in questo genere di approccio si ricordano:

- i solutori

- i generatori algebrici di modelli
- i linguaggi di modellazione

Il solutore è lo strumento base per affrontare il problema permettendo di determinare la soluzione. Spesso il solutore richiede di formulare il problema in una modalità molto complicata. In questo caso l'utilizzo di un generatore algebrico di modelli, insieme ad un linguaggio di programmazione ad alto livello detto di modellazione, permette di formulare in modo semplice un problema che possa essere interpretato e risolto dal solutore.

3.1.1 I solutori di Programmazione Lineare Intera

Un solutore di Programmazione Lineare, o solutore lineare, è un applicativo che riceve in ingresso un modello matematico ed i dati che caratterizzano un istanza del problema e restituisce una soluzione ottima. Tipicamente un solutore lineare determina la soluzione con il metodo del simplesso e le sue varianti. Un solutore lineare può, in genere, anche risolvere problemi di Programmazione Lineare Intera, servendosi di algoritmi di *branch & bound* predefiniti, eventualmente con l'aggiunta di piani di taglio, e di euristiche d'arrotondamento per generare i *bound* richiesti dal metodo.

Molto spesso il solutore lineare rende disponibile un insieme di librerie in linguaggi di programmazione ad alto livello, che permettono di richiamare le funzioni del solutore da programmi scritti ad hoc. É quindi possibile utilizzare metodi di risoluzione piú raffinati quali, ad esempio, le tecniche di *branch & bound*, i piani di taglio e i metodi di *branch & price*. In circolazione si trovano vari pacchetti software come, per citare uno tra i piú potenti e diffusi, CPLEX della ILog [ILO].

Un solutore può quindi essere utilizzato in due modi differenti:

1. come libreria esterna che applica il solutore a strutture dati condivise;
2. come libreria esterna che applica il solutore a strutture dati su file;
3. come programma stand alone che processa il modello ed i dati definiti in modo opportuno.

Esistono diversi formati attraverso i quali descrivere il modello ed i dati. Nel caso piú semplice questi formati prevedono una formulazione estesa del modello, cioè richiedono di definire uno per uno i vincoli e le variabili che lo compongono. Questo è poco pratico per istanze di grande dimensione. Tuttavia molti risolutori consentono anche di impiegare un linguaggio di modellazione per la descrizione del problema e dei dati. Successivamente un generatore algebrico di modelli ha il compito di traduttore il modello, descritto mediante il linguaggio di modellazione, in un interpretabile dal solutore.

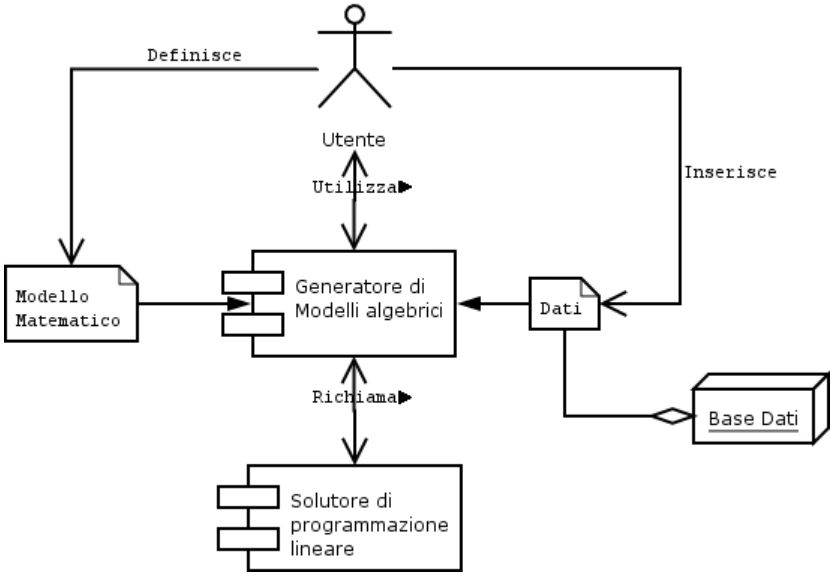

Fig. 3.1: Utilizzo di un solutore di programmazione lineare e di un linguaggio di modellazione

3.1.2 I linguaggi di modellazione

I linguaggi di modellazione sono veri e propri linguaggi ad alto livello. Sono quindi molto semplici da utilizzare, avvicinandosi molto al formalismo matematico, se non al linguaggio naturale.

L'utilizzo di una grammatica libera dal contesto, la definizione di variabili ed alias e la possibilità di aggiungere commenti rendono i modelli descritti tramite un linguaggio di modellazione facilmente interpretabili da parte di persone non esperte di programmazione, purchè esperte di modellazione.

In figura 3.1 è riportato lo schema tipico dell'utilizzo di un solutore attraverso un linguaggio di modellazione. L'utente deve definire, tramite il linguaggio, il modello ed i dati, i quali possono anche risiedere in una base dati. Mediante un generatore algebrico di modelli l'utente applica il solutore direttamente alla descrizione ad alto livello. Il vantaggio fondamentale dell'utilizzo di un linguaggio di modellazione è che il passaggio ad un'altra istanza dello stesso problema, richiede solamente la modifica dei dati, mentre il modello non cambia. D'altra parte, una diversa formulazione dello stesso problema richiede, in genere, la modifica di una o piú famiglie di vincoli, spesso senza cambiare i dati. Il generatore di modelli algebrici ed il solutore rimangono, in ogni caso, invariati. Normalmente un generatore di modelli algebrici può interfacciarsi con tutti principali solutori presenti in commercio, è quindi possibile sostituire il solutore mantenendo immutati il modello e il generatore.

Ad esempio il solutore `glpsol` utilizza il linguaggio di modellazione GNU

Math Prog [GNU04a] e un generatore algebrico interno. Il modello ed i dati passati a `glpsol` sono, di fatto, contenuti in due semplici file di testo. Il primo descrive il modello matematico, definendo dati, variabili, funzioni obiettivo e vincoli. Il secondo riporta i valori numerici dei dati che compaiono nel primo.

Queste caratteristiche di un linguaggio di programmazione matematica permettono di apportare piccole modifiche al modello, per poi verificare in modo semplice e per aggiustamenti successivi, in un ottica *what if*, come si modifica la soluzione, o meglio il processo che porta ad ottenere la soluzione.

3.2 GLPK: GNU Linear Programming Kit

Per la nostra applicazione abbiamo scelto di utilizzare il pacchetto GLPK, acronimo che sta a significare GNU Linear Programming Kit. La scelta di GLPK è dettata principalmente da un fattore economico: il fatto di essere open source riduce a zero il costo della licenza d'utilizzo. L'acquisto di una licenza di CPLEX, ad esempio, è improponibile in realtà medio piccole, dato che il costo si aggira intorno ai sei-settemila euro per la licenza d'uso[1]. Ovviamente, il solutore presente in GLPK, ha prestazioni nettamente inferiori rispetto a CPLEX.

Tra l'altro, GLPK ha il vantaggio di permettere di definire il modello tramite il linguaggio di modellazione GNU Math Prog. Permettendo, quindi, di non dover definire il modello in maniera estesa, e di separare i dati dal modello vero e proprio.

GLPK può essere utilizzato, come accennato in precedenza per i solutori, come:

1. come libreria esterna che applica il solutore sia a strutture dati condivise che a strutture dati su file;

2. come programma stand alone che processa il modello ed i dati definiti tramite file, grazie ad un generatore algebrico interno.

Nel primo caso le chiamate alle librerie di GLPK sono molto simili a quelle di CPLEX, permettendo con semplici sostituzioni il passaggio da una all'altra libreria. Nel secondo GLPK può utilizzare gli stessi formati usati da altri solutori commerciali e può tradurre un modello scritto nel linguaggio di modellazione Math Prog nei formati standard estesi MPS e CPLEX LP o in un file di testo[GNU04b].

In quest'ottica possiamo considerare GLPK come lo strumento ideale per la costruzione di un prototipo per poi passare a CPLEX o a un altro solver commerciale, quando le condizioni economiche lo consentiranno.

3.2.1 Il linguaggio di modellazione GNU Math Prog

GNU Math Prog, o solamente Math Prog, è un sottoinsieme del linguaggio di programmazione matematica AMPL[AMP]. Math Prog permette di descrivere

[1] Dato riferito al 2002

il modello in un modo molto intuitivo e naturale, cosa che non si verifica nè offrendo direttamente i dati al solutore mediante le librerie di GLPK utilizzate in un programma C, nè descrivendo in uno dei formati standard il problema. Questo accelera la fase di scelta della formulazione per il problema, dato che consente di provarne molte in breve tempo.

I costituenti del linguaggio Math Prog

Gli elementi base di un modello in Math Prog sono:

Insiemi : rappresentano il dominio dei valori sia dei parametri che delle variabili; possono essere numerici o costituiti da simboli rappresentati da stringhe di testo;

Parametri : sono i dati, eventualmente organizzati in vettori mono o pluridimensionali, i quali possono essere caricati anche da altri file;

Variabili : descrivono la soluzione ed è il compito del solutore fissarne il valore al termine dell'esecuzione;

Vincoli : definiscono il problema, distinguendo tra soluzioni ammissibili e non ammissibili;

Funzioni obiettivo : necessarie a valorizzare le soluzioni generate; il solutore ne ottimizza una sola durante la sua esecuzione, ma al termine valuta il valore di tutte.

A questi mattoni si aggiunge un nutrito insieme di funzioni predefinite, la possibilità di aggiungere regole di validazione dei dati e, come ci si attende da un linguaggio ad alto livello, di inserire commenti in qualsiasi punto del testo. La grammatica è libera dal contesto o *context free*. Ogni istruzione è terminata da un punto e virgola. É quindi possibile spezzare un'istruzione complessa su piú righe, permettendo anche di aggiungere spazi, tabulazioni o righe vuote, al fine di facilitare la lettura del modello stesso. Math Prog permette di definire dei valori di default per i dati, di utilizzare uno speciale formato tabulare per assegnare anche piú di un parametro contemporaneamente e di utilizzare dei caratteri jolly per valorizzare solo delle sezioni di un insieme o di un vettore. In questo modo risulta essere molto semplice l'inserimento, anche manuale, di un alto numero di dati.

3.2.2 Il solutore glpsol

`glpsol` è il solutore lineare presente nel pacchetto GLPK. È un applicazione richiamabile solo da linea di comando, per la quale non esiste un'interfaccia grafica. Il suo utilizzo base richiede l'indicazione di uno o piú file contenenti il modello ed i dati.

Inizialmente `glpsol` risolve il problema come un problema di Programmazione Lineare (PL). Se il modello è effettivamente di PL, il processo termina

con la soluzione. Se invece si tratta di un modello di Programmazione Lineare Intera (PLI), si è di fatto risolto il rilassamento continuo del problema. In questo modo `glpsol` può verificare l'eventuale insoddisfacibilità e, eventualmente, determinare un bound primale del problema[2]. Successivamente `glpsol`, basandosi sulla soluzione rilassata, determina una soluzione ottima del problema di PLI.

Durante il processo `glpsol` invia allo standard output un insieme di informazioni. Tra queste i valori della soluzione LP e della migliore soluzione intera trovata via via. Al termine del processo, `glpsol` riporta il tempo impiegato e la dimensione della memoria utilizzata.

É possibile intervenire, in ogni caso, sul comportamento di `glpsol` tramite alcuni parametri. Tra questi ricordiamo, perché interessanti ai nostri scopi:

- verificare la correttezza sintattica del modello e la completezza dei dati;
- tradurre il modello e i dati in uno dei formati standard;
- limitarsi a risolvere il problema LP;
- utilizzo di un presolver;
- scelta della variabile di branch;
- scelta della tecnica di back-tracking;
- limitazione del tempo d'esecuzione;
- salvare la soluzione.

Limitarsi alla valutazione del rilassamento lineare Sebbene nel caso di NRP il rilassamento lineare non dia la soluzione del problema, può essere utilizzato per ottenere alcune informazioni importanti quali:

- una condizione sufficiente per l'inammissibilità dall'istanza: se il rilassamento è inammissibile anche l'istanza lo è [Wol98, §2.2];
- una soluzione iniziale da utilizzare come base per altre tecniche, che ne ricavano una intera tramite arrotondamento;
- in casi particolari la soluzione ottima del problema se quella del rilassamento è intera.

Il sapere a priori se un problema non è ammissibile ci permette di introdurre rilassamenti al modello, richiedere all'operatore di diminuire l'inammissibilità o di applicare qualche tecnica per ridurne la complessità.

[2] Lower bound per un problema di minimizzazione, upper bound per uno di massimizzazione

Metodo di costruzione e di esplorazione dell'albero di valutazione Normalmente `glpsol` utilizza criteri euristici propri per determinare sia la variabile di branch[3] sia la tecnica di *back tracking*. È però possibile forzare questa scelta con opportuni parametri. In particolare se lo scopo consiste nel cercare una soluzione ammissibile conviene esplorare l'albero di decisione in profondità (*depth first search*), in modo da trovare il prima possibile una soluzione ammissibile, anche se non particolarmente buona. Per quanto riguarda la variabile di branch è possibile forzare `glpsol` perché utilizzi sempre la prima o l'ultima variabile disponibile. Nessuna tecnica garantisce, anche se il problema è ammissibile, di trovare una soluzione intera nel tempo prefissato. Questo caso si presenta per istanze molto complesse e grandi, sia in termine di numero di vincoli che in termine di numero di variabili. Comunque alcune strategie possono essere piú efficaci di altre nel determinare una soluzione intera.

Limitazione del tempo di valutazione In assenza di indicazioni, `glpsol` prosegue a risolvere il modello finche non trova la soluzione ottima o ne dimostra l'inammissibilità. Per un modello complesso questo può portare ad esecuzioni che durano ore, se non giorni. L'esempio riportato in figura 3.2, che si riferisce ad l'esecuzione su di un istanza di piccole dimensioni[4] della casa di riposo di Ghedi, riporta il tempo necessario per trovare una migliore soluzione ammissibile. In figura è riportata anche la regione, definita dai lower bound e upper bound trovati da CPLEX dopo una lunga esecuzione, nella quale risiede la soluzione ottima. Questi tempi lunghi contrastano con l'esigenza di tempi di calcolo ridotti e di una forte interazione con l'operatore. Può quindi essere necessario limitare il tempo macchina dedicato alla soluzione del modello. Al termine del tempo prefissato possiamo quindi trovarci di fronte a tre casi:

- `glpsol` trova la soluzione ottima, o dimostra che non esiste soluzione ammissibile, eventualmente terminando prima del limite fissato;
- `glpsol` trova, nel tempo limite fissato, una soluzione ammissibile e una stima della ottimalità di questa: in figura 3.2 sono anche riportati i valori dei lower bound stimati da GLPK e CPLEX;
- `glpsol` non trova nessuna soluzione nel tempo limite fissato, il che non vuol dire che l'istanza sia inammissibile per il problema MIP.

Nel primo caso, a meno che l'operatore non voglia apportare delle modifiche, la procedura ha termine. Nei rimanenti casi si può modificare l'istanza, rilassando qualche vincolo o guidando `glpsol` tramite il fissaggio di qualche variabile. questo può portare a soluzioni non ottime, ma semplifica il problema permettendo di trovare una soluzione in termini piú rapidi. Un'altra possibilità consiste nel concedere piú tempo al solutore. In tutti i casi è richiesta una nuova ricerca della soluzione.

[3] GLPK 4.4.1 dichiara di utilizzare l'euristica di Driebeck e Tomlin per determinare la prossima variabile di branch.

[4] In particolare è l'istanza `I-9-9` con 9 operatori e 9 turni.

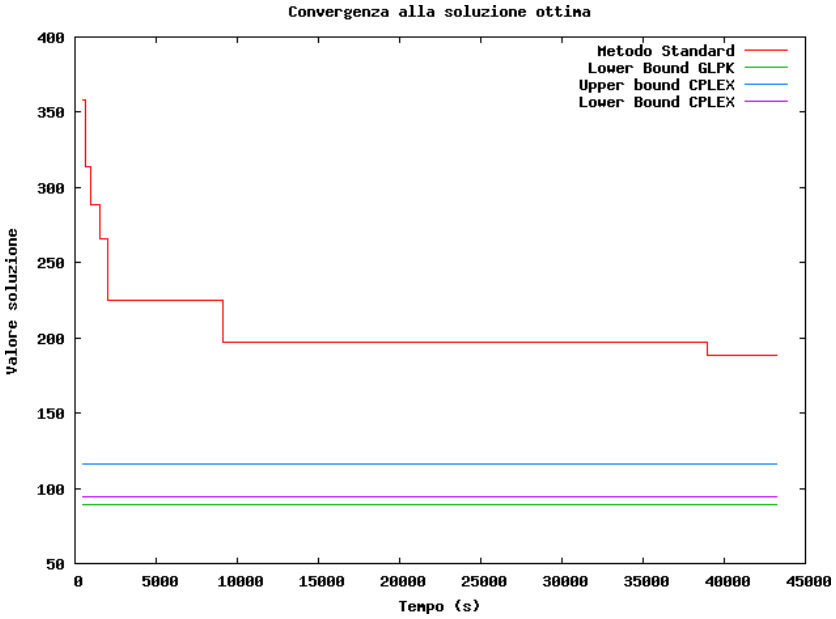

Fig. 3.2: Esempio dei tempi d'esecuzione di GLPK con un istanza ridotta della casa di riposo di Ghedi.

3.3 Risultati sperimentali del modello in Math Prog

Abbiamo definito tre modelli differenti, implementati tramite il linguaggio Math Prog:

Modello CSP ricerca una soluzione qualsiasi che soddisfi rigorosamente tutti i vincoli;

Modello OP ricerca una soluzione che può violare alcuni vincoli, ma ottimizza una funzione obiettivo che tiene conto da un lato del costo e dall'altro delle violazioni compiute;

Modello Pattern aggiunge al modello OP l'esistenza di sequenze cicliche di turni, e ricerca una soluzione che, oltre a tener conto del costo e delle violazioni ai vincoli rilassati si avvicina il piú possibile alle sequenze date.

I tre modelli condividono i medesimi dati. Questo rende molto semplice il passaggio da uno all'altro ed il loro confronto.

3.3.1 Complessità delle istanze e prestazioni

Il solutore, relativamente a NRP, deve scontrarsi con la complessità delle istanze. Questo problema è aggravato dalle scarse prestazioni di `glpsol` rispetto ad altri

solutori. Per verificare i limiti del solutore, sono stati fatti alcuni test su un insieme di istanze campione.

L'insieme di test

L'insieme di test è stato generato a partire da un'istanza reale fornita dalla casa di riposo di Ghedi, comprendente i turni e gli operatori del mese di marzo 2004. In questa istanza sono presenti 16 operatori e 21 differenti tipi di turni, ed il periodo in esame è di 31 giorni. Le istanze di test sono state ottenute diminuendo progressivamente, il numero di operatori disponibili e il numero di tipologie di turni differenti. Ci si è fermati a 9 operatori e 9 turni, dato che questi costituiscono la parte costante delle richieste che la casa di riposo pone a FT Service, particolare sono i turni e gli operatori del nucleo verde. Nei turni sono sempre inclusi i riposi, le ferie e le malattie. Complessivamente questo dà luogo a 104 istanze. Per praticità indicheremo ogni istanza riportando il numero di operatori e di turni che la caratterizzano, in questo modo `I-10-9` identifica l'istanza con 10 operatori e 9 turni, di cui 6 lavorativi piú i riposi, le ferie e le malattie. Definiamo istanza di riferimento, l'istanza `I-16-21` che rappresenta quella originale, con tutti gli operatori e tutti i turni della casa di riposo.

Al fine di poter classificare meglio le varie istanze, è stato definito il rapporto tra la disponibilità media degli operatori ed il carico medio richiesto nel periodo. La disponibilità degli operatori è ridotta dai riposi, dalle ferie e dalle malattie. Il carico è caratterizzato da una parte fissa, i 6 turni lavorativi di base, e da una parte variabile, cioè non presente tutti i giorni. quanto è piú alto il rapporto tra queste due variabili tanto piú semplice è determinare una soluzione, anche in modo manuale. Viceversa tanto piú il rapporto è basso tanto piú è difficile, se non impossibile, trovare una soluzione. Un rapporto di 1 costituisce il confine tra i due comportamenti. In figura 3.3 è riportato il rapporto tra queste due quantità nelle 104 istanze generate. Come si vede, la non diagonalità del grafico evidenzia il fatto che i turni aggiunti non sono sempre presenti, per cui l'aggiunta di un operatore semplifica il problema piú di quanto l'aggiunta di un turno lo complichi. Le istanze con meno di 11 operatori appaiono di difficile soluzione.

La complessità del problema

Il primo ostacolo alla soluzione consiste nella dimensione del modello. Per l'istanza di riferimento `I-16-21`, che è la piú grande del test, che è caratterizzata da 21 turni, 16 operatori e un orizzonte di 31 giorni, vengono generati da `glpsol` quasi 40.000 vincoli, piú di 10.000 variabili superando i 190.000 coefficienti non nulli (occorrenza delle variabili nei vincoli). Queste cifre possono dare una stima dei tempi necessari per trovare una possibile soluzione intera da parte di `glpsol`. Va anche considerato che, spesso, le istanze proprie della casa di riposo non sono ammissibili.

Il nostro scopo consiste nel confrontare i tre modelli. Si cerca di determinare, a seconda del modello, quali istanze sono inammissibili, quali sono facili e quali

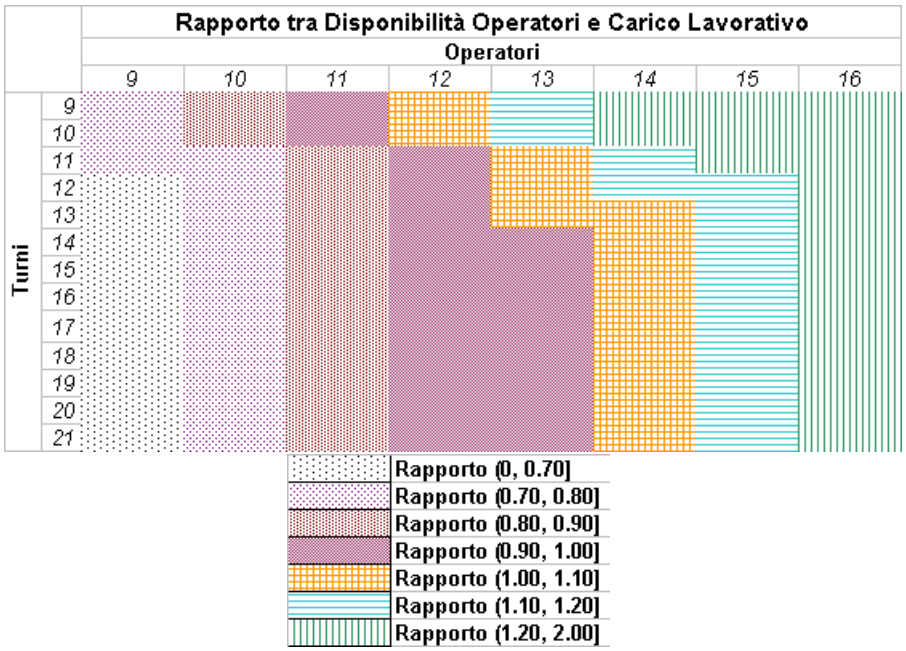

Fig. 3.3: Rapporto tra gli operatori disponibili e il carico di servizio richiesto.

difficili; e di individuare la motivazione del risultato in base alle caratteristiche dell'istanza e del modello.

Il confronto è stato effettuato imponendo un limite di tempo di 1200 secondi, pari a 20 minuti, utilizzano GLPK 4.4.1 per Win32 eseguito su di un AMD Athlon XP 1800.

In linea teorica, ci si aspetta che il solutore determini soluzioni che vadano da istanza inammissibile a soluzione ottima intera, passando da soluzione ottima del rilassamento e soluzione ammissibile intera, proporzionalmente al rapporto raffigurato in figura 3.3.

Nelle figure 3.5, 3.6 e 3.7 sono rappresentate le mappe delle soluzioni trovate da `glpsol` nel tempo limite. Al termine del tempo concesso `glpsol` può trovarsi in cinque stati differenti:

1. Problema riconosciuto inammissibile tramite semplici tecniche di presolving. `glpsol` utilizza delle tecniche, quali particolari assegnamenti iniziali o riduzioni, per determinare l'inammissibilità dell'istanza prima ancora di iniziare a risolvere il rilassamento continuo;

2. Problema riconosciuto inammissibile dal solutore perché il rilassamento continuo non è ammissibile;

3. Problema per il quale è stato risolto il rilassamento continuo, determi-

Fig. 3.4: Leggenda per le mappe 3.5, 3.6, 3.7

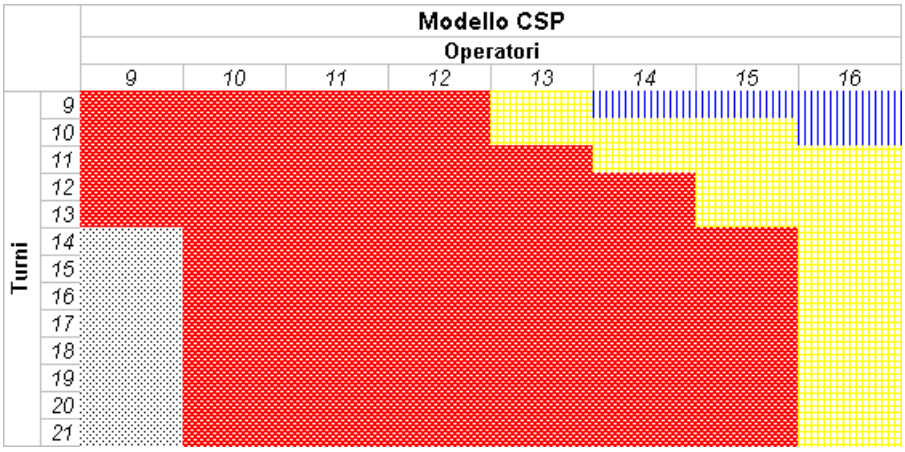

Fig. 3.5: Soluzioni del modello CSP dopo 20'

nando un lower bound, ma per il quale non è ancora stata trovata una soluzione intera;

4. Problema per il quale è stata trovata una soluzione intera;

5. Problema per il quale è stata trovata la soluzione ottima.

Questi cinque stati sono rappresentati nelle figure 3.5, 3.6 e 3.7 mediante la legenda riportata in figura 3.4.

Va sottolineato che le uniche soluzioni direttamente utilizzabili sono quelle intere, a cui corrisponde un'effettivo assegnamento ammissibile dei turni per gli operatori. Ciò non significa che le altre soluzioni siano inutili: i primi due casi caratterizzano un'istanza inammissibile, il terzo caso fornisce un lower bound[5] della funzione obiettivo per la nostra istanza.

Modello CSP Il modello CSP (figura 3.5) ha la particolarità di non ricadere mai nel quarto caso. Infatti, non essendo definita alcuna funzione obiettivo, qualsiasi soluzione intera risulta essere immediatamente ottima. Il modello determina ben 81 istanze sicuramente inammissibili e, delle rimanenti 23, solamente per 4 riesce a determinare una soluzione ottima intera, mentre le altre 19

[5] Salvo per il caso del modello CSP la cui funzione obiettivo è rigorosamente pari a 0 perché il modello non ne prevede una.

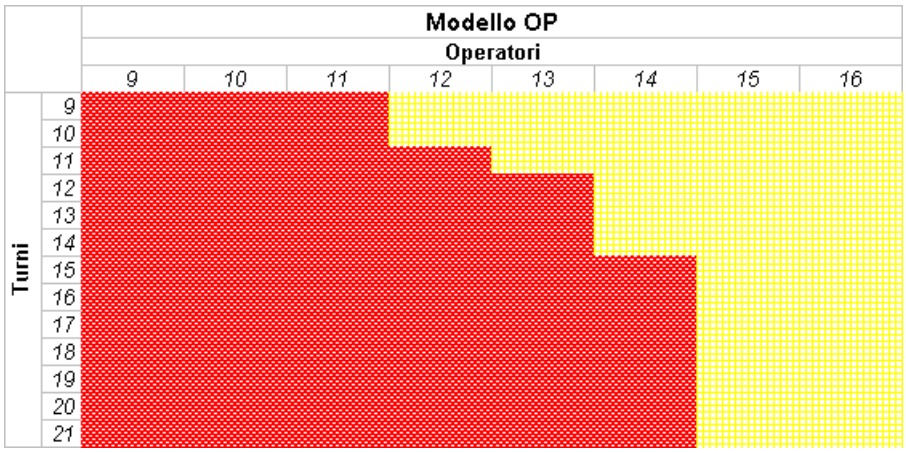

Fig. 3.6: Soluzioni del modello OP dopo 20'

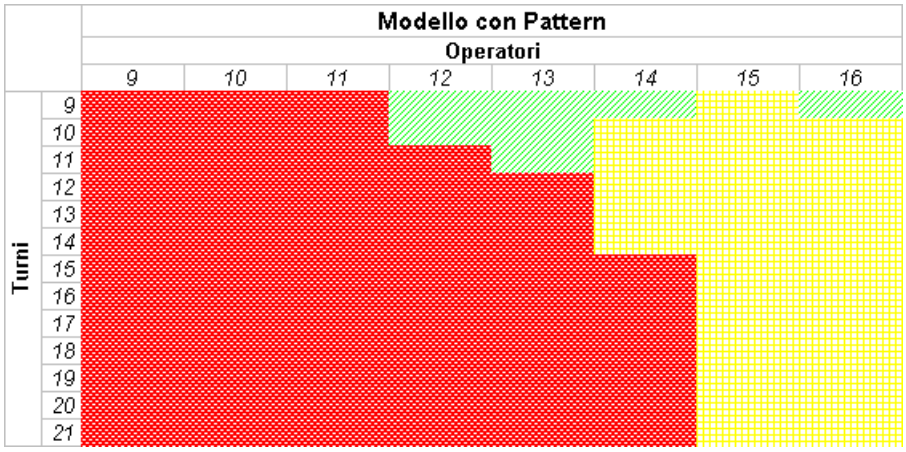

Fig. 3.7: Soluzioni del modello Pattern dopo 20'

Soluzione	CSP	OP	Pattern
Problema inammissibile	8	0	0
Rilassamento continuo non ammissibile	73	67	67
Rilassamento continuo risolto	19	37	30
Soluzione Ammissibile Intera	0	0	7
Soluzione Ottima Intera	4	0	0

Tab. 3.1: Soluzioni trovate con un tempo limite di 20 minuti

rimangono aperte. Per l'istanza di riferimento in particolare non è stato trovata una soluzione intera. Le istanze per le quali si è trovata una soluzione sono distanti da essa.

Modello OP Poichè il modello OP è un rilassamento del modello CSP il numero di istanze inammissibili cala, passando da 81 a sole 67 istanze (tabella 3.1). Inoltre, sempre rispetto a CSP, non esistono piú istanze per le quali l'inammissibilità è determinata facilmente in fase di presolving. Purtroppo per nessuna istanza il modello OP ha potuto trovare una soluzione intera. Questo dovuto al maggior numero di variabili, necessarie a gestire le violazioni dei vincoli. Inoltre la presenza della funzione obiettivo modifica il comportamento delle procedure euristiche che determinano come generare e attraversare l'albero di valutazione. Questo approccio, come sottolineato anche da [CLLR03], risulta quindi essere poco efficace per un prodotto software commercializzabile.

Modello Pattern Questo modello ha le stesse zone di ammissibilità e inammissibilità del modello OP. Tuttavia, la presenza di pattern, che suggeriscono l'andamento di una soluzione ideale, permette di determinare ben 7 soluzioni ammissibili intere. Purtroppo, come per il modello CSP, queste istanze risolte sono le piú piccole, lontane da quella di riferimento.

Introduzione degli operatori Jolly In tutti e tre i modelli non si fa uso degli operatori Jolly. Questi operatori avventizi non sono presenti nell'organigramma base della casa di riposo e rappresentano un costo molto alto. La tendenza organizzativa della cooperativa è quella di eliminare completamente il loro utilizzo. In ogni caso l'introduzione di un operatore jolly può ridurre l'inammissibilità di alcune istanze o permettere dei determinare una soluzione ammissibile intera nel tempo limite prefissato. Ad esempio l'introduzione di 2 operatori jolly e la richiesta di minimizzarne l'utilizzo, mediante un adeguata funzione obiettivo, permette di migliorare il numero di soluzioni ammissibili, come descritto in tabella 3.2. L'introduzione degli operatori jolly ha piú che triplicato il numero di istanze per le quali è stata trovata una soluzione intera ed ha diminuito di piú un terzo il numero di istanze rifiutate, portandole dalle 81 originali alle 52 attuali. Va notato che, sebbene si sia introdotta una funzione obiettivo che minimizza il numero di turni assegnati, una soluzione intera non ottima rispetta tutti i vincoli imposti da CSP e di conseguenza è ottima per il modello CSP originale.

Soluzione	Jolly	CSP	Differenza
Problema inammissibile	0	8	-100%
Rilassamento continuo non ammissibile	52	73	-29%
Rilassamento continuo risolto	38	19	+100%
Soluzione Ottima Intera	14	4	+250%

Tab. 3.2: Soluzioni trovate per il modello CSP introducendo fino a 2 jolly. Tempo limite di 20 minuti

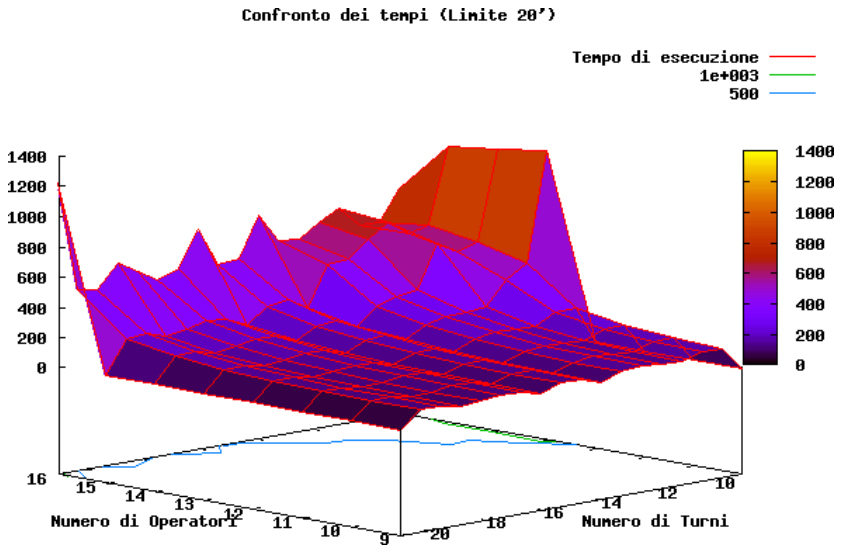

Fig. 3.8: Modello CSP. Tempo richiesto per determinare le soluzioni. Tempo limite 20'

I tempi di calcolo

I test sono stati effettuati limitando il tempo di esecuzione a 20 minuti. Se `glpsol` trova una soluzione ottima intera o se l'istanza è inammissibile il tempo impiegato risulta inferiore. Le figure 3.8 e 3.9 riportano i grafici dei tempi d'esecuzione delle varie istanze per il modello CSP e per il modello Pattern. I grafici per il modello OP ed per il modello Pattern sono pressochè identici. Infatti, benchè pattern trovi soluzioni intere, non trova la soluzione ottima intera, per cui non termina entro 20 minuti. Per semplicità riportiamo solamente il grafico relativo al modello Pattern. In particolare in ambedue i grafici, si delineano due livelli. Il primo, nell'ordine delle decine di secondi, identifica le istanze inammissibili; mentre il secondo, tendenzialmente vicino al tempo massimo, identifica le istanze per le quali esiste almeno la soluzione LP. Nel caso del modello CSP il tempo necessario per trovare la soluzione ottima è molto prossimo al tempo limite.

La tabella 3.3 riporta i tempi medi d'esecuzione di `glpsol`, per risolvere

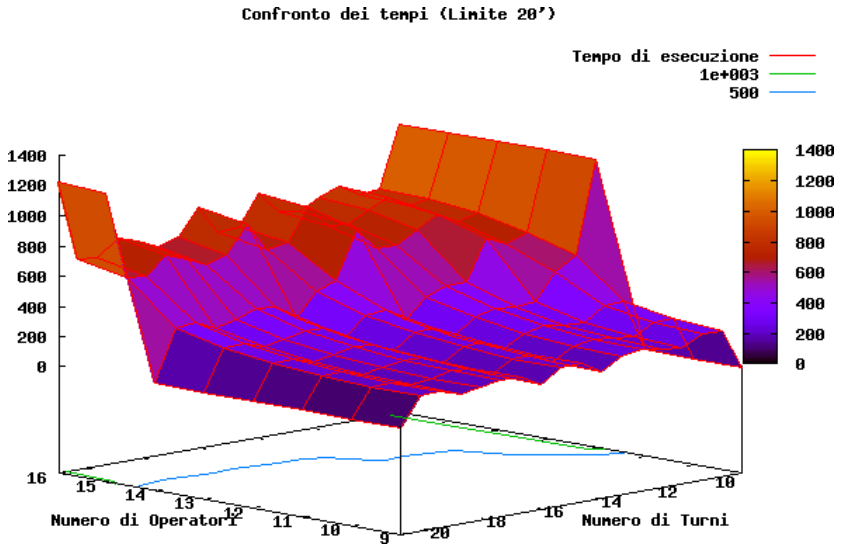

Fig. 3.9: Modello Pattern. Tempo richiesto per determinare le soluzioni. Tempo limite 20'

le 104 a seconda dello stato finale della soluzione. Oltre ai tempi finali sono riportati anche i tempi medi necessari per risolvere il rilassamento continuo del modello.

Soluzione	**CSP**	**OP**	**Pattern**
Problema inammissibile	0,4	-	-
Rilassamento continuo non ammissibile	7,4	9,5	9,0
Rilassamento continuo risolto	6,5	21,6	25,4
Soluzione Ammissibile intera non ancora trovata	1200,0	1200,0	1200,0
Soluzione Ammissibile Intera trovata	-	-	1200,0
Soluzione Ottima Intera trovata	985,9	-	-
Inammissibile Generale	6,7	9,5	9,0
Soluzione Intera	985,9	-	1200,0

Tab. 3.3: Tempi medi d'esecuzione, in secondi, di `glpsol` a seconda dell'esito

Dai grafici e, in modo ancor piú evidente, dai dati della tabella 3.3, risulta chiaro che, sebbene i tre modelli falliscano nel determinare una soluzione intera, possono essere utilizzati per rivelare le istanze non ammissibili. Il modello CSP, infatti, impiega in media meno di 7 secondi per identificare una soluzione sicuramente non ammissibile. Questo tempo risulta essere inferiore a 10 secondi sia per il modello OP che per il modello Pattern. Purtroppo l'alto numero di istanze aperte non ci permette di definire se questi tempi sono validi per identificare in

generale qualsiasi istanza non ammissibile, ma per la quale, ad esempio, il rilassamento continuo è ammissibile. Il tempo medio necessario al modello CSP per risolvere il rilassamento continuo è inferiore a 7 secondi. Il modello OP impiega in media poco piú di 21 secondi, mentre il modello Pattern ne impiega meno di 26. D'ora in poi tutti i confronti sono riferiti al modello Pattern, poichè questo è il modello che riesce a determinare il maggior numero di soluzioni intere.

Rapporto tra la qualità della soluzione e il tempo necessario a determinarla
Il responsabile del personale della cooperativa FT Service impiega quattro ore ogni mese per determinare tutti i turni, e circa un'ora al giorno per gestire gli imprevisti. Il limite di 20 minuti può quindi sembrare riduttivo, ma permette al responsabile di generare piú soluzioni e, scegliere quella che, a suo giudizio, è la migliore. Infatti il modello non include tutte le sfumature della realtà, e il responsabile può essere a conoscenza di vincoli e preferenze secondarie, ma non trascurabili. Può, quindi, succedere che l'operatore rifiuti una prima soluzione, modifichi alcuni elementi della richiesta[6] e ripeta la procedura da capo. Questo procedimento può essere ripetuto piú volte, fino a quando l'operatore non ritiene la soluzione soddisfacente. In questo caso sono richieste diverse valutazioni, tutte di ugual durata. Un tempo limite di 20 minuti permette di ottenere circa due soluzioni ogni ora[7]. In ogni caso il limite di tempo può essere fissato dall'operatore a suo piacimento: un tempo breve consente piú valutazioni ma, probabilmente, porta a soluzioni peggiori e aumenta il rischio di non trovare una soluzione intera.

Diventa quindi interessante indagare quanto migliori, all'aumentare del tempo limite, la soluzione trovata da `glpsol`. In tabella 3.4 sono riportati i comportamenti del solutore al variare del tempo. La prova è stata effettuata limitando il tempo a 600 secondi (10 minuti), 1000 secondi, (16 minuti e 40 secondi), 1200 secondi (20 minuti), 1400 secondi (23 minuti e 10 secondi) e 1800 secondi (30 minuti). Il test è stato effettuato solamente per le 37 istanze in cui il rilassamento lineare è ammissibile. Dei vari risultati si riportano:

1. l'incremento medio, inteso come il miglioramento delle soluzioni trovate rispetto al passo precedente (nel caso di una nuova soluzione l'incremento vale 1);

2. il numero di istanze per le quali è stata trovata una soluzione intera e la loro percentuale rispetto al totale;

3. il numero di istanze per le quali è avvenuto un miglioramento della soluzione precedente;

Per le istanze testate appare conveniente investire del tempo nella ricerca di una soluzione migliore. Questo vantaggio va soppesato rispetto a quello di

[6] Potrebbe, ad esempio, fissare alcune variabili rispetto alla soluzione proposta e sottrarle alle decisioni del solutore.

[7] Va infatti calcolato anche il tempo necessario a preparare i dati necessari alle valutazioni e quello richiesto per valutare la soluzione.

Limite Temporale (s)	600	1000	1200	1400	1800
Incremento Medio	1,00	0,78	0,70	0,46	0,69
Istanze risolte	1	4	7	9	16
	3%	11%	19%	24%	43%
Istanze migliorate	-	1	2	5	4

Tab. 3.4: Rapporto tra la qualità della soluzione e il tempo a disposizione.

effettuare piú valutazioni successive. Solo l'operatore acquisirà con l'esperienza la sensibilità necessaria a determinare il limite piú efficace.

Fissaggio del metodo di scelta della variabile di branch `Glpsol` utilizza una suo criterio euristico per la prossima variabile di branch. É facoltà dell'utente imporre al solutore di fissare la prima, o l'ultima, variabile di branch piuttosto che determinarla tramite l'euristica predefinita in `glpsol`. Poichè noi, in questa fase, ci accontentiamo di una soluzione ammissibile, può darsi che il metodo naive di fissare sempre la prima variabile di branch, senza la necessità di valutare una procedura euristica, aumenti il numero di istanze risolte, anche se a scapito della bontà della soluzione. Grazie ai vincoli (2.1) e (2.2) ad ogni fissaggio di $x_{ijl} = 1$ di ogni variabile equivale a imporre circa $|T-1| \times |N|$ fissaggi del tipo $x_{ijl} = 1$. In questo modo dopo circa $|N| \times |L|$ fissaggi abbiamo una prima soluzione che può essere valutata. Se la soluzione è ammissibile allora abbiamo raggiunto il primo obiettivo di trovare una soluzione ammissibile; d'ora in poi il compito del solutore è solamente quello di determinare una soluzione migliore. Se invece la soluzione non è ammissibile, o è peggiore di una già nota, lasciamo che sia `glpsol` a determinare il miglio metodo di back tracking per correggere la soluzione. Di fatto si tratta di effettuare una visita dell'albero delle soluzioni in profondità. In linea di principio il trovare fin da subito una soluzione ammissibile, potrebbe definire un upper bound migliore sull'ottimo e, quindi, convergere ad esso in tempi piú rapidi. Per contro, il continuare a visitare l'albero delle soluzioni in profondità può rallentare la convergenza verso la soluzione ottima se, ad esempio, questa risiede in rami lontani dell'albero stesso.

Sperimentalmente la strategia del fissaggio della prima variabile sembra essere un buon metodo per migliorare l'efficacia del solutore. Il numero di soluzioni intere trovate nel limite di 20' triplica, passando da 7 a 24, cioè il 65% del numero delle istanze con rilassamento limeare ammissibile, come evidenziato dalla figura 3.10. Si risolvono istanze molto vicine a quella di riferimento, cioè con un solo turno o operatore in meno. La strategia di fissare la prima variabile si è dimostrata piú efficace, dato che migliora in quasi tutti i casi la soluzione trovata con la strategia standard, come descritto in tabella 3.5. Questa condizione permane anche se il tempo a disposizione della strategia standard viene aumentato. In 20 minuti, con l'attraversamento in profondità, si riescono a trovare piú soluzioni che col metodo normale in 30 minuti. Generalmente, quando tutte e due i metodi trovano una soluzione, quella generata tramite il fissaggio della

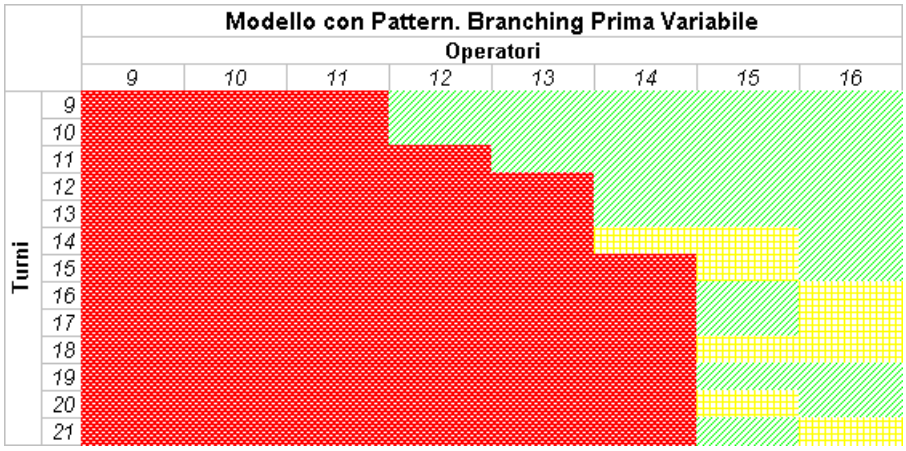

Fig. 3.10: Soluzioni del modello Pattern dopo 20 minuti, forzando `glpsol` ad effettuare il branch sulla prima variabile disponibile.

Gap istanza		Fissaggio/LB	standard/LB	Delta
Miglior risultato standard	`I-12-10`	+98%	+113%	-15%
Miglior risultato depth first	`I-13-11`	+63%	+372%	-209%
Peggior risultato standard	`I-13-9`	+112%	+665%	-553%
Peggior risultato depth first	`I-16-9`	+258%	+198%	+60%
Risultato medio		+145%	+372%	+227%

Tab. 3.5: Rapporto tra le due differenti soluzioni ed il Lower Bound del rilassamento. Test a parità di tempo.

Gap istanza		depth first/LB	standard/LB	Delta
Miglior risultato standard	`I-12-10`	+98%	+113%	-15%
Miglior risultato depth first	`I-13-11`	+63%	+261%	-198%
Peggior risultato standard	`I-14-12`	+78%	+981%	-903%
Peggior risultato depth first	`I-15-13`	+336%	+399%	-63%
Risultato medio		+191%	+383%	-192%

Tab. 3.6: Rapporto tra le due differenti soluzioni ed il Lower Bound del rilassamento. Tempo a disposizione: First 20 minuti, Standard 30 minuti.

prima variabile risulta essere migliore (tabella 3.6). Il peggior risultato ottenuto dal metodo standard è una soluzione intera che costa quasi 11 volte il valore del lower bound. Per la medesima istanza e con minor tempo, il fissaggio della prima variabile fornisce una soluzione che non supera il doppio del lower bound. Questo suggerisce che, per il problema in questione, l'euristica di scelta della variabile di branch utilizzata da `glpsol` risulta essere poco efficiente. Se proviamo ad indagare quale sia la convergenza di `glpsol` al valore ottimo, effettuando il branch sulla prima variabile, si nota che il solutore impiega molto tempo per migliorare una soluzione trovata. Il comportamento risulta chiaro dalla figura 3.11. La figura presenta il confronto tra la soluzione ottenuta con il metodo standard e quella ottenuta effettuando sempre il branch sulla prima variabile. Il metodo che fissa il branch sulla prima variabile è migliore del metodo standard in ogni fase. Entrambi i metodi sono distanti dalla regione, definita dai bound trovati da CPLEX, nella quale risiede il valore della soluzione ottima. Risulta essere conveniente cercare un metodo, o una formulazione del modello, che velocizzi la convergenza. In questo modo non è necessario aumentare il tempo limite o la potenza del calcolatore a disposizione. Situazioni perseguibili in un ambiente lavorativo medio quale è una casa di riposo per due motivi:

1. il tempo a disposizione è limitato e, normalmente, un utente preferisce procedure immediate rispetto a tempi lunghi;

2. il tipico elaboratore da ufficio non presenta eccezionali potenze di calcolo.

.

3.4 Tecniche di riduzione euristiche

I test precedenti mostrano che il solutore riesce a trovare soluzioni, per istanze di dimensioni medie, solo dopo l'introduzione dei pattern. Questa consiste nel

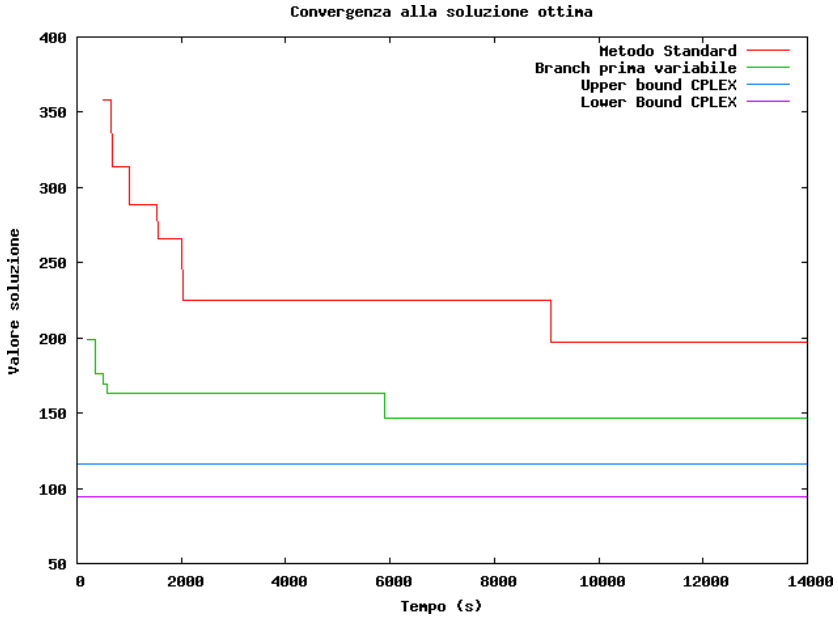

Fig. 3.11: Convergenza alla soluzione ottima intera per l'istanza `I-12-9`

suggerire un fissaggio iniziale delle variabili, in modo da garantire in automatico il rispetto della maggior parte dei vincoli e un costo basso della soluzione. Il pattern non viene rigorosamente imposto, ma penalizzare la sua violazione indirizza `glpsol` verso soluzioni che gli somiglino, aiutando quindi il suo lavoro. Descriviamo ora alcune tecniche per arrivare a questo anche negli altri casi anche rafforzando il modello o riducendo le dimensioni del problema.

3.4.1 Disuguaglianze valide

Possiamo individuare due tagli validi per NRP. Il primo consiste nell'individuare una terna di turni successivi che, a causa della distanza tra di loro, risultano essere incompatibili. Il secondo taglio ricerca un turno che è incompatibile, a causa della distanza, con insieme di turni suoi successori.

Terna di turni incompatibili

Dal vincolo (2.6) sappiamo che due turni successivi sono incompatibili se la fine del primo è a meno di 11 ore dall'inizio del secondo. Siano $f, h \in W$ due turni effettuati rispettivamente nei giorni $l-1, l+1 \in L$. Possiamo cercare un turno $g \in W$ che effettuato nel giorno $l \in L$ sia incompatibile, perché troppo vicino, sia con f che con h. Condizione perché questo avvenga è che $24 - e_f + 24 + b_h < 22 + d_g$. Questa condizione è necessaria ma non sufficiente poichè se il turno g non è lavorativo i turni f e h sono validi.

Possiamo quindi estendere l'incompatibilità di g con f e h ad un insieme di turni $G \cup W$. E piú precisamente

$$x_{ifl-1} + \sum_{g} x_{igl} + x_{ihl+1} \leq 1 + \sum_{g \in F \cup I \cup R} x_{igl}, \quad i \in N, l-1,\, l,\, l+1 \in L$$
$$\text{se vale che} \begin{cases} f, g, h \in W \\ 24 - e_f + b_g < 11 \\ 24 - e_g + b_h < 11 \end{cases} \tag{3.1}$$

Turno incompatibile con un insieme di turni successori

$$x_{ifl-1} + \sum_{g:24-e_f+b_g<11} x_{igl} \leq 1, \quad i \in N,\, f, g \in W,\, l-1, l \in L \tag{3.2}$$

3.4.2 Pre- e postprocessing dei dati

Il solutore trova soluzioni intere per istanze con pochi turni. Si può quindi favorire la ricerca di una soluzione riducendo il numero di variabili tramite una fase di preprocessing.

Una prima manipolazione banale consiste nel rimuovere i turni che non sono richiesti durante l'orizzonte temporale. In questo modo si riduce il numero di variabili e di vincoli presenti nel modello. Ad esempio dall'istanza di riferimento, e per quelle con piú di 19 turni, è possibile eliminare due turni, passando cosi da 21 a 19 turni.

Accorpamento dei turni

Un procedimento piú efficace consiste nel accorpare i turni con uguale orario. Come evidenziato in tabella 1.3 i turni si ripetono, identici per orario, nei vari nuclei. Possiamo quindi ridurre il dominio delle variabili raggruppando i vari turni con il medesimo orario (come suggerito da [CLLR03] e [HW96]). Due turni con lo stesso orario vengono raggruppati in un unico turno per il quale, giorno per giorno, la richiesta è pari alla somma delle richieste dei due turni originali. Di fatto si tratta di un rilassamento del vincolo flessibile che lega alcuni operatori ad un nucleo. La soluzione ottenuta deve quindi essere modificata, in fase di postprocessing, per distinguere nuovamente i turni originali e ripristinare il vincolo violato ammettendo che ciò sia possibile. Se ciò non è possibile si viola il vincolo di permanenza di un operatore nel reparto. Nel caso del modello CSP, ciò significa che non si è trovata una soluzione, mentre nel caso dei modelli OP e Pattern, la soluzione costa effettivamente piú di quanto calcolato, dato che bisogna aggiungere il costo della violazione stessa. Il processo è descritto in figura 3.12.

Questa riduzione permette di diminuire il numero di reparti presenti sino ad uno solo, oppure due: il primo raggruppa tutti quei nuclei ai quali è assegnato almeno un operatore. Nel secondo ricadono i rimanenti nuclei. Introducendo

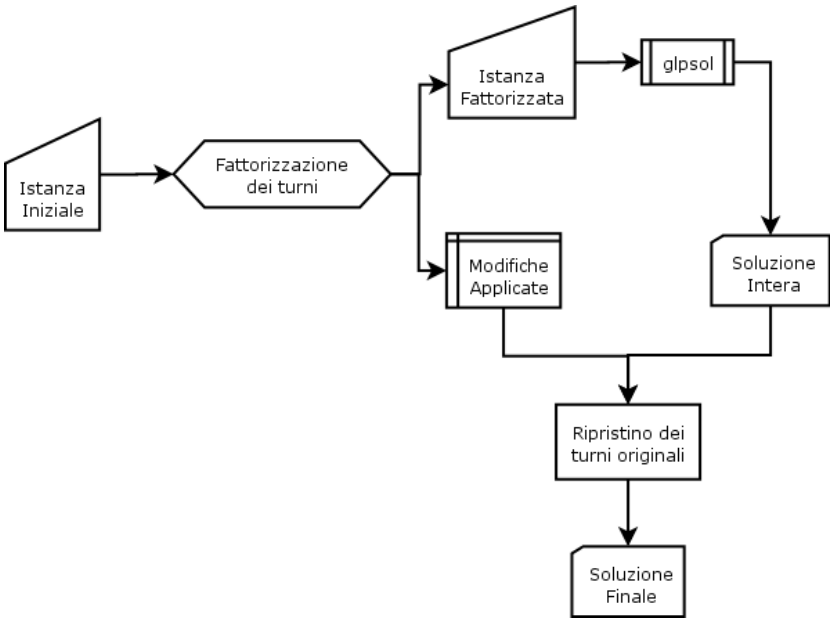

Fig. 3.12: Riduzione dei turni. Pre- e postprocessing

due super nuclei, anzichè uno solo, si cerca di arginare la violazione del vincolo di permanenza nel nucleo.

3.5 Conclusioni in merito all'utilizzo del solutore

3.5.1 Soluzione delle istanze

Dagli esperimenti descritti risulta chiaro che il solutore è difficilmente utilizzabile all'interno di un'applicazione software, per la sua difficoltà di determinare una soluzione intera e, nel caso se ne trovi una, il lungo tempo necessario. É anche vero che, prestazioni permettendo, l'utilizzo del solutore potrebbe risolvere uno spettro di problemi maggiore, perché piccole modifiche al modello consentirebbero di adattare l'applicazione a realtà differenti dalla sola casa di riposo di Ghedi. Di fatto ci si ricondurrebbe solo a dover specificare degli schemi di vincoli per ogni realtà, mantenendo una unica base dati e un unico flusso applicativo.

3.5.2 Informazioni fornite dal solutore lineare

Sebbene il solutore non sia utilizzabile per determinare una soluzione intera può essere utilizzato per evidenziare due aspetti importanti dell'istanza. Infatti esso permette di determinare, in tempi accettabili (come indicato in sezione 3.3.1):

- una parte delle istanze non ammissibili;

- la soluzione del rilassamento lineare, che è un lower bound sul costo della soluzione intera.

Nel primo caso possiamo notificare all'operatore la condizione chiedendogli di modificare l'istanza o di rilassare i vincoli. Nel secondo caso possiamo avvisare l'operatore di quale sia la condizione migliore che, in teoria, può raggiungere. Entrambe le informazioni vanno considerate in senso negativo. La prima cioè non garantisce che le istanze che superano il test siano effettivamente ammissibili. La seconda non garantisce che il costo ottimo sia pari al lower bound, ma solo che non è inferiore ad esso. Ad esempio se la soluzione LP non utilizza straordinari non è detto che una soluzione intera non ne richieda. D'altro canto se la soluzione LP richiede straordinari, questi saranno presenti, probabilmente in numero superiore, anche nella soluzione intera.

3.5.3 Utilizzo congiunto con altre tecniche

Un altro interessante utilizzo del solutore lineare consiste nel generare una soluzione del rilassamento lineare ed arrotondare il valore di ogni variabile al'intero piú vicino. In questo modo otteniamo una soluzione intera, tipicamente non ammissibile, del nostro problema. Possiamo cercare quindi di rendere ammissibile prima, e di migliorare poi, tale soluzione tramite una tecnica di ricerca locale. Non è detto che l'arrotondamento intero della soluzione reale sia vicina alla soluzione ottima intera, ma in ogni caso è una base di partenza tanto valida quanto una soluzione euristica.

Il tempo per determinare questa soluzione reale è, per l'istanza di riferimento, inferiore ai 30 secondi.

3.5.4 La libreria CROS-GLPK

Per poter utilizzare, come accennato poc'anzi, la soluzione generata dal solutore è necessario definire dei metodi per interpretare gli output di `glpsol`. A tal scopo è stata definita una libreria Java, **CROS-GLPK**, come interfaccia con `glpsol`.

Al momento la libreria si limita, principalmente, a realizzare un parser di soluzioni. Questo parser permette di:

1. conoscere lo stato della soluzione;
2. conoscere il valore della soluzione rilassata o intera, se esiste;
3. accedere ai valori delle variabili, o colonne, del problema;
4. accedere ai valori dei vincoli, o righe, del problema;

Nelle future versioni della libreria è possibile prevedere l'inserimento delle seguenti funzionalità:

- interpretazione di un modello scritto in Math Prog per GLPK;

- generazione del file di dati in Math Prog partendo da uno schema template;
- fissaggio dei bound del modello basandosi sui dati di una precedente valutazione dell'istanza.

4. ALGORITMI EURISTICI

Lunga è una notte
Lunga è la seconda,
come posso aspettare per tre?
Spesso un mese
mi sembrò piú breve
che questa mezza notte d'attesa.
Edda - Snorri Sturluson

Sommario

In questo capitolo vengono presentati diversi algoritmi euristici per risolvere il problema. Il primo è un algoritmo costruttivo di tipo greedy, cioè non rivede mai le scelte fatte. La soluzione che esso determina, quindi, può essere non ottimale, e talvolta anche non ammissibile per alcuni vincoli secondari. A questo provvede il secondo algoritmo, che applica alle soluzioni fornite dal primo la tecnica di ricerca locale nota come Tabu Search, la quale può essere applicata anche a soluzioni ottenute mediante un'euristica d'arrotondamento, come suggerito in sezione 3.5.3. Sono quindi elencate e descritte diverse varianti dell'algoritmo di Tabu Search. Si riportano i risultati computazionali delle prove effettuate sull'istanza di riferimento dell'insieme di test.

4.1 Algoritmo base

L'idea alla base della procedura euristica, descritta dall'algoritmo 1, consiste nel costruire, giorno per giorno, la schedulazione di ogni operatore.

Algorithm 1 Euristica costruttiva: algoritmo base

```
1: (wb, we, rest, ww, pw, day) ← inizializzazione()
2: shift ← ordinaTurni(T)
3: for all l ∈ L do
4:     verificaPeriodo(l)
5:     (wb, we, rest, ww, pw, day) ← aggiornaFinestre(l)
6:     op ← ordinaOperatori(N)
7:     table[*][l] ← assegnamento(l, op, shift, wb, we, rest, ww, pw, day)
8: end for
```

L'algoritmo cerca di assegnare i turni agli operatori considerando sia la difficoltà di accoppiare i turni tra di loro sia la difficoltà di effettuare un turno da parte dell'operatore.

Per determinare quali turni un operatore può effettuare nel giorno corrente, l'algoritmo associa ad ogni operatore i una finestra temporale `[wb[i],we[i]]` che definisce le ore lavorative utili. Un turno j può essere effettuato dall'operatore se e solo se è completamente contenuto nella finestra temporale. La dimensione della finestra temporale è determinata dal turno effettuato il giorno precedente attraverso la procedura descritta dall'algoritmo 2. La matrice `table` : $N \times L \rightarrow T$, rappresenta l'assegnazione dei turni secondo uno schema *nurse-day*. La variabile `day[i]` riporta il numero di giorni lavorati consecutivamente fino ad oggi, mentre `rest[i]` indica il numero di riposi di cui l'operatore ha goduto. Le variabili `ww[i]` e `pw[i]` indicano, rispettivamente, il montante delle ore lavorate nella settimana e nel periodo di riferimento, mentre `w[i]` e `h[i]` sono il numero teorico di ore che dovrebbero essere effettuate nel periodo e l'orario settimanale dell'operatore i. Come da modello `v[i]`, `d[j]` e `e[j]` rappresentano, rispettivamente, la durata di una giornata lavorativa per l'operatore i, la durata e l'orario di termine del turno j.

4.1.1 Ordinamento dei turni

Come prima cosa l'algoritmo ordina i turni la difficoltà di assegnamento cioè secondo il numero di turni predecessori leciti. Un turno j può essere assegnato nel giorno l, a un operatore, solo se il turno effettuato da questi il giorno $l-1$ termina piú di 11 ore prima dell'inizio di j. Si enumerano quindi i turni del giorno $l-1$ che possono precedere j. Minore è la dimensione di questo insieme e maggiore è la difficoltà di assegnare il turno j.

4.1.2 Verifica del periodo

Se il giorno l è il primo giorno della settimana, si azzera l'accumulatore delle ore lavorate nella settimana. Durante l'esecuzione dell'algoritmo il valore dell'accumulatore non deve mai superare le 48 ore (MAX_WEEK_HOURS).

4.1.3 Aggiornamento delle finestre

Se il turno effettuato dall'operatore è un riposo la procedura azzera il numero di giorni lavorati consecutivi, aumenta il numero di riposi effettuati e imposta la finestra alla dimensione massima, pari al'intera giornata. Nel caso che l'operatore abbia effettuato un giorno di ferie o di malattia, la procedura incrementa il numero di giorni lavorati e aumenta sia il totale ore del periodo che quello della settimana di un numero di ore pari alla durata media di una giornata, v_i, stabilito dal contratto dell'operatore. Anche in questo caso la finestra temporale viene fissata alla giornata completa. In ultimo, se l'operatore ha effettuato il turno lavorativo j, la procedura incrementa il numero di giorni lavorati e aggiunge ai totali del periodo e della settimana la durata in ore del turno j. L'inizio della finestra temporale è fissato a 11 ore (REST_TIME) dal termine del turno j.

Algorithm 2 Procedura che definisce, giorno per giorno, la dimensione della finestra temporale.

```
1: procedure AGGIORNAFINESTRE(Giorno l)
2:     for all i ∈ N do
3:         if table[i][l-1] = REST_SHIFT then
4:             day[i] ← 0
5:             wb[i] ← 0.0
6:             we[i] ← 24.0
7:             rest[i] ← rest[i] + 1
8:         else if table[i][l-1] ∈ { ILL_SHIFT, VACATION_SHIFT } then
9:             day[i] ← day[i] + 1
10:            wb[i] ← 0.0
11:            we[i] ← 24.0
12:            ww[i] ← ww[i] + v[i]
13:            pw[i] ← pw[i] + v[i]
14:        else
15:            j ← table[i][l-1]
16:            day[i] ← day[i] + 1
17:            wb[i] ← MAX(e[j] + REST_TIME - 24.0, 0.0)
18:            we[i] ← 24.0
19:            ww[i] ← ww[i] + d[j]
20:            pw[i] ← pw[i] + d[j]
21:        end if
22:    end for
23: end procedure
```

4.1.4 Ordinamento degli operatori

Per ogni giornata, gli operatori sono ordinati per:

1. valori crescenti della dimensione della finestra temporale;
2. valori decrescenti delle ore non lavorate nell'intero periodo;
3. valori decrescenti delle ore non lavorate nella settimana corrente.

Il primo criterio riflette la difficoltà di assegnare un turno all'operatore, mentre i rimanenti due criteri cercano di assegnare un turno a chi ha un maggior numero di ore da lavorare.

4.1.5 Assegnamento dei turni

La procedura verifica, per ogni operatore, se deve effettuare un riposo, un giorno di ferie o uno di malattia. In caso contrario controlla se l'operatore appartiene all'insieme di quelli che svolgono la sequenza ciclica. Se l'operatore è tra questi cerca un turno valido per la sequenza, e che appartenga al medesimo reparto dell'operatore. Se non esiste nessun turno di questo tipo, o l'operatore non può

svolgerlo, allora prova con un qualsiasi turno valido per la sequenza. Se, anche in questo caso, non esiste nessun turno disponibile che può essere svolto dall'operatore, allora l'algoritmo prova ad assegnargli un qualsiasi turno, partendo da quelli piú difficili da assegnare. Se, alla fine, non è stato trovato nessun turno valido per l'operatore, allora l'algoritmo gli assegna automaticamente un riposo.

L'assegnamento dei turni agli operatori è effettuato dall'algoritmo 3. Esso è suddivisa in tre parti, la prima effettua gli assegnamenti che sono forzati dai vincoli generali, la seconda effettua gli assegnamento obbligatori per i vincoli dei reparti, mentre la terza cerca di assegnare i rimanenti turni agli operatori liberi.

Algorithm 3 Assegnamento iniziale

```
1: procedure ASSEGNAMENTO(Giorno l, Operatore[] op, Turno[] shift)
                                                        ▷ Rispetto dei vincoli
2:     assegnamentoVincolato(l, op, shift);
                                        ▷ Assegnamenti all'interno dei reparti
3:     assegnamentoReparti(l, op, shift);
                                                    ▷ Assegnamenti rimanenti
4:     assegnamentoGenerico(l, op, shift);
5: end procedure
```

Rispetto dei vincoli

L'algoritmo 4 effettua alcuni assegnamenti, che risultano obbligatori per garantire il rispetto dei seguenti vincoli:

- limite massimo di giorni consecutivi lavorati;
- i periodi di ferie e di malattia;
- le sequenza cicliche di turni predefinite.

Questa fase consiste nel:

1. assegnare un turno di riposo ai lavoratori che hanno raggiunto il limite massimo di giorni consecutivi lavorati;
2. assegnare un giorno di ferie o di malattia agli operatori che ne hanno diritto;
3. assegnare un turno secondo la sequenza ciclica predefinita, cercando di rispettare il reparto d'appartenenza dell'operatore.

La funzione `appartieneASequenza` identifica gli operatori che devono effettuare i turni secondo una sequenza prestabilita. Il compito di cercare un turno in sequenza che possa essere eseguito dall'operatore è affidato alle funzioni `cercaTurnoInSequenzaEInReparto`, limitatamente al reparto d'appartenenza dell'operatore, e `cercaTurnoInSequenza`, per i rimanenti reparti. Un

Algorithm 4 Assegnamento iniziale: Rispetto dei riposi, ferie e malattie, e delle sequenze di turni ciclici

```
1: richiesteTerminate ← false
2: for all i ∈ N do
3:     if d[op[i]] ≥ MAX_WORKING_DAY then
4:         table[op[i]][l] ← REST_SHIFT;
5:     else if d[op[i]] ≥ maxd[op[i]] ∧¬ appartieneASequenza(op[i]) then
6:         table[op[i]][l] ← REST_SHIFT;
7:     else if l ∈ ill[op[i]] then
8:         table[op[i]][l] ← ILL_SHIFT;
9:     else if l ∈ vac[op[i]] then
10:        table[op[i]][l] ← VACATION_SHIFT;
11:    else if richiesetTerminate then
12:        table[op[i]][l] ← REST_SHIFT;
13:    else
14:        if appartieneASequenza(op[i]) then
15:            trovato ← cercaTurnoInSequenzaEInReparto(op[i], j)
16:            if trovato then
17:                assegna(op[i], l, j)
18:            else
19:                trovato ← cercaTurnoInSequenza(op[i], j)
20:            end if
21:        end if
22:    end if
23: end for
```

turno, per poter essere effettuato da un operatore, deve rispettare la seguente condizione

$$\text{orario(j)} \in [\text{wb}[i], \text{we}[i]] \wedge \text{ww}[i] + d[j] \leq \text{MAX\_WEEK\_HOURS} \tag{4.1}$$

verificata tramite la funzione `turnoValido`. Se si trova un turno valido, la procedura `assegna` associa il turno all'operatore e, contestualmente, decrementare il carico giornaliero sia complessivo sia quello relativo al turno.

Al termine di questa prima fase, agli altri operatori non è stato assegnato nessun turno. I turni che non appartengono alla sequenza ciclica non risultano coperti, e alcuni di quelli della sequenza ciclica, possono essere non coperti a causa di ferie o malattie.

Assegnamenti all'interno dei reparti

La seconda parte dell'algoritmo (algoritmo 5) assegna i turni non coperti agli operatori che appartengono al medesimo reparto del turno.

Algorithm 5 Rispetto dei vincoli

```
1:  for all i ∈ N do
2:      if op[i] non assegnato then
3:          trovato ← false
4:          for j ← 1; j ∈ T ∧ ¬ trovato; j ← j + 1 do
5:              if c[shift[j]][l] > 0 then
6:                  if turnoValido(shift[j], op[i]) ∧ repartoIdentico(shift[j], op[i])
                        then
7:                      assegna(op[i], l, shift[j])
8:                      trovato ← true
9:                  end if
10:             end if
11:         end for
12:     end if
13: end for
```

Assegnamenti rimanenti

L'algoritmo 6 assegna, i rimanenti turni scoperti agli operatori liberi. Se non è possibile assegnare all'operatore un turno, o i turni da coprire sono terminati, assegna automaticamente un turno di riposo.

Al termine ad ogni operatore è assegnato un turno, ma è possibile che esistano turni non coperti.

Complessità dell'algoritmo La complessità temporale asintotica dell'algoritmo di assegnazione dei turni (algoritmo 3) è $O(T \log T + L(N \log N + NT))$ che, ipotizzando $T \approx N$, risulta essere $O(LNT)$. L'occupazione in spazio è di dimensione $O(LN)$, principalmente a causa della matrice degli assegnamenti.

Algorithm 6 Assegnamento iniziale: Assegnamenti rimanenti

```
1: for all i ∈ N do
2:     if op[i] non assegnato then
3:         trovato ← false
4:         for j ← 1; j ∈ T ∧ ¬ trovato; j ← j + 1 do
5:             if c[shift[j]][l] > 0 then
6:                 if turnoValido(shift[j], op[i]) then
7:                     assegna(op[i], l, shift[j])
8:                     trovato ← true
9:                 end if
10:            end if
11:        end for
12:        if ¬ trovato then
13:            table[op[i]][l] ← REST_SHIFT
14:        end if
15:    end if
16: end for
```

Correttezza dell'algoritmo L'algoritmo garantisce il rispetto rigoroso dei seguenti vincoli rigidi:

- numero massimo di giorni lavorati consecutivamente;
- rispetto delle ferie;
- rispetto delle malattie;
- riposo minimo tra due turni successivi;
- assegnamento, ad operatore, di un solo turno al giorno.

Cerca di rispettare il piú possibile i seguenti vincoli flessibili:

- sequenze cicliche di turni;
- permanenza nel reparto.

La natura greedy dell'algoritmo fa si che nessuna delle decisioni presa in precedenza venga mai messa in discussione. Ciò può portare ad ottenere soluzioni non ottimali oppure, in alcuni casi, non ammissibili. L'algoritmo, infatti, potrebbe non riuscire garantire i seguenti vincoli:

- copertura dei turni richiesti;
- il numero minimo di riposi al mese.

4.2 Un algoritmo di ricerca locale

L'algoritmo costruttivo può non riuscire a soddisfare i vincoli di copertura del servizio. Possiamo intervenire modificando la soluzione non ammissibile, tramite una ricerca locale.

La ricerca locale permette di migliorare una soluzione tramite piccole modifiche. Ad ogni passo il metodo esplora la soluzione alla ricerca di intorni che possono ridurre il costo della soluzione. Al termine di ogni esplorazione viene applicata la modifica che presenta il miglioramento piú alto. La procedura si ferma se raggiunge un ottimo locale.

È possibile applicare la ricerca locale sia durante la costruzione della soluzione, sia sulla soluzione completa finale. Il primo caso offre il vantaggio di un ulteriore grado di libertà nell'effettuare le mosse. Uno scambio è ammissibile quando rispetta i vincoli imposti dagli assegnamenti delle giornate precedenti. Gli scambi effettuati su soluzioni complete invece devono tenere in considerazione anche gli assegnamenti successivi.

Per contro, il primo approccio è possibile solo durante l'esecuzione di un algoritmo che costruisce la soluzione procedendo giorno per giorno. Il secondo approccio, invece, è sempre possibile indipendentemente dal metodo di generazione della soluzione iniziale.

L'algoritmo 7 riporta le modifiche da apportare all'algoritmo 1 per applicare la ricerca locale.

Algorithm 7 Applicazione della ricerca locale all'algoritmo base

```
1: (wb, we, rest, ww, pw, day) ← inizializzazione()
2: shift ← ordinaTurni(T)
3: for all l ∈ L do
4:     verificaPeriodo(l)
5:     (wb, we, rest, ww, pw, day) ← aggiornaFinestre(l)
6:     op ← ordinaOperatori(N)
7:     table[*][l] ← assegnamento(l, op, shift, wb, we, rest, ww, pw, day)
8:     correzioneSoluzioneParziale()
9: end for
10: correzioneSoluzioneCompleta()
```

4.2.1 Definizione degli intorni di una soluzione

Data una soluzione iniziale la ricerca locale si basa sul tentativo di determinare una nuova soluzione "vicina" ad essa ma che ne riduca l'inammissibilità o ne migliori il valore. Cercare una soluzione vicina significa esplorare un *intorno* della soluzione attuale, per trovare una seconda soluzione la cui distanza da quella originale, definita in modo opportuno, sia minore di una certa soglia, e che porti ad un valore migliore della funzione obiettivo. In pratica, la differenza tra le due soluzioni deve essere circoscritta a poche variabili.

Diventa quindi necessario definire in modo formale gli intorni di una soluzione che andranno esplorati. Possiamo anche usare diversi intorni combinando quindi diversi meccanismi di ricerca locale. Un elenco di possibili intorni può essere trovato in [BCP01].

Ricordiamo che i modelli OP e Pattern ammettono che si rilassi un vincolo flessibile per ottenere una soluzione che rispetti tutti quelli rigidi.

Nel seguito elencheremo quelli impiegati nell'algoritmo proposto, che si dividono in due grandi categorie, secondo lo scopo che viene perseguito durante la loro esplorazione:

1. intorni che riducono la violazione dei vincoli rigidi;
2. intorni che migliorano il valore della funzione obiettivo.

4.2.2 Intorni per ridurre la violazione di copertura dei turni

Assegnazione di un turno non coperto ad un operatore in riposo

	Giorni				
	1	...	l	...	L
Soluzione iniziale					
Op. i	...	...	R	...	...
Soluzione finale					
Op. i	...	...	j	...	...

Fig. 4.1: Assegnazione di un turno non coperto ad un operatore in riposo

Sia j un turno non coperto il giorno l. Se esiste un operatore i, in riposo il giorno l, tale che:

1. il numero di giorni lavorati da i consecutivamente, sia prima che dopo il giorno l, è inferiore al limite massimo, MAX_WORKING_DAY (MWD), di giorni lavorativi;
2. i può effettuare il turno j il giorno l ($j \in W_{il}$);
3. i non è in ferie o in malattia il giorno l.

allora all'operatore i è assegnato il turno l per il giorno l.

La figura 4.1 riporta un esempio di scambio.

Violazione dei vincoli In alcuni casi un operatore per il quale valgono le precedenti condizioni, è inquadrato nel ciclo 3+1. Assegnargli un quarto turno di lavoro consecutivo è una violazione del vincolo secondario del 3+1. Questo però permette di ridurre la violazione del vincolo principale della copertura del carico.

Costo computazionale Il costo consiste nella ricerca, per ogni turno non coperto di ogni ogni giorno dell'orizzonte temporale, di un operatore a riposo che lo possa coprire. Complessivamente si ha che il costo è asintoticamente uguale a $O(LTN)$.

Assegnazione di un turno non coperto tramite scambio di un turno tra due operatori

	Giorni				
	1	...	l	...	L
Soluzione iniziale					
Op. i	...	...	R	...	...
Op. i_2	...	...	j_2	...	...
Soluzione finale					
Op. i	...	...	j_2	...	...
Op. i_2	...	...	j	...	...

Fig. 4.2: Assegnazione di un turno non coperto tramite scambio di un turno tra due operatori

Sia j un turno non coperto il giorno l. Se esistono un operatore i, in riposo il giorno l, ed un operatore i_2 al quale è assegnato il turno j_2 il giorno l, tali che:

1. il numero di giorni lavorati da i è inferiore a MWD;
2. i può effettuare il turno j_2 il giorno l;
3. i_2 può effettuare il turno j il giorno l;
4. i non è in ferie o in malattia il giorno l.

allora, per il giorno l, all'operatore i è assegnato il turno j_2, mentre all'operatore i_2 è assegnato il turno j.

La figura 4.2 riporta un esempio di questo scambio.

Violazione dei vincoli Se l'operatore i, è inquadrato nel ciclo 3+1 è possibile aver violato il vincolo debole del 3+1. Questo però permette di ridurre la violazione del vincolo principale della copertura del carico.

Costo computazionale Anche in questo caso il costo è asintoticamente pari a $O(LN^2T)$.

	Giorni						
	1	...	l_2	...	l	...	L
Soluzione iniziale							
Op. i	...	...	j_2	...	R	...	...
Op. i_2	...	...	R	...	j_3	...	...
Soluzione finale							
Op. i	...	...	R	...	j	...	...
Op. i_2	...	...	j_2	...	j_3	...	...

Fig. 4.3: Assegnazione di un turno non coperto tramite scambio di un riposo tra due operatori

Assegnazione di un turno non coperto tramite scambio di un riposo tra due operatori

Sia j un turno non coperto il giorno l. Sia i un operatore a riposo il giorno l che potrebbe effettuare il turno j, ma violerebbe il massimo numero di giorni di lavoro consecutivi, e sia j_2 il turno effettuato dall'operatore il giorno l_2 che sta al massimo MWD da l. Sia i_2 un operatore che ha effettuato l'ultimo riposo il giorno l_2. Se vale che:

1. l'assegnazione di un riposo il giorno l_2 all'operatore i, gli permetterebbe di effettuare il turno j senza violare il limite di giorni;
2. la cancellazione del riposo nel giorno l_2 non comporta, per l'operatore i_2, il superamento del limite di giorni lavorati consecutivamente;
3. i_2 può effettuare il turno j_2 il giorno l_2;
4. i non è in ferie o in malattia il giorno l;
5. i_2 non è in ferie o in malattia il giorno l_2.

allora l'operatore i effettua un riposo il giorno l_2 ed il turno j il giorno l. L'operatore i_2, invece, effettua il turno j_2 il giorno l_2.

In figura 4.3 è riportata la soluzione prima e dopo dello scambio.

Violazione dei vincoli Entrambi gli operatori, i e $i2$, potrebbero essere inquadrato nel ciclo 3+1 e quindi è, possibile aver violato il vincolo debole del 3+1. Per l'operatore j_2 è possibile aver violato il vincolo debole del numero minimo di riposi. Questo però permette di ridurre la violazione del vincolo principale della copertura del carico.

Costo computazionale Se effettuato per tutti i giorni e tutti i turni è necessario effettuare una seconda ricerca tra gli operatori nei MWD giorni sia precedenti che successivi a l. Il costo è quindi, asintoticamente, pari a $O(LN^2T)$.

Copertura di un turno tramite assegnazione di un riposo ad un operatore in ferie o malattia

	Giorni						
	1	...	l_2	...	l	...	L
	Soluzione iniziale						
Op. i	...	...	F	...	R	...	...
	Soluzione finale						
Op. i	...	...	R	...	j	...	...

Fig. 4.4: Copertura di un turno tramite assegnazione di un riposo ad un operatore in ferie o malattia

Sia j un turno non coperto il giorno l. Sia i un operatore a riposo il giorno l il quale potrebbe effettuare il turno j, ma violerebbe il massimo numero di giorni di lavoro consecutivi. Sia l_2 un giorno nel quale l'operatore i era in ferie, o in malattia. Se vale che:

1. assegnando un riposo il giorno l_2, l'operatore i non supera piú il limite di giorni effettuando il turno j il giorno l.

allora l'operatore i effettua un riposo il giorno l_2 e il turno j il giorno l.

In figura 4.4 sono riportate le soluzioni prima e dopo lo scambio.

Violazioni dei vincoli Questo intorno permette di ridurre la violazione del vincolo rigido della copertura del carico. Avendo assegnato un riposo ad un operatore in ferie potrebbe anche ridurre il numero di straordinari o aumentare il numero di ore non lavorate da i, ma non si introducono nuove violazioni di vincoli.

Costo computazionale Il costo è asintoticamente pari a $O(LTN)$.

4.2.3 Intorni per ridurre la violazione dei vincoli flessibili

Rientro di un operatore nella sequenza ciclica di riferimento

Sia i un operatore che il giorno l effettua un turno diverso da quello che la sequenza ciclica prevede. Sia j il turno che l'operatore i dovrebbe effettuare il giorno l. Sia j_2 il turno che l'operatore i effettua il giorno l. Sia i_2 un operatore che effettua il turni j il giorno l. Se si verifica che:

1. l'operatore i può effettuare il turno j il giorno l
2. l'operatore i_2 può effettuare il turno j_2 il giorno l

allora l'operatore i effettua il turno j il giorno l e l'operatore i_2 il turno j_2.

In figura 4.5 è riportata la soluzione prima e dopo dello scambio.

	Giorni				
	1	...	l	...	L
Soluzione iniziale					
Op. i	...	...	j_2	...	...
Op. i_2	...	...	j	...	...
Soluzione finale					
Op. i	...	...	j	...	...
Op. i_2	...	...	j_2	...	...

Fig. 4.5: Rientro di un operatore nella sequenza ciclica di riferimento

Violazione dei vincoli Questo scambio permette di ridurre la violazione delle sequenze cicliche. Se i turni i e i_2 non sono di pari durata, lo scambio aumenta il numero di ore effettuate da un operatore e diminuisce quelle effettuate dall'altro. Questo può introdurre, o ridurre, gli straordinari e aumentare, o ridurre, il numero di ore non lavorate per i due operatori. La variazione dipende da come sono distribuiti i turni dei due operatori prima dello scambio.

Costo computazionale Essendo necessario considerare, giorno per giorno, ogni coppia di operatori il costo è asintoticamente pari a $O(LN^2)$, a patto che il calcolo della funzione obiettivo sia in tempo costante.

Eliminazione di un turno di riposo

	Giorni						
	1	...	$l-1$	l	$l+1$	...	L
Soluzione iniziale							
Op. i	...	...	R	R	R	...	...
Op. i_2	...	...	...	j	...	...	...
Soluzione finale							
Op. i	...	...	R	j	R	...	...
Op. i_2	...	...	...	R	...	...	...

Fig. 4.6: Eliminazione di un turno di riposo

Sia i un operatore che effettua un alto numero di riposi. Sia l un giorno centrale di una sequenza di tre o piú riposi consecutivi assegnati all'operatore i. Sia i_2 un operatore che effettua pochi riposi. Sia j il turno effettuato dall'operatore i_2 il giorno l. Se:

1. il turno j introduce per l'operatore i meno ore straordinarie di quante ne rimuove per i_2;
2. il turno j non è della sequenza dei turni per l'operatore i_2 nel giorno l;
3. i non è in ferie o in malattia il giorno l.

oppure:

1. l'operatore i_2 non effettua il numero minimo di riposi;
2. i non è in ferie o in malattia il giorno l.

allora, per il giorno l, si assegna il turno j all'operatore i e l'operatore i_2 effettua un turno di riposo.

In figura 4.6 sono riportate le soluzione prima e dopo dello scambio.

Violazione dei vincoli Se per l'operatore i_2 non è rispettato il vincolo flessibile del numero minimo di riposi allora lo scambio diminuisce la violazione del vincolo stesso. Altrimenti migliora la soluzione diminuendo il numero di straordinari e permette di ridurre la violazione del vincolo flessibile che chiede una distribuzione equa dei carichi.

Costo computazionale Essendo necessario cercare, giorno per giorno, una coppia di operatori il costo è asintoticamente pari a $O(LN^2)$.

4.3 Algoritmo di Tabu Search

Come accennato in precedenza, la ricerca locale termina quando giunge ad un ottimo locale non riuscendo piú a migliorare la soluzione. Se volessimo imporre alla ricerca locale, per uscire dall'ottimo locale, di effettuare anche mosse che non migliorino la soluzione o, sotto determinate regole, la peggiorino, potremmo correre il rischio di entrare in un ciclo infinito di ripetizioni [Wol98, §12.3]. Se, ad esempio, dopo un certo numero di scambi si ritorna ad una soluzione precedente, la ricerca locale, essendo un algoritmo deterministico, rischia di continuare ad effettuare sempre la stessa sequenza di scambi. Questa ipotesi non è poi cosi tanto remota, basti pensare a differenti intorni che hanno come scopo il migliorare una soluzione nell'ottica di due distinte funzioni obiettivo, cosa molto comune per NRP. Se le due funzioni obiettivo sono in netta contrapposizione un intorno potrebbe cercare di cancellare, o diminuire, le modifiche alla soluzione apportate dall'altro. Ad esempio, se il primo intorno per ridurre la violazione del vincolo del 3+1 viola la distribuzione delle notti, un secondo intorno che distribuisce in modo equo le notti potrebbe cancellare la mossa precedente.

Tabu Search definisce un insieme di mosse come vietate, o lista tabu. Di volta in volta applica soltanto le sostituzioni che non appartengono all'insieme di quelle presenti nella lista tabu. Una mossa viene inserita nella lista tabu quando, se applicata, ricondurrebbe ad una soluzione precedente.

4.3.1 Definizione di mossa vietata

Resta da definire quale sia una mossa vietata e quanto deve essere grande la lista tabu.

Determinare se una mossa è vietata

Una prima idea consiste nel memorizzare in una lista tutte le soluzioni precedenti. In questo modo, la verifica si riduce a cercare, fra loro, la soluzione che sarebbe generata se fosse effettuata la mossa. Se la soluzione futura non è presente nella lista allora la mossa è valida, altrimenti viene rifiutata. Questo metodo richiede un grande spazio di memoria, dovendo memorizzare molte soluzioni precedenti. La dimensione di una soluzione è nell'ordine di $N \times L$ variabili, e cioè nell'ordine dei KiloByte: l'esplorazione di un migliaio di mosse rende l'occupazione di memoria pesante, ma ancora piú pesante è il tempo necessario ad effettuare la verifica di validità. L'esito della verifica richiede nel caso peggiore kNL confronti, dove k è la lunghezza della lista.

In realtà non è necessario considerare l'intera soluzione, basta memorizzare solamente le differenze tra soluzioni successive e limitarsi a verificare le regioni modificate. In questo caso confrontiamo i singoli *attributi* modificati. Ad esempio si potrebbe salvare i turni della soluzione iniziale (dominio) che sono coinvolti nella mossa e quelli della soluzione finale (codominio). Se il codominio della mossa scelta si sovrappone al dominio, definito dagli attributi presente nella tabu list, di una mossa precedente, la nuova mossa viene rifiutata.

Ogni mossa effettuata viene registrata come due insiemi di triple. Ogni tripla, definita come `<Operatore, Giorno, Turno>`, indica lo stato di una variabile decisionale `x[Operatore,Giorno] = Turno`. Il primo insieme contiene lo stato delle variabili prima della mossa mentre il secondo contiene lo stato successivo alla mossa. Per ridurre lo spazio occupato dalla lista tabu ogni coppia può essere descritta da una tupla di quattro valori `<Operatore, Giorno, Turno Vecchio, Turno Nuovo>`.

Dimensione della lista tabu

La dimensione della lista tabu è critica sia in termini di prestazioni che in termini di correttezza del rifiuto. Una lista troppo breve vanificherebbe di fatto l'utilizzo della tabu search. Una lista troppo lunga, d'altro canto, correrebbe il rischio di rifiutare troppe mosse valide, tenendo in considerazione mosse avvenute molto tempo prima. Una lista di sette elementi sembra essere spesso citata come una dimensione accettabile [Wol98, §12.3.1].

4.4 Risultati computazionali

Di seguito sono riportati gli esiti dell'applicazione degli algoritmi su due istanze originali del problema. La tabella 4.1 riporta i dati del test effettuato sull'istanza di riferimento. Invece, la tabella 4.2 riporta i dati del test effettuato sull'istanza `I-12-9`, per la quale esiste una stima piú corretta dei bound effettuata tramite CPLEX.

Soluzione			Violazione dei vincoli					T
Metodo	Valore	Gap	Cop.	Rip.	3+1	Rep.	Seq.	
Base	4273,94	9805%	4	0	12	22	3	$< 1s$
Ricerca Locale	327,54	659%	0	0	35	30	17	$< 1s$
Lower bound	43,15							29,4s
Soluzione `glpsol`	Non trovata							30 min

Tab. 4.1: Comparazione delle euristiche applicate all'istanza di riferimento

Soluzione			Violazione dei vincoli					T
Metodo	Valore	Gap	Cop.	Rip.	3+1	Rep.	Seq.	
Base	43730,61	46020%	43	1	12	26	3	$< 1s$
Ricerca Locale	21325,74	22391%	20	3	94	40	46	$< 1s$
Lower bound `CPLEX`	94,82							-
Upper bound `CPLEX`	116,30							-
Lower bound `glpsol`	89,23							17,5 s
Soluzione `glpsol`	198,72	110%						195 s

Tab. 4.2: Comparazione delle euristiche applicate all'istanza `I-12-9`

5. ANALISI DELL'APPLICAZIONE

> "Quarantadue!" urlò Loonquawl. "E' tutto quello che hai da dirci dopo sette milioni e mezzo di anni di lavoro?"
> "Ho controllato con grande minuziosità" disse il Computer "e questa è la risposta veramente definitiva. Credo che, se devo essere franco, il problema stia nel fatto che voi non avete mai realmente saputo quale fosse la domanda."
>
> *Guida Galattica per gli Autostoppisti* - Douglas Adams

Sommario

Questo capitolo consiste nell'analisi dell'applicazione. Inizialmente si definiscono e descrivono i requisiti. Successivamente, i vari requisiti vengono formalizzati tramite diagrammi UML. Altri diagrammi UML descrivono i flussi applicativi e la struttura dell'applicazione.

5.1 Definizione dei requisiti

Di seguito sono riportati i principali requisiti che l'applicazione deve rispettare. Per ogni requisito è stata definita una scheda che riporta:

- il nome;
- il codice assegnatogli;
- la classificazione;
- l'eventuale requisito generale nel quale esso è incluso o da cui deriva;
- gli eventuali requisiti che contrastano, anche solo in parte, con esso;
- una breve descrizione.

La classificazione dei requisiti avviene per importanza. Sono infatti definiti tre livelli che indicano se e in che fase deve essere implementato il singolo requisito. I livelli definiti sono:

MUST il requisito è di primaria importanza e deve, quindi, essere gestito fin dalle prime fasi del progetto;

SHOULD deve essere obbligatoriamente realizzato entro il termine del progetto, ma non è necessario che sia presente fin da subito;

MAY il requisito è opzionale cioè può, sotto certe condizioni, essere tralasciato anche dalla versione finale dell'applicativo.

Nel caso che il requisito sia un sotto requisito il suo livello è subordinato al livello del padre. Ad esempio un sotto requisito MUST di un requisito MAY è meno importante di un requisito principale di livello SHOULD.

Nei requisiti quando si fa riferimento ad un *operatore* si indica un lavoratore per il quale deve essere definita la schedulazione dei turni. Il termine *responsabile* fa invece riferimento al responsabile del servizio, il quale ha il compito di definire i turni che gli operatori andranno a svolgere.

5.1.1 Requisiti funzionali

I requisiti elencati di seguito sono descritti con maggior dettaglio nel capitolo 1.

Titolo	Normativa del lavoro				
Codice	F1	**Classe**	MUST	**Super**	Nessuno
Requisiti opposti	Nessuno				
Il software, nel definire la pianificazione, deve rispettare le leggi dello stato Italiano che regolano il mondo del lavoro. Queste norme sono state piú volte modificate nel tempo: si parte quindi da regi decreti fino ad arrivare a leggi promulgate solo di recente. In particolare, le ultime modifiche alla normativa sono presenti in [Ita03] e [Ita04].					

Titolo	Bisogna rispettare i limiti di orario				
Codice	F1.1	**Classe**	MUST	**Super**	F1
Requisiti opposti	F1				
In una settimana è possibile lavorare al massimo 48 ore. Un lavoratore può effettuare al massimo sei giornate di lavoro consecutive. Tra due turni successivi al lavoratore spetta un riposo di almeno 11 ore. L'orario base può essere superato solo per sei settimane consecutive.					

Titolo	Gli straordinari vanno calcolati su base mensile				
Codice	F1.2	**Classe**	SHOULD	**Super**	F1
Requisiti opposti	F1				
Gli straordinari sono calcolati come la differenza tra le ore effettivamente lavorate nel mese e quelle teoriche.					

Titolo	Diritto alle ferie e alle malattie				
Codice	F1.3	**Classe**	MUST	**Super**	F1
Requisiti opposti	F1				
Se ad un operatore sono state assegnate delle giornate di malattia o ferie, queste devono essere rispettate. La schedulazione non può imporre delle giornate di malattia ad un operatore che non ne richiede. La schedulazione può imporre, o consigliare, delle giornate di ferie.					

Titolo	CCNL UNEBA				
Codice	F2	**Classe**	MUST	**Super**	Nessuno
Requisiti opposti		Nessuno			
Il Contratto Collettivo del Lavoro UNEBA ([ACCU03] e [UCCU01]) definisce, a livello nazionale prima e a livello regionale dopo, dei vincoli aggiuntivi rispetto alla normativa del lavoro.					

Titolo	Riduzione d'orario base				
Codice	F2.1	**Classe**	SHOULD	**Super**	F2
Requisiti opposti		F2			
L'orario base settimanale è di 38 ore organizzate su sei giorni alla settimana. La durata della singola giornata lavorativa è di 7 ore e 20 minuti.					

Titolo	Giorno di riposo				
Codice	F2.2	**Classe**	SHOULD	**Super**	F2
Requisiti opposti		F2			
Per ogni lavoratore deve essere stabilito un giorno della settimana durante il quale è a riposo. Se non è possibile per il lavoratore usufruire della giornata di riposo, gli si deve garantire un riposo compensativo entro la settimana.					

Titolo	Limite annuo al numero di straordinari				
Codice	F2.3	**Classe**	SHOULD	**Super**	F2
Requisiti opposti		F2			
Un operatore può effettuare al massimo 120 ore di straordinario in un anno.					

Titolo	Organizzazione del lavoro nella casa di riposo				
Codice	F3	**Classe**	MUST	**Super**	Nessuno
Requisiti opposti		Nessuno			
La gestione della casa di riposo può organizzare il proprio lavoro come le aggrada. Nei limiti di legge la pianificazione deve rispettare le regole definite dalla casa di riposo.					

Titolo	Operatori aderenti alla sequenza di turni 3+1				
Codice	F3.1	**Classe**	MUST	**Super**	F3
Requisiti opposti		F3			
Alcuni operatori sono inquadrati in una sequenza ciclica di turni che prevede una successione di tre giorni lavorativi ed uno di riposo.					

Titolo	Operatori assegnati ad un reparto				
Codice	F3.2	**Classe**	MUST	**Super**	F3
Requisiti opposti		F3			
Alcuni operatori possono essere assegnati ad un reparto chiamato nucleo. La schedulazione deve cercare di farli operare all'interno del nucleo a cui sono assegnati.					

Titolo	Operatori assegnati ad un turno specifico				
Codice	F3.3	**Classe**	SHOULD	**Super**	F3
Requisiti opposti		F3			
Ad alcuni operatori può essere assegnato un turno specifico. Generalmente la schedulazione deve assegnare questo turno all'operatore predefinito.					

Titolo	Numero di riposi minimi				
Codice	F3.3	**Classe**	SHOULD	**Super**	F3
Requisiti opposti		F3			
La casa di riposo garantisce ai propri operatori un numero minimo di riposi al mese, attualmente pari a 5.					

Titolo	Carico lavorativo variabile				
Codice	F3.4	**Classe**	SHOULD	**Super**	F3
Requisiti opposti		F3			
Il carico lavorativo che la casa di riposo richiede alla cooperativa, è composto da una parte fissa e da una variabile.					

Titolo	Utilizzo di lavoratori temporanei				
Codice	F3.5	**Classe**	MAY	**Super**	F3
Requisiti opposti		F3			
La cooperativa può utilizzare dei lavoratori temporanei per coprire le richieste della casa di riposo, nel caso che non riesca a sopperire con i propri operatori.					

Titolo	Limitazione dell'uso di operatori temporanei				
Codice	F3.6	**Classe**	SHOULD	**Super**	F3.5
Requisiti opposti		F3.5			
La schedulazione deve ridurre, se non eliminare, l'utilizzo dei lavoratori temporanei.					

Titolo	Copertura del carico lavorativo				
Codice	F3.7	**Classe**	MUST	**Super**	F3.4
Requisiti opposti		F3.4			
Tutti i turni richiesti al carico lavorativo stabilito dalla casa di riposo devono essere coperti. Non è possibile assegnare piú operatori del necessario.					

Titolo	Limitazione del numero di straordinari				
Codice	F3.8	**Classe**	SHOULD	**Super**	F3
Requisiti opposti		F3			
La schedulazione deve ridurre, se non eliminare, la richiesta di straordinari agli operatori.					

Titolo	Richieste degli operatori				
Codice	F4	**Classe**	MAY	**Super**	Nessuno
Requisiti opposti		Nessuno			
É facoltà degli operatori esprimere delle preferenze in merito agli orari da effettuare. Queste preferenze non sono vincolanti.					

5.1.2 Requisiti operativi

Titolo	Tempo necessario alla pianificazione				
Codice	O1	**Classe**	MUST	**Super**	Nessuno
Requisiti opposti		Nessuno			
Il tempo necessario deve essere inferiore a quello necessario per effettuare la schedulazione in modo manuale. Attualmente si impiegano 4 ore per la definizione di un mese e 1 ora al giorno per la gestione degli imprevisti.					

Titolo	Possibilità di inserire le richieste della casa di riposo				
Codice	O2	**Classe**	MUST	**Super**	F3.4
Requisiti opposti		F3.4			
Il responsabile deve poter indicare le richieste di turni extra della casa di riposo.					

Titolo	Possibilità di inserire i periodo di ferie e malattie				
Codice	O3	**Classe**	MUST	**Super**	F1.3
Requisiti opposti		F1.3			
Il responsabile deve poter inserire le giornate di malattia riconosciute agli operatori. Deve anche poter inserire e, se necessario, modificare, le giornate di ferie concordate con gli operatori.					

Titolo	Possibilità di indicare le differenze tra i turni effettivamente svolti e quelli pianificati				
Codice	O4	**Classe**	SHOULD	**Super**	Nessuno
Requisiti opposti		Nessuno			
Deve essere possibile registrare i turni effettivamente eseguiti dagli operatori, per potere tenere conto dei dati nelle successive schedulazioni.					

Titolo	Possibilità di rischedulare il periodo				
Codice	O5	**Classe**	SHOULD	**Super**	Nessuno
Requisiti opposti		Nessuno			
Deve essere possibile rieffettuare la schedulazione anche a periodo in corso. In questo caso tutti i turni effettuati prima di una certa data non possono essere modificati.					

Titolo	Possibilità di fissare dei turni				
Codice	O6	**Classe**	SHOULD	**Super**	Nessuno
Requisiti opposti		Nessuno			
Deve essere possibile fissare dei turni a discrezione del responsabile.					

Titolo	Rendiconto delle ore effettuate				
Codice	O7	**Classe**	MAY	**Super**	Nessuno
Requisiti opposti		Nessuno			
Alla fine del periodo il responsabile deve poter ottenere l'elenco delle ore effettuate dagli operatori a seconda del nucleo e della fascia oraria.					

Titolo	Indicazione delle ore non ancora assegnate				
Codice	O8	**Classe**	MAY	**Super**	Nessuno
Requisiti opposti		Nessuno			
Nel caso che la schedulazione non riesca ad assegnare tutte le ore che in teoria l'operatore deve effettuare, il software deve evidenziare, operatore per operatore, le ore che mancano per colmare il monte ore.					

5.1.3 Requisiti tecnologici

Titolo	Applicazione Win32				
Codice	T1	**Classe**	SHOULD	**Super**	Nessuno
Requisiti opposti		Nessuno			
L'applicazione deve poter funzionare su macchine che utilizzano i sistemi operativi Windows.					

Titolo	Macchine datate				
Codice	T2	**Classe**	MAY	**Super**	Nessuno
Requisiti opposti		Nessuno			
L'applicazione deve poter funzionare anche su macchine non recenti che utilizzino sistemi operativi relativamente vecchi (ad esempio Pentium con Windows 98 installato).					

Titolo	Reportistica tramite Microsoft Office				
Codice	T3	**Classe**	MAY	**Super**	T1
Requisiti opposti		T1			
Si deve provvedere ad esportare i dati in formato Excel.					

5.2 Descrizione degli scenari tramite casi d'uso

Di seguito sono riportati i principali scenari dell'applicazione. Per ogni scenario è presente la descrizione tramite diagrammi UML dei casi d'uso ed una breve scheda riepilogativa. Per un'approfondimento in merito alla sintassi relativa ai diagrammi dei casi d'uso è possibile far riferimento a [BRJ98, §16-§17].

5.2.1 Schedulazione del periodo

In figura 5.1, sono riportati i principali casi d'uso che si presentano durante la schedulazione dei turni. Il caso d'uso principale consiste nella definizione della schedulazione del periodo. I rimanenti cinque sono sottocasi di quello principale.

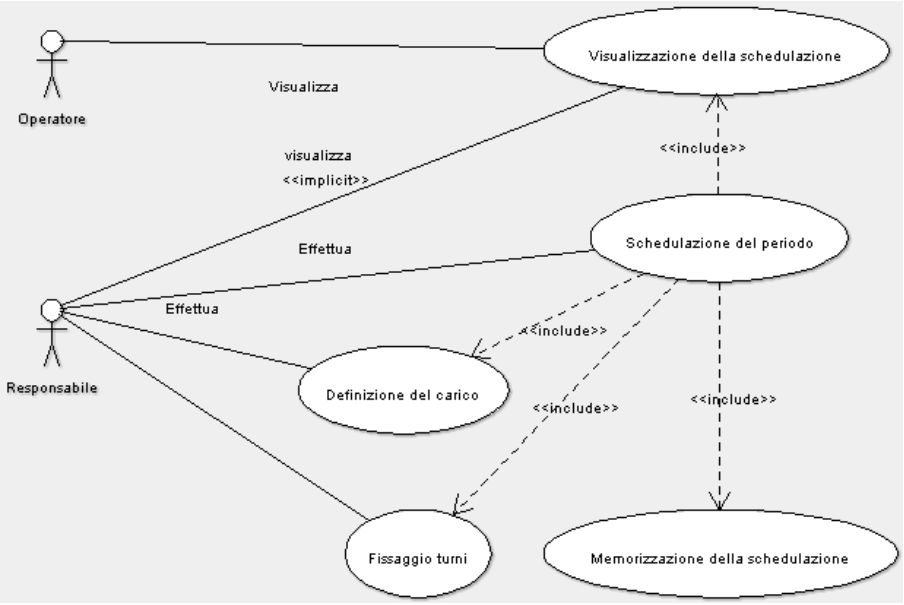

Fig. 5.1: Caso d'uso che descrive la schedulazione del periodo

Schedulazione del periodo

Informazioni caratteristiche

Obiettivo	Definire i turni che gli operatori svolgeranno nel periodo
Ambito	Casa di riposo
Livello	Organizzativo
Requisiti	La casa di riposo deve definire in modo preciso i turni da effettuare. Gli operatori devono comunicare i giorni di malattia. La cooperativa e gli operatori devono concordare i giorni di ferie.
Condizioni di successo	La procedura determina i turni per l'intero periodo.
Condizioni di fallimento	La procedura non riesce a determinare i turni per l'intero periodo e il responsabile non riesce a rilassare i vincoli.
Attore primario	Il responsabile del servizio
Evento scatenante	La necessità di definire i turni per il periodo

Scenario principale di successo

In figura 5.2 è riportato il diagramma degli stati nel caso che lo scenario abbia successo.

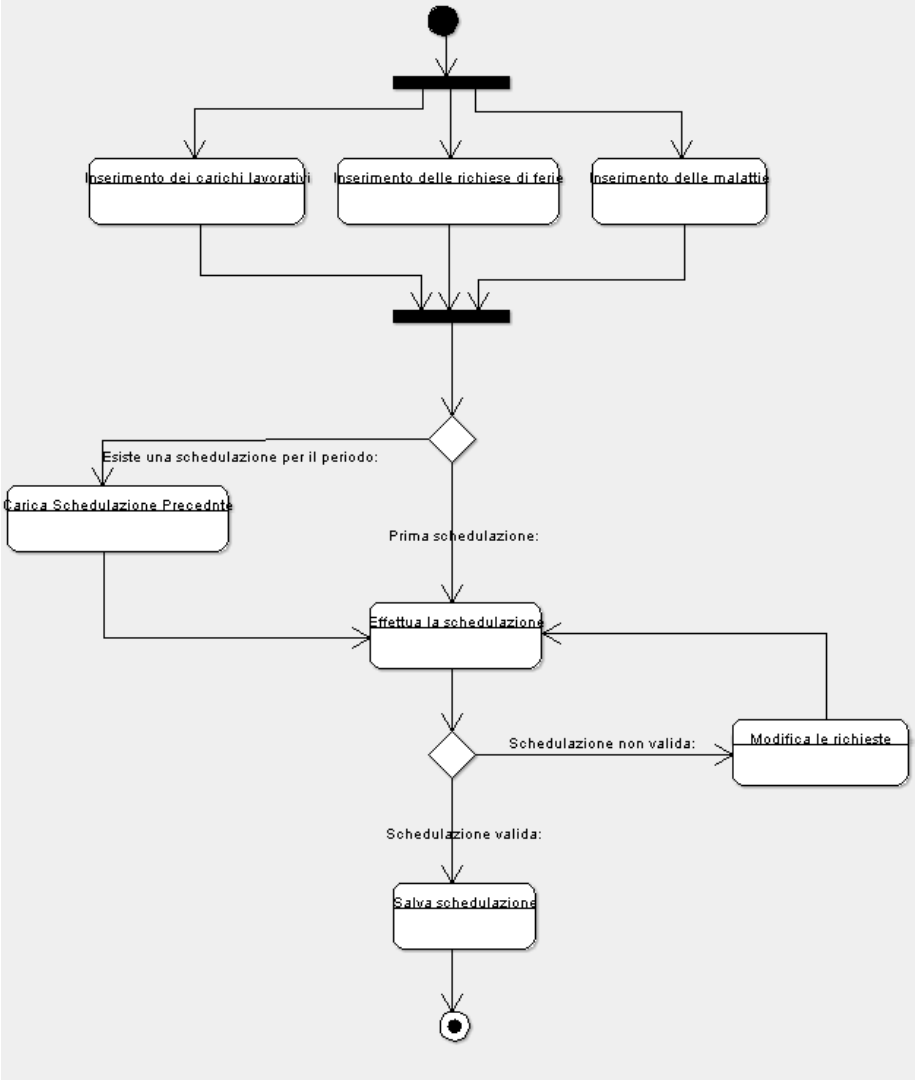

Fig. 5.2: Diagramma degli stati per la definizione con successo della schedulazione dei turni

Informazioni correlate

Schedulazione	Prima dell'inizio del periodo va effettuata la definizione completa. All'inizio della giornata vanno gestite le eccezioni
Performance	La schedulazione complessiva deve durare meno di 4 ore. Quella per gestire le eccezioni deve durare al massimo 1 ora.
Frequenza	La schedulazione complessiva viene eseguita mensilmente. La schedulazione per gestire le eccezioni viene effettuata giorno per giorno solamente se necessario.
Super caso d'uso	Nessuno
Sotto casi d'uso	Visualizzazione della schedulazione Fissaggio del carico Fissaggio dei turni Memorizzazione della schedulazione.
Canale dell'attore principale	Applicativo software
Attori secondari	Nessuno
Canali degli attori secondari	Nessuno

Visualizzazione della schedulazione

Informazioni caratteristiche

Obiettivo	Visualizzare i turni che gli operatori svolgeranno nel periodo
Ambito	Casa di riposo
Livello	Operativo
Requisiti	Deve essere stata definita una schedulazione per il periodo.
Condizioni di successo	L'interessato può visionare la tabella dei turni.
Condizioni di fallimento	L'interessato non riesce a visionare la tabella dei turni.
Attore primario	Il responsabile del servizio
Evento scatenante	La necessità di conoscere i turni per il periodo

Scenario principale di successo

Lo scenario ha successo se l'interessato riesce ad accedere alla tabella dei turni ed ad interpretarne le informazioni.

Informazioni correlate

Schedulazione	Può verificarsi in qualsiasi momento.
Performance	Si deve ottenere l'informazione nel tempo piú breve possibile.
Frequenza	Ogni qual volta si renda necessario consultare la tabella dei turni.
Super caso d'uso	Schedulazione dei turni
Sotto casi d'uso	Nessuno
Canale dell'attore principale	Applicativo software
Attori secondari	Operatori
Canali degli attori secondari	Bacheca

Definizione del carico

Informazioni caratteristiche

Obiettivo	Definire i turni che dovranno essere coperti dagli operatori. Inserire le giornate di ferie e di malattia che spettano agli operatori
Ambito	Casa di riposo
Livello	Organizzativo
Requisiti	Devono essere noti i dati da inserire.
Condizioni di successo	Il sistema ha acquisito i dati.
Condizioni di fallimento	Il sistema non ha acquisito i dati.
Attore primario	Il responsabile del servizio
Evento scatenante	La necessità di inserire i dati

Scenario principale di successo

Lo scenario ha successo se il responsabile del servizio riesce ad inserire tutti i dati desiderati nel sistema.

Informazioni correlate

Schedulazione	Può verificarsi in qualsiasi momento.
Performance	L'inserimento del'informazione deve essere immediata
Frequenza	Ogni qual volta si renda necessario aggiornare i dati.
Super caso d'uso	Schedulazione dei turni
Sotto casi d'uso	Nessuno
Canale dell'attore principale	Applicativo software
Attori secondari	Nessuno
Canali degli attori secondari	Nessuno

Fissaggio dei turni

Informazioni caratteristiche

Obiettivo	Collaborare con l'applicativo software nella definizione della schedulazione
Ambito	Casa di riposo
Livello	Organizzativo
Requisiti	Devono essere noti i turni da fissare.
Condizioni di successo	Il sistema ha acquisito i dati.
Condizioni di fallimento	Il sistema non ha acquisito i dati.
Attore primario	Il responsabile del servizio
Evento scatenante	La necessità di fissare qualche turno

Scenario principale di successo

Lo scenario ha successo se il responsabile del servizio ha fissato con successo i turni e l'applicazione ne tiene conto nel definire la schedulazione. Di fatto consiste nella fase di correzione di una soluzione non valida, mostrata in figura 5.2.

Informazioni correlate

Schedulazione	Durante la schedulazione dei turni.
Performance	L'inserimento dell'informazione deve essere immediata.
Frequenza	Ogni qual volta si renda necessario fissare un turno.
Super caso d'uso	Schedulazione dei turni
Sotto casi d'uso	Nessuno.
Canale dell'attore principale	Applicativo software
Attori secondari	Nessuno
Canali degli attori secondari	Nessuno

5.2.2 Rilevazione dei turni effettivamente svolti

In certi casi limitati i turni effettivamente svolti dagli operatori possono discostarsi da quelli preventivati dal responsabile. In questo caso è necessario registrare la variazione. In figura 5.3 sono riportati i casi d'uso coinvolti in questa fase.

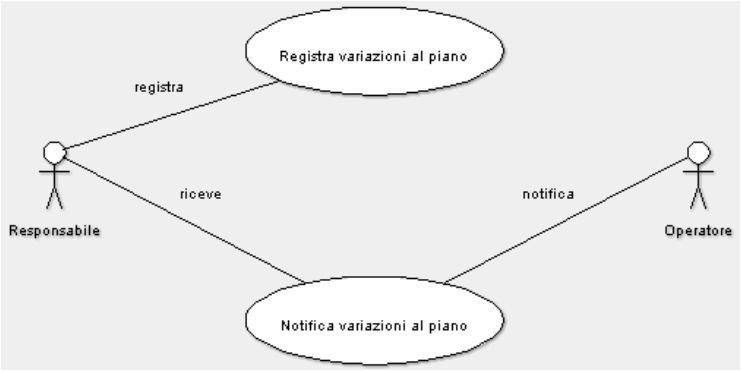

Fig. 5.3: Casi d'uso relativi alla registrazione di una variazione rispetto alla schedulazione

Notifica di una variazione rispetto al piano

Informazioni caratteristiche

Obiettivo	Notificare una modifica la piano
Ambito	Casa di riposo
Livello	Operativo
Requisiti	Deve essere nota la modifica.
Condizioni di successo	Il responsabile è stato informato.
Condizioni di fallimento	Il responsabile non è stato informato o non è d'accordo alla variazione.
Attore primario	Operatore
Evento scatenante	La necessità di variare un turno

Scenario principale di successo

Lo scenario ha successo se l'operatore riesce a comunicare la variazione responsabile e questi, accetta la modifica.

Informazioni correlate

Schedulazione	Durante il periodo.
Performance	Immediata.
Frequenza	Ogni qual volta un operatore desidera cambiare il turno.
Super caso d'uso	Nessuno
Sotto casi d'uso	Nessuno.
Canale dell'attore principale	Verbale
Attori secondari	Responsabile
Canali degli attori secondari	Verbale

Registrazione della variazione

Informazioni caratteristiche

Obiettivo	Registrare una variazione rispetto al piano originale
Ambito	Casa di riposo
Livello	Organizzativo
Requisiti	Deve essere nota la modifica da registrare.
Condizioni di successo	Il sistema ha acquisito i dati.
Condizioni di fallimento	Il sistema non ha acquisito i dati.
Attore primario	Il responsabile del servizio
Evento scatenante	La necessità di registrare una modifica.

Scenario principale di successo

Lo scenario ha successo se il responsabile registra la modifica.

Informazioni correlate

Schedulazione	In qualsiasi istante, purchè prima di una schedulazione giornaliera.
Performance	L'inserimento dell'informazione deve essere immediata.
Frequenza	Ogni qual volta si renda necessario registrare una variazione.
Super caso d'uso	Nessuno.
Sotto casi d'uso	Nessuno.
Canale dell'attore principale	Applicativo software
Attori secondari	Nessuno
Canali degli attori secondari	Nessuno

5.3 Struttura dell'applicazione

In figura 5.4 è riportato il diagramma dei componenti che costituiscono l'applicazione. L'idea di base consiste nel rendere il piú modulare possibile l'applicazione. In questo modo si cerca di ridurre le componenti da modificare se si volesse applicare l'applicazione anche ad altre realtà lavorative.

Le componenti sono:

Il motore per la definizione dei turni che genera la schedulazione per il periodo;

Il connettore al database che permette agli altri moduli di accedere ai dati presenti nel database;

L'esportatore di soluzioni che permette di salvare le schedulazioni in vari formati;

L'interfaccia grafica che permette all'utente di interagire con l'applicazione.

Di fatto questi componenti realizzano il pattern noto come *Model View Controller* o MVC, che separa l'applicazione in tre domini, nel primo dei quali risiedono i dati (Model), nel secondo la visualizzazione dei dati (View) e nel terzo i metodi per manipolare i dati e determinare le visualizzazioni (Controller). Nel

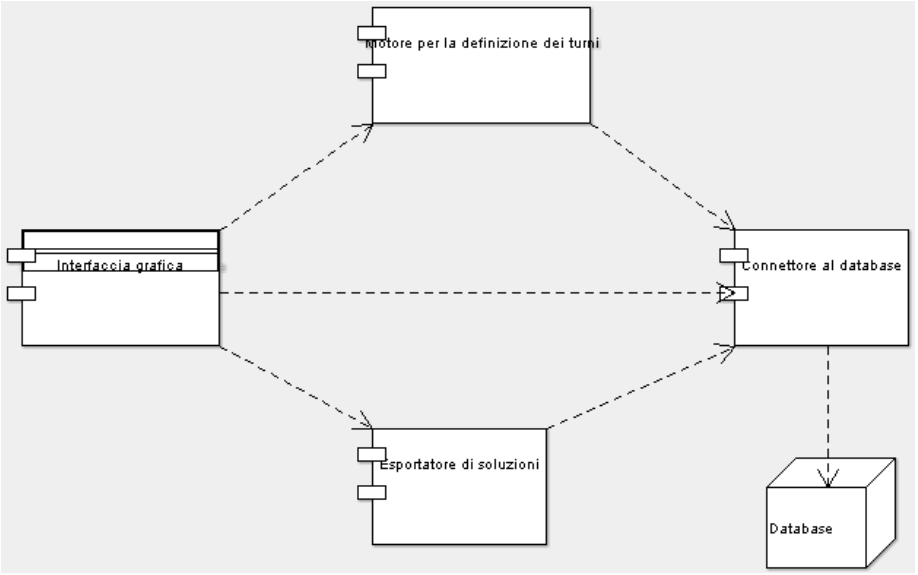

Fig. 5.4: Componenti dell'applicazione

caso che l'applicazione o le richieste crescano, è possibile, sempre con questa struttura base, separare l'applicazione in tre livelli distinti. In questo caso si definisce l'applicazione come *three-tier*. In riferimento a dove si posizionano le componenti dell'applicazione, nel percorso ideale che va dai dati fino all'utente, abbiamo che:

- il primo livello o *data level* contiene il database ed il connettore;
- il secondo livello o *application level* include i moduli del motore e dell'esportatore;
- il terzo livello o *presentation level* è costituito dall'interfaccia grafica.

In questo modo possiamo suddividere l'applicazione in tre parti che, possono anche essere realizzate da tre applicazioni che operano su macchine differenti.

Quindi l'applicazione gode di scalabilità e intercambiabilità. Se risulta necessario modificare il database o l'interfaccia grafica è sufficiente sostituire solamente i livelli interessati. Questa separazione di componenti permette che differenti database o interfacce grafiche coesistano nel medesimo livello. Ad esempio, potremmo avere un'applicazione client server che lavora con dati locali, i quali sono sincronizzati con un database centrale, che a sua volta, è interrogabile tramite un'applicazione web.

5.3.1 Descrizione dei componenti dell'applicazione

I componenti che costituiscono l'applicazione possono essere delineati in modo piú preciso definendone le caratteristiche piú salienti, quali ad esempio il compito, il numero di istanze necessarie e le possibili varianti. Possiamo anche suggerire il modo con cui realizzarli, tramite l'utilizzo di *design pattern*, senza per questo entrare nei dettagli implementativi.

Il motore per la definizione dei turni

Compito Il motore realizza gli algoritmi definiti nei capitoli precedenti. Questi algoritmi richiedono dei dati, sia attuali che storici, e dei parametri definiti dal responsabile del servizio. É compito del motore raccogliere tutte le informazioni necessarie agli algoritmi e, successivamente, definire una schedulazione valida dei turni.

Istanze necessarie In generale è sufficiente che esista un'unica istanza del motore. In alcune situazioni, ad esempio nel caso di un'applicazione distribuita o di un *application server* è obbligatorio che esista solamente un motore attivo. Se, ad esempio, due utenti richiedono una schedulazione per il medesimo periodo, potrebbe succedere che i due processi, basati su di un unico insieme di dati, forniscano soluzioni incoerenti per la schedulazione dello stesso periodo, a causa dei vincoli imposti dagli operatori. Tra l'altro, l'imposizione di un solo processo decisore ricalca il workflow del processo aziendale, indipendentemente dal contesto nel quale si opera.

Possibili varianti Risulta immediata la necessità di realizzare una variante per ogni differente ambito operativo, se non per ogni azienda che adotti l'applicativo.

Suggerimenti relativi all'implementazione Essendo questo il cuore dell'applicazione conviene spingere verso la completa modularità del codice. Il fatto che una strutturazione *object oriented* male si combini con le richieste di velocità e semplicità proprie di un processo di ottimizzazione che può richiedere anche tempi lunghi, è mitigato, se non cancellato, dalle prestazioni delle procedure euristiche descritte nel capitolo 4.

La modularità necessaria a coprire le possibili varianti future, può essere garantita dall'adozione dei pattern di progettazione *Strategy* [GHJV94, p315] e *Template Method* [GHJV94, p325] presentati, rispettivamente, in figura 5.5 e 5.6.

Il pattern *Strategy* permette di dividere il contesto (`Context`) che gestisce i dati dalle strategie, o algoritmi, che li manipolano (`Strategy`) permettendo di modificare l'algoritmo, e di averne differenti implementazioni, senza dover alterare il contesto (`ConcreteStrategy`). *Template Method* permette di definire lo scheletro di un algoritmo (`AbstractClass`) lasciando che siano le sottoclassi a specificarne i singoli passi (`ConcreteClass`).

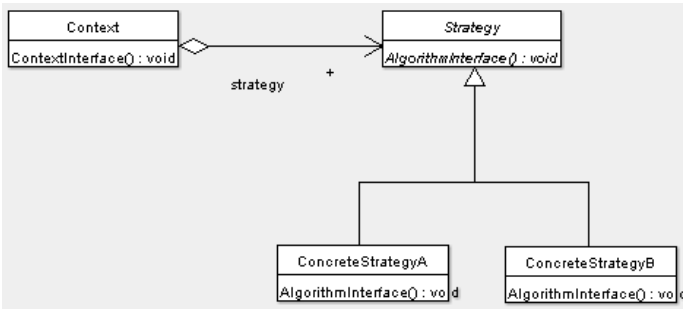

Fig. 5.5: Struttura dei componenti del pattern Strategy

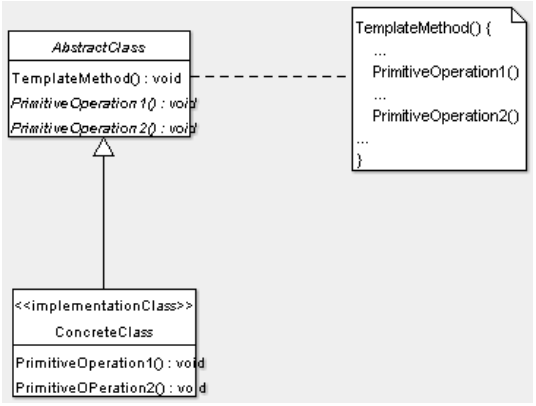

Fig. 5.6: Struttura dei componenti del pattern Template Method

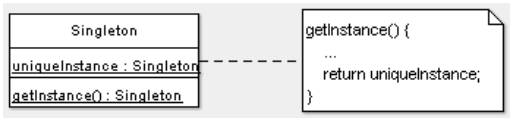

Fig. 5.7: Struttura dei componenti del pattern Singleton

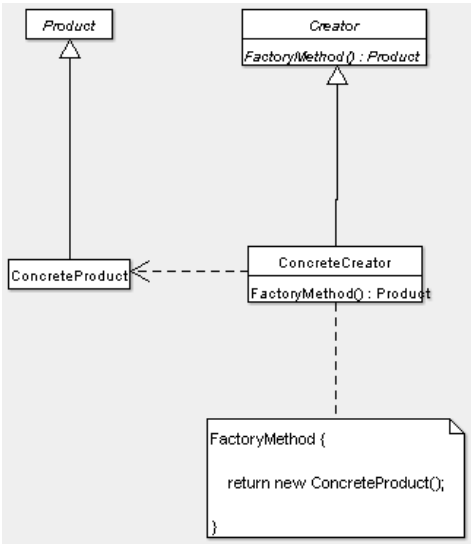

Fig. 5.8: Struttura dei componenti del pattern Factory Method

La richiesta di avere un'unica istanza del motore è realizzabile tramite il pattern *Singleton* [GHJV94, p127], che permette di definire un unico punto di creazione per un oggetto o componente (figura 5.7).

In ultimo la necessità di poter definire differenti versioni del motore a seconda della realtà nella quale viene utilizzata l'applicazione, può essere gestita tramite il pattern *Factory Method* [GHJV94, p107]. Questo pattern, descritto in figura 5.8, permette ad una classe di istanziare un oggetto (`Product`), sebbene non possa anticipare cosa debba istanziare. Per far ciò si può appoggiare ad una classe (`Creator`) nota a *compile time*, la quale delega ad una sua sottoclasse (`ConcreteCreator`), nota a *run time*, di istanziare l'oggetto corretto (`ConcreteProduct`).

Il connettore al database

Compito Il connettore ha il compito di rendere disponibile agli altri componenti i dati presenti in un database. Ha anche il compito di astrarre l'applicazione dal database sottostante, permettendo di utilizzare differenti *Data Base Management System* a seconda delle esigenze e delle dimensioni dell'applicazione quando si trova in produzione.

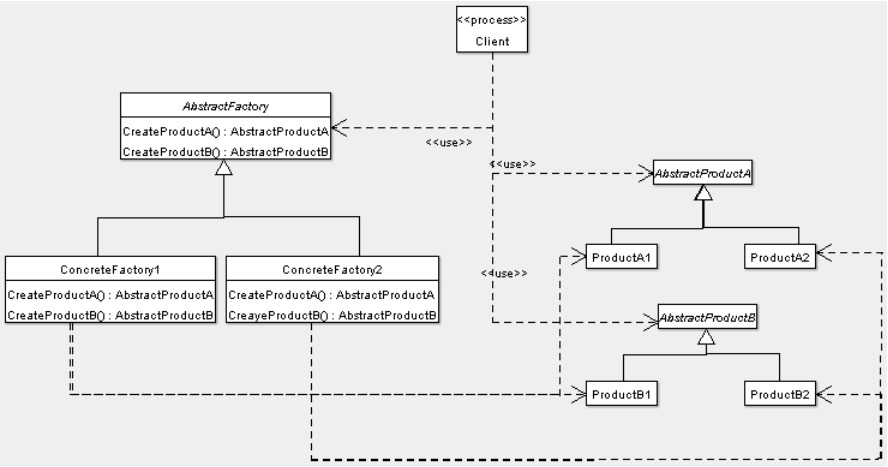

Fig. 5.9: Struttura dei componenti del pattern Abstract Factory

Istanze necessarie Generalmente è consigliabile utilizzare una sola istanza alla quale tutti i componenti accedono. In questo modo è possibile verificare il corretto utilizzo del connettore. Tra l'altro è possibile gestire in modo piú efficiente le connessioni con il database.

Possibili varianti Può essere utile definire una variante del connettore per ogni database utilizzabile in produzione.

Suggerimenti relativi all'implementazione Si può creare un unico punto d'accesso al database tramite il pattern *Singleton*. La necessità di gestire tipologie differenti di base dati, può essere facilmente risolta tramite l'utilizzo del pattern *Abstract Factory* [GHJV94, p87] descritto in figura 5.9. *Abstract Factory* permette di definire un'interfaccia (`AbstractFactory`) per creare una famiglia di oggetti (`AbstractProductA` e `AbstractProductB`) senza dover definire in anticipo le classi concrete. Per aggiungere un nuovo database basta quindi aggiungere una nuova famiglia di oggetti. La gestione efficiente delle connessioni al database può essere realizzata tramite l'utilizzo del pattern *Pooling* [KJ02], realizzando di fatto un Connection Pool. *Pooling*, presentato in figura 5.10, permette di ridurre il costo dell'acquisizione ed il rilascio di risorse (`Resource`) tramite il riciclo di queste. L'accesso alle risorse viene mediato da un gestore (`Pool`) che ha il compito di creare, riciclare e distruggere le risorse.

L'esportatore di soluzioni

Compito A seconda delle richieste è possibile dover esportare le soluzioni in vari formati. Si rende necessario creare un esportatore per ogni formato specifico.

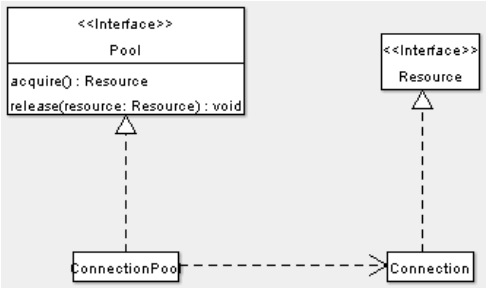

Fig. 5.10: Struttura dei componenti del pattern Pooling. Esempio applicato ad un connection pool generico

Istanze necessarie Le istanze dell'esportatore sono tipicamente usa e getta. Non si pone, quindi, il problema di definirne il numero massimo o minimo necessario all'applicazione.

Possibili varianti Sarà necessario realizzare una o piú varianti dell'oggetto per ogni formato gestito.

Suggerimenti relativi all'implementazione Anche in questo caso l'uso congiunto dei pattern *Strategy*, *Template Method* e *Abstract Factory* è indicato. Il primo pattern permette di separare il contesto dei dati dagli algoritmi che li manipolano. Il secondo pattern, suddividendo gli algoritmi in passi, permette di isolare le parti personalizzabili a seconda dell'azienda di destinazione. Il terzo pattern permette di aggiungere nuovi formati anche in un secondo tempo.

L'interfaccia grafica

Compito L'interfaccia ha il compito di permettere all'utente di accedere ai dati e ai metodi degli altri oggetti.

Istanze necessarie Tipicamente serve un'istanza per ogni utente che utilizza l'applicazione. Questa condizione può variare fortemente a seconda del tipo di applicativo client realizzato.

Possibili varianti La struttura modulare dovrebbe rendere facile la coesistenza di interfacce di differente natura.

Suggerimenti relativi all'implementazione Poichè l'implementazione è fortemente legata al mezzo e alla tecnologia utilizzata, è difficile definire, a questo livello, delle linee guida per la realizzazione dell'interfaccia grafica. A esempio librerie complesse, come le swing Java o le servlet, già sono organizzate per utilizzare dei pattern ben noti.

6. STRUTTURAZIONE DELLE CLASSI

> Al pannello comandi era affisso un biglietto. Il biglietto, sul quale era disegnata una freccia che indicava una delle manopole dei comandi, diceva: QUESTO è PROBABILMENTE IL BOTTONE MIGLIORE DA Ppremere.
>
> *Guida Galattica per gli Autostoppisti* - Douglas Adams

Sommario

In questo capitolo si definisce in dettaglio la struttura statica dell'applicazione. Vengono infatti elencate, tramite diagrammi UML, le principali classi che andranno realizzate, descrivendone anche il ruolo ed il comportamento nell'applicazione.

Dei quattro componenti descritti nel capitolo 5, ci soffermiamo a definire meglio i due principali, il motore per la schedulazione ed il connettore con il database. Questo non significa che gli altri due componenti siano meno importanti, ma essi risultano essere fortemente caratterizzati dalla azienda presso la quale l'applicazione va in produzione. Per i primi due componenti, invece, è addirittura possibile definire un framework, sia applicativo che strutturale, applicabile ad ogni realtà produttiva.

Nella descrizione dei componenti, il concetto di utente può essere inteso sia come una persona fisica (Operatore) sia come un altro componente del sistema (Processo Client).

6.1 Il motore per la schedulazione

In figura 6.1 è riportato lo schema delle classi che realizzano il motore per la schedulazione. Queste classi possono essere suddivise in due gruppi. Il primo gruppo rappresenta lo scheletro vero e proprio del motore, mentre il secondo definisce la personalizzazione del motore a seconda del campo d'applicazione.

6.1.1 Il ruolo delle classi

Lo scheletro del motore può esser visto come costituito da tre classi base:

1. `Engine`;
2. `Context`;

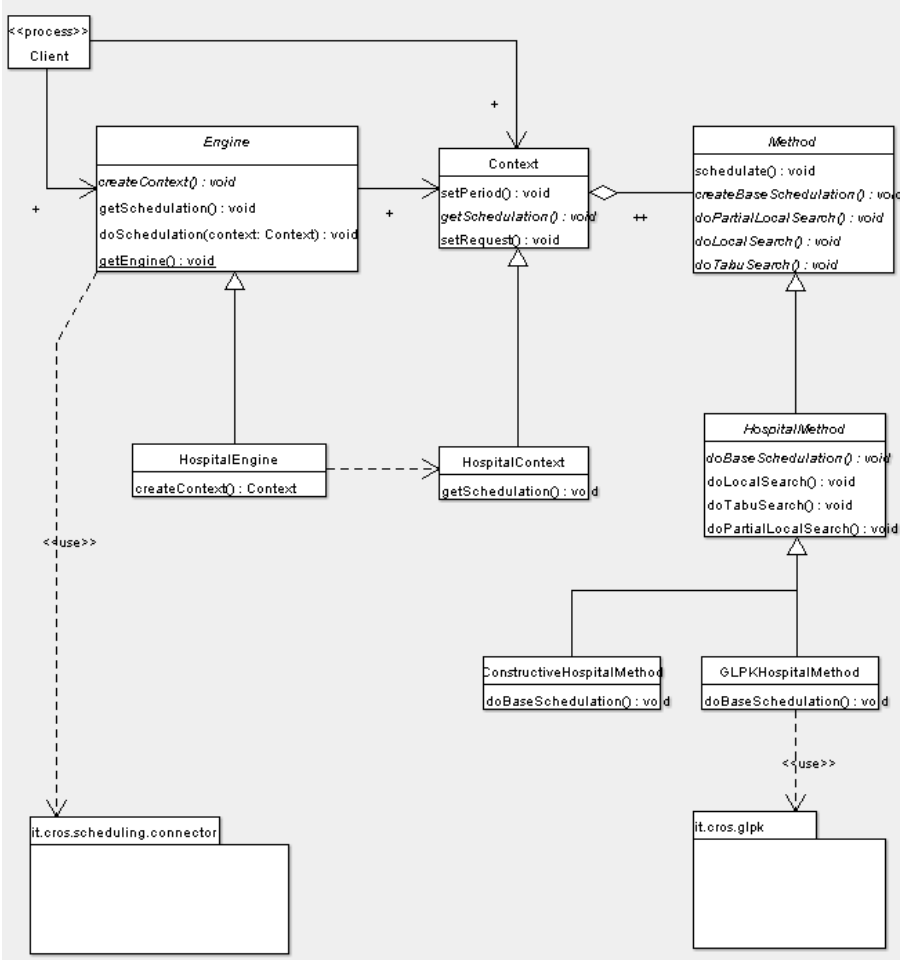

Fig. 6.1: Schema delle classi che compongono il motore per la schedulazione

3. `Method`.

`Engine` e `Context` sono direttamente utilizzate dall'utente che in questo caso è l'interfaccia grafica. `Engine` rappresenta la procedura mentre `Context` descrive una particolare istanza, o contesto, del problema. `Engine` mette a disposizione dell'utente i metodi per generare e modificare una soluzione e crea la versione iniziale di `Context`, valorizzandola con i dati in suo possesso. Tramite `Context` l'utente può modificare le richieste e richiedere ad `Engine` di generare una nuova soluzione. `Context` permette ad `Engine` di generare la soluzione tramite uno o piú metodi. Questi metodi sono realizzati da `Method`. `Method` suddivide il processo di creazione della soluzione in quattro fasi:

1. Costruzione della soluzione base;
2. Modifica di una soluzione parziale tramite ricerca locale;
3. Modifica di una soluzione completa tramite ricerca locale;
4. Modifica di una soluzione completa tramite tabu search.

Questi tre oggetti sono raggruppati in un unico *namespace* o *package* identificato come `it.cros.scheduling.engine`.

Le classi presentate in precedenza devono essere declinate secondo la realtà presso la quale viene applicata la soluzione. Ad esempio, possiamo introdurre il *namespace* `it.cros.nrp.engine`, che raggruppa tutte le classi concrete che determinano il reale comportamento del motore, per gestire il problema delle case di riposo che condividono la medesima organizzazione del lavoro di Ghedi. Abbiamo quindi le classi:

1. `HospitalEngine`;
2. `HospitalContext`;
3. `HospitalMethod`, ulteriormente specializzato dalle sottoclassi:
   (a) `ConstructiveHospitalMethod`;
   (b) `GLPKHospitalMethod`.

6.1.2 Interazione tra le classi

Nello schema presentato in figura 6.2 si evidenziano le interazioni tra gli oggetti che compongono la parte centrale del motore.

Il generico cliente del motore richiede ad `Engine` un istanza di `Context`. `Engine` provvede a crearne una e a valorizzarla con gli eventuali dati di default, successivamente la cede al cliente. Il cliente modifica `Context` secondo le sue necessità. Per effettuare una schedulazione, il cliente effettua una richiesta a `Engine`, il quale utilizza `Context`, precedentemente modificato dal cliente, per ottenere una soluzione. La soluzione viene fisicamente generata da `Method`.

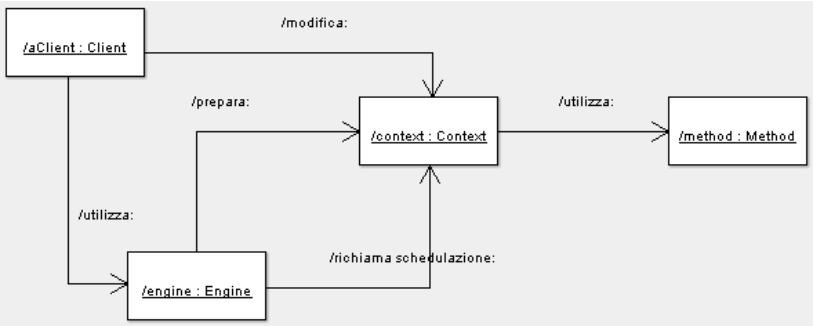

Fig. 6.2: Schema delle collaborazioni tra le classi che compongono il motore

6.1.3 Il rapporto tra le classi ed i pattern che realizzano

Engine realizza il pattern *Singleton* e il componente *Creator* del pattern *Factory Method*.

Context realizza i componenti *Product* e *Context* rispettivamente dei pattern *Factory Method* e *Strategy*.

Method realizza i componenti *Strategy* e *AbstractClass* rispettivamente dei pattern *Strategy* e *Template Method*.

HospitalEngine realizza il componente *ConcreteCreator* del pattern *Factory Method*.

HospitalContext realizza il componente *ConcreteProduct* del pattern *Factory Method*.

HospitalMethod, ConstructiveHospitalMethod, GLPKHospitalMethod realizzano i componenti *ConcreteStrategy* e *ConcreteClass* rispettivamente dei pattern *Strategy* e *Template Method*.

6.2 Il connettore al database

Le classi presentate nel diagramma in figura 6.3 definiscono un generico connettore al database. Queste devono essere declinate rispetto al database utilizzato. Tutte queste classi sono raggruppate in un unico *namespace* chiamato `it.cros.scheduling.connector`.

6.2.1 Il ruolo delle classi

Il connettore è composto dalle seguenti classi:

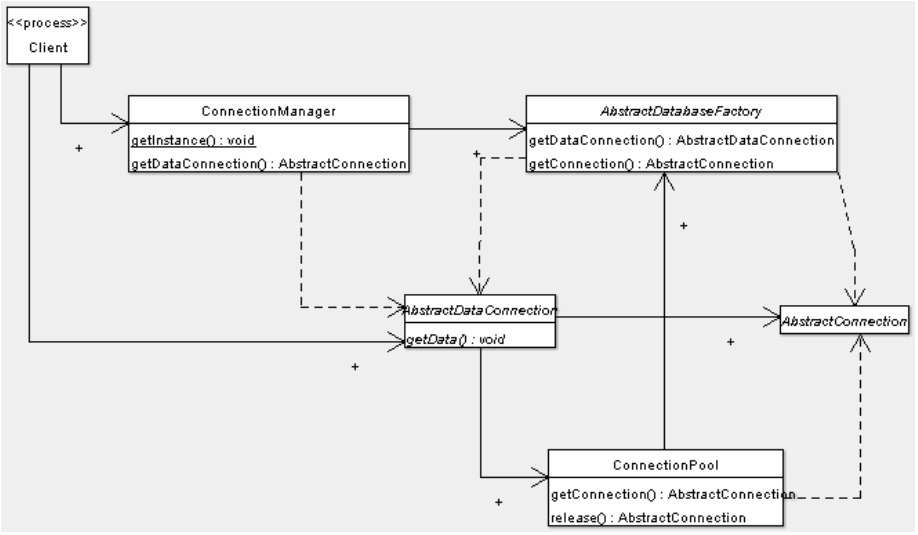

Fig. 6.3: Schema delle classi che compongono il connettore al database

1. `ConnectionManager`;
2. `AbstractDataBaseFactory`;
3. `AbstractDataConnection`;
4. `AbstractConnection`;
5. `ConnectionPool`.

Tramite `ConnectionManager` l'utente può avere accesso ad un'istanza di `AbstractDataConnection`. `AbstractDataConnection` ha il compito di gestire tutte le richieste che l'utente effettua al database. Per far ciò deve utilizzare `AbstractConnection`, unico vero collegamento fisico al database. Per ragioni di prestazione e di ottimizzazione delle risorse, la creazione e distruzione delle connessioni al database sono delegate al `ConnectionPool`. Prima di poter operare `AbstractDataConnection` deve richiedere in prestito una connessione a `ConnectionPool`, la quale dovrà restituire al termine delle operazioni. In ultimo, la classe `AbstractDataBaseFactory` ha il compito di creare `AbstractDataConnection` ed `AbstractConnection` su richiesta, rispettivamente, di `ConnectionManager` e `ConnectionPool`.

La personalizzazione rispetto ai differenti database si limita a specializzare le tre classi `AbstractDataBaseFactory`, `AbstractDataConnection` e, se le API del linguaggio non la prevedono, `AbstractConnection`.

6.2.2 Interazione tra le classi

Il cliente deve accede a `ConnectionManager` per poter ottenere un'istanza di `AbstractDataConnection`. Per poter eseguire la richiesta `ConnectionManager` richiede un'istanza di `AbstractDataConnection` a `AbstractDatabaseFactory`. A sua volta `AbstractDataConnection` richiede a `ConnectionPool` un'istanza di `AbstractConnection`. Al termine del suo ciclo vita, `AbstractDataConnection` restituisce l'istanza a `ConnectionPool`. `ConnectionPool` ottiene un'istanza di `AbstractConnection` direttamente da `AbstractDatabaseFactory`.

6.2.3 Il rapporto tra le classi ed i pattern che realizzano

ConnectionManager è implementato tramite un *Singleton* e riveste il ruolo di *client* nel pattern *Abstract Factory*.

AbstractDataBaseFactory realizza la classe *AbstractFactory* nel pattern omonimo.

AbstractDataConnection implementa *AbstractProductN* del pattern *Abstract-Factory*.

AbstractConnection oltre ad essere *AbstractProductN* nel pattern *Abstract-Factory*, implementa la classe *Resource* del pattern *Pooling*.

ConnectionPool è *ResourcePool* del pattern *Pooling* e riveste anche il ruolo di *client* nel pattern *Abstract Factory*.

7. DEFINIZIONE DELLA BASE DATI

This way she moves in the logic of all my dreams
Desert Rose - Sting

Sommario

In questo capitolo si descrive la base dati dell'applicazione. Tramite diagrammi di classe UML si individuano le tabelle necessarie e le operazioni ad esse relative.

La definizione di una base dati tramite il linguaggio standard SQL permette di definire lo schema generale del database in modo univoco, mantenendo la possibilità di realizzare il tutto con i principali database relazionali presenti in commercio[1].

Discorso a parte va fatto per le *stored procedure*. Una buona implementazione del connettore al database, presentato nei capitoli 5 e 6, potrebbe fare un massiccio uso di queste. Purtroppo non esiste uno standard a riguardo, ed ogni produttore è libero di definire il suo linguaggio, o anche di non definirlo affatto. È compito del connettore al database, descritto nella sezione 6.2, astrarre le richieste da effettuare al database, lasciando all'implementatore la scelta se utilizzare o meno le *stored procedure*.

Lo schema in figura 7.1 è uno sguardo d'insieme alla base dati. Le due tabelle principali sono *Operatori* e *Turni*. Attorno a queste sono costruite le tabelle relative la gestione del personale (*Ferie*, *Malattie* e *Orari*), quelle che descrivono le schedulazioni previste (*Schedulazioni* e *Assegnazioni*) ed i turni effettivamente svolti (*Cartellino*). A queste si aggiunge la tabella *Reparti* che si lega direttamente sia a *Operatori* che a *Turni*.

7.1 Descrizione delle tabelle

Prima di poter definire le tabelle principali dobbiamo descrivere tutte le tabelle minori o di supporto. Queste, infatti, completano il significato delle tabelle principali. Tutte le tabelle sono state decomposte in forma normale di Boyce-Codd [ACPT96]. Sono, quindi, state rimosse tutte le ridondanze nei dati e le anomalie strutturali e sono state evidenziate le dipendenze funzionali. Oltre a

[1] Possono esistere delle piccole discrepanze tra lo standard e le singole implementazioni del linguaggio di definizione delle tabelle, comunque circoscritte ad aspetti marginali. Per una lista esaustiva delle differenze tra i principali RDBMS, vedi [KK01].

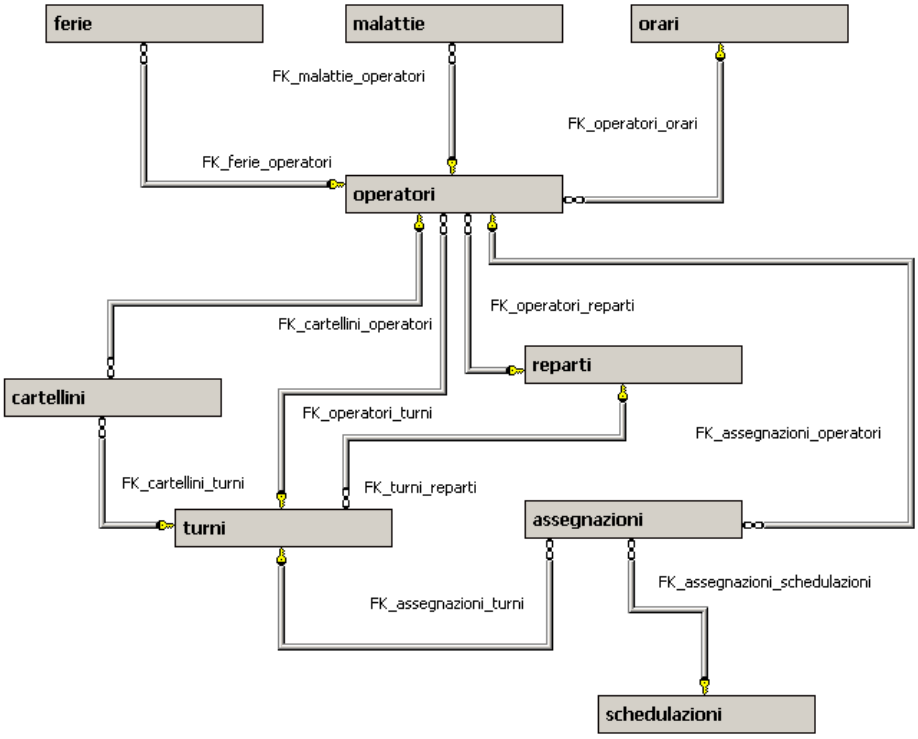

Fig. 7.1: Schema semplificato del database

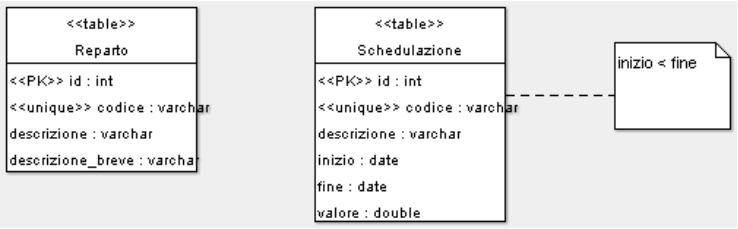

Fig. 7.2: Struttura delle tabelle Reparti e Schedulazioni

questo è stata definita, per ogni tabella, una chiave che non dipende dai valori della singola tupla.

7.1.1 Reparti e Schedulazioni

Le tabelle *Reparti* e *Schedulazioni*, presentate in figura 7.2, sono tabelle molto semplici e non dipendono da nessun'altra tabella.

Reparti

La tabella *Reparti* è utilizzata per classificare gli operatori e i turni a seconda del reparto d'appartenenza. Di fatto ogni tupla della tabella modella un reparto. Per ogni reparto è definito, oltre ad una chiave primaria (PK) definita da un valore progressivo, un codice univoco e due descrizioni, una lunga ed una breve. La descrizione lunga è il nome vero e proprio, mentre quella breve può essere utilizzata, ogni qual volta sia necessario ridurre lo spazio occupato.

Schedulazioni

Schedulazioni contiene l'intestazione di tutte le schedulazioni effettuate dall'applicazione. Per ogni schedulazione è registrato un codice univoco ed una descrizione, oltre che il periodo di riferimento (inizio e fine) e il costo della schedulazione stessa.

7.1.2 Turni

In figura 7.3 è riportato lo schema della tabella *Turni*. Ogni tupla è caratterizzata da una chiave, da un codice univoco e da una descrizione del turno. L'appartenenza ad un reparto particolare è identificata da una chiave esterna (FK). Per ogni turno sono indicati l'ora d'inizio e di fine del turno e l'effettiva durata di questo. In ultimo esiste la possibilità di indicare il numero di persone che devono effettuare il turno ogni giorno.

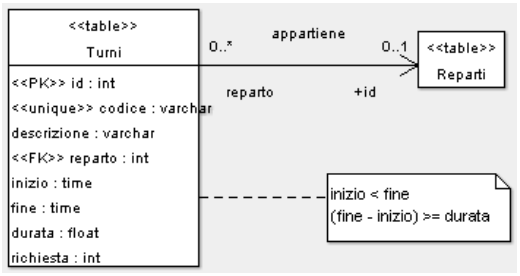

Fig. 7.3: Struttura della tabella Turni

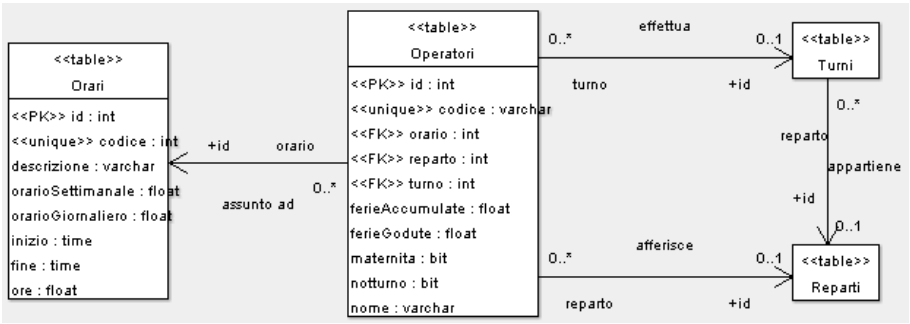

Fig. 7.4: Struttura delle tabelle Operatori e Orari

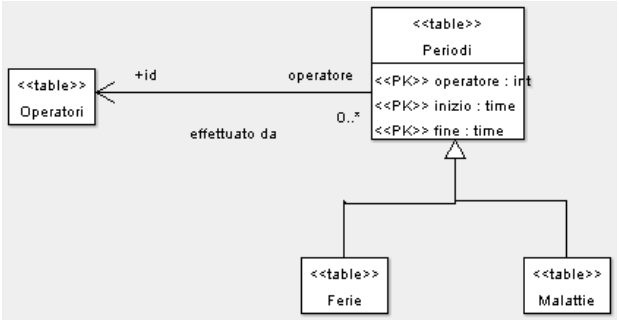

Fig. 7.5: Struttura delle tabelle Ferie e Malattie

7.1.3 Orari e Operatori

Le tabelle che rappresentano gli operatori e il loro contratto d'assunzione, o orario, sono rappresentati in figura 7.4.

Orari

L'orario descrive il contratto tra l'operatore e la casa di riposo. Sono infatti riportati il numero di ore che l'operatore deve effettuare in una settimana e in un giorno medio. A questo si affiancano gli orari di inizio e di fine disponibilità per ogni giorno della settimana oltre il numero di ore che l'operatore dovrebbe effettuare in tale giorno. Il giorno di riposo stabilito per l'operatore è indicato dal fatto che la richiesta di ore da effettuare è pari a zero.

Operatori

Per ogni operatore sono definiti una chiave primaria, un codice univoco ed il nome dell'operatore. A queste informazioni sono aggiunti i riferimenti all'orario con il quale l'operatore è assunto, al reparto al quale afferisce e l'eventuale turno preferenziale. Sono anche registrati il numero di giorni di ferie accumulate dall'inizio dell'anno ad oggi e di quelli già goduti. In ultimo, sono indicate le informazioni in merito al fatto che l'operatore sia in maternità o che possa effettuare lavoro notturno.

7.1.4 Ferie e Malattie

Le tabelle che rappresentano le ferie e le malattie sono praticamente identiche. Possono infatti essere considerate come due estensioni di una tabella *Periodi* (figura 7.5) che descrive un generico periodo di astensione dal lavoro. Di fatto le due tabelle *Ferie* e *Malattie* cambiano solamente il significato della tabella *Periodi*. Fisicamente le due tabelle sono realizzate in modo distinto e non esiste la tabella *Periodi*. Il periodo è associato ad un operatore, ed è caratterizzato da un inizio e da una fine.

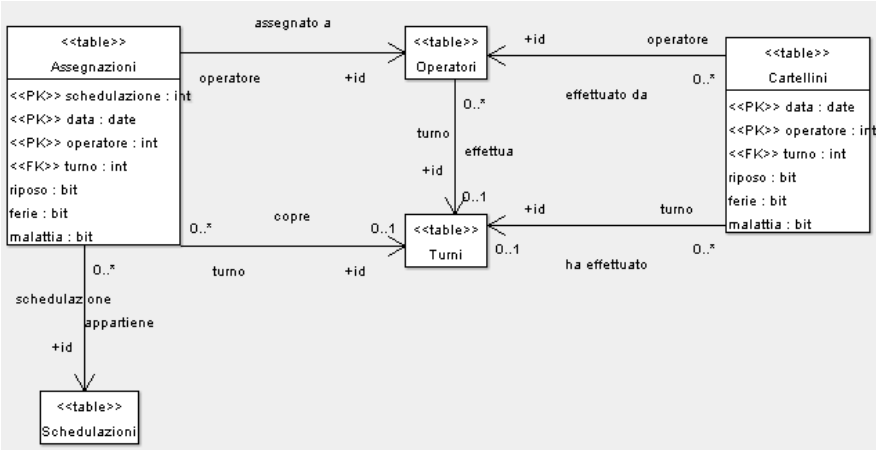

Fig. 7.6: Struttura delle tabelle Assegnazioni e Cartellini

7.1.5 Cartellini e Assegnazioni

In figura 7.6 sono riportati gli schemi delle tabelle *Cartellini*, che rappresenta i turni effettivamente svolti da un operatore, e *Assegnazioni* che riporta i turni assegnati da una schedulazione.

Cartellini

Ogni tupla rappresenta una coppia giorno-operatore. Per ogni copia è indicato se è stato effettuato un turno o se l'operatore era in riposo, ferie o malattia. Le quattro informazioni sono tra di loro incompatibili.

Assegnazioni

La tabella *Assegnazioni* è pressochè identica a *Cartellini*. L'unica differenza consiste nel fatto che ad ogni assegnazione è associata alla schedulazione dalla quale è stata generata.

8. CONCLUSIONI

Questo lavoro affronta il problema di ottimizzare i turni di lavoro per gli operatori della casa di riposo di Ghedi. Nella prima fase si è definito un modello matematico del problema stesso a partire dalla normativa del lavoro, dai CCNL e dalle esigenze organizzative della casa di riposo di Ghedi. Questo modello rispetta pienamente tutti i vincoli di legge e di contratto e, dove possibile, garantisce l'elasticità necessaria per svolgere al meglio i compiti assegnati a FT Service, tenendo conto il piú possibile di una serie di vincoli organizzativi (struttura ciclica dei turni, modello 3+1, assegnazione ai reparti, preferenze,...), di ulteriori vincoli sociali non di primaria importanza, ma dei quali si potrebbe tener conto ed è stata impostata la definizione dei parametri con cui le diverse funzioni obiettivo possono essere ricondotto ad una sola. Queste aggiunte possono essere definite solo successivamente alla fase di test che è prevista prossima mente.

Nella definizione di un modello matematico GLPK si è dimostrato un valido strumento, permettendo di realizzare per approssimazioni successive il modello che rappresenta al meglio la realtà del mondo del lavoro italiano e le particolarità della casa di riposo di Ghedi. L'utilizzo di GLPK ha anche suggerito alcune disuguaglianze che potrebbero essere introdotte in future versioni del modello per rendere piú rapida la sua risoluzione. In ultimo, il comportamento del solutore lineare ha ulteriormente ribadito la proficuità di investire tempo nella ricerca di una formulazione piú forte piuttosto che inseguire il progresso tecnologico.

L'introduzione degli algoritmi euristici si è rivelata un valido mezzo per definire una soluzione software applicabile in qualsiasi contesto lavorativo. Essi infatti non richiedono nè grosse capacità di calcolo nè ingenti investimenti in licenze software, e al tempo stesso generano una soluzione piú che accettabile in tempi ridotti.

L'analisi dell'applicazione ha permesso di definire un framework aperto, che si può adattare sia ad altre realtà organizzative sia alla futura introduzione di algoritmi euristici con migliori prestazioni. Il framework rende molto semplice l'utilizzo dell'applicazione da parte dell'utente, permettendo di realizzare differenti interfacce grafiche, formati d'esportazione e metodi di soluzione. Il framework risulta essere anche flessibile ben adattandosi a realtà differenti, permettendo di variare facilmente gli strumenti utilizzati, quali i database. Il massimo della flessibilità lo si ottiene nel fatto che il framework permette ad strumento d'ottimizzazione di trasformarsi in piattaforma di simulazione per verificare il comportamento di differenti turnazioni nella gestione di eventi im-

previsti. Questo framework apre la strada ad innumerevoli sviluppi, quali ad esempio, l'introduzione di metodi di *Branch & Cut* o la realizzazione di euristiche parallele quali, ad esempio, *Ant Colony System* o algoritmi quantistici.

La realizzazione pratica di queste estensioni future è piuttosto probabile considerato l'interesse che l'applicazione, allo stato attuale di prototipo, ha suscitato anche in ambiti differenti rispetto alla casa di riposo. A breve partirà infatti lo sviluppo di una versione speciale dedicata alla catena di negozi US Fashion Store.

RINGRAZIAMENTI

I miei ringraziamenti vanno principalmente a Sara e alla mia famiglia per la pazienza dimostrata in questi otto anni. E poi, in ordine quasi sparso, ai professori Cordone e Righini; a Matteo, Alberto e gli altri membri del laboratorio OptLab. Difficile non ricordare i vari membri del LAE e affini: Michele, Mastro, Stefano, Massimo, Samuele, Simone, Bone ed i lontani Boss e Raffaele. Un saluto va anche ai "colleghi" Mauro e Davide, e alla ex MDC del polo Paola.

Riferimenti bibliografici

[ACCU03] AGIDAE, FP CGIL, FISASCAT CISL e UILTuCS UIL: *Ipotesi di CCNL 2002-2005*. Verbale Incontro, Giugno 2003.

[ACPT96] Azteni, Paolo, Stefano Ceri, Stefano Paraboschi e Riccardo Torlone: *Basi di dati. Concetti, linguaggi e architetture*. McGraw-Hill, Milano, First edizione, Oct 1996.

[AD01] Aickelin, U. e K. Dowsland: *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem*. Journal of Scheduling, 2001.

[aH01] Hofe, H. Meyer auf'm: *Constraint Programming and Large Scale Discrete Optimization*, volume 57 della serie *DIMACS Series in Discrete Mathematics and Teorethical Computer Science*, capitolo Nurse rostering as constraint satisfaction with fuzzy constraints and inferred control strategies, pagine 67–99. E. C. Freuder and R. J. Wallace, 2001.

[AMP] AMPL: *AMPL® A Modeling Language for Mathematical Programming*. http://www.ampl.com/.

[AW04] Aickelin, Uwe e Paul White: *Building Better Nurse Scheduling Algorithms*. Annals of Operations Research, 128:159–177, 2004.

[BC97] Burke, Edmund K. e Michael W. Carter (curatori): *PATAT 1997 Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling*, August 1997. `http://www.asap.cs.nott.ac.uk/patat/patat97/`.

[BC02] Burke, Edmund K. e Patrick De Causmaecker (curatori): *PATAT 2002 Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, KaHo Sint-Lieven, Gent, August 2002. ISBN 90-806096-1-7.

[BCB01] Burke, E., P. D. Causmaecker e G. V. Berghe: *A memetic approach to the nurse rostering problem*. Applied Intelligence, 15:199–214, 2001.

[BCP01] Burke, Edmund, Patrick De Causmaecker e Sanja Petrovic: *Variable Neighbourhood Search for Nurse Rostering Problems*. MIC 2001 - 4th Metaheuristics International Conference, July 2001. `citeseer.ist.psu.edu/682034.html`, Porto, Portugal, July 16-20, 2001.

[BCPR95] Burke, Edmund K., Dave Corne, Ben Paechter e Peter Ross (curatori): *PATAT 1995 Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling*, 1995. `http://www.asap.cs.nott.ac.uk/patat/patat95`.

[BE00] Burke, Edmund K. e Wilhelm Erben (curatori): *PATAT 2000 Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling*, 2000. `http://www.asap.cs.nott.ac.uk/patat/patat00/`.

[BRJ98] Booch, Grady, James Rumbaugh e Ivar Jacobson: *The Unified Modeling Language User Guide*. Addison-Wesley Object Technology Series. Addison-Wesley, Reading, Massachusetts, 1998. 4th Printing April 1999.

[BT04] Burke, Edmund K. e Michael Trick (curatori): *PATAT 2004 Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, Pittsburg USA, August 2004. `http://www.asap.cs.nott.ac.uk/patat/patat04`.

[CB02] Causmaecker, Patrick De e Greet Vanden Berghe: *Relaxation of Coverage Constraints in Hospital Personnel Rostering*. Nel *PATAT 2002 4th international conference on the Practice and Theory of Automated Timetabling*, Gebr. Desmetstraat 1, 9000 Gent, Belgium, 2002.

[CLLR03] Cheang, B., H. Li, A. Lim e B. Rodrigues: *Nurse rostering problem - a bibliographic survey*. European Journal of Operational Research, 151:447–460, 2003.

[EJK+04] Ernst, A.T., H. Jiang, M. Krishnamoorthy, B. Owens e D. Sier: *An Annotated Bibliography of Personnel Scheduling and Rostering*. Annals of Operations Research, 127:21–144, 2004.

[GHJV94] Gamma, Erich, Richard Helm, Ralph Johnson e John Vlissides: *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley professional computing series. Addison-Wesley, Reading, Massachusetts, 1994. 18th Printing September 1999.

[GL97] Glover, F. e M. Laguna: *Tabu Search*. Kluwer Academic Publishers, 1997.

[GNU04a] GNU Linear Programming Kit: *Modeling Language GNU MathProg*, Draft edizione, Jan 2004. Version 4.4.

[GNU04b] GNU Linear Programming Kit: *Reference Manual*, Draft edizione, Jan 2004. Version 4.4.

[Hun95] Hung, R.: *Hospital Nurse scheduling*. Journal of Nursing Administration, 7/8(25):21–23, 1995.

[HW96] Heus, K. e G. Weil: *Constraint programming a nurse scheduling application*. Nel *Proceedings of the Second International Conference on the Practical Application of Constraint Technology*, pagine 115–127, 1996.

[ILO] ILOG: *ILOG CPLEX: High-performance software for mathematical programming and optimization.* http://www.ilog.com/products/cplex/.

[Ita03] Italiano, Governo: *Attuazione delle direttive 93/104/CE e 2000/34/CE concernenti taluni aspetti dell'organizzazione dell'orario di lavoro.* Gazzetta Ufficiale N. 87 del 14 Aprile 2003, Aprile 2003. Decreto Legislativo 8 aprile 2003, n.66.

[Ita04] Italiano, Governo: *Modifiche ed integrazioni al decreto legislativo 8 aprile 2003, n. 66, in materia di apparato sanzionatorio dell'orario di lavoro.* Gazzetta Ufficiale n. 192 del 17 agosto 2004, Luglio 2004. Decreto Legislativo 19 luglio 2004, n.213.

[JK92] Jelinek, R. C. e J. A. Kavois: *Nurse staffing and scheduling: past solutions and future directions.* Journal of the Society for Health System, (3), 1992.

[JSV98] Jaumard, B., F. Semet e T. Vovor: *A generalized linear programming model for nurse scheduling.* European Journal of Operational Research, pagine 1–18, 1998.

[KJ02] Kircher, Michael e Prashant Jain: *Pooling.* Rapporto Tecnico, Corporate Technology, Siemens AG, Munich, Germany, 2002.

[KK01] Kline, Kevin e Kevin Kline: *SQL Guida di riferimento.* O'REALLY. Apogeo, Milano, 2001. SLQ in a nutshell.

[MK92] Millar, H. e M. Kiragu: *Cyclic and non-cyclic scheduling of 12h shift nurses by network programming.* European Journal of Operational Research, 3(104):582–592, 1992.

[MWG76] Miller, H. E., P. William e J. R. Gustave: *Nurse scheduling using mathematical programming.* Operations Research, 5(24):857–870, 1976.

[Pla01] Plane: *Soft constraints in the Nurse Rostering Problem, Full description of all the constraint types in use in Plane System*, Feb 2001. `http://www.cs.nott.ac.uk/~gvb/constraints.ps`.

[SW77] Smith, L. D. e A. Wiggins: *A computer-based nurse scheduling system.* Computers and Operations Research, (4):195–212, 1977.

[UCCU01] UNEBA, FISASCAT CISL, Funzione Pubblica CGIL e UILTuCS UIL: *CONTRATTO INTEGRATIVO COLLETTIVO REGIONALE DI LAVORO Per il Personale dipendente dalle realtà del settore: Assistenza, Sociale, Socio-Sanitario, Educativo, nonché da tutte le altre Istituzioni di Assistenza e Beneficenza. UNEBA. 2001-2003*, Settembre 2001. Testo ufficiale.

[Wol98] Wolsey, Laurence A.: *Integer Programming.* Wiley-Interscience, Toronto, First edizione, 1998.