POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

CONTROLLO DI UN GRUPPO DI ASCENSORI TRAMITE APPRENDIMENTO PER RINFORZO CON SELEZIONE DELLE VARIABILI

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Marcello Restelli
Correlatore: Dott. Matteo Pirotta

Tesi di Laurea di:
Alberto Lucini Paioni, matricola 782216

Anno Accademico 2012-2013

Sommario

Chiunque abbia frequentato un edificio con numerosi piani e popolato da un elevato numero di persone ha certamente usufruito dei servizi di un gruppo di ascensori e, probabilmente, ha avuto occasione di lamentarsi della scarsa qualità del servizio offertogli, ad esempio a causa di tempi di attesa eccessivamente lunghi. Il controllo di un gruppo di ascensori, infatti, è un compito molto complesso, in quanto il controllore deve tener conto di un altissimo numero di variabili e soddisfare allo stesso tempo esigenze differenti, anche contrastanti tra loro. Per raggiungere questo obiettivo il controllore ha a disposizione una quantità di dati che è allo stesso tempo elevata ma non sufficiente a inquadrare lo stato completo del sistema, introducendo così una certa dose di incertezza.

Lo scopo di questa tesi è quello di esplorare l'area dell'apprendimento per rinforzo per la progettazione di un controllore di gruppo di ascensori. L'obiettivo principale che ci proponiamo è quello di progettare un controllore che faccia sperimentare attese brevi agli utenti del sistema e che allo stesso tempo sia in grado di adattarsi a possibili cambiamenti del sistema. A tal proposito proponiamo una nuova architettura gerarchica, studiata per semplificare il controllo, e la utilizziamo per l'apprendimento sia tramite metodi di apprendimento offline (FQI), sia tramite metodi gradiente (RLG). Per contrastare le difficoltà dovute all'alta dimensionalità del problema, inoltre, proponiamo due nuovi algoritmi di feature selection: l'algoritmo garantito, che fornisce delle garanzie sulla bontà della politica apprendibile con l'insieme ridotto delle feature, e l'algoritmo euristico, che crea un albero delle dipendenze tra le varie feature.

La tesi mostra che FQI necessita di un numero troppo elevato di dati per poter apprendere una politica buona, mentre RLG riesce a ottenere delle prestazioni discrete garantendo allo stesso tempo un controllore adattativo. L'algoritmo di selezione delle feature euristico, inoltre, si dimostra essere efficace, mentre quello garantito non è risultato essere utilizzabile nella pratica a causa di tempi computazionali eccessivamente lunghi.

Abstract

People who spent some time in high and populated buildings has surely benefited from the services of elevator groups, probably complaining for poor quality of service, such as long waiting time. As a matter of fact, elevator groups control is a very complex task, due to the high number of control variables and different, contrasting goals to achive. For its purpose the controller must take into account a large number of information, but not all the relevant ones can be obtained, thus introducing a certain amount of randomness.

The aim of this thesis is to explore the area of reinforcement learning to design an elevator group control system. Our main goal is to propose a self-tuning controller which guaranties short waiting times and is able to react to changes in the environment. For this purpose we propose a new hierarchical architecture, designed in order to simplify the duty of the controller; on top of this architecture we project learning agents, both with Fitted Q Iteration (FQI) and gradient methods (RLG). In order to face the complexity due to the high dimensionality of the problem we propose also two new feature selection algorithms, one which provide some guarantees for the optimal policy of the reduced model, the other that generates the feature dependency tree.

We show that FQI controller can't achieve very good performance due to the huge amount of data that it would need, while RLG controller can achive quite good performance and it is a reliable adaptive controller. Besides this, we show that the guaranteed feature selection algorithm can't be used in real application because of computational time, while the one that builds the feature dependency tree is effective.

Ringraziamenti

Ringrazio il professor Restelli e il dottor Pirotta per avermi seguito e aiutato nello sviluppo e nella redazione di questa tesi. Un grazie anche a tutti gli altri docenti che mi hanno permesso di arrivare fino a questo punto.

Ringrazio Nicola e Simone, con cui ho condiviso parte del percorso di questa tesi.

Ringrazio Claudio, Alessandro e Riccardo, assieme a cui ho vissuto l'intera esperienza della laurea magistrale, con un pensiero a tutti gli altri colleghi incontrati durante questi due anni. Ringrazio anche tutti i colleghi di Cremona, con cui ho passato i tre anni di laurea triennale.

Ringrazio tutti i miei amici, con cui ho passato momenti fantastici e mi sono sempre stati vicini, seppur dall'esterno, durante questo percorso; scusate se non vi nomino personalmente, ma siete veramente tanti!

Ringrazio i miei genitori, che mi hanno sempre sostenuto e mi hanno permesso di arrivare tranquillamente a questa laurea. Un grazie anche a tutti i miei parenti.

Ringrazio infine tutte le persone che sono entrate nella mia vita, una parte seppur minima di questo percorso è anche merito loro.

Indice

Sommario **III**

Abstract **V**

Ringraziamenti **VII**

Indice **IX**

Elenco delle figure **XI**

Elenco delle tabelle **XV**

1 Introduzione **1**
- 1.1 Obiettivi . . . 2
- 1.2 Metodologia . . . 2
- 1.3 Contributi . . . 3
- 1.4 Struttura della tesi . . . 4

2 Stato dell'arte **7**
- 2.1 Configurazioni . . . 8
- 2.2 Criteri da ottimizzare . . . 11
- 2.3 Tipi di traffico . . . 12
- 2.4 Tecniche di controllo . . . 14
  - 2.4.1 Logica Fuzzy . . . 15
  - 2.4.2 Reti Neurali . . . 16
  - 2.4.3 Algoritmi Genetici e Reti di Programmazione Genetica . . . 17
  - 2.4.4 Apprendimento per Rinforzo . . . 18
  - 2.4.5 Altre tecniche . . . 20
- 2.5 Formalismo . . . 20
  - 2.5.1 Riduzione dell'MDP . . . 21
  - 2.5.2 MDP fattorizzati . . . 26
  - 2.5.3 Selezione delle Feature . . . 27

3 Selezione delle feature **31**
3.1 Trasformazione dell'MDP . . . 32
3.1.1 Definizione del modello ridotto . . . 33
3.1.2 Definizione della riespansione del modello ridotto . . . 33
3.1.3 Proprietà . . . 36
3.2 Problema di esempio . . . 38
3.3 Algoritmo di feature selection garantito . . . 40
3.3.1 Cifra di merito . . . 42
3.3.2 Considerazioni . . . 43
3.4 Algoritmo di feature selection euristico . . . 44
3.4.1 Algoritmo base . . . 45
3.4.2 Algoritmo di predizione della feature . . . 46
3.4.3 Costruzione del modello e classificazione delle variabili . . . 47
3.4.4 Considerazioni . . . 48
3.5 Utilità di una variabile in base alla sua controllabilità . . . 50

4 Configurazione degli esperimenti **55**
4.1 Struttura dello stato, dell'azione e del rinforzo . . . 58
4.2 Ambiente . . . 61
4.2.1 Messaggi riconosciuti dall'Ambiente . . . 63
4.3 Agente . . . 66
4.3.1 Messaggi riconosciuti dall'Agente . . . 68
4.4 Esperimento . . . 70
4.5 Configurazione utilizzata per gli esperimenti . . . 70

5 Risultati Sperimentali **73**
5.1 Controllori benchmark . . . 74
5.2 Architetture dei controllori proposti . . . 78
5.2.1 Architettura piatta . . . 78
5.2.2 Architettura gerarchica . . . 81
5.3 Controllore FQI . . . 95
5.3.1 Architettura piatta . . . 95
5.3.2 Architettura gerarchica . . . 97
5.4 Controllore RLG . . . 100
5.4.1 Architettura piatta . . . 101
5.4.2 Architettura gerarchica . . . 102

6 Conclusioni e sviluppi futuri **149**

Bibliografia **153**

Elenco delle figure

2.1 Andamento tipico del traffico in un edificio durante una giornata lavorativa. 13

3.1 Problema di esempio. 39
3.2 Albero delle dipendenze delle feature generato dall'algoritmo di riduzione delle feature euristico per il problema d'esempio. L'ordine da sinistra a destra dei nodi indica l'ordine di selezione delle variabili, mentre i numeri riportati sugli archi indicano il valore di R^2 ottenuto con la riduzione che comprende tutte le variabili selezionate fino a quel punto. 49
3.3 Alberi di dipendenza delle variabili, semplici ma generali, nei tre casi significativi, cioè con distribuzione stazionaria non controllabile, rinforzo non controllabile e entrambi controllabili. Nello schema con R si è indicato il rinforzo, an una variabile di azione, sn una variabile di stato. 51
3.4 Albero di dipendenza delle variabili, semplice ma generale, nel caso siano presenti dei disturbi. Nello schema con R si è indicato il rinforzo, an una variabile di azione, sn una variabile di stato, wn una variabile di disturbo. 54

4.1 Struttura del progetto interfacciato tramite RL-Glue. 57
4.2 Schema della struttura generale dell'Ambiente. Il diagramma contiene solamente gli elementi utili per illustrare questa struttura, mentre i dettagli implementativi sono stati omessi per semplicità e chiarezza. 62
4.3 Visualizzazioni grafiche permesse dal simulatore. 64
4.4 Schema della struttura generale dell'Agente. Il diagramma contiene solamente gli elementi utili per illustrare questa struttura, mentre i dettagli implementativi sono stati omessi per semplicità e chiarezza. 67

5.1 Tempo di attesa medio, in secondi, ottenuto dai controllori benchmark con traffico leggero. 75

5.2 Tempo di attesa medio, in secondi, ottenuto dai controllori benchmark con traffico intenso. 76
5.3 Feature selezionate dal metodo basato su alberi per l'architettura piatta. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte. 82
5.4 Feature selezionate dal metodo basato su alberi al livello con ascensori pieni dell'architettura gerarchica. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte. 89
5.5 Feature selezionate dal metodo basato su alberi al livello con ascensori vuoti dell'architettura gerarchica. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte. 93
5.6 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo uppeak, al variare di α; l'insieme di feature utilizzato è quello completo. 104
5.7 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak, al variare di α; l'insieme di feature utilizzato è quello completo. 105
5.8 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor, al variare di α; l'insieme di feature utilizzato è quello completo. 106
5.9 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo. 107
5.10 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo. 108
5.11 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo uppeak; l'insieme di feature utilizzato è quello completo. 113
5.12 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello completo. 114

5.13 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello completo. 115
5.14 Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello completo. 116
5.15 Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello completo. 117
5.16 Confronto dei risultati ottenuti con i diversi schemi di traffico dal nostro controllore rispetto a quelli benchmark, con traffico leggero. . 118
5.17 Confronto dei risultati ottenuti con i diversi schemi di traffico dal nostro controllore rispetto a quelli benchmark, con traffico intenso. . 119
5.18 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo. 122
5.19 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo. I grafici mostrano una vista ingrandita sui valori una volta raggiunta una situazione vicina alla convergenza. 123
5.20 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo. 124
5.21 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo. 125
5.22 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo. 126
5.23 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo uppeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 129
5.24 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 130
5.25 Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 131
5.26 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 132

5.27 Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 133
5.28 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo uppeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 136
5.29 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 137
5.30 Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 138
5.31 Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 139
5.32 Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi. 140
5.33 Andamento dell'apprendimento del rinforzo con traffico leggero di diverse tipologie, al variare dell'insieme delle feature utilizzato. . . . 143
5.34 Andamento dell'apprendimento del rinforzo con traffico intenso di diverse tipologie, al variare dell'insieme delle feature utilizzato. . . . 144

Elenco delle tabelle

3.1 Evoluzione delle variabili t e d nel problema di esempio. 39

4.1 Configurazione del sistema utilizzata negli esperimenti. 71

5.1 Tempo di attesa medio e massimo, in secondi, ottenuto dai controllori benchmark con traffico leggero. 74
5.2 Tempo di attesa medio e massimo, in secondi, ottenuto dai controllori benchmark con traffico intenso. 75
5.3 Risultati ottenuti dai controllori benchmark con traffico leggero. . . . 77
5.4 Risultati ottenuti dai controllori benchmark con traffico intenso. . . . 77
5.5 Risultati qualitativi ottenuti con il controllore FQI con architettura piatta, al variare delle feature utilizzate e del traffico generato nel sistema. È riportato il trend dei risultati al variare del numero di iterazioni dell'algoritmo FQI effettuate. La dicitura NO indica che il controllore non è riuscito a servire i passeggeri presenti nel sistema. . 96
5.6 Tempo di attesa medio e massimo, in secondi, ottenuti in condizioni di traffico leggero con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema. 97
5.7 Risultati ottenuti in condizioni di traffico leggero con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema. 97
5.8 Tempo di attesa medio e massimo, in secondi, ottenuti in condizioni di traffico intenso con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema. 98
5.9 Risultati ottenuti in condizioni di traffico intenso con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema. 98
5.10 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature con $\alpha = 0,1$ in situazione di traffico leggero. 109

5.11 Risultati ottenuti dal controllore definito a mano utilizzando l'insieme completo delle feature in situazione di traffico leggero. 112
5.12 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico intenso. . . . 118
5.13 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico intenso per controllare il sistema con traffico leggero. 120
5.14 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico leggero per controllare il sistema con traffico intenso. 120
5.15 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature e effettuando un numero elevato di simulazioni in situazione di traffico leggero. 127
5.16 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto attraverso il metodo basato su alberi in situazione di traffico leggero. 134
5.17 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto attraverso il metodo basato su alberi in situazione di traffico intenso. 141
5.18 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature completo in situazione di traffico leggero di tutte le tipologie. 145
5.19 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature completo in situazione di traffico intenso di tutte le tipologie. 145
5.20 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto in situazione di traffico leggero di tutte le tipologie. 146
5.21 Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto in situazione di traffico intenso di tutte le tipologie. 146

Capitolo 1

Introduzione

Chiunque abbia frequentato un edificio con numerosi piani e popolato da un elevato numero di persone ha certamente usufruito dei servizi di un gruppo di ascensori e, probabilmente, ha avuto occasione di lamentarsi della scarsa qualità del servizio offertogli, ad esempio a causa di tempi di attesa eccessivamente lunghi. Il controllo di un gruppo di ascensori, infatti, è un compito molto complesso, in quanto il controllore deve tener conto di un altissimo numero di variabili e soddisfare allo stesso tempo esigenze differenti, anche contrastanti tra loro. Le persone che usufruiscono del sistema, infatti, desiderano aspettare il minor tempo possibile l'arrivo di un ascensore che possa servirle, avere un viaggio breve e, allo stesso tempo, avere una buona quantità di spazio libero all'interno della cabina dell'ascensore; il gestore del sistema, invece, desidera che ciascun ascensore effettui il minor numero possibile di fermate, in modo da minimizzare il consumo energetico e, di conseguenza, le spese per il funzionamento del sistema. Per raggiungere questo obiettivo il controllore ha a disposizione una quantità enorme di dati, che spaziano dalle richieste presenti nel sistema, sia nei corridoi per richiedere un ascensore, sia a bordo delle cabine degli ascensori per indicare la destinazione desiderata, fino alla posizione e alla destinazione di ogni singolo ascensore; oltre a essere tanti, però, i dati a disposizione non sono sufficienti a inquadrare lo stato completo del sistema, in quanto è difficile sapere quante persone sono in attesa a ciascun piano o quali saranno le richieste future che il sistema sperimenterà, introducendo così una certa dose di casualità. Per una ampia analisi del problema è possibile consultare i lavori [2][38][56].

La maggior parte dei controllori di gruppi di ascensori installati nei sistemi reali sono di tipo euristico; negli ultimi 15 anni, però, è stato proposto un elevato numero di controllori progettati secondo le tecniche dell'intelligenza artificiale, in modo da ottenere prestazioni migliori. La maggior parte di questi controllori sono basati sulla logica fuzzy (come quello proposto in [32]) oppure sugli algoritmi genetici (si veda ad esempio il lavoro esposto in [10]), mentre sono state esplorate poco le tecniche di apprendimento per rinforzo applicate a questo tipo di problema (il lavoro a tal proposito più significativo è presentato in [11]); in generale, inoltre, i controllori

proposti fino a ora non sono adattativi e, quindi, non sono in grado di apportare delle modifiche significative alla loro strategia nel caso in cui le condizioni del sistema varino rispetto a quelle per cui il controllore è stato progettato.

1.1 Obiettivi

Lo scopo di questa tesi è quello di esplorare l'area dell'apprendimento per rinforzo per la progettazione di un controllore di gruppo di ascensori. L'obiettivo principale che ci proponiamo è quello di progettare un controllore che faccia sperimentare attese brevi agli utenti del sistema e che allo stesso tempo sia in grado di adattarsi a possibili cambiamenti del sistema.

Il sistema per cui abbiamo deciso di studiare il controllore è un sistema convenzionale: esso prevede due bottoni di chiamata a ogni piano, utilizzati dagli utenti per richiedere un ascensore indicando in che direzione intendono viaggiare, e una tastiera all'interno di ogni cabina degli ascensori, con cui gli utenti indicano il piano specifico a cui vogliono dirigersi; il sistema, nel suo complesso, prevede una serie di ascensori a singola cabina che servono tutti i piani dell'edificio e possono comportarsi indipendentemente l'uno dall'altro, muovendosi in condotti separati. Il controllore proposto deve, quindi, essere adatto a lavorare in un sistema di questo tipo.

1.2 Metodologia

I controllori che proponiamo sono basati su due differenti architetture: l'architettura piatta, che delega completamente il controllo del sistema agli agenti che apprendono, e l'architettura gerarchica, che delega agli agenti solamente le decisioni strategiche. Anche le tecniche di apprendimento per rinforzo che abbiamo utilizzato, mappandole sulle due architetture proposte, sono due: Fitted Q Iteration (FQI) [61] e metodi gradiente (RLG) [48].

FQI è un metodo offline, che apprende la politica migliore sulla base di traiettorie di dati raccolti in precedenza. Il controllore che ne risulta, quindi, è adattativo solamente fino a un certo punto, perché per adattarsi richiederebbe di immagazzinare una serie di dati e, periodicamente, calcolare una nuova politica utilizzando i dati raccolti; la nuova politica risulta essere sostanzialmente indipendente da quella vecchia e, inoltre, necessita di tempi lunghi per essere calcolata. I risultati ottenuti con un controllore di questo tipo, inoltre, si sono rivelati essere abbastanza scarsi, anche con l'architettura gerarchica.

RLG, invece, è un metodo che, data una politica parametrica, iterativamente aggiorna, in base agli ultimi dati osservati, i parametri di questa politica, seguendo una direzione che permetta di migliorare le prestazioni. Il controllore che ne risulta, quindi, è intrinsicamente adattativo; i risultati che abbiamo ottenuto con un control-

lore di questo tipo con architettura gerarchica, inoltre, si sono rivelati abbastanza buoni, anche se in alcuni casi migliorabili.

Per contrastare il problema della dimensione del problema, che è già elevata nel caso di studio utilizzato che prevede un edificio di 10 piani servito da 4 ascensori e che diventerebbe ingestibile in edifici più grandi, abbiamo investigato il campo della selezione delle feature. In particolare, di ogni controllore proposto abbiamo testato due versioni differenti, una delle quali utilizza tutte le informazioni disponibili, mentre l'altra utilizza solamente quelle selezionate da un algoritmo di selezione delle variabili proposto in questa tesi.

1.3 Contributi

Il contributo principale di questa tesi relativo al campo del controllo di gruppi di ascensori è rappresentato dall'architettura gerarchica, studiata per superare i problemi emersi dall'utilizzo di una banale architettura piatta.

L'architettura piatta prevede che il controllo del sistema sia delegato completamente agli agenti che apprendono: essi, infatti, ricevono le informazioni sullo stato del sistema e propongono di conseguenza un'azione che viene utilizzata direttamente per il controllo e che, quindi, deve rispettare alcuni vincoli sul comportamento degli ascensori (per una dettagliata analisi dei vincoli si confronti il lavoro in [9]). Questa architettura, a causa della complessità del problema da risolvere, si è rivelata non adeguata per permettere l'apprendimento di una buona strategia di controllo.

Per contrastare le difficoltà incontrate con l'utilizzo dell'architettura piatta con i metodi di apprendimento per rinforzo proponiamo, appunto, un'architettura gerarchica, che divide il controllo nel caso gli ascensori abbiano delle persone a bordo da quello con ascensori vuoti, dato che in questi due casi il comportamento da tenere è completamente differente. Con questa architettura è possibile delegare agli agenti che apprendono solamente le decisioni strategiche, imponendo dall'esterno i vincoli necessari; ne consegue che il compito degli agenti è notevolmente semplificato e, quindi, essi riescono a ottenere dei risultati significativi per il controllo di gruppi di ascensori.

Nel campo della selezione delle variabili proponiamo, invece, due nuovi algoritmi, l'algoritmo garantito e l'algoritmo euristico, studiati per superare i due principali problemi che emergono nei metodi di feature selection presenti in letteratura, i quali non forniscono nessuna garanzia teorica sulla bontà della selezione che comportano e non generano un albero delle dipendenze tra le feature, utilizzabile per farsi un'idea del motivo per cui una feature è utile per la soluzione del problema.

L'algoritmo garantito è studiato per risolvere il primo problema; esso, infatti, seleziona le variabili migliori in base a un criterio che garantisce la bontà della scelta rispetto all'individuazione della politica ottima. Questo algoritmo, purtroppo, oltre a non costruire una gerarchia delle variabili, si è rivelato inutilizzabile nella prati-

ca, in quanto i suoi tempi computazionali sono eccessivamente lunghi; esso, quindi, necessita di un ulteriore sviluppo.

L'algoritmo eurisitico, basato sul lavoro preliminare presentato in [7], è invece in grado di costruire, durante la selezione, un albero delle dipendenze tra le varie feature; in particolare, le dipendenze sono costruite individuando le variabili necessarie a prevedere il valore futuro che le variabili del livello precedente (o il rinforzo per quanto riguarda la radice dell'albero) assumeranno all'istante successivo. Questo algoritmo, nonostante non fornisca nessuna garanzia teorica sulla bontà della selezione per scopi di controllo, si è rivelato efficace nella pratica.

1.4 Struttura della tesi

La tesi è strutturata nel modo seguente.

Nel Capitolo 2 riassumiamo i risultati presenti in letteratura sia per quanto riguarda il controllo di gruppi di ascensori, sia per quanto riguarda la selezione delle feature. Per prima cosa inquadriamo il problema del controllo di gruppi di ascensori, descrivendo le varie configurazioni in cui può essere applicato, gli obiettivi che deve raggiungere e le diverse tipologie di traffico che deve servire; proseguiamo, poi, illustrando le principali tecniche di controllo basate su tecniche di intelligenza artificiale proposte in letteratura, dividendole per categoria. Dopo aver presentato i risultati riguardanti il controllo di gruppi di ascensori illustriamo brevemente il formalismo utilizzato per la modellizzazione di un problema di apprendimento per rinforzo, cioè gli MDP, soffermandoci sulle tecniche di riduzione degli MDP presentate in letteratura; in particolare vediamo sia tecniche generali, applicabili a qualsiasi MDP, sia tecniche di feature selection, applicabili a MDP fattorizzati.

Nel Capitolo 3 presentiamo due nuovi algoritmi di selezione delle feature. Il capitolo contiene inizialmente alcune premesse, che formalizzano le trasformazioni necessarie per passare dall'MDP originale a quello ridotto e per confrontare le quantità di interesse nei due modelli. Esso presenta, poi, un problema d'esempio semplice ma significativo, utilizzato per testare gli algoritmi proposti, che sono oggetto delle sezioni successive del capitolo. Infine il capitolo contiene un'analisi dell'utilità, al fine di determinare la politica ottima, delle variabili del problema in base alla loro controllabilità e alle dipendenze che hanno con le altre feature del problema.

Il Capitolo 4 contiene una descrizione dell'ambiente utilizzato per effettuare gli esperimenti. In particolare, in esso descriviamo inizialmente il simulatore di gruppi di ascensori utilizzato e la sua integrazione nell'architettura RL-Glue [62]. Riportiamo, poi, la codifica delle informazioni che abbiamo utilizzato. Il capitolo prosegue con un'analisi dettagliata dei diversi componenti software che costituiscono l'architettura generale del sistema. Infine presentiamo il caso di studio utilizzato, cioè i parametri del sistema fisico che abbiamo deciso di utilizzare per le simulazioni e i parametri utilizzati per le simulazioni stesse.

Nel Capitolo 5 descriviamo i controllori proposti e i risultati ottenuti con essi. Il capitolo si apre con una breve presentazione dei controllori benchmark utilizzati come base per il confronto delle prestazioni, scelti in modo da coprire le proposte più significative presenti in letteratura. Esso prosegue con una descrizione delle due architetture utilizzate per i controllori, quella piatta e quella gerarchica proposta in questa tesi; oltre a una descrizione qualitativa delle architetture riportiamo in dettaglio le feature che esse prevedono e la selezione effettuata su esse con il metodo euristico proposto. Il capitolo prosegue con una ampia analisi dei risultati ottenuti combinando le due architetture con metodi di apprendimento FQI e RLG; vengono anche comparati i risultati ottenuti con l'insieme completo e quello ridotto delle feature, mostrando in questo modo l'efficacia del metodo di feature selection proposto.

Nel Capitolo 6 riassumiamo i risultati principali di questa tesi e indichiamo alcune possibili direzioni di ricerca per un ulteriore sviluppo delle tematiche presentate.

Capitolo 2

Stato dell'arte

Un sistema di controllo di un gruppo di ascensori (**EGCS**, dall'inglese *Elevator Group Control System*) è un sistema che si occupa di assegnare un ascensore agli utenti che ne hanno fatto richiesta, in modo da migliorare le prestazioni, in accordo ad alcuni criteri predefiniti.

I primi controllori elettronici per i gruppi di ascensori, realizzati tramite relé, appaiono negli anni '70, mentre agli inizi degli anni '80 si ha lo sviluppo di controllori completamente basati su microprocessori [56]. Inizialmente, le strategie di controllo sono implementate in accordo ad alcune euristiche, definite da esperti del settore, che cercano di esplicitare i comportamenti apparentemente più efficienti; tra queste strategie, le più famose e diffuse sono:

- *Scheduling Method*, in cui gli ascensori, a specifici intervalli di tempo, vengono inviati da un piano terminale all'altro, servendo tutte le chiamate presenti lungo il loro tragitto [65];
- *Demand Zoning Method*, in cui le chiamate vengono suddivise in zone e ciascun ascensore viene assegnato a una zona specifica, in modo che possa servire tutte le chiamate in essa emerse [65][2];
- *Call Assignment Method*, in cui, basandosi sulle condizioni del traffico e sullo stato di ciascun ascensore, si cerca di prevedere il valore di alcuni indici predefiniti (il tempo necessario a ciascun ascensore per arrivare al piano in cui è emersa la chiamata, il numero di passeggeri che scenderanno e saliranno a ciascun piano, ecc.), che vengono poi utilizzati per selezionare l'ascensore più adeguato per servire ciascuna chiamata [65];
- *Collective Control*, in cui ogni ascensore, indipendentemente e senza coordinarsi con gli altri, risponde in sequenza a tutte le chiamate lungo il suo percorso, per poi invertire la sua direzione di marcia [40];
- *Nearest Car*, simile a Collective Control, ma le chiamate sone servite solamente dall'ascensore più vicino ad esse [2][33];

- *Earliest Car*, simile a Collective Control, ma le chiamate sone servite solamente dall'ascensore il cui tempo di arrivo al piano in cui è emersa la chiamata è il minore [33].

Dagli inizi degli anni '90, si comincia a introdurre tecniche di intelligenza artificiale all'interno degli EGCS, in modo da ottenere prestazioni sempre migliori. Sebbene siano stati proposti metodi per sfruttare una serie di tecniche diverse, tra cui anche l'apprendimento per rinforzo, gli approcci su cui si è concentrata la letteratura e per cui sono stati ottenuti risultati significativi sono essenzialmente tre: logica fuzzy, reti neurali e algoritmi genetici [49], usati sia per sviluppare dei veri e propri controllori, sia come metodi al servizio di altre tecniche. Negli ultimi anni, inoltre, si è cominciato ad applicare la programmazione intera e mista intera al problema del controllo dei gruppi di ascensori, anche se in questo campo la ricerca sembra essere solamente agli inizi [54].

Il capitolo prosegue nel seguente modo: la Sezione 2.1 contiene una presentazione delle diverse configurazioni possibili per l'EGCS, nella Sezione 2.2 vengono presentati diversi criteri che si possono ottimizzare per ottenere le prestazioni desiderate per il sistema, la Sezione 2.3 contiene una descrizione delle caratteristiche proprie del traffico riguardante il sistema, mentre la Sezione 2.4 contiene una panoramica sulle principali tecniche di controllo per l'EGCS proposte in letteratura. Infine, la sezione 2.5 contiene le nozioni di base sul formalismo utilizzato in questa tesi per studiare il sistema di controllo di un gruppo di ascensori.

2.1 Configurazioni

La configurazione base e più diffusa per i gruppi di ascensori prevede dai due agli otto ascensori a singola cabina, due tasti a ogni piano attraverso cui gli utenti possono chiamare un ascensore indicando la direzione in cui intendono viaggiare, un pannello all'interno di ogni cabina attraverso cui gli utenti selezionano il piano di destinazione desiderato, e un indicatore luminoso o un segnale acustico per segnalare agli utenti quale ascensore è destinato a rispondere alle chiamate [38]. In questo modo, il sistema rimane semplice e intuitivo da usare per gli utenti, ma il controllore ha a disposizione un numero molto ridotto di informazioni utilizzabili per prendere una buona decisione; ad esempio, infatti, il sistema non conosce quanti passeggeri sono in attesa a ogni piano, mentre le destinazioni sono note solamente dopo che i passeggeri le hanno selezionate dalla cabina dell'ascensore. È presente anche un'ulteriore incertezza dovuta a comportamenti non coerenti degli utenti, che, ad esempio, possono cambiare idea dopo aver già effettuato una richiesta. Per sopperire, almeno in parte, alla carenza di informazioni disponibili, la configurazione base solitamente prevede che la decisione sull'ascensore inviato a servire una chiamata sia cambiata continuamente e segnalata il più tardi possibile agli utenti, causando però un allungamento del tempo di attesa psicologico e una certa confusione all'arrivo dell'ascensore.

Allo scopo di avere a disposizione un numero maggiore di informazioni e, di conseguenza, poter proporre delle strategie di gestione degli ascensori migliori, in letteratura sono state proposte diverse alternative strutturali, consistenti in variazioni alla configurazione base descritta in precedenza. L'alternativa più diffusa e applicata nella pratica è il cosiddetto *Destination Control System*. Questo sistema prevede una tastiera digitale a ogni piano, attraverso cui i passeggeri indicano direttamente in fase di chiamata il piano a cui vogliono andare; dopo aver ricevuto la richiesta, il sistema seleziona direttamente un ascensore destinato a servirla e lo indica all'utente, che così può recarsi immediatamente all'ascensore che lo servirà. La cabina dell'ascensore, invece, è sprovvista di qualsiasi pannello per la scelta della destinazione, che è già nota al sistema; in questo modo, si prevengono comportamenti anomali da parte dei passeggeri, che non possono più cambiare idea circa la loro destinazione una volta saliti sull'ascensore [38]. In questa configurazione, come accennato in precedenza, è necessario che il controllore assegni immediatamente, alla ricezione di una nuova richiesta, l'ascensore destinato a servirla; l'imposizione di una scelta anticipata impedisce, ovviamente, di fornire una soluzione ottima al problema considerato, in quanto le informazioni successive alla scelta non possono essere usate, ma si rende necessaria per il corretto funzionamento pratico del sistema. Un passeggero, infatti, rimarrebbe indispettito nel non ricevere una risposta immediata e potrebbe pensare che il sistema non stia funzionando; inoltre, è necessario che egli liberi la postazione di scelta per il passeggero successivo in tempi brevi. Nonostante questa limitazione pratica, i vantaggi portati dal Destination Control System sono elevati; con esso, infatti l'incertezza è drasticamente ridotta: il sistema conosce, in modo abbastanza preciso, quanti passeggeri sono in attesa a ogni piano e la loro destinazione. Esso, di conseguenza, ha molti più elementi disponibili, rispetto alla configurazione base, per prendere una buona decisione.

L'informazione sul numero di passeggeri, sia in attesa sia all'interno degli ascensori, è molto utile in tutti i casi, anche in presenza dei soli bottoni di direzione ai piani. Queste informazioni, ad esempio, possono essere usate proficuamente per considerare l'affollamento di un ascensore o per individuare i casi in cui è necessario inviare più di un ascensore allo stesso piano. Mentre per il conteggio delle persone all'interno della cabina è abbastanza consolidato l'utilizzo di sistemi precisi di peso, non esistono tecniche standard per contare quelle in attesa ai vari piani; una possibilità adatta a questo scopo è l'utilizzo di telecamere e tecniche di riconoscimento di immagini [36]. Con le tecniche attuali, infatti, i risultati ottenibili sono abbastanza precisi, mentre il costo per l'installazione delle telecamere è basso e, inoltre, è possibile sfruttare le telecamere di sicurezza già installate. Si può, inoltre, cercare di utilizzare alcuni criteri per capire quante persone vogliono viaggiare in una direzione e quante in quella opposta.

L'introduzione di configurazioni più complicate rispetto a quella base non ha come unico scopo quello di rendere disponibili maggiori informazioni al controllo-

re. Un obiettivo, ad esempio, potrebbe essere quello di ottenere buone prestazioni occupando meno spazio rispetto a quello necessario con la configurazione base, al fine di ottenere una infrastruttura più economica. Questo obiettivo è facilmente raggiungibile tramite ascensori a due (o più) cabine connesse [29][57]. In questa configurazione, due cabine sono connesse tra loro formando un unico ascensore, che può quindi servire contemporaneamente due piani adiacenti; questo, però, comporta un controllore del gruppo di ascensori più complicato, perché deve tenere in considerazione anche le dipendenze tra le cabine di ciascun ascensore, in quanto, ovviamente, il movimento di una delle due cabine comporta il movimento anche dell'altra cabina dell'ascensore considerato. Una variante della soluzione precedente è rappresentata da ascensori a più cabine indipendenti [66]. Questa configurazione è più flessibile della precedente, in quanto i movimenti relativi delle cabine appartenenti allo stesso ascensore sono maggiormente indipendenti; si ha, però, una maggiore complicazione, dovuta alla necessità di evitare scontri tra le cabine appartenenti allo stesso ascensore. Innanzitutto, il controllore deve tenerne conto, per cui un Destination Control System è necessario, in quanto è l'unico in grado di fornire abbastanza informazioni. Inoltre, bisogna prevedere anche un sistema di prevenzione, che ignora il controllore del gruppo di ascensori per fermare, in caso di necessità, le cabine; questa situazione, comunque, è da evitare, in quanto porta il sistema in una situazione di funzionamento eccezionale, che peggiora sensibilmente le prestazioni e potrebbe anche portare a deadlock.

A volte, inoltre, si vuole che alcuni piani o alcuni utenti abbiano un servizio speciale. Per esempio, si potrebbe volere che un certo piano abbia un servizio particolarmente veloce, mentre un altro piano sia servito con ascensori poco affollati [23]. I motivi per avere servizi differenziati sono diversi [38]:

- *Restrizione degli accessi*: per ragione di sicurezza o privacy, in edifici popolati da categorie diverse di persone, è necessario che alcuni piani siano accessibili soltanto alle persone autorizzate;

- *Servizi VIP*: alcuni passeggeri, detti VIP, come ad esempio pompieri o medici per le emergenze, devono essere serviti con la massima priorità;

- *Separazione di gruppi di passeggeri*: è necessario non far incontrare sullo stesso ascensore persone appartenenti a gruppi incompatibili, come ad esempio dipendenti che consegnano il cibo e dipendenti che svuotano i cestini della spazzatura.

Per offrire questi servizi non esistono delle configurazioni specifiche: tutte quelle viste in precedenza possono essere impiegate per raggiungere gli obiettivi prefissati, anche se, ovviamente, risulta più semplice ottenerli quante più sono le informazioni a disposizione. Al contrario, il controllore del EGCS deve essere configurato in modo da poter considerare anche gli obiettivi appena esposti nella sua scelta.

2.2 Criteri da ottimizzare

Le performance del controllore del gruppo di ascensori dipendono pesantemente dai criteri considerati nel processo di ottimizzazione. La scelta di questi criteri è molto complicata, infatti gli obiettivi che si vogliono raggiungere sono molti e, spesso, in contrasto tra loro: si tratta, quindi, di un vero e proprio problema di ottimizzazione multiobiettivo.

I principali criteri proposti in letteratura, che tentato di catturare i diversi obiettivi desiderabili per le performance del gruppo di ascensori, sono, divisi per categoria, i seguenti [56] [11]:

- **Criteri temporali**: sono i criteri correlati con la velocità del viaggio:
  - *Tempo di attesa medio*: è la media dei tempi che i passeggeri passano in attesa dell'arrivo di un ascensore; dato che solitamente questa informazione non è disponibile, questo criterio viene approssimato con la media dei tempi dalla pressione del bottone di chiamata di un ascensore all'arrivo dell'ascensore;
  - *Tempo di attesa massimo*: come il criterio precedente, ma prendendo il massimo e non la media;
  - *Quadrato del tempo di attesa medio*: viene preso in considerazione il quadrato della media dei tempi di attesa dei passeggeri (o dei tempi di pressione dei bottoni di chiamata degli ascensori), in modo da offrire un servizio più equo, diminuendo la varianza dei tempi di attesa e prevenendo la formazione di lunghe code;
  - *Tempo di viaggio (medio e massimo)*: è il tempo che le persone passano all'interno della cabina dell'ascensore;
  - *Tempo di sistema (medio e massimo)*: è il tempo totale che le persone passano nel sistema, comprendente sia il tempo di attesa sia il tempo di viaggio;
  - *Percentuale di lunga attesa*: è la percentuale di utenti che sono costretti ad attendere un tempo superiore a un certo tempo prefissato, solitamente 60 secondi, che rappresenta il limite massimo di tolleranza prima che il servizio venga considerato scadente;
- **Criteri ambientali**: sono i criteri correlati con la qualità dell'ambiente in cui avviene il viaggio:
  - *Affollamento*: è la quantità di persone contemporaneamente presenti sulla stessa cabina di un ascensore;
- **Criteri energetici**: sono i criteri correlati con il consumo di energia causato dal gruppo di ascensori:

- *Consumo di energia*: è l'energia utilizzata dall'intero sistema di ascensori;
- *Numero di partenze e fermate*: è il numero di volte che un ascensore parte o si ferma; questa misura è un'approssimazione del consumo di energia, in quanto la maggior parte di energia utilizzata dal sistema è dovuta appunto alla partenza da fermo e all'arresto dell'ascensore.

Inoltre, può essere utile considerare anche la varianza [15] o la dispersione lineare [54] dei valori assunti dai criteri considerati, in modo da offrire un servizio più equo ai passeggeri.

Sebbene sia possibile ottimizzare un solo criterio, che risponde alle esigenze maggiori emerse, e verificare a posteriori come il sistema si comporta nei confronti degli altri criteri [11], in letteratura solitamente si preferisce tenere in considerazione più criteri in contemporanea, ottimizzando una loro combinazione lineare. I modi in cui è possibile attribuire i pesi alla combinazione lineare sono diversi. La soluzione più semplice consiste nell'assegnare dei pesi standard, selezionati tramite dati sperimentali o basandosi su considerazioni fatte a priori [30]; inoltre, per ottenere risultati migliori, è possibile scegliere questi pesi in base alle caratteristiche dell'edificio in cui viene installato l'EGCS [55]. Un'alternativa consiste nel permettere all'amministratore del sistema di scegliere i pesi da dare alla combinazione lineare, in modo da consentirgli di scegliere che cosa è più importante ottimizzare [35].

Tutti i metodi proposti finora, però, hanno lo svantaggio di necessitare di conoscenze specifiche, in modo da poter scegliere dei pesi significativi; per risolvere questo problema, è possibile dare al sistema la facoltà di scegliere automaticamente i coefficienti della combinazione lineare più opportuni, ad esempio tramite una rete neurale [31]. Oppure, sempre tenendo il sistema autonomo, è possibile definire una rete genetica che si struttura in modo da utilizzare i vari criteri (ognuno dei quali rappresenta un nodo della rete) nel modo più opportuno [29]. In particolare, l'autocalibrazione dei parametri consente al sistema di utilizzare i criteri più adeguati per il particolare tipo di traffico presente in quel momento (si veda la Sezione 2.3), sfruttando il fatto che l'obiettivo e le performance dipendono appunto, e in modo significativo, dalla tipologia di traffico.

2.3 Tipi di traffico

I passeggeri non si comportano allo stesso modo durante tutto l'arco della giornata, ma il numero di chiamate agli ascensori, il piano a cui avvengono le chiamate e il piano di destinazione dipendono dal particolare orario. In particolare, gli schemi di traffico individuabili sono i seguenti [2]:

- **up peak**: si ha quando il flusso dominante, o unico, di traffico è in direzione verso l'alto, con tutti o la maggioranza dei passeggeri che entrano nel sistema di ascensori al piano d'entrata e uscita dell'edificio;

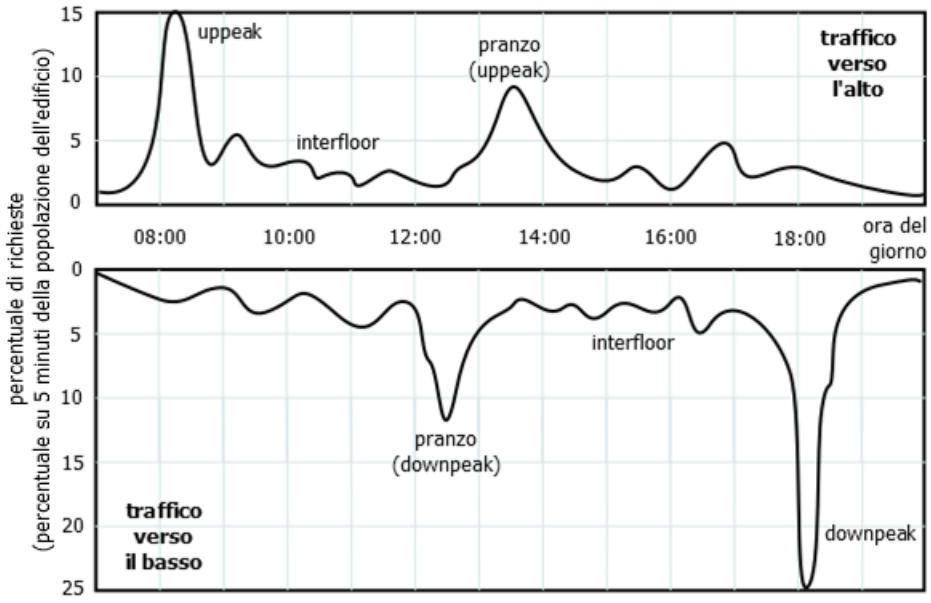


Figura 2.1: Andamento tipico del traffico in un edificio durante una giornata lavorativa.

- **down peak**: si ha quando il flusso dominante, o unico, di traffico è in direzione verso il basso, con tutti o la maggioranza dei passeggeri che escono dal sistema di ascensori al piano d'entrata e uscita dell'edificio;
- **two way**: si ha quando il flusso dominante di traffico è da e verso uno specifico piano, che può essere il piano di entrata e uscita dell'edificio;
- **random interfloor**: si ha quando nessuno dei precedenti schemi di traffico è applicabile.

In Figura 2.1 è riportato l'andamento tipico del traffico durante una giornata lavorativa in un edificio che ospita uffici. Come si può notare, la mattina si ha un forte up peak, corrispondente all'orario in cui le persone si recano al lavoro; in seguito, si ha traffico random interfloor, corrispondente ai normali spostamenti tra piani; a metà giornata si ha traffico two way (chiamato, in questo caso, anche traffico mid day o traffico lunch time), dovuto alle persone che si recano a pranzo e ritornano dallo stesso; segue, poi, traffico random interfloor; infine, alla sera si ha traffico down peak, corrispondente alle persone che, terminato il turno di lavoro, lasciano l'edificio.

Ovviamente, quello riportato in Figura 2.1 è solo un possibile andamento del traffico durante la giornata, che dipende fortemente da diversi fattori, come, ad esempio, il tipo di utilizzo dell'edificio, la presenza o meno di orari flessibili, ecc. Esso, comunque, è significativo, perché, oltre a rappresentare un andamento tipico del traffico in uffici standard, mostra come è possibile comporre gli schemi di traffico presentati in precedenza per ottenere l'andamento del traffico durante una intera giornata.

La divisione del traffico in schemi non è fine a se stessa, ma gioca un ruolo fondamentale nella definizione di un buon controllore di gruppi di ascensori. Infatti, la strategia da seguire per ottenere buoni risultati non è sempre la stessa, ma dipende dal particolare tipo di traffico in corso. Di conseguenza, affinché i controllori possano raggiungere buoni risultati, è importante che considerino il tipo di traffico in corso [31]; questo può essere fatto sia indirettamente, con il controllore che si adatta a cambiamenti generici [33], sia direttamente, attraverso un modulo che si occupa di individuare lo schema di traffico in corso [15]. Inoltre, può essere utile anche predire il traffico futuro, permettendo al controllore di adeguarsi progressivamente e non all'improvviso al cambiamento di tipologia del traffico [55].

2.4 Tecniche di controllo

In letteratura sono presenti svariati controllori di gruppi di ascensori, ognuno con le proprie caratteristiche, dipendenti fortemente soprattutto dalla tecnica utilizzata per effettuare il controllo. È però possibile trovare degli aspetti comuni a tutti i controllori, indipendentemente dalla tecnica usata.

Tutti i controllori, ad esempio, rispettano le regole di Closs [9]; si tratta di regole basilari, che eliminano comportamenti non accettabili da parte del gruppo di ascensori. Grazie a queste semplici regole, però, è possibile ridurre di molto lo spazio di ricerca di una buona strategia di gestione degli ascensori, semplificando il lavoro del controllore, che, di conseguenza, può essere più efficiente. In particolare, le regole sono le seguenti:

- Un ascensore non può fermarsi a un piano se nessun passeggero entra o esce a quel piano;
- Un ascensore non può oltrepassare un piano a cui un passeggero vuole uscire;
- Un passeggero non può entrare in un ascensore che sta trasportando altri passeggeri e sta viaggiando nella direzione opposta a quella desiderata;
- Un ascensore non può cambiare direzione di marcia mentre sta trasportando passeggeri.

Un'altra caratteristica comune a molti controllori, anche se non a tutti, è la strutturazione del sistema su due livelli. Il primo livello si occupa del riconoscimento del tipo di traffico corrente e, a volte, anche della previsione del tipo di traffico futuro (si veda a tal proposito la Sezione 2.3). Questa informazione è sfruttata dal secondo livello, che può effettuare il controllo vero e proprio, gestendo i vari ascensori, adattandosi al tipo di traffico riconosciuto; ovviamente, la qualità delle sue decisioni dipende anche dalla bontà e dalla precisione del sottosistema di primo livello.

Nelle sottosezioni seguenti vengono riportati i risultati ottenuti fino a ora con le principali tecniche utilizzate per il controllo del gruppo di ascensori.

2.4.1 Logica Fuzzy

Un insieme fuzzy è un insieme che rientra in un'estensione della teoria classica degli insiemi: esso è caratterizzato da una funzione di grado di appartenenza, che mappa gli elementi di un universo in un intervallo reale continuo [0;1], in cui il valore 0 indica che l'elemento non è per niente incluso nell'insieme, il valore 1 indica che l'elemento è certamente incluso nell'insieme, mentre i valori tra zero e uno indicano il grado di appartenenza dell'elemento all'insieme fuzzy in questione. Corrispondentemente, la logica fuzzy è una logica in cui si può attribuire a ciascuna proposizione un grado di verità (detto anche valore di appartenenza) compreso tra 0 e 1, in cui 0 indica che la proposizione è falsa, 1 che è vera, mentre i valori intermedi corrispondono a valori intermedi di verità. I concetti di insieme e logica fuzzy possono essere utilizzati anche per definire delle regole fuzzy; si tratta di regole del tipo:

$$IF\ x_1\ is\ L_1\ AND\ \ldots\ AND\ x_n\ is\ L_n$$
$$THEN\ y_1\ is\ K_1\ AND\ \ldots\ AND\ y_m\ is\ K_m$$

in cui x_i e y_i sono variabili linguistiche, mentre L_i e K_i sono valori appartenenti a insiemi fuzzy [53].

La definizione di un controllore di gruppo di ascensori fuzzy ha diversi vantaggi [25]:

- Le strategie di allocazione delle chiamate sono spesso espresse come un insieme di regole (fuzzy) fornite da esperti di progettazione di ascensori;
- Insiemi diversi di regole possono essere facilmente raggruppati in una base di conoscenza strutturata per coprire le diverse condizioni di traffico discusse nella Sezione 2.3;
- Vincoli logici possono essere agevolmente inclusi nella base di conoscenza;
- È possibile sviluppare e incorporare facilmente meta-regole, al fine di scegliere le regole di allocazione per gli ascensori più adeguate a un dato tipo di traffico;
- Il controllore risponde in modo prevedibile a eventi di sistema;
- Concetti fuzzy, come alto, medio e basso, sono facilmente comprensibili e usabili da esperti.

In letteratura sono presenti diversi lavori in cui la logica fuzzy viene applicata al problema di controllare un gruppo di ascensori, spaziando da sistemi relativamente piccoli, con ad esempio 8 piani e 3 ascensori [45], fino ad arrivare a sistemi composti da 30 piani e 6 ascensori [32] o da 35 piani e 8 ascensori [33].

Le regole fuzzy sono molto utilizzate per riconoscere il tipo di traffico in corso (e prevedere quello futuro a breve termine) e poter quindi fornire informazioni più

complete al controllore di gruppo vero e proprio. È possibile accoppiare la previsione del traffico effettuata tramite regole fuzzy sia a controllori euristici, per esempio definendone uno che sfrutta la previsione per poter proporre una strategia migliore [55], oppure prevedendo un insieme di controllori diversi e scegliendo, di volta in volta, quello più adatto per il particolare tipo di traffico in corso [45], sia a controllori anch'essi fuzzy [35][25].

Nell'approccio appena descritto, però, le prestazioni del sistema dipendono fortemente dalla qualità dell'individuazione e della previsione del traffico; per evitare questo problema, è possibile definire un controllore di gruppo che si adatta automaticamente ai cambiamenti di traffico, senza la necessità di dover prevedere esplicitamente il tipo di traffico in corso. L'adattamento si può ottenere scegliendo tra insiemi diversi di regole fuzzy, a intervalli predefiniti di tempo, quello che ha le performance migliori [33]; modificando adeguatamente il dominio dei valori fuzzy, seguendo il trend degli indici utilizzati per misurare le performance [33][25]; oppure, definendo delle regole che tengono in considerazione delle misure significative del traffico in corso, basate, ad esempio, sulle fermate schedulate per un ascensore in un certo range di piani [32]. Inoltre, è possibile accoppiare il controllore fuzzy con una rete neurale, in modo da sfruttare la predisposizione di quest'ultima ad adattarsi ai cambiamenti [31].

All'approccio fuzzy è possibile affiancare anche i sistemi esperti, in modo da poter definire delle regole precise e non vaghe come quelle fuzzy [65]; in questo modo, quindi, è possibile fornire dei comportamenti completamente deterministici che il sistema deve tenere in certe situazioni particolari, come, ad esempio, impedire a un ascensore assegnato a una chiamata ad alta priorità di servire un'altra chiamata.

I lavori presenti in letteratura basati sull'approccio fuzzy sono abbastanza completi sia per quanto riguarda il traffico sia per quanto riguarda gli indici di prestazioni. Solitamente, infatti, vengono analizzate le prestazioni del controllore a fronte di più di un tipo di traffico, compreso il traffico completo di una giornata di ufficio. Inoltre, solitamente vengono ottimizzati e analizzati diversi dei criteri presentati nella sezione 2.2, utilizzando principalmente una loro combinazione lineare per l'ottimizzazione, anche se il più utilizzato è il tempo di attesa medio. Per quanto riguarda la configurazione del sistema, invece, sono presenti essenzialmente lavori che utilizzano la configurazione standard, con due bottoni di chiamata ai piani e destinazione scelta sulla cabina (singola) dell'ascensore.

2.4.2 Reti Neurali

Una rete neurale artificiale è un sistema adattivo studiato in modo da tentare di replicare il funzionamento del cervello umano; essa, infatti, durante un periodo di apprendimento, cambia la sua struttura in base alle informazioni, esterne o interne, che le vengono presentate. Si tratta, di fatto, di un complesso approssimatore non lineare, che apprende la funzione da approssimare tramite dati sperimentali [4].

Le reti neurali possono essere utilizzate con profitto per la realizzazione di un sistema di controllo per un gruppo di ascensori [14]; esse, infatti, possono distinguere i diversi flussi di traffico, hanno capacità di apprendimento e, di conseguenza, possono costruire un modello adattativo.

In letteratura, però, nella definizione di EGCS le reti neurali sono presentate soprattutto come supporto ad altre tecniche. Esse, infatti, grazie alla loro capacità di approssimare funzioni non lineari, possono essere utilizzate per salvare i valori Q, necessari per l'apprendimento per rinforzo, in un ambiente esteso come quello del gruppo di ascensori [3][11]. Oppure, è possibile accoppiare una rete neurale a un controllore fuzzy, in modo da permettere al modello di adattarsi ai cambiamenti a lungo termine [31].

2.4.3 Algoritmi Genetici e Reti di Programmazione Genetica

Un algoritmo genetico è un algoritmo euristico ispirato al principio della selezione naturale ed evoluzione biologica: partendo da un insieme (*popolazione*) di soluzioni (*cromosomi*), le ricombina (*crossover*) e vi introduce elementi di disordine (*mutazione*) per ottenere soluzioni nuove, per poi tenere gli elementi con caratteristiche migliori rispetto a certe misure (*fitness*). Si tratta, quindi, di un algoritmo iterativo di ottimizzazione, che non ha la garanzia di trovare la soluzione ottima ma che solitamente, nella pratica, trova una buona soluzione in un tempo ragionevole [43].

Le prestazioni degli algoritmi genetici sono migliorabili sfruttando reti di programmazione genetica. Esse sono formate da grafi diretti aciclici, i cui nodi sono nodi di giudizio o di computazione, la cui struttura, che varia seguendo gli stessi principi degli algoritmi genetici, corrisponde al flusso di esecuzione dell'algoritmo di ottimizzazione [34].

In letteratura, a partire dalla fine degli anni '90, sono stati presentati diversi lavori che sfruttano gli algoritmi genetici e le reti di programmazione genetica per migliorare le performance degli EGCS, applicandoli anche a edifici relativamente grandi, composti ad esempio da 29 piani serviti con 8 ascensori [57]. Inoltre, sono state prese in considerazione quasi tutte le configurazioni presentate nella Sezione 2.1: quella classica [10][16][15], destination control [29][76], ascensori a due cabine [29][57][76], servizi speciali [23], informazioni aggiuntive tramite telecamere [36]. I lavori citati, inoltre, coprono bene anche i vari schemi di traffico presentati nella Sezione 2.3.

L'alta capacità di adattamento delle tecniche genetiche al contesto permette di non prevedere esplicitamente il traffico; infatti, inserendo nei geni informazioni opportune legate al traffico, è possibile ottener un controllore che si adatta automaticamente alle variazioni del traffico [36], per esempio attraverso una rete di programmazione genetica la cui struttura cambia in base ai cambiamenti del traffico [15].

È possibile, inoltre, affiancare altre tecniche di apprendimento agli algoritmi genetici e alle reti di programmazione genetica, al fine di ottenere delle maggiori ca-

pacità di adattamento e, di conseguenza, delle strategie di controllo migliori. In [73][74][75][76], ad esempio, il controllore è implementato tramite reti di programmazione genetica estese con apprendimento per rinforzo: la parte genetica crea una rete di macro nodi, mentre il nodo particolare da utilizzare per la scelta della strategia viene appreso per rinforzo, ottenendo così un controllore più versatile.

Essendo gli algoritmi genetici essenzialmente degli algoritmi di ottimizzazione, è possibile definire il problema del controllo del gruppo di ascensori come un problema di programmazione matematica (in particolare, come problema di pickup and delivery) e risolverlo tramite algoritmi genetici [57]. In questo modo, sebbene non si ottenga la soluzione ottima, è possibile ottenere una buona soluzione in tempi più brevi rispetto a quelli necessari per calcolare la soluzione esatta tramite le tecniche classiche di soluzione dei problemi di programmazione matematica.

Gli algoritmi genetici, comunque, a causa della loro natura iterativa, rimangono critici per quanto riguarda il tempo computazionale; risulta, quindi, necessario definire dei geni semplici e mantenere una popolazione numericamente ridotta, in modo da essere in grado di fornire la soluzione nei tempi ridotti necessari per un controllo effettivo del sistema [10].

2.4.4 Apprendimento per Rinforzo

L'apprendimento per rinforzo è una tecnica di apprendimento automatico che punta ad attuare sistemi in grado di apprendere e adattarsi alle mutazioni dell'ambiente in cui sono immersi attraverso la distribuzione di una ricompensa che consiste nella valutazione delle loro prestazioni e che prende appunto il nome di rinforzo. In particolare, l'agente che apprende ha un'osservazione, corrispondente alla sua percezione dello stato corrente, e sceglie un'azione da compiere in quello stato; di conseguenza, esso ottiene una ricompensa e ha un'altra osservazione, potendo così ripetere il ciclo. Il suo scopo è quello di massimizzare la somma (solitamente scontata) delle ricompense [60].

In letteratura è possibile trovare alcuni lavori in cui viene implementato un controllore di gruppo di ascensori basato sull'apprendimento per rinforzo, ma, stando alla nostra conoscenza, i risultati ottenuti fino a ora sono abbastanza limitati.

La maggior parte dei lavori, ad esempio, considera una sola situazione di traffico, up peak [46] o down peak [3][11][72][77], considerando quest'ultimo come il più adeguato in quanto quello che richiede il maggior numero di decisioni, invece di istruire l'algoritmo e testare i risultati ottenibili con schemi di traffico diversi o, meglio ancora, con traffico variante durante la giornata.

Inoltre, stando alla nostra conoscenza, tutti i lavoro sono applicati alla configurazione standard, ad eccezione di [66], in cui vengono presi in considerazione ascensori a più cabine indipendenti.

Il controllore di gruppo, nella maggior parte dei casi, è implementato in modo distribuito, assegnando un agente differente ad ogni cabina degli ascensori; ovvia-

mente, è necessario prevedere dei meccanismi di sincronizzazione, per evitare che agenti differenti decidano di rispondere alla stessa chiamata, causando un decadimento delle prestazioni del sistema. Ad esempio, i vari agenti possono essere inseriti in un contesto multiagente, in cui si coordinano attraverso comunicazione [46]. Oppure, gli agenti possono attivarsi in sequenza e non in contemporanea: in questo modo, quando un agente decide di rispondere a una chiamata, la toglie dalla lista di chiamate in attesa e nessun altro agente può quindi decidere di rispondervi [66]. Un'altra alternativa consiste nel tenere solamente l'apprendimento separato, mentre la strategia di controllo è condivisa [68]; tuttavia, in [66] si evidenzia che l'apprendimento è più rapido se i diversi agenti condividono il loro apprendimento. Infine, un'ulteriore alternativa consiste nel lasciare all'apprendimento degli agenti anche la sincronizzazione, semplicemente apprendendo una strategia che dipende anche dallo stato dei vari ascensori [3][11][77].

In linea teorica, è possibile avere anche un unico agente centralizzato, come avviene in [72], in cui la soluzione viene però applicata a un problema giocattolo e non significativo, consistente di 5 piani, 1 ascensore con capienza 4 persone, traffico puramente down peak e al massimo una persona in attesa per ogni piano.

L'utilizzo di un problema giocattolo permette anche di salvare i valori Q appresi in forma tabulare, cosa che normalmente non è possibile date le enormi dimensioni del problema in una configurazione realistica. Tuttavia, è possibile provare a mantenere una rappresentazione tabulare anche in problemi seri, rappresentando lo stato con opportune feature; questa strategia è presentata in [68], in cui viene applicata a un problema realistico, con 18 piani e 6 ascensori. Tuttavia, gli stessi autori affermano che utilizzando una rappresentazione tabulare non è possibile scalare su un problema più grande (stando alle nostre conoscenze, comunque, questa è la configurazione più estesa a cui è stato applicato fino a ora un controllore tramite apprendimento per rinforzo).

Per scalare su problemi più grandi o per utilizzare una rappresentazione più estesa dello stato è necessario utilizzare approssimatori per rappresentare la funzione di decisione che viene appresa. Una possibilità consiste nell'utilizzare una rete neurale, come proposto in [3][11][77]; tuttavia, nei lavori citati, come viene sottolineato in [38], la struttura della rete proposta dipende fortemente dalla particolare configurazione e quindi non è sufficientemente generalizzabile.

In letteratura, inoltre, è possibile trovare dei lavori in cui l'apprendimento per rinforzo è accoppiato ad altre tecniche, per migliorare i risultati ottenibili con queste ultime; ad esempio, in [73][74][75][76] l'apprendimento per rinforzo è utilizzato per migliorare i risultati ottenibili con reti di programmazione genetica (si veda a tal proposito la sezione 2.4.3).

2.4.5 Altre tecniche

In letteratura sono stati proposti controllori di gruppo di ascensori che sfruttano tecniche diverse rispetto a quelle presentate nelle sezioni precedenti. Si tratta, però, di proposte recenti oppure di proposte che non sono state prese in considerazione da autori diversi da quelli che originalmente hanno effettuato la proposta; di conseguenza, i controllori che sfruttano queste tecniche non sono stati molto approfonditi. Per questo motivo, essi vengono solamente accennati nel seguito di questa sezione.

Un modello che si presta bene a definire il problema in questione è quello della programmazione matematica (intera o intera mista) e, in particolare, del problema di pickup and delivery [57][54]. Questa tecnica, però, necessita di tempi molto lunghi per trovare una soluzione; per ovviare a questa difficoltà, è possibile accontentarsi di una soluzione non esatta, trovabile con algoritmi genetici [57][59], oppure è necessario utilizzare una configurazione che fornisce un numero elevato di informazioni al controllore, come destination control [54].

Un'altra combinazione interessante di tecniche è composta dalla pianificazione di intelligenza artificiale e dai sistemi multiagente: a ogni evento, il singolo ascensore cerca, mediante pianificazione, la soluzione migliore per se stesso dato lo stato istantaneo in cui il sistema si trova; quindi, il sistema multiagente, e in particolare il meccanismo delle aste, viene utilizzato per scegliere l'ascensore migliore per servire una particolare chiamata [38].

Altre tecniche proposte in letteratura sono Immune Particle Swarm Hybrid Optimization Algorithm, per effettuare il controllo [41], Support Vector Machine, per predire il traffico [42] e automi cellulari, per modellare il sistema [71].

2.5 Formalismo

In questa tesi studiamo l'EGCS tramite apprendimento per rinforzo. Di conseguenza, è utile formalizzare il sistema come un Markov Decision Process (MDP) [60].

Definizione 2.1. Un **MDP** è una tupla $< S, A, P, R, \gamma, \mu >$, dove S è l'insieme degli stati, A è l'insieme delle azioni, $P : S \times A \times S \to [0, 1]$ è la funzione probabilistica di transizione, essendo $P(s, a, s')$ la probabilità di transizione dallo stato s allo stato s' compiendo l'azione a, $R : S \times A \to \mathbb{R}$ è la funzione di rinforzo, essendo $R(s, a)$ il rinforzo atteso per eseguire l'azione a nello stato s, $\gamma \in [0, 1]$ è il discount factor e μ è la distribuzione di probabilità dello stato iniziale.

Una *politica* è una relazione $\pi : S \times A \to [0, 1]$, essendo $\pi(s, a)$ la probabilità di scegliere l'azione a nello stato s. La qualità di una politica π può essere valutata attraverso la somma con orizzonte infinito del rinforzo scontato; più precisamente, si può definire una funzione $V^\pi : S \to \mathbb{R}$, chiamata *value function*, che associa a ogni stato il suo valore sotto la politica π: $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, \pi]$, dove s_0 è lo stato al tempo 0, mentre r_{t+1} è il rinforzo ottenuto al tempo $t + 1$. È possibile

definire anche una *action-value function* $Q^\pi : S \times A \to \mathbb{R}$: $Q^\pi(s,a)$ è il valore che si ottiene eseguendo l'azione a nello stato s e seguendo poi la politica π.

La soluzione di un MDP è una *politica ottima* π^*, tale per cui $V^{\pi^*}(s) \geq V^\pi(s)$ per ogni s e per ogni π. La *value function ottima* $V^* = V^{\pi^*}$ soddisfa l'equazione di ottimalità di Bellman: $V^*(s) = max_{a \in A} \sum_{s' \in S} P(s,a,s')[R(s,a) + \gamma V^*(s')]$. Corrispondentemente, è possibile definire anche la *action-value function ottima* $Q^{\pi^*} = Q^*$, legata a V^* dalla relazione $V^*(s) = max_{a \in A} Q^*(s,a)$. Data la value function ottima, è possibile ottenere una politica ottima semplicemente agendo in modo greedy rispetto a essa.

Quanto appena esposto è solamente il caso più diffuso, ossia rinforzo scontato, in cui si definisce $V = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$; lo sconto è introdotto per evitare che, accumulando tutti i rinforzi, si ottenga un valore infinito della value function. Un'alternativa altrettanto valida che permette di ottenere questo obiettivo è il rinforzo medio, che prevede di mediare (invece che accumulare) tutti i rinforzi ottenuti: $V = \lim_{n \to \infty} \frac{r_1 + \cdots + r_n}{n}$.

L'MDP è un ottimo framework per la soluzione di piccoli problemi di controllo, mentre ha problemi a scalare su problemi di dimensioni medie e grandi. Essendo il problema di controllo di un gruppo di ascensori molto grande, è necessario adottare degli approcci che tentano di ridurre le dimensioni del problema di partenza. Nelle sezioni successive è presente una panoramica delle soluzioni proposte in letteratura per questo scopo.

2.5.1 Riduzione dell'MDP

Una strategia utilizzabile per ridurre un MDP consiste nell'aggregare alcuni suoi stati. Ovviamente, gli stati aggregati devono avere delle caratteristiche comuni, in modo da ottenere un modello equivalente, o quantomeno simile, a quello di partenza; un concetto adeguato a definire quanto due stati sono simili è quello di bisimulazione [21]. Intuitivamente, due stati sono bisimili se hanno lo stesso comportamento immediato e hanno transizioni equiprobabili verso classi equivalenti di stati. Formalmente:

Definizione 2.2. Una **relazione di bisimulazione stocastica** è una relazione di equivalenza R su S che soddisfa la seguente proprietà:

$$sRs' \Leftrightarrow \forall a \in A(R(s,a) = R(s',a) \wedge \forall C \in S/R(P(s,a,C) = P(s',a,C)))$$

dove S/R è la partizione degli stati indotta da R e $P(s,a,C) = \sum_{c \in C} P(s,a,c)$. La relazione di bisimulazione stocastica più grande è detta semplicemente **bisimulazione (stocastica)**, $\sim$.

Aggregando tutti gli stati che sono bisimili, si ottiene un MDP ridotto completamente equivalente a quello di partenza; è, quindi, possibile cercare una politica ottima per il nuovo MDP, portandola poi su quello originale con la garanzia di avere ancora una politica ottima.

La bisimulazione, però, è un concetto troppo stringente: infatti, è sufficiente una piccola variazione dell'MDP di partenza affinché due stati che erano bisimili non lo siano più. Di conseguenza, questo concetto non è adeguato al caso (comune) in cui ci sia incertezza nel modello. È, quindi, necessario definire una nozione quantitativa di bisimulazione, che dica quanto due stati sono bisimili. A tale scopo, è utile introdurre il concetto di (semi)metrica:

Definizione 2.3. Una **semimetrica** su S è una relazione $d : S \times S \to [0, \infty)$ tale che per ogni s, s', s'':

1. $s = s' \Rightarrow d(s, s') = 0$
2. $d(s, s') = d(s', s)$
3. $d(s, s'') \leq d(s, s') + d(s', s'')$

Se vale anche l'opposto del primo assioma, d è detta **metrica**.

L'utilizzo di metriche porta a un'approssimazione dei risultati ottenibili; di conseguenza, una politica ottima dell'MPD ridotto potrebbe non corrispondere a una politica ottima dell'MDP di partenza. È, comunque, possibile definire dei limiti alla perdita di prestazioni dovute all'approssimazione introdotta; in particolare, in letteratura solitamente si utilizza la differenza, in valore assoluto, tra la value function di due stati, $|V^*(s) - V^*(s')| \leq f(d(s, s'))$, dove $f(\cdot)$ è una funzione generica.

Data la natura stocastica degli MDP, una metrica adeguata deve essere una metrica tra distribuzioni; in particolare, in letteratura è stata proposta una metrica per la bisimulazione basata sulla metrica Kantorovich, sia per MDP finiti [18], sia per MDP con stati infiniti [20]. Queste metriche sono molto adeguate da un punto di vista teorico, ma in pratica risultano inutilizzabili, a causa delle difficoltà computazionali legate alla Kantorovich; per alleviare questo problema, sono state definite altre metriche, sia come rilassamento di quella basata sulla Kantorovich, sia basate sulla Total Variation, in modo da avere dei tempi computazionali più brevi, a discapito della qualità dei risultati ottenibili [19]. Inoltre, recentemente sono stati proposti dei tool per calcolare in modo più efficiente rispetto ai metodi classici la distanza tra stati con metriche basate sulla Kantorovich [1].

Finora sono state prese in considerazioni riduzioni in cui la corrispondenza tra azioni dell'MDP originale e di quello ridotto è basata semplicemente sull'etichetta dell'azione. Nei casi pratici, però, può essere utile in alcuni stati mettere in corrispondenza azioni diverse, in modo da ottenere riduzioni altrimenti non ottenibili; ad esempio, questo approccio risulta utile per tenere in considerazione le simmetrie dell'ambiente. Si consideri, a titolo illustrativo, un agente che si muove in una stanza quadrata senza ostacoli con l'obiettivo di raggiungere il centro di essa; in questo caso, è assolutamente equivalente trovarsi nell'angolo in alto a destra e muoversi di un'unità verso sinistra, oppure trovarsi nell'angolo in basso a sinistra e muoversi di

un'unità verso destra. Un concetto utile per tenere in considerazione questo tipo di riduzione è quello di bisimulazione rilassata [63]:

Definizione 2.4. Una **relazione di bisimulazione stocastica rilassata** è una relazione di equivalenza R su S che soddisfa la seguente proprietà:

$$sRs' \Leftrightarrow \forall a \in A \exists a' \in A(R(s,a) = R(s',b) \wedge \forall C \in S/R(P(s,a,C) = P(s',a,C)))$$

dove S/R è la partizione degli stati indotta da R e $P(s,a,C) = \sum_{c \in C} P(s,a,c)$. L'unione di tutte le relazioni di bisimulazione rilassata è detta semplicemente **bisimulazione rilazzata**, $\sim_L$.

Anche in questo caso è utile definire una metrica, per gli stessi motivi riportati in precedenza; in particolare, in [63] ne è stata definita una che sfrutta la metrica Hausdorff [44], ricavando dei limiti alla perdita di prestazioni dovute all'approssimazione introdotta.

Un altro concetto che tenta di definire l'equivalenza tra stati di un MDP è quello di equivalenza per traiettoria [8]. Intuitivamente, due stati sono equivalenti per traiettoria se generano la stessa distribuzione di probabilità condizionata sulle traiettorie osservabili del sistema. Formalmente:

Definizione 2.5. Due stati $s, s' \in S$ di un MDP sono **equivalenti per traiettoria** se e solo se:

$$\forall \theta \in \Theta, \forall \alpha \in (\mathbb{R} \times S)^*(P(s,\theta,\alpha) = P(s',\theta,\alpha))$$

dove $\theta : \mathbb{N}^+ \to A$ è una funzione, detta sequenza di azioni, che mette in corrispondenza timestamp con azioni, Θ è l'insieme di tutte le sequenze di azioni, α è una traiettoria finita rinforzo-stato, $P(s,\theta,\alpha)$ è la probabilita di osservare α partendo da s e scegliendo le azioni specificate da θ.

Sebbene a prima vista i concetti di bisimulazione ed equivalenza per traiettoria possano sembrare equivalenti, in realtà non è così; in particolare, equivalenza per traiettoria implica bisimulazione, ma bisimulazione non implica equivalenza per traiettoria. Risulta, quindi, utile cercare di dare una definizione alternativa di equivalenza per traiettoria:

Definizione 2.6. Data una decomposizione $\Psi(S)$, due stati $s, s' \in S$ di un MDP sono **Ψ-equivalenti per traiettoria** se e solo se:

$$\psi(s) = \psi(s') \wedge \forall \theta \in \Theta, \forall \kappa \in (\mathbb{R} \times \Psi(S))^*(P(s,\theta,\kappa) = P(s',\theta,\kappa))$$

dove θ e Θ sono come in precedenza, κ e $P(s,\theta,\kappa)$ sono i corrispondenti di α e $P(s,\theta,\alpha)$.

Definendo Ψ_R in modo che $\psi_R(s) = \psi_R(s') \Leftrightarrow \forall a \in A(R(s,a) = R(s',a))$, si ha che bisimulazione implica Ψ_R-equivalenza per traiettoria, ma Ψ_R-equivalenza

per traiettoria non implica bisimulazione. Definendo poi un operatore Γ tale che $\forall D \subseteq S, D \in \Gamma(\Psi(S)) \Leftrightarrow \forall s, s' \in D$, s e d sono Ψ-equivalenti per traiettoria e $\forall a \in A(R(s,a) = R(s',a))$, definendo Γ^* come il punto fisso delle iterazioni di Γ, si ha che Γ^*-equivalenza implica bisimulazione e bisimulazione implica Γ^*-equivalenza.

Un altro concetto utile per la riduzione di un MDP è quello di omomorfismo tra MDP [50]:

Definizione 2.7. Un **omomorfismo tra MDP** h da un MDP $M =< S, A, P, R, \gamma, \mu >$ a un MDP $M' =< S', A', P', R', \gamma', \mu' >$ è una suriezione da $S \times A$ a $S' \times A'$, definita da una tupla di suriezioni $< f, g_s | s \in S >$, con $h((s,a)) = (f(s), g_s(a))$, dove $f : S \to S'$ e $g_s : A \to A'$ per $s \in S$, tali che $\forall s, s' \in S, a \in A$:

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s'')$$

$$R'(f(s), g_s(a)) = R(s, a)$$

dove $[s']_f$ indica la classe della partizione indotta da f a cui appartiene s'. Inoltre, $\gamma = \gamma'$, mentre la relazione tra μ e μ' deriva come conseguenza di f.

È, quindi, possibile ridurre un MDP M trovando un altro MDP M', a esso omomorfo; come si può evincere dalla definizione precedente, questa riduzione tiene anche in considerazione equivalenze con azioni diverse. Una volta trovata una politica π per M', la politica corrispondente per M, π'_M, è definita in modo che $\forall a \in g_s^{-1}(a'), \pi'_M(s,a) = \pi'(f(s), a')/|g_s^{-1}(a')|$, dove $g_s^{-1}(a')$ è l'insieme delle azioni che hanno la stessa immagine $a' \in A'$ sotto g_s. Essendo l'omomorfismo tra MDP una relazione esatta, una politica ottima π'^* per M' induce una politica π'^*_M che è ottima per M.

La relazione di omomorfismo, in quanto impone una corrispondenza esatta tra l'MDP di partenza e quello ridotto, è troppo forte per poter ottenere, nella pratica, delle riduzioni significative. Per sopperire a questo problema, è possibile usare un omomorfismo approssimato [51]:

Definizione 2.8. Un **omomorfismo approssimato tra MDP** h da un MDP $M =< S, A, P, R, \gamma, \mu >$ a un MDP $M' =< S', A', P', R', \gamma', \mu' >$ è una suriezione da $S \times A$ a $S' \times A'$, definita da una tupla di suriezioni $< f, g_s | s \in S >$, con $h((s,a)) = (f(s), g_s(a))$, dove $f : S \to S'$ e $g_s : A \to A'$ per $s \in S$, tali che $\forall s, s' \in S, a \in A$:

$$P'(f(s), g_s(a), f(s')) = \sum_{(q,b) \in [(s,a)]_h} w_{qb} \sum_{s'' \in [s']_f} P(q, b, s'')$$

$$R'(f(s), g_s(a)) = \sum_{(q,b) \in [(s,a)]_h} w_{qb} R(q, b)$$

dove $[s']_f$ come prima, mentre w_{qb} è un fattore di peso per la coppia (q,b) con $\sum_{(q,b) \in [(s,a)]_h} w_{qb} = 1$. Inoltre, $\gamma = \gamma'$, mentre la relazione tra μ e μ' deriva come conseguenza di f.

Ovviamente, essendo la corrispondenza approssimata, non è detto che, data una politica π'^* ottima per M', la politica corrispondente π'^*_M sia ottima per M. È, però, possibile definire un limite alla perdita di prestazioni, in termini di $\|V^* - V^{\pi'^*_M}\| \leq K(h)$, dove $K(h)$ è un valore che dipende dall'omomorfismo approssimato.

Il concetto di omomorfismo ha una stretta corrispondenza con quello di bisimulazione. Infatti, non considerando la trasformazione delle azioni, cioè imponendo $g_s(a) = a \quad \forall s \in S$, bisimulazione e omomorfismo sono due concetti equivalenti [50]; invece, tenendo in considerazione la trasformazione delle azioni, si ha che omomorfismo e bisimulazione rilassata sono due concetti equivalenti [63].

Un ulteriore metodo per mettere in corrispondenza due MDP è basato sui concetti di ϵ-equivalenza e partizione ϵ-omogenea [17]:

Definizione 2.9. Due MDP $M1 =< S_1, A, R_1, P_1, \gamma, \mu_1 >$ e $M2 =< S_2, A, R_2, P_2, \gamma, \mu_2 >$ sono **ϵ-equivalenti** rispetto a una norma L_k se esistono due suriezioni $\phi_1 : S_1 \to S$ e $\phi_2 : S_2 \to S$ rispetto a un MDP $M =< S, A, R, P, \gamma, \mu >$, tali che $\forall s_1 \in S_1, s_2 \in S_2$ tali che $\phi_1(s_1) = \phi_2(s_2) = s$ valgono le seguenti relazioni:

$$\forall a \in A \quad (\sum_{s' \in S} (\sum_{\widetilde{s}:\phi(\widetilde{s})=s'} P_i(s_i, a, \widetilde{s}) - P(s, , as'))^k)^{1/k} \leq \epsilon \quad i = 1, 2$$

$$\forall a \in A \quad |R_i(s_i, a) - R(s, a)| \leq \epsilon \quad i = 1, 2$$

mentre la relazione tra μ, μ_1 e μ_2 viene di conseguenza.

Definizione 2.10. Un MDP $M1 =< S_1, A, R_1, P_1, \gamma, \mu_1 >$ è una **partizione ϵ-omogenea** di un MDP $M =< S, A, R, P, \gamma, \mu >$ rispetto a una norma L_k se esiste una suriezione $\phi : S \to S_1$ tale che $\forall s$ valgono le seguenti relazioni:

$$\forall a \in A \quad (\sum_{s' \in S} (\sum_{\widetilde{s}:\phi(\widetilde{s})=s'} P(s, a, \widetilde{s}) - P_1(\phi(s), a, s'))^k)^{1/k} \leq \epsilon$$

$$\forall a \in A \quad \max_{\widetilde{s}:\phi(\widetilde{s})=s'} |R(\widetilde{s}, a) - R_1(s, a)| \leq \epsilon$$

mentre la relazione tra μ, μ_1 e μ_2 viene di conseguenza.

Dati un MDP M_ϵ ϵ-omogeneo a un MDP M e una politica π in M_ϵ, la politica π^M indotta su M è definita come $\pi^M(s) = \pi(\phi(s))$. In generale, una politica ottima in M_ϵ può indurre una politica non ottima in M, senza nessuna garanzia sulla distanza tra le due politiche. Utilizzando, però, una partizione ϵ-omogenea rispetto alla norma L_1, definendo una politica π ϵ-ottima se $\|V^* - V^\pi\| \leq \epsilon$, si ha la garanzia che una politica ottima in M_ϵ induce una politica $\frac{2\epsilon V_{max}}{1-\gamma}$-ottima in M.

Un concetto simile, ma riferito al caso average reward, è quello di partizione ϵ-adeguata [47]:

Definizione 2.11. Dato un MDP $M =< S, A, R, P, \gamma, \mu >$, $\widehat{S} = \{S_1, \ldots, S_k\}$ è un **partizione ϵ-adeguata** di S se è una sua partizione e per ogni s,s' nello stesso blocco S_j si ha, per qualche valore di c_r e c_p, che:

$$c_r|R(s,a) - R(s',a)| \leq \epsilon$$
$$c_p|\sum_{s'' \in S_i} P(s,a,a'') - \sum_{s'' \in S_i} P(s',a,s'')| \leq \epsilon \quad \forall S_i \in \widehat{S}$$

Dato un MDP e una sua riduzione ottenuta con una partizione ϵ-omogenea dei suoi stati si ha la garanzia che la differenza di performance dei due modelli per una politica qualsiasi π è boundata dal valore $\left(\frac{1}{c_r} + \frac{(K_\pi - 1)|\widehat{S}}{c_p}\right)\epsilon$, dove K_π è il mixing time della catena originale seguendo la politica π; la differenza di performance nel modello originale tra la politica ottima del modello originale e quella del modello ridotto è, invece, $\left(\frac{2}{c_r} + \frac{2(K_\pi - 1)|\widehat{S}}{c_p}\right)\epsilon$. Questi limiti, comunque, sono decisamente peggiori di quelli che si possono garantire nel caso discounted.

2.5.2 MDP fattorizzati

Quanto visto fino a ora utilizza stati e azioni atomici. Nella pratica, però, questi due componenti solitamente hanno una struttura, che può essere sfruttata per definire un modello più semplice e ridotto. Una possibile estensione degli MDP finalizzata a incorporare la struttura di stati e azioni è rappresentata dagli MDP fattorizzati [5]:

Definizione 2.12. Un **MDP fattorizzato** è una tupla $< S, A, P, R, \gamma, \mu >$, dove $S = S_1 \times \cdots \times S_n$ è l'insieme degli stati, $A = A_1 \times \cdots \times A_m$ è l'insieme delle azioni, mentre gli altri parametri sono definiti come per un normale MDP.

In pratica, quindi, un MDP fattorizzato è un MDP in cui stati e azioni sono composti rispettivamente da n e m variabili, dette *feature*. In questo modo, quindi, si ottiene una rappresentazione più compatta, in quanto non è necessario elencare esplicitamente tutti gli stati/azioni, che sono in numero esponenziale rispetto al numero di feature.

Ovviamente, affinché la riduzione sia effettiva, occorre utilizzare un metodo opportuno per rappresentare la funzione di transizione e quella di rinforzo. A questo proposito, un formalismo adeguato è rappresentato dalle reti bayesiane dinamiche a due strati. In un MDP finito, riferendosi alla funzione di transizione degli stati[1], si ha una rete per ogni azione, in cui i nodi del primo strato rappresentano lo stato prima dell'azione, quelli del secondo strato rappresentano lo stato dopo l'azione, mentre gli archi diretti, da nodi del primo strato a nodi del secondo strato, rappresentano l'influenza causale tra le varie variabili di stato. Inoltre, è utile rappresentare le matrici di probabilità condizionata come alberi di decisione, in modo da poter rappresentare

[1] La struttura della rete per la funzione di rinforzo si ottiene in modo simile, con le opportune sostituzioni

anche l'indipendenza tra gli assegnamenti di variabili; in questi alberi, ogni nodo interno determina un assegnamento di una variabile, mentre le foglie determinano la probabilità in questione.

In un MDP fattorizzato continuo, invece, la rappresentazione appena descritta non risulta essere adeguata [67]. In questo caso, le $|A|$ reti bayesiane dinamiche possono essere sostituite da un'unica rete bayesiana dinamica, il cui primo strato contiene, oltre alle variabili di stato, anche quelle di azione (che, quindi, a differenza del caso precedente, in cui era presente una rete per ogni azione, vengono utilizzate esplicitamente), mentre il resto della rete è costruita con lo stesso criterio del caso finito. L'albero di decisione, invece, può essere sostituito da una funzione di densità di probabilità condizionata, come ad esempio una mistura di gaussiane.

Per risolvere un MDP fattorizzato è possibile utilizzare gli stessi algoritmi utilizzati per MDP senza struttura, ma in questo modo la struttura non viene sfruttata; in letteratura, quindi, sono stati definiti degli algoritmi risolutivi progettati appositamente per lavorare su MDP fattorizzati [26]. Per poter utilizzare questi algoritmi, però, è necessario avere una rappresentazione esplicita della rete bayesiana dinamica, cosa che spesso nella pratica non si ha. Per sopperire a questi problema, sono stati studiati degli algoritmi che cercano di apprendere online la struttura del problema, sia per MDP finiti [58], sia per MDP continui [67]; inoltre, recentemente si è cercato anche di proporre algoritmi che, mentre cercano di estrarre la struttura presente nel problema, utilizzino il miglior compromesso possibile tra esplorazione e sfruttamento dei dati [52].

Nonostante l'utilizzo di una rappresentazione fattorizzata permetta di ottenere un modello più compatto, spesso, nella pratica, il modello ottenuto risulta essere comunque intrattabile; risulta, quindi, necessario proporre anche in questo caso delle tecniche di riduzione del modello. A questo scopo, è possibile estendere i concetti visti nel paragrafo precedente agli MDP fattorizzati: ad esempio, è possibile utilizzare la bisimulazione sia per riduzioni esatte [12], sia per riduzioni approssimate [13]; oppure, è possibile definire omomorfismi tra MDP fattorizzati [50]. In questo contesto, però, esiste un metodo più semplice ed effettivo per ridurre il modello: la selezione delle feature.

2.5.3 Selezione delle Feature

Definizione 2.13. Il problema di **selezione delle feature** di un MDP M consiste nel ricavare un altro MDP M' le cui feature sono un sottoinsieme di quelle di M; in particolare, le variabili di M eliminate sono quelle più irrilevanti.

Selezionando solamente alcune variabili, si ottiene una riduzione del modello tramite aggregazione di stati: il modello ridotto, infatti, aggrega tutti gli stati con uguale valore delle feature selezionate e valore qualsiasi di quelle non selezionate.

La definizione precedente è piuttosto generica e non dice né come effettuare la selezione, né quale criterio può essere utilizzato per misurare la rilevanza di una feature. Una possibile procedura utilizzabile per effettuare questa selezione è la seguente [7]: inizialmente, si selezionano tutte le variabili (di stato e di azione) necessarie a spiegare il rinforzo; poi, per ognuna di queste variabili, si ripete la procedura, in modo da spiegare esse stesse, e si continua in questo modo fino a quando tutte le variabili selezionate non sono state spiegate.

Selezionando tutte le variabili necessarie a spiegare esattamente il rinforzo o le altre variabili già selezionate, si ottiene un MDP ridotto M' equivalente a quello di partenza M; la politica ottima di M' induce, di conseguenza, una politica ottima su M. Nella pratica, però, una riduzione esatta, come già sottolineato per le tecniche precedenti, non porta a riduzioni significative. Risulta, quindi, necessario fornire dei criteri per ordinare le variabili da quella più significativa a quella meno significativa per la spiegazione in atto e scegliere, in base a questo ordinamento, solo quelle più significative. Inoltre, bisognerebbe evitare si selezionare variabili ridondanti. A tal proposito, è possibile costruire un modello predittivo per la variabile da spiegare basato su Extra-Trees e ordinare le variabili in base alla riduzione di varianza che comporterebbe una loro selezione, inserire nel modello solamente la prima classificata e ripetere, fermandosi quando non si ottiene più alcun miglioramento della capacità predittiva.

Quello appena descritto è solo uno dei tanti metodi proposti in letteratura per fare selezione delle feature. Ad esempio, in [22] è proposto un metodo molto simile a quello appena illustrato, ma che sfrutta il modello creato dall'algoritmo fitted Q iteration durante la sua esecuzione. Un'altra possibilità consiste nell'assegnare a ogni variabile un punteggio basato sull'interazione tra la variabile e l'azione intrapresa e sulla proporzione di dati per cui la scelta della variabile ottima cambia se si conosce o no la variabile in questione, selezionando poi le migliori di esse [27]. Un'ulteriore alternativa consiste nell'ordinare le variabili in base alla misura dell'informazione reciproca per variabili stocastiche tra la variabile in questione e il rinforzo, scegliendo poi le variabili da tenere in base a criteri di selezione in avanti ed eliminazione all'indietro [28]. Oppure, ancora, è possibile avere una selezione delle feature fatta online, con anche il numero di variabili selezionate che aumenta, se necessario, all'aumentare dei dati visti, basandosi su una misura della dipendenza tra il cambiamento di stato e l'azione intrapresa [39].

In letteratura, quindi, sono stati proposti parecchi metodi, più o meno diversi tra loro, per fare selezione delle feature. In tutti i casi, si è mostrato la loro utilità e bontà attraverso esempi pratici; per nessuno di essi, però, è stato fornito un limite sulla perdita di prestazioni che si ha utilizzando una politica indotta da una politica ottima per il modello ridotto. Di conseguenza, quando si utilizza uno degli algoritmi proposti, si può solamente sperare di ottenere una buona riduzione, ma non si ha nessuna garanzia teorica in proposito.

La selezione delle feature può essere combinata con le tecniche di riduzione degli MDP viste in precedenza. Per esempio, in [70] essa viene utilizzata in combinazione con l'omomorfismo, anche se il tutto viene applicato non agli MDP, ma ai Controlled Markov Process, che, a differenza degli MDP, hanno una funzione di uscita e non un rinforzo, in modo da aver maggiore riutilizzabilità (infatti, dalla funzione di uscita è possibile derivare funzioni di rinforzo differenti a seconda delle necessità). Inoltre, a differenza dei metodi descritti in precedenza, in questo caso il modello utilizzato per fare selezione delle feature, basato su alberi di decisione, viene utilizzato per predire l'etichetta dello stato.

Capitolo 3

Selezione delle feature

Il controllo di un gruppo di ascensori è un problema molto complesso e di dimensioni elevate, che scalano col numero di ascensori e di piani presenti nel sistema; per questo motivo, soprattutto per il controllo di gruppi di ascensori installati in edifici grandi, non è possibile utilizzare tutte le informazioni effettivamente disponibili, ma serve un metodo per selezionare solamente le informazioni più significative ai fini del controllo.

In letteratura sono presenti molti algoritmi di selezione delle feature (o, equivalentemente, delle variabili), come presentato nella Sezione 2.5. In questi algoritmi, però, si notano generalmente due difetti: sono tutti completamente euristici e, quindi, non forniscono nessun tipo di garanzia sulla bontà della selezione che comportano, ma sono supportati solamente da risultati pratici; selezionano generalmente un insieme ritenuto utile delle variabili, ma non mostrano quali sono le dipendenze che hanno portato a effettuare questa selezione. In questo capitolo, quindi, analizziamo il campo della selezione delle feature, con l'obiettivo di fornire degli algoritmi che superino i problemi evidenziati; in particolare, proponiamo due algoritmi differenti: uno, che chiameremo algoritmo garantito, seleziona le variabili fornendo alcune garanzie; l'altro, che chiameremo euristico e sviluppa il lavoro preliminare riportato in [7], seleziona le variabili in base a dei criteri euristici, generando durante la selezione l'albero di dipendenza delle variabili.

Per studiare la selezione delle feature formalizziamo il sistema da ridurre tramite Markov Decision Process (MDP). Nella pratica, selezionando solo alcune variabili del problema di partenza, passiamo dall'MDP originale a un MDP ridotto, da esso ottenuto.

La prossima sezione analizza le trasformazioni necessarie per passare dall'MDP originale a una sua versione ridotta, ottenuta eliminando alcune variabili dal problema, che soddisfi alcune proprietà desiderate. In seguito, presentiamo un problema giocattolo e i due algoritmi di selezione delle feature proposti, analizzandoli attraverso il problema giocattolo. Infine, analizziamo alcuni aspetti relativi alla dipendenza tra variabili e alla loro controllabilità.

3.1 Trasformazione dell'MDP

Formalizziamo ora la trasformazione utilizzata per ridurre l'MDP, in cui l'insieme degli stati e delle azioni vengono partizionati per creare dei nuovi insiemi con minor cardinalità; questo, infatti, corrisponde a utilizzare solamente alcune variabili, in quanto tutti gli stati (azioni) che hanno ugual valore delle variabili di stato (azione) mantenute e, eventualmente, valore diverso delle variabili di stato (azione) eliminate vengono aggregati.

L'analisi non è effettuabile utilizzando solamente l'MDP originale e quello ridotto, in quanto le varie quantità vettoriali, essendo di dimensionalità diversa, non sono facilmente confrontabili. Risulta, quindi, necessario introdurre un terzo MDP, che sia equivalente a quello ridotto ma abbia le stesse dimensionalità di quello originale; chiamiamo questo MDP riespansione del modello ridotto. Così facendo, infatti, è possibile confrontare direttamente le quantità del modello originale e di quello riespanso, generalizzando i risultati ottenuti al confronto tra modello originale e modello ridotto.

La notazione utilizzata di seguito è la seguente:

- $\mathfrak{R}$ insieme di tutte le riduzioni possibili, $\rho \in \mathfrak{R}$ una riduzione particolare;
- $M =< S, A, P, R, \gamma, \mu >$ modello originale, $M_\rho =< S_\rho, A_\rho, P_\rho, R_\rho, \gamma, \mu_\rho >$ modello ridotto secondo la riduzione ρ, $M_\rho^e =< S_{\rho^e}, A_{\rho^e}, P_{\rho^e}, R_{\rho^e}, \gamma, \mu_{\rho^e} >$ riespansione di M_ρ;
- $X(M)$ quantità X calcolata nel modello M, X_τ quantità X calcolata fino al tempo τ;
- π^* politica ottima di M, π_ρ^* di M_ρ, π_ρ^{*e} espansione della politica ottima di M_ρ (in generale, π^e è l'espansione di π);
- J expected return, V value function;
- γ discount factor;
- $[s]$ aggregato che contiene lo stato s, $[a]$ aggregato che contiene l'azione a, $\#[s]$ numero di stati contenuti nell'aggregato $[s]$, $\#[a]$ numero di azioni contenuti nell'aggregato $[a]$.

L'espansione π^e di una politica π prevede che a tutti gli stati componenti un aggregato venga assegnata la probabilità assegnata da π all'aggreagato; in formule:

$$\pi^e(s, a) = \frac{\pi(\overline{s}, \overline{a})}{\#[\overline{a}]} \qquad \forall s \in [\overline{s}], a \in [\overline{a}]$$

3.1.1 Definizione del modello ridotto

Intuitivamente, il metodo proposto che, data una riduzione $\rho \in R$, permette di passare dal modello originale M al modello ridotto M_ρ, catturando l'idea di aggregazione di stati, prevede che in M_ρ la distribuzione iniziale di un aggregato è data dalla somma dei vari componenti, mentre la probabilità di transizione e il rinforzo di un aggregato sono ottenuti mediando quella dei componenti, in quanto non è possibile sapere in quale stato originale si è realmente.

Formalizziamo ora l'idea espressa a parole. Per il momento supponiamo, per semplicità, di fissare una politica $\pi \in \Pi_\rho$ per il modello ridotto[1]. La trasformazione che permette di passare da M a M_ρ è la seguente:

$$\mu_\rho(\overline{s}) = \sum_{s \in [\overline{s}]} \mu(s) \qquad \forall\, \overline{s} \in S_\rho$$

$$P^\pi_\rho(\overline{s}, \overline{s}') = \frac{1}{\#[\overline{s}]} \sum_{\substack{s \in [\overline{s}] \\ s' \in [\overline{s}']}} P^{\pi^e}(s, s') \qquad \forall\, \overline{s}, \overline{s}' \in S_\rho$$

$$R^\pi_\rho(\overline{s}) = \frac{1}{\#[\overline{s}]} \sum_{s \in [\overline{s}]} R^{\pi^e}(s) \qquad \forall\, \overline{s} \in S_\rho$$

Per generalizzare le equazioni appena scritte bisogna aggiungere un vincolo per cui esse siano valide per ogni $\pi \in \Pi_\rho$; questo equivale a richiedere che equazioni simili, ma più complicate per la presenza dell'azione, siano valide per tutte le azioni. Esplicitiamo di seguito queste equazioni, in cui si evidenzia anche l'effetto di eventuali aggregazioni sulle azioni (effetto che, invece, rimane sottointeso nella politica nella formulazione precedente):

$$\mu_\rho(\overline{s}) = \sum_{s \in [\overline{s}]} \mu(s) \qquad \forall\, \overline{s} \in S_\rho$$

$$P_\rho(\overline{s}, \overline{a}, \overline{s}') = \frac{1}{\#[\overline{s}]\#[\overline{a}]} \sum_{\substack{s \in [\overline{s}] \\ s' \in [\overline{s}'] \\ a \in [\overline{a}]}} P(s, a, s') \qquad \forall\, \overline{s}, \overline{s}' \in S_\rho, \overline{a} \in A_\rho$$

$$R_\rho(\overline{s}, \overline{a}) = \frac{1}{\#[\overline{s}]\#[\overline{a}]} \sum_{\substack{s \in [\overline{s}] \\ a \in [\overline{a}]}} R(s, a) \qquad \forall\, \overline{s} \in S_\rho, \overline{a} \in A_\rho$$

3.1.2 Definizione della riespansione del modello ridotto

La sola definizione di M_ρ non è sufficiente, dato che non ha senso confrontare le quantità vettoriali espresse nell'MDP originale con quelle espresse nell'MDP ridotto, in quanto le loro dimensioni sono differenti; ad esempio, scrivere $R^{\pi^e} = R^\pi_\rho$ non

[1] Non ha senso considerare politiche non nel modello ridotto, in quanto comunque, lavorando sul modello ridotto, queste politiche non possono essere apprese e, quindi, non sono significative.

ha alcun significato. Quindi, affinché abbia senso una scrittura di questo tipo, è necessario riportare le quantità espresse nel modello ridotto in quantità analoghe espresse nel modello originale. In particolare, risulta utile rimappare l'MDP ridotto, M_ρ, nel modello originale, dando origine a un nuovo modello, M_ρ^e; ovviamente, perché questa trasformazione sia utile, bisogna garantire che essa mantenga inalterate alcune proprietà di interesse. In particolare, per i nostri scopi dobbiamo garantire che risolvendo M_ρ^e e poi riaggregando si ottengono gli stessi risultati che si otterrebbero lavorando direttamente su M_ρ.

Formalmente, la proprietà espressa a parole è la seguente:

$$J^{\pi^e}(M_\rho^e) = J^\pi(M_\rho) \qquad \forall \pi \in \Pi_\rho$$

Per mantenere questa proprietà è sufficiente che l'espansione sia sensata, cioè che le probabilità di un aggregato vengano ripartite in un qualche modo tra gli stati appartenenti all'aggregato e che il rinforzo di un aggregato venga assegnato a tutti gli stati contenuti nell'aggregato. Il seguente teorema formalizza e dimostra quanto appena esposto:

Teorema 3.1.1. *Per garantire che* $J^{\pi^e}(M_\rho^e) = J^\pi(M_\rho)\ \forall \pi \in \Pi_\rho$ *è sufficiente che le seguenti proprietà siano verificate:*

$$\sum_{s \in [\overline{s}]} \mu_{\rho^e}(s) = \mu_\rho(\overline{s})$$

$$\sum_{s' \in [\overline{s}']} P_{\rho^e}^{\pi^e}(s, s') = P_\rho^\pi(\overline{s}, \overline{s}') \qquad \forall\ s \in [\overline{s}], \pi \in \Pi_\rho$$

$$R_{\rho^e}^{\pi^e}(s) = R_\rho^\pi(\overline{s}) \qquad \forall\ s \in [\overline{s}], \pi \in \Pi_\rho$$

Dimostrazione. Si scriva la performance di M_ρ con una generica politica $\pi \in \Pi_\rho$ come:

$$\begin{aligned} J^\pi(M_\rho) =& d_{\mu_\rho}^{\pi T}(M_\rho) R_\rho^\pi \\ =& \sum_{s \in S_\rho} [(\mu_\rho(s) + \gamma \sum_{s_1 \in S_\rho} \mu_\rho(s_1) P_\rho^\pi(s_1, s) + \\ & \gamma^2 \sum_{s_1, s_2 \in S_\rho} \mu_\rho(s_1) P_\rho^\pi(s_1, s_2) P_\rho^\pi(s_2, s) + \dots) R_\rho^\pi(s)] \end{aligned}$$

Similmente, per M_ρ^e:

$$\begin{aligned} J^{\pi^e}(M_\rho^e) =& d_{\mu_{\rho^e}}^{\pi^e T}(M_\rho^e) R_{\rho^e}^{\pi^e} \\ =& \sum_{s \in S_{\rho^e}} [(\mu_{\rho^e}(s) + \gamma \sum_{s_1 \in S_{\rho^e}} \mu_{\rho^e}(s_1) P_{\rho^e}^{\pi^e}(s_1, s) + \\ & \gamma^2 \sum_{s_1, s_2 \in S_{\rho^e}} \mu_{\rho^e}(s_1) P_{\rho^e}^{\pi^e}(s_1, s_2) P_{\rho^e}^{\pi^e}(s_2, s) + \dots) R_{\rho^e}^{\pi^e}(s)] \end{aligned}$$

Spezzando le sommatorie in modo da sommare su tutti gli stati aggregati e poi su tutti gli stati che compongono l'aggregato, portando dentro le sommatorie in modo da poter effettuare le sostituzioni espresse nel teorema, si ha:

$$
\begin{aligned}
J^{\pi}(M^e_\rho) = &\sum_{\overline{s}\in S_\rho}\sum_{s\in[\overline{s}]}\mu_{\rho^e}(s)R^{\pi}_{\rho}(\overline{s}) + \sum_{\overline{s}\in S_\rho}\sum_{s\in[\overline{s}]}\gamma\sum_{\overline{s}_1\in S_\rho}\sum_{s_1\in[\overline{s}_1]}\mu_{\rho^e}(s_1)P^{\pi^e}_{\rho^e}(s_1,s)R^{\pi}_{\rho}(\overline{s}) + \\
&+\sum_{\overline{s}\in S_\rho}\sum_{s\in[\overline{s}]}\gamma^2\sum_{\overline{s}_1,\overline{s}_2\in S_\rho}\sum_{\substack{s_1\in[\overline{s}_1]\\ s_2\in[\overline{s}_2]}}\mu_{\rho^e}(s_1)P^{\pi^e}_{\rho^e}(s_1,s_2)P^{\pi^e}_{\rho^e}(s_2,s)R^{\pi}_{\rho}(\overline{s}) + \cdots = \\
= &\sum_{\overline{s}\in S_\rho}\mu_{\rho}(\overline{s})R^{\pi}_{\rho}(\overline{s}) + \sum_{\overline{s}\in S_\rho}\gamma\sum_{\overline{s}_1\in S_\rho}\sum_{s_1\in[\overline{s}_1]}\mu_{\rho^e}(s_1)P^{\pi}_{\rho}(\overline{s}_1,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \\
&+\sum_{\overline{s}\in S_\rho}\gamma^2\sum_{\overline{s}_1,\overline{s}_2\in S_\rho}\sum_{\substack{s_1\in[\overline{s}_1]\\ s_2\in[\overline{s}_2]}}\mu_{\rho^e}(s_1)P^{\pi^e}_{\rho^e}(s_1,s_2)P^{\pi}_{\rho}(\overline{s}_2,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \cdots = \\
= &\sum_{\overline{s}\in S_\rho}\mu_{\rho}(\overline{s})R^{\pi}_{\rho}(\overline{s}) + \sum_{\overline{s}\in S_\rho}\gamma\sum_{\overline{s}_1\in S_\rho}\mu_{\rho}(\overline{s}_1)P^{\pi}_{\rho}(\overline{s}_1,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \\
&+\sum_{\overline{s}\in S_\rho}\gamma^2\sum_{\overline{s}_1,\overline{s}_2\in S_\rho}\sum_{s_1\in[\overline{s}_1]}\mu_{\rho^e}(s_1)P^{\pi}_{\rho}(\overline{s}_1,\overline{s}_2)P^{\pi}_{\rho}(\overline{s}_2,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \cdots = \\
= &\sum_{\overline{s}\in S_\rho}\mu_{\rho}(\overline{s})R^{\pi}_{\rho}(\overline{s}) + \sum_{\overline{s}\in S_\rho}\gamma\sum_{\overline{s}_1\in S_\rho}\mu_{\rho}(\overline{s}_1)P^{\pi}_{\rho}(\overline{s}_1,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \\
&+\sum_{\overline{s}\in S_\rho}\gamma^2\sum_{\overline{s}_1,\overline{s}_2\in S_\rho}\mu_{\rho}(\overline{s}_1)P^{\pi}_{\rho}(\overline{s}_1,\overline{s}_2)P^{\pi}_{\rho}(\overline{s}_2,\overline{s})R^{\pi}_{\rho}(\overline{s}) + \ldots
\end{aligned}
$$

È chiaro che la dimostrazione è equivalente per tutti gli (infiniti) termini lasciati sottointesi, arrivando così a concludere la tesi. □

L'equivalenza proposta permette un numero infinito di trasformazioni; per averne una unica proponiamo il seguente metodo:

- Si riespandono tutti gli stati aggregati, in modo da tornare alla cardinalità degli stati originale;
- Le transizioni entranti in un aggregato vengono ripartite uniformemente tra tutti gli stati espansi dall'aggregato; in questo modo si introduce la maggior quantità di casualità possibile, permettendo alla catena di mixare più velocemente;
- Il rinforzo di un aggregato viene assegnato a tutti gli stati espansi dall'aggregato;
- La distribuzione iniziale μ_{ρ^e} viene posta uguale a quella del problema originale, μ, in modo da creare il minor numero possibile di differenze con il modello originale.

In formule:

$$\begin{aligned} \mu_{\rho^e}(s) &= \mu(s) && \forall\, s \in S \\ P_{\rho^e}(s,a,s') &= \frac{1}{\#[\overline{s}']} P_\rho(\overline{s},\overline{a},\overline{s}') && \forall\, s \in [\overline{s}], s' \in [\overline{s}'], a \in [\overline{a}] \\ R_{\rho^e}(s,a) &= R_\rho(\overline{s},\overline{a}) && \forall\, s \in [\overline{s}], a \in [\overline{a}] \end{aligned}$$

È facile verificare che questa scelta soddisfa le condizioni esposte nel Teorema 3.1.1; in particolare si noti che in questa formalizzazione non compare nessuna politica, ma compaiono anche le azioni (e eventuali aggregazioni su esse), con trasformazioni scelte in modo che le condizioni dimostrate nel teorema citato valgano per tutte le politiche.

Il processo complessivo di riduzione del modello attraverso eliminazione di variabili, cioè attraverso aggregazione di stati e azioni, prevede, quindi, di ridurre M ottenendo M_ρ, che viene a sua volta espanso in M_ρ^e per poterlo confrontare con M; in pratica, quindi, la riduzione del modello diventa una modifica della probabilità delle transizioni e del rinforzo. È possibile formalizzare questa affermazione con delle equazioni che permettono di passare direttamente da M a M_ρ^e:

$$\begin{aligned} \mu_{\rho^e}(s) &= \mu(s) && \forall\, s \in S \\ P_{\rho^e}(s,a,s') &= \frac{1}{\#[\overline{s}]\#[\overline{s}']\#[\overline{a}]} \sum_{\substack{s_1 \in [\overline{s}] \\ s_1' \in [\overline{s}'] \\ a_1 \in [\overline{a}]}} P(s_1,a_1,s_1') && \forall\, s \in [\overline{s}], s' \in [\overline{s}'], a \in [\overline{a}] \\ R_{\rho^e}(s,a) &= \frac{1}{\#[\overline{s}]\#[\overline{a}]} \sum_{\substack{s_1 \in [\overline{s}] \\ a_1 \in [\overline{a}]}} R(s_1,a_1) && \forall\ \ s \in [\overline{s}], a \in [\overline{a}] \end{aligned}$$

Risulta facile verificare che queste formule per la trasformazione diretta sono coerenti con le formule precedenti, cioè danno lo stesso risultato ottenibile applicando prima quelle di aggregazione e poi quelle di espansione.

3.1.3 Proprietà

In questa sezione presentiamo alcune proprietà relative alla trasformazione descritta in precedenza.

Lemma 3.1.2. *Nell'espansione del modello ridotto, data una qualsiasi politica $\pi \in \Pi_\rho$, il numero scontato di visite di un aggregato viene ripartito uniformemente tra i vari stati contenuti in esso:*

$$d^{\pi^e}_{\mu_{\rho^e}}(s)(M_\rho^e) = \frac{1}{\#[\overline{s}]} d^{\pi}_{\mu_\rho}(\overline{s})(M_\rho) \qquad \forall s \in [\overline{s}]$$

Dimostrazione. Per quanto riguarda il modello aggregato si ha:

$$\begin{aligned} d^{\pi}_{\mu_\rho}(s)(M_\rho) =& \mu_\rho(s) + \gamma \sum_{s_1 \in S_\rho} \mu_\rho(s_1) P^{\pi}_{\rho}(s_1, s) \\ &+ \gamma^2 \sum_{s_1, s_2 \in S_\rho} \mu_\rho(s_1) P^{\pi}_{\rho}(s_1, s_2) P^{\pi}_{\rho}(s_2, s) \\ &+ \gamma^3 \sum_{s_1, s_2, s_3 \in S_\rho} \mu_\rho(s_1) P^{\pi}_{\rho}(s_1, s_2) P^{\pi}_{\rho}(s_2, s_3) P^{\pi}_{\rho}(s_3, s) + \dots \end{aligned}$$

Per quanto riguarda il modello riespanso, tenendo conto delle trasformazioni di riespansione, si ha:

$$\begin{aligned} d^{\pi^e}_{\mu_{\rho^e}}(s)(M^e_\rho) =& \mu_{\rho^e}(s) + \gamma \sum_{s_1 \in S_{\rho^e}} \mu_{\rho^e}(s_1) P^{\pi^e}_{\rho^e}(s_1, s) \\ &+ \gamma^2 \sum_{s_1, s_2 \in S_{\rho^e}} \mu_{\rho^e}(s_1) P^{\pi^e}_{\rho^e}(s_1, s_2) P^{\pi^e}_{\rho^e}(s_2, s) \\ &+ \gamma^3 \sum_{s_1, s_2, s_3 \in S_{\rho^e}} \mu_{\rho^e}(s_1) P^{\pi^e}_{\rho^e}(s_1, s_2) P^{\pi^e}_{\rho^e}(s_2, s_3) P^{\pi^e}_{\rho^e}(s_3, s) + \dots \\ =& \frac{1}{\#[\overline{s}]} \mu_\rho(\overline{s}) + \gamma \sum_{\overline{s}_1 \in S_\rho} \sum_{s_1 \in [\overline{s}_1]} \frac{1}{\#[\overline{s}_1]} \mu_\rho(\overline{s}_1) \frac{1}{\#[\overline{s}]} P^{\pi}_{\rho}(\overline{s}_1, \overline{s}) \\ &+ \gamma^2 \sum_{\overline{s}_1, \overline{s}_2 \in S_\rho} \sum_{\substack{s_1 \in [\overline{s}_1] \\ s_2 \in [\overline{s}_2]}} \frac{1}{\#[\overline{s}_1]} \mu_\rho(\overline{s}_1) \frac{1}{\#[\overline{s}_2]} P^{\pi}_{\rho}(\overline{s}_1, \overline{s}_2) \frac{1}{\#[\overline{s}]} P^{\pi}_{\rho}(\overline{s}_2, \overline{s}) \\ &+ \gamma^3 \sum_{\overline{s}_1, \overline{s}_2, \overline{s}_3 \in S_\rho} \sum_{\substack{s_1 \in [\overline{s}_1] \\ s_2 \in [\overline{s}_2] \\ s_3 \in [\overline{s}_3]}} \frac{1}{\#[\overline{s}_1]} \mu_\rho(\overline{s}_1) \frac{1}{\#[\overline{s}_2]} P^{\pi}_{\rho}(\overline{s}_1, \overline{s}_2) \\ &\frac{1}{\#[\overline{s}_3]} P^{\pi}_{\rho}(\overline{s}_2, \overline{s}_3) \frac{1}{\#[\overline{s}]} P^{\pi}_{\rho}(\overline{s}_3, \overline{s}) + \dots \\ =& \frac{1}{\#[\overline{s}]} d^{\pi}_{\mu_\rho}(\overline{s})(M_\rho) \end{aligned}$$

□

Risulta utile sottolineare che, per quanto riguarda il numero di visite, la scelta che si effettua per la suddivisione delle probabilità influisce sul risultato finale; in particolare, nel caso generale è possibile solamente garantire che $\sum_{s \in [\overline{s}]} d^{\pi}_{\mu_{\rho^e}}(s)(M^e_\rho) = d^{\pi}_{\mu_\rho}(\overline{s})(M_\rho)$ [2].

Lemma 3.1.3. *Se l'MDP di partenza M è regolare, anche quello ottenuto tramite aggregazione e riespansione M^e_ρ rimane regolare.*

[2] La dimostrazione viene omessa; essa comunque è simile alle dimostrazioni precedenti.

Dimostrazione. Essendo M regolare, si ha che:

$$\exists k, \pi : P^{\pi k}(s, s') > 0 \; \forall s, s'$$

Sviluppando questa condizione, si ha che:

$$(P^{\pi}(s, s'))^k = \sum_{\substack{s_1 \in S \\ s_2 \in S \\ \dots \\ s_{k-1} \in S}} P^{\pi}(s, s_1) P^{\pi}(s_1, s_2) \dots P^{\pi}(s_{k-1}, s') > 0$$

Ogni termine della somma è ≥ 0, in quanto prodotto di termini di probabilità; dato che la somma è > 0, almeno un termine risulta quindi essere > 0 e, di conseguenza,:

$$\begin{aligned} \exists s, s_1, s_2, \dots, s_{k-1}, s' \in S : & P^{\pi}(s, s_1) > 0 \\ & P^{\pi}(s_1, s_2) > 0 \\ & \dots \\ & P^{\pi}(s_{k-1}, s') > 0 \end{aligned}$$

Inoltre, per come è definita la trasformazione e tendendo presente che si sta parlando di probabilità, si ha che:

$$P^{\pi}(s, s') > 0 \Rightarrow P^{\pi}_{\rho^e}(s, s') > 0$$

da cui segue:

$$\begin{aligned} \exists s, s_1, s_2, \dots, s_{k-1}, s' \in S : & P^{\pi}_{\rho^e}(s, s_1) > 0 \\ & P^{\pi}_{\rho^e}(s_1, s_2) > 0 \\ & \dots \\ & P^{\pi}_{\rho^e}(s_{k-1}, s') > 0 \end{aligned}$$

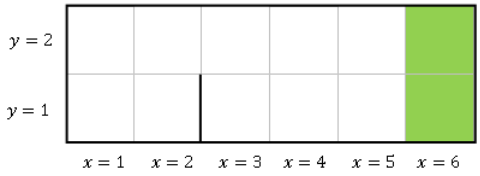
$$(P^{\pi}_{\rho^e}(s, s'))^k = \sum_{\substack{s_1 \in S \\ s_2 \in S \\ \dots \\ s_{k-1} \in S}} P^{\pi}_{\rho^e}(s, s_1) P^{\pi}_{\rho^e}(s_1, s_2) \dots P^{\pi}_{\rho^e}(s_{k-1}, s') > 0$$

da cui segue direttamente la tesi. □

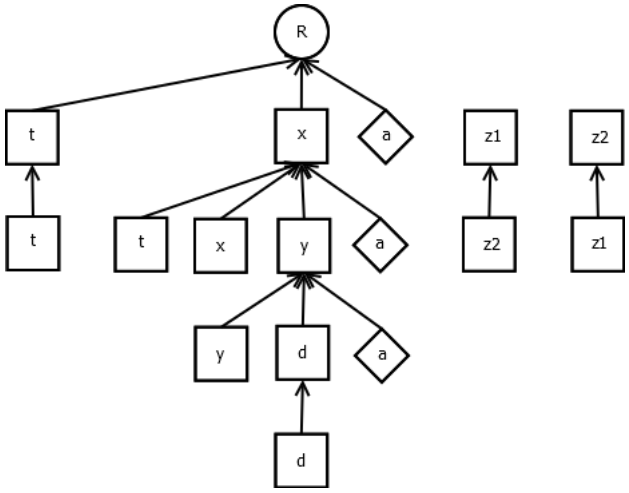
3.2 Problema di esempio

Definiamo ora un esempio semplice ma significativo per quanto riguarda il tipo di variabili da cui è composto, in modo da poter valutare efficacemente gli algoritmi di feature selection proposti in seguito.

L'MDP utilizzato per le prove ha sei variabili di stato (x,y,t,d,z1,z2) e una di azione, le cui dipendenze sono riportate in Figura 3.1(b). Le variabili x e y evolvono,



(a) Griglia



(b) Albero delle dipendenze

Figura 3.1: Problema di esempio.

(a) variabile t

corrente	futuro	probabilità
0	0	0,5
0	1	0,5
1	0	0,1
1	1	0,9

(b) variabile d

corrente	futuro	probabilità
0	0	0,9
0	1	0,1
1	0	0,4
1	1	0,6

Tabella 3.1: Evoluzione delle variabili t e d nel problema di esempio.

a seconda dell'azione, come da griglia di Figura 3.1(a). Nella griglia le linee più chiare rappresentano possibilità di movimento, in orizzontale e verticale, da una cella a quella adiacente, mentre le linee più scure rappresentano dei muri attraverso cui l'agente non può muoversi; gli stati di goal, rappresentati in verde, prevedono un comportamento speciale, descritto più avanti. Le variabili z1 e z2 sono due variabili booleane che a ogni passo si scambiano il valore e non influenzano nient'altro. Le variabili t e d sono autoregressive e modificano il comportamento della griglia. In particolare, con t=0 e d=0 il comportamento è normale; con t=1, con probabilità 0,5 la variabile x non cambia valore, indipendentemente dall'azione; con d=1, con probabilità 0,5 la variabile y ottiene il valore opposto a quello che otterrebbe in condizioni normali. L'evoluzione delle variabili t e d è mostrata in Tabella 3.1.

Il rinforzo che l'agente ottiene è sempre 0 tranne in x=5, con l'azione destra, dato che in questo caso si ha la possibilità di arrivare nello stato di goal; in questo caso si ottiene 1 se t=0, 0,5 se t=1.

Abbiamo definito tre versioni dell'esempio, in modo da generare utilità diverse, a parità di albero delle dipendenze, per le variabili che compongono il problema; in questo modo, infatti, è possibile ottenere delle indicazioni più precise sulla bontà degli algoritmi proposti. Nella versione *assorbente*, in x=6 qualsiasi azione non cambia il valore della x; nella versione *non assorbente*, in x=6 le azioni hanno gli stessi effetti che hanno nel resto dell'MDP; nella versione *regolarizzata*, con probabilità 0,9 le dinamiche sono come nella versione non assorbente, mentre con probabilità 0,1 le variabili z1 e z2 assumono uno qualsiasi dei loro valori ammissibili e le variabili x e y assumono un valore qualsiasi tra quelli che rappresentano o la posizione in cui si trova l'agente o una posizione ad essa adiacente (anche in diagonale), rispettando le dinamiche imposte dai muri.

3.3 Algoritmo di feature selection garantito

Come algoritmo base per selezionare le variabili più utili fornendo delle garanzie proponiamo un algoritmo di tipo *forward selection*: esso prevedere di partire da un insieme vuoto di variabili e a ogni passo di aggiungere una (o, in caso di necessità, anche più di una) variabile, scelta in modo da ottimizzare una certa cifra di merito, fino a quando non si raggiunge una certa condizione terminale, che nelle prove effettuate consiste in un non miglioramento della cifra di merito[3]. Un algoritmo di questo tipo è stato preferito a uno di tipo *backward elimination*, che parte da un insieme composto da tutte le variabili e a ogni passo ne elimina una (o più), principalmente per due motivi:

- **Qualità della riduzione**: la nostra scelta permette di selezionare prima le variabili che portano informazioni riassuntive, privilegiandole rispetto a quelle

[3]Nei casi pratici è, comunque, necessario fornire una condizione di termine più stringente, in modo da non selezionare un numero eccessivo di variabili.

che portano singole informazioni, in quanto da sole portano un maggior numero di informazioni;

- **Tempi computazionali**: la cifra di merito proposta, che analizziamo più avanti, necessita di compiere delle operazioni, anche onerose, sul modello ridotto - ad esempio, risolverlo; se si partisse dal modello completo e lo si volesse ridurre, queste operazioni sarebbero computazionalmente proibitive.

Una peculiarità della cifra di merito proposta è la dipendenza da un certo fattore temporale τ, che indica l'istante temporale fino a cui calcolare alcune quantità; per questo motivo l'algoritmo di tipo forward selection proposto è integrato con un meccanismo di *time shifting*. Nella pratica, forniti in input i valori τ_{min} e τ_{max}, la selezione viene effettuata partendo dall'istante temporale τ_{min} e aumentandolo di una unità ogni qual volta che si raggiunge la condizione terminale, fino a quando non si supera il valore τ_{max}; a questo punto la selezione è considerata conclusa.

Lo pseudocodice dell'algoritmo appena presentato è riportato nell'Algoritmo 1.

Algoritmo 1
(∗ Guaranteed Feature Selection Algorithm $GFSA(\tau_{min}, \tau_{max}, F, S)$ ∗)
Input: τ_{min}, τ_{max}, set of all the features F, score to optimize S
Output: set of selected feature F_S

```
1.  F_S ←∅
2.  τ ←τ_min
3.  while τ ≤ τ_max and F_S ⊂ F
4.          n ←1
5.          while n ≤ cardinality of F \ F_S
6.                  S_C ←∅, Set of Candidate Selections
7.                  for each tuple of n variables T_V ⊆ F \ F_S
8.                          add to S_C the selection F_S ∪ T_V
9.                  if exists a unique candidate selection s ∈ S_C that optimize the
                    score and this is better with respect to selection F_S
10.                   then
11.                         add s to F_S
12.                         n ←1
13.                 else
14.                         n ←n + 1
15.         τ ←τ + 1
16. return F_S
```

Questo algoritmo, come quelli già presenti in letteratura, è euristico; infatti, esso non considera tutte le possibili riduzioni e tra esse sceglie la migliore, ma si limita a confrontare riduzioni semplici, ottenute iterativamente aggiungendo una o più variabili, in modo da risultare computazionalmente accettabile. La novità

che introduciamo, quindi, non è al livello dell'algoritmo base, che rimane euristico. Vogliamo, però, cercare di fornire delle garanzie teoriche sulla bontà della scelta della variabile che viene effettua a ogni iterazione; il contributo, quindi, è al livello della cifra di merito.

3.3.1 Cifra di merito

L'algoritmo proposto necessita, quindi, di una cifra di merito per selezionare un modello ridotto tra un insieme di possibili riduzioni. L'obiettivo è quello di proporre una cifra di merito che permetta di selezionare la variabile più utile per l'individuazione della politica ottima, cioè che minimizzi la differenza di return nel modello originale calcolato utilizzando la politica ottima del modello originale in un caso, l'estensione di quella ottima del modello ridotto nell'altro caso; in formule:

$$\overline{\rho} = \arg\min_{\rho \in \mathfrak{R}} |J^{\pi^*}(M) - J^{\pi^{*e}_{\rho}}(M)| \tag{3.1}$$

Questa cifra di merito, ovviamente, non è utilizzabile nella pratica, in quanto è computazionalmente troppo difficile; deriviamo quindi un'altra cifra di merito, più semplice da calcolare, che limiti superiormente quella ideale 3.1:

$$\begin{aligned}
|J^{\pi^*}(M) - J^{\pi^{*e}_{\rho}}(M)| =& |J^{\pi^*}(M) - J^{\pi^{*e}_{\rho}}(M) + J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^*_{\rho}}(M_\rho)| \\
\leq& |J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho)| + |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}(M)| \\
=& J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho) + |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}(M)| \\
=& J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho) + \\
& |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}_{\tau}(M) + J^{\pi^{*e}_{\rho}}_{\tau}(M) - J^{\pi^{*e}_{\rho}}(M)| \\
\leq& J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho) + \\
& |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}_{\tau}(M)| + |J^{\pi^{*e}_{\rho}}_{\tau}(M) - J^{\pi^{*e}_{\rho}}(M)| \\
\leq& J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho) + |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}_{\tau}(M)| + \\
& \max_{\pi \in \Pi} |J^{\pi}_{\tau}(M) - J^{\pi}(M)|
\end{aligned}$$

Nel bound derivato fino a ora si evince il contributo di tre termini: il primo cattura la distanza di prestazioni che si possono ottenere utilizzando il modello ridotto invece che il modello completo; il secondo è indicativo della differenza causata dall'utilizzo della politica ottima del modello ridotto nel modello completo invece che in quello ridotto, fermandosi inoltre dopo τ passi; il terzo, infine, cattura l'approssimazione dovuta a un troncamento dopo τ passi del calcolo delle prestazioni nel modello completo, rispetto al calcolo completo.

Considerando, inoltre, le indipendenze dalla riduzione otteniamo:

$$\arg\min_{\rho \in \mathfrak{R}} J^{\pi^*}(M) - J^{\pi^*_{\rho}}(M_\rho) + |J^{\pi^*_{\rho}}(M_\rho) - J^{\pi^{*e}_{\rho}}_{\tau}(M)| + \max_{\pi \in \Pi} |J^{\pi}_{\tau}(M) - J^{\pi}(M)| =$$

$$\arg\max_{\rho\in\Re} J^{\pi^*_\rho}(M_\rho) - |J^{\pi^*_\rho}(M_\rho) - J_\tau^{\pi^{*e}_\rho}(M)|$$

Il modello ridotto migliore secondo la cifra di merito derivata, è, quindi:

$$\overline{\rho} = \arg\max_{\rho\in\Re} J^{\pi^*_\rho}(M_\rho) - |J^{\pi^*_\rho}(M_\rho) - J_\tau^{\pi^{*e}_\rho}(M)| \tag{3.2}$$

3.3.2 Considerazioni

Nel caso ideale, cioè l'utilizzo della cifra di merito 3.1, la selezione delle variabili nelle varie versioni del problema di esempio è la seguente. Con la versione assorbente del problema d'esempio, con γ=0,5/0,9/0,9999, viene inizialmente selezionata la variabile d'azione a, poi x e y insieme, poi t e d insieme (con una variazione molto piccola della cifra di merito: l'unica variazione della politica si ha in in x=1,y=1), mentre z1 e z2 non vengono selezionate perché non portano miglioramenti rispetto alla selezione già effettuata; con $\gamma = 0,1$ non viene selezionata nemmeno la coppia t e d, perché porterebbe a un miglioramento della cifra di merito solo all'ottava cifra decimale, mentre nell'esecuzione dell'algoritmo si è utilizzata una tolleranza di 10^{-6} per il confronto dei valori. Con la versione non assorbente e γ=0,5/0,9 viene selezionata prima a, poi x, poi y, poi t e d insieme, abbassando γ a 0,1 t e d non vengono selezionate, mentre alzandola a 0,9999 a e x vengono selezionate insieme. Con la versione regolarizzata viene selezionata prima a, poi x e infine y, mentre anche la coppia t e d non viene selezionata.

La selezione effettuata sulle variabili del problema di esempio utilizzando l'algoritmo proposto con la cifra di merito derivata 3.2 risulta essere molto buona. Infatti, se partiamo con un valore di τ adeguato, la selezione risulta essere come nel caso ideale, a parte il fatto che la coppia t-d viene selezionata anche in average reward col problema regolarizzato perché crea una differenza nello score entro i limiti di tolleranza usati, anche se molto bassa (la differenza è alla quarta/quinta cifra decimale). Anche partendo con $\tau = 0$, inoltre, la selezione non varia; essa, ovviamente, non viene completata alla prima iterazione dell'algoritmo, ma il meccanismo di time shifting permette di arrivare alla soluzione finale corretta, raggiunta in particolare quando τ assume il valore 5.

I dubbi che riguardano questo algoritmo sono legati ai tempi computazionali: risolvere il numero elevato di problemi ridotti che esso richiede potrebbe essere computazionalmente proibitivo.

Osserviamo, inoltre, il tipo di selezione effettuata con questo algoritmo. Essa, infatti, non evidenzia le dipendenze tra le varie variabili del problema e la loro posizione gerarchica. In compenso, però, con questo tipo di selezione vengono mantenute solamente le variabili utili effettivamente per scegliere la politica migliore.

Si osservi, infine, che l'esposizione del metodo presuppone di avere una conoscenza totale del modello; nella realtà, comunque, è sufficiente avere delle traiettorie di

dati del sistema, con le quali è possibile calcolare le quantità di interesse in modo approssimato.

3.4 Algoritmo di feature selection euristico

Il metodo di selezione delle feature euristico che proponiamo è basato sul lavoro preliminare esposto in [7].

L'idea generale di questo metodo prevede di selezionare per prima cosa le variabili che servono per modellizzare la funzione di rinforzo; in seguito, si ricomincia la selezione cercando di spiegare i modelli di transizione a un passo delle variabili di stato selezionate in precedenza, ripetendo il procedimento fino a quando non sono state spiegate tutte le variabili di stato selezionate. Col problema di esempio esposto nella Sezione 3.2, quindi, verrebbero selezionate inizialmente le variabili t, x, a, dato che esse servono a prevedere il valore del rinforzo. A questo punto si passa all'analisi della variabile t: per prevederne il valore all'istante temporale t+1 è sufficiente conoscere il suo valore al tempo t, quindi non è necessario aggiungere nessuna variabile alla selezione. La successiva variabile da analizzare è x: per spiegarne il modello di transizione sono necessarie le variabili t,x,y,a, quindi aggiungiamo alla selezione y, l'unica non ancora presente. A questo punto risulta necessario analizzare y, che è predicibile attraverso le variabili y,d,a; concludiamo, quindi, che anche la variabile d va aggiunta alla selezione. Ora dobbiamo analizzare d, che risulta spiegabile solo con d stessa e quindi non aggiungiamo nessuna variabile alla selezione. Avendo terminato di analizzare tutte le variabili selezionate possiamo concludere che la versione ridotta dell'esempio contiene le variabili x,y,t,d,a, mentre non contiene z1 e z2.

Questo metodo, quindi, induce la selezione di tutte le variabili necessarie a spiegare il rinforzo direttamente o indirettamente; in pratica, parlando in termini di MDP, esso seleziona le variabili necessarie per poter calcolare esattamente la funzione di rinforzo e la parte di funzione di transizione che riguarda in qualche modo le variabili coinvolte nella funzione di rinforzo. Nell'esempio, infatti, le variabili z1 e z2 non sono state selezionate, in quanto non servono a prevedere, nè direttamente nè indirettamente, il valore del rinforzo. La previsione indiretta del rinforzo, invece, è necessaria: ad esempio, il rinforzo non è collegato direttamente con la variabile y, ma il valore di y è necessario per scegliere l'azione migliore quando x vale 2.

Come è facilmente intuibile, il procedimento appena spiegato permette una selezione senza perdite, cioè induce un modello ridotto ρ tale per cui $J^{\pi^*}(M) = J^{\pi^*_\rho}(M)$ e anche, ancora più restrittivo, $V^{\pi^*}(M) = V^{\pi^{*_e}_\rho}(M)$. Esso, però, soffre di due importanti difetti. Innanzitutto, per poter applicare questo metodo è necessario conoscere il modello completo delle dipendenze tra le varie variabili, modello che difficilmente in una applicazione reale è disponibile. Inoltre, anche supponendo di conoscere esattamente il modello, la riduzione ottenibile con questo metodo difficilmente può essere significativa, in quanto contiene tutte le variabili coinvolte in qualche modo nella pre-

visione diretta o indiretta del rinforzo, senza considerare quanto sono importanti per questa previsione.

L'algoritmo che proponiamo, quindi, si basa sull'idea esposta, aggiungendo un concetto di utilità di una variabile, in base al quale decide quali variabili possono essere utili; questa utilità, inoltre, è calcolata da un dataset contenente una serie di traiettorie di dati, in modo da eliminare la necessità del modello del sistema.

Come conseguenza dell'idea esposta risulta naturale suddividere le feature in due insiemi, quelle di input F^i e quelle di output F^o. L'idea, infatti, è quella di riuscire a prevedere il valore che avranno le feature di output più significative in base al valore che assumono quelle di input significative. In particolare, le feature di input sono quelle necessarie a descrivere lo stato corrente e l'azione selezionata ($F^i = \{S_j\}_{j=1}^n \cup \{A_k\}_{k=1}^m$), mentre quelle di output sono le feature necessarie a descrivere lo stato futuro e il rinforzo ottenuto ($F^o = \{S_j\}_{j=1}^n \cup \{r\}$).

3.4.1 Algoritmo base

L'algoritmo base che proponiamo è un algoritmo ricorsivo che, dati in input un dataset di traiettorie D, la feature da predire $f^o \in F^o$ e l'insieme di feature già selezionate $F_S^i \subset F^i$, restituisce l'insieme $F_{f^o}^i \subset F^i$ di feature necessarie per spiegare f^o.

Il dataset è definito come $D = \{< \mathbf{f}_k^i, \mathbf{f}_k^o >\}_{k=1}^N$, cioè contiene una serie di transizioni di un passo osservate nel sistema. Il suo utilizzo permette di effettuare la selezione senza necessitare del modello del sistema, ma avendo a disposizione solamente alcune sue traiettorie.

L'algoritmo viene lanciato con $f^o = r$ e $F_S^i = \emptyset$, in modo da poter selezionare le variabili necessarie a predire il rinforzo; per questo scopo è necessario definire un algoritmo, che chiameremo algoritmo di predizione (FPA, descritto nella Sezione 3.4.2), in grado di individuare le variabili necessarie a spiegarne una particolare utilizzando le traiettorie di dati contenute in un dataset. Dato il sottoinsieme di variabili selezionate per predire il rinforzo, si ripete l'algoritmo ricorsivamente utilizzando come variabile di output da spiegare ogni variabile di stato contenuta in questo sottoinsieme. L'algoritmo continua con la selezione di variabili fino a quando non viene selezionata alcuna ulteriore variabile di stato che necessita di essere spiegata; quando termina, l'algoritmo restituisce l'insieme di tutte le feature selezionate.

Lo pseudocodice dall'algoritmo appena esposto è riportato nell'Algoritmo 2.

Algoritmo 2

($*$ Heuristic Feature Selection Algorithm $HFSA(D, f^o, F_S^i, FPA)$ $*$)

Input: dataset $D = \{< \mathbf{f}_k^i, \mathbf{f}_k^o >\}_{k=1}^N$, variable to be explained $f^o \in F^o$, set of already selected features $F_S^i \subset F^i$, feature prediction algorithm FPA

Output: set of selected feature $F_{f^o}^i \subset F^i$ (which are needed to estimate f^o)

1. $F_{f^o}^i \leftarrow FPA(D, f^o)$

2. $F^{new}_{state} \leftarrow (F^i_{f^o} \setminus F^i_S) \cap \{S_j\}^n_{j=1}$
3. **for each** $f^o_{state} \in F^{new}_{state}$
4. $\quad F^i_{f^o} \leftarrow F^i_{f^o} \cup HFSA(D, f^o_{state}, (F^i_S \cup F^i_{f^o}))$
5. **return** $F^i_{f^o}$

3.4.2 Algoritmo di predizione della feature

L'algoritmo euristico di selezione delle feature necessita, quindi, di un algoritmo in grado di identificare quali variabili di input sono le più significative per spiegare una particolare variabili di output. Un algoritmo di questo tipo dovrebbe proporre un insieme di feature che sia allo stesso tempo significativo e non ridondante, cioè dovrebbe selezionare tutte le feature più utili per predire la variabile di output, evitando però di inserirne alcune che portano all'incirca le stesse informazioni. L'algoritmo, inoltre, deve anche essere in grado di gestire in tempi ragionevoli dataset con grandi dimensionalità, in modo da poter essere utilizzato con dataset che contengano un numero di dati adeguato per essere rappresentativo del sistema oggetto della riduzione.

L'algoritmo che proponiamo per prima cosa classifica tutte le feature di input, utilizzando una opportuna misura statistica della loro significatività per la predizione della variabile di output, calcolata tramite la procedura VRP descritta nella Sezione 3.4.3. Esso, in base a questa classificazione, seleziona solamente la feature più significativa, f^*, utilizzandola per costruire, attraverso la procedura MBP descritta nella Sezione 3.4.3, un modello $\widehat{M}_\rho$ [4] che spieghi f^o, in modo da evitare di selezionare variabili ridondanti; una volta selezionata una feature, infatti, tutte quelle correlate con essa possono diventare ridondanti e, quindi, assumere un valore basso nella classificazione. L'algoritmo, quindi, ripete l'operazione utilizzando come nuovo output la componente non spiegata dal modello costruito fino a ora, cioè $\widehat{f}^o_k = f^o_k - \widehat{M}_\rho(\widehat{\mathbf{f}}^i_k), k = 1, \ldots, N$, fino a quando la feature classificata prima è già selezionata oppure l'indice di accuratezza del modello $\widehat{M}_\rho$ non migliora in modo significativo. Per misurare questa accuratezza utilizziamo il coefficiente di determinazione R^2 tra la variabile di output f^o e il suo valore predetto da $\widehat{M}_\rho$, $\widehat{f}^o$:

$$R^2(f^o, \widehat{f}^o) = 1 - \frac{\sum_{k=1}^N (\widehat{f}^o_k)^2}{\sum_{k=1}^N (f^o_k - \overline{f}^o)^2}$$

dove $\overline{f}^o = \frac{1}{N} \sum_{k=1}^N f^o_k$ è la media delle feature di output.

Lo pseudocodice dall'algoritmo appena esposto è riportato nell'Algoritmo 3.

Algoritmo 3

($*$ Feature Prediction Algorithm $FPA(D, f^o, MBP, VRP)$ $*$)

[4] In questo caso il modello di cui stiamo parlando è un modello predittivo per la variabile di output considerata e non un MDP ridotto; il simbolo $\widehat{\cdot}$ serve per sottolineare questo fatto e non creare confusione.

Input: dataset $D = \{< \mathbf{f}_k^i, \mathbf{f}_k^o >\}_{k=1}^N$, variable to be explained $f^o \in F^o$, model building procedure MBP, variable ranking procedure VRP
Output: set of features selected to estimate f^o $F_S \subset F^i$

1. $F_S \leftarrow \emptyset$
2. $\widehat{f^o} \leftarrow f^o$
3. $R^2_{old} \leftarrow 0$
4. **repeat**
5. $\quad f^* \leftarrow \arg\max_{f \in F^i} VRP(D, \widehat{f^o}, f)$
6. $\quad$ **if** $f^* \in F_S$
7. $\quad\quad$ **then return** F_S
8. $\quad F_S \leftarrow F_S \cup \{f^*\}$
9. $\quad \widehat{M_\rho} \leftarrow MBP(D, f^o, F_S)$
10. $\quad \widehat{f^o} \leftarrow f^o - \widehat{M_\rho}(F_S)$
11. $\quad \Delta R^2 \leftarrow R^2(D, f^o, \widehat{f^o}) - R^2_{old}$
12. $\quad R^2_{old} \leftarrow R^2(D, f^o, \widehat{f^o})$
13. **until** $\Delta R^2 < \epsilon$
14. **return** F_S

3.4.3 Costruzione del modello e classificazione delle variabili

Per completare la descrizione dell'algoritmo proposto dobbiamo specificare come costruiamo il modello di predizione della variabile di output e come classifichiamo le variabili di input.

Il modello che abbiamo deciso di utilizzare è basato su una foresta di alberi *Extremely Randomized Trees*(Extra-Trees) [24]. I metodi basati su alberi sono metodi di apprendimento supervisionato incentrati sull'idea di alberi decisionali, strutture che rappresentano un insieme di regole a cascata che portano a valori numerici [6]. Gli alberi sono ottenuti dividendo ripetutamente, secondo un certo criterio, i dati in ingresso, in modo da tenere insieme quelli più simili e dividere quelli più diversi, fino a quando le differenze in un nodo sono minime oppure il nodo contiene un numero minimo di elementi; così facendo si crea una struttura ad alternative che portano alle foglie, le quali sono associate con un valore numerico finale.

Gli Extra-Trees sono un tipo particolare di alberi decisionali che randomizzano sia le variabili in ingresso, sia il punto di taglio per dividere un nodo. In particolare, per dividere un nodo si selezionano casualmente K differenti alternative di variabili di input e, per ciascuna di esse, si sceglie un punto di taglio casuale; si assegna, poi, un punteggio a ciascuna direzione di taglio, utilizzando quindi quella con punteggio maggiore per dividere il nodo. Quando un nodo contiene un numero inferiore di n_{min} elementi non viene più diviso, risultando quindi una foglia a cui viene assegnato, come valore numerico, la media degli output corrispondenti alle feature di input che esso contiene. Per compensare la casualità introdotta si crea una foresta di M alberi e la stima finale è costituita dalla media delle diverse stime proposte dagli alberi.

I valori K, n_{min} e M possono essere assegnati attraverso delle prove sperimentali; una buona euristica consiste nel porre M il più alto possibile compatibilmente con i requisiti sui tempi computazionali (negli eseprimenti che verranno riportati in seguito si sono utilizzati 100 alberi), K uguale al numero di feature in input, mentre n_{min} scelto provando alcuni valori e selezionando quello che consente di ottenere il valore di R^2 maggiore.

Abbiamo deciso di utilizzare gli Extra-Trees per la costruzione del modello perché essi, oltre a essere ragionevolmente efficienti dal punto di vista computazionale e accurati nella predizione, possono essere sfruttati direttamente per classificare le variabili in input [69][22]. L'idea consiste nell'assegnare a ogni variabile di input un punteggio pari alla percentuale di riduzione della varianza sugli M alberi della foresta ottenuta grazie alla variabile considerata. Più precisamente, in formule il punteggio assegnato a ogni feature di input f^i per spiegare la feature in output f^o basandosi sui dati contenuti nel dataset D è:

$$VRP(D, f^o, f^i) = \frac{\sum_{m=1}^{M} \sum_{j=1}^{\Omega_m} \delta(\eta_{j,m}, f^i) \cdot \Delta_{var}(\eta_{j,m})}{\sum_{m=1}^{M} \sum_{\eta_{j,m}}^{\Omega_m} \Delta_{var}(\eta_{j,m})}$$

dove $\eta_{j,m}$ è il j-esimo nodo non terminale dell'm-esimo albero, Ω_m è il numero di nodi non terminali dell'm-esimo albero, $\delta(\eta_{j,m}, f^i)$ è uguale a 1 se f^i è usata per dividere il nodo $\eta_{j,m}$, 0 altrimenti, $\Delta_{var}(\eta_{j,m})$ è la riduzione di varianza che si ha quando si divide il nodo $\eta_{j,m}$, cioè:

$$\Delta_{var}(\eta_{j,m}) = |D| var\{f^o|D\} - |D_{i,l}| var\{f^o|D_{i,l}\} - |D_{i,r}| var\{f^o|D_{i,r}\}$$

dove i termini $D_{i,l}$ e $D_{i,r}$ sono i due sottoinsiemi ottenuti dividendo D lungo la dimensione di input f^i.

3.4.4 Considerazioni

L'algoritmo proposto è un algoritmo euristico per selezionare le feature più utili di un dato problema a partire da un insieme di traiettorie di dati. Esso, quindi, data la sua natura euristica non fornisce alcuna garanzia sulla bontà della politica che è possibile apprendere utilizzando il modello ridotto con questo metodo. Inoltre, essendo finalizzato alla predizione delle variabili di output, può selezionare delle variabili inutili per la determinazione della politica ottima, nel caso in cui queste variabili abbiano un'influenza sulla Q function ma non sul suo massimo. L'algoritmo, però, risulta essere interessante in quanto, oltre a permettere di ottenere la riduzione in tempi ragionevoli, effettuando la selezione costruisce implicitamente l'albero di dipendenze tra le variabili utili che compongono il problema. Quando viene spiegata una variabile di output f^o, infatti, si estrae un insieme di variabili $F^i_{f^o}$ che non contiene altro che le variabili da cui f^o dipende; di conseguenza, basta seguire le sequenze di selezioni per costruire l'albero.

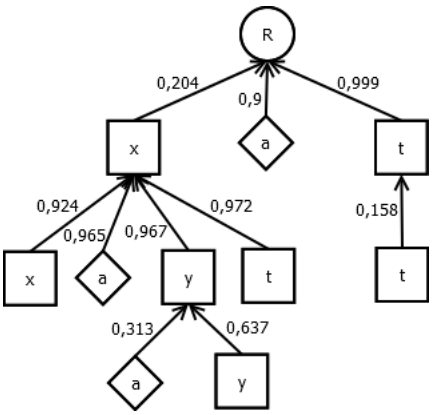


Figura 3.2: Albero delle dipendenze delle feature generato dall'algoritmo di riduzione delle feature euristico per il problema d'esempio. L'ordine da sinistra a destra dei nodi indica l'ordine di selezione delle variabili, mentre i numeri riportati sugli archi indicano il valore di R^2 ottenuto con la riduzione che comprende tutte le variabili selezionate fino a quel punto.

Per verificare la bontà di questo algoritmo abbiamo provato ad applicarlo al problema di esempio descritto nella Sezione 3.2, generando un dataset di 20 mila tuple con la versione non assorbente dell'esempio; l'albero risultante, ottenuto assegnando i parametri M=100, K=7, n_{min}=10, è riportato in Figura 3.2. Come si può notare, l'albero che si ottiene è molto simile a quello reale (lo si confronti con quello in Figura 3.1(b)), con l'unica differenza costituita dall'assenza della variabile d per spiegare la y[5]; la sua assenza è, in parte, spiegabile considerando che, anche conoscendo il valore di d, la predizione che si potrebbe fare per y non migliorerebbe in maniera eccessiva. Inoltre, è possibile notare come anche il valore di R^2 sia coerente con il sistema; infatti, nei casi in cui la variabile è determinabile abbastanza precisamente con la conoscenza delle variabili da cui dipende esso è prossimo a 1, mentre nei casi più stocastici il suo valore è significativamente più basso.

I tempi computazionali dell'algoritmo, inoltre, potrebbero essere resi ancora più rapidi e la riduzione più stretta nel caso non interessi costruire l'albero di dipendenze della variabili. È, infatti, possibile modificare l'algoritmo di predizione delle feature in modo che consideri le variabili già selezionate in precedenza in altri punti dell'algoritmo base; in questo modo, quindi, si avrebbero già delle informazioni nel modello di base, che potrebbero permettere di predire (almeno in parte) la variabile di interesse. In questo modo, ovviamente, l'albero delle dipendenze non viene più costruito, in quanto il modello di base per la predizione della variabile di interesse contiene già alcune feature, che potrebbero essere o non essere utili per la sua spiegazione. La riduzione, però, potrebbe essere maggiore, in quanto le feature già presenti nel modello di partenza possono potenzialmente sostituire come utilità per

[5]L'assenza delle variabili z1 e z2 è, invece, corretta, in quanto queste due variabili non influenzano né direttamente né indirettamente il rinforzo.

la predizione alcune delle variabili che invece verrebbero selezionate se si partisse da un modello vuoto.

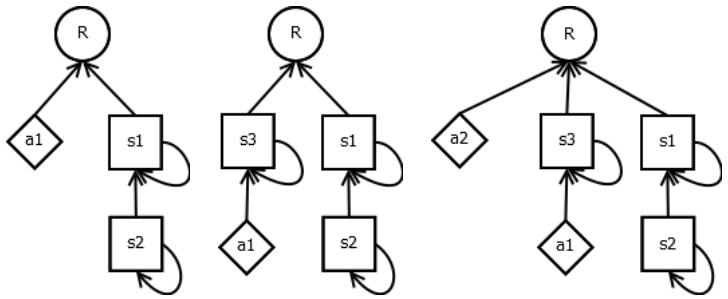
3.5 Utilità di una variabile in base alla sua controllabilità

Analizziamo ora l'utilità, al fine della scelta della politica ottima, delle variabili che compongono il problema, in base alla cotrollabilità loro e delle altre variabili e alle dipendenze tra queste variabili. L'analisi svolta è basata su esempi di alberi delle dipendenze, scelti in modo da avere una buona generalità e esaustività dei casi che possono capitare; questa analisi, inoltre, non è totalmente formale, ma è fondata su considerazioni generali e abbastanza intuitive. Inoltre, l'analisi è effettuata per il caso rinforzo medio, dato che in questo scenario emergono più chiaramente i legami tra le variabili, la cui utilità non viene modificata dal fatto di avere una vista ridotta sul futuro.

Prima di partire con l'analisi vera e propria è utile sottolineare che, a parità di albero delle dipendenze, una variabile può essere utile o no per la determinazione della politica ottima a seconda del problema specifico. Si consideri l'esempio presentato nella Sezione 3.2; in esso le variabili t e d determinano la politica ottima negli stati caratterizzati da x=1 e y=1. Se definissimo una variante di questo esempio in cui tutto rimane come è, tranne per il fatto che dalla griglia vengono eliminate le caselle con x=1, l'albero delle dipendenze non cambia, ma le variabili t e d diventano totalmente inutili per determinare la politica ottima (anche se comunque utili per calcolare la value function ottima), in quanto l'unico stato in cui in base al loro valore cambiava la politica ottima viene eliminato dal problema di esempio. Di conseguenza, l'analisi che facciamo vuole distinguere le variabili sicuramente inutili da quelle potenzialmente, ma non certamente, utili.

Osserviamo, innanzitutto, che se una variabile che influenza in uno o più passi il rinforzo è controllabile, direttamente o attraverso altre variabili controllabili da cui dipende, non può essere tolta a prescindere; infatti, è possibile che influenzando il valore di questa variabile con la scelta delle azioni si cambi il rinforzo in modo che anche la politica ottima cambi. I casi interessanti da esaminare, quindi, sono quelli che contengono delle variabili non controllabili che influenzano, direttamente o indirettamente, il rinforzo, in modo da capire se esistono dei casi in cui, date le loro dipendenze con le altre variabili, queste variabili possono essere tolte o meno dal problema solo in base all'albero delle dipendenze tra variabili.

Esistono essenzialmente tre casi significativi da analizzare. Scrivendo la performance in un MDP M di una politica π come $J^\pi = d^{\pi T} R^\pi$, dove con d^π abbiamo indicato la distribuzione stazionaria di M seguendo la politica π, si può avere che lo stato non è controllabile, cioè la distribuzione stazionaria non dipende dalla politica, oppure il rinforzo non è direttamente controllabile, oppure entrambi sono controlla-



(a) $J^\pi = d^T R^\pi$ (b) $J^\pi = d^{\pi T} R$ (c) $J^\pi = d^{\pi T} R^\pi$

Figura 3.3: Alberi di dipendenza delle variabili, semplici ma generali, nei tre casi significativi, cioè con distribuzione stazionaria non controllabile, rinforzo non controllabile e entrambi controllabili. Nello schema con R si è indicato il rinforzo, an una variabile di azione, sn una variabile di stato.

bili[6]. Inoltre, oltre a questi tre casi strutturali, è utile chiedersi come cambia l'ultilità in due scenari differenti, in cui si usa una variazione degli MDP tradizionali, cioè il caso multi-obiettivo e il caso con disturbi esogeni.

Stato non controllabile

Nel caso con stato non controllabile la performance di una politica π è esprimibile nel seguente modo:

$$J^\pi = d^T R^\pi$$

Un esempio significativo di albero delle dipendenze tra variabili per questa situazione è riportato in Figura 3.3(a); si possono notare due tipologie di variabili di stato: quelle che influenzano direttamente il rinforzo (s1 nello schema) e quelle che influenzano direttamente altre variabili di stato, ma solo indirettamente il rinforzo (s2 nello schema).

La variabile s1 può influenzare la politica ottima. Infatti, a ogni passo, l'obiettivo è scegliere l'azione (attraverso a1) che consente di ottimizzare il rinforzo immediato; per far ciò, è necessario conoscere il valore che attualmente assume s1, in quanto essa influenza direttamente R e, quindi, solo in base a questo valore si può scegliere l'azione che massimizza il rinforzo.

La variabile s2, invece, non ha alcuna utilità nella determinazione della politica ottima. Infatti, dato che l'unica influenza che si può imporre dall'esterno è sul rinforzo, è sufficiente conoscere il valore che ha attualmente la variabile s1 per poter determinare l'azione migliore, mentre non è necessario dover prevedere il valore che

[6]Il caso in cui sia lo stato che il rinforzo non sono controllabili non è significativo, in quanto non si parlerebbe di MDP, ma di Markov Reward Process, in cui non si ha il concetto di politica, ma l'azione "sceglibile" in ogni stato è una sola; non ha quindi alcun senso chiedersi quali variabili influenzano la politica ottima in questo caso.

assumerà s1 in futuro, poiché con la scelta delle azioni non si può influenzare in nessun modo lo stato futuro del sistema; di conseguenza, la variabile s2 non è utile per determinare la politica ottima, dato che essa potrebbe essere usata solo per prevedere il valore futuro della variabili s1.

Generalizzando, in questo caso, se l'obiettivo è unicamente rappresentare la politica ottima, si possono eliminare tutte le variabili che non influenzano direttamente il rinforzo. Questa forte semplificazione è giustificabile se si considera che, nel caso lo stato non sia controllabile, non si è in presenza di un problema di apprendimento per rinforzo vero e proprio, poiché non c'è nessun tipo di programmazione del futuro.

Rinforzo non controllabile

Nel caso con rinforzo non controllabile la performance di una politica π è esprimibile nel seguente modo:

$$J^{\pi} = d^{\pi T} R$$

Un esempio significativo di albero delle dipendenze tra variabili per questa situazione è riportato in Figura 3.3(b). Affinché la situazione rappresentata sia generale, lo stato è solo parzialmente controllabile. In particolare, s3, che influenza direttamente il rinforzo, è controllabile attraverso a1, mentre non sono presenti variabili controllabili indirettamente, in quanto comunque la situazione non cambierebbe, poiché in entrambi i casi l'effetto che ha la controllabilità della variabile si ha a più passi; s1 e s2, invece, non sono controllabili, con s1 che influenza direttamente il rinforzo, mentre s2 lo influenza solo indirettamente attraverso s1.

I ragionamenti fatti nel caso precedente non sono più validi, in quanto adesso gli effetti delle azioni scelte non sono immediati ma, al contrario, determinano l'evoluzione futura dello stato. Difatti, in questa situazione nessuna variabile può essere eliminata a priori.

Le variabili s1 e s3 sono, ovviamente, essenziali per poter determinare la politica ottima. s3, infatti, determina R e può essere controllata direttamente, ma è anche autoregressiva[7], quindi è necessario conoscere il suo valore per sapere qual è l'azione più opportuna da compiere per fare in modo che essa influenzi nel modo voluto il rinforzo; s1, invece, non è controllabile, ma influenza direttamente il rinforzo, quindi anch'essa serve per scegliere l'azione migliore. In pratica, per poter influenzare R nel modo voluto è necessario conoscere sia s1 che s3.

Il contributo di s2, invece, è meno evidente. Nei fatti, comunque, essa può essere utile per determinare la politica ottima. Un esempio in cui questo accade è il seguente: l'agente che apprende si muove in una griglia 3x1, in cui s3 è la posizione dell'agente e può valere -1,0,1; s1 è una variabile booleana che determina se il rinforzo

[7] Il caso di variabili di stato non autoregressive non è interessante, perché esse non fanno altro che introdurre un ritardo nel sistema.

è diverso da 0 in s3=-1 o s3=1, la cui evoluzione dipende da s2; s2 è una variabile booleana, tale che se s2(t)=-1 s1(t+1)=s1(t), altrimenti s1(t+1)=-s1(t), e mantiene il suo valore con probabilità 0,9, mentre lo cambia con probabilità 0,1; a1 può valere sinistra o destra e muove l'agente, se non si è all'estremo, nella casella adiacente indicata dall'azione, con una probabilità di fallimento dell'azione pari a 0,9 se s3=0, 0,1 altrimenti. Risolvendo questo problema giocattolo si vede che anche s2 serve a determinare la politica ottima.

Generalizzando le affermazioni precedenti possiamo concludere che nel caso con stato controllabile e rinforzo non controllabile non è possibile, in base alla controllabilità, eliminare nessuna variabile, tra quelle che influenzano direttamente o indirettamente il rinforzo, solo osservando il grafo delle loro dipendenze.

Stato e rinforzo controllabili

Nel caso con sia stato che rinforzo controllabili la performance di una politica π è esprimibile nel seguente modo:

$$J^{\pi} = d^{\pi T} R^{\pi}$$

Un esempio significativo di albero delle dipendenze tra variabili per questa situazione è riportato in Figura 3.3(c); questo esempio è una generalizzazione di quelli in Figura 3.3(a) e 3.3(b).

Dato che questo caso è una generalizzazione dei due casi precedenti, uno dei quali non ammette nessuna eliminazione solo in base al grafo delle dipendenze, anche in esso non è possibile togliere alcune variabili solo osservando il grafo e la controllabilità delle variabili che esso contiene.

Multi-obiettivo

In un MDP multi-obiettivo lo scopo è ottimizzare contemporaneamente più di un rinforzo, che solitamente sono contrastanti. In questo caso, quindi, invece di avere un unico albero delle dipendenze, se ne ha uno per ogni rinforzo. Di conseguenza, per sapere quali variabili possono essere necessarie è sufficiente applicare quanto esposto nel caso singolo-obiettivo a ogni albero e, infine, unire gli insiemi delle variabili ricavati.

Disturbi

Il sistema da controllare potrebbe essere soggetto anche a dei disturbi, cioè ingressi non dinamici al sistema; essi, quindi, vengono modellati come delle variabili stocastiche non correlate nel tempo e che non dipendono da nessun altro fattore interno al sistema.

Un esempio generico di albero delle dipendenze tra variabili per un sistema soggetto a disturbi è rappresentato in Figura 3.4, in cui abbiamo utilizzato come base

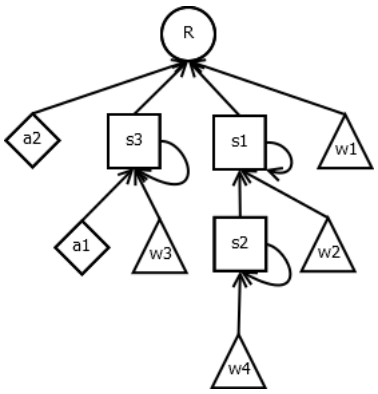


Figura 3.4: Albero di dipendenza delle variabili, semplice ma generale, nel caso siano presenti dei disturbi. Nello schema con R si è indicato il rinforzo, an una variabile di azione, sn una variabile di stato, wn una variabile di disturbo.

l'esempio del caso in cui sia stato che rinforzo sono controllabili e abbiamo applicato un disturbo differente a ogni variabile di stato e al rinforzo.

Risulta abbastanza evidente che tutte le variabili di disturbo dell'esempio possono influenzare la politica ottima. w1 e w3, infatti, influenzano direttamente il rinforzo o la variabile di stato controllabile e, quindi, servono direttamente per scegliere l'azione migliore; w2 e w4, invece, influenzano altre variabili di stato che a loro volta hanno un'influenza diretta o indiretta sul rinforzo, quindi la loro utilità è paragonabile a quella di s2 e, di conseguenza, come dimostrato in precedenza con un esempio, possono servire a determinare la politica ottima.

Generalizzando le osservazioni fatte, possiamo dire che l'utilità delle variabili di disturbo è esattamente la stessa di quella delle variabili di stato che si trovano in una posizione simile nell'albero delle dipendenze, nonostante il significato e l'evoluzione di questi due tipi di variabili sia completamente diverso.

Capitolo 4

Configurazione degli esperimenti

Il controllore di un gruppo di ascensori è un sistema installato in un edificio in cui sono presenti più ascensori che, parallelamente, servono i piani che compongono l'edificio; l'obiettivo di questo controllore è il coordinamento dei diversi ascensori, in modo da poter servire tutte le richieste presenti nel sistema in modo efficiente. Il generico concetto di efficienza può corrispondere a diverse misure pratiche, anche in conflitto tra loro, come esposto nella Sezione 2.2; il nostro scopo è, principalmente, quello di fare aspettare meno tempo possibile le persone in attesa dell'arrivo di un ascensore e, quindi, il nostro obiettivo è riassumibile nella minimizzazione del tempo di attesa medio e massimo.

I controllori di gruppi di ascensori possono essere classificati a seconda della tecnologia utilizzata per la loro progettazione. I controllori più diffusi sono di tipo euristico, ma stanno emergendo anche dei controllori basati su tecniche di intelligenza artificiale, come abbiamo analizzato nella Sezione 2.4; seguendo questa seconda strada, ci proponiamo di progettare un controllore che apprenda la politica migliore tramite apprendimento per rinforzo.

Un'altra classificazione dei controllori di gruppi di ascensori è basata sulla quantità e tipologia di informazioni che i controllori possono utilizzare per ottenere il loro scopo, una cui panoramica è contenuta nella Sezione 2.1. La nostra scelta è ricaduta sull'utilizzo di una configurazione classica, che utilizza solamente le informazioni ottenibili facilmente; in questo modo si evita di complicare in modo eccessivo il sistema, anche se si introduce una limitazione nella qualità del controllo ottenibile. In particolare, questa configurazione prevede che a ogni piano siano presenti solamente due bottoni di chiamata, che gli utenti utilizzano per indicare in che direzione vogliono viaggiare, mentre selezionano la destinazione effettiva da raggiungere solamente una volta saliti a bordo dell'ascensore; un indicatore luminoso è presente a ogni piano, per ogni ascensore, con lo scopo di indicare alle persone in attesa la direzione in cui l'ascensore continuerà il suo viaggio dopo essersi fermato al piano. Inoltre, abbiamo lasciato la possibilità al controllore di cambiare la propria decisione a ogni istante decisionale, a patto di rispettare alcune regole che garantiscono un servizio

ammissibile e accettabile dagli utenti del sistema, come le regole di Closs, descritte nella Sezione 2.4, e l'impossibilità per un ascensore di invertire direzione di marcia senza prima fermarsi a un piano intermedio. Sempre con lo scopo di non complicare il sistema abbiamo, infine, deciso di utilizzare ascensori a una sola cabina e di non differenziare i servizi offerti dai diversi ascensori.

Il controllore del gruppo di ascensori deve fornire al sistema il comportamento da avere in seguito ad alcuni eventi, che quindi generano dei momenti decisionali. Abbiamo deciso che il controllore fornisce al sistema le proprie decisioni indicando, per ogni ascensore, la sua destinazione e la direzione con cui viaggerà una volta arrivato a quella destinazione. Gli eventi che generano momenti decisionali sono due: quando una persona preme un bottone a un piano per richiedere un ascensore (richiesta di tipo *requestElevator*) e quando un ascensore fermo a un piano, dopo aver chiuso le proprie porte, è pronto per ripartire (richiesta di tipo *setNextDirection*); solo in seguito a questi eventi il sistema può prendere una decisione o, anche, modificare una decisione presa in precedenza.

L'elemento fondamentale per l'apprendimento e la valutazione di controllori di gruppi di ascensori è il simulatore del gruppo di ascensori; utilizzare un sistema reale a questo scopo, infatti, non è possibile sia per il disagio che creerebbe agli utenti del sistema, sia per i tempi lunghi che richiederebbe. Risulta, quindi, fondamentale avere a disposizione un simulatore che sia allo stesso tempo realistico, per garantire di ottenere dei dati significativi, e rapido nell'eseguire le simulazioni, in modo da poter ottenere dei risultati in tempi ragionevoli.

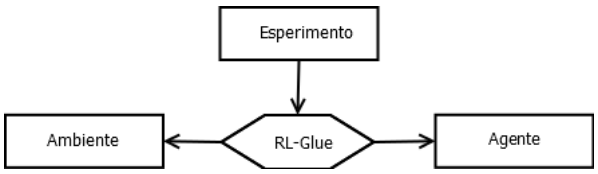
Il simulatore che abbiamo deciso di utilizzare è *elevatorsim*[1], un simulatore open source sviluppato in Java da Neil McKellar e Chris Dailey e rilasciato con licenza LGPLv2, che possiede tutte le caratteristiche necessarie per i nostri scopi; questa scelta ci ha permesso di concentrarci sull'implementazione dei controllori, senza perdere tempo nello sviluppare da zero un nuovo simulatore. I motivi principali che ci hanno spinto a scegliere questo simulatore e non altri sono i seguenti:

- Realismo del simulatore, che permette di settare tutti i parametri principali che caratterizzano il sistema fisico[2];

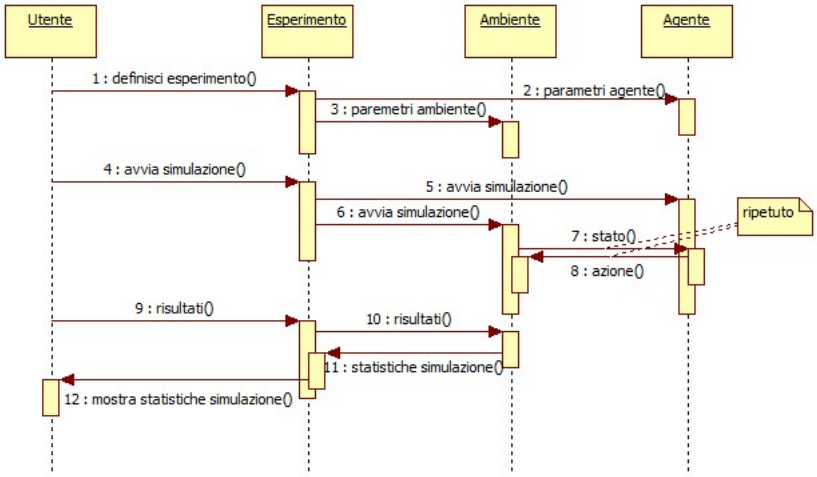
- Flessibilità dei tempi computazionali: nella modalità grafica, utile per avere un'idea qualitativa delle caratteristiche del controllore, la simulazione è controllata da un orologio in tempo reale, la cui velocità può essere modificata tramite un moltiplicatore esponenziale; in modalità non grafica, utile per ottenere dei risultati quantitativi, la simulazione è controllata dagli eventi e quindi la sua velocità è limitata solamente dalle capacità dell'elaboratore utilizzato;

[1] http://elevatorsim.sourceforge.net/

[2] Una lista completa di questi parametri è presente nella Sezione 4.3.1.



(a) Architettura



(b) Flusso di esecuzione

Figura 4.1: Struttura del progetto interfacciato tramite RL-Glue.

- Struttura modulare e ingegnerizzata del codice, che permette di apportare facilmente le modifiche necessarie e di definire senza problemi dei nuovi controllori per il sistema;

- Licenza open source, che non limita l'utilizzo fattibile del simulatore.

Il simulatore, inoltre, è stato interfacciato con RL-Glue [62], un framework che permette di separare i vari aspetti concettuali dell'apprendimento per rinforzo, astraendo dal linguaggio in cui essi sono implementati; esso, quindi, permette di mantenere il codice più ordinato, garantendo inoltre una certa flessibilità e indipendenza nelle scelte effettuate per i vari componenti: ad esempio, nonostante il simulatore sia scritto in Java, i controllori possono essere scritti in altri linguaggi, come C++.

L'architettura, in particolare, prevede un Agente, un Ambiente e un Esperimento che comunicano tra di loro attraverso RL-Glue, come in Figura 4.1. L'utente definisce i parametri di suo interesse nell'Esperimento, che si occupa di comunicarli all'Agente e all'Ambiente. In seguito, l'Esperimento fa partire la simulazione dando i comandi necessari ad Ambiente e Agente; a questo punto, l'Ambiente si occupa di simulare il gruppo di ascensori, mentre l'Agente di fornire l'azione di controllo per esso. Infine,

l'utente chiede attraverso l'Esperimento i risultati della simulazione. La Figura 4.1(b) mostra un diagramma qualitativo di questo flusso di esecuzione.

L'architettura RL-Glue necessita di definire una codifica per lo stato e le azioni e di calcolare un rinforzo da fornire all'agente; questi dettagli vengono descritti nella prossima sezione. Le sezioni successive, invece, approfondiscono la struttura dei tre componenti; in seguito, viene descritta la configurazione del sistema utilizzata per gli esperimenti.

4.1 Struttura dello stato, dell'azione e del rinforzo

RL-Glue richiede che lo stato e l'azione siano codificati con una serie di valori di tipo intero, double o carattere, in modo da garantire una comunicazione semplice tra Agente e Ambiente.

Stato Le feature di stato che abbiamo considerato, che contengono tutte e sole le informazioni disponibili con la configurazione che abbiamo deciso di utilizzare, sono le seguenti:

VALORI DI TIPO INTERO.

tipo ultima richiesta ricevuta (1:requestElevator;2:setNextDirection)

ALTERNATIVA:

caso tipo ultima richiesta ricevuta = requestElevator:

piano a cui c'è la richiesta
direzione richiesta

caso tipo ultima richiesta ricevuta = setNextDirection:

ascensore di cui è richiesta la prossima direzione
valore non significativo

per ogni ascensore ripeto:

piano dove si trova
direzione in cui sta viaggiando (-1:DOWN,0:FERMO,+1:UP)
piano di destinazione dell'ascensore (-1 se non esiste)
numero di persone sull'ascensore
per ogni piano ripeto:

1 se le persone sull'ascensore hanno richiesto quel piano, 0 altrimenti

direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN,0:FERMO,+1:UP)

per ogni piano ripeto:

numero di persone in attesa a quel piano

VALORI DI TIPO DOUBLE.

tempo corrente (in millisecondi)

per ogni piano ripeto:

istante temporale di pressione del bottone up in millisecondi (-1 se non premuto)

istante temporale di pressione del bottone down in millisecondi (-1 se non premuto)

average waiting time
maximum waiting time
average travel time
maximum travel time
average squared waiting time
long waiting percentage
average system time
maximum system time
average crowding
number of travels
integral squared waiting time
integral number of travels
integral crowding
integral waiting time
integral traveling time
integral long waiting percentage
integral system time
integral squared waiting time without approximation

Gli ultimi valori di tipo double contenuti nello stato (da average waiting time compreso in poi) non contengono delle informazioni di stato vere e proprie, ma sono dei rinforzi inseriti nello stato per superare la limitazione imposta da RL-Glue di avere un unico valore di rinforzo; per questo motivo essi verranno trattati nel paragrafo riguardante il rinforzo.

Azione Le feature di azione che abbiamo considerato, che riflettono il tipo di decisione che il controllore può prendere, sono le seguenti:

VALORI DI TIPO INTERO.

per ogni ascensore ripeto:

prossimo piano da visitare (-1 se non esiste)

direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN,0:FERMO,+1:UP)

Rinforzo Dato che RL-Glue non prevede rinforzi multipli, essi sono stati inseriti nello stato. Le quantità utilizzate come rinforzo sono quelle classiche presenti in letteratura (si veda a tal proposito la Sezione 2.2), presenti in versione istantanea e integrale; le quantità temporali sono espresse in secondi.

La versione istantanea del rinforzo, identificabile dal nome della quantità senza alcun prefisso, contiene appunto il valore che la quantità presa in considerazione assume istantaneamente nel momento in cui viene calcolata.

Il gruppo di ascensori, però, è un sistema a eventi discreti, in cui il tempo che intercorre tra un evento e il successivo è variabile. In esso, quindi, non ha senso scontare uniformemente il rinforzo immediato che si ottiene a ogni passo, ma lo sconto deve essere fatto sull'istante temporale; la cifra da ottimizzare, quindi, cambia nel seguente modo [11]:

$$\sum_{t=0}^{\infty} \gamma^t r_t \qquad diventa \qquad \int_0^{\infty} e^{-\beta\tau} r_\tau d\tau \tag{4.1}$$

dove r_t è il rinforzo immediato al tempo discreto t, r_τ è il rinforzo istantaneo al tempo continuo τ, mentre β controlla la velocità del decadimento esponenziale del rinforzo ottenuto, sostituendo praticamente il fattore di sconto discreto γ. Questa versione dei rinforzi è identificata, nell'implementazione, con il nome della quantità preceduta dal prefisso integral.

È utile notare che le quantità in gioco nel nostro caso sono da minimizzare, quindi non sono dei rinforzi, ma dei costi; il problema, comunque, è facilmente risolvibile aggiungendo un segno meno alla quantità calcolata. In particolare, nell'implementazione le quantità sono tenute col segno originale, mentre vengono rese negative dagli esperimenti che le utilizzano.

La quantità utilizzata negli esperimenti descritti nel Capitolo 5 è l'integral squared waiting time; questa scelta, in linea con quella presente in [11], è dovuta al fatto che questa quantità consente di rendere basso il tempo d'attesa medio delle persone, che è l'obiettivo primario del controllore del gruppo di ascensori, privilegiando, grazie alla presenza del quadrato, un abbassamento anche dei tempi di attesa massimi. Dato che questa quantità è quella utilizzata negli esperimenti, abbiamo deciso di inserirla in RL-Glue, oltre che nello stato, anche come rinforzo.

Esplicitando e specializzando l'integrale dell'equazione 4.1, la formula per calcolare questa quantità è, a ogni passo, la seguente:

$$\int_{t_x}^{t_y} \sum_p e^{-\beta(\tau - t_x)} (\tau - t_x + w_p)^2 d\tau$$

dove t_x è l'istante temporale dell'evento precedente, t_y l'istante temporale dell'evento corrente, w_p è, per ogni bottone di chiamata che risulta essere premuto al tempo t_y, la quantità di tempo trascorsa tra la pressione di p e l'istante t_x [3]. Risolvendo l'integrale si ha:

$$\sum_p \left(\frac{\beta^2 {w_p}^2 + 2\beta w_p + 2}{\beta^3} - \frac{e^{-\beta t_y} \left(\beta^2 e^{\beta t_x} {w_p}^2 + \left(2\beta^2 e^{\beta t_x} t_y + \left(2\beta - 2\beta^2 t_x \right) e^{\beta t_x} \right) w_p \right)}{\beta^3} + \right.$$

[3] I tempi ai bottoni di chiamata sono utilizzati come un'approssimazione dei tempi di attesa delle persone nel sistema, dato che questi non sono noti in un sistema reale.

$$\frac{e^{-\beta t_y}\left(\beta^2 e^{\beta t_x} {t_y}^2 + \left(2\beta - 2\beta^2 t_x\right) e^{\beta t_x} t_y + \left(\beta^2 {t_x}^2 - 2\beta t_x + 2\right) e^{\beta t_x}\right)}{\beta^3})$$

Si noti che non è necessario prendere in considerazione i casi limite in cui un bottone viene premuto oppure passa da premuto a non premuto in un istante intermedio tra t_x e t_y perché, per come è strutturato il simulatore, si ha sempre un evento corrispondente a questi cambiamenti.

Le altre quantità integrali presenti nello stato sono calcolate in modo equivalente a quella appena mostrata.

4.2 Ambiente

L'ambiente è, come anticipato, il componente che si occupa di simulare il sistema fisico; esso, quindi, è essenzialmente costituito da elevatorsim a cui sono stati aggiunti l'interfacciamento con RL-Glue, un controllore che semplicemente si occupa di attuare i comandi ricevuti dall'Agente e un generatore di traffico conforme con i risultati presenti in letteratura, dato che quello predefinito non era realistico.

Lo schema generale dell'Ambiente è rappresentato dal diagramma UML di Figura 4.2. La classe Environment si occupa di creare il collegamento tra RL-Glue e il simulatore vero e proprio. Essa, infatti, è in grado di riconoscere i vari messaggi per la configurazione del simulatore (si veda a tal proposito la Sezione 4.2.1) e di salvarli nelle strutture adeguate, ossia SettingSetup, SettingRLGlue, TrafficGenerator, VisualizationMode. Inoltre, essa fa partire la simulazione, interagendo con la classe RemoteSimulator, coordinandosi poi con la classe RemoteController per fornirle l'azione da compiere ottenuta tramite RL-Glue e restituire al framework il nuovo stato. Infine, essa si occupa di fornire, su richiesta, i risultati ottenuti dalla simulazione.

La classe TrafficGenerator, oltre ad avere le informazioni sul tipo di traffico richiesto per l'esperimento, viene invocata da RemoteSimulator per creare effettivamente gli eventi di arrivo delle persone. Come suggerito in letteratura [56], gli arrivi vengono generati seguendo una distribuzione poissoniana, con parametro dato dall'intensità del traffico (come persone/unità di tempo). Il piano di partenza e di arrivo delle persone dipende dalla tipologia del traffico (una cui analisi esaustiva è presente nella Sezione 2.3): per traffico di tipo *uppeak* l'origine è per tutti il piano terra, mentre la distribuzione è scelta uniformemente tra tutti gli altri piani; per traffico di tipo *downpeak*, il ruolo di origine e destinazione è invertito rispetto a quanto si ha per il traffico di tipo uppeak; per traffico di tipo *interfloor*, sia l'origine che la destinazione delle persone è distribuita uniformente tra tutti i piani diversi dal piano terra. Distribuendo il traffico in modo uniforme tra i piani diversi dal piano terra abbiamo supposto che tutti i piani dell'edificio di cui si simula il gruppo di ascensori siano equivalenti, cioè sono popolati da un numero paragonabile di persone; si tratta di un'ipotesi semplificativa che, però, riflette la situazione di molti casi reali.

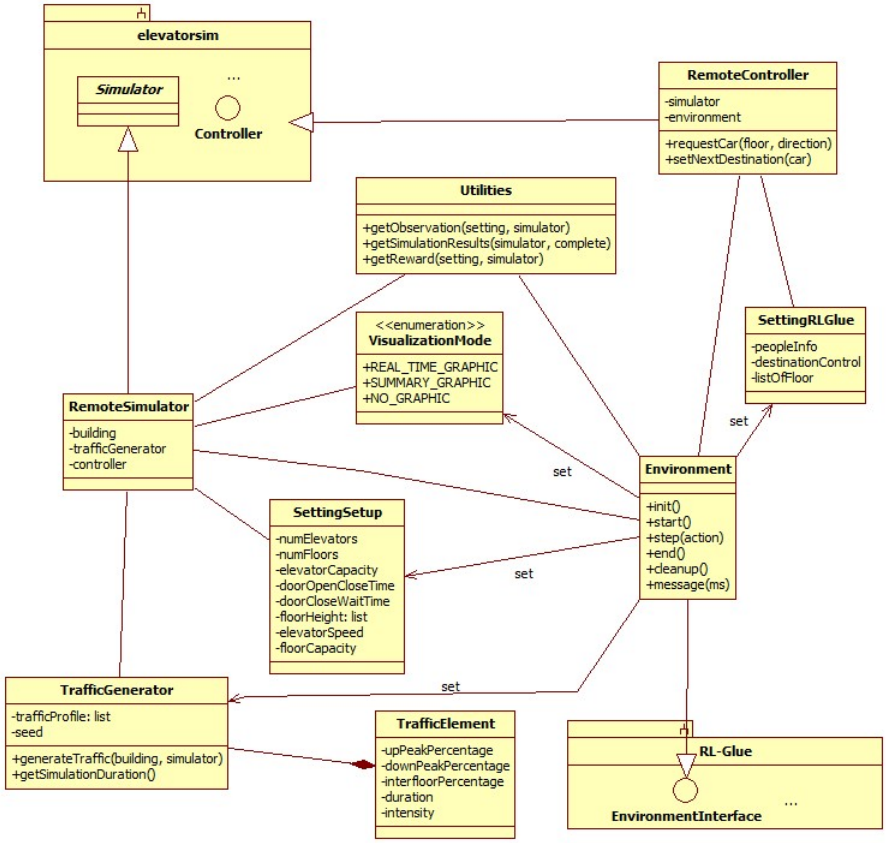


Figura 4.2: Schema della struttura generale dell'Ambiente. Il diagramma contiene solamente gli elementi utili per illustrare questa struttura, mentre i dettagli implementativi sono stati omessi per semplicità e chiarezza.

4.2.1 Messaggi riconosciuti dall'Ambiente

I messaggi riconosciuti dall'ambiente servono per impostare i parametri di simulazione o per ottenere i risultati della simulazione.

Parametri fisici dell'edificio e del gruppo di ascensori. Questi parametri vengono impostati tramite un messaggio *SettingSetup*[4]:

```
SettingSetup:
    numFloors=<val>;
    elevatorCapacity=<val>;
    floorCapacity=<val>;
    numElevators=<val>;
    doorOpenCloseTime=<val>;
    doorCloseWaitTime=<val>;
    elevatorSpeed=<val>;
    floorHeight=(<val>,)*;
```

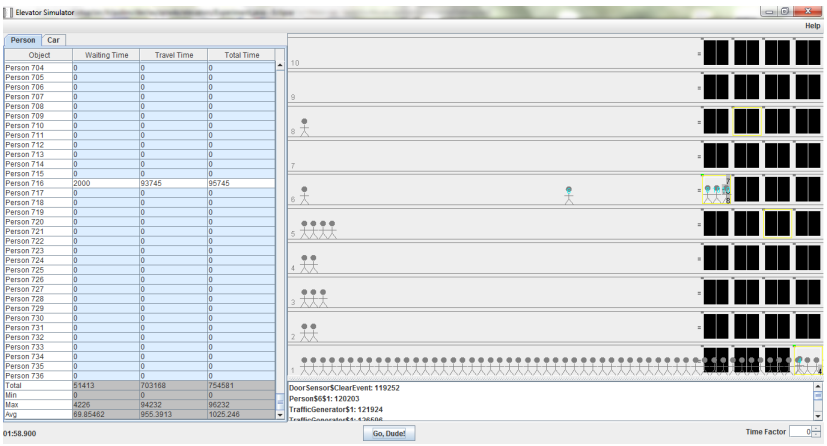
I valori temporali sono espressi in secondi, quelli spaziali in metri. L'ordine dei parametri non è rilevante, mentre se un parametro viene omesso il suo valore è deciso dall'Ambiente. Con floorHeight si intende l'altezza a cui si trova la base di ogni piano; in particolare, il primo valore è l'altezza a cui si trova la base del piano terra, e così via, in ordine di piani; se il numero di valori non è coerente con il numero di piani che costituiscono l'edificio, le altezze vengono adeguate in modo arbitrario dall'Ambiente.

Parametri sulle informazioni disponibili per il controllore. Questi parametri vengono impostati tramite un messaggio *SettingRLGlue*:

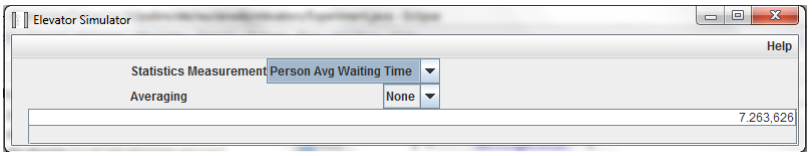
```
SettingRLGlue:
    peopleInfo=<val>;
    destinationControl=<val>;
    listOfFloor=<val>;
```

Tutte e tre le variabili possono assumere valori booleani. La variabile peopleInfo serve per indicare se le informazioni sulle singole persone che usufruiscono del sistema sono disponibili; destinationControl indica se il sistema è, appunto, destinationControl o a due bottoni di chiamata; listOfFloor (omissibile, di default vale false) indica se come azione viene specificato solamente il prossimo piano a cui ogni ascensore

[4]Nel seguito viene utilizzata la notazione tipica delle espressioni regolari; inoltre, con <val>, si intende il valore effettivo da assegnare ai parametri. Si tenga anche presente che gli spazi inseriti servono per migliorare la leggibilità, ma il messaggio nella realtà è costituito da un'unica stringa senza spazi.



(a) realTimeGraphic



(b) summaryGraphic

Figura 4.3: Visualizzazioni grafiche permesse dal simulatore.

deve dirigersi o se vanno indicati, in ordine, tutti i piani a cui ogni ascensore dovrà dirigersi.

Questo messaggio è stato introdotto per predisporre il simulatore a utilizzi futuri con scenari differenti da quello classico; attualmente, però, solo le funzioni che prevedono tutte le variabili a false sono implementate.

Parametri sulla visualizzazione del sistema da utilizzare. Questi parametri vengono impostati tramite un messaggio *VisualizationMode*:

```
VisualizationMode:
    realTimeGraphic|summaryGraphic|noGraphic
```

Se il parametro viene settato a realTimeGraphic si ha una rappresentazione dell'evoluzione del sistema per via grafica, come in Figura 4.3(a); se il parametro viene settato a summaryGraphic si ha una rappresentazione grafica solamente dei risultati della simulazione, come in Figura 4.3(b); se il parametro viene settato a noGraphic non si ha alcuna rappresentazione grafica.

Fattore di sconto. Questo parametro, utilizzato per scontare il rinforzo, viene settato con il seguente messaggio:

```
DiscountFactorBeta:<val>;
```

Traffico. Esso viene configurato con un messaggio di tipo *TrafficPattern*:

```
TrafficPattern:
    randomSeed=<val>;
    (upPeakPerc=<val>,downPeakPerc=<val>,interfloorPerc=<val>,
        duration=<val>,intensity=<val>;)+
```

Come si può notare, il traffico viene definito tramite i profili che esso assume in istanti successivi; ogni profilo è specificato tramite la percentuale di traffico di tipo uppeak, downpeak e interfloor da cui è composto, la durata del profilo, espressa in minuti, e l'intensità di traffico che lo caratterizza, espressa in persone/ora.

Per i tipi di traffico standard sono presenti delle definizioni semplificate:

```
TrafficPattern:
    randomSeed=<val>;
    (UPPEAK|DOWNPEAK|INTERFLOOR),
        duration=<val>,intensity=<val>[,noisePerc=<val>];
```

Il parametro opzionale noisePerc, di default messo a zero e non utilizzabile nel caso INTERFLOOR, indica in che percentuale il traffico non è puramente del tipo indicato.

Inoltre, è possibile impostare una simulazione del traffico tipico di una intera giornata, che ricalca il profilo mostrato in Figura 2.1:

```
TrafficPattern:
    randomSeed=<val>;
    OFFICEDAY,intensity=<val>;
```

Infine, esiste anche la possibilità di creare il traffico in modo più specifico, definendo in un file di testo il comportamento esatto di ogni persona o di gruppi di persone; per impostare questa tipologia di traffico il messaggio da utilizzare è il seguente:

```
TrafficPattern:
    randomSeed=<val>;
    file=<val>;
```

dove il parametro file indica il percorso del file contenente le specifiche del traffico. In particolare, questo file deve essere composto da 5-ple $< n\_pers, p\_part, p\_dest, t\_inizio, t\_fine >$, una per riga, i cui elementi indicano rispettivamente quante persone hanno il comportamento specificato dalla tupla, il piano di partenza e quello di destinazione di queste persone, gli istanti temporali (in millisecondi) tra cui queste persone fanno la richiesta per un ascensore; l'istante temporale specifico per ogni persona viene scelto uniformemente nell'intervallo specificato.

Risultati della simulazione. Il seguente messaggio ottiene, come risposta, i risultati della simulazione:

```
SimulationResults:(statistic|complete)
```

I risutati sono restituiti nel seguente formato:

```
PEOPLE STATISTICS
Waiting Time:min=<val>;max=<val>;avg=<val>;total=<val>;(person<val>=<val>;)*
Travel Time:min=<val>;max=<val>;avg=<val>;total=<val>;(person<val>=<val>;)*
Total Time:min=<val>;max=<val>;avg=<val>;total=<val>;(person<val>=<val>;)*
ELEVATORS STATISTICS
Travel Distances:min=<val>;max=<val>;avg=<val>;total=<val>;(elevator<val>=<val>;)*
Number of Stops:min=<val>;max=<val>;avg=<val>;total=<val>;(elevator<val>=<val>;)*
```

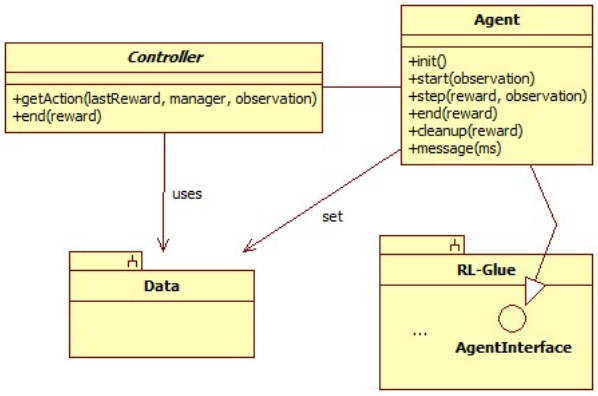
I risultati sulle singole persone e sui singoli ascensori sono inseriti solo se si utilizza l'opzione complete. I valori temporali sono espressi in millisecondi, quelli spaziali in metri.

4.3 Agente

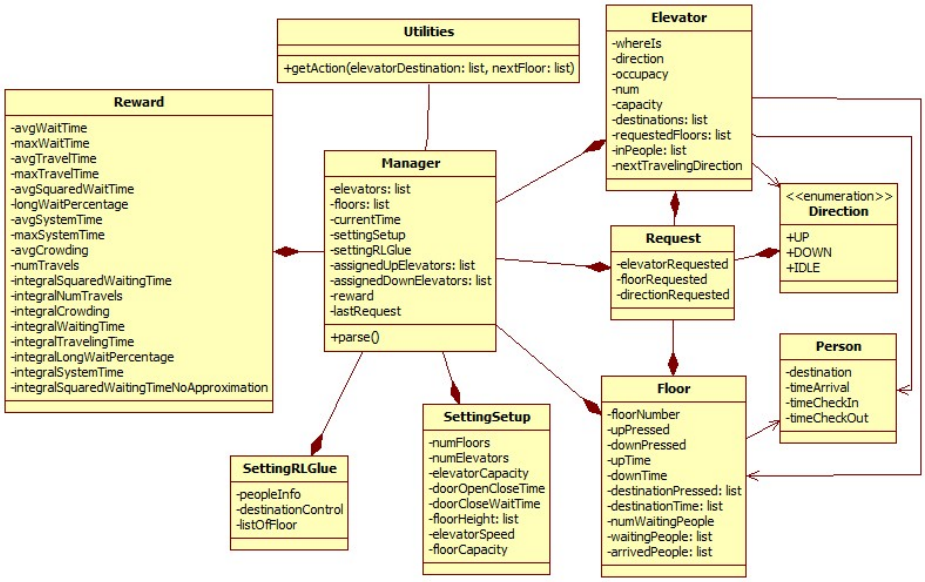
L'agente è il componente che si occupa di effettuare il controllo; esso, quindi, implementa gli algoritmi del sistema di controllo del gruppo di ascensori.

Abbiamo implementato due versioni dell'Agente, una in Java, l'altra in C++; la struttura delle due implementazioni, comunque, è pressoché identica. Il motivo delle due versioni è semplice: se tutti e tre i componenti sono scritti in Java, RL-Glue effettua le comunicazioni in modo diretto, senza servirsi dei socket, ottenendo così delle prestazioni migliori; per questo gli algoritmi benchmark sono implementati nella versione Java. Gli algoritmi di controllo proposti (Capitolo 5), invece, sono scritti in C++, in quanto le librerie di cui necessitano sono disponibili in questo linguaggio di programmazione.

L'Agente è composto da tre elementi principali, mostrati in Figura 4.4(a): l'interfacciamento con RL-Glue, che si occupa delle comunicazioni con il framework; le strutture dati, che immagazzinano tutte le informazioni disponibili sul sistema; il controllore vero e proprio che, dato in input lo stato del sistema, restituisce l'azione che ritiene essere la migliore per il controllo del sistema. Il sottosistema delle strutture dati, a sua volta, viene specializzato come in Figura 4.4(b), per consentire un utilizzo comodo e ordinato delle informazioni a disposizione. Si noti che alcune strutture dati possono essere utilizzate solo in parte o non essere utilizzate, a seconda delle impostazioni di tipo SettingRLGlue selezionate; la versione in C++, inoltre, contiene solamente le strutture necessarie per le impostazioni attualmente supportate dall'Ambiente, mentre le versione in Java le contiene tutte; quale struttura serve



(a) Elementi principali



(b) Strutture dati

Figura 4.4: Schema della struttura generale dell'Agente. Il diagramma contiene solamente gli elementi utili per illustrare questa struttura, mentre i dettagli implementativi sono stati omessi per semplicità e chiarezza.

per ogni tipo di impostazioni deriva direttamente dalle informazioni contenute nello stato, come illustrato nella Sezione 4.1. Il Controllore, invece, è una struttura astratta da cui derivano i controllori concreti.

4.3.1 Messaggi riconosciuti dall'Agente

I messaggi riconosciuti dall'agente servono per fornire le informazioni riguardanti il sistema controllato e per impostare il controllore da utilizzare.

Parametri fisici dell'edificio e del gruppo di ascensori. Questi parametri vengono impostati tramite un messaggio *SettingSetup*, in modo analogo a quanto avviene per l'Ambiente:

```
SettingSetup:
    numFloors=<val>;
    elevatorCapacity=<val>;
    floorCapacity=<val>;
    numElevators=<val>;
    doorOpenCloseTime=<val>;
    doorCloseWaitTime=<val>;
    elevatorSpeed=<val>;
    floorHeight=(<val>,)*;
```

Parametri sulle informazioni disponibili per il controllore. Questi parametri vengono impostati tramite un messaggio *SettingRLGlue*, in modo analogo a quanto avviene per l'Ambiente:

```
SettingRLGlue:
    peopleInfo=<val>;
    destinationControl=<val>;
    listOfFloor=<val>;
```

Questo messaggio viene effettivamente riconosciuto dall'agente scritto in Java, mentre quello scritto in C++ lo ignora e assume implicitamente che tutte le variabili siano impostate a false; questa scelta è stata fatta perché, al momento, il simulatore implementa effettivamente solo questa configurazione.

Creazione di un dataset. Questa opzione viene impostata con il seguente messaggio:

```
CreateDataset:<val>;
```

Il valore è booleano. Se esso è impostato a true, se l'opzione è supportata dalla configurazione scelta, viene creato un dataset contenente la storia vissuta dal

sistema durante la simulazione; la struttura del dataset dipende dalla particolare configurazione scelta.

Scelta del controllore. Il controllore da utilizzare viene impostato tramite un messaggio di tipo *Controller*; la forma di questo messaggio dipende dall'Agente utilizzato; per la versione Java esso assume semplicemente la seguente forma:

```
Controller:<val>
```

Il valore è il nome del controllore, che viene invocato tramite reflection; non è previsto nessun parametro da passare al controllore in questo scenario.

La versione C++ dell'Agente, invece, ha un messaggio personalizzato per ogni tipo di controllore che implementa, in cui, oltre al controllore da utilizzare, viene indicato il valore dei parametri di cui esso necessita. In particolare, i messaggi riconosciuti sono i seguenti due:

```
Controller:FQI(complete|lasso|tree)=<val>
```

```
Controller:HierarchicFQI:
    gamma=<val>;
    lastEpisode=<val>;
    lastRun=<val>;
    endRunInternal=<val>;
    file=<val>;
    MarginalContributionReward=<val>;
    trafficFeature=<val>;
    selectedFeatures=<val>;
```

```
Controller:RLG:
    gamma=<val>;
    GradientEstimator=<val>;
    GradientEstimator_internal=<val>;
    lastEpisode=<val>;
    alpha=<val>;
    alpha_internal=<val>;
    maxSteps=<val>;
    lastRun=<val>;
    endRunInternal=<val>;
    InformationAvailable=<val>;
    learn=<val>;
    parametersFile=<val>;
    Hierarchic=<val>;
    trafficFeature=<val>;
    selectedFeatures=<val>;
```

Il significato di questi true messaggi sarà chiaro quando verranno illustrati i controllori proposti, nel Capitolo 5. Si tenga presente che, nel secondo messaggio, l'ordine dei vari parametri è arbitrario e che il valore di alcuni di essi è significativo o meno a seconda del valore di altri parametri; nel caso un valore non sia significativo, il parametro corrispondente può essere omesso.

Altri messaggi. L'Agente scritto in C++ può riconoscere anche altri due messaggi. Il seguente serve per sapere a che istante temporale delle simulazione si è arrivati:

```
getCurrentSimulationTime()
```

Attualmente, solo il controllore RLG gerarchico è in grado di rispondere in modo significativo; il valore restituito è espresso in minuti. Lo stesso controllore, inoltre, risponde al seguente messaggio, il cui significato sarà chiaro quando illustreremo il controllore in questione:

```
endEpisode
```

4.4 Esperimento

L'Esperimento semplicemente contiene le istruzioni necessarie per configurare il sistema e per eseguire le simulazioni volute. In esso, quindi, abbiamo implementato (in Java) i metodi necessari per effettuare i test descritti e analizzati nel Capitolo 5; inoltre, abbiamo anche aggiunto il necessario per parsare la stringa contente i risultati restituita dall'Ambiente, in modo da poter effettuare delle analisi numeriche sui risultati ottenuti, in particolare mediando i risultati su diverse simulazioni per avere dei dati più robusti e significativi.

4.5 Configurazione utilizzata per gli esperimenti

Negli esperimenti fatti per testare gli algoritmi di controllo del gruppo di ascensori abbiamo definito un sistema configurato come in Tabella 4.1. Come traffico abbiamo utilizzato separatamente le tipologie uppeak puro, downpeak puro e interfloor, in modo da vedere come reagiscono i diversi controllori in situazioni differenti. I diversi tipi di traffico sono stati testati in due configurazioni differenti, cioè traffico leggero (720 persone/ora) e traffico intenso (1500 persone/ora); questa scelta è stata effettuata in modo da poter verificare come variano le prestazioni dei controllori al variare delle condizioni esterne e all'aumentare della difficoltà del problema, in modo da poterne analizzare la robustezza. In alcuni casi, inoltre, abbiamo effettuato delle simulazioni utilizzando il traffico di un'intera giornata lavorativa, con intensità 1000 persone/ora, oppure un traffico misto, di intensità 600 persone/ora, composto

Numero di piani	10
Numero di ascensori	4
Portata degli ascensori (persone)	15
Portata dei piani (persone)	10000
Tempo di apertura/ chiusura delle porte degli ascensori (s)	1,5
Tempo di attesa prima della chiusura delle porte degli ascensori (s)	2
Velocità degli ascensori (m/s)	2,5
Altezza di ogni piano (m)	3

Tabella 4.1: Configurazione del sistema utilizzata negli esperimenti.

per il 15% da componente uppeak, 15% componente downpeak e 70% componente interfloor.

Al fine di avere dei dati significativi, abbiamo testato ogni condizione su un'ora di simulazione, mediando i risultati ottenuti su 500 diverse simulazioni.

Come fattore di sconto β utilizzato nel calcolo dei costi integrali abbiamo utilizzato in alcuni casi il valore 0,01, ricalcando la scelta effettuata in [11], mentre in altri casi il valore 0,0001, in modo da permettere agli agenti che apprendono di avere una vista sufficientemente lunga sul futuro, consentendo loro di poter valutare gli effetti delle azioni intraprese su un orizzonte temporale significativamente lungo.

Capitolo 5

Risultati Sperimentali

In questo capitolo analizziamo i risultati ottenuti dai controllori di gruppi di ascensori che abbiamo progettato, confrontandoli con alcuni controllori di benchmark, scelti tra quelli proposti in letteratura.

La prossima sezione contiene un riferimento ai controllori benchmark e illustra le loro prestazioni nelle situazioni di interesse per i nostri studi.

Le sezioni successive, invece, contengono una descrizione dettagliata dei controllori proposti e i risultati ottenuti da essi. In particolare, per prima cosa descriviamo la struttura generale dei controllori; in seguito, proseguiamo descrivendo i controllori specifici, costruiti con le strutture esposte in precedenze, e i risultati da essi ottenuti, confrontandoli con quelli ottenuti dai benchmark.

Le architetture proposte sono due: una piatta, che delega la totalità del controllo agli agenti che apprendono, e una gerarchica, che suddivide il controllo su due differenti livelli di gerarchia, delegando solo la componente strategica del controllo, in modo da rendere più semplice il compito degli agenti che apprendono. Inoltre, per entrambe le strutture abbiamo provato ad applicare la selezione delle feature, in modo da ridurre la dimensionalità della formalizzazione iniziale del problema e permettere agli agenti di avere un apprendimento più rapido. Purtroppo, dei metodi proposti nel Capitolo 3 abbiamo potuto testare solamente quello euristico basato su alberi, in quanto i tempi computazionali del metodo garantito sono risultati essere proibitivi: risolvere tanti modelli ridotti, infatti, si è rivelato decisamente più lungo rispetto a risolvere il modello completo.

I controllori specifici che proponiamo sono basati sulle architetture appena menzionate e sono divisibili in due classi: quelli che apprendono tramite fitted Q iteration (FQI) e quelli che apprendono tramite metodi gradiente (RLG). Di questi controllori analizziamo le prestazioni, confrontando i risultati ottenuti con l'insieme completo delle feature e con l'insieme ridotto, oltre ovviamente a confrontarne le prestazioni rispetto ai benchmark.

(a) Tempo di attesa medio

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	4,448841	7,180169	10,31465	7,174503	7,806833	5,901691
downpeak	37,56491	13,39817	14,67839	13,89361	26,71041	18,47375
interfloor	31,20921	20,09089	15,23864	18,34249	38,38411	19,03251

(b) Tempo di attesa massimo

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	27,55416	28,34412	93,10261	28,32959	49,9335	30,17004
downpeak	100,1521	68,48423	60,8611	71,59984	125,2858	112,589
interfloor	121,8372	317,3788	96,62565	262,6973	175,8795	145,4209

Tabella 5.1: Tempo di attesa medio e massimo, in secondi, ottenuto dai controllori benchmark con traffico leggero.

5.1 Controllori benchmark

Come benchmark per confrontare i nostri algoritmi di controllo abbiamo deciso di utilizzare alcuni dei controllori euristici attualmente più diffusi e utilizzati nei sistemi reali e un rappresentate per ciascuna delle due aree dell'intelligenza artificiale maggiormente sfruttate per la progettazione di controllori intelligenti, ossia la logica fuzzy e gli algoritmi genetici. I controllori euristici selezionati, una cui descrizione è presente nel Capitolo 2, sono CollectiveControl, EarliestCarControl, NearestCarControl, SchedulingMethod; come rappresentante dell'area fuzzy abbiamo scelto il Fuzzy area-based control algorithm presentato nella Sezione III di [32] (in seguito denotato con FuzzyController), in quanto riesce a ottenere delle buone prestazioni nonostante una struttura molto semplice; come rappresentante dell'area degli algoritmi genetici abbiamo selezionato il controllore proposto in [10] (in seguito denotato con GAHCAController), per la sua progettazione non convenzionale[1].

In Tabella 5.1 e 5.2 sono riportati i tempi di attesa medi e massimi[2], ottenuti mediando i risultati singoli su 500 simulazioni da 60 minuti l'una, sperimentati in caso di traffico rispettivamente leggero (720 persone/ora) e intenso (1500 persone/ora) con il sistema controllato dai diversi algoritmi benchmark, in condizioni di traffico uppeak, downpeak, interfloor; i tempi di attesa medi sono riportati rispettivamente anche nei grafici di Figura 5.1 e 5.2.

[1] Nelle tabelle che riassumono i risultati ottenuti da questi controllori, i loro nomi vengono abbreviati in Collec, Earliest, Fuzzy, Nearest, Schedul, GAHCA, corrispondenti rispettivamente CollectiveControl, EarliestCarControl, FuzzyController, NearestCarControl, SchedulingMethod e GAHCAController.

[2] Tutte le statistiche dei tempi di attesa, viaggio e sistema relativi a una simulazione sono riferite ai tempi sperimentati dalle diverse persone presenti nel sistema simulato; similmente, il numero totale di fermate di una simulazione è calcolato sommando il numero di fermate di ciascun ascensore del sistema simulato.

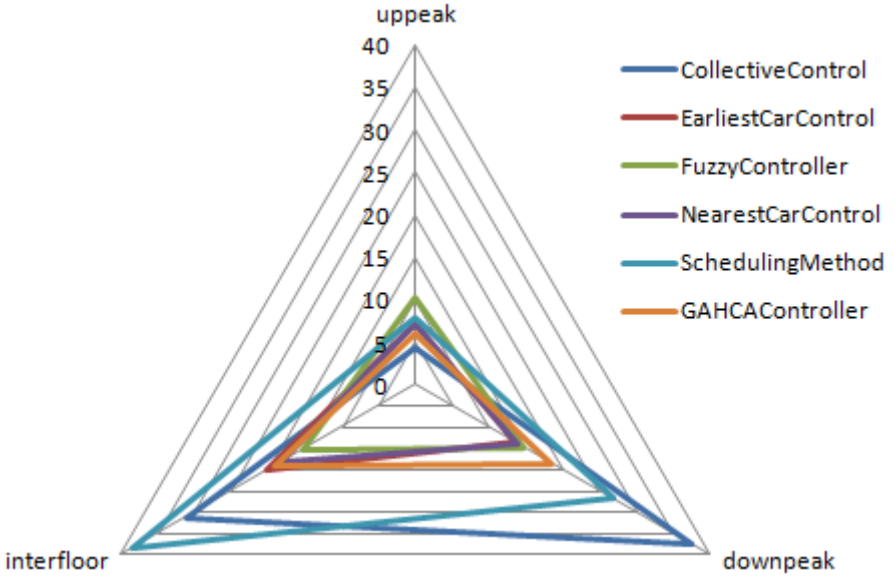


Figura 5.1: Tempo di attesa medio, in secondi, ottenuto dai controllori benchmark con traffico leggero.

(a) Tempo di attesa medio

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	16,713	472,600	56,067	475,026	152,493	21,502
downpeak	66,672	37,988	32,116	39,194	112,893	42,179
interfloor	116,178	427,485	40,666	405,239	121,783	66,840

(b) Tempo di attesa massimo

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	67,905	980,567	160,952	981,741	480,188	78,642
downpeak	548,704	330,940	203,377	338,895	1375,204	277,970
interfloor	295,051	3665,907	267,765	3640,612	388,993	644,213

Tabella 5.2: Tempo di attesa medio e massimo, in secondi, ottenuto dai controllori benchmark con traffico intenso.

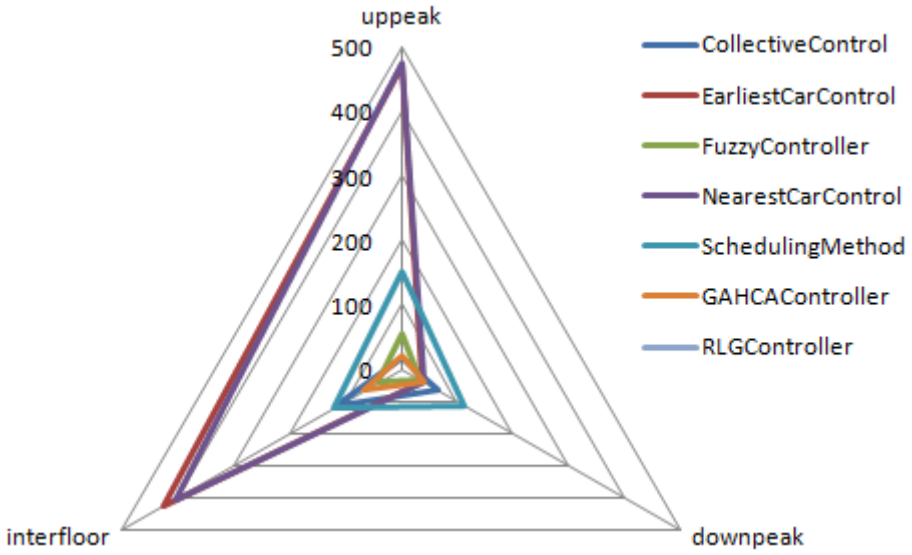


Figura 5.2: Tempo di attesa medio, in secondi, ottenuto dai controllori benchmark con traffico intenso.

Come si può notare dai dati e dai grafici riportati, i controllori benchmark reagiscono differentemente a diversi schemi di traffico; ad esempio, CollectiveControl si dimostra essere il controllore più efficiente in uppeak, mentre è tra i peggiori nelle altre situazioni di traffico. Questo suggerisce che applicare un controllore unico a condizioni di traffico differenti non è una buona idea, ma conviene specializzarli per i casi uppeak, downpeak e interfloor, anche se questo implica la necessità di saper distinguere che tipo di traffico è presente nel momento del controllo.

Un altro aspetto che si nota dai dati è la notevole differenza di prestazioni che i controllori benchmark hanno al variare del sistema, in particolare dell'intensità del traffico; ad esempio, EarliestCarControl e NearestCarControl sono tra i migliori con traffico leggero e tra i peggiori con traffico intenso. Questo implica che un controllore studiato per un certo sistema può non funzionare adeguatamente bene su un altro sistema e, ancora più grave, può diventare inadeguato a fronte di cambiamenti del sistema su cui è installato.

In Tabella 5.3 e 5.4 sono riportate le misure di altri indici significativi ottenute dai controllori benchmark in condizioni di traffico rispettivamente leggero e intenso, ottenute anche in questo caso mediando i risultati singoli su 500 simulazioni da 60 minuti ciascuna. Da questi dati si evince che il tempo di viaggio medio peggiora, all'aumento del traffico, in modo abbastanza contenuto e giustificabile con l'aumento del numero di persone da servire; questo è giustificabile con il fatto che il comportamento degli ascensori con a bordo dei passeggeri è vincolato dalle regole di Closs (elencate nella Sezione 2.4) e, quindi, essi hanno meno gradi di libertà per effettuare le loro scelte. Il numero di fermate degli ascensori, invece, rimane paragonabile sia

(a) Tempo di viaggio medio, in secondi

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	36,773	61,812	36,651	61,823	63,540	32,353
downpeak	34,420	30,125	18,063	31,793	39,156	17,682
interfloor	25,057	29,871	18,545	30,136	36,953	17,838

(b) Tempo di sistema medio, in secondi

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	41,222	68,992	46,965	68,998	71,347	38,255
downpeak	71,985	43,523	32,742	45,686	65,866	36,156
interfloor	56,266	49,962	33,784	48,478	75,337	36,870

(c) Numero totale di fermate degli ascensori

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	821,724	480,304	690,92	480,43	719,956	746,204
downpeak	1318,72	747,036	887,716	713,826	861,048	943,996
interfloor	1294,29	968,856	1056,21	959,446	1091,292	1106,52

Tabella 5.3: Risultati ottenuti dai controllori benchmark con traffico leggero.

(a) Tempo di viaggio medio, in secondi

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	59,218	63,908	56,874	63,913	64,602	56,018
downpeak	46,279	43,023	30,303	43,223	44,436	28,321
interfloor	52,872	56,431	41,762	57,140	63,991	43,725

(b) Tempo di sistema medio, in secondi

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	75,931	536,508	112,941	538,939	217,095	77,521
downpeak	112,952	81,011	62,418	82,418	157,330	70,500
interfloor	169,051	483,916	82,429	462,379	185,774	110,564

(c) Numero totale di fermate degli ascensori

	Collec	Earliest	Fuzzy	Nearest	Schedul	GAHCA
uppeak	907,63	844	882,132	843,974	955,208	912,096
downpeak	863,676	625,726	755,25	615,966	803,83	734,798
interfloor	915,828	979,934	1007,266	970,4	825,99	962,292

Tabella 5.4: Risultati ottenuti dai controllori benchmark con traffico intenso.

con traffico leggero sia intenso; questo è dovuto al fatto che con traffico intenso il numero di persone servite con un'unica fermata è, solitamente, maggiore.

5.2 Architetture dei controllori proposti

Nella progettazione di un controllore del gruppo si ascensori che, tramite tecniche di apprendimento per rinforzo, cerca di imparare la politica migliore da attuare abbiamo considerato due architetture differenti: una piatta, in cui abbiamo lasciato completa libertà all'apprendimento, e una gerarchica, in cui abbiamo definito due livelli gerarchici di apprendimento che limitano lo spazio delle politiche attuabili.

5.2.1 Architettura piatta

L'architettura piatta prevede un agente che apprende per ogni ascensore, completamente independente dagli altri agenti, in modo da evitare l'esplosione del numero di azioni che si avrebbe con un controllore centralizzato. In questa architettura ogni agente deve specificare il comportamento dell'ascensore sotto il suo controllo in modo diretto, cioè indicandone la destinazione e la direzione futura, che vengono poi passati senza alcuna modifica al sistema controllato.

Le feature di stato che abbiamo reso disponibili agli agenti in questo scenario sono le seguenti che sono, in numero, 16 + numeroAscensori*27 + numeroPiani*5 + numeroAscensori*numeroPiani:

tipo ultima richiesta ricevuta (1:requestElevator;2:setNextDirection)
piano a cui c'è la richiesta in caso requestElevator, -1 altrimenti
direzione richiesta in caso requestElevator, -1 altrimenti
ascensore di cui è richiesta la prossima destinazione in caso setNextDirection, -1 altrimenti
per ogni ascensore ripeto:
    piano dove si trova
    direzione in cui sta viaggiando (-1:DOWN,0:FERMO,+1:UP)
    piano di destinazione dell'ascensore (-1 se non esiste)
    numero di persone sull'ascensore
    per ogni piano ripeto:
        1 se le persone sull'ascensore hanno richiesto quel piano, 0 altrimenti
        direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN,0:FERMO,+1:UP)
istante temporale corrente (in millisecondi)
per ogni piano ripeto:
    istante temporale di pressione del bottone up in millisecondi (-1 se non premuto)
    istante temporale di pressione del bottone down in millisecondi (-1 se non premuto)
per ogni ascensore ripeto:

ultimo piano con una richiesta nella direzione corrente

ultimo piano con una richiesta nella direzione opposta

ultimo piano, prima del corrente, con una richiesta nella direzione corrente

è al piano inferiore? (1/0)

è al piano superiore? (1/0)

è al piano col passeggero in attesa da più tempo? (1/0)

numero di richieste di destinazioni da parte dei passeggeri a bordo

numero di chiamate da piani superiori rispetto alla posizione corrente, direzione alto

numero di chiamate da piani superiori rispetto alla posizione corrente, direzione basso

numero di chiamate da piani superiori rispetto alla posizione corrente, qualsiasi direzione

numero di chiamate da piani inferiori rispetto alla posizione corrente, direzione alto

numero di chiamate da piani inferiori rispetto alla posizione corrente, direzione basso

numero di chiamate da piani inferiori rispetto alla posizione corrente, qualsiasi direzione

numero di richieste di destinazioni da parte dei passeggeri a bordo per piani lungo la direzione di marcia corrente

numero di richieste di destinazioni da parte dei passeggeri a bordo per piani compresi tra la nuova chiamata, o -1 se la richiesta non è per una nuova chiamata

controllo se la chiamata emersa coincide con una chiamata a bordo dell'ascensore (1 se coincide, 0 se non coincide, -1 se la richiesta non è per una nuova chiamata)

distanza, in numero di piani, tra questo ascensore e quello più vicino

distanza, in numero di piani, tra questo ascensore e il piano dove è emersa una nuova chiamata, considerando anche la direzione di marcia dell'ascensore, o -1 se la richiesta non è per una nuova chiamata

numero di destinazioni richieste da parte dei passeggeri a bordo per piani compresi tra la nuova chiamata e la posizione corrente, considerando anche la direzione di marcia, o -1 se la richiesta non è per una nuova chiamata

l'ascensore è inutilizzato? (1/0)

l'ascensore ha dei posti liberi? (1/0)

numero di posti liberi sull'ascensore

per ogni piano ripeto:

numero di ascensori al piano

bottone di chiamata verso l'alto è premuto? (1/0)

bottone di chiamata verso il basso è premuto? (1/0)

ascesore con a bordo il numero minore di passeggeri

ascensore con a bordo il secondo numero minore di passeggeri

ascensore con il numero minore di destinazioni richieste da passeggeri a bordo

ascensore con il secondo numero minore di destinazioni richieste da passeggeri a bordo

numero di richieste di ascensori, direzione alto

numero di richieste di ascensori, direzione basso

numero di richieste di ascensori, direzione qualsiasi

piano in cui c'è la chiamata col tempo di attesa maggiore

direzione della chiamata col tempo di attesa maggiore (-1: basso; 1: alto)

piano in cui c'è la chiamata col tempo di attesa secondo maggiore

direzione della chiamata col tempo di attesa secondo maggiore (-1: basso; 1: alto)

Come si può notare, la prima parte dell'insieme di feature contiene le informazioni di stato disponibili, mentre le altre sono quantità non banali calcolabili dallo stato. Nel problema definito per i nostri test le feature sono 214.

L'azione, invece, è definita dalle seguenti feature, in numero 2*numeroAscensori:

per ogni ascensore ripeto:

prossimo piano da visitare (-1 se non esiste)

direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN, 0:FERMO, +1:UP)

Esse, quindi, corrispondono alla codifica utilizzata per l'azione, coerentemente con il fatto che il controllo viene effettuato direttamente dagli agenti che apprendono. Il numero totale di azioni che esse generano nel sistema utilizzato, in uno scenario centralizzato, è $11^4 \cdot 3^4 = 1185921$, decisamente troppo elevato; per questo motivo l'apprendimento è fatto con ascensori indipendenti e, quindi, l'azione è costituita solo dal prossimo piano e dalla prossima direzione dell'ascensore in questione.

Non tutte le azioni sono sceglibili in qualsiasi stato, perché possono capitare situazioni in cui un'azione violi le regole di Closs o l'impossibilità per un ascensore di invertire direzione di marcia mentre è in movimento; affinché queste regole vengano rispettate è necessario porre dei vincoli alle azioni apprendibili dagli agenti che utilizzano l'architettura piatta.

Il rinforzo che abbiamo utilizzato in questo scenario, infine, è costituito dall'opposto del quadrato del tempo di attesa medio integrale, calcolato in secondi.

Per diminuire la dimensione del problema abbiamo usato sia un metodo basato su Lasso [64],[37] sia il metodo euristico basato sugli alberi proposto nel capitolo 3. Per fornire agli algoritmi di riduzione delle feature i dati riguardanti il nostro problema abbiamo creato un dataset simulando il sistema per 105 ore, suddivise nel seguente modo:

- 10 ore di uppeak puro, 720 persone/ora
- 10 ore di downpeak puro, 720 persone/ora
- 10 ore di interfloor puro, 720 persone/ora
- traffico di 5 giornate d'ufficio (10 ore per giornata), 1000 persone/ora come traffico massimo
- 10 ore di traffico misto (15% uppeak, 15% downpeak, 70% interfloor), 550 persone/ora

Durante queste simulazioni, il controllo del sistema è stato diviso, uniformemente per ogni tipologia, tra i controllori benchmark[3]; in questo modo nel dataset è presente un numero abbastanza significativo di strategie di controllo. Il dataset risulta essere composto da circa 100000 tuple.

Il metodo basato su Lasso non ha prodotto nessuna selezione significativa; esso, infatti, al variare dei parametri in ingresso ci ha restituito o una selezione contenente praticamente nessuna feature, oppure una selezione contenente più della metà delle feature proposte. Abbiamo comunque deciso di testare i risultati ottenibili utilizzando la selezione che introduce un numero elevato di variabili.

Il metodo basato sugli alberi ci ha, invece, permesso di ottenere una riduzione decisamente più significativa; in particolare, le feature che esso ha selezionato sono le seguenti:

ascensore di cui è richiesta la prossima destinazione in caso setNextDirection, -1 altrimenti

piano dove si trova l'ascensore 1,2,3

direzione in cui sta viaggiando l'ascensore 1,2,3

numero di richieste di asensori, direzione qualsiasi

prossimo piano da visitare per l'ascensore 1,2,3 (azione)

Una visualizzazione grafica, in cui sono mantenuti l'ordine e le dipendenze di scelta delle feature, è riportata in Figura 5.3.

Come si può notare, gli ascensori 1,2 e 3 sono trattati allo stesso modo, mentre non è selezionata nessuna variabile riguardante l'ascensore 4; questo è spiegabile dal fatto che gli algoritmi utilizzati per la generazione del dataset, in generale, per come sono implementati, tendono a usare maggiormente gli ascensori con indice più basso. Per avere uno scenario più generale, quindi, se si decide di utilizzare solamente le feature selezionate con questo metodo può essere utile aggiungere alle variabili selezionate quelle di tutti gli ascensori del sistema assimilabili a quelle effettivamente selezionate; anche nel caso in cui non si effettua questa generalizzazione, comunque, è necessario introdurre tutte le azioni disponibili, altrimenti diventerebbe impossibile controllare completamente il sistema.

5.2.2 Architettura gerarchica

La versione gerarchica è studiata per semplificare il compito degli agenti che apprendono, definendo il comportamento del sistema quando questo è vincolato e delegando solamente le scelte strategiche; in questo modo dovrebbe essere più facile per gli agenti riuscire ad apprendere una buona politica per il controllo del sistema, senza diminuirne le capacità potenziali.

[3]Il controllore genetico è stato escluso dal controllo durante la creazione del dataset perché, quando lo abbiamo generato, non avevamo ancora a disposizione una sua implementazione.

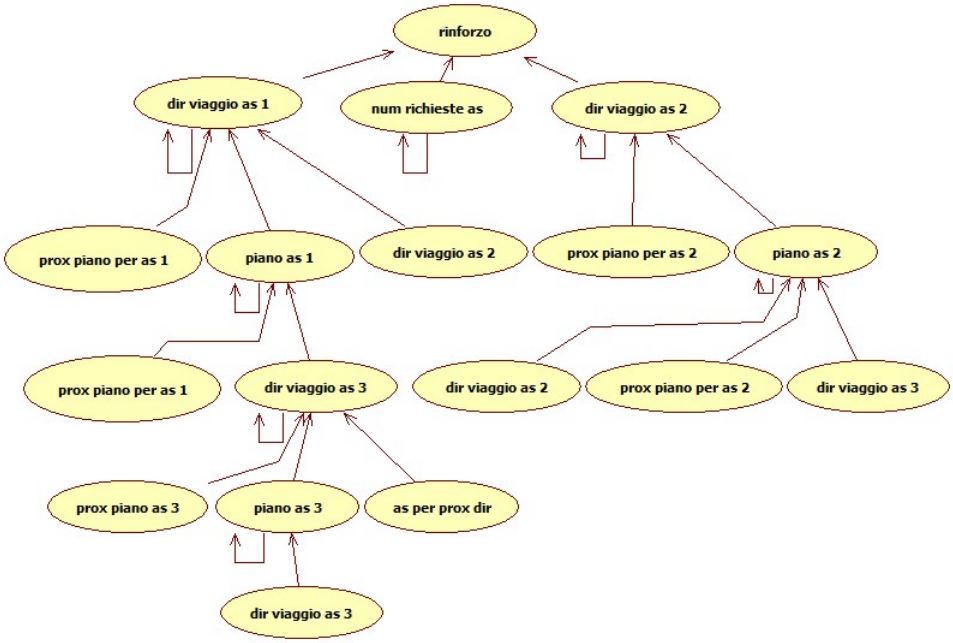


Figura 5.3: Feature selezionate dal metodo basato su alberi per l'architettura piatta. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte.

Questa architettura è chiamata gerarchica perché è strutturata su due livelli decisionali, in ordine gerarchico il livello con ascensori vuoti e il livello con ascensori pieni, più una parte pre-programmata (non soggetta a processi di apprendimento) che gestisce i due livelli decisionali e interpreta le loro decisioni per trasformarle in un'azione comprensibile all'Ambiente.

Entrambi i livelli decisionali sono caratterizzati da agenti, uno per ogni ascensore, separati per l'apprendimento, ma non totalmente indipendenti. In questa architettura, infatti, abbiamo deciso di tenere una politica unica (una per ogni livello) e di utilizzare le traiettorie di tutti gli agenti per l'apprendimento di questa politica. L'unificazione della politica non porta a nessuna limitazione, in quanto il comportamento degli ascensori è indipendente dal particolare ascensore, ma dipende solo dalla vista che l'ascensore ha sul sistema; inoltre, questa scelta permette, a parità di numero di simulazioni, di avere più dati per l'apprendimento e, quindi, di ottenere dei risultati più robusti e in meno tempo[4].

Livello con ascensori pieni. Il livello con ascensori pieni è il livello più basso nella gerarchia e prende le decisioni strategiche relative al comportamento degli ascensori con persone a bordo.

In questo scenario, un ascensore che rispetta le regole di Closs può solamente proseguire nella sua direzione di marcia, fermandosi obbligatoriamente ai piani richiesti dalle persone a bordo e opzionalmente a quelli in cui c'è una richiesta nella direzione di marcia, fino a quando non ci sono più persone a bordo oppure tutte le persone rimaste a bordo necessitano, per essere servite, un'inversione di marcia dell'ascensore[5]. Nel primo caso lo scenario è cambiato (e quindi il controllo non è più a questo livello), mentre nel secondo caso l'ascensore deve invertire il senso di marcia e poi continuare a seguire le regole appena descritte.

L'agente che apprende a questo livello è progettato in modo che qualsiasi decisione prenda non violi le regole sopra esposte. In particolare, esso deve decidere se l'ascensore deve fermarsi o meno ai piani dove c'è una richiesta per la direzione in cui sta viaggiando l'ascensore in questione. Nella pratica, l'agente riceve il controllo quando una persona sale a bordo di un'ascensore vuoto. Data la direzione di marcia dell'ascensore, se non è presente nessuna richiesta nei corridoi servibile dall'ascensore compresa tra la sua posizione e il primo piano richiesto a bordo, l'ascensore si

[4] La decisione di tenere una politica unica per ogni ascensore sottointende l'assunzione che tutti gli ascensori sono equivalenti, cioè non sono previsti i servizi speciali descritti nella Sezione 2.1, in linea con le scelte esposte all'inizio di questo capitolo. Se invece si volesse prevedere anche dei servizi speciali, cosa comunque abbastanza rara, sarebbe necessario definire una politica separata per ogni gruppo di ascensori dedicato a un certo servizio.

[5] Il secondo caso prospettato teoricamente non potrebbe accadere, ma nella pratica capita quando una persona sale a bordo di un ascensore che ha dichiarato che continuerà nella direzione opposta a quella in cui deve viaggiare la persona in questione; questo scenario si verifica nella realtà così come nel simulatore utilizzato.

dirige a quel piano senza la necessità di prendere alcuna decisione; altrimenti, l'agente deve decidere se fermare l'ascensore alla prima richiesta nei corridoi servibile dall'ascensore compresa tra la sua posizione e il primo piano richiesto a bordo. Se la decisione è di fermarsi, l'ascensore si dirige al piano della richiesta nei corridoi; altrimenti, si ripete il comportamento appena descritto, eliminando dall'insieme di possibili richieste nei corridoi servibili dall'ascensore quella corrispondente al piano a cui si è deciso di non fermarsi[6]. La direzione che l'ascensore dichiara di seguire una volta arrivato a destinazione è sempre quella in cui sta viaggiando (al primo passo quella che ha dichiarato prima che salisse la prima persona a bordo); una volta servite tutte le persone nella direzione di viaggio, se sono rimaste altre persone a bordo semplicemente inverte la direzione di marcia e continua come appena descritto. Quando tutte le persone a bordo dell'ascensore hanno richiesto lo stesso piano e, in seguito a una decisione o agli eventi accaduti, la destinazione dell'ascensore coincide con questo piano, il controllo torna al livello superiore e la direzione con cui proseguirà l'ascensore dipende dalla decisione presa al livello superiore; così facendo si preclude la possibilità di cambiare decisione nel caso nel sistema accada un evento prima che l'ascensore arrivi a destinazione, ma non si può fare altrimenti, perché non è detto che tale evento accada, mentre a noi serve la certezza di poter specificare completamente il comportamento dell'ascensore.

Le feature che ogni agente ha a disposizione sono 31 e non dipendono dalla particolare configurazione del sistema; queste feature, ovviamente, contengono solamente le informazioni utili nel caso un ascensore abbia a bordo delle persone. Nel dettaglio, esse sono:

per l'ascensore in questione:

- piano dove si trova
- prossimo piano richiesto dalle persone a bordo
- prossimo piano dove potrebbe fermarsi
- numero di persone a bordo
- tempo di attesa della richiesta al prossimo piano dove potrebbe fermarsi
- numero di chiamate da piani lungo la direzione di marcia, per andare nella direzione di marcia
- numero di chiamate da piani lungo la direzione di marcia, per andare nella direzione opposta a quella di marcia
- numero di chiamate da piani lungo la direzione di marcia, per andare in qualsiasi direzione
- numero di chiamate da piani lungo la direzione opposta a quella di marcia, per andare nella direzione di marcia
- numero di chiamate da piani lungo la direzione opposta a quella di marcia, per andare nella direzione opposta a quella di marcia

[6] Si noti che a un passo dell'architettura generale implementata attraverso RL-Glue possono corrispondere più passi a questo livello decisionale.

numero di chiamate da piani lungo la direzione opposta a quella di marcia, per andare in qualsiasi direzione

numero di richieste in attesa da più tempo di quella del prossimo piano dove potrebbe fermarsi

numero di altri piani dove potrebbe decidere di fermarsi prima del prossimo piano richiesto dalle persone a bordo

numero di richieste a bordo

numero di richieste a bordo per la direzione di marcia

numero di posti liberi a bordo

per la situazione generale del sistema:

numero di altri ascensori che devono fermarsi al prossimo piano dove potrebbe fermarsi l'ascensore in questione

numero di altri ascensori diretti al prossimo piano dove potrebbe fermarsi l'ascensore in questione e che continueranno nella direzione di marcia dell'ascensore in questione

numero di altri ascensori fermi al prossimo piano dove potrebbe fermarsi l'ascensore in questione e che continueranno nella direzione di marcia dell'ascensore in questione

massimo del tempo d'arrivo stimato per gli altri ascensori al prossimo piano dove potrebbe fermarsi l'ascensore in questione

minimo del tempo d'arrivo stimato per gli altri ascensori al prossimo piano dove potrebbe fermarsi l'ascensore in questione

media del tempo d'arrivo stimato per gli altri ascensori al prossimo piano dove potrebbe fermarsi l'ascensore in questione

direzione della chiamata col tempo di attesa secondo maggiore (-1: basso; 1: alto)

piano in cui c'è la chiamata col tempo di attesa secondo maggiore

tempo di attesa secondo maggiore

direzione della chiamata col tempo di attesa maggiore (-1: basso; 1: alto)

piano in cui c'è la chiamata col tempo di attesa maggiore

tempo di attesa maggiore

numero di richieste di ascensori, direzione qualsiasi

numero di richieste di ascensori, direzione basso

numero di richieste di ascensori, direzione alto

Le quantità temporali sono espresse in secondi, divisi per 100 per evitare problemi numerici. Si noti che l'istante di pressione dei bottoni di chiamata utilizzato nell'architettura piatta è stato sostituito dal tempo passato dalla pressione dei bottoni, in quanto questa è un'informazione lineare e quindi più facilmente utilizzabile.

Per il tempo di attesa stimato per gli arrivi degli ascensori a un piano abbiamo utilizzato un adattamento della stima proposta dal *minimum long-wait algorithm* utilizzato in [32]. In particolare, il tempo viene stimato col tempo necessario all'ascensore per andare dal piano dove si trova al piano richiesto più il tempo necessario per fermarsi a tutti i piani intermedi richiesti dalle persone a bordo; nel calcolo del

tempo per andare dal piano dove si trova al piano richiesto vengono considerate anche le direzioni: se, ad esempio, il piano richiesto non si trova lungo la direzione di marcia dell'ascensore, bisogna calcolare il tempo per arrivare all'ultimo piano da servire nella direzione di marcia e poi ripartire nella direzione opposta; inoltre, se la direzione richiesta è opposta a quella in cui sta viaggiando l'ascensore all'arrivo al piano, il calcolo del tempo continua facendo proseguire l'ascensore fino all'ultimo piano nella direzione in cui sta viaggiando e poi invertendone la direzione fino ad arrivare nuovamente al piano richiesto, stavolta anche nella direzione richiesta. Si noti che nella stima non sono considerate le fermate alle richieste ai piani, in quanto queste variano a ogni evento e quindi non sono significative fino a quando non vengono effettivamente servite.

Le azioni tra cui gli agenti possono scegliere sono semplicemente due, corrispondenti rispettivamente al fermarsi e non fermarsi al prossimo piano a cui l'ascensore potrebbe fermarsi.

Per il rinforzo da assegnare agli agenti abbiamo considerato due alternative. La prima (a cui ci riferiremo con il termine rinforzo generale) prevede che il rinforzo ottenuto a ogni passo dagli agenti è, in generale, 0. All'ultimo passo, prima di cedere il controllo al livello superiore, l'agente ottiene la differenza tra l'opposto del quadrato del tempo di attesa medio integrale in secondi, diviso per 10^5, in quell'istante e quello calcolato nel momento in cui il controllo era passato all'agente, in modo da poter valutare quanto la situazione è migliorata grazie alle decisioni prese da questo agente. Inoltre, quando in seguito alla decisione di fermarsi a un piano intermedio un ascensore si trova fermo alla stesso piano in cui sono fermi uno o più altri ascensori che continueranno nella stessa direzione, l'agente che ha preso la decisione di fermarsi ottiene una piccola penalità; in base ad alcune prove sperimentali effettuate, abbiamo fissato questa penalità a 0,02, un valore tra i due e i tre ordini di grandezza più piccolo del massimo che solitamente si ottiene per il quadrato del tempo di attesa medio integrale in secondi, diviso per 10^5. Questa penalità è necessaria perché altrimenti gli agenti, che a questo livello hanno una vista abbastanza ridotta sul futuro, tendono a fermarsi a quasi tutte le richieste intermedie, trovandosi spesso in più di uno a servire le stesse chiamate.

Il rinforzo appena descritto, però, tende a non funzionare nel caso ci sia poco traffico e ben servito dagli ascensori; in questo caso, infatti, gli ascensori tendono a non fermarsi mai, per diminuire la probabilità che durante il loro controllo compaia una nuova chiamata, dato che nella situazione descritta essa costituirebbe la parte predominante della differenza di tempi di attesa tra l'inizio e la fine del controllo a questo livello, portando a un rinforzo basso e spesso negativo. Per questo motivo abbiamo definito un rinforzo alternativo (a cui ci riferiremo con il termine rinforzo personalizzato); esso prevede di assegnare, se la decisione è quella di fermarsi, un bonus proporzionale al tempo di attesa della richiesta a cui si è deciso di fermarsi (con fattore moltiplicativo 20) e una penalità proporzionale al numero di ascensori

fermi per quella richiesta (fattore moltiplicativo 10), agli ascensori diretti a servire quella richiesta (fattore moltiplicativo 2), agli ascensori che devono fermarsi a quel piano a causa delle richieste fatte dalle persone a bordo di essi (fattore moltiplicativo 0,001) e ai tempi di attesa delle due richieste in attesa da più tempo nel sistema (fattore moltiplicativo 0,0001); se invece la decisione prevede di non fermarsi a quel piano, il rinforzo ottenuto è l'opposto di quello che si sarebbe ottenuto con la decisione di fermarsi.

Per ridurre la dimensionalità del problema abbiamo utilizzato la feature selection. Per effettuare la selezione abbiamo creato un dataset di circa 100 mila tuple, generato in condizioni di traffico equamente divise tra uppeak, downpeak e interfloor e controllo effettuato, anche in questo caso con divisione equa, con un controllore random, uno deterministico con buone prestazioni e uno non deterministico con prestazioni accettabili; durante la generazione del dataset abbiamo utilizzato simulazioni da 30 minuti, con intensità del traffico di 1200 persone/ora. Il rinforzo utilizzato per la creazione del dataset è quello di tipo personalizzato.

Inoltre, alle feature di stato ne abbiamo aggiunta anche una che indica la condizione di traffico in corso, in modo da poter stabilire se questa informazione può essere utile per praticare una strategia di controllo migliore.

Le variabili selezionate dal metodo euristico basato su alberi sono le seguenti 16 (15 di stato e 1 di azione)[7]:

per l'ascensore in questione:

1 - piano dove si trova

2 - prossimo piano richiesto dalle persone a bordo

3 - prossimo piano dove potrebbe fermarsi

5 - tempo di attesa della richiesta al prossimo piano dove potrebbe fermarsi

6 - numero di chiamate da piani lungo la direzione di marcia, per andare nella direzione di marcia

7 - numero di chiamate da piani lungo la direzione di marcia, per andare nella direzione opposta a quella di marcia

13 - numero di altri piani dove potrebbe decidere di fermarsi prima del prossimo piano richiesto dalle persone a bordo

14 - numero di richieste a bordo

per la situazione generale del sistema:

18 - numero di altri ascensori diretti al prossimo piano dove potrebbe fermarsi l'ascensore in questione e che continueranno nella direzione di marcia dell'ascensore in questione

19 - numero di altri ascensori fermi al prossimo piano dove potrebbe fermarsi l'ascensore in questione e che continueranno nella direzione di marcia dell'ascensore in questione

[7] Il numero che precede ogni feature è l'indice (base 1) della feature nell'elenco completo; esso è comodo per disegnare l'albero delle dipendenze estratte dalla selezione, dato che l'elevato numero di feature selezionate non renderebbe leggibile uno schema che contiene la spiegazione completa della feature, come quello fatto per l'architettura piatta.

24 - piano in cui c'è la chiamata col tempo di attesa secondo maggiore
27 - piano in cui c'è la chiamata col tempo di attesa maggiore
29 - numero di richieste di ascensori, direzione qualsiasi
31 - numero di richieste di ascensori, direzione alto
32 - tipologia di traffico
33 - azione

La Figura 5.4 riporta l'albero delle dipendenze generato dalla selezione delle feature. Le dipendenze del rinforzo sono quelle ovvie: per spiegarlo, infatti, sono state selezionate le variabili di cui il rinforzo è combinazione lineare (escludendo quelle con pesi non significativi) e l'azione, in quanto determina il segno dei pesi della combinazione lineare. Potrebbe sembrare strano, invece, che alcune variabili, come quelle che indicano gli ascensori fermi e diretti al piano a cui bisogna decidere se fermarsi, non siano autoregressive; questo, però, è plausibile, in quanto se un ascensore arriva o supera quel piano, oppure decide di non fermarsi, al passo successivo le variabili si riferiscono a un piano differente, nello specifico il prossimo a cui è necessario decidere se fermarsi. Ancora più strana potrebbe sembrare la non autoregressività della variabile che indica la tipologia di traffico in corso. Questo fatto è spiegabile dalla episodicità del problema a questo livello e dalla breve durata di ciascun episodio; infatti, quando si finisce nello stato finale, abbiamo deciso di porre questa variabile a un valore diverso da quelli utilizzati per indicare la particolare tipologia di traffico e, dato che questo tipo di transizione del valore avviene spesso a causa della breve durata degli episodi, la variabile non risulta essere autoregressiva.

Livello con ascensori vuoti. Il livello con ascensori vuoti è il livello più alto della gerarchia e prende le decisioni strategiche relative al comportamento degli ascensori con nessuna persona a bordo.

A questo livello praticamente ogni scelta è valida. Infatti, essendo l'ascensore vuoto, il suo comportamento non è vincolato dalle regole di Closs. L'unico vincolo è posto dai limiti del sistema simulato, in cui agli ascensori non è permesso invertire direzione di marcia mentre sono in movimento. Nonostante questo, abbiamo deciso di permettere agli agenti che apprendono di proporre un'azione che farebbe invertire il senso di marcia; in questo caso, l'azione che effettivamente imponiamo all'Ambiente prevede che l'ascensore si fermi al primo piano possibile dichiarando che continuerà nella direzione opposta rispetto a quella in cui sta viaggiando. Questa scelta non modifica l'insieme di politiche che gli agenti potrebbe apprendere introducendo il suddetto vincolo, ma semplicemente ne semplifica l'apprendimento; infatti, una scelta di questo tipo manifesta l'intenzione di invertire la direzione di marcia che, nella pratica, è ottenibile solo col comportamento che abbiamo imposto.

Le feature di stato che abbiamo reso disponibili agli agenti per questo compito sono le seguenti:

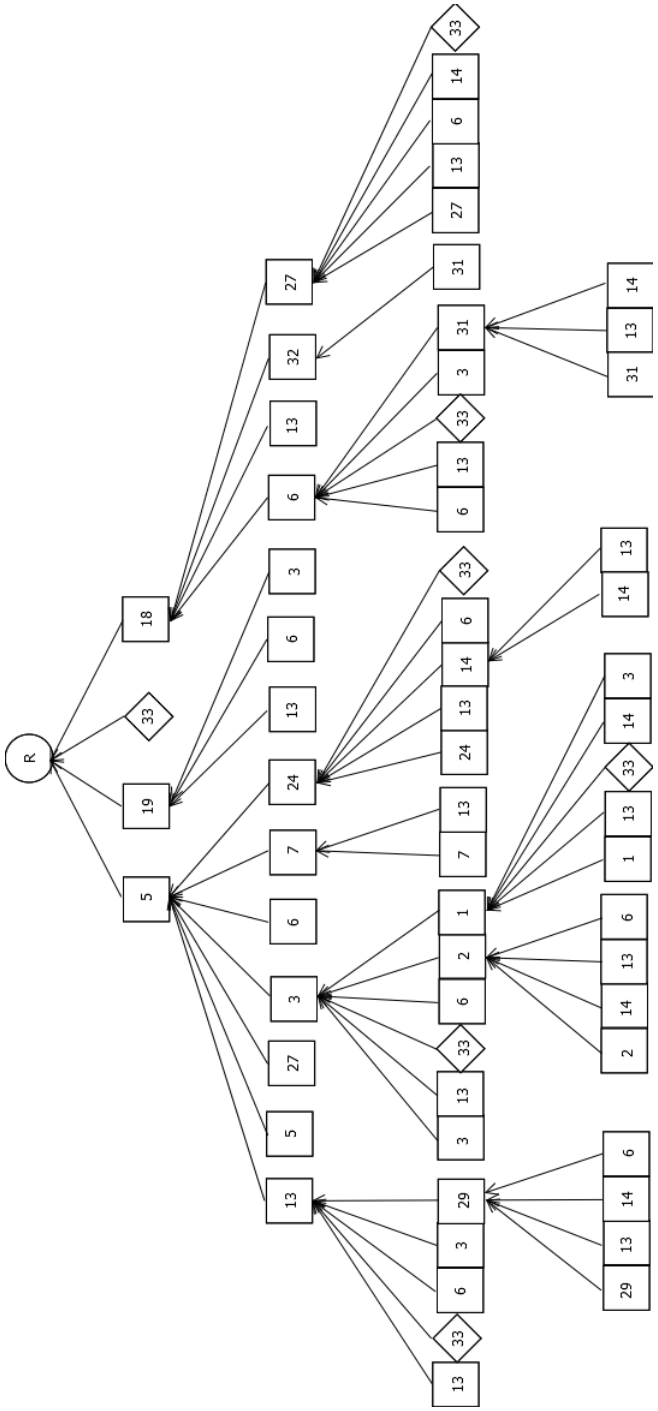


Figura 5.4: Feature selezionate dal metodo basato su alberi al livello con ascensori pieni dell'architettura gerarchica. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte.

per l'ascensore in questione:

- piano dove si trova
- direzione in cui sta viaggiando (-1:DOWN, 0:FERMO, +1:UP)
- piano di destinazione dell'ascensore (-1 se non esiste)
- direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN, 0:FERMO, +1:UP)
- ultimo piano con una richiesta nella direzione corrente
- ultimo piano con una richiesta nella direzione opposta
- ultimo piano, prima del corrente, con una richiesta nella direzione corrente
- è al piano inferiore? (1/0)
- è al piano superiore? (1/0)
- è al piano col passeggero in attesa da più tempo? (1/0)
- numero di chiamate da piani superiori rispetto alla posizione corrente, direzione alto
- numero di chiamate da piani superiori rispetto alla posizione corrente, direzione basso
- numero di chiamate da piani superiori rispetto alla posizione corrente, qualsiasi direzione
- numero di chiamate da piani inferiori rispetto alla posizione corrente, direzione alto
- numero di chiamate da piani inferiori rispetto alla posizione corrente, direzione basso
- numero di chiamate da piani inferiori rispetto alla posizione corrente, qualsiasi direzione
- distanza, in numero di piani, tra questo ascensore e quello più vicino
- distanza, in numero di piani, tra questo ascensore e il piano dove è emersa una nuova chiamata, considerando anche la direzione di marcia dell'ascensore, o -1 se la richiesta non è per una nuova chiamata

per ogni piano ripeto:

- tempo di pressione del bottone up in millisecondi (-1 se non premuto)
- tempo di pressione del bottone down in millisecondi (-1 se non premuto)
- numero totale di posti disponibili negli ascensori al piano che viaggiano verso l'alto (o senza direzione)
- numero totale di posti disponibili negli ascensori al piano che viaggiano verso il basso (o senza direzione)
- numero di ascensori diretti a questo piano e che continueranno verso l'alto (o senza direzione)
- numero di ascensori diretti a questo piano e che continueranno verso il basso (o senza direzione)

per la situazione generale del sistema:

- direzione della chiamata col tempo di attesa secondo maggiore (-1: basso; 1: alto)
- piano in cui c'è la chiamata col tempo di attesa secondo maggiore
- direzione della chiamata col tempo di attesa maggiore (-1: basso; 1: alto)
- piano in cui c'è la chiamata col tempo di attesa maggiore
- numero di richieste di ascensori, direzione qualsiasi
- numero di richieste di ascensori, direzione basso

numero di richieste di ascensori, direzione alto

Si tratta, quindi, di 25+6*numeroPiani feature che dipendono ovviamente dall'ascensore di cui è richiesta la decisione e dallo stato generale del sistema, ma sono spersonalizzate rispetto all'identità degli altri ascensori; questo, ovviamente, non crea problemi, dato che, tranne in casi di servizi speciali, il comportamento di tutti gli ascensori è equivalente, e, nello stesso tempo, semplifica l'apprendimento, dato che l'agente non deve imparare una politica equivalente basata sullo stato di ogni altro ascensore. A differenza del livello con ascensori pieni, invece, abbiamo deciso di mantenere informazioni personalizzate per ogni piano, in quanto la decisione da prendere a questo livello è più generale e, quindi, richiede una visione completa dello stato del sistema.

Le azioni tra cui gli agenti possono scegliere sono $a_i\,, 1 \leq i \leq 2(numPiani-1)+1$, definite come:

$$a_i = \begin{cases} < i+1, -1 > & i \leq numPiani - 1 \\ < i - numPiani + 1, +1 > & numPiano \leq i \leq 2(numPiani - 1) \\ < -1, 0 > & i = 2(numPiani - 1) + 1 \end{cases}$$

Come si può notare, rispetto alle azioni disponibili con l'architettura piatta abbiamo eliminato la possibilità che un ascensore, una volta arrivato al piano a cui è diretto, dichiari che rimarrà fermo. Questa scelta deriva dal fatto che il simulatore, per come è strutturato, dichiara comunque una direzione di continuazione del viaggio, quindi non ha senso che l'agente possa apprendere un'azione che nella pratica non esiste nel sistema. Nel caso in cui invece all'ascensore non è assegnata nessuna destinazione la direzione successiva non è significativa e, quindi, abbiamo deciso di metterla arbitrariamente a nessuna direzione.

L'azione scelta dagli agenti normalmente viene passata così come è all'Ambiente; ci sono però due casi in cui essa viene modificata. Un caso è, come già accennato, quando questa azione imporrebbe un cambiamento di direzione a un ascensore in movimento: l'azione, quindi, si tramuta mettendo come piano di destinazione il primo piano a cui l'ascensore può fisicamente fermarsi e come prossima direzione quella opposta rispetto a quella in cui sta viaggiando. L'altro caso si ha quando il controllo è appena stato restituito dal livello con persone a bordo; in questo caso l'ascensore deve comunque arrivare alla destinazione richiesta dalle restanti persone a bordo, che quindi risulta essere il piano di destinazione dell'ascensore, mentre la direzione in cui continuerà il viaggio dipende dall'azione scelta a questo livello: essa dipende dalla posizione del piano selezionato rispetto a quello a cui l'ascensore è effettivamente diretto o, se questi due piani coincidono, è la direzione selezionata.

Il rinforzo assegnato a ogni passo è, come al solito, l'opposto del quadrato del tempo di attesa medio integrale, calcolato in secondi e scalato di un fattore 10^5 per evitare problemi numerici; inoltre, quando il controllo non è a questo livello, il

rinforzo che sarebbe ottenuto viene accumulato e aggiunto a quello del primo passo in cui il controllo torna a questo livello, in modo da avere una visione generale di tutto quello che è accaduto nel sistema, anche in seguito a una decisione che ha portato a far salire persone sugli ascensori e quindi a far passare il controllo a un livello inferiore.

Anche in questo caso abbiamo utilizzato la feature selection per ridurre la dimensionalità del problema. Per questo scopo abbiamo costruito un dataset equivalente a quello costruito per il livello con ascensori pieni, aggiungendo anche in questo caso una feature che indica la tipologia di traffico.

Le variabili selezionate dal metodo euristico basato su alberi sono le seguenti 18 (16 di stato e 2 di azione):

per l'ascensore in questione:

1 - piano dove si trova

2 - direzione in cui sta viaggiando (-1:DOWN, 0:FERMO, +1:UP)

3 - piano di destinazione dell'ascensore (-1 se non esiste)

4 - direzione di viaggio dell'ascensore una volta raggiunto il piano a cui è diretto (-1:DOWN, 0:FERMO, +1:UP)

7 - ultimo piano, prima del corrente, con una richiesta nella direzione corrente

12 - numero di chiamate da piani superiori rispetto alla posizione corrente, direzione basso

15 - numero di chiamate da piani inferiori rispetto alla posizione corrente, direzione basso

per la situazione generale del sistema:

21 - numero totale di posti disponibili negli ascensori al piano 1 che viaggiano verso l'alto (o senza direzione)

22 - numero totale di posti disponibili negli ascensori al piano 1 che viaggiano verso il basso (o senza direzione)

23 - numero di ascensori diretti al piano 1 e che continueranno verso l'alto (o senza direzione)

27 - numero totale di posti disponibili negli ascensori al piano 2 che viaggiano verso l'alto (o senza direzione)

33 - numero totale di posti disponibili negli ascensori al piano 3 che viaggiano verso l'alto (o senza direzione)

34 - numero totale di posti disponibili negli ascensori al piano 3 che viaggiano verso il basso (o senza direzione)

74 - tempo di pressione del bottone down al piano 10 in millisecondi (-1 se non premuto)

83 - numero di richieste di ascensori, direzione qualsiasi

86 - tipologia di traffico

87 - azione piano di destinazione

88 - azione direzione una volta arrivato al piano di destinazione

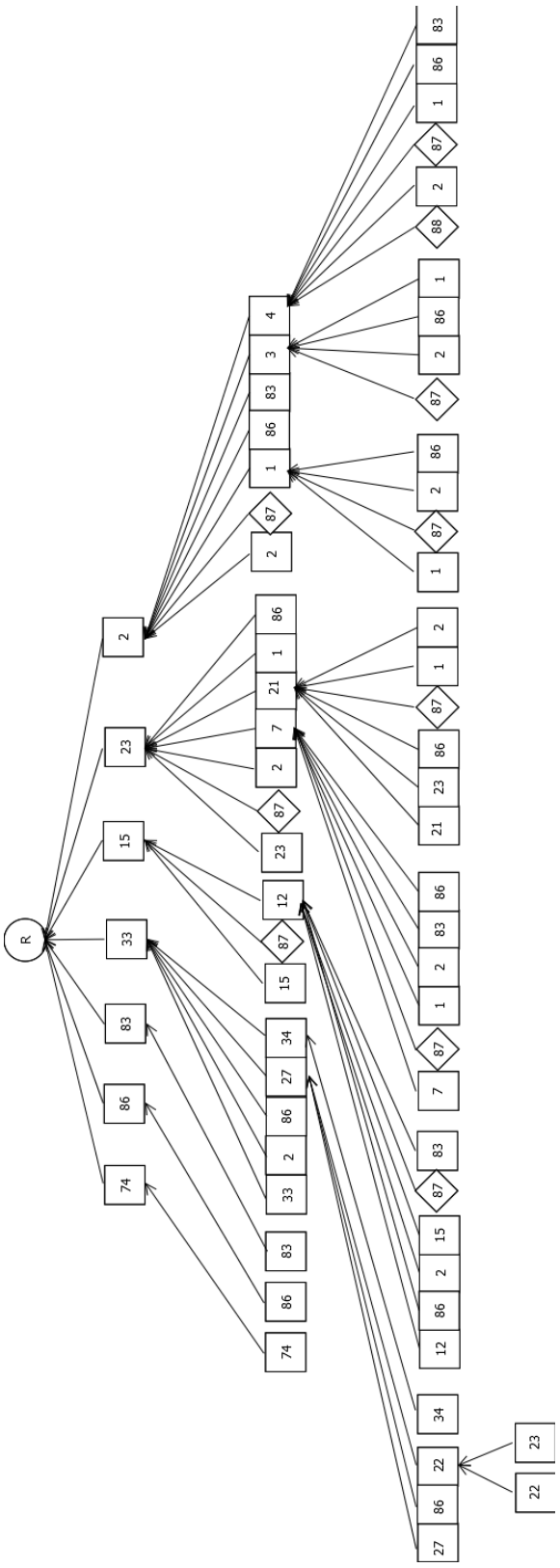


Figura 5.5: Feature selezionate dal metodo basato su alberi al livello con ascensori vuoti dell'architettura gerarchica. L'ordinamento da sinistra a destra riflette l'ordine di selezione delle variabili, mentre le frecce indicano le dipendenze che hanno indotto a selezionare una variabile; dato che una feature può essere selezionata per spiegare più di una variabile, è ammesso avere nel diagramma una feature ripetuta più volte.

La Figura 5.5 riporta l'albero delle dipendenze delle feature generato dall'algoritmo di selezione. Da esso emerge che due variabili sono fondamentali per spiegare un numero elevato di altre feature; in particolare, esse sono la variabile che indica la tipologia di traffico e quella dell'azione che decide il piano di destinazione. Risulta confermato, quindi, che l'azione che si sceglie è fondamentale per ottenere delle buone prestazioni (e la presenza nell'albero di anche l'altra variabile di azione indica che tutte le componenti della decisione prendibile sono importanti) e anche che un controllore che è in grado di capire la tipologia di traffico in corso può applicare una strategia più efficace rispetto a uno che non utilizza questo tipo di informazione.

Rispetto al livello con ascensori pieni la variabile che indica la tipologia di traffico è autoregressiva e non dipende da nessun'altra variabile; in questo caso ha senso che sia così, in quanto non esiste uno stato terminale e il cambiamento di schema di traffico avviene molto raramente[8].

Un'altro fatto evidente è che sono state selezionate poche variabili che indicano la situazione particolare ai vari piani. Questo fatto probabilmente indica che a questo livello è più importante la strategia generale degli ascensori, che li coordina e indica, in base a informazioni riassuntive, in che zona dell'edificio è utile che si rechi ogni ascensore; le chiamate specifiche possono poi essere servite in modo adeguato (anche se ovviamente non ottimo) dal livello inferiore una volta che è salita almeno una persona all'interno dell'ascensore. Le uniche variabili specifiche inserite sono quella che indica il tempo di attesa all'ultimo piano, probabilmente utile per evitare che un ascensore vi si rechi, dato che rappresenta comunque il limite massimo non superabile, e alcune che indicano quanti ascensori sono presenti o diretti ai piani bassi, probabilmente per evitare che tutti gli ascensori si portino a questi piani, lasciando completamente scoperto il resto dell'edificio.

Parte pre-programmata. La parte pre-programmata del controllore RLG gerarchico ha, essenzialmente, due compiti: gestire e coordinare i due livelli decisionali, trasformando anche le loro scelte in azioni comprensibili dall'Ambiente; selezionare le azioni da compiere quando queste non richiedono alcun tipo di decisione.

Il primo dei due compiti è già stato sostanzialmente spiegato nella trattazione dei due livelli decisionali; infatti, i momenti di inizio e fine del controllo a un certo livello, così come le trasformazioni necessarie delle azioni, il calcolo dei rinforzi non immediatamente uguali a quanto ricevuto dall'ambiente sono tutti compiti assegnati alla parte pre-programmata.

Risulta, invece, più interessante analizzare le decisioni obbligate prese dalla parte pre-programmata. Essa, ad esempio, si occupa di selezionare l'azione quando un

[8] In realtà nel dataset che abbiamo creato non è stato inserita nessuna transizione che riproduce il cambiamento di tipologia di traffico, ma i risultati ottenuti sono da considerarsi comunque validi, in quanto il numero di queste transizioni risulta non essere significativo rispetto al numero totale di transizioni che si hanno in una tipica giornata lavorativa (si veda a tal proposito l'andamento del traffico in una tipica giornata lavorativa, riportato in Figura 2.1).

ascensore è totalmente pieno. In questo caso, infatti, non si ha la possibilità di fermarsi a un piano intermedio, in quanto comunque non potrebbe salire nessuno a bordo dell'ascensore, che quindi è vincolato a dirigersi al primo piano richiesto dalle persone già a bordo; di conseguenza, non ha alcun senso interpellare l'agente del livello con ascensori pieni, ma l'azione viene impostata direttamente dalla parte pre-programmata.

Un altro caso in cui non è richiesta alcuna decisione si ha quando un ascensore è fermo a un piano con le porte aperte, infatti qualsiasi azione presa in questa situazione non ha nessun effetto, perché la direzione di viaggio in cui l'ascensore continuerà il viaggio è già stata mostrata agli utenti e quindi non è modificabile, mentre la prossima destinazione verrà comunque modificata appena l'ascensore chiude le porte, in seguito all'evento relativo. In questa situazione, quindi, la parte pre-programmata può semplicemente selezionare un'azione qualsiasi, che tanto non ha alcun effetto sul sistema.

5.3 Controllore FQI

Fitted Q Iteration (FQI) è una tecnica di apprendimento per rinforzo il cui obiettivo è quello di approssimare iterativamente i valori Q, utilizzando per questo scopo delle traiettorie di dati raccolti preliminarmente [61]. Essendo il metodo iterativo è necessario fermare l'apprendimento dopo un certo numero di iterazioni, che deve essere sufficientemente elevato per garantire di essere arrivati abbastanza vicini alla convergenza dei valori approssimati con quelli reali; nei nostri esperimenti abbiamo deciso di effettuare 100 iterazioni dell'algoritmo.

La dimensionalità del problema che analizziamo non permette di salvare i valore Q in forma tabulare, ma è necessario utilizzare un approssimatore per essi; come approssimatore abbiamo deciso di utilizzare una foresta di alberi di tipo *extra-trees*, che permettono di catturare dipendenze non lineari e di ottenere un'approssimazione sufficientemente adeguata. Negli esperimenti abbiamo utilizzato una foresta composta da 100 alberi, costruiti con un numero di tagli casuali pari al numero di feature utilizzate e numero minimo di elementi contenuti nelle foglie deciso tramite cross-validazione, in modo da massimizzare il valore R^2 per la spiegazione del rinforzo definito dal metodo di riduzione delle feature basato su alberi.

5.3.1 Architettura piatta

Abbiamo definito un controllore che apprende tramite FQI la politica migliore, con una struttura piatta. Per l'apprendimento di questo controllore abbiamo utilizzato lo stesso dataset utilizzato per la selezione delle feature della struttura piatta. L'apprendimento, inoltre, è uno solo e non specializzato per tipologia di traffico, senza nemmeno introdurre una feature che differenzi tra le varie tipologie di traffico.

	complete	alberi	lasso
uppeak	trend miglioramento, verso 6/6,5 sec	trend oscillante, tra 6 e 8 sec	migliore tra iterazioni 20 e 30, sui 7 sec
downpeak	NO	oscillante (meglio iterazioni 40 e 90-100), sui 35-45 sec	da iterazione 80 in poi, sui 50-60 sec
interfloor	NO	oscillante sui 36 sec	oscillante tra 40/50 sec
giornata	NO	NO, sui 32-35 sec quando ci riesce	NO
misto	NO	oscillante tra 30 e 35 sec	oscillante tra 35 e 45 sec

Tabella 5.5: Risultati qualitativi ottenuti con il controllore FQI con architettura piatta, al variare delle feature utilizzate e del traffico generato nel sistema. È riportato il trend dei risultati al variare del numero di iterazioni dell'algoritmo FQI effettuate. La dicitura NO indica che il controllore non è riuscito a servire i passeggeri presenti nel sistema.

Abbiamo effettuato 100 iterazioni di FQI, provando tutti e tre gli insiemi di feature (completo e con le due riduzioni), salvando la politica appresa ogni 10 iterazioni; i risultati qualitativi dei controllori che applicano queste politiche sono riportate in Tabella 5.5, mentre risultati più dettagliati e precisi non sono significativi dati i risultati non accettabili emersi.

Dai dati emerge che, essenzialmente, non ci sono grandi differenze tra le politiche apprese ogni 10 iterazioni e, anche dove emergono delle differenze, lo fanno senza seguire uno schema; non si può dire, quindi, che la politica migliora all'aumentare delle iterazioni e nemmeno che da una certa iterazione in poi si ha un peggioramento dovuto alle approssimazioni numeriche, come dovrebbe essere in teoria.

Un altro aspetto che emerge dai dati è che le prestazioni ottenute sono migliori al diminuire delle informazioni disponibili, contrariamente a quanto ci si aspetterebbe. Infatti, il controllore con tutte le informazioni e, in parte, quello che utilizza le feature ridotte tramite Lasso fanno fatica a servire, entro i 30 minuti concessi dal simulatore seguenti la durata imposta della simulazione, tutti i passeggeri; solitamente, questo si ha in seguito alla decisione del controllore di mantenere fermi tutti gli ascensori.

I due fenomeni descritti sono spiegabili considerando la quantità di dati contenuti nel dataset paragonata alla quantità delle feature che compongono il problema. Infatti, la quantità di tuple inserite nel dataset è limitata, al fine di avere tempi computazionali per la riduzione delle feature e per il calcolo della politica accettabili; la quantità di tuple presenti per feature, però, è molto limitata e questo impedisce agli agenti che apprendono di esplorare tutto lo spazio di stato.

(a) Tempo di attesa medio

	complete	alberi
uppeak	6,152	5,414
downpeak	23,930	30,474
interfloor	39,59	35,158

(b) Tempo di attesa massimo

	complete	alberi
uppeak	37,533	30,234
downpeak	150,040	225,192
interfloor	263,383	229,199

Tabella 5.6: Tempo di attesa medio e massimo, in secondi, ottenuti in condizioni di traffico leggero con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema.

(a) Tempo di viaggio medio, in secondi

	complete	alberi
uppeak	31,597	44,642
downpeak	25,798	29,752
interfloor	33,576	32,458

(b) Tempo di sistema medio, in secondi

	complete	alberi
uppeak	37,749	50,057
downpeak	49,729	60,226
interfloor	73,166	67,617

(c) Numero totale di fermate degli ascensori

	complete	alberi
uppeak	973,042	988,376
downpeak	793,368	833,824
interfloor	1026,68	1292,636

Tabella 5.7: Risultati ottenuti in condizioni di traffico leggero con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema.

5.3.2 Architettura gerarchica

Analizziamo ora i risultati ottenuti dal controllore con architettura gerarchica che utilizza FQI per l'apprendimento. I dataset che abbiamo utilizzato per l'apprendimento sono, per entrambi i livelli della gerarchia, quelli utilizzati per la rispettiva riduzione delle feature, adeguatamente tagliati nel caso di apprendimento con l'insieme ridotto di feature.

Il controllore risultante da questo apprendimento è unico, adibito al controllo in tutte le condizioni di traffico. Tuttavia esso può differenziare la strategia da adottare a seconda della tipologia di traffico; nei dataset utilizzati per l'apprendimento, infatti, è presente una variabile che indica la tipologia di traffico e FQI con alberi è abbastanza potente da poter apprendere strategie differenti a seconda del valore di una variabile.

In Tabella 5.6 abbiamo riportato i tempi di attesa medio e massimo ottenuti con il controllore in questione al variare dell'insieme di feature utilizzate in condizioni di

(a) Tempo di attesa medio

	complete	alberi
uppeak	203,809	53,393
downpeak	36,022	46,711
interfloor	59,072	55,022

(b) Tempo di attesa massimo

	complete	alberi
uppeak	644,237	134,237
downpeak	259,094	316,240
interfloor	252,841	222,084

Tabella 5.8: Tempo di attesa medio e massimo, in secondi, ottenuti in condizioni di traffico intenso con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema.

(a) Tempo di viaggio medio, in secondi

	complete	alberi
uppeak	53,95	57,672
downpeak	32,938	37,727
interfloor	52,916	53,377

(b) Tempo di sistema medio, in secondi

	complete	alberi
uppeak	257,759	111,066
downpeak	68,961	84,438
interfloor	111,988	108,399

(c) Numero totale di fermate degli ascensori

	complete	alberi
uppeak	848,102	947,762
downpeak	560,894	661,468
interfloor	947,448	1294,782

Tabella 5.9: Risultati ottenuti in condizioni di traffico intenso con il controllore FQI con architettura gerarchica, al variare delle feature utilizzate e della tipologia di traffico generato nel sistema.

traffico leggero, mentre in Tabella 5.7 abbiamo riportato le altre quantità significative con la medesima configurazione; similmente, le Tabelle 5.8 e 5.9 riportano i risultati ottenuti dal controllore considerato, al variare dell'insieme di feature utilizzate, in condizioni di traffico intenso. Tutti i dati numerici sono stati ottenuto mediando i risultati di 500 simulazioni da 60 minuti l'una.

Confrontando i risultati ottenuti dal controllore FQI gerarchico con quelli ottenuti dai controllori benchmark notiamo che, in generale, le prestazioni del nostro controllore rimangono, in qualsiasi situazione di traffico, inferiori a quelle di almeno un controllore benchmark; solamente nel caso di uppeak leggero le sue prestazioni risultano essere abbastanza vicine al migliore di essi. Inoltre, anche considerando le prestazioni di un singolo controllore in tutte le condizioni di traffico quello da noi proposto risulta non essere il migliore, in quanto quello fuzzy e quello genetico riescono a ottenere prestazioni generalmente migliori sia con traffico leggero, sia con traffico intenso.

Confrontando invece le prestazioni ottenute dal nostro controllore con i due diversi insieme di feature possiamo notare che in uppeak e interfloor si comporta meglio quello che utilizza l'insieme ridotto di feature, mentre in downpeak le prestazioni del controllore che utilizza l'insieme completo di feature sono migliori.

Analizzando il comportamento qualitativo delle politiche apprese possiamo notare che non ci sono molte differenze tra quelle apprese con tutte le informazioni disponibili e quelle apprese solo con l'insieme ridotto di variabili. In particolare, gli ascensori liberi vengono solitamente mandati ai piani ritenuti più utili a seconda della tipologia di traffico in corso (il primo in uppeak, l'ultimo o uno degli ultimi in downpeak), indipendentemente dalla presenza di chiamate o altri ascensori a quei piani; questo, quindi, porta a un deterioramento delle prestazioni, sia per fermate non necessarie, sia per un coordinamento non perfetto tra gli ascensori. Sebbene questo comportamento fosse abbastanza prevedibile nel caso di utilizzo delle feature ridotte, dato che non contengono le informazioni necessarie per evitare i problemi descritti, è più strano nel caso di utilizzo delle feature complete, dato che esse contengono tutte le informazioni necessarie.

Un'altro comportamento che avviene in tutti i casi emerge alla fine delle simulazioni; a volte, infatti, capita che gli ascensori si fermino completamente e non servono le ultime persone arrivate nel sistema. Questo comportamento è spiegabile dal fatto che, probabilmente, nel dataset non è compreso un insieme significativo di situazioni terminali e, quindi, l'agente non ha imparato una strategia da attuare in alcune di queste situazioni terminali. Quello appena descritto, comunque, non costituisce un problema vero e proprio; in uno scenario reale, infatti, non esiste una situazione terminale, dopo la quale non si hanno più eventi, ma si hanno semplicemente delle transizioni da una tipologia di traffico a un'altra.

A questo punto non risulta ancora chiaro il motivo della differenza di prestazioni riscontrato tra il controllore appreso con l'insieme completo delle feature e quello

appreso con l'insieme ridotto. Anche questa particolarità è spiegabile con le informazioni contenute nel dataset. Infatti, osservando il comportamento degli ascensori che seguono la politica appresa con tutte le feature, si può notare come essi, soprattutto in condizioni di traffico intenso, a volte si fermano per un certo intervallo di tempo prima di ricominciare a servire le chiamate; questo comportamento probabilmente è dovuto alla presenza di situazioni mai viste, che creano problemi alle decisioni degli agenti fino a quando non si ritorna in uno stato abbastanza noto. Ovviamente il problema emerge maggiormente se si utilizzano tutte le feature, perché con esse, essendo molte di più, si hanno molte più possibilità di riscontrare valori non noti.

Da questa analisi emerge, quindi, che il dataset creato non è abbastanza informativo per la soluzione del problema attraverso FQI; purtroppo, però, non possiamo aggiungere un numero eccessivo di traiettorie ad esso per inserire tutte le situazioni possibili, in quanto le sue dimensioni aumenterebbero a tal punto da rendere i tempi computazionali per l'apprendimento eccessivamente lunghi.

5.4 Controllore RLG

I metodi gradiente (RLG) sono dei metodi che, data una politica parametrizzata, iterativamente modificano i parametri di questa politica, seguendo appunto la direzione indicata dal gradiente, in modo da massimizzare il rinforzo scontato ottenuto; esistono diversi metodi in questa famiglia, sia basati sul gradiente classico, come REINFORCE e GPOMDP (con o senza baseline), sia basati sul gradiente naturale, come NAC [48].

La politica parametrizzata che abbiamo deciso di utilizzare per l'apprendimento e il controllo del gruppo di ascensori è la politica Gibbs con ultima azione indipendente; indicando con $a \in A$ una generica azione e $\overline{a} = a_m$, dove m è la cardinalità di |A|, l'ultima azione, questa politica è definita come:

$$\pi(s,a) = \frac{e^{\tau\phi(s,a)^T w}}{1 + \sum_{a' \in A \setminus \{\overline{a}\}} e^{\tau\phi(s,a')^T w}} \qquad \forall a \neq \overline{a}$$

$$\pi(s,\overline{a}) = \frac{1}{1 + \sum_{a' \in A \setminus \{\overline{a}\}} e^{\tau\phi(s,a')^T w}}$$

dove τ è un parametro, chiamato temperatura, che regola l'equiprobabilità delle azioni ed è posto a 1 nei nostri esperimenti; $\phi(s,a)$ è un insieme di basis; w sono i parametri che caratterizzano la politica.

Nei nostri esperimenti abbiamo definito basis lineari e parametri indipendenti per ogni azione, in modo che tutti i parametri possano variare in dipendenza della loro azione. Nell'implementazione pratica, quindi, indicando con n_f il numero di feature di stato e con a_j, $1 \leq j \leq m$ l'azione j-esima, si ha che, $\forall s \in S$ e $\forall a_j \in A \setminus \{\overline{a}\}$:

$$\phi(s,a_j) \in \mathbb{R}^{((n_f+1)\cdot(m-1))\times 1}$$

$$\phi(s,a_j)_i = \begin{cases} s_{i-(n_f+1)(j-1)} & (n_f+1)(j-1)+1 \leq i \leq (n_f+1)j-1 \\ 1 & i=(n_f+1)j \\ 0 & altrimenti \end{cases}$$

dove $\phi(s,a)_i$ indica l'elemento i-esimo di $\phi(s,a)$ e s_k indica la feature k-esima dello stato s.

Essendo la politica parametrica è anche necessario definire il valore iniziale dei parametri che costituiscono la prima politica utilizzata durante l'apprendimento; nei nostri esperimenti abbiamo deciso di inizializzarli tutti a zero, che corrispondono a una politica random.

5.4.1 Architettura piatta

Per il controllore RLG che utilizza l'architettura piatta abbiamo utilizzato principalmente l'insieme completo di feature definito per questa architettura, ma abbiamo fatto anche alcune prove con gli insiemi ridotti; inoltre, abbiamo anche deciso di scalare le feature temporali in secondi/100, in modo da evitare problemi numerici.

Rispetto alla versione base dell'architettura piatta, inoltre, abbiamo escluso a priori le 4 azioni mai valide: $< -1,-1 >$,$< -1,1 >$,$< 1,-1 >$,$< numPiani,1 >$[9]. Per soddisfare la necessità di escludere le azioni non valide abbiamo deciso semplicemente di ridistribuire la probabilità calcolata dalla politica Gibbs sulle sole azioni valide.

Le prove di apprendimento che abbiamo effettuato prevedono, in generale, simulazioni da 500 passi e aggiornamenti della politica ogni 1000 episodi se il metodo di stima del gradiente utilizzato è GPOMDP o REINFORCE, ogni 100 episodi se è NAC; comunque, per essere sicuri che i risultati ottenuti non siano dovuti a mancanza di quantità sufficiente di dati, abbiamo fatto delle prove anche con un numero maggiore di dati, ottenuti sia aumentando il numero delle simulazioni effettuate prima di aggiornare i parametri della politica, sia la lunghezza delle simulazioni. Come fattore di apprendimento abbiamo provato a utilizzare diversi valori, compresi tra 1 e 10^{-4}.

I risultati ottenuti sono indipendenti da tutti i parametri in gioco; al variare della quantità di informazioni disponibili, del fattore di apprendimento, del fattore di sconto e dell'insieme di feature utilizzato, non si riscontra nessun tipo di apprendimento. In particolare, in tutte le tipologie di traffico le prestazioni continuano a

[9] Abbiamo rappresentato le azioni con tuple $< a_1, a_2 >$, in cui a_1 è il prossimo piano a cui è diretto l'ascensore, mentre a_2 è la direzione in cui proseguirà l'ascensore una volta fermatosi al piano a cui è diretto.
La non validità delle prime due azioni escluse è spiegabile dal fatto che, se un ascensore non è diretto a nessun piano, non può proseguire il suo viaggio dopo essere arrivato a quel piano; le altre due azioni, invece, sono escluse perché una volta arrivato al piano terra (indicato con 1) non può proseguire verso il basso e, similmente, una volta arrivato all'ultimo piano (indicato con $numPiani$) non può proseguire verso l'alto.

rimanere, anche in seguito ai vari aggiornamenti della politica, paragonabili a quelle ottenute con la politica iniziale, e cioè non accettabili, dato che inizialmente la politica utilizzata è casuale.

La mancanza di apprendimento è spiegabile con la difficoltà del problema. Infatti, l'agente deve decidere su uno spazio di politiche enorme, la maggior parte del quale tra l'altro spesso non è nemmeno ammissibile; di conseguenza, può facilmente capitare che l'azione che l'agente sceglierebbe non è valida e, quindi, ne viene utilizzata un'altra che esso riterrebbe peggiore. In uno scenario così vasto e complicato, quindi, per gli agenti risulta impossibile apprendere una politica che sia adeguata; pertanto, risulta necessario aiutare gli agenti che apprendono semplificando loro il problema e, possibilmente, evitando di lasciare loro uno spazio delle politiche più ampio di quello da cui, di volta in volta, viene effettivamente scelta l'azione da compiere.

5.4.2 Architettura gerarchica

Il controllore RLG con architettura gerarchica segue esattamente la descrizione generale di questa architettura, che è stata studiata appositamente in modo da superare i problemi riscontrati con l'architettura piatta.

Nell'implementazione pratica, è necessario aggiornare la stessa politica utilizzando i dati dei diversi ascensori; in più, al livello con ascensori pieni c'è la complicazione che gli episodi dei diversi ascensori iniziano e terminano in istanti diversi, in quanto questi istanti sono governati dal riempimento e dallo svuotamento degli ascensori, che ovviamente non sono controllabili. Per questo motivo abbiamo deciso, al livello con ascensori pieni, di accumulare separatamente le traiettorie compiute dai diversi agenti e, al momento di aggiornare la politica, creare un insieme unico di traiettorie composto da tutte quelle viste dai diversi agenti, come se fossero di un unico agente. L'aggiornamento della politica viene fatto dopo un certo numero di episodi, che negli esperimenti abbiamo solitamente messo a 15.

Una situazione simile si ha anche al livello con ascensori vuoti; dopo un certo numero di episodi simulati, avviene l'aggiornamento della politica con i dati raccolti da tutti gli agenti corrispondenti ai diversi ascensori. Questo, ove il metodo di stima del gradiente lo permette in quanto comporta solo dei calcoli lineari, come GPOMDP, viene fatto tramite una media aritmetica degli aggiornamenti proposti dai diversi agenti[10]; in questo modo non è necessario occupare inutilmente della memoria per l'accumulo delle traiettorie, in quanto ogni agente può aggiornare online la sua stima. Nei casi in cui, al contrario, il metodo di stima del gradiente non permette questo approccio, come GPOMDP con baseline, vengono accumulate le traiettorie seguite dai diversi agenti e infine queste traiettorie vengono messe in un unico insieme per

[10]Questo approccio è corretto perché la quantità di dati visti dai diversi agenti è paragonabile; un aggiornamento ancora più preciso, ma anche più complicato da effettuare, prevederebbe di fare una media dei vari gradienti pesata con una misura dell'incertezza legata a questa stima.

il calcolo del gradiente, in modo equivalente a quanto fatto al livello con ascensori pieni.

Risultati con insieme completo delle feature

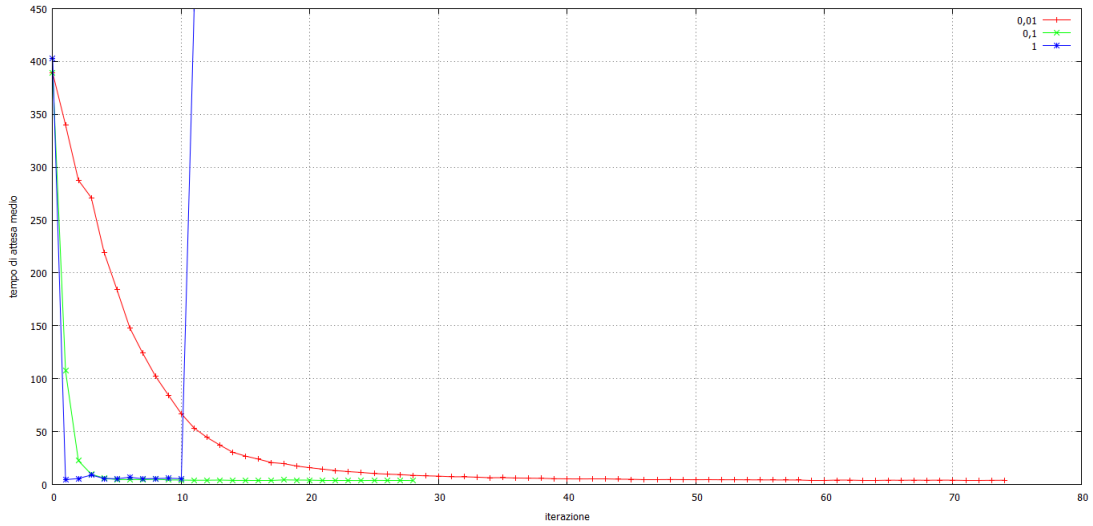
Abbiamo provato a far apprendere agli agenti un controllore utilizzando l'insieme completo delle feature progettate per l'architettura gerarchica e, al livello con ascensori pieni, il rinforzo generale. Per permettere al controllore RLG gerarchico di apprendere una buona politica per il controllo del gruppo di ascensori, abbiamo effettuato simulazioni da 15 minuti con aggiornamento dei parametri della politica ogni 450 simulazioni al livello gerarchico superiore, ogni 15 simulazioni al livello gerarchico inferiore; il numero di run effettuati è variabile e dipende dalla mancanza di un cambiamento apprezzabile nelle prestazioni del controllore, anche in seguito a degli aggiornamenti della politica[11]. Tutti gli aggiornamenti dei parametri sono stati fatti con normalizzazione del gradiente. Il metodo di stima del gradiente utilizzato a entrambi i livelli è GPOMDP, utilizzando un fattore di apprendimento α pari a 0,01 al livello inferiore, mentre al livello superiore lo abbiamo fatto variare per controllarne gli effetti sull'apprendimento. Per scontare il rinforzo ottenuto abbiamo utilizzato $\beta = 10^{-4}$ nel calcolo del tempo di attesa medio quadratico e, quindi, nell'algoritmo di stima tramite gradiente abbiamo lasciato $\gamma = 1$.

Le prove di apprendimento effettuate sono separate per tipologia di traffico e, quindi, i controllori appresi sono più di uno, ognuno specializzato per una singola tipologia di traffico.

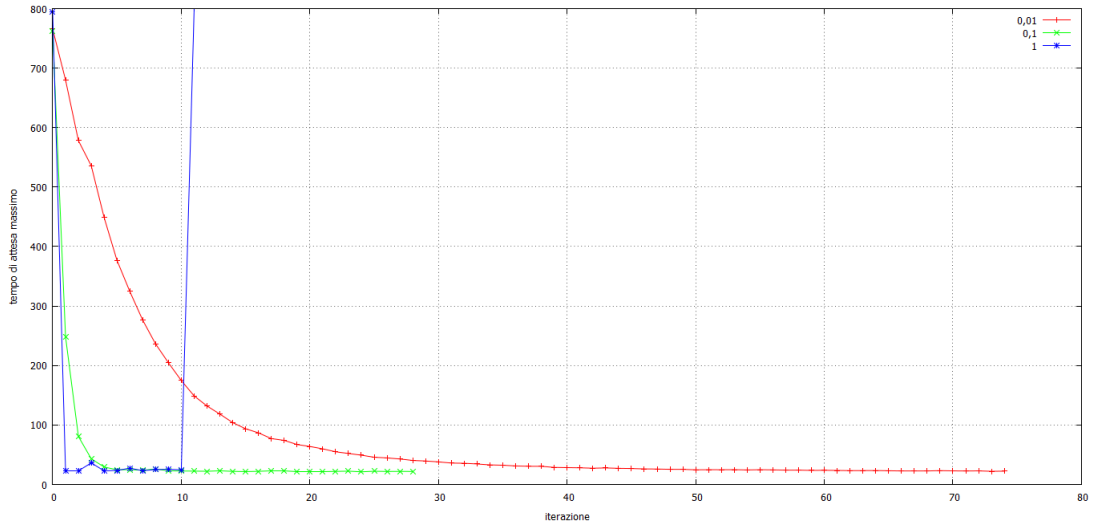
Traffico leggero. Le Figure 5.6, 5.7 e 5.8 riportano le curve di apprendimento della politica al livello con ascensori vuoti, mostrando come variano il tempo di attesa medio, il tempo di attesa massimo e il rinforzo ottenuto dagli agenti a ogni run, con i vari valori di α utilizzati; similmente, le Figure 5.9 e 5.10 riportano le curve di apprendimento della politica al livello con ascensori pieni, mostrando come variano il rinforzo assegnato agli agenti a ogni run e con che rapporto medio viene preferita una delle due azioni possibili rispetto all'altra. Per il livello più basso non abbiamo riportato i risultati ottenuti in uppeak, semplicemente perché in questo livello non è mai richiesta nessuna decisione a questo livello; infatti, tutte le richieste ai piani sono al piano terra e, quindi, non si presenta mai la possibilità di fermarsi a un piano intermedio per far salire altre persone sull'ascensore.

Dai dati riportati emerge chiaramente che l'assegnamento α=1 al livello superiore non è adeguato, in quanto porta a un netto oscillamento delle prestazioni, che non raggiungono nemmeno i livelli ottenuti dalle altre due scelte; inoltre, con questo assegnamento si rischia di avere problemi numerici, come è accaduto con traffico

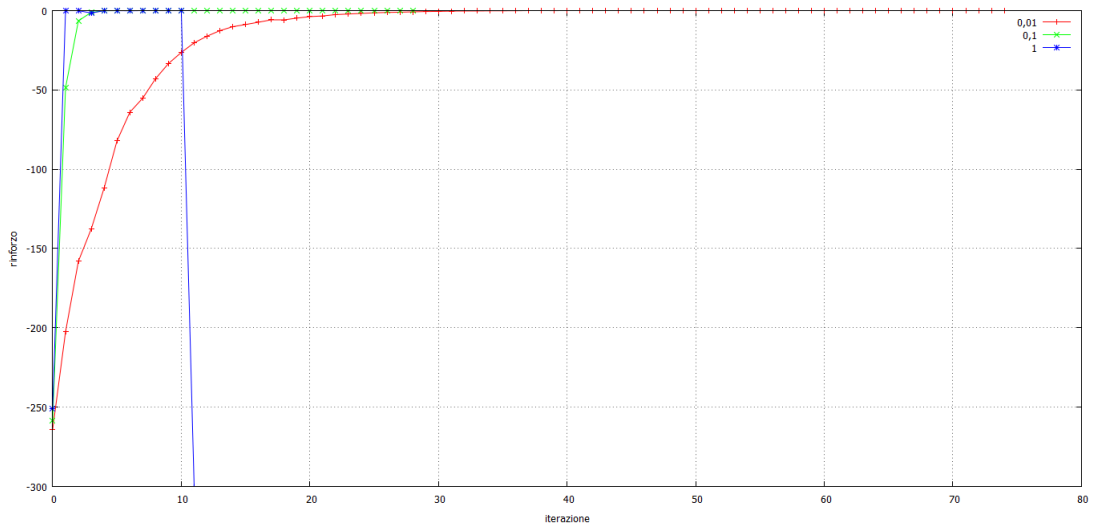
[11] Per questo motivo nei grafici successivi che riportano l'andamento dell'apprendimento il numero di iterazioni varia da caso a caso.



(a) Tempo di attesa medio (secondi)

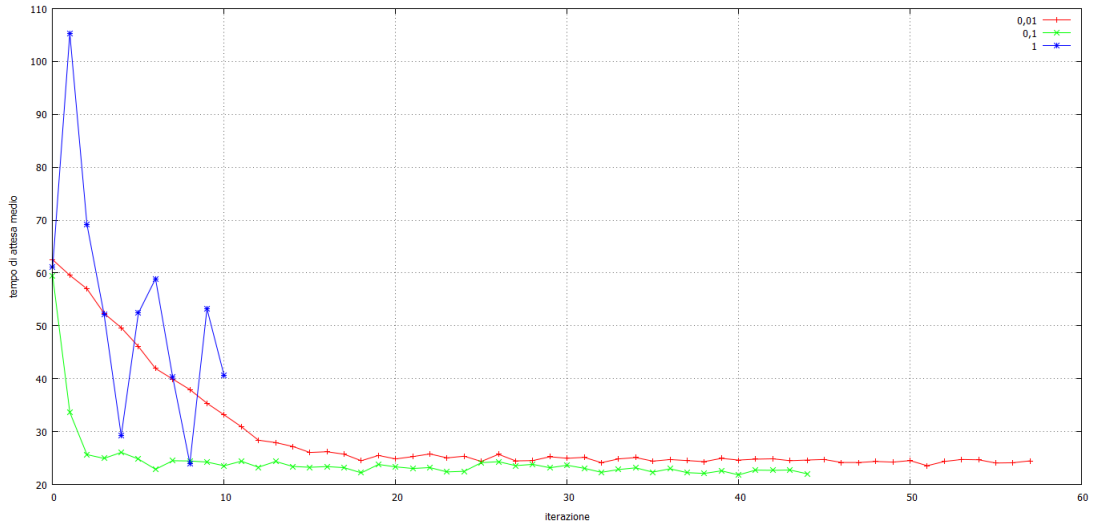


(b) Tempo di attesa massimo (secondi)

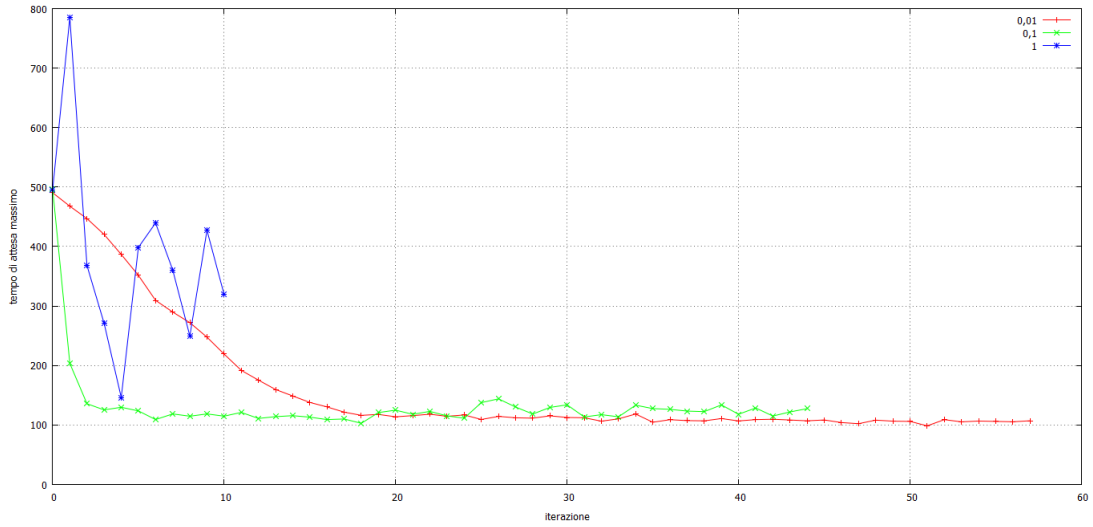


(c) Rinforzo

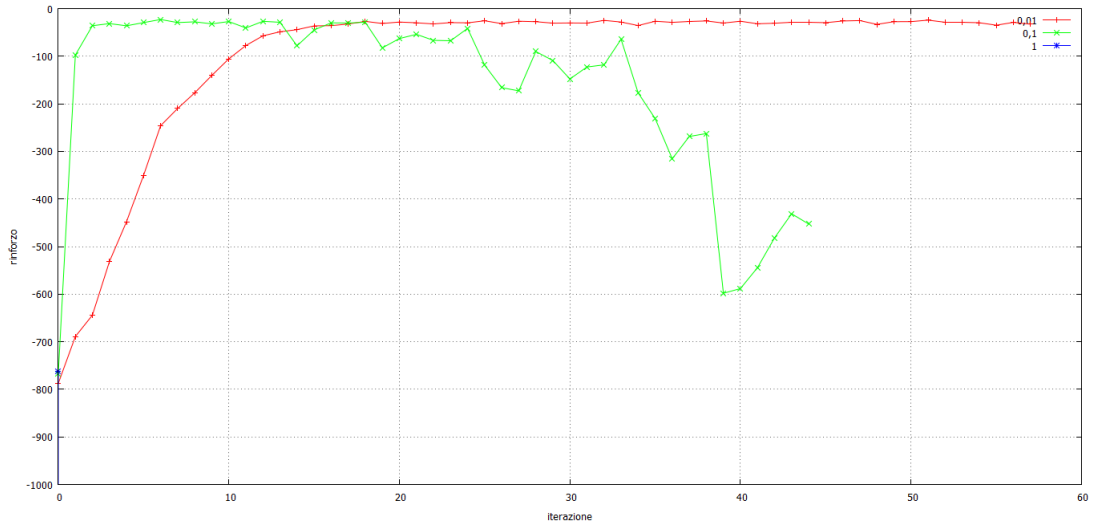
Figura 5.6: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo uppeak, al variare di α; l'insieme di feature utilizzato è quello completo.



(a) Tempo di attesa medio (secondi)

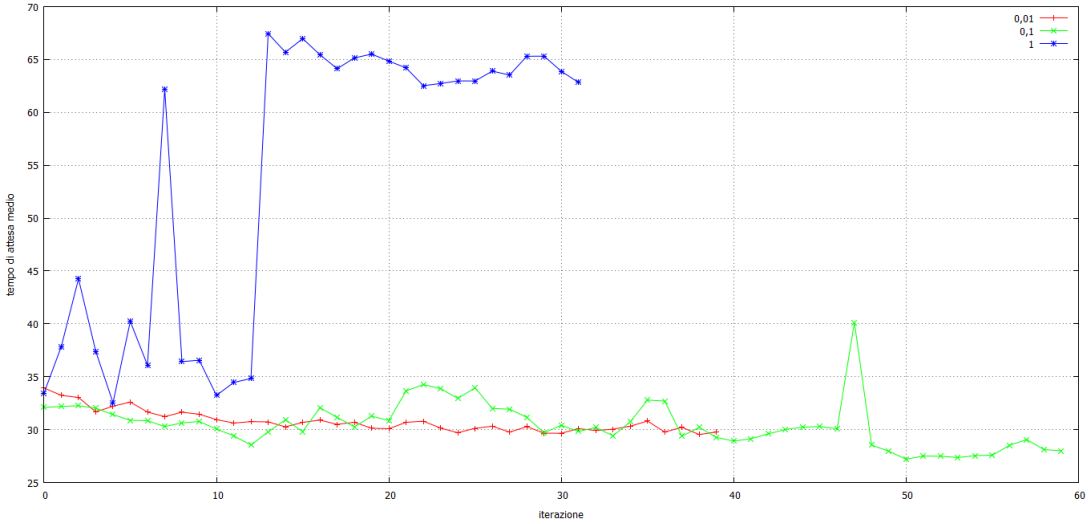


(b) Tempo di attesa massimo (secondi)

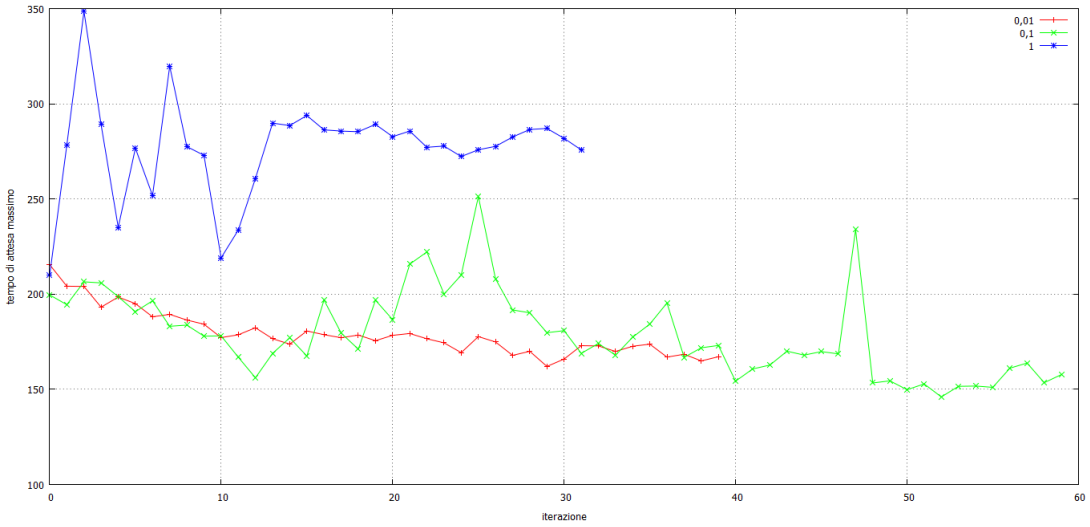


(c) Rinforzo

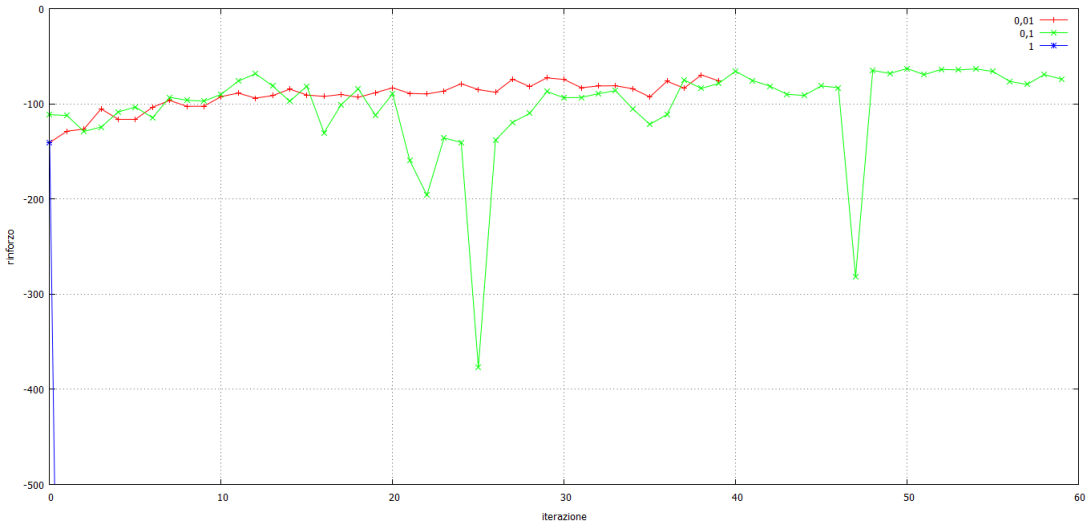
Figura 5.7: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak, al variare di α; l'insieme di feature utilizzato è quello completo.



(a) Tempo di attesa medio (secondi)

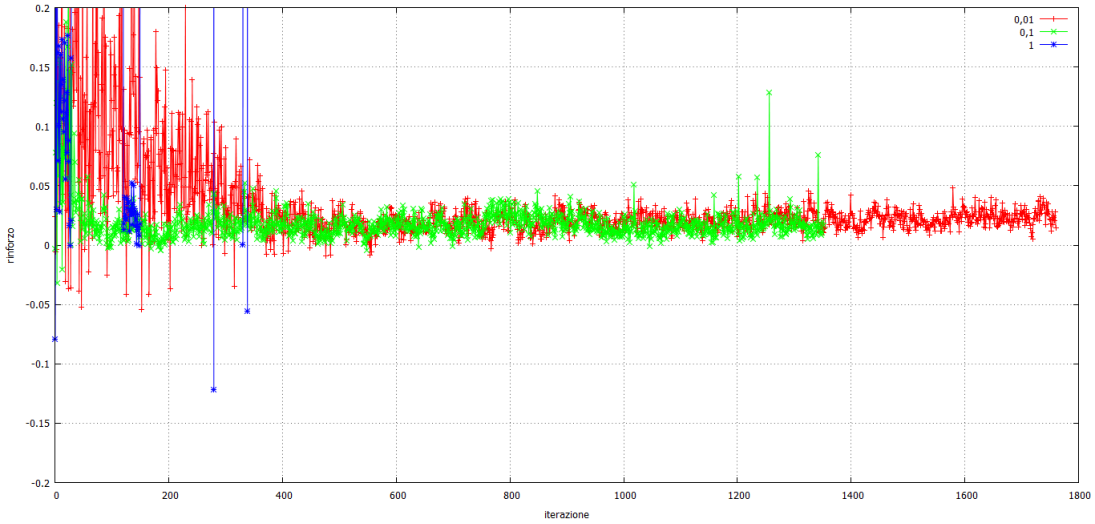


(b) Tempo di attesa massimo (secondi)

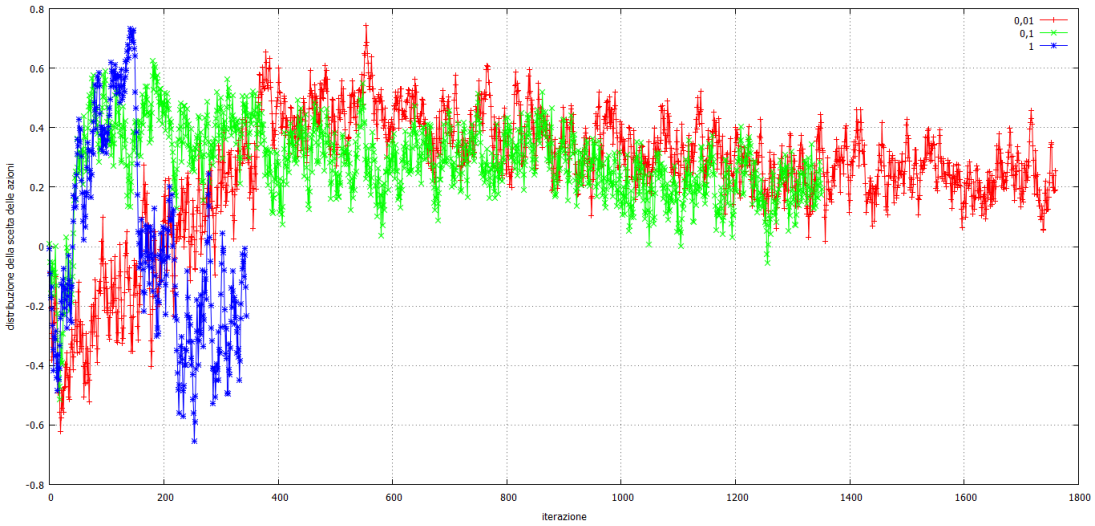


(c) Rinforzo

Figura 5.8: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor, al variare di α; l'insieme di feature utilizzato è quello completo.

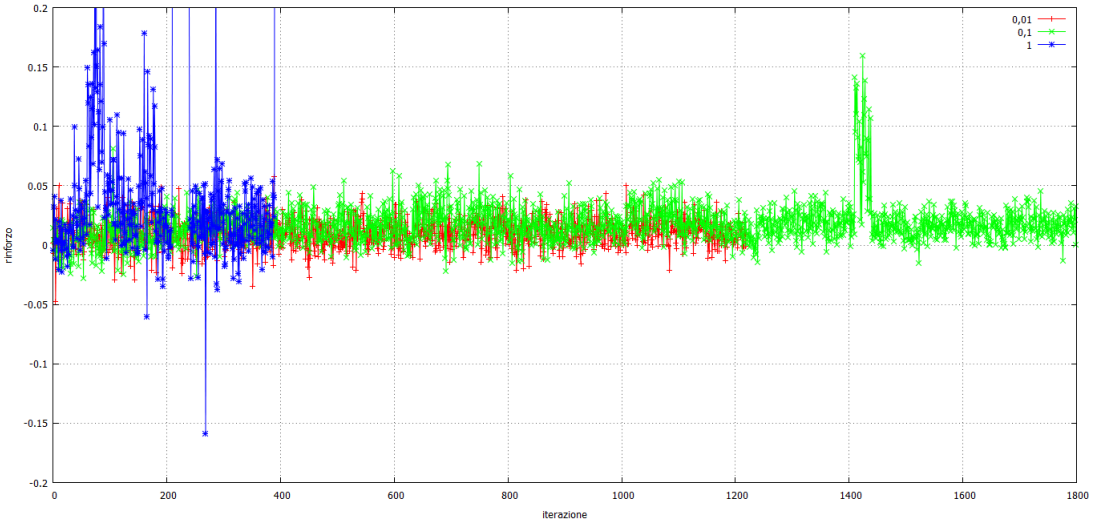


(a) Rinforzo

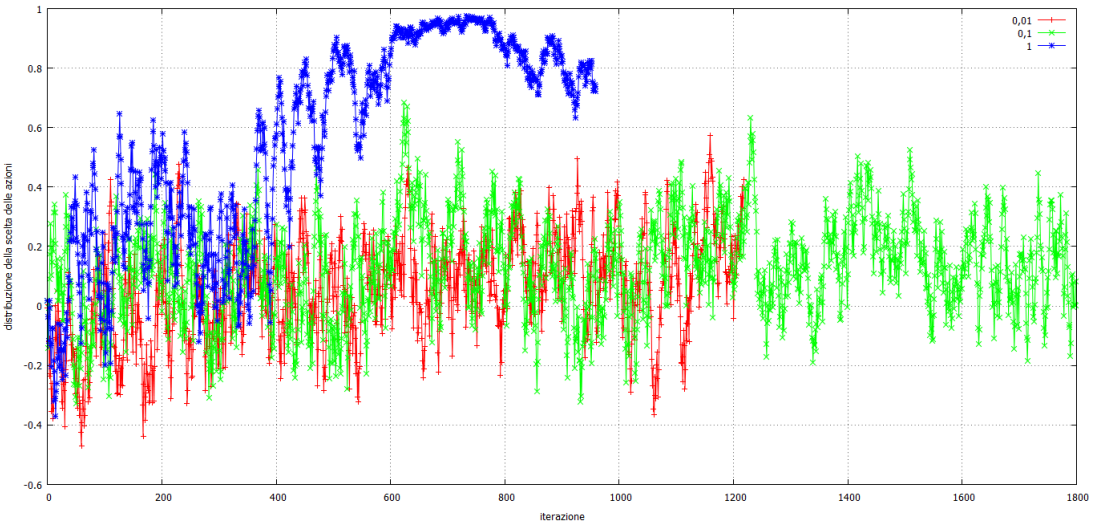


(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.9: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo.



(a) Rinforzo



(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.10: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo.

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	3,897	21,960	27,159
tempo di attesa massimo (s)	27,414	144,800	183,622
tempo di viaggio medio (s)	27,347	23,406	25,444
tempo di sistema medio (s)	31,244	45,365	52,603
numero di fermate degli ascensori	880,354	1819,856	1883,684

Tabella 5.10: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature con $\alpha = 0,1$ in situazione di traffico leggero.

uppeak (per questo motivo il tempo di attesa è salito verso un valore molto alto), oppure di ottenere una politica che non riesce a servire tutte le persone nel sistema come è accaduto ad esempio nel caso interfloor, in cui gli ultimi tre piani, dopo le prime iterazioni, sono stati ignorati dal controllore.

I valori 0,1 e 0,01, invece, sono risultati più adeguati per il parametro α al livello superiore; in particolare, scegliendo 0,1 e non 0,01 il controllore raggiunge più velocemente delle buone prestazioni e, a lungo termine, le prestazioni raggiungono un livello leggermente migliore, anche se tutto sommato paragonabile, ma, escludendo il traffico di tipo uppeak, abbastanza instabile e con qualche incongruenza nell'andamento delle quantità prese in esame. Di conseguenza, riteniamo che per l'apprendimento del controllore vada utilizzato un valore di α al livello superiore compreso tra questi due estremi, con la scelta che dipende da quali aspetti si vuole privilegiare.

Si può notare, inoltre, che il valore di α scelto al livello superiore influenza anche l'apprendimento al livello inferiore; infatti, se la politica con ascensori vuoti è buona, al livello con ascensori pieni si riesce a gestire la situazione e portare a una diminuzione del tempo di attesa totale nel sistema, mentre in caso contrario la situazione generale del sistema continua a peggiorare. Inoltre, si può notare che in downpeak risulta più spesso preferibile fermarsi ai piani intermedi, mentre in interfloor la scelta è molto variabile.

L'andamento del rinforzo al livello con ascensori pieni potrebbe sembrare strano; esso, infatti, durante le prime iterazioni tende a diminuire. Questo fatto, però, è normale, in quanto il rinforzo che si ottiene a questo livello dipende anche dalle prestazioni dell'altro livello della gerarchia; quando la politica al livello con ascensori vuoti migliora si hanno anche meno richieste servibile da ascensori pieni e, di conseguenza, il rinforzo che si ottiene a questo livello non può più assumere i valori elevati che assumeva all'inizio.

In Tabella 5.10 abbiamo riportato i risultati ottenuti dal controllore appreso con α=0,1 mediati su 500 simulazioni di 60 minuti, in modo da poterli confrontare con quelli ottenuti dai controllori benchmark. Il controllore appreso risulta il migliore con traffico di tipo uppeak, mentre le prestazioni con traffico di tipo downpeak e interfloor risultano abbastanza distanti da quelle ottenute dai controllori benchmark.

La politica appresa in uppeak è molto semplice: appena un ascensore è libero viene mandato al piano terra; per questo motivo anche la curva di apprendimento è molto semplice, abbastanza rapida e poco rumorosa. Di conseguenza, il controllore riesce a ottenere delle prestazioni migliori rispetto ai benchmark, che sono progettati per funzionare indifferentemente in qualsiasi situazione di traffico e quindi non possono utilizzare una politica così semplice. Inoltre, questa politica risulta problematica nel caso il traffico uppeak non sia puro, ma rumoroso, in quanto le persone che hanno delle richieste diverse da quella tipica della condizione di traffico non sarebbero considerate; essendo, però, il controllore adattativo, la politica può variare in modo da essere adeguata anche per servire i disturbi particolari che caratterizzano il traffico di tipo uppeak dell'edificio in cui il controllore è installato.

In downpeak e in interfloor, invece, la strategia è molto più articolata che in uppeak e le performance del controllore RLG gerarchico rimangono abbastanza distanti rispetto a quelle ottenute dai migliori controllori benchmark. In particolare, il nostro controllore riesce a coordinare bene i diversi ascensori presenti nel sistema, ma fa fatica a sfruttare le informazioni sulle chiamate effettivamente presenti nel sistema; nella pratica, un ascensore vuoto viene inviato al piano ritenuto più conveniente data la distribuzione degli altri ascensori e la tipologia del traffico presente, apparentemente indipendente dalla presenza o meno di richieste a quel piano. Ad esempio, in downpeak generalmente gli ascensori vengono inviati ai piani più alti, in modo da servire eventuali richieste presenti a quei piani e poi, mentre scendono verso il piano terra, servire anche quelle ai piani intermedi; se, però, ci sono già degli ascensori che servono i piani alti, il controllore preferisce mandare un nuovo ascensore ai piani intermedi in modo da poter servire subito eventuali richieste presenti a quei piani e più velocemente quelle ai piani bassi, dato che i piani alti sono già servibili in breve tempo.

Questo comportamento spiega anche il fatto che in downpeak, durante l'apprendimento, ci sono dei miglioramenti nelle performance delle politiche rispetto a quella iniziale casuale, mentre in interfloor i miglioramenti sono minimi. Infatti, mentre in downpeak la strategia appena descritta porta dei notevoli vantaggi rispetto a una strategia completamente casuale, in interfloor, se non si usano le informazioni sulle chiamate, non si può fare molto di meglio che andare a un piano qualsiasi, dichiarando una direzione qualsiasi, e sperare di trovare una chiamata da servire; i miglioramenti minimi ottenuti durante l'apprendimento sono dovuti a un certo coordinamento tra gli ascensori, che tendono a distribuirsi sull'estensione dell'edificio, ma non sono molto significativi in quanto anche una politica casuale difficilmente tende a mandare spesso tutti gli ascensori nello stesso posto.

Per contrastare le difficoltà di utilizzo delle informazioni sulle chiamate abbiamo tentato diversi approcci, ma nessuno di questi ha dato i risultati sperati.

Una possibilità consiste nel cambiare il valore che indica bottone non premuto; infatti, essendo le feature utilizzate in modo lineare, potrebbe essere che una sua

modifica renda più facilmente utilizzabile l'informazione di bottone premuto o meno. Nelle pratica, abbiamo provato a utilizzare diversi valori compresi tra -10^3 e -10^{-2}, ma il controllore non ha mai imparato a utilizzare l'informazione in questione; l'unico effetto che questa modifica ha prodotto è simile a quello che ha il fattore di apprendimento, infatti valori più alti portano a un apprendimento più rapido ma molto meno preciso e più problematico. Abbiamo, quindi, deciso di lasciare questo valore a -1, in quanto non influenza le prestazioni e risulta più facilmente leggibile da una persona.

Un altro modo per inserire l'informazione premuta consiste nell'aggiungere una feature booleana per ogni bottone di chiamata, per indicare se il bottone in questione è premuto o meno; anche questa informazione, però, non viene sfruttata dagli agenti che apprendono, ma le prestazioni ottenute con o senza di essa rimangono uguali.

Il problema potrebbe essere anche di natura diversa: una possibilità è che il rinforzo assegnato agli agenti non sia del tutto adeguato e, quindi, essi non riescono ad apprendere che conviene dirigersi ai piani dove ci sono delle richieste. Per questo motivo, abbiamo provato ad assegnare una penalità ogni volta che l'azione scelta porta l'ascensore a un piano in cui non c'e nessuna richiesta da servire; l'ammontare della penalità che abbiamo testato è di diversi tipi: pari al tempo totale di attesa di tutte le richieste presenti nel sistema, nell'ordine di grandezza del massimo rinforzo ottenibile senza penalità, oppure di uno o due ordini di grandezza superiore ad esso. In tutti i casi testati, però, le prestazioni della politica appresa non hanno mostrato alcun miglioramento, ma sono rimaste invariate rispetto al caso senza penalità.

Il rinforzo potrebbe essere non adeguato anche perché è molto rumoroso, in quanto è legato al comportamento generale del sistema e non a quello del singolo ascensore a cui viene assegnato. Abbiamo, quindi, provato ad assegnare un rinforzo personalizzato agli ascensori, dando un punteggio positivo proporzionale al tempo di attesa delle chiamate servite dall'ascensore e togliendo una penalità nel caso l'ascensore si diriga a un piano dove è già fermo o sta per fermarsi un altro ascensore. Anche con questo rinforzo, però, il comportamento degli ascensori rimane immutato.

Un'ultima possibilità consiste nel vincolare la scelta delle azioni ai soli piani in cui sono presenti delle chiamate; così facendo, però, si perde completamente una possibilità di coordinamento tra gli ascensori, che tendono a comportarsi nello stesso modo, soprattutto quando il numero di richieste è basso e quindi è effettivamente corretto mandare ascensori a piani senza richieste in modo da predisporli a servire meglio le chiamate future. Di conseguenza, le prestazioni ottenibili con questo vincolo risultano peggiori di quelle ottenibili senza alcun tipo di vincolo.

Per essere certi di aver definito e formalizzato il problema in modo corretto abbiamo provato a impostare manualmente i parametri della politica Gibbs. Così facendo siamo risusciti a ottenere una politica che abbia il comportamento desiderato, le cui prestazioni, come si può vedere dai dati riportati in Tabella 5.11, sono abbastanza vicine a quelle ottenute dai migliori controllori benchmark. La principale differenza

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	5,464	15,451	16,045
tempo di attesa massimo (s)	29,090	63,063	76,595
tempo di viaggio medio (s)	46,880	20,835	22,172
tempo di sistema medio (s)	52,344	36,286	38,217
numero di fermate degli ascensori	1283,392	1225,936	1351,388

Tabella 5.11: Risultati ottenuti dal controllore definito a mano utilizzando l'insieme completo delle feature in situazione di traffico leggero.

che possiamo notare tra i pesi impostati da noi e quelli appresi tramite gradiente è l'ordine di grandezza che assumono; infatti, mentre quelli appresi difficilmente hanno un ordine di grandezza superiore a 10^{-3}, quelli impostati da noi arrivano anche a ordini di grandezza di 10^{2}.

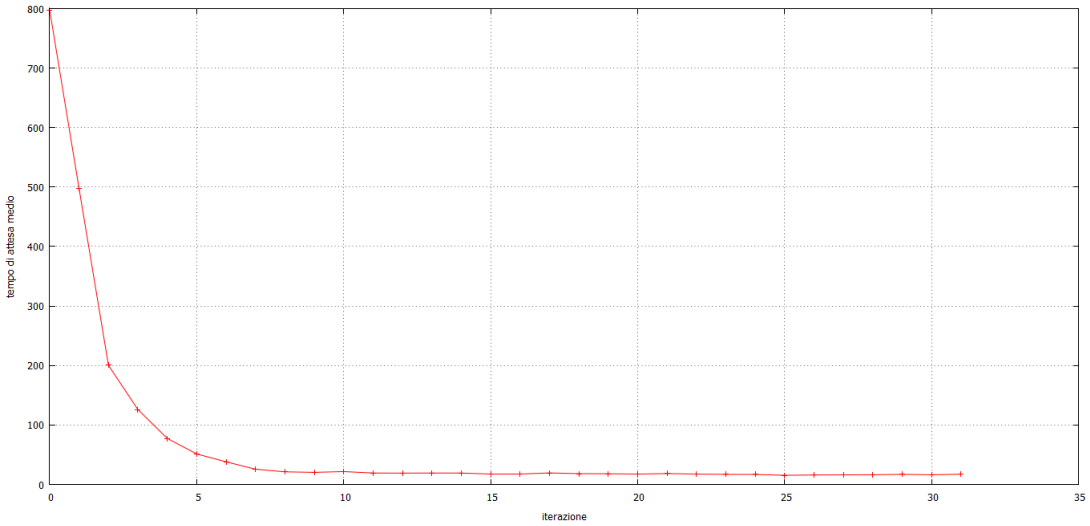
A questo punto, verificato che il problema è definito correttamente, abbiamo provato a far partire l'apprendimento con pesi casuali. Se il range di casualità dei pesi è piccolo, nell'ordine di grandezza dei pesi appresi partendo da 0, il punto di convergenza dell'apprendimento risulta essere molto simile a quello raggiunto partendo con pesi a 0. Se, invece, il range è più ampio la politica iniziale ha prestazione talmente pessime da avere rinforzi troppo elevati, in valore assoluto, che causano problemi numerici; non è nemmeno possibile scalare ulteriormente il rinforzo assegnato, in quanto una volta raggiunta una buona politica esso arriverebbe ad essere troppo piatto.

Provando, invece, a lanciare l'apprendimento con i pesi fissati da noi le prestazioni della politica rimangono sullo stesso livello di quelle di partenza.

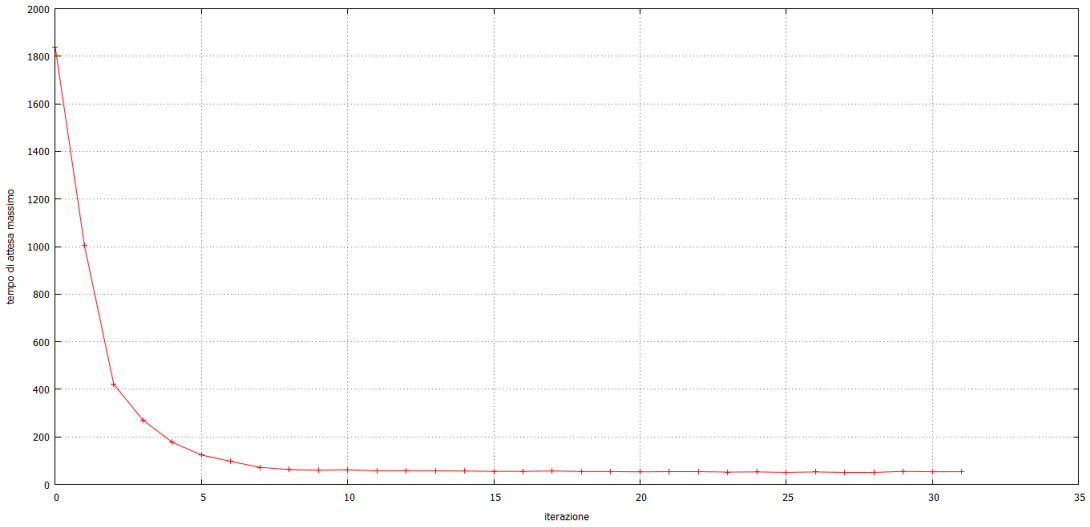
In base ai risultati ottenuti siamo, quindi, giunti alla conclusione che la funzione da ottimizzare al livello con ascensori vuoti è piuttosto difficile per essere trattata con metodi gradiente; essa, infatti, sembrerebbe avere molti massimi locali vicini tra loro e che garantiscono prestazioni simili, con altri massimi locali che garantiscono prestazioni differenti in altri punti della funzione separati da zone le cui prestazioni non sono accettabili.

Traffico intenso. Dati i risultati ottenuti con traffico leggero, per quanto riguarda il traffico intenso abbiamo deciso di utilizzare per il parametro α degli agenti al livello con ascensori vuoti il valore 0,1 in uppeak, dato che in questo caso l'apprendimento è abbastanza semplice, mentre in downpeak e interfloor abbiamo utilizzato il valore 0,025, in modo da ottenere un apprendimento maggiormente robusto.

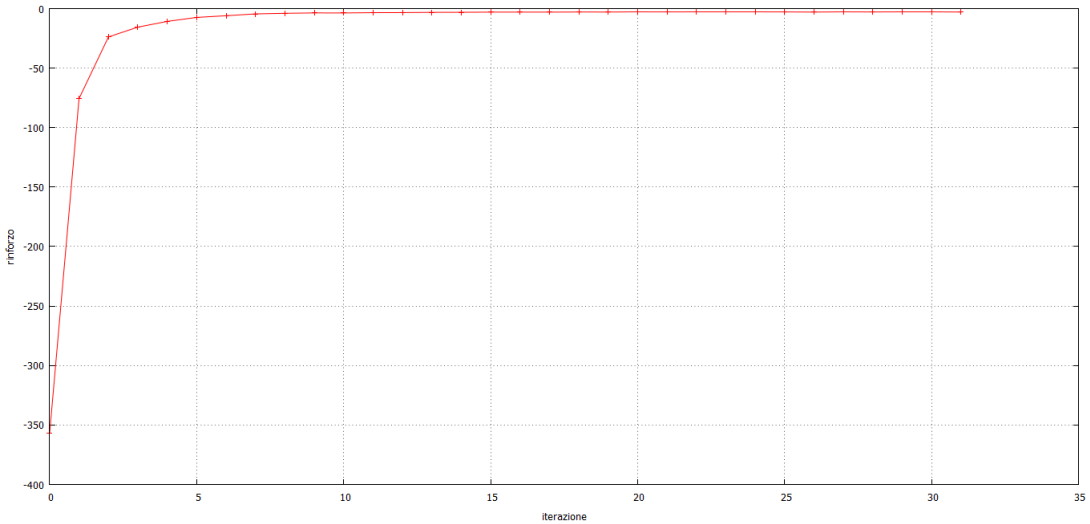
Le curve d'apprendimento degli agenti al livello gerarchico superiore sono mostrate in Figura 5.11, 5.12 e 5.13, mentre quelle per l'apprendimento al livello inferiore sono mostrate in Figura 5.14 e 5.15; le quantità mostrate nei grafici sono le stesse dello scenario con traffico leggero, con l'ovvia eccezione che in questo caso non c'è



(a) Tempo di attesa medio (secondi)

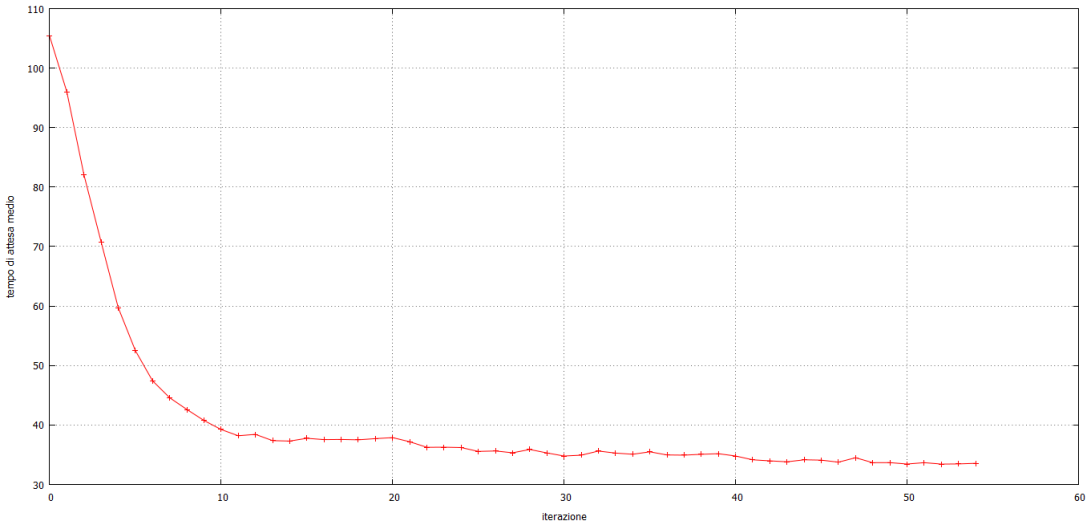


(b) Tempo di attesa massimo (secondi)

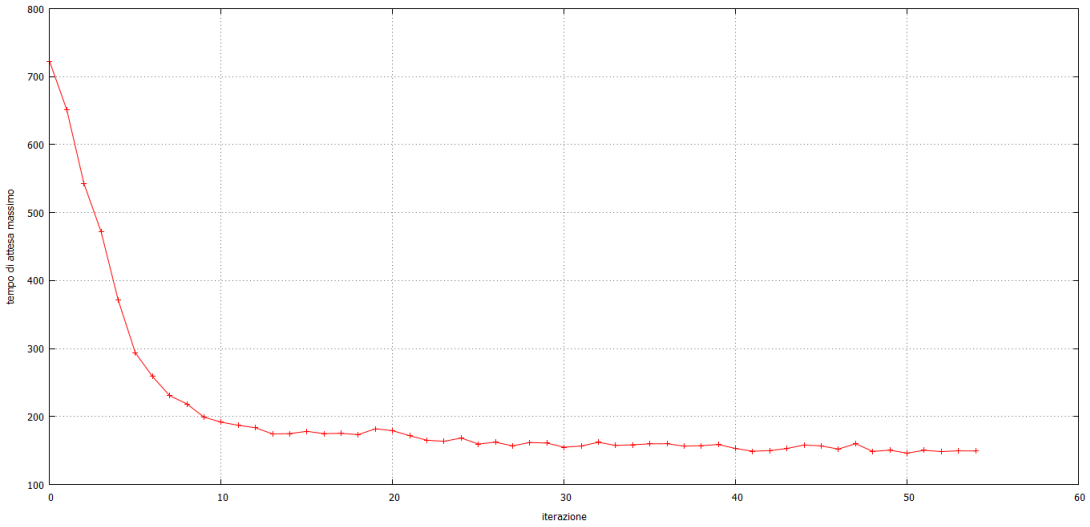


(c) Rinforzo

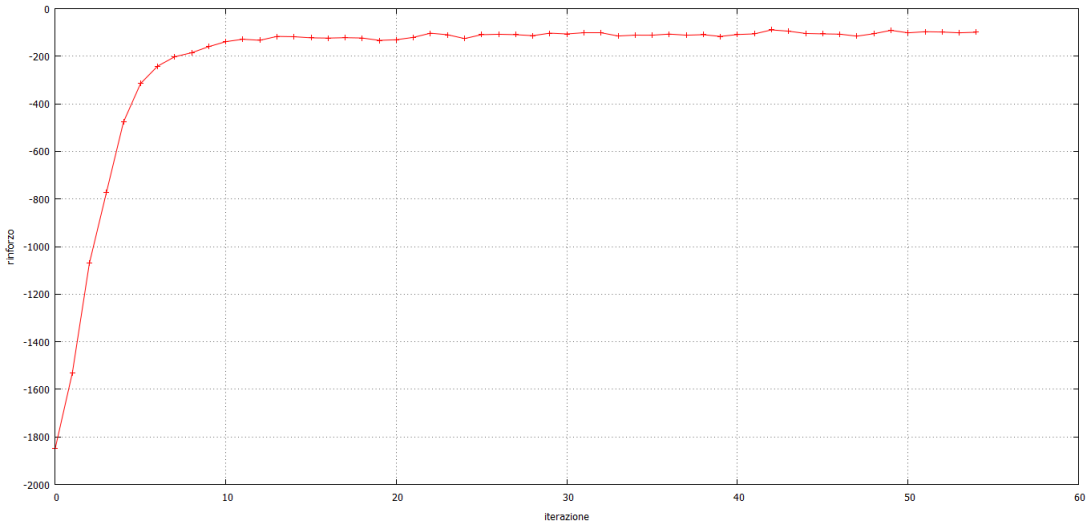
Figura 5.11: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo uppeak; l'insieme di feature utilizzato è quello completo.



(a) Tempo di attesa medio (secondi)

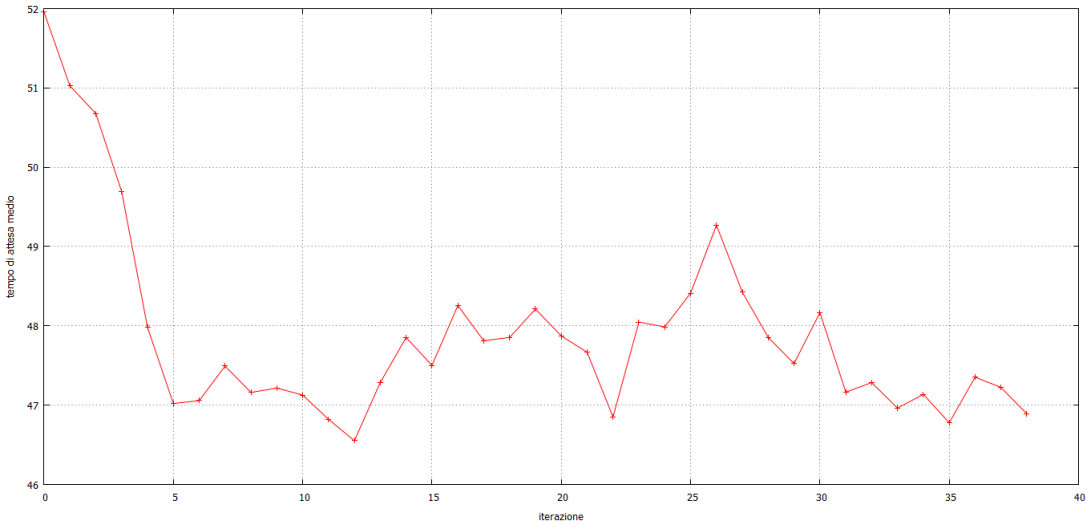


(b) Tempo di attesa massimo (secondi)

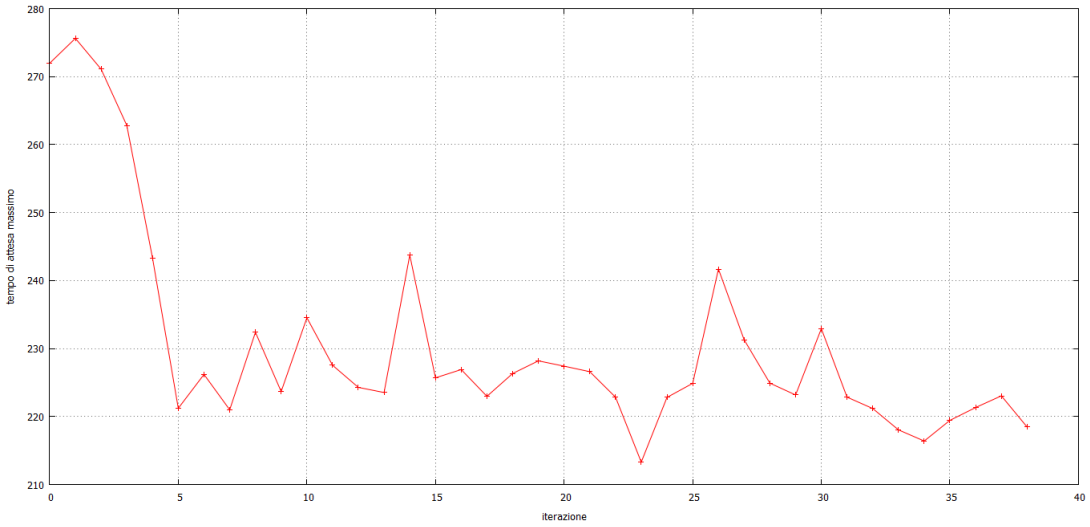


(c) Rinforzo

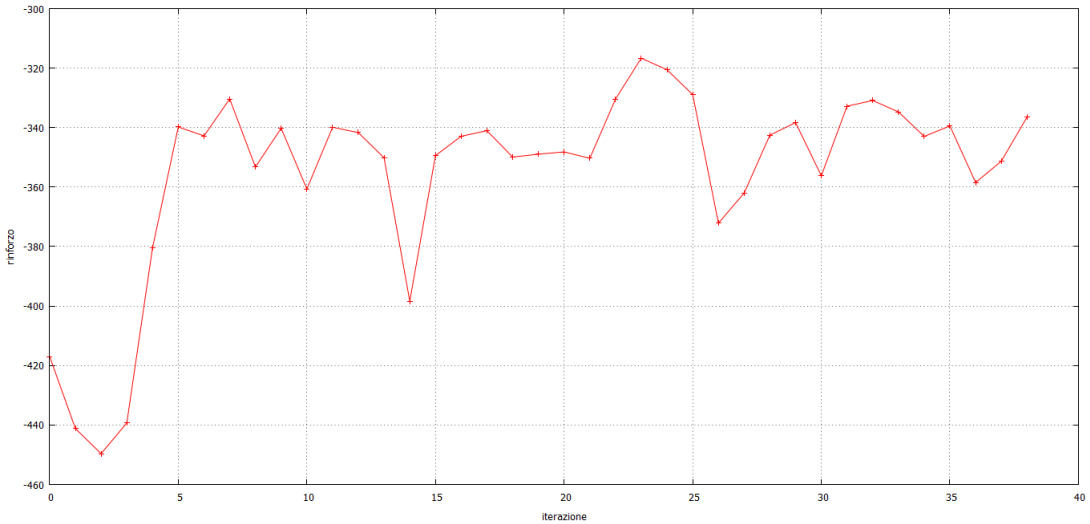
Figura 5.12: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello completo.



(a) Tempo di attesa medio (secondi)

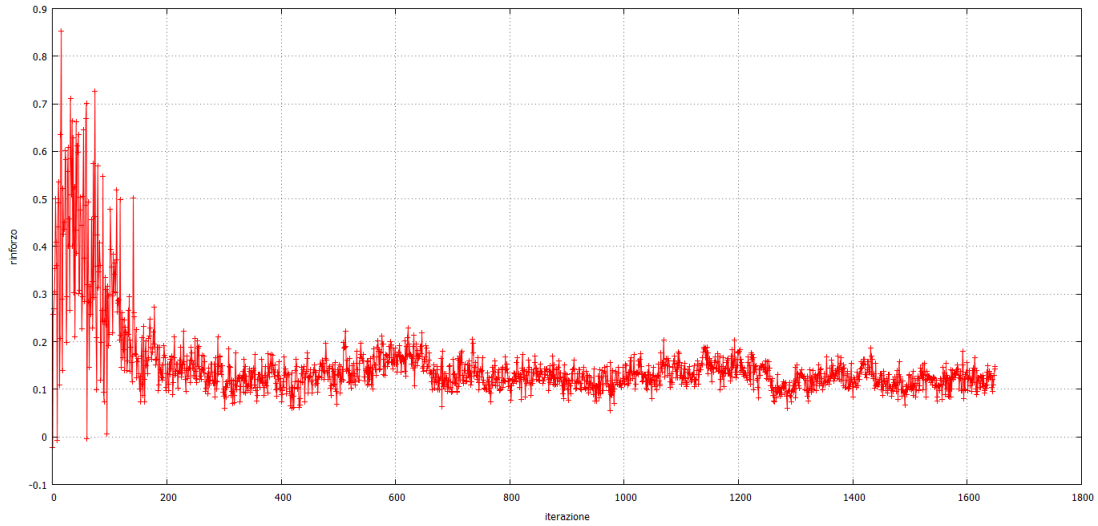


(b) Tempo di attesa massimo (secondi)

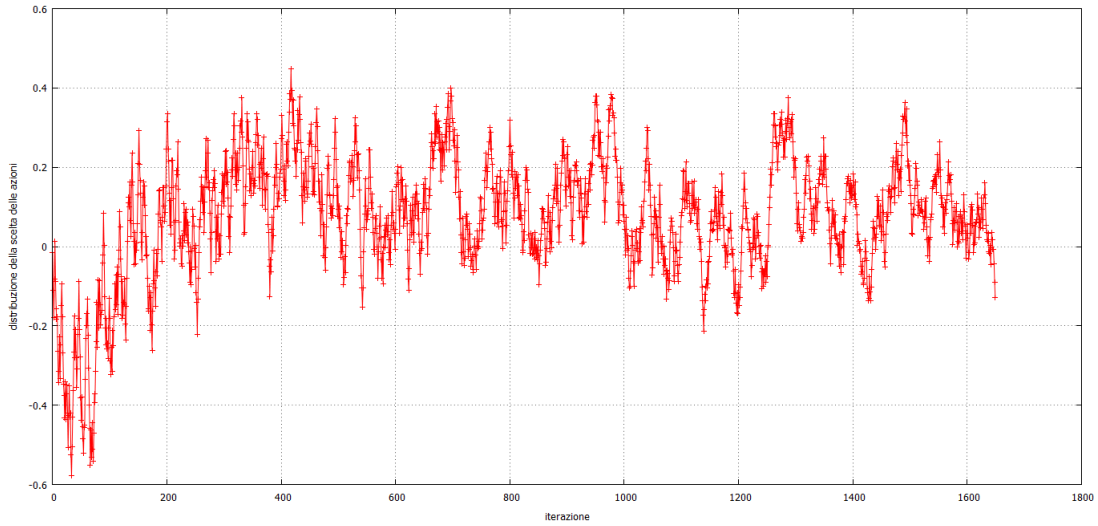


(c) Rinforzo

Figura 5.13: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello completo.

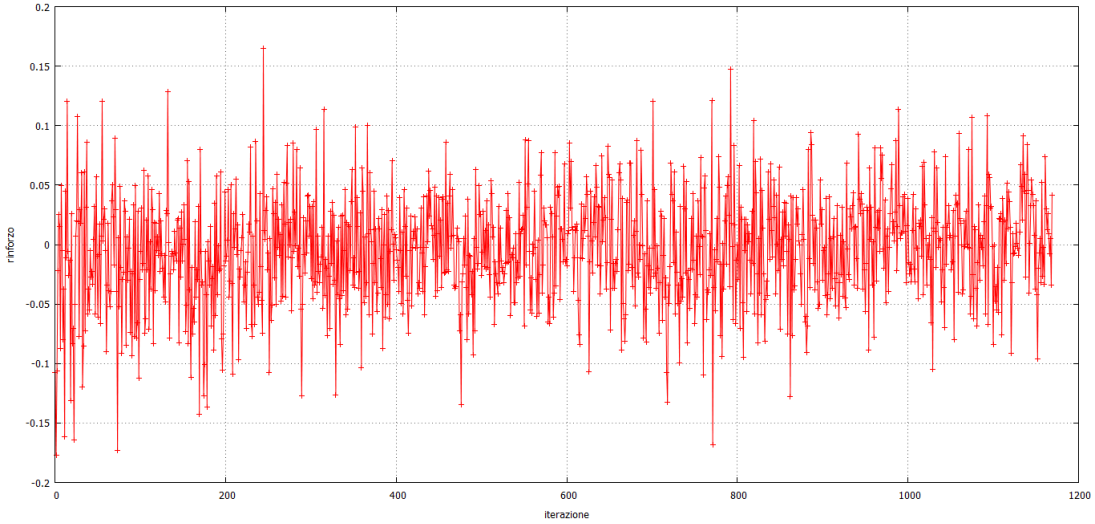


(a) Rinforzo

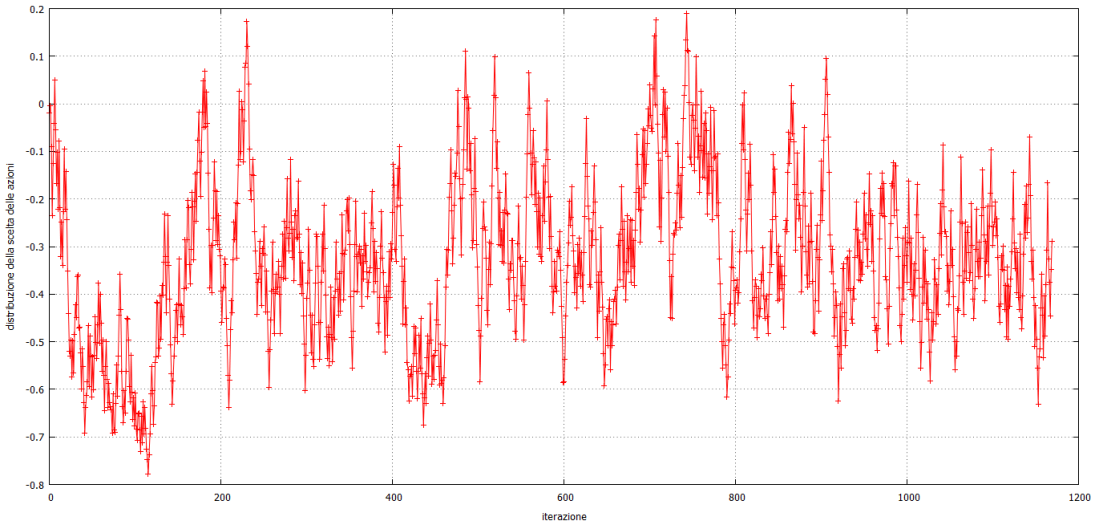


(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.14: Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello completo.



(a) Rinforzo



(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.15: Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello completo.

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	25,134	35,996	50,087
tempo di attesa massimo (s)	84,739	196,093	276,101
tempo di viaggio medio (s)	55,941	28,965	47,794
tempo di sistema medio (s)	81,075	64,962	97,881
numero di fermate degli ascensori	1614,324	1486,876	1585,582

Tabella 5.12: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico intenso.

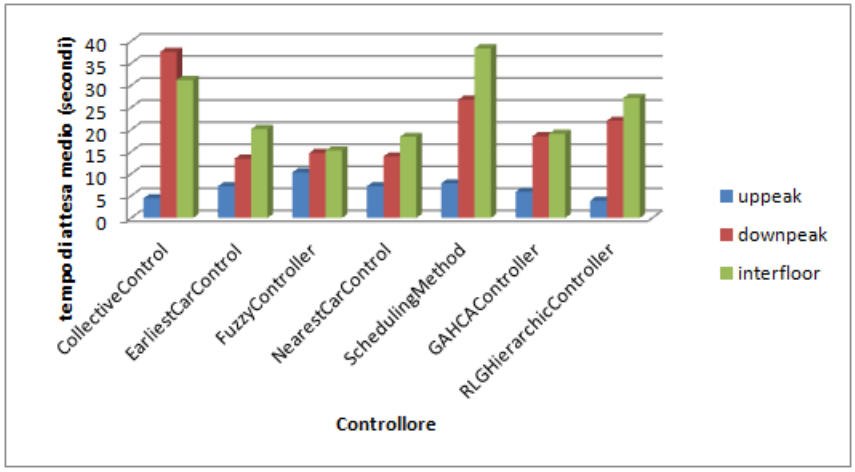


Figura 5.16: Confronto dei risultati ottenuti con i diversi schemi di traffico dal nostro controllore rispetto a quelli benchmark, con traffico leggero.

nessun confronto con valori differenti di α. Le prestazioni del controllore appreso mediate su 500 simulazioni di 60 minuti sono invece riportate in Tabella 5.12.

Confrontando le prestazioni del nostro controllore con quelle dei controllori benchmark si può notare che, anche con traffico intenso, in tutte le tipologie esiste almeno un controllore che si comporta meglio del nostro. A differenza di quanto avviene col traffico leggero, però, la differenza risulta abbastanza ridotta e, comunque, il nostro controllore risulta peggiore di solamente uno o al massimo due benchmark; inoltre, assumendo una visione generale della situazione, che comprende tutte e tre le tipologie di traffico testate, il nostro controllore è preferibile a tutti i benchmark presi singolarmente e gli unici con cui compete realmente sono il controllore fuzzy e quello genetico, mentre tutti i controllori euristici hanno prestazioni completamente inaccettabili in almeno una tipologia di traffico.

Considerazioni. Come risulta evidente dagli istogrammi di Figura 5.16, per quanto concerne la configurazione con traffico leggero, e Figura 5.17, per il traffico intenso, il controllore RLG gerarchico proposto ha delle prestazioni nettamente inferiori ri-

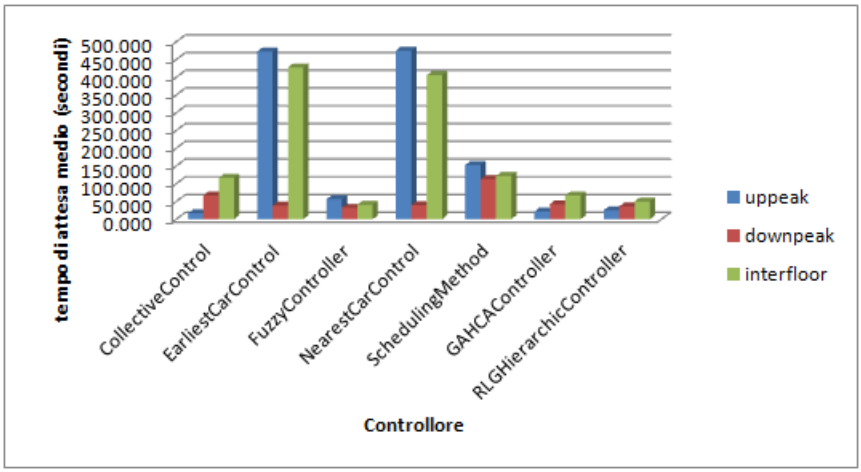


Figura 5.17: Confronto dei risultati ottenuti con i diversi schemi di traffico dal nostro controllore rispetto a quelli benchmark, con traffico intenso.

spetto alla maggior parte dei controllori benchmark, mentre risulta essere quello con prestazioni mediamente migliori in condizioni di traffico intenso.

Nei controllori benchmark euristici e, in misura minore, in quelli progettati con tecniche di intelligenza artificiale si può notare un degrado netto delle prestazioni tra la configurazione con traffico leggero e quella con traffico intenso, certamente non spiegabile solamente con il maggior numero di persone da servire; inoltre, cambia anche l'efficienza relativa tra questi controllori e, ancor più gravemente, l'efficienza relativa dello stesso controllore tra le diverse tipologie di traffico.

Il fenomeno appena descritto, invece, non si verifica col controllore proposto; in esso, infatti, il degrado delle prestazioni è abbastanza contenuto e comunque spiegabile semplicemente con l'aumento del numero di persone da servire. Inoltre, esso si dimostra coerente rispetto alla capacità di servire le diverse tipologie di traffico.

Alla considerazione appena fatta si potrebbe obiettare che, nella realtà, il controllore RLG gerarchico usato per controllare il sistema con traffico leggero e quello usato con traffico intenso sono due controllori diversi, in quanto essi sono frutto di apprendimenti separati nelle due condizioni e quindi anche i parametri a cui convergono sono differenti. Questa obiezione è senz'altro significativa, ma i due controllori sono comunque frutto dello stesso approccio e, immaginando di lasciare la possibilità di apprendimento anche dopo l'installazione in un sistema, il controllore studiato per una certa condizione è in grado di adeguarsi in breve tempo al cambiamento della suddetta condizione. Inoltre, come emerge dai dati riportati in Tabella 5.13 e Tabella 5.14, utilizzare un controllore appreso in una condizione di traffico per controllare il sistema caratterizzato dall'altra situazione di traffico porta comunque a dei buoni risultati e, quindi, non è nemmeno necessario richiedere un adattamento del controllore in questione e, comunque, se si lascia avvenire l'adattamento si ha una buona base di partenza. Ne consegue, quindi, che le considerazioni fatte fino a

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	4,891	23,701	34,133
tempo di attesa massimo (s)	27,215	124,851	282,091
tempo di viaggio medio (s)	30,537	23,872	22,695
tempo di sistema medio (s)	35,428	47,573	56,829
numero di fermate degli ascensori	1742,494	1779,206	1870,098

Tabella 5.13: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico intenso per controllare il sistema con traffico leggero.

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	23,500	35,494	46,898
tempo di attesa massimo (s)	84,692	276,233	211,457
tempo di viaggio medio (s)	54,795	28,455	47,430
tempo di sistema medio (s)	78,295	63,949	94,327
numero di fermate degli ascensori	947,544	1521,214	1627,438

Tabella 5.14: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature in situazione di traffico leggero per controllare il sistema con traffico intenso.

ora sono corrette e significative.

Nella pratica, quindi, installare in un sistema reale un controllore adattativo come quello proposto è, a lungo termine, molto conveniente; infatti, anche se dopo la sua installazione avviene un cambiamento nella configurazione del sistema rispetto a quanto previsto, come può essere l'aumento del numero di persone che popolano l'edificio, esso è in grado di adeguarsi autonomamente al cambiamento, senza richiedere nessun tipo di intervento e, quindi, senza comportare nessuna spesa aggiuntiva. Al contrario, se si decide di installare un controllore tradizionale, che magari offre anche prestazioni migliori rispetto al controllore proposto nello scenario previsto, in seguito a un possibile cambiamento si rischia di ritrovarsi con un sistema di controllo non più adeguato, che quindi necessita di modifiche o, in casi estremi, di una completa riprogettazione, comportando di conseguenza notevoli costi.

Altre prove Dai risultati esposti è emerso che la politica appresa tramite gradiente utilizzando l'insieme delle feature ha prestazioni accettabili, ma con traffico leggero non raggiunge i livelli dei controllori benchmark. Abbiamo, quindi, provato a effettuare delle prove di apprendimento molto più lunghe, non interrompendole nemmeno dopo che le prestazioni sono apparentemente arrivate a una condizione di convergenza, per verificare se col tempo si possono ottenere comunque dei miglioramenti; poiché queste prove necessitano di molto tempo, abbiamo deciso di effettuarle

solamente in condizioni di traffico downpeak e interfloor leggero, dato che la politica appresa in uppeak è già buona e che abbiamo verificato che in genere il controllore si adatta bene a intensità di traffico diverse da quelle con cui è stato appreso. Abbiamo, inoltre, deciso di utilizzare al livello con ascensori pieni la versione personalizzata del rinforzo, in modo da capire se con essa la politica appresa a tale livello ne trae vantaggio.

La Figura 5.18 mostra le curve di apprendimento in downpeak per il livello con ascensori vuoti, la Figura 5.19 mostra un ingrandimento delle stesse curve centrato sui valori vicini alla convergenza, mentre la Figura 5.20 mostra le curve di apprendimento in interfloor; l'andamento degli apprendimenti per il livello con ascensori pieni è invece mostrato in Figura 5.21 e 5.22, rispettivamente per traffico downpeak e interfloor.

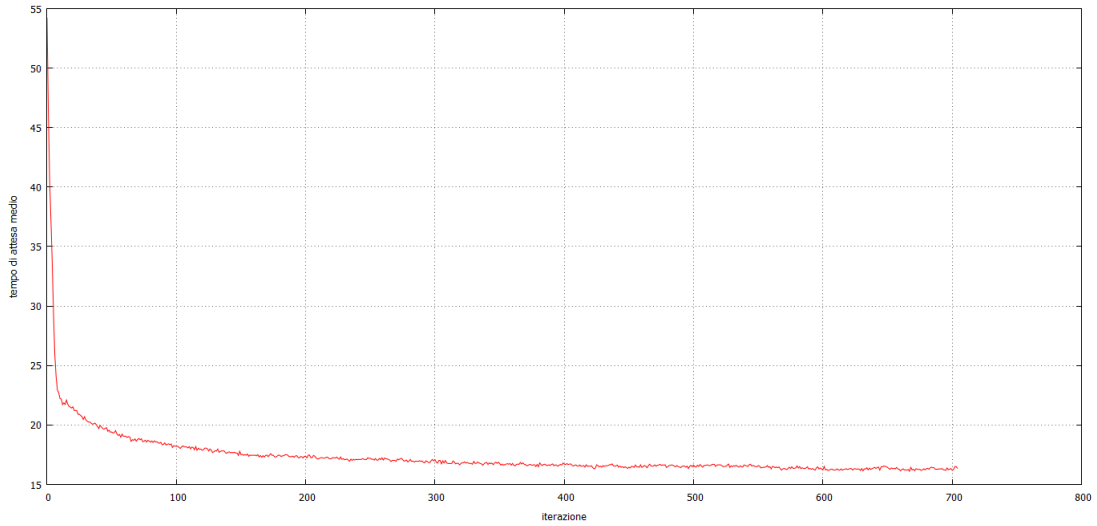
Dalle curve di apprendimento riportate emerge che al livello con ascensori pieni la politica appresa è decisamente più stabile di quella appresa con rinforzo generale; ne consegue che il rinforzo personalizzato è più adeguato per l'apprendimento a questo livello. Inoltre, in interfloor gli ascensori apprendono a fermarsi sempre ai piani intermedi, probabilmente perché la le probabilità che la fermata coincida con quella di un altro ascensore è molto bassa nelle condizioni simulate e, comunque, il tempo che si impiega a fermarsi sempre non penalizza eccessivamente altre chiamate in attesa da più tempo.

Al livello con ascensori vuoti possiamo, invece, notare che l'apprendimento continua, anche se molto lentamente e rumorosamente, anche dopo essere arrivati a una situazione di apparente convergenza; questo andamento è giustificabile sia dalla rumorosità intrinseca del problema, caratterizzato da molta casualità negli eventi, sia da come abbiamo formalizzato il problema, che prevede un rinforzo che dipende da tutti gli agenti del sistema e non solo da quello a cui è assegnato, sia dall'utilizzo per la stima del gradiente di GPOMDP. Inoltre è possibile notare che la curva arriva, anche dopo un numero di iterazioni abbastanza ridotto, a prestazioni leggermente migliori rispetto a quelle precedenti, a indicare che per l'apprendimento con ascensori pieni il rinforzo personalizzato è migliore di quello generale.

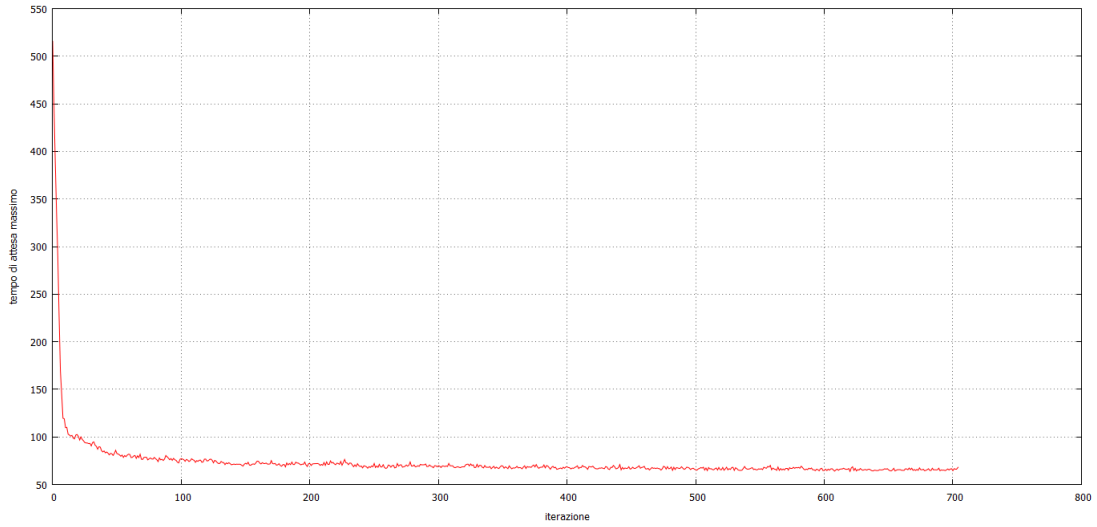
I risultati ottenuti dai controllori appresi in questa configurazione, mediati su 500 simulazioni da 60 minuti, sono riportati in Tabella 5.15.

In condizioni di traffico leggero, quelle utilizzate per l'apprendimento, è possibile notare che le prestazioni sono migliorate sensibilmente e in downpeak non sono nemmeno troppo lontane da quelle dei migliori controllori benchmark; questo è dovuto a un coordinamento molto buono tra gli ascensori, che però mantengono la tendenza a non considerare accuratamente le chiamate presenti nel sistema quando sono vuoti.

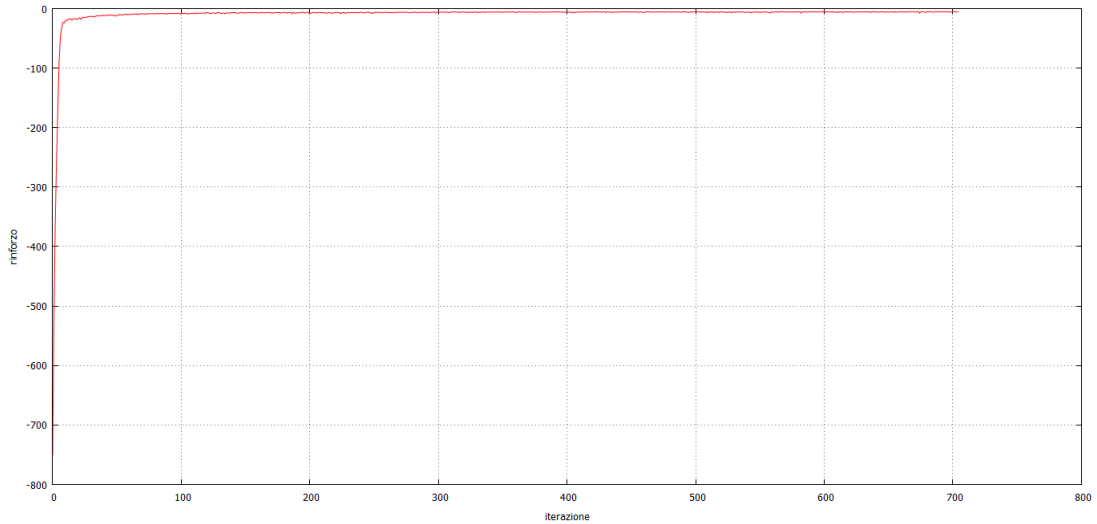
Osservando le prestazioni del controllore con traffico intenso possiamo notare che, mentre in downpeak la politica risulta essere abbastanza generale indipendentemente dall'intensità del traffico (e difatti le prestazioni ottenute in downpeak intenso sono le migliori in assoluto tra tutti i controllori che abbiamo testato), in interfloor la politica



(a) Tempo di attesa medio (secondi)

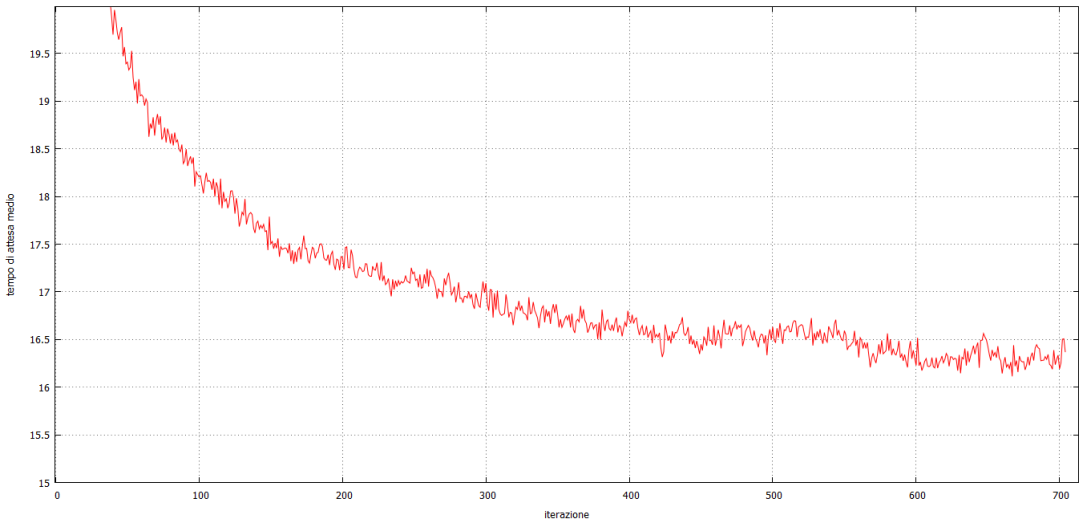


(b) Tempo di attesa massimo (secondi)

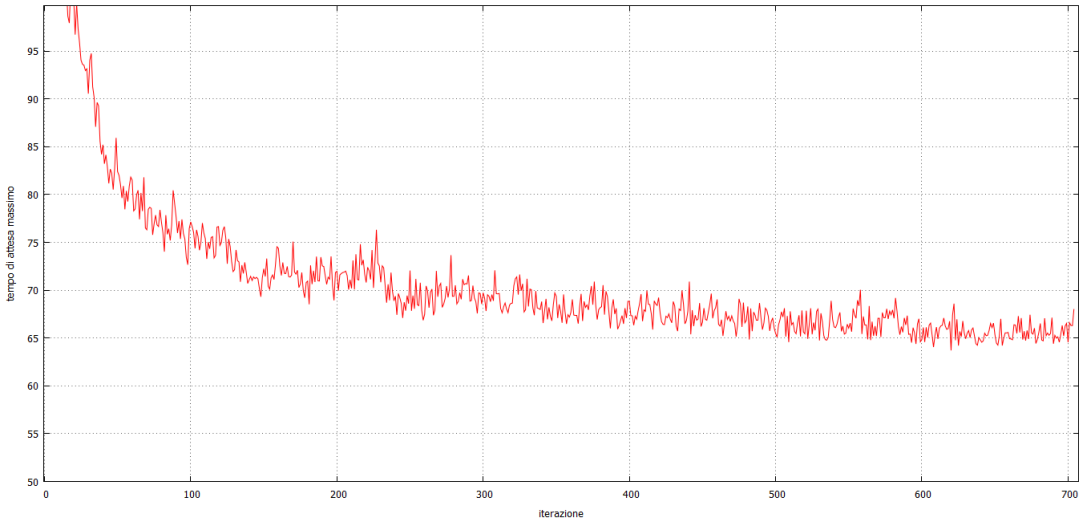


(c) Rinforzo

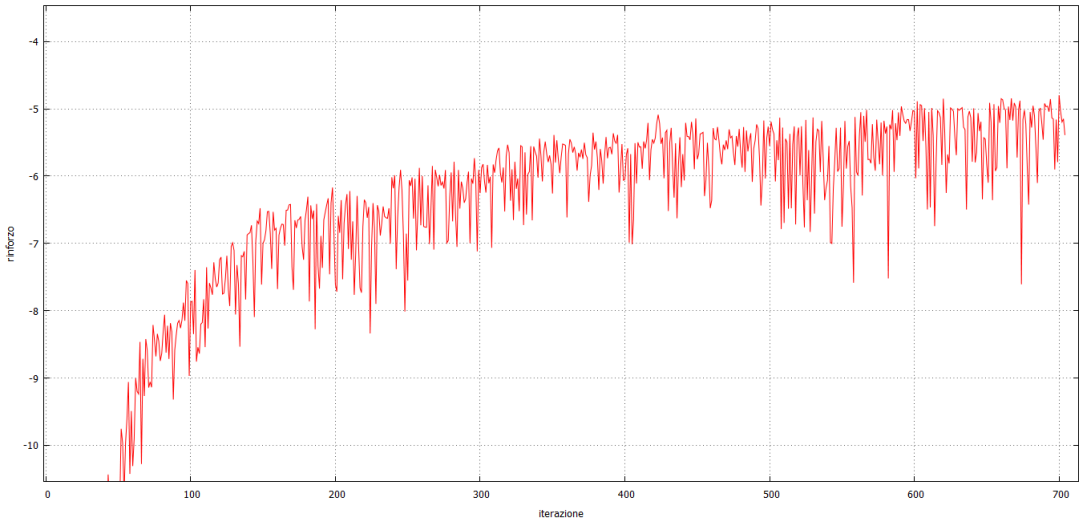
Figura 5.18: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo.



(a) Tempo di attesa medio (secondi)

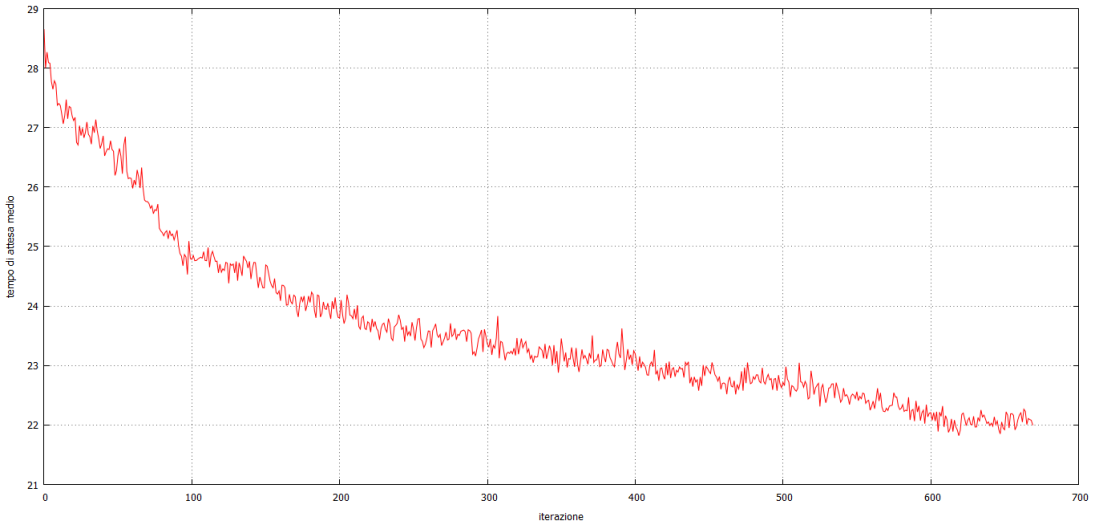


(b) Tempo di attesa massimo (secondi)

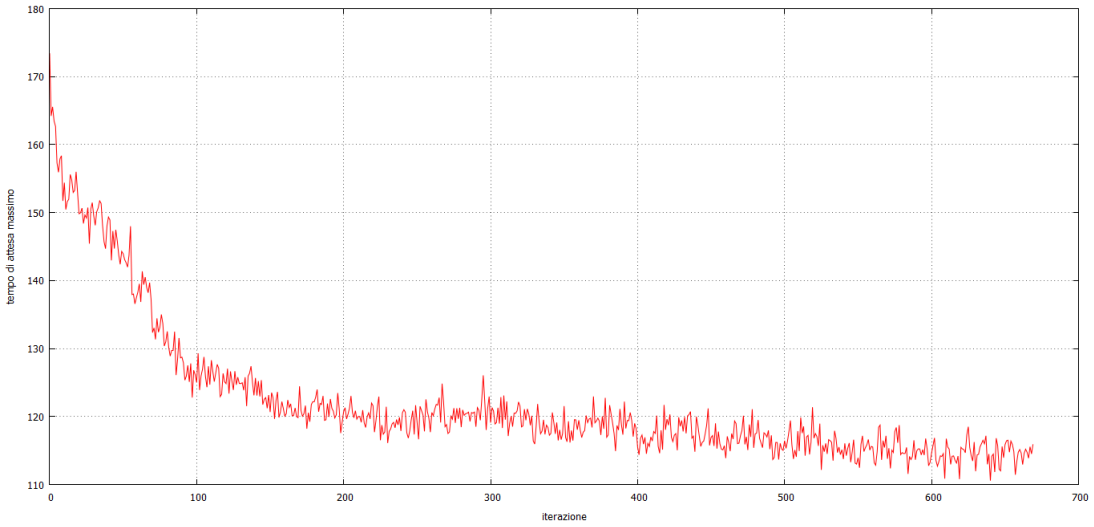


(c) Rinforzo

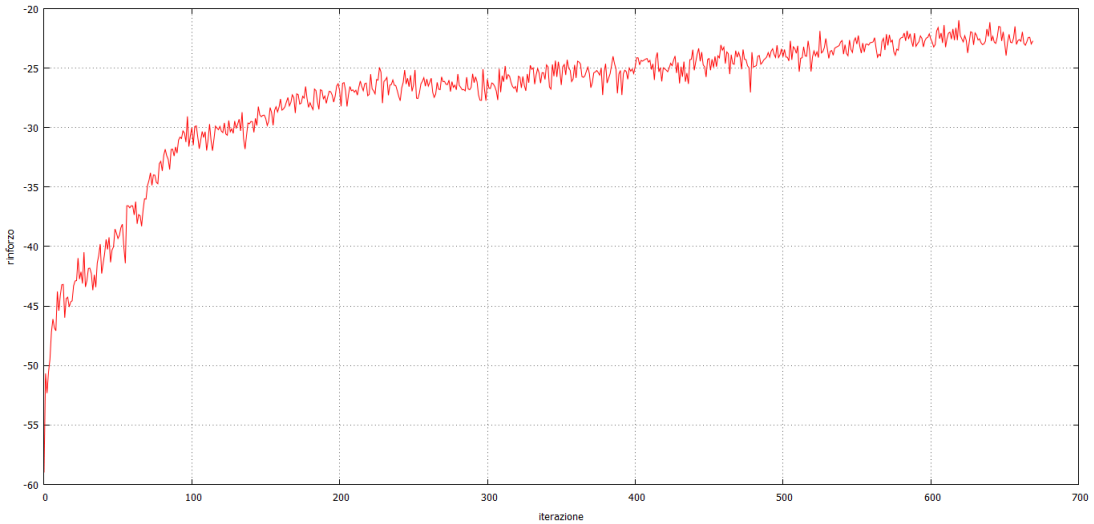
Figura 5.19: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo. I grafici mostrano una vista ingrandita sui valori una volta raggiunta una situazione vicina alla convergenza.



(a) Tempo di attesa medio (secondi)

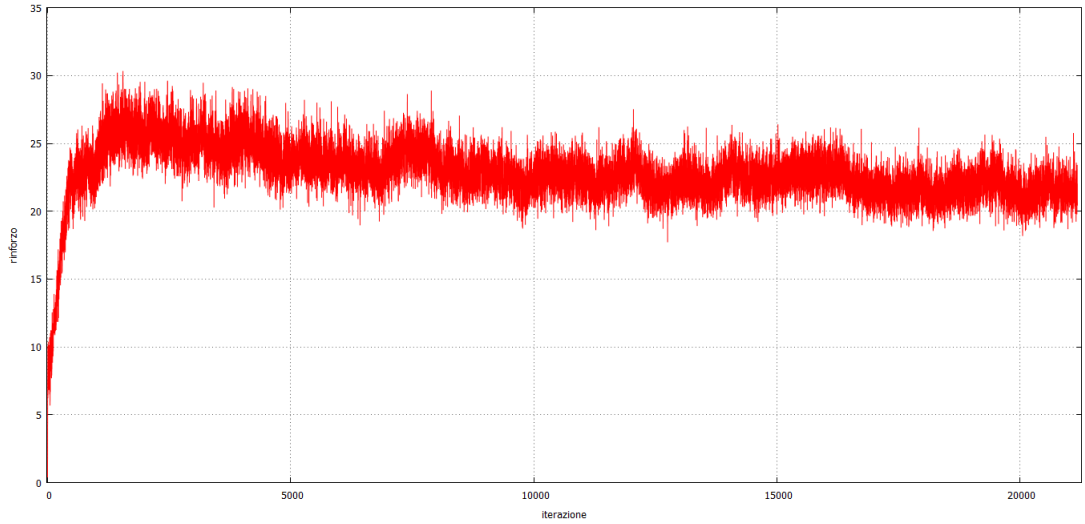


(b) Tempo di attesa massimo (secondi)

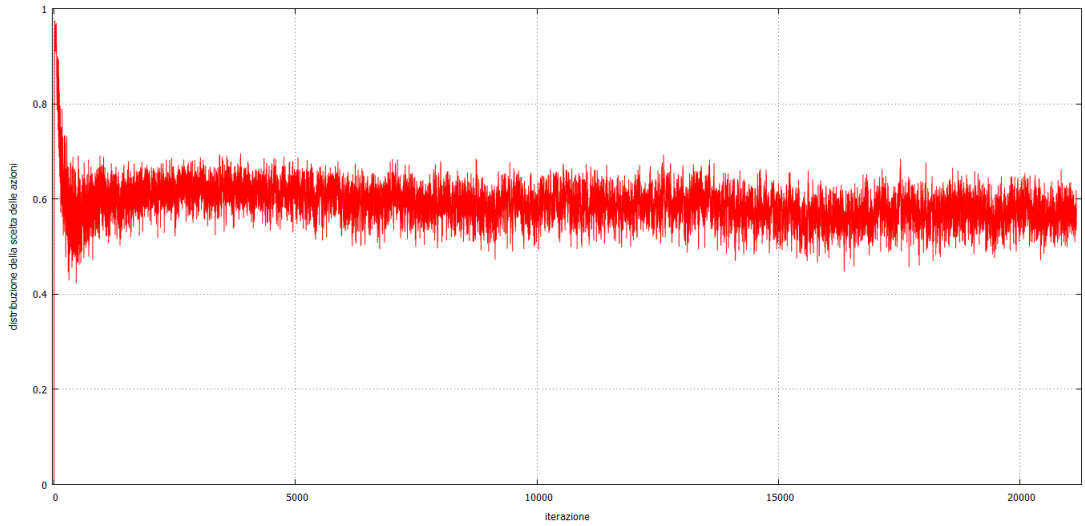


(c) Rinforzo

Figura 5.20: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor con un numero elevato di simulazioni; l'insieme di feature utilizzato è quello completo.

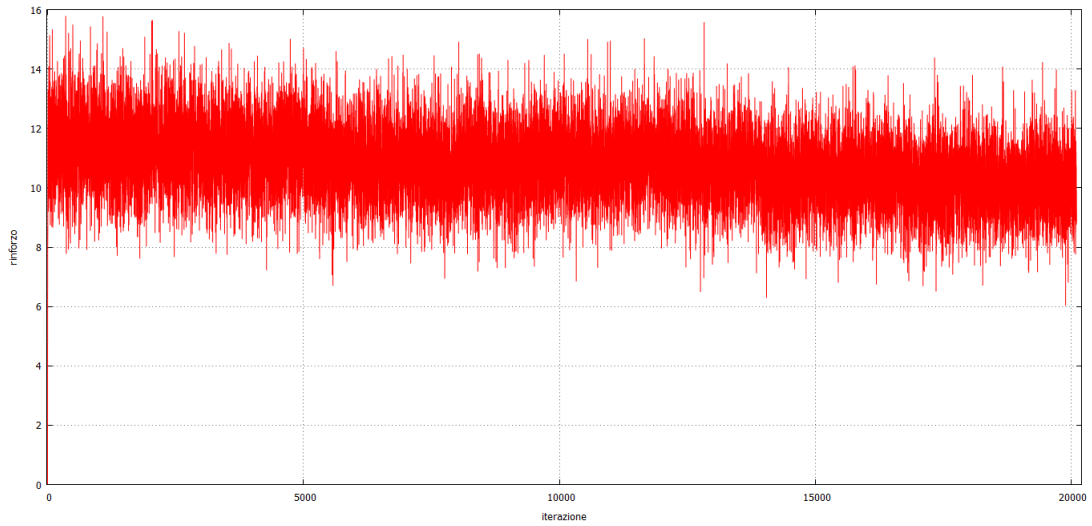


(a) Rinforzo

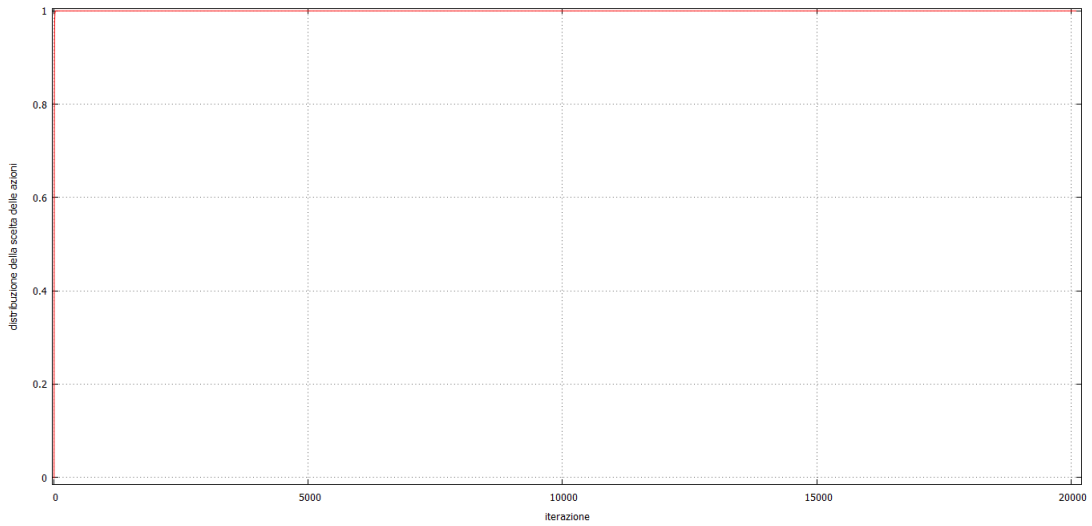


(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.21: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo.



(a) Rinforzo



(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.22: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor, al variare di α per il livello superiore; l'insieme di feature utilizzato è quello completo.

(a) Risultati ottenuti con traffico leggero

	downpeak	interfloor
tempo di attesa medio (s)	16,4	22,171
tempo di attesa massimo (s)	80,787	142,227
tempo di viaggio medio (s)	26,562	28,989
tempo di sistema medio (s)	42,962	51,16
numero di fermate degli ascensori	1302,998	1398,74

(b) Risultati ottenuti con traffico intenso

	downpeak	interfloor
tempo di attesa medio (s)	29,358	75,743
tempo di attesa massimo (s)	184,672	273,143
tempo di viaggio medio (s)	38,797	58,164
tempo di sistema medio (s)	68,155	133,908
numero di fermate degli ascensori	957,004	1052,284

Tabella 5.15: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme completo delle feature e effettuando un numero elevato di simulazioni in situazione di traffico leggero.

non si comporta altrettanto bene e le prestazioni risultano essere non accettabili. Questo, probabilmente, è dovuto alla particolarità della politica appresa al livello con ascensori pieni, che prevede di fermarsi sempre alle richieste intermedie; questo, comunque, non è un problema particolarmente grave, perché il controllore, essendo adattativo, può modificare la propria politica per renderla performante anche nelle mutate condizioni.

Risultati con insieme delle feature ridotto attraverso il metodo basato su alberi

Vediamo ora i risultati dell'apprendimento tramite gradiente effettuato utilizzando l'insieme delle feature ridotto col metodo basato su alberi, sia per quanto riguarda il livello con ascensori pieni, sia per quello con ascensori vuoti; al livello con ascensori pieni abbiamo deciso di utilizzare il rinforzo personalizzato, dato che nelle prove con l'insieme completo delle feature si è rivelato migliore di quello generale. La configurazione dell'algoritmo che abbiamo utilizzato è simile a quella utilizzata nell'apprendimento con tutte le feature: abbiamo, infatti, effettuato simulazioni da 15 minuti con aggiornamento dei parametri della politica ogni 450 simulazioni al livello gerarchico superiore, ogni 15 simulazioni al livello gerarchico inferiore; il numero di run effettuati è variabile e dipende dalla mancanza di un cambiamento apprezzabile nelle prestazioni del controllore, anche in seguito a degli aggiornamenti della poli-

tica. Tutti gli aggiornamenti dei parametri sono stati fatti con normalizzazione del gradiente.

A differenza di quanto fatto per l'insieme completo delle feature, non abbiamo considerato valori differenti per α, ma lo abbiamo fissato a 0,02 a entrambi i livelli.

Anche in questo caso il metodo di stima del gradiente che abbiamo utilizzato è GPOMDP.

Per scontare il rinforzo ottenuto al livello con ascensori vuoti abbiamo utilizzato $\beta = 10^{-4}$ nel calcolo del tempo di attesa medio quadratico, mentre non abbiamo scontato (ponendo $\beta = 0$) il rinforzo ottenuto al livello con ascensori pieni; di conseguenza, nell'algoritmo di stima tramite gradiente abbiamo lasciato $\gamma = 1$ per entrambi i livelli gerarchici.

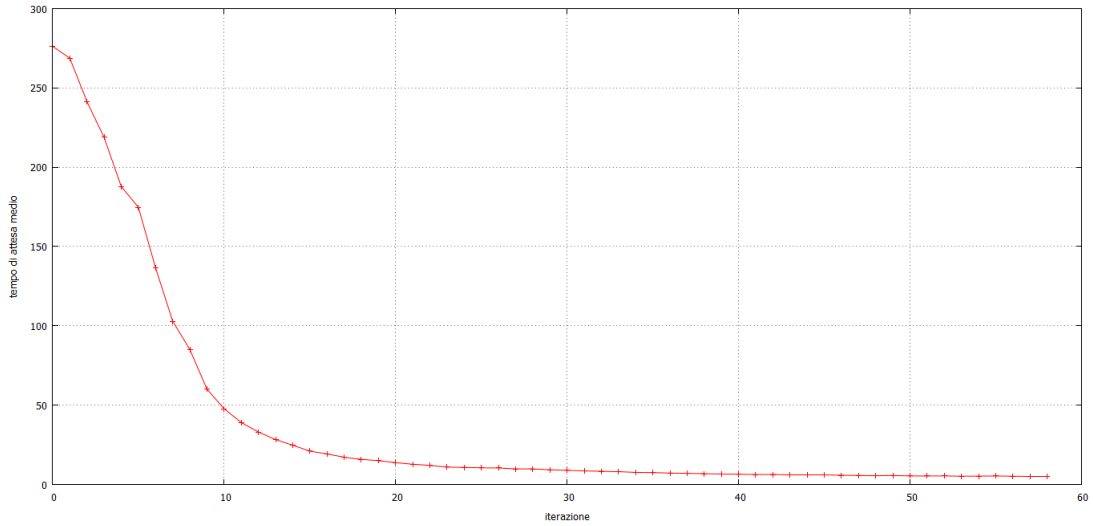
L'insieme di feature considerato contiene anche una variabile che indica la tipologia di traffico in corso. La politica Gibbs con basis lineare, però, non ha una capacità tale da poter creare delle politiche indipendenti a seconda del valore di questa feature; di conseguenza, abbiamo deciso di effettuare prove di apprendimento separate per tipologia di traffico e, quindi, i controllori appresi sono più di uno, ognuno specializzato per una singola tipologia di traffico, esattamente come abbiamo fatto nell'apprendimento con l'insieme completo di feature.

Traffico leggero. Le Figure 5.23, 5.24 e 5.25 riportano le curve di apprendimento della politica al livello con ascensori vuoti, mostrando come variano il tempo di attesa medio, il tempo di attesa massimo e il rinforzo ottenuto dagli agenti a ogni run[12]; similmente, le Figure 5.26 e 5.27 riportano le curve di apprendimento della politica al livello con ascensori pieni, mostrando come variano il rinforzo assegnato agli agenti a ogni run e con che rapporto medio viene preferita una delle due azioni possibili rispetto all'altra. Anche in questo caso, per le stesse motivazioni precedenti, per il livello più basso non abbiamo riportato i risultati ottenuti in uppeak.

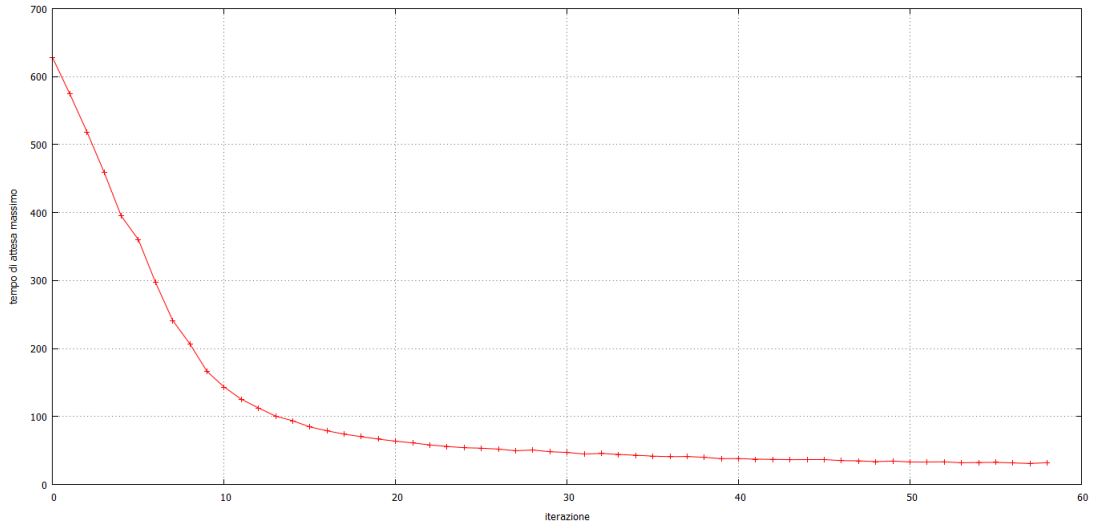
In Tabella 5.15(a) abbiamo riportato i risultati ottenuti dal controllore appreso mediati su 500 simulazioni di 60 minuti con traffico leggero.

Dai dati emerge che le prestazioni del controllore sono simili a quelle ottenute dal controllore appreso in analoghe condizioni con l'insieme completo delle feature e utilizzando il rinforzo generale per il livello con ascensori pieni; in particolare, i tempi di attesa in uppeak risultano essere leggermente più elevati, quelli in interfloor simili, mentre quelli in downpeak leggermente migliori. Risultano, invece, più lunghi

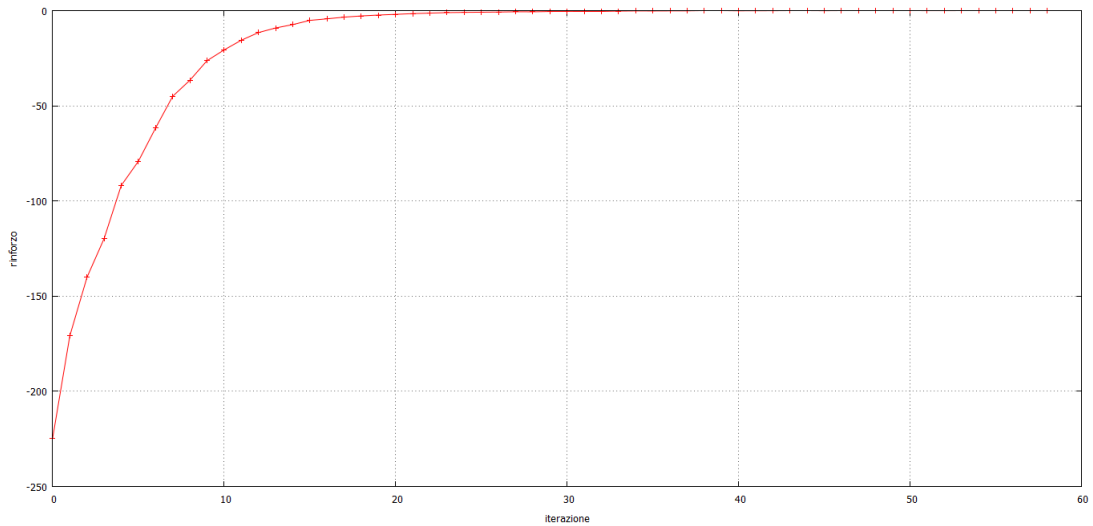
[12] Il lettore attento potrebbe notare che il punto di partenza delle curve è in alcuni casi molto differente rispetto a quello di partenza nel caso in cui abbiamo utilizzato tutte le feature, mentre dovrebbero essere abbastanza simili dato che in entrambi i casi partiamo con una politica casuale. Questa discrepanza è dovuta al fatto che in queste prove, per avere una maggiore rapidità, abbiamo interrotto le simulazioni 10 minuti (invece che 30) dopo l'ultima richiesta, con l'effetto che con le prime politiche il sistema non riesce a esaurire tutte le richieste presenti e, quindi, i dati ottenuti sono troncati prima che vengano effettivamente utilizzati; questo non creare nessun tipo di problema all'apprendimento, che semplicemente utilizza qualche dato in meno.



(a) Tempo di attesa medio (secondi)

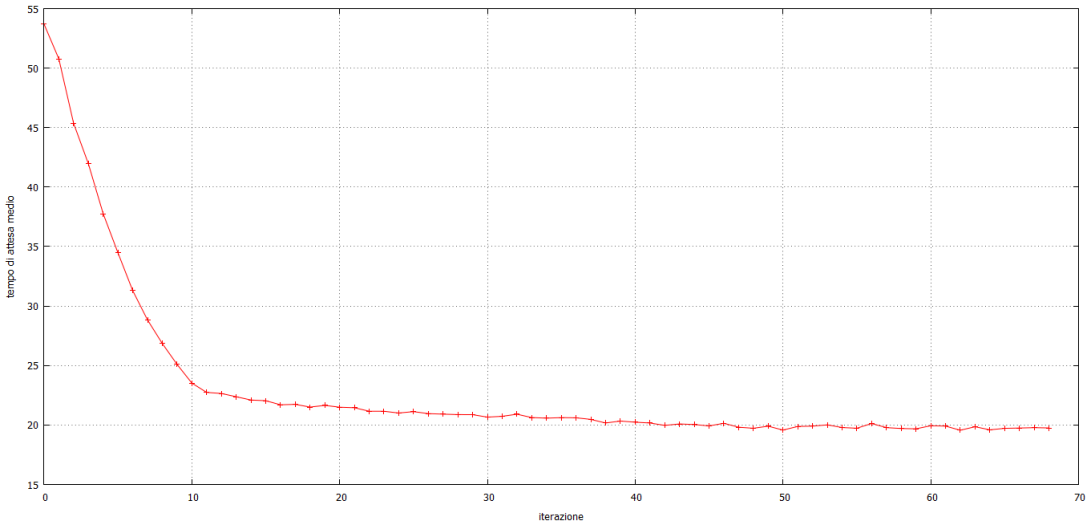


(b) Tempo di attesa massimo (secondi)

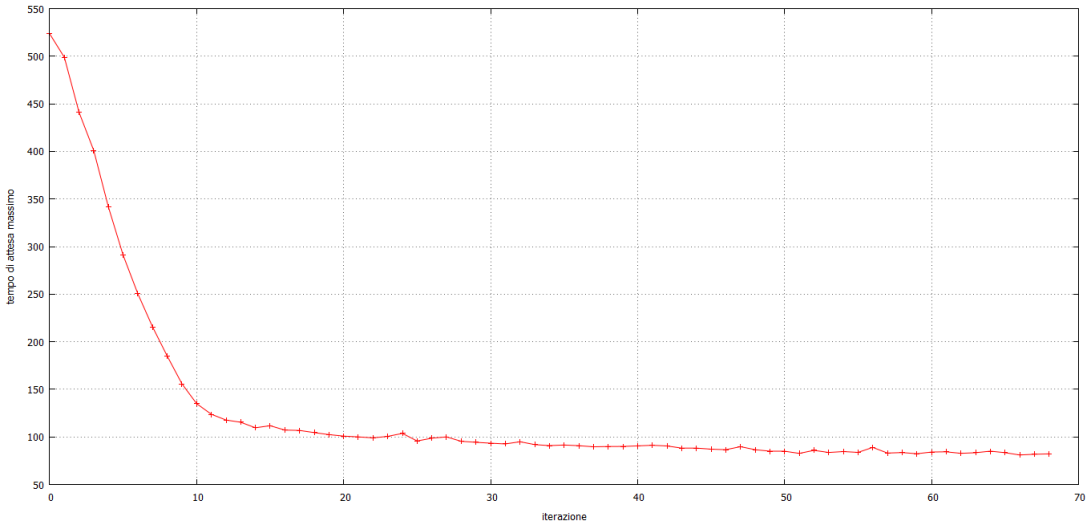


(c) Rinforzo

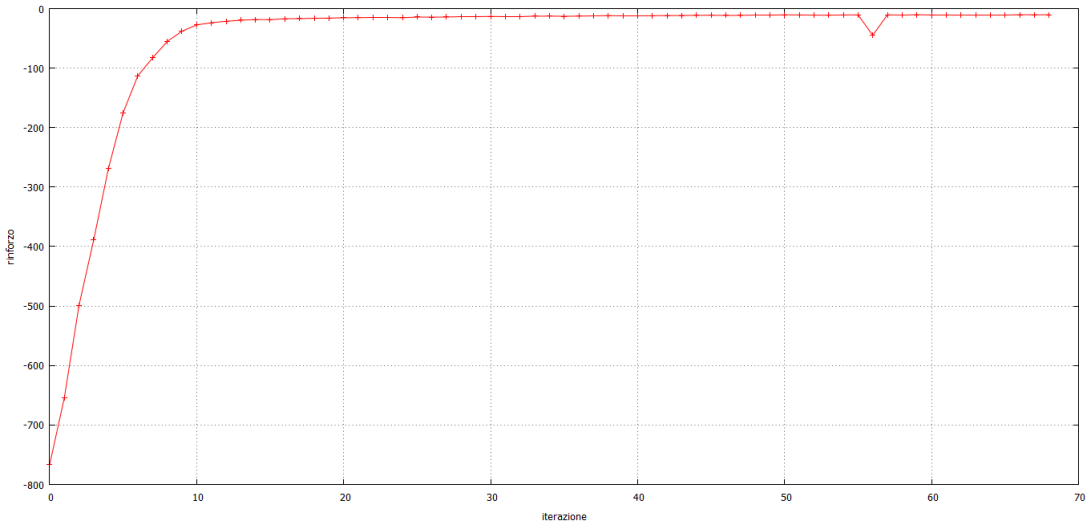
Figura 5.23: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo uppeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Tempo di attesa medio (secondi)

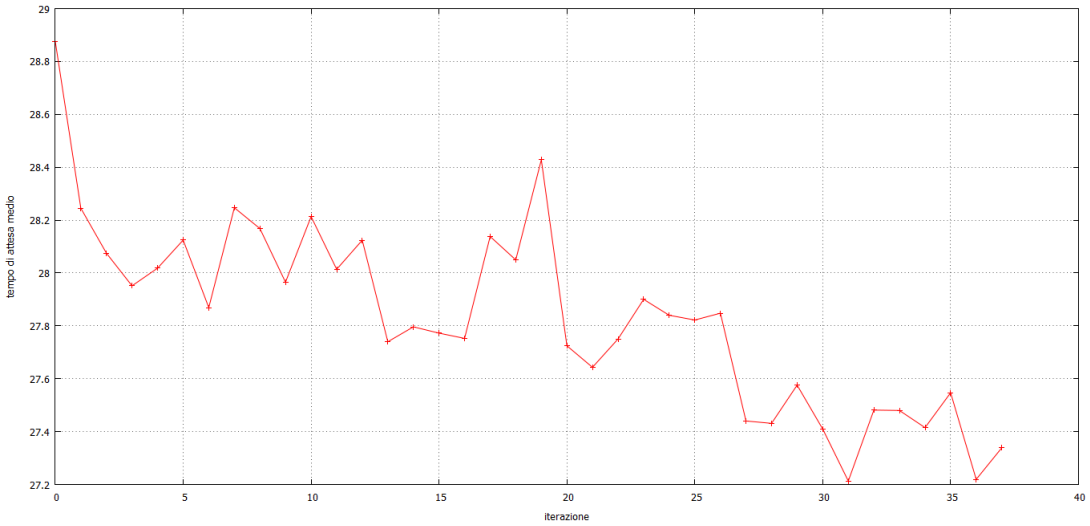


(b) Tempo di attesa massimo (secondi)

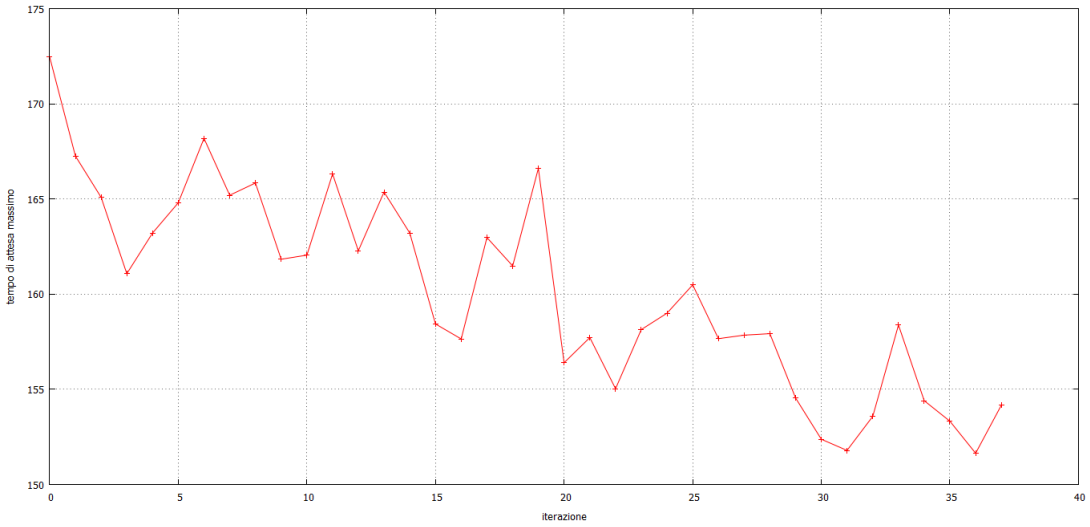


(c) Rinforzo

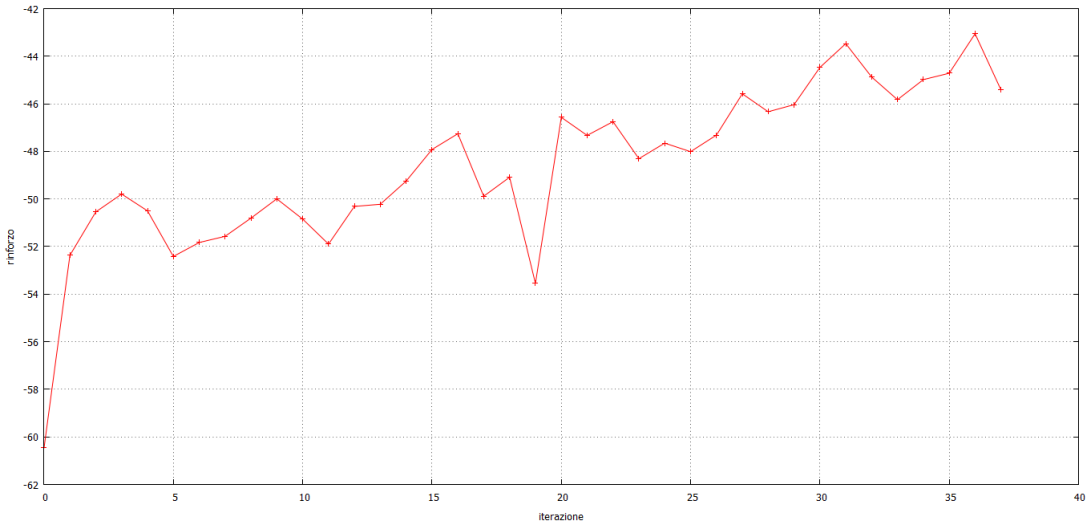
Figura 5.24: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Tempo di attesa medio (secondi)

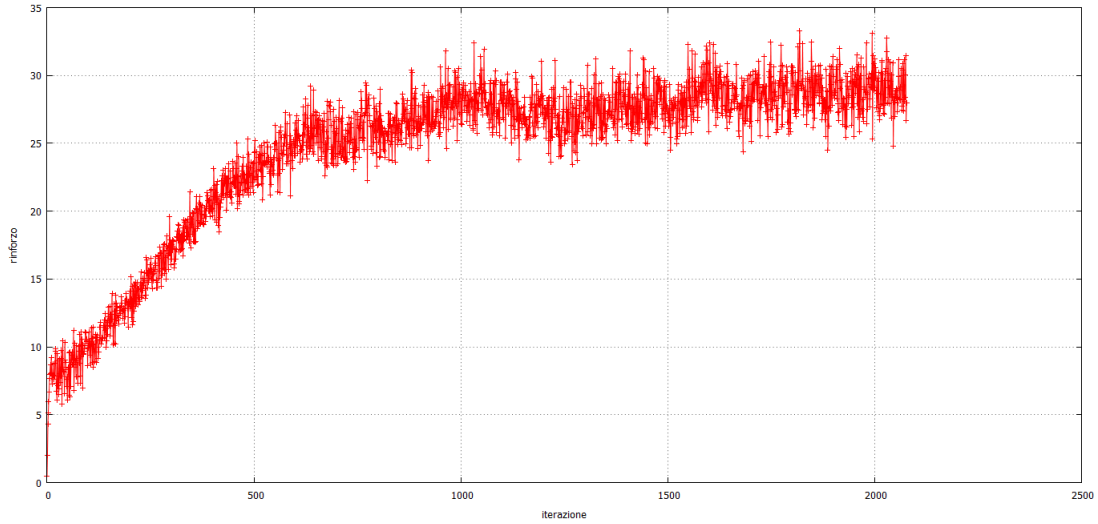


(b) Tempo di attesa massimo (secondi)

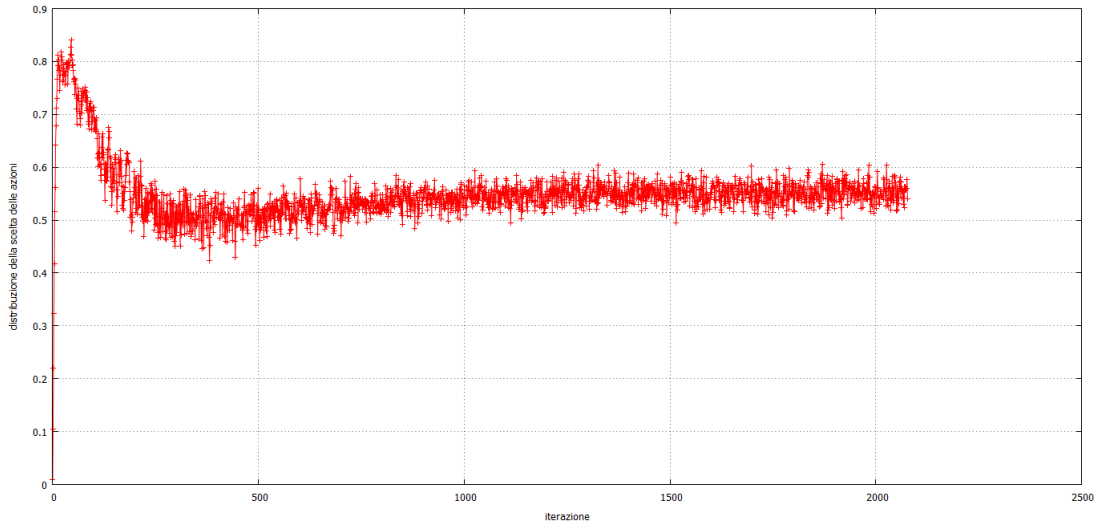


(c) Rinforzo

Figura 5.25: Andamento dell'apprendimento al livello più alto ottenuto con traffico leggero di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.

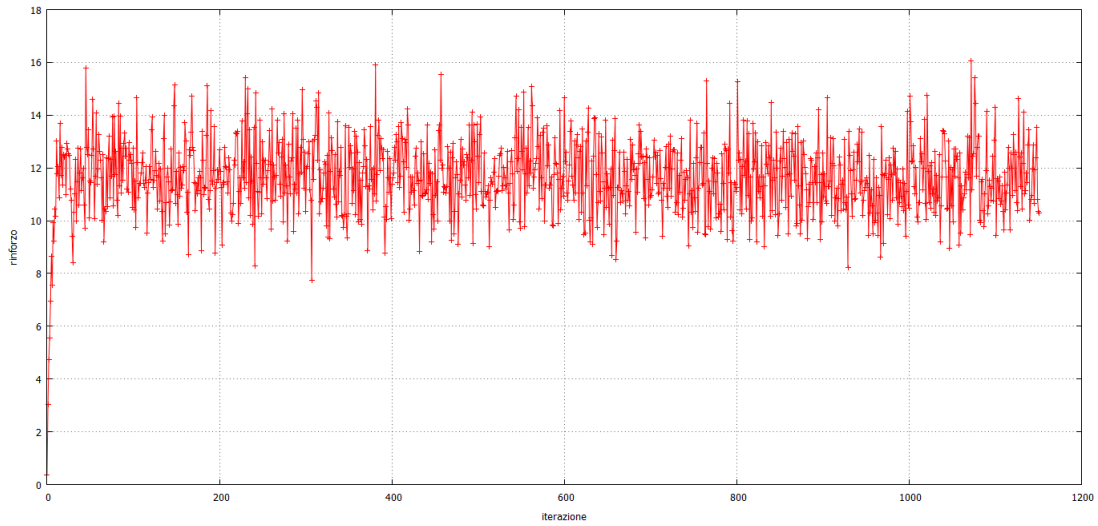


(a) Rinforzo

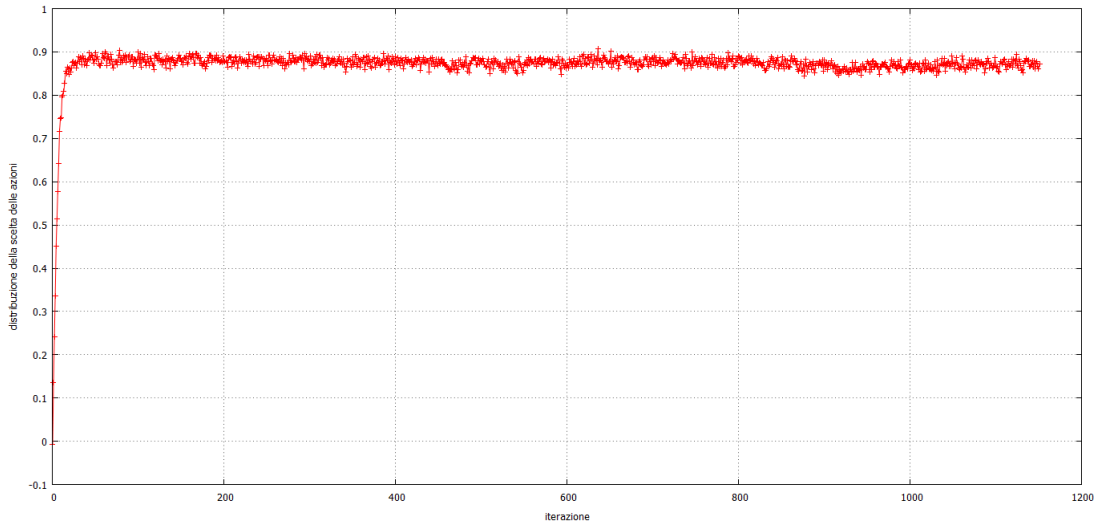


(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.26: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Rinforzo



(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.27: Andamento dell'apprendimento al livello più basso ottenuto con traffico leggero di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	5,063	19,782	27,188
tempo di attesa massimo (s)	38,843	99,225	190,920
tempo di viaggio medio (s)	29,643	28,972	30,018
tempo di sistema medio (s)	34,706	48,753	57,206
numero di fermate degli ascensori	946,768	1249,01	1382,6

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	35,340	35,464	79,103
tempo di attesa massimo (s)	121,747	215,113	279,686
tempo di viaggio medio (s)	56,542	39,674	57,645
tempo di sistema medio (s)	91,882	75,138	136,743
numero di fermate degli ascensori	989,616	901,516	1050,892

Tabella 5.16: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto attraverso il metodo basato su alberi in situazione di traffico leggero.

i tempi di viaggio, a fronte però di un notevole risparmio energetico, come testimoniato dal minor numero di fermate compiuto dagli ascensori; questa differenza è spiegabile anche dall'utilizzo di un rinforzo differente al livello con ascensori pieni, che porta all'apprendimento di una politica differente per questo livello. I risultati, invece, risultano essere inferiori, anche se non di molto, a quelli appresi in analoghe condizioni con l'insieme completo delle feature e utilizzando il rinforzo personalizzato per il livello con ascensori pieni. Questo è spiegabile certamente con il minor numero di informazioni concesse agli agenti, ma anche con il minor numero di iterazioni dell'algoritmo di apprendimento effettuate; la differenza, infatti, risulta essere molto più ridotta se confrontiamo le curve di apprendimento a parità di numero di iterazioni.

Qualitativamente, il comportamento che mantengono gli ascensori risulta essere simile a quello che mantengono utilizzando la politica appresa con l'insieme completo delle feature.

Possiamo, quindi, concludere che la riduzione delle feature ha portato a una notevole diminuzione della dimensionalità del problema, rendendo l'apprendimento più semplice e rapido, senza però pregiudicare eccessivamente le capacità di apprendimento attraverso metodi gradiente. Si tenga presente, comunque, che l'assenza di informazioni specifiche sui singoli piani rende intrinsicamente impossibile l'attuazione di una strategia migliore e difficilmente si può riuscire a fare meglio di quanto appreso a questo stadio; con l'insieme completo delle feature, invece, anche se con i metodi gradiente si arriva a una politica che in pratica non sfrutta in modo significativo le informazioni sui singoli piani, rimane possibile attuare una politica più

efficiente che sfrutti queste informazioni.

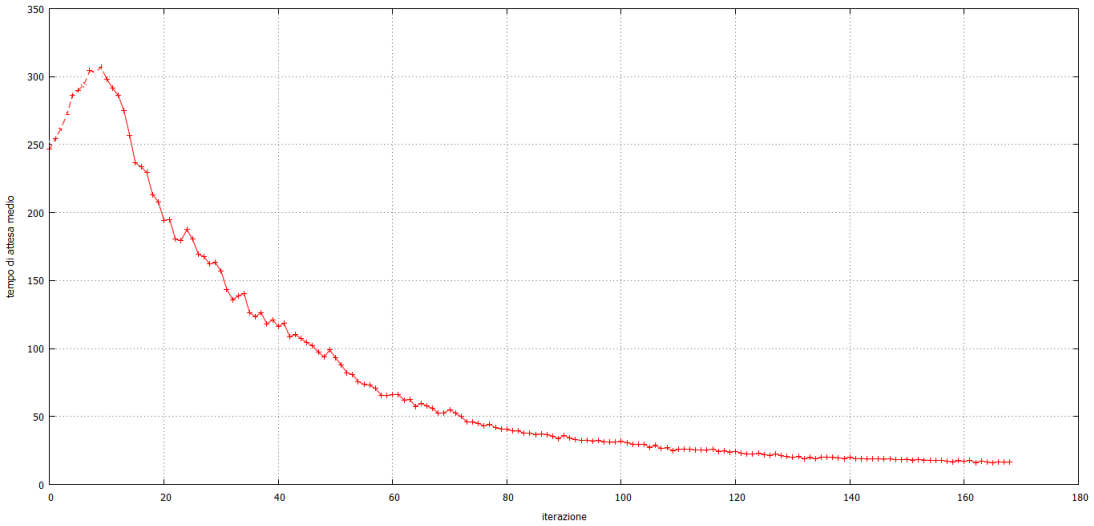
Analizzando i dati riportati in Tabella 5.15(b), invece, è possibile notare che la generalizzazione della politica a situazioni di traffico intenso avviene in modo simile a quanto si ha con l'insieme completo delle feature, se si utilizza il rinforzo personalizzato per il livello con ascensori pieni; le prestazioni in interfloor, infatti, a differenza di quanto accade con le altre tipologie di traffico risultano essere abbastanza scadenti in questo scenario. Questo fatto, comunque, non rappresenta un problema primario, in quanto l'utilizzo di un controllore adattativo permette di modificare la politica appresa per seguire i cambiamenti delle condizioni esterne; inoltre, le prestazioni in interfloor sono quelle che contribuiscono in maniera minore alla qualità del servizio offerto in quanto, come emerge dalla Figura 2.1, l'intensità del traffico in interfloor in una tipica giornata lavorativa è molto minore di quella in uppeak e downpeak.

Traffico intenso. Per l'apprendimento con traffico intenso abbiamo deciso di utilizzare la stessa configurazione utilizzata con traffico leggero. In Figura 5.28, 5.29 e 5.30 abbiamo riportato le curve di apprendimento della politica al livello con ascensori vuoti, mostrando come variano il tempo di attesa medio, il tempo di attesa massimo e il rinforzo ottenuto dagli agenti a ogni run, mentre in Figura 5.26 e 5.27 abbiamo riportato le curve di apprendimento della politica al livello con ascensori pieni, mostrando come variano il rinforzo assegnato agli agenti a ogni run e con che rapporto medio viene preferita una delle due azioni possibili rispetto all'altra.

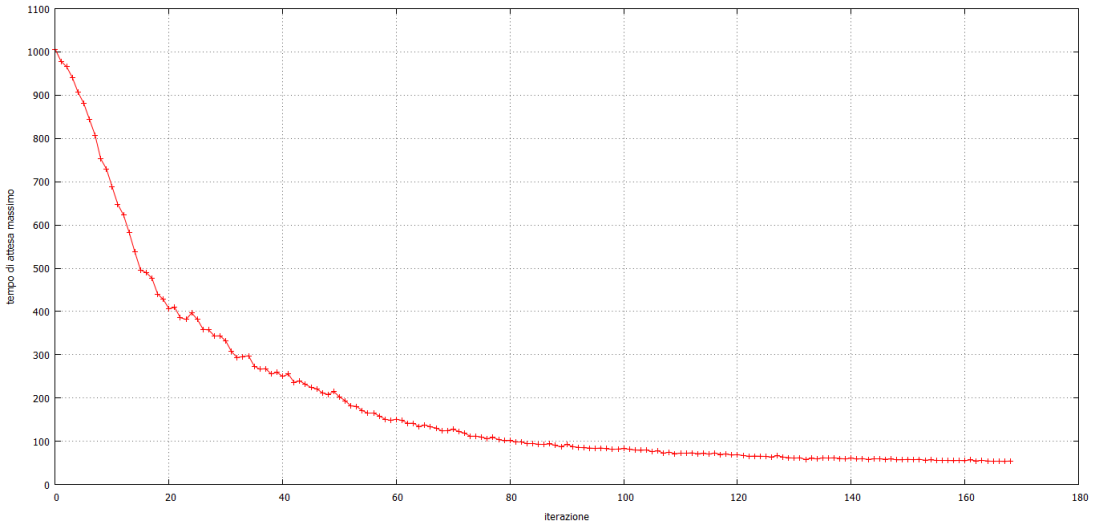
Dalle curve emerge che, mentre in downpeak e in interfoor non si hanno grandi differenze rispetto agli altri casi considerati, l'apprendimento in uppeak risulta essere molto più difficoltoso; come si può notare, infatti, il numero di iterazioni necessarie per arrivare a convergenza è significativamente più lungo di quello sperimentato in tutti gli altri casi. Il fatto, invece, che il tempo di attesa medio in uppeak inizialmente aumenta e poi comincia a diminuire, in contrasto con quanto succede per il rinforzo, non è significativo. All'inizio, infatti, la politica non riesce a servire tutti i passeggeri entro 10 minuti dal loro arrivo e, quindi, alcuni utenti, non essendo serviti, contribuiscono con un non significativo tempo di attesa nullo; quando la politica inizia a migliorare, invece, il sistema riesce a servire tutti gli utenti e i tempi di attesa medi diventano significativi, creando l'impressione di iniziale peggioramento delle prestazioni[13].

In Tabella 5.17 abbiamo riportato i risultati ottenuti dal controllore RLG gerarchico che utilizza l'insieme ridotto di feature, mediati su 500 simulazioni da 60 minuti. Confrontando questi risultati con quelli ottenuti utilizzando l'insieme completo delle feature e il rinforzo generale per il livello con ascensori pieni possiamo notare che i risultati ottenuti, per quanto riguarda i tempi di attesa, sono abbastanza simili; le altre quantità, invece, hanno delle variazioni significative (tempo di viaggio più

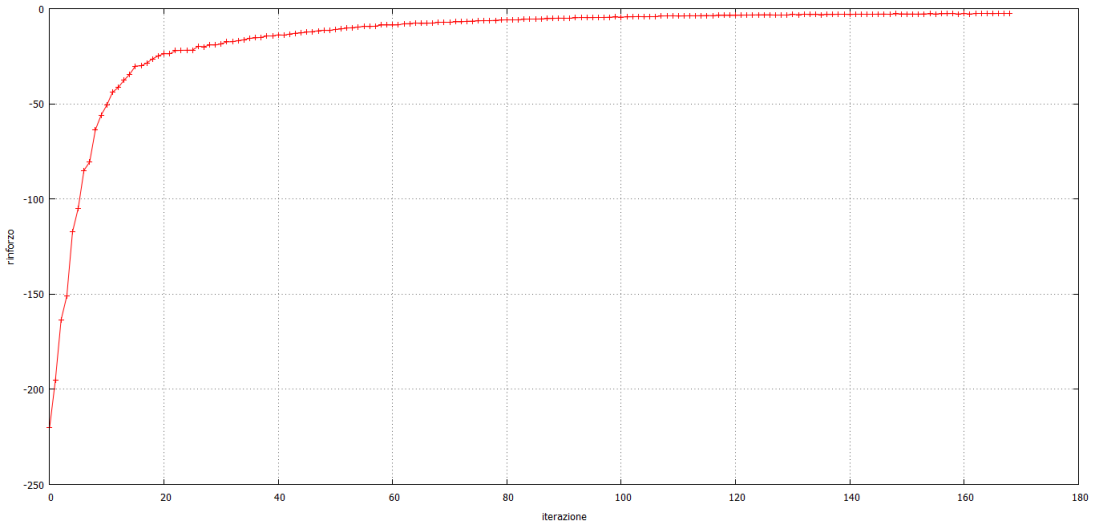
[13] Abbiamo deciso di tratteggiare la prima parte della curva in Figura 5.28(a) per sottolineare la sua poca significatività.



(a) Tempo di attesa medio (secondi)

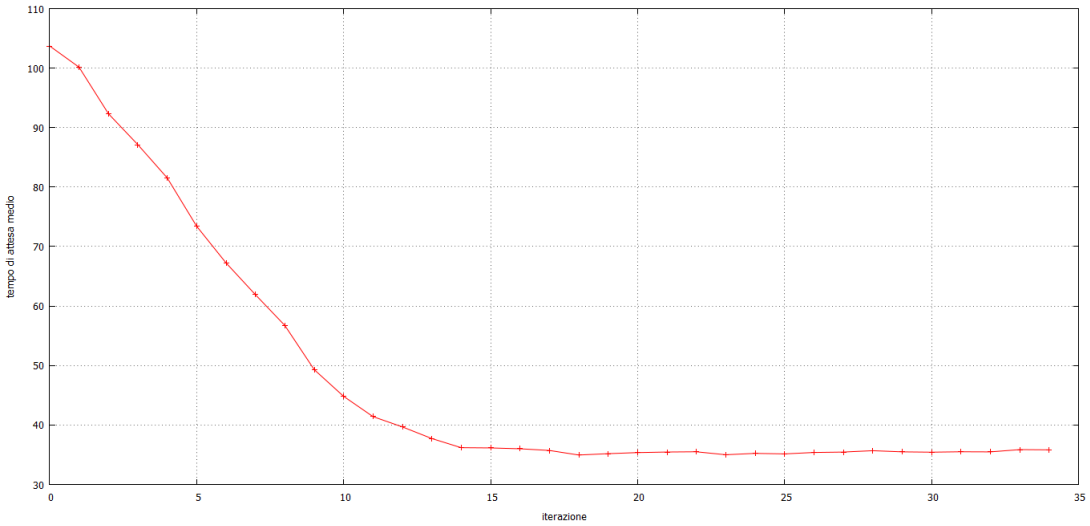


(b) Tempo di attesa massimo (secondi)

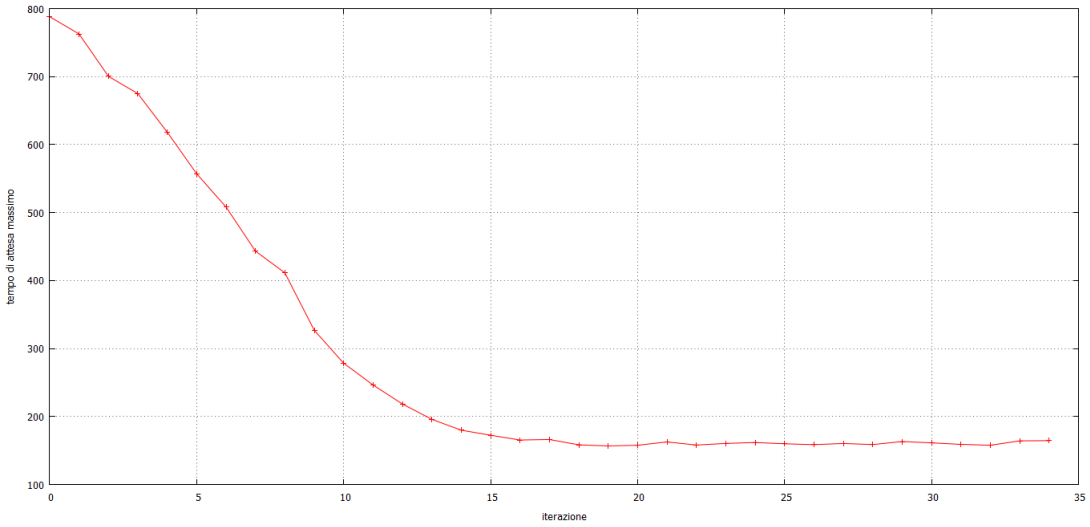


(c) Rinforzo

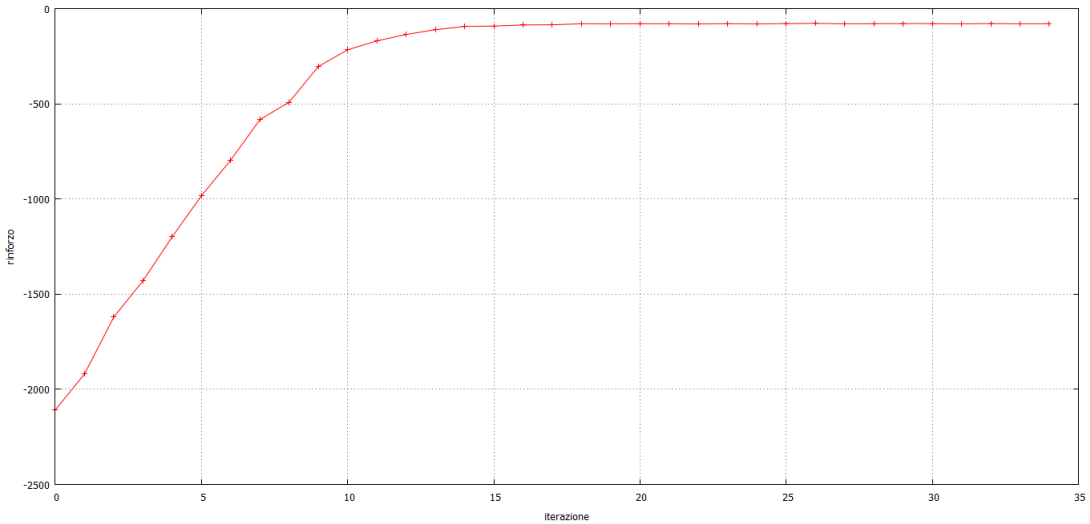
Figura 5.28: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo uppeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Tempo di attesa medio (secondi)

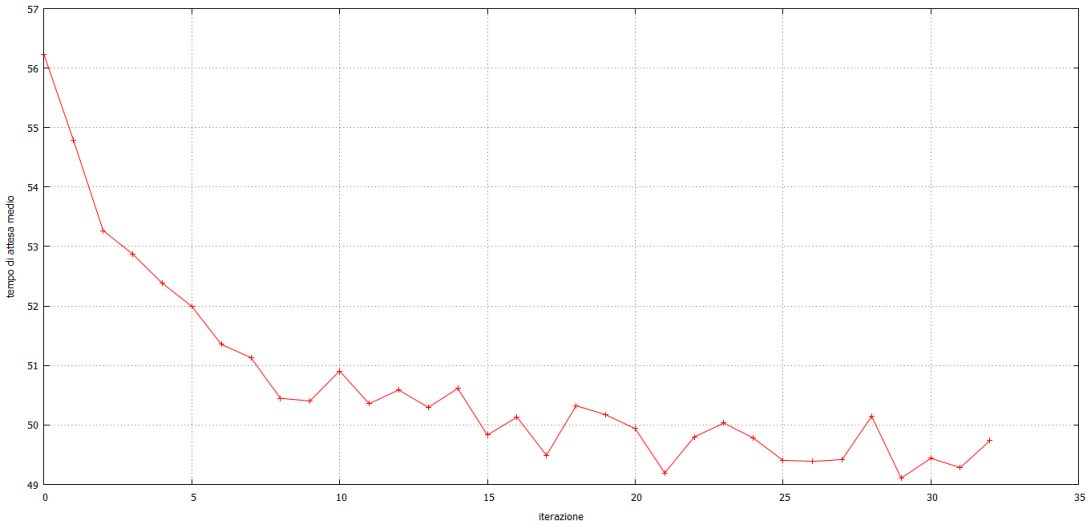


(b) Tempo di attesa massimo (secondi)

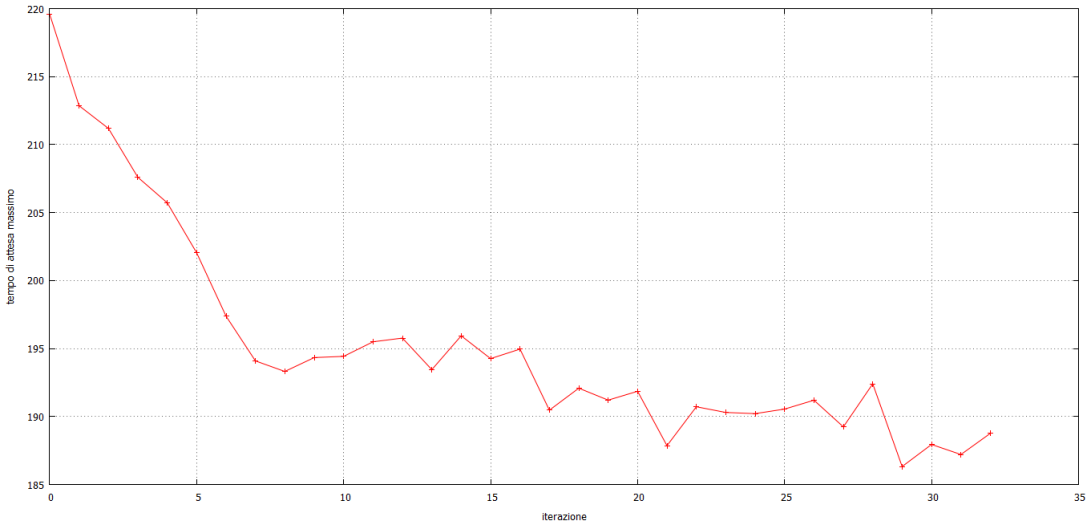


(c) Rinforzo

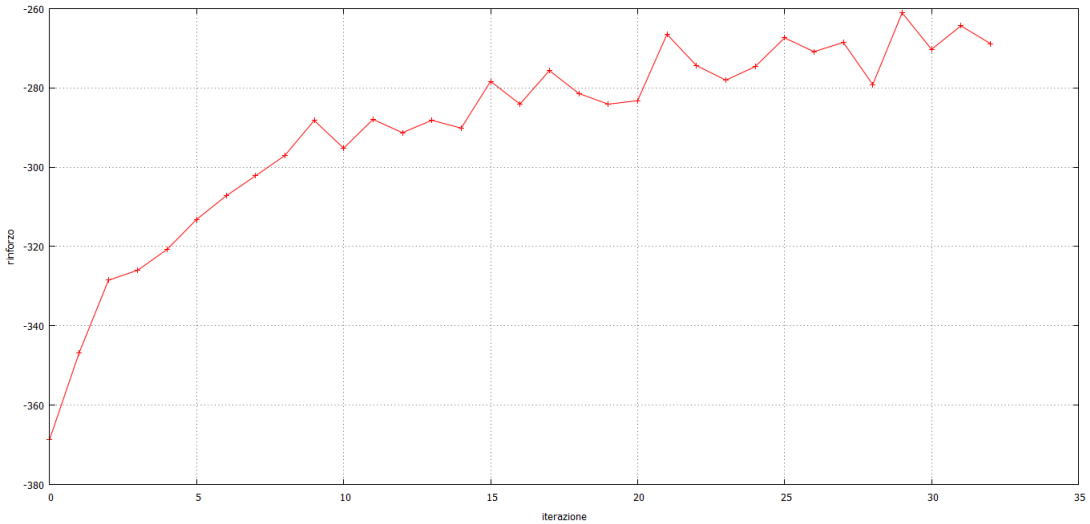
Figura 5.29: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Tempo di attesa medio (secondi)

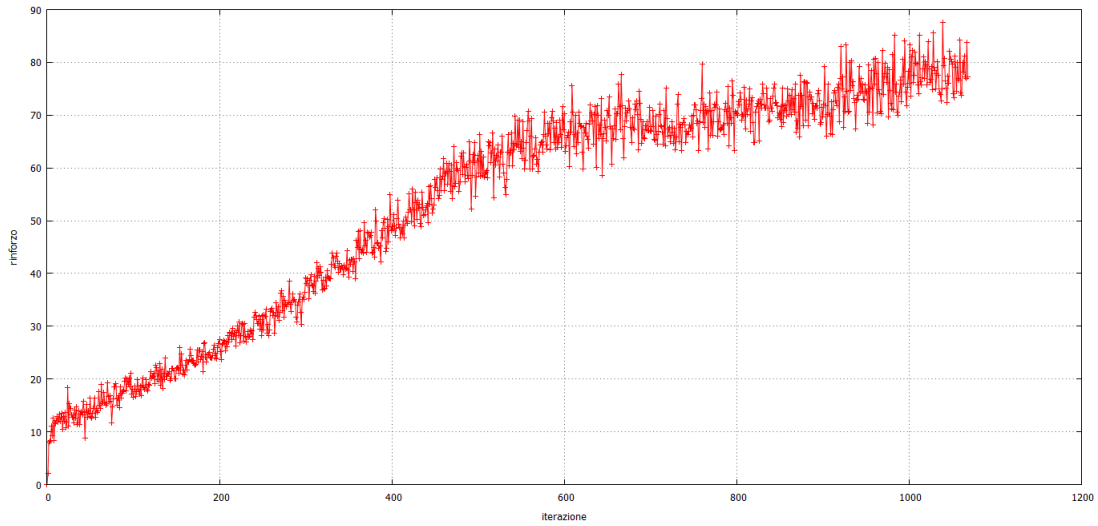


(b) Tempo di attesa massimo (secondi)

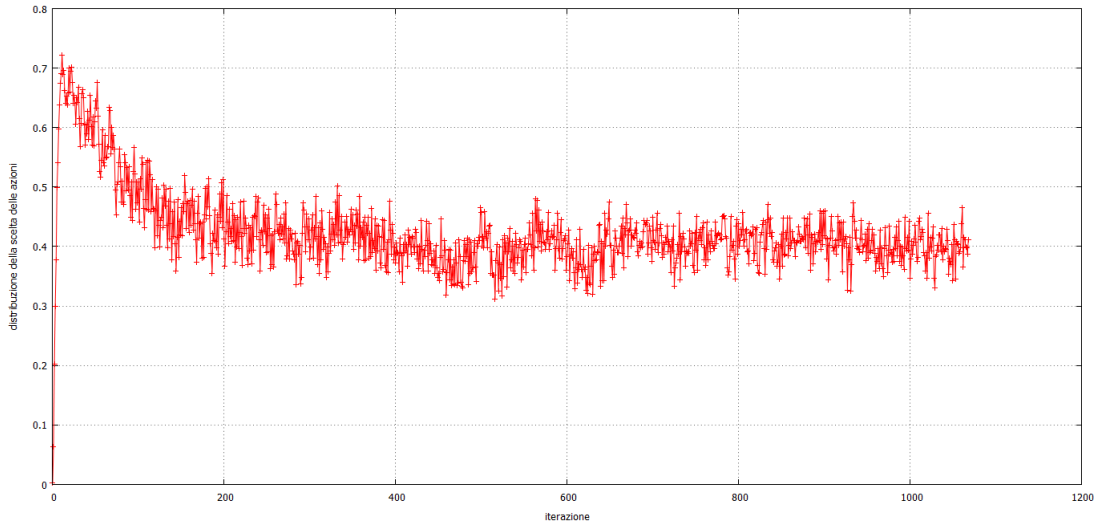


(c) Rinforzo

Figura 5.30: Andamento dell'apprendimento al livello più alto ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.

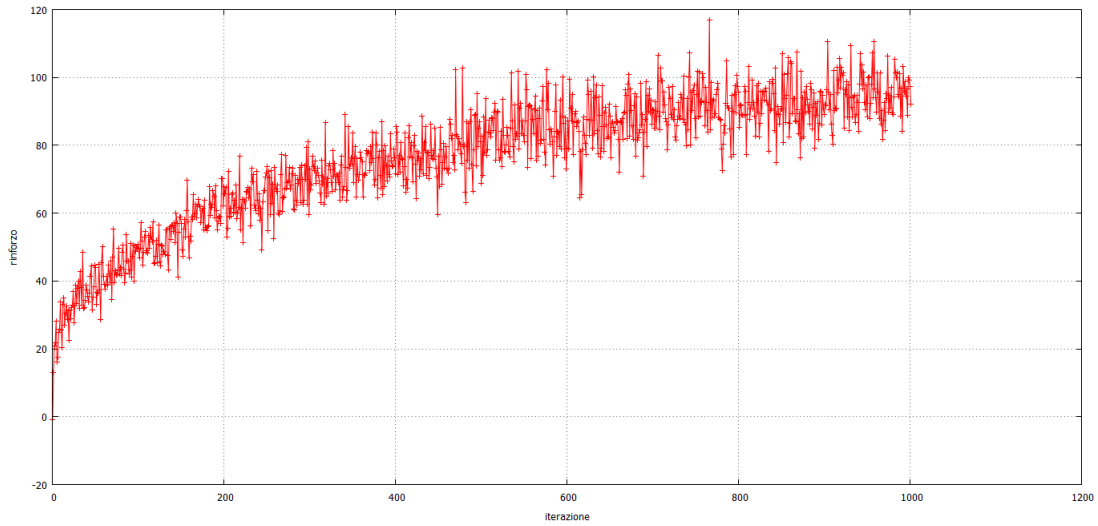


(a) Rinforzo

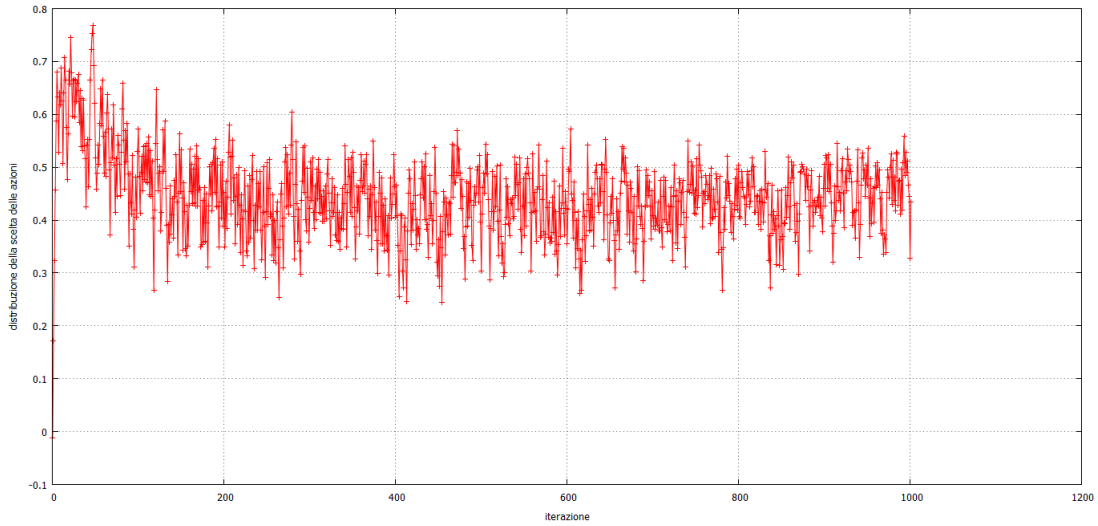


(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.31: Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo downpeak; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.



(a) Rinforzo



(b) Rapparto medio tra la scelta di fermarsi al piano intermedio (valori positivi) e quella di non fermarsi (valori negativi).

Figura 5.32: Andamento dell'apprendimento al livello più basso ottenuto con traffico intenso di tipo interfloor; l'insieme di feature utilizzato è quello ridotto attraverso il metodo basato su alberi.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	4,671	22,819	28,487
tempo di attesa massimo (s)	30,279	133,343	217,931
tempo di viaggio medio (s)	34,539	29,629	28,427
tempo di sistema medio (s)	39,21	52,448	56,913
numero di fermate degli ascensori	1329,85	1213,61	1374,196

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	24,346	38,094	54,669
tempo di attesa massimo (s)	85,859	227,771	228,592
tempo di viaggio medio (s)	57,246	39,323	54,225
tempo di sistema medio (s)	81,592	77,417	108,894
numero di fermate degli ascensori	1188,964	895,786	1070,506

Tabella 5.17: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto attraverso il metodo basato su alberi in situazione di traffico intenso.

lungo, numero di fermate inferiori), a causa anche del differente rinforzo usato al livello con ascensori pieni.

In questo caso, inoltre, possiamo notare che si ha una buona generalizzazione della politica appresa in situazione di traffico intenso al controllo con traffico leggero.

Risultati con un controllore unico

I controllori RLG analizzati fino a ora sono tutti specializzati per tipologia di traffico, mentre tutti gli altri controllori analizzati in questa tesi sono unici e devono funzionare ugualmente bene in tutte le tipologie di traffico. Di conseguenza, ora proviamo ad apprendere anche un controllore RLG unico, per verificare se in questo modo si ha o meno una perdita di prestazioni. Le condizioni per l'apprendimento sono le solite. Abbiamo, infatti, effettuato simulazioni da 15 minuti con aggiornamento dei parametri della politica ogni 450 simulazioni al livello gerarchico superiore, ogni 15 simulazioni al livello gerarchico inferiore; il numero di run effettuati è variabile e dipende dalla mancanza di un cambiamento apprezzabile nelle prestazioni del controllore, anche in seguito a degli aggiornamenti della politica. Tutti gli aggiornamenti dei parametri sono stati fatti con normalizzazione del gradiente stimato attraverso GPOMDP e α=0,02. Per scontare il rinforzo ottenuto al livello con ascensori vuoti abbiamo utilizzato $\beta = 10^{-4}$ nel calcolo del tempo di attesa medio quadratico, mentre non abbiamo scontato (ponendo $\beta = 0$) il rinforzo personalizzato ottenuto al livello con ascensori pieni; di conseguenza, nell'algoritmo di stima tramite gradiente abbiamo lasciato $\gamma = 1$ per entrambi i livelli gerarchici.

Per ottenere l'apprendimento di un controllore unico abbiamo alternato durante l'apprendimento simulazioni con traffico uppeak, downpeak e interfloor, cambiando tipologia a ogni simulazione. Per contrastare in qualche modo le difficoltà di apprendimento di un controllore unico, inoltre, abbiamo inserito nell'insieme di feature anche una variabile che indichi la tipologia di traffico in corso, similmente a quanto abbiamo fatto per il controllore FQI.

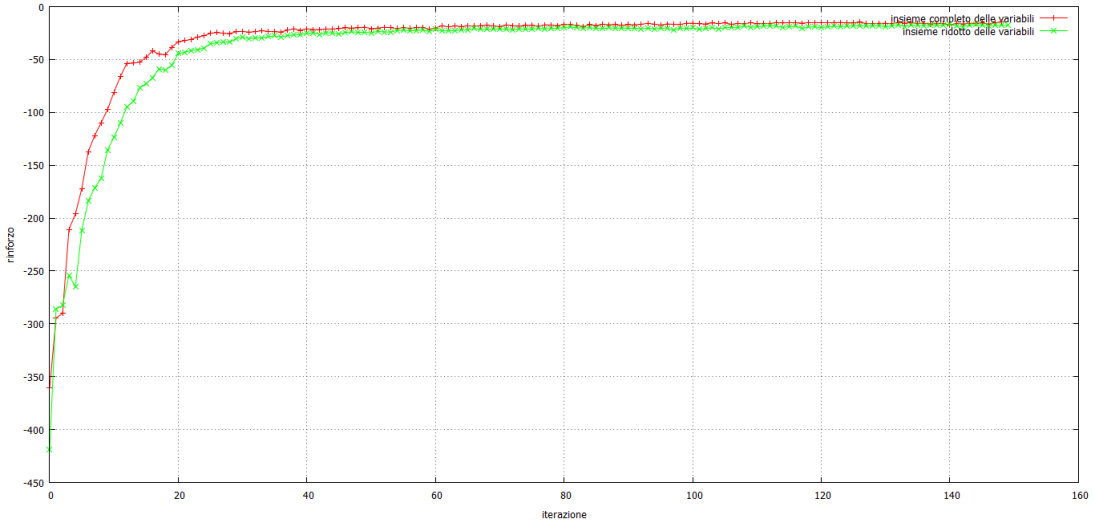
L'andamento dell'apprendimento in condizioni di traffico leggero è riportato in Figura 5.33, mentre quello in condizioni di traffico intenso è riportato in Figura 5.34. A differenza degli altri apprendimenti effettuati in questo caso riportiamo solamente le curve di apprendimento del rinforzo ai due livelli gerarchici, in quanto l'apprendimento contemporaneo nelle diverse tipologie di traffico rende le altre quantità poco significative. Abbiamo, inoltre, riportato negli stessi grafici l'apprendimento con l'insieme completo e quello ridotto delle feature, in modo da poter effettuare un confronto più diretto. Infine, abbiamo effettuato apprendimenti più lunghi in condizioni di traffico intenso rispetto a quelli con traffico leggero perché nel primo caso l'apprendimento si è rilevato, almeno in apparenza, più difficoltoso.

Dall'analisi dei grafici possiamo concludere che l'utilizzo dell'insieme ridotto di feature non ha particolari effetti sulle capacità di apprendimento degli agenti, ma semplicemente rende necessario un numero leggermente maggiore di iterazioni dell'algoritmo per raggiungere una situazione vicina alla convergenza. Possiamo, inoltre, notare che anche in questo caso l'apprendimento arriva a una condizione abbastanza stabile in un numero relativamente ridotto di iterazioni, poi prosegue con miglioramenti molto rumorosi e lenti.

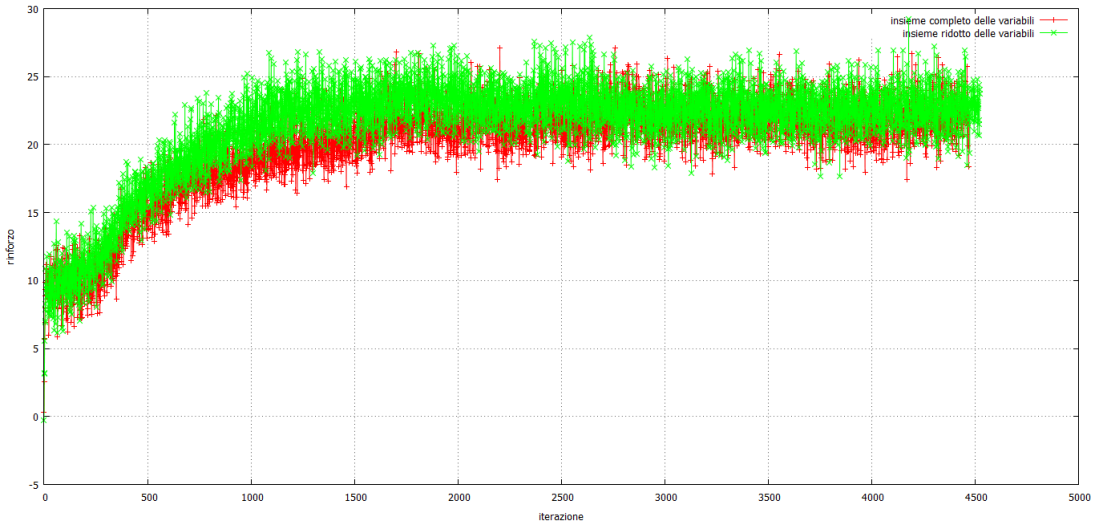
I risultati numerici mediati su 500 simulazioni da 60 minuti l'una ottenuti con i diversi controllori appresi sono riportati nelle Tabelle 5.18, 5.19, 5.20 e 5.21. Analizzando questi dati e confrontandoli con quelli ottenuti con apprendimento specializzato per tipologia di traffico possiamo affermare che le prestazioni in downpeak e in interfloor risentono in maniera molto limitata dell'utilizzo di un controllore unico; inoltre, in queste condizioni l'utilizzo dell'insieme completo o ridotto delle feature non causa alcuna differenza degna di nota.

Le prestazioni in uppeak, invece, risentono negativamente della mancanza di una politica completamente personalizzata. Infatti, mentre i tempi di attesa con traffico leggero sono accettabili anche se più lunghi rispetto al caso con controllore specializzato, quelli con traffico intenso non sono assolutamente tollerabili. Inoltre, il controllore appreso con un numero ridotto di variabili si comporta in maniera decisamente peggiore rispetto a quello che dispone di tutte le informazioni, probabilmente perché la riduzione ha eliminato delle feature che in questo caso sono utili.

In uppeak, quindi, il calo di prestazioni dovuto all'utilizzo di un controllore unico è decisamente più significativo che in downpeak o in interfloor. Questo fatto, però, è anche abbastanza ovvio: la politica appresa dal controllore specializzato in uppeak, infatti, è molto particolare, in quanto prevede di mandare tutti gli ascensori liberi al

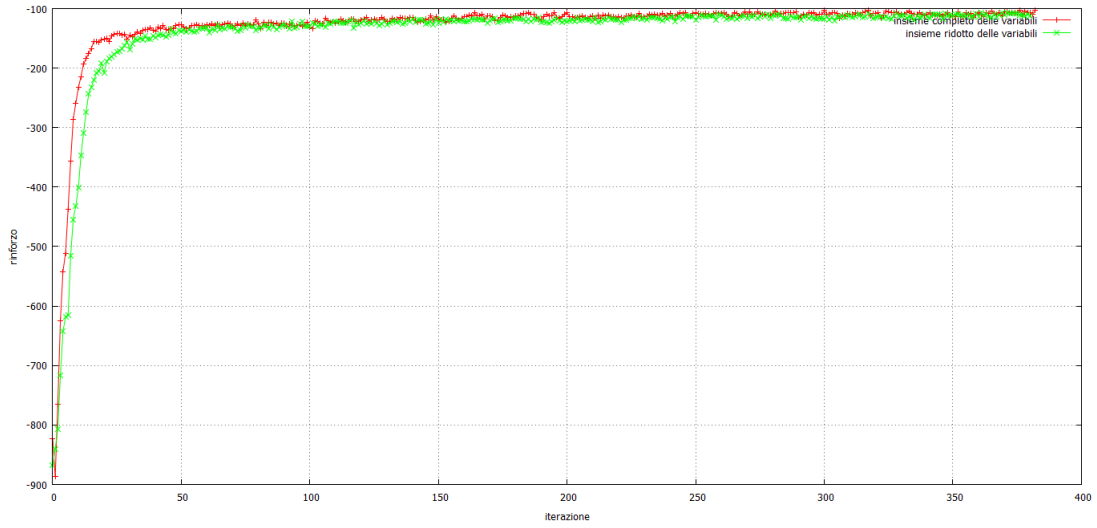


(a) Livello con ascensori vuoti.

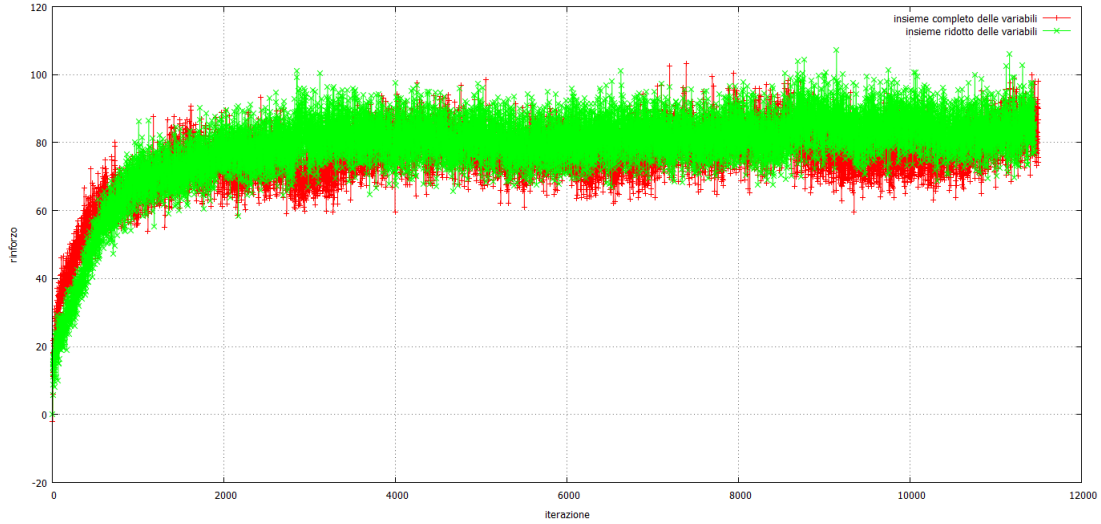


(b) Livello con ascensori pieni.

Figura 5.33: Andamento dell'apprendimento del rinforzo con traffico leggero di diverse tipologie, al variare dell'insieme delle feature utilizzato.



(a) Livello con ascensori vuoti.



(b) Livello con ascensori pieni.

Figura 5.34: Andamento dell'apprendimento del rinforzo con traffico intenso di diverse tipologie, al variare dell'insieme delle feature utilizzato.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	11,435	20,726	24,521
tempo di attesa massimo (s)	64,424	119,49	162,172
tempo di viaggio medio (s)	44,692	28,874	29,856
tempo di sistema medio (s)	56,126	49,6	54,377
numero di fermate degli ascensori	1337,362	1229,786	1334,038

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	267,381	35,051	54,663
tempo di attesa massimo (s)	565,752	196,717	229,433
tempo di viaggio medio (s)	57,043	39,25	55,857
tempo di sistema medio (s)	324,424	74,301	110,521
numero di fermate degli ascensori	1215,020	893,32	1030,166

Tabella 5.18: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature completo in situazione di traffico leggero di tutte le tipologie.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	8,98	21,239	26,977
tempo di attesa massimo (s)	52,495	135,243	197,246
tempo di viaggio medio (s)	46,529	29,058	30,569
tempo di sistema medio (s)	55,509	50,297	57,546
numero di fermate degli ascensori	1350,078	1250,98	1345,258

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	286,353	34,015	53,577
tempo di attesa massimo (s)	603,365	192,135	229,101
tempo di viaggio medio (s)	56,695	39,66	55,604
tempo di sistema medio (s)	343,047	73,676	109,181
numero di fermate degli ascensori	1227,656	909,098	1043,828

Tabella 5.19: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature completo in situazione di traffico intenso di tutte le tipologie.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	24,268	20,795	24,972
tempo di attesa massimo (s)	116,47	122,591	171,554
tempo di viaggio medio (s)	53,914	28,265	29,665
tempo di sistema medio (s)	78,182	49,061	54,638
numero di fermate degli ascensori	1368,758	1257,638	1354,274

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	529,315	35,839	54,473
tempo di attesa massimo (s)	1381,652	210,261	222,225
tempo di viaggio medio (s)	45,299	39,001	55,877
tempo di sistema medio (s)	574,614	74,842	110,35
numero di fermate degli ascensori	1296,21	901,56	1039,384

Tabella 5.20: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto in situazione di traffico leggero di tutte le tipologie.

(a) Risultati ottenuti con traffico leggero

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	27,01	21,485	25,519
tempo di attesa massimo (s)	124,066	130,064	191,229
tempo di viaggio medio (s)	55,384	28,681	29,733
tempo di sistema medio (s)	82,394	50,166	55,252
numero di fermate degli ascensori	1350,866	1232,192	1342,486

(b) Risultati ottenuti con traffico intenso

	uppeak	downpeak	interfloor
tempo di attesa medio (s)	551,628	36,099	54,937
tempo di attesa massimo (s)	1501,686	198,422	226,821
tempo di viaggio medio (s)	43,341	39,086	55,874
tempo di sistema medio (s)	594,969	75,186	110,811
numero di fermate degli ascensori	1292,422	886,618	1030,292

Tabella 5.21: Risultati ottenuti dal controllore appreso tramite RLG utilizzando l'insieme delle feature ridotto in situazione di traffico intenso di tutte le tipologie.

piano terra, mentre un controllore unico non può certamente permettersi di attuare una politica di questo tipo; in downpeak e in interfloor, invece, le politiche sono più generali e, quindi, possono essere riprodotte più facilmente da un controllore unico.

Sottolineiamo, infine, che come in quasi tutti gli altri casi non si hanno notevoli differenze nelle prestazioni delle politiche apprese con traffico leggero o con traffico intenso, ma entrambe si comportano similmente in tutte le condizioni.

Analizzando il comportamento qualitativo indotto dalla politica appresa possiamo notare che le supposizioni fatte in precedenza per spiegare i dati numerici sono corrette. Il comportamento degli ascensori in downpeak e in interfloor, infatti, sono molto simili a quelli che si hanno con controllori specializzati. In uppeak, invece, gli ascensori liberi vengono mandati soltanto alcune volte, e non sempre, al piano terra; inoltre, la frequenza con cui essi si dirigono al piano terra è superiore nel caso di utilizzo dell'insieme completo delle feature.

Da questa analisi possiamo, quindi, concludere che conviene mantenere dei controllori separati per le diverse tipologie di traffico, in quanto un controllore unico appreso tramite metodi gradienti e politica Gibbs con feature lineari non è in grado di servire adeguatamente il traffico di tipo uppeak.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi abbiamo applicato i metodi dell'apprendimento per rinforzo al controllo di gruppi di ascensori, definendo una nuova architettura per il controllore; per ridurre la dimensionalità del problema, inoltre, siamo ricorsi alla selezione delle variabili, definendo due nuovi algoritmi adatti a questo scopo. Il controllo di un gruppo di ascensori, infatti, è un compito decisamente difficile, sia per la dimensionalità del problema, sia per la stocasticità del problema, sia per gli obiettivi numerosi e conflittuali che si pone; di conseguenza, esso necessita di uno studio approfondito tramite tecniche in grado di contrastare questi problemi, come sono appunto l'apprendimento per rinforzo e la selezione delle variabili.

Per quanto riguarda la selezione delle variabili abbiamo definito due nuovi algoritmi, l'algoritmo garantito e l'algoritmo euristico, studiati in modo da coprire le due mancanze principali emerse dall'analisi degli algoritmi presenti in letteratura, ossia la mancanza di selezioni basate su garanzie teoriche e la non generazione di modelli delle dipendenze tra le variabili selezionate.

L'algoritmo garantito seleziona le variabili in base alla loro bontà per l'individuazione della politica ottima del problema. In particolare, esso è basato su un algoritmo esterno che valuta l'aggiunta di ogni variabile (presa singolarmente oppure in gruppi composti da un numero crescente di variabili se la valutazione precedente non ha portato a nessuna aggiunta), aggiungendo la variabile o il gruppo che riesce a ottenere la valutazione migliore; l'algoritmo termina quando nessuna aggiunta porta, entro un certo margine di tolleranza, a una valutazione migliore. Per valutare le variabili o i gruppi di variabili abbiamo definito una cifra di merito, la quale limita superiormente le perdite di prestazioni della politica ottima dovute alla riduzione. L'algoritmo proposto, quindi, garantisce che a ogni aggiunta di variabile si selezioni quella che, in quel momento, risulta essere, entro certi limiti dovuti alle approssimazioni, la più utile per l'individuazione della politica ottima; ovviamente non si ha alcuna garanzia analoga che l'insieme finale di feature selezionate sia, a parità di cardinalità, quello migliore, perché questo implicherebbe un confronto tra tutte le possibili combinazioni di variabili e non una loro aggiunta iterativa.

L'algoritmo euristico, invece, seleziona le variabili in base alla loro capacità predittiva nei confronti del rinforzo, sia immediato che futuro. In particolare, esso prevede di selezionare inizialmente le variabili necessarie a prevedere il valore che assumerà il rinforzo immediato, ripetendo poi il processo utilizzando come obiettivo della predizione ognuna delle variabili selezionate in un passo precedente. Il modello predittivo che costruiamo è basato su una foresta di alberi di tipo Extremely Randomized Trees. Abbiamo fatto questa scelta in quanto essi, oltre a permettere di costruire un modello adeguato, definiscono implicitamente un criterio di ordinamento delle variabili, in base al quale selezioniamo (una alla volta e ripetendo sul residuo lasciato dal modello costruito in precedenza) le variabili necessarie a prevedere quella desiderata; questo criterio è, semplificando, una misura della riduzione della varianza che la variabile considerata permette di ottenere nelle previsioni del modello. Questo algoritmo, quindi, permette di costruire l'albero di dipendenza delle variabili, derivato appunto dal procedimento di selezione delle variabili che servono a predire una variabile al livello precedente dell'albero. Esso, però, non fornisce alcuna garanzia teorica sulle prestazioni ottenibili con il modello ridotto che genera; inoltre, avendo un approccio predittivo, potrebbe considerare importanti delle variabili effettivamente utili per la predizione (diretta o indiretta) del rinforzo, ma che non influenzano minimamente la politica ottima del sistema.

Per quanto riguarda il controllore di gruppi di ascensori, dopo aver constatato che un'architettura piatta in cui il controllo è lasciato completamente e unicamente agli agenti che apprendono non è adeguata, abbiamo definito una nuova architettura, detta gerarchica, in grado di rendere più semplice l'apprendimento di una buona politica. L'architettura gerarchica prevede di dividere il controllo nei casi in cui a bordo degli ascensori ci siano o meno delle persone, dato che le strategie di controllo applicabili in questi due casi sono differenti; in particolare, un ascensore con nessuno a bordo può recarsi a qualsiasi piano a suo piacimento, mentre uno con a bordo delle persone è vincolato a servire le persone a bordo, in ordine di piano richiesto, con le uniche alternative consistenti nel fermarsi o meno ai piani situati lungo il percorso. L'architettura che abbiamo definito, quindi, delega agli agenti solamente le decisioni strategiche del controllo, limitando così lo spazio delle politiche e rendendo l'apprendimento più semplice.

Utilizzando l'architettura gerarchica abbiamo poi definito i controllori specifici. Per l'apprendimento di questi controllori abbiamo testato due diverse metodologie: Fitted Q Iteration (FQI) e metodi gradiente (RLG). In particolare, a FQI abbiamo abbinato un regressore basato su alberi, mentre con RLG abbiamo utilizzato una politica Gibbs con feature lineari. Per il controllore FQI abbiamo effettuato un apprendimento unico che considera tutte le situazioni di traffico, mentre per quello RLG abbiamo provato sia un apprendimento specifico per condizioni di traffico, sia uno unico. Abbiamo testato i controllori appresi in condizioni di traffico leggero e intenso di tipo uppeak, downpeak e interfloor, confrontando i risultati ottenuti dai

nostri controllori rispetto a un insieme di controllori benchmark proposti in letteratura; abbiamo, inoltre, analizzato le differenze ottenute utilizzando l'insieme completo di feature rispetto a uno ridotto.

Dagli esperimenti effettuati possiamo concludere che l'architettura gerarchica è un'architettura adeguata sulla cui base costruire un controllore di gruppi di ascensori; essa, infatti, permette di ottenere una buona politica attraverso tecniche di apprendimento per rinforzo, a differenza di quanto accade con l'utilizzo di un'architettura piatta. Inoltre, l'utilizzo di metodi di apprendimento della politica ottima tramite gradiente si è rivelato efficace per la soluzione del problema: essi, infatti, permettono di ottenere delle prestazioni buone, abbastanza vicine a quelle dei benchmark con traffico leggero e migliori con traffico intenso, garantendo allo stesso tempo un controllore adattativo. Questo ultimo aspetto è essenziale: il controllore, infatti, è in grado di modificare autonomamente la sua strategia a fronte di cambiamenti nel sistema, consentendo quindi una installazione iniziale che non necessita di modifiche nel corso del tempo. Infine, abbiamo anche constatato che la riduzione delle feature attraverso l'algoritmo euristico è efficace; essa, infatti, ci ha concesso di ridurre significativamente il numero delle variabili, permettendo comunque l'apprendimento di politiche con prestazioni abbastanza vicine a quelle che si hanno utilizzando l'insieme completo delle feature.

Al contrario, gli esperimenti effettuati hanno evidenziato che tramite FQI è difficile ottenere una buona politica per il controllo di un gruppo di ascensori; in particolare, possiamo dedurre che lo spazio degli stati, anche dopo aver effettuato la selezione delle variabili, è troppo grande e, quindi, per permettere l'apprendimento di una politica di controllo adeguata sarebbe necessario utilizzare un numero elevato di dati, che però renderebbe eccessivamente lunghi i tempi di calcolo della politica stessa. Per questo motivo, unitamente anche alla maggiore adattatività, concludiamo che i metodi gradiente sono più adeguati per l'apprendimento di un controllore di gruppi di ascensori rispetto a FQI. Inoltre, nemmeno l'algoritmo di selezione delle variabili garantito è risultato essere utilizzabile nella pratica. Esso, infatti, nonostante mostri buoni risultati su un problema giocattolo, è computazionalmente troppo lento per essere utilizzato su un problema reale, in quanto necessita di tempi molto più lunghi rispetto a quelli necessari per risolvere il problema completo; possiamo, quindi, dire che questo algoritmo è, in pratica, inutile.

Da queste considerazioni consegue che è necessario approfondire ulteriormente lo studio nel campo della selezione delle variabili; in particolare, è necessario rivedere l'algoritmo garantito, in modo da ridurre i suoi tempi computazionali e quindi renderlo utilizzabile nelle applicazioni pratiche. Per raggiungere questo obiettivo è possibile seguire due strade diverse, oltre ovviamente all'alternativa che prevede di riprogettare l'algoritmo da capo. Una possibilità consiste nel rivedere la cifra di merito utilizzata dall'algoritmo, derivandone un'altra che non necessiti la risoluzione dei modelli ridotti, dato che questa operazione occupa la maggior parte del tempo

computazionale. L'altra possibilità, invece, prevede di diminuire il numero di modelli da confrontare; in particolare, si potrebbe provare a trovare delle condizioni tali per cui, data la cifra di merito dei modelli composti da una sola variabile, è possibile selezionare a priori un insieme ridotto di variabili da testare nei passi successivi. Purtroppo, però, gli studi preliminari effettuati per percorrere questa strada non hanno dato risultati positivi, in quanto gli effetti dovuti all'accoppiamento di più variabili sembrano essere indipendenti rispetto ai risultati ottenibili con le variabili singole.

Una volta trovato un algoritmo di feature selection efficiente che offra delle garanzie teoriche lo step successivo prevederebbe l'unione dei concetti dei due algoritmi proposti; in pratica, sarebbe utile studiare un algoritmo che, oltre a offrire delle garanzie teoriche, costruisca anche l'albero delle dipendenze tra le variabili selezionate. In questo modo, infatti, si otterrebbe un albero che evidenzi le dipendenze tra variabili legate al controllo del sistema, e non orientate alla predizione, come invece avviene con l'algoritmo euristico proposto.

Per quanto riguarda il controllo di gruppi di ascensori tramite metodi gradiente sarebbe, invece, utile indagare per capire il motivo che impedisce l'utilizzo delle informazioni sulle chiamate effettive presenti nel sistema; infatti, se si riuscisse a risolvere questo inconveniente, le prestazioni migliorerebbero ulteriormente e, probabilmente, riuscirebbero a eguagliare o battere quelle dei benchmark anche con traffico leggero.

Per utilizzare il controllore proposto è necessario sapere la tipologia di traffico (uppeak, downpeak o interfloor) in corso; nella pratica è possibile stabilirla a priori in base all'orario, dato che lo schema con cui si susseguono le diverse tipologie è prevedibilmente sempre lo stesso ogni giorno e dipende dall'utilizzo dell'edificio in cui il sistema è installato. In questo modo, però, il controllore non sarebbe più adattativo rispetto al cambiamento dello schema di traffici che si ha durante una giornata. È, quindi, utile studiare un metodo che permetta di capire autonomamente, basandosi sulle richieste presente nel sistema e sugli altri dati utilizzabili in un sistema reale, la tipologia di traffico in corso. Per questo scopo è anche possibile utilizzare uno dei metodi proposti in letteratura (si veda ad esempio [55]); in questo caso è necessario studiare come si comporta il controllore proposto in questa tesi accoppiato a un metodo di riconoscimento della tipologia di traffico.

Infine, un'ulteriore direzione di sviluppo percorribile riguarda gli obiettivi posti al controllore. Il controllore da noi proposto, infatti, è costruito in modo da minimizzare i tempi di attesa medi e massimi. Come abbiamo visto, però, i criteri da soddisfare sarebbero molteplici: tempi di viaggio, consumo energetico, affollamento delle cabine, etc.; per tener conto di tutti questi obiettivi sarebbe, quindi, possibile apprendere il controllore tramite metodi gradiente multi-obiettivo.

Bibliografia

[1] Giorgio Bacci, Giovanni Bacci, Kim G Larsen, and Radu Mardare. The bisimdist library: Efficient computation of bisimilarity distances for markovian models.

[2] Gina Carol Barney. *Elevator traffic handbook: theory and practice*. Taylor & Francis, 2004.

[3] AG Barto and RH Crites. Improving elevator performance using reinforcement learning. *Advances in neural information processing systems*, 8:1017–1023, 1996.

[4] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Exploiting structure in policy construction. In *IJCAI*, volume 14, pages 1104–1113, 1995.

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[7] Andrea Castelletti, Stefano Galelli, Marcello Restelli, and Rodolfo Soncini-Sessa. Tree-based variable selection for dimensionality reduction of large-scale control systems. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 62–69. IEEE, 2011.

[8] Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Notions of state equivalence under partial observability. 2009.

[9] Gordon David Closs. *The computer control of passenger traffic in large lift systems*. PhD thesis, University of Manchester Institute of Science and Technology, 1970.

[10] Pablo Cortés, Juan Larrañeta, and Luis Onieva. Genetic algorithm for controllers in elevator groups: analysis and simulation during lunchpeak traffic. *Applied Soft Computing*, 4(2):159–174, 2004.

[11] Robert H Crites and Andrew G Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

[12] Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, pages 106–111, 1997.

[13] Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 124–131. Morgan Kaufmann Publishers Inc., 1997.

[14] Zhu Dewen, Jiang Li, Zhou Yuwen, Shan Guanghui, and He Kai. Modern elevator group supervisory control systems and neural networks technique. In *Intelligent Processing Systems, 1997. ICIPS'97. 1997 IEEE International Conference on*, volume 1, pages 528–532. IEEE, 1997.

[15] Toru Eguchi, Kotaro Hirasawa, Jinglu Hu, and S Markon. Elevator group supervisory control system using genetic network programming with functional localization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 328–335. IEEE, 2005.

[16] Toru Eguchi, Kotaro Hirasawa, Jinglu Hu, and Sandor Markon. Elevator group supervisory control systems using genetic network programming. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1661–1667. IEEE, 2004.

[17] Eyal Even-Dar and Yishay Mansour. Approximate equivalence of markov decision processes. In *Learning Theory and Kernel Machines*, pages 581–594. Springer, 2003.

[18] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 162–169. AUAI Press, 2004.

[19] Norman Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in markov decision processes. *arXiv preprint arXiv:1206.6836*, 2012.

[20] Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for markov decision processes with infinite state spaces. *arXiv preprint arXiv:1207.1386*, 2005.

[21] Norman Francis Ferns. *Metrics for markov decision processes.* PhD thesis, McGill University, 2003.

[22] Raphael Fonteneau, Louis Wehenkel, and Damien Ernst. Variable selection for dynamic treatment regimes: a reinforcement learning approach. 2008.

[23] Atsuya Fujino, Toshimitsu Tobita, Kazuhiro Segawa, Kenji Yoneda, and Akihiro Togawa. An elevator group control system with floor-attribute control method and system optimization using genetic algorithms. *Industrial Electronics, IEEE Transactions on*, 44(4):546–552, 1997.

[24] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[25] Ricardo Gudwin, Fernando Gomide, and M Andrade Netto. A fuzzy elevator group controller with linear context adaptation. In *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, volume 1, pages 481–486. IEEE, 1998.

[26] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *J. Artif. Intell. Res.(JAIR)*, 19:399–468, 2003.

[27] Lacey Gunter, Ji Zhu, and Susan Murphy. Variable selection for optimal decision making. In *Artificial Intelligence in Medicine*, pages 149–154. Springer, 2007.

[28] Hirotaka Hachiya and Masashi Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *Machine Learning and Knowledge Discovery in Databases*, pages 474–489. Springer, 2010.

[29] Kotaro Hirasawa, Toru Eguchi, Jin Zhou, Lu Yu, Jinglu Hu, and Sandor Markon. A double-deck elevator group supervisory control system using genetic network programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(4):535–550, 2008.

[30] Ming Ho and Brent Robertson. Elevator group supervisory control using fuzzy logic. In *Electrical and Computer Engineering, 1994. Conference Proceedings. 1994 Canadian Conference on*, pages 825–828. IEEE, 1994.

[31] Naoki Imasaki, Susumu Kubo, Shoji Nakai, T Yoshitsugu, Jun-Ichi Kiji, and Tsunekazu Endo. Elevator group control system tuned by a fuzzy neural network applied method. In *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference on*, volume 4, pages 1735–1740. IEEE, 1995.

[32] T Ishikawa, A Miyauchi, and M Kaneko. Supervisory control for elevator group by using fuzzy expert system which also addresses traveling time. In *Industrial Technology 2000. Proceedings of IEEE International Conference on*, volume 2, pages 87–94. IEEE, 2000.

[33] Jafferi Jamaludin, Nasrudin Abd Rahim, and Wooi Ping Hew. An elevator group control system with a self-tuning fuzzy logic group controller. *Industrial Electronics, IEEE Transactions on*, 57(12):4188–4198, 2010.

[34] Hironobu Katagiri, Kotaro Hirasawa, Jinglu Hu, Junichi Murata, and Michitaka Kosaka. Network structure oriented evolutionary model: Genetic network programming-its comparison with genetic programming. *TRANSACTIONS-SOCIETY OF INSTRUMENT AND CONTROL ENGINEERS*, 38(5):485–494, 2002.

[35] ChangBum Kim, Kyoung A Seong, Hyung Lee-Kwang, and Jeong O Kim. Design and implementation of a fuzzy elevator group control system. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(3):277–287, 1998.

[36] Jung-Hwan Kim and Byung-Ro Moon. Adaptive elevator group control with cameras. *Industrial Electronics, IEEE Transactions on*, 48(2):377–382, 2001.

[37] Yongdai Kim and Jinseog Kim. Gradient lasso for feature selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 60. ACM, 2004.

[38] Jana Koehler and Daniel Ottiger. An ai-based approach to destination control in elevators. *AI Magazine*, 23(3):59, 2002.

[39] Steven Loscalzo, Robert Wright, Kevin Acunto, and Lei Yu. Sample aware embedded feature selection for reinforcement learning. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 887–894. ACM, 2012.

[40] Peter B Luh, Bo Xiong, and Shi-Chung Chang. Group elevator scheduling with advance information for normal and emergency modes. *Automation Science and Engineering, IEEE Transactions on*, 5(2):245–258, 2008.

[41] Fei Luo, Xiaolan Lin, Yuge Xu, and Huijuan Li. Hybrid elevator group control system based on immune particle swarm hybrid optimization algorithm with full digital keypads. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 1482–1487. IEEE, 2008.

[42] Fei Luo, Yu-Ge Xu, and Jian-Zhong Cao. Elevator traffic flow prediction with least squares support vector machines. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 7, pages 4266–4270. IEEE, 2005.

[43] Melanie Mitchell. *An Introduction To Genetic Algorithms*. MIT Press, 1998.

[44] James Munkres. Topology. 1999.

[45] Daniel M Munoz, Carlos H Llanos, Mauricio Ayala-Rincon, and Rudi H van Els. Distributed approach to group control of elevator systems using fuzzy logic and fpga implementation of dispatching algorithms. *Engineering Applications of Artificial Intelligence*, 21(8):1309–1320, 2008.

[46] Yasuhiro Ogoshi, Haruhiko Kimura, Sadaki Hirose, and Nobuyasu Osato. Elevator group control system using multiagent system. *Systems and Computers in Japan*, 34(1):45–58, 2003.

[47] Ronald Ortner. Pseudometrics for state aggregation in average reward markov decision processes. In *Algorithmic Learning Theory*, pages 373–387. Springer, 2007.

[48] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[49] JianDong Qiu and ZhaoYuan Jiang. The research and simulation on the elevator group control system scheduling algorithm. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 1346–1349. IEEE, 2011.

[50] Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2004.

[51] Balaraman Ravindran and A Barto. Approximate homomorphisms: A framework for nonexact minimization in markov decision processes. In *Proc. 5th Int. Conf. Knowledge-Based Computer Systems*, 2004.

[52] Stéphane Ross and Joelle Pineau. Model-based bayesian reinforcement learning in large structured domains. *arXiv preprint arXiv:1206.3281*, 2012.

[53] Timothy J Ross. *Fuzzy logic with engineering applications*. Wiley, 2009.

[54] Mirko Ruokokoski, Harri Ehtamo, Janne Sorsa, and Marja-Liisa Siikonen. Passenger allocation to capacitated elevator problem, manuscript 2.1 22.10. 2008.

[55] Marja-Liisa Siikonen. *Elevator group control with artificial intelligence*. Citeseer, 1997.

[56] Marja-Liisa Siikonen. *Planning and control models for elevators in high-rise buildings*. Helsinki University of Technology, 1997.

[57] Janne Sorsa, M-L Siikonen, and Harri Ehtamo. Optimal control of double-deck elevator group using genetic algorithm. *International Transactions in Operational Research*, 10(2):103–114, 2003.

[58] Alexander L Strehl, Carlos Diuk, and Michael L Littman. Efficient structure learning in factored-state mdps. In *AAAI*, volume 7, pages 645–650, 2007.

[59] Jin Sun, Qian-Chuan Zhao, and Peter B Luh. Optimization of group elevator scheduling with advance information. *Automation Science and Engineering, IEEE Transactions on*, 7(2):352–363, 2010.

[60] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[61] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

[62] Brian Tanner and Adam White. RL-Glue : Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, 10:2133–2136, September 2009.

[63] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. In *Advances in Neural Information Processing Systems*, pages 1649–1656, 2008.

[64] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[65] S Tsuji, M Amano, and S Hikita. Application of the expert system to elevator group-supervisory control. In *Artificial Intelligence Applications, 1989. Proceedings., Fifth Conference on*, pages 287–294. IEEE, 1989.

[66] Alex Valdivielso and Toshiyuki Miyamoto. Multicar elevator group control: Average reward learning method for service completion time reduction and interference prevention. In *Control Applications (CCA), 2010 IEEE International Conference on*, pages 234–239. IEEE, 2010.

[67] Christopher M Vigorito and Andrew G Barto. Incremental structure learning in factored mdps with continuous states and actions. *University of Massachusetts Amherst-Department of Computer Science, Tech. Rep*, 2009.

[68] Tomasz Walczak and Paweł Cichosz. A distributed learning control system for elevator groups. In *Artificial Intelligence and Soft Computing–ICAISC 2006*, pages 1223–1232. Springer, 2006.

[69] Louis A Wehenkel. *Automatic learning techniques in power systems*. Number 429. Springer, 1998.

[70] Alicia P Wolfe and Andrew G Barto. Decision tree methods for finding reusable mdp homomorphisms. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 530. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[71] Yuge Xu, Fei Luo, and Jianguo Wang. A new modeling method for elevator group control system with cellular automata. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 4, pages 3596–3599. IEEE, 2004.

[72] Xu Yuan, Lucian Busoniu, and Robert Babuška. Reinforcement learning for elevator control. In *Proceedings 17th IFAC World Congress (IFAC-08)*, pages 2212–2217, 2008.

[73] Jin Zhou, Toru Eguchi, Kotaro Hirasawa, Jinglu Hu, and Sandor Markon. Elevator group supervisory control system using genetic network programming with reinforcement learning. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 336–342. IEEE, 2005.

[74] Jin Zhou, Toru Eguchi, Shingo Mabu, Kotaro Hirasawa, Jinglu Hu, and Sandor Markon. A study of applying genetic network programming with reinforcement learning to elevator group supervisory control system. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3035–3041. IEEE, 2006.

[75] Jin Zhou, Lu Yu, Shingo Mabu, Kotaro Hirasawa, Jinglu Hu, and Sandor Markon. Service area-based elevator group supervisory control system using gnp with rl. In *SICE-ICASE, 2006. International Joint Conference*, pages 5967–5972. IEEE, 2006.

[76] Jin Zhou, Lu Yu, Shingo Mabu, Kotaro Hirasawa, Jinglu Hu, and Sandor Markon. Double-deck elevator systems using genetic network programming with reinforcement learning. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2025–2031. IEEE, 2007.

[77] Ziliang Zong, Xugang Wang, Zheng Tang, and Guangzhou Zeng. Elevator group control algorithm based on residual gradient and q-learning. In *SICE 2004 Annual Conference*, volume 1, pages 329–331. IEEE, 2004.