POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

Un metodo di ricerca della struttura ottima di un sistema multiagente per l'individuazione di anomalie in applicazioni spaziali

Relatore:

prof. Francesco AMIGONI

Correlatore:

dott. Alessandro RIVA

Laureando:

Maurizio FERRARI DACREMA
Matricola 804354

Anno Accademico 2015-2016

Abstract

Implementing an anomaly detection system for a plant or a technological equipment is a potentially problematic activity requiring to take into consideration the caracteristics of the monitored processes, the underlying physical phenomena, and their interdependencies. Many anomaly detection methods rely either on detailed domain knowledge or on big datasets from wich to extract information about the behaviour of the monitored system. Therefore the possible absence of domain knowledge combined with a lack of data gathered during nominal operations can represent a significant obstacle. For this reason space missions constitute a particularly challenging context in which anomaly detection might prove tricky. To tackle this problem we propose to use a multiagent system whose flexibility in the addition, removal and aggregation of the agents provides good scalability and adaptability. Each agent is associated with a model of the nominal behaviour of a given variable and will be used to evaluate the data coming from the space system and to estimate how anomalous the mesurement is. The purpose of this thesis is to integrate an existing anomaly detection multiagent system with a method to search the optimal structure of the agents to be used for the aggregation of their results. Several search algorithms are proposed together with many aggregation functions, whose performance is evaluated with respect to different test cases. It is shown that the proposed approach shows promising results as it allows to obtain a structure of agents able to achieve high accuracy in the detection of anomalies for all test cases.

Sommario

Implementare un sistema di anomaly detection per un impianto o un apparato tecnologico è un'attività complessa che coinvolge considerazioni legate alla natura dei processi monitorati, ai fenomeni fisici sottostanti e alle loro interdipendenze. Molti metodi di anomaly detection sfruttano dettagliata domain knowledge o vasti dataset dai quali estrarre informazioni relative al comportamento del sistema. Pertanto l'eventuale assenza di domain knowledge unita ad una scarsa disponibilità di dati di funzionamento nominale potrebbe costituire un significativo ostacolo. Per questo motivo l'ambito delle missioni spaziali è un contesto in cui la anomaly detection è particolarmente delicata. Per affrontare questo problema proponiamo di utilizzare un sistema multiagente, la cui flessibilità nell'aggiunta, rimozione ed aggregazione degli agenti offre buona scalabilità e adattabilità. Ogni agente è associato ad un modello del comportamento nominale di una variabile ed è utilizzato per analizzare i dati provenienti dal sistema monitorato e valutare quanto la misurazione sia anomala. Lo scopo di questa tesi è integrare un sistema di anomaly detection multiagente esistente con un metodo di ricerca della struttura ottima per l'aggregazione dei risultati calcolati dagli agenti. Nella tesi sono proposti diversi algoritmi per la ricerca della struttura ottima del sistema multiagente e funzioni di aggregazione le cui performance sono state valutate con diversi casi di test. Si è osservato che L'approccio proposto è promettente in quanto permette di ottenere un'elevata accuratezza nel riconoscimento delle anomalie per tutti i casi di test.

Ringraziamenti

Indice

Elenco delle figure **19**

Elenco delle tabelle **21**

1 Introduzione **23**

2 Stato dell'arte **29**
2.1 Fault Detection Isolation and Recovery 29
2.1.1 Caratteristiche di un sistema FDIR 30
2.1.2 Trasformazione dei dati 31
2.1.3 Model-based fault detection 32
2.1.4 Data-driven fault detection 34
2.2 Fault detection in applicazioni spaziali 38
2.3 Sistema multiagente 40

3 Impostazione del problema di ricerca **43**
3.1 Caso applicativo Cryosat-2 43
3.2 Flight Control Team 44
3.2.1 Stazioni di terra 45
3.2.2 Flight Control Team Multi Agent System 45
3.2.3 Aggregazione gerarchica 47
3.3 Anomalie individuabili 48
3.4 Formalizzazione 49
3.4.1 Formalizzazione di alto livello 49
3.4.2 Formalizzazione tramite grafo 50
3.5 Funzione di aggregazione e spazio delle soluzioni 51
3.5.1 Spazio delle strutture possibili 51
3.5.2 Funzione di aggregazione massimo 51
3.5.3 Funzione di aggregazione media 53
3.5.4 Funzione di aggregazione negoziazione cooperativa . . 58
3.6 Apporti innovativi 59

4 Algoritmi di ricerca **63**
4.1 Caratteristiche della struttura degli agenti 63
4.2 Algoritmi di ricerca globale 65
4.2.1 Algoritmo enumerativo 65
4.2.2 Algoritmo branch and bound 65
4.3 Algoritmi di ricerca locale 67
4.3.1 Simulated annealing 67
4.3.2 Algoritmo greedy 71
4.4 Ricerca su grafo degli agenti 78

5 Architettura del sistema software **81**
5.1 Struttura di alto livello 81
5.2 Package OptimalStructure 82

6 Risultati sperimentali **87**
6.1 Cost matrix e timestamp significativi 87
6.2 Dimensione del problema gestibile 89
6.2.1 Algoritmo enumerativo 89
6.2.2 Simulated annealing 92
6.2.3 Algoritmo greedy 94
6.3 Proprietà del sistema multiagente per l'identificazione delle anomalie 97
6.3.1 Che tipologie di anomalie si possono riconoscere . . . 97
6.3.2 Propagazione dell'anomalia verso la radice 97
6.4 Prove sperimentali 97
6.4.1 Log di un mese 98
6.4.2 Forti anomalie, agenti indipendenti 103
6.4.3 Gruppo correlato di agenti 111
6.4.4 Soglie di anomalia differenziate 117
6.5 Conclusioni 124

7 Conclusioni e sviluppi futuri **127**

A Documentazione della programmazione **131**
A.1 Inizializzazione e avvio ricerca 131
A.1.1 StructureBuilder 133
A.2 Algoritmo enumerativo 135
A.3 Simulated Annealing 136
A.4 Enumerazione per funzione media 138
A.5 Algoritmo greedy 139
A.6 Calcolo della predizione tramite Jade 139
A.6.1 File di configurazione della struttura 142
A.7 Log delle soluzioni esplorate 143
A.8 File di configurazione dei sottoalberi 145

A.9 File di configurazione OptimalStructureManager 146

Bibliografia **149**

Elenco delle figure

2.1 Schema di un sistema FIDR [1]. 30
2.2 Trasformazione dei dati in un sistema di fault detection. . . . 32

3.1 Struttura di una soluzione con tre nodi intermedi di interesse. 52
3.2 Una possibile struttura. 53
3.3 Soluzione strutturalmente diversa ma equivalente a quella in Figura 3.2. 55
3.4 Una struttura possibile. 58

4.1 Struttura iniziale. 70
4.2 Scambio tra le foglie D1 e D3. 70
4.3 Spostamento della foglia D3 da M2 a M1. 70
4.4 Aggiunto mediatore M3, inglobando D1 e D2. 70
4.5 Rimosso il mediatore M2, le sue foglie sono acquisite da M1. . 70
4.6 Aggiunto alla radice l'insieme formato da D1 e D2. Ancora da inserire D3, D4 e D5. 73
4.7 Aggiunto ai figli della radice l'insieme di agenti diagnostici composto da D3 e D4. Ancora da inserire D5. 73
4.8 Creato un nuovo agente mediatore figlio della radice, avente per figli D3, D4 e D5. Tutti gli agenti diagnostici sono stati inseriti. 73
4.9 Soluzioni raggiungibili con struttura ad albero perfettamente sbilanciato. 74
4.10 Soluzioni raggiungibili con struttura ad albero sbilanciato $z = 2$ 75
4.11 Grafo delle soluzioni 78
4.12 Grafo degli agenti 79

5.1 Diagramma dei package e delle classi principali del sistema FCTMAS. 82
5.2 UML di alto livello del package StructureBuilder 83
5.3 Gerarchia delle classi per la valutazione di una soluzione . . . 85

6.1 Albero perfettamente sbilanciato con almeno due figli per nodo 95
6.2 Albero binario perfettamente bilanciato 95

6.3 Soluzioni individuate tramite enumerazione per il test con il log di un mese, disposte in ordine di costo. 100
6.4 Soluzione ottima in cui l'agente A13957 è figlio di M0001. . . 100
6.5 Soluzioni individuate per il test con il log di un mese, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. I valori sono medi su 10 esecuzioni. 101
6.6 Alcune strutture visitate tramite simulated annealing. 102
6.7 Soluzioni individuate per il test con il log di un mese, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni. 103
6.8 Albero di livello 3, FN ai timestamp 7 e 15. 105
6.9 Albero di livello 3, FN ai timestamp 3 e 7. 105
6.10 Soluzioni individuate per il test con forti anomalie ed agenti indipendenti, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni. 106
6.11 In queste soluzioni gli agenti A2, A3 ed A5 sono sempre collocati al secondo livello. Al primo invece si trovano le diverse combinazioni di agenti che esibiscono forti anomalie. Alla radice è applicata la funzione di aggregazione massimo, agli altri mediatori la negoziazione cooperativa. 107
6.12 Soluzione ad albero perfettamente bilanciato. Alla radice è applicata la funzione di aggregazione massimo, agli altri mediatori la negoziazione cooperativa. 108
6.13 Soluzioni individuate tramite enumerazione per il test con forti anomalie ed agenti indipendenti, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa. 109
6.14 Le soluzioni con coefficiente massimo 12 sono di livello 3, le altre di livello 4. Tutte queste soluzioni hanno costo 650, con 13 FN la cui posizione cambia a seconda della specifica soluzione. 110
6.15 Soluzione di livello 3. Il suo costo è 500, i FN si trovano in corrispondenza dei timestamp: 1, 2, 10, 11, 12, 13, 14, 16, 17, 18. 110
6.16 Soluzione ad albero perfettamente bilanciato. 113

6.17 Configurazione ottima con dataset modificato come in Tabella 6.12. 113
6.18 Soluzioni individuate per il test con un gruppo di agenti correlati, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori solo medi su 10 esecuzioni. 114
6.19 Massimo applicato al livello 1, il gruppo di agenti correlati nel dataset è direttamente figlio della radice. 115
6.20 Configurazione simile a quella di Figura 6.19, il gruppo di agenti correlati è raggruppato nel mediatore 1. 115
6.21 Il gruppo di agenti correlati in questo caso è separato. Poichè A5 non esibisce nessuna anomalia, le probabilità di anomalia esisbite da M2 sono uquali a quelle di A2. 115
6.22 Talvolta la funzione massimo è applicata alla radice, ma nel qual caso A3 si trova sempre a livello 3 o inferiore. 115
6.23 Soluzioni individuate tramite enumerazione per il test con un gruppo di agenti correlati, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa. 116
6.24 Soluzione di livello 4. Il suo costo è 150 ed i FN sono relativi ai timestamp: 10, 17, 19. Una sola anomalia per il gruppo A1, A2, A3 è riconosciuta, quella più marcata nel timestamp 18. 116
6.25 Soluzione di livello 2. Il suo costo è 200 ed i FN sono relativi ai timestamp: 10, 17, 18, 19. Si tratta dell'unica soluzione ottima. Nessuna anomalia per i gruppo A1, A2, A3 è riconosciuta. Nel timestamp 18 la probabilità di anomalia associata al sistema è 0.15. 117
6.26 Soluzione ad albero perfettamente sbilanciato. Il suo costo è 50 a causa di FP nei timestamp: 12, 13, 18, 19, 20. 119
6.27 Soluzione ottima ottenuta aggiungendo al dataset l'agente A6. Il suo costo è pari a 10 a causa di un FP nel timestamp 20. 120
6.28 Soluzione ottima ottenuta aggiungendo al dataset gli agenti A6 ed A7. Il suo costo è pari a 0. 120
6.29 Soluzioni individuate per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni. 121

6.30 Soluzione con costo 50 a causa di FP nei timestamp: 12, 13, 18, 19, 20. 122

6.31 Soluzione ottima ottenuta aggiungendo al dataset l'agente A6. Il suo costo è 30 a causa di FP nei timestamp: 13, 19, 20. 122

6.32 Soluzione ottima ottenuta aggiungendo al dataset gli agenti A6 ed A7. Il suo costo è 20 a causa di FP nei timestamp: 13, 20. 122

6.33 Soluzioni individuate tramite enumerazione per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa. 123

6.34 Soluzioni ottime ottenute applicando la funzione di aggregazione media, il loro costo è 200. Le soluzioni con coefficiente massimo 8 sono di livello 2, le altre di livello 4. 124

6.35 Il costo di questa soluzione è 20 a causa di FP nei timestamp: 19, 20. La funzione di aggregazione media è applicata ad M1 e M2, R applica negoziazione cooperativa. 124

A.1 Diagramma di sequenza del costruttore di OptimalStructure-Manager. 132

A.2 Diagramma di sequenza del metodo generateOptimalStructure in OptimalStructureManager. 133

A.3 Metodo getInstance in StructureBuilder per tre dei quattro algoritmi, istanzia l'oggetto relativo all'algoritmo specificato nel file di configurazione. 134

A.4 Metodo loadAnomalyAndTargetProbabilities in StructureBuilder. 134

A.5 Metodo logBestSolution in StructureBuilder. Si integra la soluzione ottima trovata con i sottoalberi, si crea il contenuto de file di configurazione e lo si salva nel log XML. 135

A.6 Metodo getInstance nella classe SolutionIterator, restituisce l'iteratore associato alla struttura di soluzione desiderata. . . 136

A.7 Metodo buildStructure della classe BuildStructureEnumeration. Ogni thread ha assegnato un certo numero di soluzioni tra le quali sceglie la migliore. Al thread è passato come parametro un clone dell'iteratore posizionato sulla prima soluzione a lui assegnata. 137

A.8 Metodo buildStructure della classe BuildStructureSAnnealing. L'algoritmo è implmementato in modo sequenziale. . . . 138

A.11 I metodi generateAddDiagnosticAgentActions e generateNewMediatorAgentActions in BuildStructureGreedy si comportano in modo simile. Sono composti da tre cicli annidati, uno sui nodi mediatori esistenti, uno sulla dimensione del set di agenti da coinvolgere nell'azione ed uno sullo specifico set di agenti. 139
A.9 Metodo buildStructure della classe BuildStructureEnumerationAverage. Ogni thread ha assegnato un certo numero di soluzioni tra le quali sceglie la migliore. Al thread è passato come parametro un clone dell'iteratore posizionato sulla prima soluzione a lui assegnata. 140
A.10 Metodo buildStructure della classe BuildStructureGreedy. Ogni thread esegue l'algoritmo greedy sino a raggiungere una soluzione ottima locale. 141
A.12 Metodo getprediction della classe JadeWrapper, avvia JADE e legge i risultati. 142
A.13 Esempio del file di configurazione contenente una struttura. . 143
A.14 Intestazione del file XMLLog. Si possono osservare tutti i parametri che compaiono nel file di configurazione e l'elenco degli agenti coinvolti. 144
A.15 Elemento *newSolution* nel file XMLLog. Oltre alla struttura sono riportati lo score, la confusion matrix e la lista di falsi positivi e falsi negativi, se presenti. 145
A.16 Sono elencati gli agenti appartenenti a due sottosistemi. . . . 145
A.17 Prima parte del file OptimalStructure.properties. Sono riportate informazioni legate all'algoritmo ed alla tpopologia desiderata per la soluzione. 147
A.18 Seconda parte del file OptimalStructure.properties. Sono riportate informazioni legate alla funzione di aggregazione, alla neighbor function e all'uso del log XML. 148

Elenco delle tabelle

3.1 Numero di possibili soluzioni per dimensione del grafo 51
3.2 Numero di possibili soluzioni per numero di foglie nel caso di funzione di aggregazione massimo e tre sottoalberi. 53
3.3 Matrice struttura per l'albero in Figura 3.3. 56

6.1 Struttura della cost matrix utilizzata per valutare le soluzioni. 88
6.2 In tabella sono riportati i numeri di timestamp esistenti nei due log disponibili del satellite Cryosat-2. Si confronta il numero totale di timestamp presenti nel log, quelli relativi ad un evento per cui si è potuto costruire un modello e, di questi, quelli nei quali esista almeno un agente anomalo. 89
6.3 Numero di possibili soluzioni per dimensione del problema e tempo necessario ad esplorarle. 90
6.4 Il valore del livello si riferisce al numero massimo di archi tra la radice e la foglia più lontana. Nel caso di livello 2 radice e foglie sono quindi separate al massimo da un mediatore. . . . 91
6.5 Confronto tra il numero di alberi e il numero di scheletri. . . 93
6.6 Confronto tra il numero di alberi e il numero di scheletri di livello massimo 2. 94
6.7 Dati utilizzati per il test con il log di un mese. Sulle colonne è riportato il nome degli agenti diagnostici. 99
6.8 Dati utilizzati per il test con forti anomalie ed agenti indipendenti. 104
6.9 Confusion matrix delle due soluzioni ottime. Si possono osservare due falsi negativi. 105
6.10 Ai timestamp indicati in tabella è associato uno stato del sitema nominale. 105
6.11 Dati utilizzati per il test con un gruppo di agenti correlati. . . 112
6.12 Ai timestamp indicati in tabella è associata un'anomalia di A3 più elevata rispetto a quanto riportato in Tabella 6.11. . . 113
6.13 Dati utilizzati per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata. 118

Capitolo 1

Introduzione

Implementare un sistema di anomaly detection per un impianto industriale o un qualsiasi apparato tecnologicamente complesso richiede siano disponibili informazioni relativamente al suo stato, al suo funzionamento in regime nominale e una procedura che consenta di riconoscere uno scostamento da esso [1]. A causa della grande varietà di contesti e di fenomeni fisici che possono essere coinvolti, sono state sviluppate negli anni molteplici tecniche di anomaly detection [2], alcune delle quali adattabili per applicazioni in ambito spaziale. L'ambito dei veicoli spaziali presenta diverse similarità con l'ambito industriale, in cui solitamente le tecniche per l'identificazione delle anomalie nascono, ma anche una serie di peculiarità e di criticità aggiuntive. Particolarmente laborioso può essere lo sviluppo di metodi per il riconoscimento di anomalie in contesti in cui non si abbia accesso ad alcuna domain knowledge e il volume di dati disponibile sia limitato [3] [4]. In questi casi l'applicazione di un gran numero di tecniche già consolidate è preclusa.

Lo scopo di questa tesi è integrare un sistema di anomaly detection esistente, implementato tramite un sistema multiagente, in modo che la struttura con la quale gli agenti sono organizzati e i loro risultati aggregati sia ottimizzata sulla base dei dati disponibili. Si vuole che il risultato ottenuto garantisca elevata accuratezza nell'identificazione di anomalie e sufficiente robustezza. Per raggiungere questo obiettivo si propongono diversi algoritmi di ricerca nello spazio delle possibili strutture del sistema multiagente combinate a funzioni per l'aggregazione dei risultati degli agenti, la cui efficacia è valutata su dei casi di test, sia reali sia sinteticamente costruiti. Si osserva che l'approccio proposto è promettente in quanto consente di ottenere un'elevata accuratezza nell'individuazione di anomalie ed estrarre informazioni relative all'effettiva struttura del sistema monitorato.

Implementare un sistema di anomaly detection senza avere a disposizione domain knowledge richiede che ci si appoggi a dati prodotti dal funziona-

mento nominale del sistema da monitorare, dai quali estrarre informazioni relative al suo funzionamento. Poichè i satelliti sono sistemi complessi su cui possono essere disponibili centinaia di sorgenti di dati relativi a componenti in stretta relazione tra loro, riuscire a riconoscere in modo esaustivo scostamenti dal funzionamento nominale può essere difficile. Tali scostamenti possono essere relativi all'uscita da un range nominale del valore restituito dalla misurazione di un sensore, all'inconsistenza rispetto ad un contesto o all'incompatibilità tra più misurazioni.

Le prime tecniche di anomaly detection che sono state sviluppate si sono incentrate sulla costruzione di un modello che rappresenti dettagliatamente il funzionamento del sistema ed i fenomeni fisici coinvolti. L'utilizzo di un insieme di equazioni matematiche [2] solitamente offre una rappresentazione del sistema con un altissimo livello di dettaglio, rendendo possibile calcolare una predizione di quale dovrebbe essere il valore rilevato da un sensore in base ai dati ottenuti da altri. La tecnica della *ridondanza analitica* è molto usata poichè, in virtù della sua precisione, consente di individuare puntualmente l'anomalia risalendo alle sue cause [5]. Costruire un modello così preciso è però complesso e oneroso. Alcuni approcci alternativi prevedono di sviluppare un modello di tipo qualitativo, meno preciso ma più semplice da implementare. A questa famiglia di metodi appartengono per esempio i *digrammi causali* [6], metodo orientato alla scoperta delle interdipendenze tra variabili, le cui relazioni causa-effetto sono rappresentate tramite un grafo.

Utilizzare un metodo basato su un modello del sistema da monitorare richiede che sia disponibile adeguata domain knowledge per costruire il modello stesso. Nel contesto di interesse per questa tesi si assume di non avere accesso ad alcuna domain knowledge e, quindi, le informazioni relative al comportamento del sistema da monitorare dovranno essere estratte dai dati raccolti durante il suo funzionamento nominale. Sistemi di anomaly detection che non si affidino alla conoscenza del sistema fornita da esperti sono chiamati *data-driven* e stanno avendo notevole successo negli ultimi anni grazie alla semplicità della loro implementazione [4]. A cavallo tra le due famiglie di metodi, quelli basati su domain knowledge e quelli data-driven, si collocano i sistemi esperti. Il loro utilizzo è molto versatile poichè la knowledge base può essere affinata progressivamente [7], il ragionamento per mezzo del quale si è presa una decisione è trasparente e si dimostrano robusti all'incertezza [8] [9]. Altre tecniche strettamente data-driven e di tipo statistico, pensate per problemi con elevato numero di sensori i cui valori mostrano diffuse correlazioni, sono la *principal component analysis*, PCA [10] [11], e la *partial least squares regression*, PLS [12]. L'uso di sistemi multiagente è raro in questo ambito. Si può citare ad esempio il framework *MADCABS* [13] che opera combinando predizioni effettuate da metodi diversi, ciascuno associato ad un agente. I metodi utilizzati nella data-driven anomaly detection sono però pensati per essere applicati su una cospicua

mole di dati, sufficiente non solo a definire accurati modelli nominali delle variabili, ma anche per estrarre informazioni sulle reciproche correlazioni. Nel contesto di interesse per questa tesi, invece, la disponibilità di dati è limitata e quindi lo sarà anche la capacità rappresentativa del modello finale. Il progetto *Flight Control Team Multi Agent System*, FCTMAS [14], di cui questa tesi costituisce un'integrazione, non ha come obiettivo la realizzazione di un sistema autonomo di anomaly detection isolation and recovery, bensì di uno strumento di supporto al gruppo di operatori umani che si occupa da terra di gestire e supervisionare il satellite Cryosat-2, chiamato *Flight Control Team*, FCT. Il sistema FCTMAS assegna ad ogni agente il monitoraggio di una variabile relativa allo stato del satellite per la quale sia stato possibile costruire un modello di comportamento nominale. Gli agenti, sulla base dei dati contenuti nel log del satellite, utilizzano il modello per calcolare una *probabilità di anomalia* relativa a una nuova misurazione della variabile. Tanto più la misurazione sarà lontana dal comportamento nominale, tanto più alta sarà la probabilità di anomalia restituita. Ciascun agente monitora una sola variabile indipendentemente dalle altre. Le probabilità di anomalia restituite dagli agenti sono poi aggregate in una probabilità riassuntiva dello stato del satellite nel suo complesso. Questo sistema, pur consentendo di individuare diversi casi di anomalia, presenta una limitazione legata al fatto che variabili relative a fenomeni fisici completamente diversi o a sistemi indipendenti sono aggregate insieme, allo stesso livello, per rendere disponibile un unico risultato per l'intero sistema monitorato. Per superare questa limitazione, questa tesi propone lo sviluppo di un metodo per la ricerca della struttura ottima con la quale i risultati di diversi sottoinsiemi di agenti siano progressivamente aggregati sino a raggiungere un risultato complessivo per l'intero satellite.

Si è scelto di rappresentare la struttura degli agenti come un grafo e concentrarsi su una struttura ad albero, identificando gli agenti come nodi e le relazioni di aggregazione come archi. Le foglie dell'albero rappresentano gli agenti cui è associato il modello di comportamento nominale per una variabile. I nodi intermedi rappresentano agenti il cui compito è aggregare le probabilità di anomalia calcolate dai suoi figli. La radice dell'albero sarà associata alla probabilità di anomalia del satellite utilizzata per determinare se lo stato complessivo sia nominale o anomalo. Ricercare la struttura ottima è un problema difficile da affrontare in quanto l'elevato numero di variabili, e quindi agenti, rende le strutture possibili estremamente numerose. Si è studiata innanzitutto la relazione esistente tra le funzioni utilizzabili per aggregare i risultati degli agenti e lo spazio delle soluzioni. Per esempio, la funzione *massimo*, se applicata per tutte le aggregazioni, rende la struttura irrilevante e determina una eccessiva riduzione dello spazio delle soluzioni. La funzione *media* permette di definire uno spazio delle soluzioni più piccolo rispetto a quello di tutti i possibili alberi di agenti e garantisce una velocità di risoluzione molto alta, ma causa un deterioramento nella

qualità del risultato. Infine la funzione *negoziazione cooperativa*, come definita in [14], non permette di ridurre l'intero spazio degli alberi. In questa tesi proponiamo tre diversi algoritmi per affrontare il problema di ricerca. L'algoritmo *enumerativo* è pensato per esplorare l'intero spazio delle soluzioni associate alla funzione di aggregazione scelta. La scalabilità di questo algoritmo è però limitata dalla crescita esponenziale del numero di soluzioni, rendendo gestibili solo problemi dal ridotto numero di agenti. L'algoritmo *simulated annealing* è adatto a svolgere ricerche su spazi di stati discreti molto vasti, coniugandosi bene con le caratteristiche del nostro problema ed offrendo un buon compromesso tra la qualità della soluzione (in generale l'ottimalità globale non può essere garantita) e la velocità di elaborazione. Infine, l'algoritmo *greedy* è pensato per costruire la soluzione da zero, anzichè cercarla in uno spazio di soluzioni possibili, prendendo delle decisioni localmente ottime. Quest'ultimo algoritmo permette di generare soluzioni molto rapidamente e mostra una complessità computazionale molto inferiore agli algoritmi che esplorano lo spazio delle soluzioni.

Il sistema sviluppato in questa tesi costituisce un modulo aggiuntivo al sistema FCTMAS e ne integra le funzionalità. Il modulo offre un'implementazione per ciascuno degli algoritmi descritti e permette di configurare una serie di parametri.

Si lascia a sviluppi futuri il superamento di alcune limitazioni dell'attuale approccio, tra le quali l'incapacità di riconoscere anomalie legate all'inconsistenza reciproca dei valori di un insieme di variabili. Ulteriore ambito di intervento potrà essere il test del sistema proposto su problemi con numero di variabili più elevato rispetto a quelli considerati per verificarne le performance e perfezionare gli algoritmi proposti incrementandone l'efficienza. Futuri sviluppi potrebbero anche riguardare la definizione di metodi per integrare progressivamente nella struttura nuovi agenti associati a sensori per i quali sia stato possibile definire un modello di comportamento nominale, senza dover ricalcolare la struttura da zero.

Di seguito si descrive la struttura dell'elaborato di tesi:

- Nel Capitolo 2 si descrivono le diverse famiglie di metodi utilizzate per l'implementazione di sistemi di anomaly detection, le tecniche maggiormente utilizzate in ambito spaziale e il sistema FCTMAS.

- Nel Capitolo 3 si presenta il contesto applicativo in cui si inserisce questo progetto ed il satellite Cryosat-2. Si fornisce inoltre una formalizzazione del problema, la descrizione della natura delle anomalie individuabili e del legame esistente tra funzioni di aggregazione e spazio delle soluzioni.

- Nel Capitolo 4 si descrivono gli algoritmi proposti per la ricerca delle soluzioni, che sfruttano approcci sia globali che locali.

- Nel Capitolo 5 si presenta l'integrazione tra il modulo sviluppato in questa tesi ed il sistema FCTMAS.
- Nel Capitolo 6 si descrivono le analisi sperimentali effettuate su diversi casi di test e si discutono le dimensioni dei problemi gestibili.
- Nel Capitolo 7 si presentano le conclusioni di questo elaborato, riassumendone lo scopo, i risultati conseguiti e le prospettive di sviluppi futuri.
- Nell'Appendice A si descrive nel dettaglio il modulo sviluppato, compresi gli algoritmi implementati e i file di configurazione.

Capitolo 2

Stato dell'arte

2.1 Fault Detection Isolation and Recovery

Garantire la piena e continua operatività di un apparato complesso è un problema significativo a livello ingegneristico, in quanto richiede si riescano ad individuare prontamente anomalie nel suo funzionamento. Nel corso degli ultimi decenni questa attività è stata sempre più automatizzata e numerosi approcci e strategie sono stati sviluppati. Con *fault detection isolation and recovery*, FDIR, ci si riferisce all'individuazione e correzione di anomalie nel funzionamento di un sistema, intervenendo quando esso si trova ancora in una regione di funzionamento controllabile [2]. I sistemi FDIR svolgono tre funzioni principali: rilevare l'esistenza di un guasto, individuarne l'origine e ripristinare, laddove possibile, il normale funzionamento. Uno schema di alto livello è riportato in Figura 2.1.

L'attività di monitoraggio dello stato del sistema è svolta tramite dei sensori che forniscono alcune specifiche informazioni relative allo stato fisico vero e proprio. Ciascuno di questi sensori è associato ad una variabile del processo. Il loro numero varia notevolmente a seconda del contesto, fino a raggiungere le migliaia nel caso di impianti industriali.

Con il termine "guasto" o "anomalia" ci si riferisce ad uno scostamento dal normale funzionamento, in grado di produrre un effetto osservabile tramite le variabili del processo. Nel dominio applicativo dei satelliti esempi di guasti sono: impreciso posizionamento, canale di comunicazione disturbato, stress strutturale, temperatura alta, basso livello delle batterie... Con il termine "avaria" si intende l'evento fisico all'origine del guasto: rottura di un disco magnetico o di un pannello solare, spegnimento della refrigerazione...

Alcune anomalie possono essere legate a fenomeni esogeni che ci si aspetta posano occorrere ma non sono controllabili, ad esempio perturbazioni o eruzioni solari che rendano la comunicazione temporaneamente difficoltosa, altre anomalie invece possono preludere a conseguenze più gravi. Un impreciso posizionamento può essere sintomo di un problema nei propul-

sori, la temperatura alta può indicare un problema nel sistema di refrigerazione ed un anomalo livello delle batterie può indicare che esse si stiano deteriorando più rapidamente del previsto. In tutti questi casi intervenire prontamente può consentire di pianificare una strategia per risolvere il problema o minimizzare i suoi effetti a lungo termine. Effetti a lungo termine per le anomalie precedentemente menzionate possono essere: l'impossibilità di mantenere stabile l'orbita, danni permanenti all'elettronica di bordo, insufficiente disponibilità di energia.

Individuare l'esistenza di un guasto fa parte della fase di *detection*. La fase di *identification* invece è relativa all'individuazione dell'avaria che ha causato il guasto. La fase di *recovery* prevede di determinare le azioni da far compiere al controller del processo per ripristinare il normale funzionamento.

Poichè i satelliti sono apparati articolati e complessi, sono disponibili una gran quantità di dati prodotti da decine o centinaia di sensori, esistono interdipendenze tra i vari sistemi ed il preavviso disponibile in presenza di un'anomalia può essere limitato. I satelliti pertanto presentano delle similarità con un settore in cui lo sviluppo di sistemi FDIR è molto attivo, gli impianti industriali.

Le tecniche di fault detection si possono suddividere in due classi, quelle che richiedono di codificare esplicitamente un modello del sistema, Sezione 2.1.3, e quelle che si basano solo su dati prodotti dal suo funzionamento, Sezione 2.1.4.

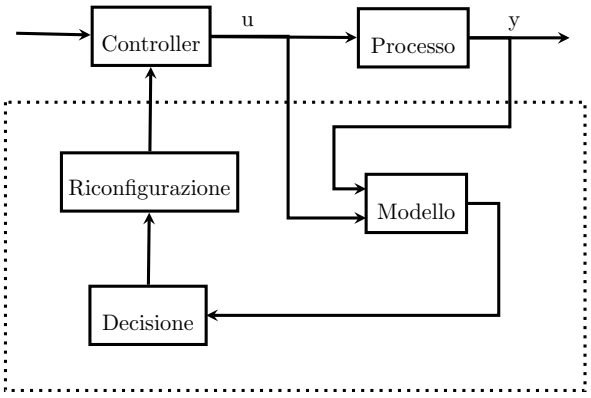

Figura 2.1: Schema di un sistema FIDR [1].

2.1.1 Caratteristiche di un sistema FDIR

Nella scelta di quale approccio adottare, ci si può basare su una serie di caratteristiche desiderabili per un sistema FDIR.

- **Completezza** - In presenza di un'anomalia il sistema la deve sempre rilevare. Se si effettua anche la fase di identificazione, l'avaria che l'ha causata deve essere tra quelle che il sistema suggerisce.

- **Rapidità** - Il sistema deve riconoscere le anomalie in un breve lasso di tempo. Si accorcia la retroazione ma aumenta la sensibilità al rumore.

- **Isolabilità** - Deve essere possibile distinguere guasti legati ad avarie diverse.

- **Robustezza** - Il sistema deve essere poco sensibile al rumore.

- **Anomalie sconosciute** - Lo spazio di funzionamento nominale può essere definito in modo relativamente completo, quello delle anomalie no. Il sistema deve riuscire ad individuare correttamente anomalie legate ad avarie non previste nel modello.

- **Comprensibile** - Deve essere possibile risalire a come il guasto si è prodotto.

- **Basso costo iniziale** - Il modello alla base del sistema deve essere definibile con un dispendio di risorse contenuto.

2.1.2 Trasformazione dei dati

Affinchè sia possibile classificare le anomalie, i dati delle misurazioni devono essere progressivamente trasformati, come mostrato in Figura 2.2. Si definiscono quattro spazi:

- **Spazio delle misurazioni** - Costituito dai valori assunti dalle variabili.

- **Spazio delle features** - Una feature è funzione delle misurazioni, rappresentando una relazione tra di loro, ed è definita grazie all'apporto di domain knowledge.

- **Spazio di decisione** - Costituito applicando alle feature una funzione di decisione, nel caso più semplice un threshold. Per tarare i parametri delle funzioni di decisione è necessario applicare algoritmi di ricerca o apprendimento.

- **Spazio delle classi** - Comprende le avarie riconoscibili.

La definizione degli spazi delle feature e di decisione è un passaggio chiave per ottenere un buon sistema di fault detection. Avendo a disposizione una sufficiente domain knowledge per definire un robusto spazio delle feature, la difficoltà della ricerca di uno spazio di decisione ottimale si riduce in modo significativo.

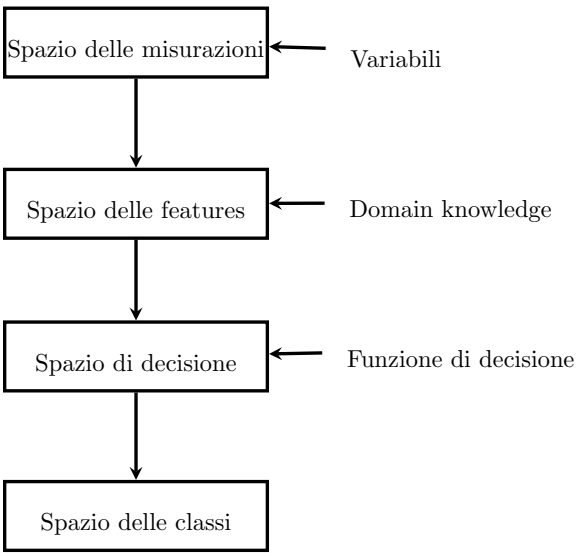

Figura 2.2: Trasformazione dei dati in un sistema di fault detection.

2.1.3 Model-based fault detection

La *model-based fault detection* si basa su un modello descrittivo del funzionamento del sistema. La natura di questo modello si può differenziare in modo significativo a seconda del contesto di applicazione, dei dati disponibili, della complessità del processo e dei fenomeni fisici coinvolti [1] [15] [5] [16] [17] [18] [19]. A più alto livello si identificano due macro categorie: la fedele rappresentazione del sistema fisico sottostante, tramite un insieme di equazioni tra variabili [2], oppure la formalizzazione di alcune relazioni di causa-effetto [3].

L'uso di un modello ha alcuni significativi vantaggi:

- Precisa rappresentazione e comprensione del funzionamento del sistema.
- Semplicità nell'individuazione della natura dell'anomalia.
- Semplicità nel definire procedura per la fault recovery.

Per contro, vi sono alcuni importanti svantaggi:

- Costruzione del modello complessa e costosa.
- Sistema di fault detection rigido, poco adattabile a modifiche.

Modelli quantitativi

Si definiscono quantitativi i modelli in cui il processo è rappresentato tramite equazioni matematiche. L'elevata precisione e dettaglio di questo approccio

permette di utilizzare la *ridondanza analitica*, ossia calcolare le misurazioni attese per un sensore in base ai dati di altri sensori. Possibili tecniche comprendono *parity relations* e filtri di Kalman. La discrepanza tra il valore effettivo e quello atteso è chiamata *residuo*. I residui possono essere utilizzati per identificare l'avaria ed adottare azioni correttive. Grazie all'uso di un modello ai residui si può associare un preciso significato fisico, il che semplifica notevolmente la fase di individuazione dell'avaria [5]. E' possibile sviluppare degli *osservatori* ciascuno addestrato per classificare i residui rispetto ad una specifica avaria, sfruttando la tecnica degli *ensemble classifiers* [20]. La natura non lineare di molti processi costituisce una importante problematica. Ulteriore forte limitazione all'uso di modelli quantitativi è la rapida crescita nella complessità del modello all'aumentare del numero di variabili, che ne rende troppo onerosa l'applicazione in sistemi complessi quali ad esempio gli impianti industriali, per i quali si tendono a preferire approcci differenti.

Modelli qualitativi

Un approccio alternativo fa uso di modelli qualitativi che rappresentano la conoscenza del processo in modo meno rigido, facendo uso di tecniche legate all'intelligenza artificiale.

Sistemi esperti: Alla knowledge base è fornita tutta la domain knowledge disponibile. Nel problema della fault detection solitamente lo spazio di funzionamento nominale è ben definito, quello di funzionamento anomalo no. Da ciò deriva la necessità di un processo di apprendimento basato sui fallimenti nell'identificazione dell'avaria che ha causato il guasto rilevato. L'incapacità di riconoscere correttamente un'avaria può portare ad una modifica della knowledge base. Solitamente quando un fatto è dedotto dalla knowledge base, esso si considera sempre vero. Tuttavia, essendo il sistema adattativo, in questo caso è possibile che una deduzione divenga falsa in presenza di ulteriori informazioni. La pocedura di ragionamento è *non monotona* [7].

Digrammi causali: Un'altra possibilità è applicare *modelli causali* espressi con digrammi rappresentanti le relazioni causa-effetto esistenti tra le varie componenti [6]. In questo modo si può rappresentare graficamente il processo in modo qualitativo e definire un guasto come un cambiamento nella struttura del digramma. L'approccio offre anche strumenti per implementare un'efficiente diagnostica grazie alla disponibilità di informazioni ben strutturate.

Alberi: Un albero è strutturato in modo che la radice corrisponda ad un guasto e le foglie a delle avarie. Ogni nodo intermedio svolge un'aggregazione

tramite un operatore logico. L'albero è costruito dalla radice verso le foglie, scomponendo di volta in volta un evento nelle sue possibili cause [21]. E' richiesta una dettagliata domain knowledge. All'albero così costruito sono poi applicati algoritmi quantitativi per determinare quali siano i set minimi di avarie tali da causare un guasto, ossia i componenti critici.

Gerarchie di astrazione: Il principio base è la decomposizione del processo o del sistema in sottosistemi, in quanto si assume sarà possibile inferire il comportamento del sistema nel suo complesso tramite il comportamento dei suoi sottosistemi. Non possono essere rappresentati legami tra i sottosistemi, quindi le regole applicate ad uno di essi non possono presumere che altri si comportino in un certo modo [22]. La scomposizione può essere sia *strutturale*, riferita a come le componenti sono collegate tra loro, che *funzionale*, ossia rappresentare il legame tra input ed output del sottosistema. Tra le possibili tecniche si menzionano i *multilevel flow models*, MFM [23].

Strategie di ricerca

In presenza di un guasto, per poter identificare l'avaria è necessario applicare una strategia di ricerca nello spazio degli stati di funzionamento del sistema previsti dal modello disponibile. Esistono due tipi di approcci: topografico e sintomatico.

La *ricerca topografica* si basa su un template dell'operatività nominale del sistema. E' quindi sufficiente sia ben definita la regione di funzionamento nominale, evitando di dover definire quella di comportamento anomalo. Questo tipo di ricerche sono robuste rispetto a guasti nuovi non previsti in fase di progettazione del sistema FDIR. Le modalità di implementazione in un caso specifico dipenderanno dal fatto che il modello sia strutturale o funzionale.

La *ricerca sintomatica* adotta un approccio opposto, prevedendo un set di template per sintomi di avarie note. E' necessario quindi disporre di un modello sufficientemente comprensivo dello spazio delle anomalie. Il vantaggio principale è una ricerca computazionalmente più semplice, a costo di una scarsa accuratezza nell'individuazione di comportamenti anomali nuovi e della difficoltà a definire in modo esaustivo una regione anomala potenzialmente molto vasta.

2.1.4 Data-driven fault detection

La *data-driven* o *history-based fault detection* sta avendo notevole successo grazie all fatto che non richiede sia fornito un modello del sistema. Esso viene costruito sulla base di dati disponibili, necessari in grande quantità. Tale approccio si propone quindi di ottenere una rappresentazione del sistema o del suo funzionamento nominale tramite i dati che esso ha effettivamente

prodotto [24] [4]. In processi industriali possono esistere centinaia o migliaia di variabili, ciascuna con delle proprie caratteristiche, tra le quali sussistono complesse correlazioni. Per poter ottenere dei risultati significativi sono necessari una gran quantità di dati per ciascuna di esse. Per rendere il problema trattabile ricorre a tecniche di riduzione della dimensionalità per individuare le feature di interesse. Si veda la Figura 2.2. Con il dataset così modificato è possibile sviluppare un modello nominale di funzionamento da confrontare con lo stato corrente. Una deviazione da tale modello costituirà un'anomalia.

L'approccio è per certi versi complementare rispetto a quello model-based, i punti di forza di uno tendono a tradursi in punti di debolezza per l'altro.

- Costruzione del modello più semplice.
- Poca domain knowledge necessaria.

Per contro, vi sono alcuni importanti svantaggi:

- Imprecisa rappresentazione del sistema.
- Difficile individuare la natura dell'anomalia.
- Significativa complessità per definire procedura per la fault recovery.

Esistono numerose diverse tecniche per implementare un sistema FDIR data-driven. Di seguito si presentano ad alto livello alcune famiglie di metodi, per le quali spesso esistono diverse varianti legate ad adattamenti per uno specifico processo o impianto.

Sistemi esperti

Le componenti fondamentali di un sistema esperto sono due: la knowledge base nella quale confluiranno le informazioni derivate dai dati, e la procedura diagnostica che sarà utilizzata per il ragionamento. I sistemi esperti sono utilizzati per diversi motivi: facilità di applicazione, ragionamento trasparente, robustezza all'incertezza e disponibilità di una dettagliata spiegazione per la soluzione proposta. Il sistema esperto per prima cosa classifica il guasto, poi può offrire funzioni di recovery per ripristinare il processo. Esistono numerosi esempi di applicazioni di sistemi esperti [8] [9] [25]. I sistemi esperti sono anche stati integrati con altre tecniche in approcci ibridi. Ad esempio una rete neurale che identifica i guasti più comuni legati ai sottosistemi e poi, al loro interno, un sistema esperto raffina la diagnosi [26]. Sono stati proposti anche approcci che utilizzano grafi diretti e fuzzy logic [27] [28]. Utili nel caso in cui siano disponibili poche informazioni quantitative ma sufficiente esperienza, sono limitati sia dalla loro forte specializzazione che dalla capacità espressiva. Sono stati proposti approcci ibridi che integrano nella

knowledge base della domain knowledge per assistere nella fase di diagnosi del guasto. Il sistema però rimale limitato a variabili dal comportamento semplice [29].

Analisi di segnale

La fault detection può essere effettuata interpretando le misurazioni come delle serie temporali ed analizzandole tramite trasformate wavelet o di Fourier. Cambiamenti nell'ampiezza o nella frequenza delle componenti principali possono significare la presenza di un guasto [30]. Questa tecnica è applicata nello *statistical process control*, SPC.

Feature extraction statistica

In un contesto reale il processo o sistema di interesse può essere soggetto a dei disturbi con una componente casuale. Le misurazioni sono pertanto considerate delle serie temporali stocastiche delle quali si vuole scoprire la distribuzione. Il comportamento nominale del sistema può quindi essere rappresentato tramite la distribuzione delle misurazioni e i suoi parametri. Qualora i parametri relativi alle misurazioni in un dato istante fossero diversi da quelli nominali, si è in presenza di un'anomalia [31]. In un contesto reale, inoltre, le misurazioni prodotte da diversi sensori non sono tra loro scorrelate. Un approccio statistico univariato non permette di modellare questo tipo di relazioni e quindi è spesso inadeguato. E' possibile mitigare il problema riducendo la dimensionalità dei dati allo scopo di ottenerne insiemi di dati tra di loro scorrelati. Molto utilizzati a questo scopo sono la *principal component analysis*, PCA [10] [11] e la *partial least squares regression*, PLS [12].

Sistemi multivariati

In un contesto reale ci si aspetta che le misurazioni si compongano di un alto numero di variabili tra loro molto correlate, essendo tutte parte dello stesso processo. I modelli multivariati [32] [33] stanno quindi avendo molto successo proprio in quanto riescono a rappresentare bene le relazioni tra le variabili, a fronte di un costo legato al design del sistema FDIR relativamente basso [4]. Si applicano tecniche per relazionare gli input ed output del processo, sono molto utilizzati sia PCA [34] [35] che PLS [36] [37].

Lo scopo principale del PCA è ridurre la dimensionalità dei dati, pertanto si scompone la matrice di covarianza lungo la direzione di massima variazione. Il procedimento è ripetuto in modo che ogni nuovo componente principale abbia un impatto sul risultato via via decrescente. Si osserva che già con un numero relativamente piccolo di componenti principali è possibile esprimere le variazioni delle misurazioni con un alto livello di precisione.

Per poter utilizzare questo approccio nella fault detection è necessario sviluppare un modello che rappresenti il funzionamento nominale, basato su dati storici, da confrontare con il funzionamento osservato in quel momento, basato su dati appena raccolti. I componenti principali sono calcolati allo stesso modo sui due dataset differenti. In seguito i componenti principali sono confrontati proiettando le nuove osservazioni sul piano definito dal PCA del modello nominale. A questo punto è possibile confrontare le componenti principali e determinare se esista o meno un'anomalia.

Una delle principali limitazioni della PCA è la sua invarianza sul tempo, aspetto incompatibile con un processo reale. Inoltre il PCA assume che i dati siano multivariati con distribuzione gaussiana, pertanto non è adatto a casi in cui il processo mostri significative non-linearità o distribuzioni non gaussiane. Per superare queste limitazioni sono state sviluppate delle versioni PCA dinamiche [38] e ricorsive [39]. Per gestire le non linearità il PCA è stato combinato con delle reti neurali che proiettano le misurazioni su un feature space ad alta dimensionalità, permettendo di applicare successivamente un PCA lineare [40]. Infine, nel caso in cui le distribuzioni non siano gaussiane è stato introdotto il PCA a minima entropia [41].

Alternativamente per dividere le misurazioni in combinazioni lineari di componenti indipendenti può essere applicata la *independent component analysis*, ICA. Non richiede che la distribuzione dei dati sia gaussiana ed è quindi meglio rappresentativa delle effettive dinamiche di processi reali. Può essere anche usato per separare i dati non-gaussiani dagli altri [42] [43].

Ulteriore possibilità, qualora siano disponibili delle relazioni tra l'input e output del processo, è applicare il metodo *partial least squares*. Anche di questo è disponibile una versione ricorsiva [44].

Classificazione statistica

Poichè la diagnosi di guasti è sostanzialmente un problema di classificazione, può essere affrontato con tecniche di classificazione statistica [45] [46]. Ad esempio con *classificatori di Bayes* sullo spazio di classi dato da tutti i possibili guasti. Ciò espone al problema di dover definire quali siano i guasti di interesse, richiedendo adeguata conoscenza del sistema e rendendo l'approccio rigido rispetto a nuovi guasti per i quali non sia stata prevista una classe.

Reti neurali

Le reti neurali sono spesso utilizzate a partire dallo spazio delle feature, in quanto in esso si mantiene la discriminazione tra le varie classi ma con una minore dimensionalità dei dati. Le reti neurali sono state applicate in una moltitudine di diversi contesti. Si riscontra che è possibile migliorare l'accuratezza della classificazione applicando data preprocessing. Un contributo

da menzionare è *Wavenet* [47], una rete neurale gerarchica nella quale uno dei livelli nascosti applica una funzione derivata da un set di wavelet ortonormali. Significativo vantaggio è la possibilità di aggiungere o rimuovere tali nodi senza riaddestrare la rete, sfruttando l'ortonormalità della trasformata wavelet.

Sistemi multiagente

Rispetto alle altre tecniche, l'uso di sistemi multiagente sembra assai meno diffuso e studiato. Un esempio di applicazione dei sistemi multiagente al problema di fault detection può essere *MADCABS* [13], un framework per il monitoraggio, l'analisi, la diagnosi e il controllo di processi. MADCABS è costituito da tre livelli: comunicazione fisica, supervisione del processo e gestione agenti. Il livello di comunicazione fisica si incarica dell'interazione col processo, raccogliendo le misurazioni e inoltrando le azioni di controllo. Il livello di supervisione del processo si compone a sua volta di altri sottolivelli:

- **Preprocessing -** I dati sono filtrati per eliminare rumore, outlier e stimare dati mancanti.
- **Monitoraggio -** Si verifica che i dati raccolti siano consistenti con il funzionamento nominale, si applicano modelli statistici multivariati PCA o DPCA. Esistono più agenti che monitorano, con metodi diversi, le stesse variabili.
- **Rilevamento -** In base ai risultati degli agenti di monitoraggio, determina quale sia lo stato operativo. Sono applicabili vari algoritmi per raggiungere un consenso.
- **Diagnosi -** In presenza di un'anomalia, si sfrutta conoscenza di processo per classificare il guasto ed individuare l'avaria.

Il livello di gestione agenti valuta le performance dei vari agenti e metodi di diagnosi, selezionando il metodo disponibile che sia più utile considerando le condizioni operative. L'idea di base è che gli agenti associati ad un metodo che non abbia riconosciuto un'anomalia ottengano uno score basso, quelli che l'hanno individuata correttamente invece ne ottengano uno alto. Il falso allarme è penalizzato meno rispetto ad un'anomalia non individuata. Nelle conclusioni di [13] è riportato che aggregare i risultati dei vari agenti, ottenuti con metodi diversi, produce un miglioramento dell'accuratezza complessiva rispetto all'utilizzo di un metodo solo.

2.2 Fault detection in applicazioni spaziali

I sistemi FDIR in applicazioni spaziali devono considerare, oltre alle problematiche già evidenziate, ulteriori fattori legati all'inaccessibilità del satellite così come al suo invecchiamento e deterioramento.

Similmente all'ambito industriale anche nel settore dei satelliti sono state applicate molteplici tecniche di fault detection, ad esempio i fuzzy set [48] e modelli neuro-fuzzy per sistemi con dinamiche ad alta frequenza e forti interdipendenze [49].

Da menzionare il framework *Advanced FDIR, AFDIR* [50], che offre agli sviluppatori di applicazioni FDIR satellitari model-based una gran varietà di strumenti. Tra i metodi di fault detection sono inclusi, ad esempio, filtri di Kalman, parity-vector, generalized likelihood test [51], random sample consensus [52] ecc... Relativamente ai metodi di diagnosi, ossia quelli utilizzati per risalire alla causa del guasto, AFDIR implementa reti Bayesiane e modelli causali, tecniche ancora non molto utilizzate in ambito spaziale.

In ambito satellitare i metodi di recovery, ossia di ripristino del satellite in uno stato nominale, tendono ad essere conservativi. Si preferisce disattivare la componente in avaria, attivandone una ridondante, o disattivando un intero sottosistema. Il satellite entra quindi in una modalità *fail-safe*. Il framework AFDIR propone tecniche più orientate al *fail-operational*, ossia al mantenere operativo il satellite per quanto possibile.

L'approccio model-based ha trovato applicazione anche nel progetto *Smart-FDIR* [53], nel quale è stato applicato il *fuzzy inductive reasoning*. Come per AFDIR, anche in questo caso il sistema sfrutta le relazioni causali esistenti tra variabili applicando un approccio misto quantitativo-qualitativo.

Un approccio più avanzato al problema della fault recovery è stato proposto con il progetto *Anomaly Resolution and Prognostic Health management for Autonomy*, ARPHA [54], pensato per operare autonomamente a bordo di un rover per l'esplorazione di pianeti. Nuovamente si utilizzano reti Bayesiane con l'eventuale ausilio di fault-tree, il che, come menzionato nella Sezione 2.1.3, permette di avere a disposizione informazioni strutturate e quindi di implementare tecniche di recovery più sofisticate. In questo sistema è anche applicata una forma di prevenzione, in quanto è possibile prevedere anomalie o avarie imminenti ed intervenire con opportune policy.

Si può inoltre menzionare un sistema data-based *Anomaly Monitoring Inductive Software System*, AMISS [55], che fa uso del tool *Inductive monitoring systems*, IMS [56]. I dati di funzionamento nominale sono analizzati e clusterizzati, in modo da estrarre informazioni sulle interazioni nominali delle variabili. Tali informazioni sono poi inserite in una knowledge base. Questo approccio è stato utilizzato per un sistema FDIR applicato a vari sottosistemi della stazione spaziale internazionale. Si tratta però di un sistema che richiede siano sviluppati dei moduli ad hoc per ogni sottosistema, risultando complesso e costoso. Del tool IMS è stata sviluppata una versione successiva chiamata *Outlier Detection Via Estimating Clusters*, OVDEC [57] che, utilizzando un set di dati molto più ampio, è in grado di offrire prestazioni migliori.

2.3 Sistema multiagente

Nel campo dell'intelligenza artificiale si definisce *agente razionale* un'entità in grado di interagire con un ambiente, potendone sia percepire che modificare lo stato, e in grado di prendere autonomamente decisioni in base ad un obiettivo descritto da una funzione di utilità [58].

Con *sistema multiagente* [59] si intende un sistema composto da un gruppo di agenti che interagiscono tra di loro. Semplificando, e interazioni possono essere di due tipi:

- **Cooperative -** gli agenti cercano di massimizzare l'utilità globale del sistema, a prescindere dalla propria.
- **Competitive -** gli agenti cercano di massimizzare la loro utilità a prescindere da quella degli altri agenti e del sistema.

Considerando nello specifico un sistema multiagente cooperativo, per implementarlo è necessario definire le modalità di interazione tra gli agenti:

- **Distribuzione dei task -** come il problema globale è scomposto e cosa ogni agente deve fare. Un esempio può essere il protocollo *contract net* [60].
- **Coordinamento -** come i risultati ottenuti dagli agenti sono uniti.
- **Comunicazione -** che approccio e protocollo è utilizzato per far comunicare tra loro gli agenti, ad esempio ACL [61].

Per ciascuno di questi aspetti esistono diverse possibili alternative, che possono adattarsi o meno al contesto applicativo.

Relativamente al coordinamento, si possono menzionare il voting e la negoziazione cooperativa.

Voting: ciascun agente, avendo a disposizione un set di candidati, esprime la propria preferenza. Il vincitore è determinato nella successiva fase di conteggio dei voti. Si menzionano in particolare:

- **Voto a pluralità -** ogni agente esprime la propria preferenza per un solo candidato. Vince quello che ha ottenuto il maggior numero di voti. Il meccanismo è semplice da implementare ma, a causa della frammentazione dei voti in presenza di numerosi candidati, è possibile il vincitore sia sgradito alla maggior parte degli agenti.
- **Borda count -** ogni agente vota definendo un ordinamento tra tutti i candidati in base alla sua preferenza, attribuendo in questo modo a ciascuno di essi un voto pesato in base alla loro posizione. Vince il candidato che ha ottenuto il punteggio maggiore. Il metodo è più equo rispetto al voto a pluralità, ma è più complesso da implementare.

Negoziazione cooperativa: ogni agente propone un'offerta che massimizza la sua utilità. Le offerte sono raccolte da un agente mediatore che le aggrega proponendo un risultato. Gli agenti intenti a negoziare a questo punto possono proporre una controfferta. Per garantire la negoziazione raggiunga la convergenza si applica il *monotonic concession protocol* [62], il quale richiede che un agente determini la sua controfferta in modo che gli altri agenti la trovino preferibile alla precedente. Dopo un sufficiente numero di iterazioni l'agente mediatore proporrà un risultato che tutti gli agenti accetteranno.

In questa tesi l'offerta degli agenti è una probabilità di anomalia e si vuole attribuire maggiore importanza agli agenti che abbiano rilevato una probabilità elevata. Il risultato proposto dal mediatore peserà le offerte degli agenti in base ad uno specifico peso W. Chiamata p_i l'offerta dell'agente i, il suo peso è definito come:

$$W_i : p_i \mapsto [0, 1]$$

$$W_i\left(p_i\left(t\right)\right) = \begin{cases} w_{min}, & \text{se } 0 < p_i(t) < p_{min} \\ w_{min} + \frac{w_{max} - w_{min}}{p_{max} - p_{min}} \cdot \left(p_i(t) - p_{min}\right), & \text{se } p_{min} \leq p_i(t) \leq p_{max} \\ w_{max}, & \text{se } p_{max} < p_i(t) < 1 \end{cases}$$

I parametri w_{min}, w_{max}, p_{min} e p_{max} possono essere adattati a seconda delle necessità. Attualmente w_{min} e p_{min} sono fissati a 0.1, w_{max} e p_{max} a 0.9.

L'offerta del mediatore è quindi definita come segue:

$$a(t) = \frac{\sum_{i=1}^{n} p_i(t) \cdot W_i}{\sum_{i=1}^{n} W_i}$$

Gli agenti potranno decidere di accettare la proposta del mediatore, o optare per una controfferta calcolata come segue:

$$F_i(p_i(t), a(t)) = p_i(t) \cdot \alpha_i \cdot (a(t) - p_i(t))$$

Con α_i un parametro di sensibilità così calcolato:

$$\alpha_i = 1 - \exp(W_i - \bar{W})$$

$\bar{W}$ è la media dei pesi calcolati in base alle offerte di tutti gli agenti.

Capitolo 3

Impostazione del problema di ricerca

In questo capitolo si presenta il contesto applicativo in cui si inserisce il progetto sviluppato in questa tesi. Nella prima parte si descrive il satellite Cryosat-2 e la sua strumentazione. Nella seconda parte si presenta e il gruppo che si occupa del controllo della missione da terra, gli strumenti attualmente a sua disposizione e si presentano gli obiettivi di questa tesi. Nella terza parte si descrivono le caratteristiche delle anomalie che il sistema potrà individuare. Nella quarta parte si presenta una formalizzazione del problema e nella quinta si descrivono i diversi possibili spazi delle soluzioni. Infine si presentano gli apporti innovativi di questo progetto.

3.1 Caso applicativo Cryosat-2

Cryosat-2 [63] è una missione ESA lanciata l'8 aprile 2010 a seguito del fallimento del lancio della precedente versione nel 2005. L'obiettivo di questo satellite è misurare l'estensione e lo spessore dei ghiacci polari, viaggiando in orbita bassa. Il satellite opera come un altimetro radar ed è in grado di raggiungere una precisione ragguardevole, 2 cm nel caso dei ghiacci artici e 2 mm per quelli dell'Antartide.

Per svolgere le sue attività il satellite è dotato di diversi strumenti tra i quali se ne citano due: SIRAL e DORIS.

SIRAL: *SAR/Interferometric Radar Altimeter* [64], ossia l'interferometro radar incaricato di effettuare la misurazione della posizione e spessore del ghiaccio. Per raggiungere l'elevata precisione richiesta, durante le fasi di misurazione la velocità, posizione e l'orientamento del satellite sono accuratamente monitorate. Lo strumento ha tre modalità di funzionamento:

- **Low-resolution mode -** Sono inviati a terra impulsi ad una frequenza tale che gli echi non si sovrappongano. In questo modo la risoluzione

delle misurazioni è bassa e viene prodotta una limitata quantità di dati da elaborare e trasmettere a terra. Questa modalità è adatta per oceani e superfici ghiacciate uniformi.

- **Synthetic Aperture Radar mode -** E' inviato a terra un treno di impulsi molto ravvicinati. L'eco di un impulso raggiunge il satellite quando ormai altri impulsi sono stati trasmessi. Si sfrutta l'effetto doppler per separare i segnali ricevuti associando ad ogni impulso il corrispondente eco. Grazie a questa tecnica sono disponibili un grande numero di misurazioni indipendenti per una specifica area, permettendo di raggiungere un'accuratezza elevata su superfici frastagliate.

- **SAR-Interferometric mode -** Si attiva una seconda antenna inclinata rispetto alla prima. Se l'eco che raggiunge il satellite non è stato riflesso da una superficie perfettamente orizzontale, la sua direzione farà si che la fase del segnale rilevato dalle due antenne sarà leggermente diversa. La lieve differenza di fase permette di calcolare la pendenza della superficie ghiacciata. Questa modalità produce una gran quantità di dati da elaborare ed è utilizzata solo in presenza di superfici a pendenza elevata o ai confini dei ghiacci.

DORIS: *Doppler Orbit and Radio Positioning Integration by Satellite* [64], responsabile di misurare la distanza del satellite da terra, sfruttando l'effetto doppler misurato su segnali provenienti da 50 stazioni trasmittenti a terra. Una misura indipendente della posizione è effettuata da terra grazie all'*International Laser Ranging Service*. Il satellite è dotato di un apparato per riflettere i raggi laser provenienti dalla superficie esattamente nella stessa direzione dalla quale sono arrivati, in questo modo da terra è possibile validare la posizione che il satellite ha stimato.

La comunicazione è assicurata grazie ad alcune stazioni sulla Terra. Si utilizza il protocollo *Space Link Extension* [65], sviluppato ad hoc per applicazioni spaziali in un'ottica di interoperabilità e standardizzazione, che supporta l'invio di dati tramite protocollo TCP/IP.

3.2 Flight Control Team

Il gruppo che si occupa da terra di gestire il satellite, ricevendone la telemetria, pianificando la missione ed inviando i comandi è chiamato *Flight Control Team*, FCT. Poichè i satelliti sono apparati complessi, il FCT è suddiviso in sotto gruppi che collaborano tra di loro, ciascuno incaricato della supervisione di un diverso sottosistema. Il FCT deve quindi avere una conoscenza dettagliata del satellite e del suo funzionamento.

I dati a disposizione del FCT provengono da diverse fonti:

- Log forniti da sistemi a terra.

- Eventi generati dal satellite.
- I dati della telemetria ed i telecommand.

3.2.1 Stazioni di terra

Le stazioni di terra fanno parte di una rete con diversi sistemi interconnessi. Le *Ground Station* si occupano di ricevere e trasmettere fisicamente dati dal satellite. Il *Control Center* invece è composto da diversi sistemi, tra i quali figurano:

- **Mission Control System -** Componente incaricato del monitoraggio dello stato del satellite, esso analizza i dati inviati dal satellite e verifica la loro consistenza. Si occupa inoltre dell'invio di comandi.
- **Network Interface System -** Coordina il trasferimento di dati tra le ground station ed il MCS. Sovrintende inoltre alla corretta comunicazione tra il satellite e terra verificando la qualità dei link.

3.2.2 Flight Control Team Multi Agent System

Tra le attività di routine del FCT vi è la verifica del corretto funzionamento delle varie componenti del satellite, attività soggetta ad un problema di information overload. Il satellite ha un grande numero di sensori e produce quindi una gran quantità di dati, la cui gestione e corretta interpretazione sono difficili da effettuare manualmente in modo esaustivo, soprattutto considerando il breve tempo che può intercorrere tra l'insorgere di un'anomalia ed un'avaria. Per supportare il FCT in questa attività è stato proposto lo studio di uno strumento chiamato *Flight Control Team Multi Agent System*, FCTMAS [14], incaricato di analizzare i dati grezzi provenienti dal satellite ed estrarre informazioni relativamente al suo stato di funzionamento. Nello specifico, dovrà segnalare al FCT se un componente del satellite sta esibendo un comportamento anomalo e non sta operando normalmente.

Il sistema FCTMAS è stato sviluppato con un'architettura multiagente, scelta in virtù della flessibilità che conferisce in merito all'aggiunta o rimozione di agenti. Esistono tre tipologie di agenti:

- **Agente estrattore -** Si occupa di leggere i log del satellite estraendo i dati.
- **Agente diagnostico -** Utilizza i dati inviati dall'estrattore per stimare quanto il comportamento di una variabile sia compatibile con il suo modello di funzionamento nominale, restituendo in output una probabilità di anomalia. I modelli utilizzati per rappresentare il comportamento nominale sono catene di Markov o reti neurali.

- **Agente mediatore -** Si occupa di coordinare gli agenti diagnostici unendone i risultati. Sono applicati due diversi metodi, la negoziazione cooperativa ed il voting.

Il sistema multiagente è sviluppato utilizzando il framework *Java Agent DEvelopment*, JADE [66], una piattaforma per lo sviluppo di applicazioni peer-to-peer basate su agenti. JADE è stato scelto per la sua elevata interoperabilità e l'utilizzo degli standard di comunicazione FIPA, *Foundation for Intelligent Physical Agents*.

L'approccio utilizzato per la costruzione dei modelli è data-based, si veda la Sezione 2.1.4. La scelta è legata al fatto che costruire un modello di tipo quantitativo è complesso e costoso, come discusso nella Sezione 2.1.3. Ogni agente diagnostico è addestrato a riconoscere il comportamento nominale di una variabile, utilizzando dei dati relativi al funzionamento del satellite. Il sistema sviluppato supporta l'aggregazione dei risultati degli agenti diagnostici secondo strutture ad albero arbitrarie, è quindi possibile configurare diverse aggregazioni in modo da meglio rappresentare la struttura dei sottosistemi del satellite. Allo stato attuale le aggregazioni possono essere configurate solo manualmente.

Modelli nominali e probabilità di anomalia

I modelli di comportamento nominale per le variabili utilizzate nel sistema FCTMAS [14] si compongono di due parti, una legata all'andamento dei valori misurati ed una legata alla distribuzione temporale delle rilevazioni. Nel caso di variabili numeriche si utilizzano reti neurali [67] o modelli ARMA [68] per predire il valore della prossima occorrenza. Nel caso di variabili categoriche invece si utilizzano catene di Markov [69] sui cui archi è indicato quanto probabile sia la transizione da uno stato ad un altro.

Tali modelli sono addestrati a partire da un log di funzionamento nominale del satellite.

Una volta addestrati i modelli è necessario definire come calcolare la probabilità di anomalia di una variabile.

La probabilità di anomalia associata alla variabile i è definita come:

$$p_i(t) : \{\hat{y}_i(t); y_i(t)\} \mapsto [0, 1]$$

Dove $\hat{y}_i(t)$ è il valore stimato dal modello e $y_i(t)$ è il valore effettivamente misurato all'istante t.

Nel caso di variabili numeriche si utilizza una funzione simile a quella della densità Gaussiana:

$$p_i(t) = 1 - \exp\left[-\frac{1}{2} \cdot \left(\frac{\epsilon(t) - \bar{\epsilon}}{\sigma}\right)^2\right] \tag{3.1}$$

Dove $\epsilon(t) = |\hat{y}_i(t) - y_i(t)|$, ossia l'errore di predizione, $\bar{\epsilon}$ è la media campionaria dell'errore e σ la varianza campionaria dell'errore entrambe calcolate durante l'addestramento.

Per i modelli categorici invece la probabilità di anomalia è stabilita a priori in base alla precisione del modello e alla regolarità della variabile considerata. In questo modo qualora una variabile mostri un comportamento molto regolare il modello restituirà immediatamente una probabilità di anomalia del 100% in caso di un cambiamento di stato non nominale. La probabilità di anomalia cresce progressivamente all'aumentare del numero di valori anomali consecutivi [14].

3.2.3 Aggregazione gerarchica

Il sistema FCTMAS nella sua originale implementazione non tiene in considerazione nè le correlazioni esistenti tra le variabili nè la suddivisione del satellite in sottosistemi. Questo costituisce una limitazione in quanto la classificazione dello stato di funzionamento è effettuata aggregando contemporaneamente gli output di tutti gli agenti diagnostici. In un caso reale questo significa combinare variabili legate a fenomeni fisici completamente diversi appartenenti a diversi sottosistemi. In questo modo è sì possibile individuare quando il satellite opera in regime anomalo, ma è più difficile individuare quale guasto si sia prodotto e quindi quale avaria possa averlo causato.

A supporto del FCT sarebbe più funzionale un sistema in grado di offrire informazioni sullo stato del satellite ad un livello di dettaglio più fine. In questo modo sarebbe possibile circoscrivere la causa delle anomalie ad un certo fenomeno fisico o la sua ubicazione in un certo sottosistema. E' quindi opportuno integrare il FCTMAS con un modulo che consenta di strutturare le aggregazioni degli agenti diagnostci componendo progressivamente modelli parziali del sistema sino ad ottenere un risultato globale.

In questa tesi si opererà avendo a disposizione dei modelli del funzionamento nominale di ogni variabile, addestrati a partire dai dati disponibili, che consideriamo come black box. A partire da essi si vuole determinare la struttura ottimale per una gerarchia di agenti che possa svolgere anomaly detection di un sistema quale il satellite Cryosat-2, caratterizzato da numerose variabili tra di loro correlate. Sono disponibili dati relativi solo al funzionamento nominale, si tratta quindi di un contesto semi-supervisionato. L'accuratezza del sistema non potrà essere confrontata con l'effettiva probabilità di guasto in quanto si hanno a disposizione solo gli eventi di guasto vero e proprio. Poichè si opera in assenza di domain knowledge, la struttura con la quale saranno aggregati i modelli parziali deve essere determinata tramite algoritmi di ricerca. Una volta individuata la struttura ottima, da essa si potranno estrarre informazioni relativamente alle relazioni tra variabili e scoprire l'esistenza di correlazioni.

3.3 Anomalie individuabili

Prima di avviarsi alla progettazione del sistema, occorre verificare quali tipologie di anomalie ci aspettiamo il satellite possa esibire e se i metodi di anomaly detection proposti riescano ad individuarle efficacemente. Il sistema FCTMAS addestra un modello per il comportamento nominale della variabile basandosi solo sui valori assunti dalla variabile stessa durante il normale funzionamento del sistema. Le anomalie possono essere classificate in diversi modi, a seconda delle caratteristiche di interesse. Nel nostro caso effettuiamo una prima suddivisione sulla base dell'orizzonte temporale che è necessario osservare per poterle individuare.

Istante di tempo Osservando lo stato delle variabili ad un istante di tempo indipendentemente dagli altri è possibile riconoscere le seguenti anomalie:

- **Puntuali** - la variabile ha un range nominale, dal quale è uscita.
- **Contestuali** - una variabile ha diversi range nominali sulla base di un contesto, definito tramite altre variabili. Esistono delle relazioni di causalità e quindi i reciproci valori sono correlati. Ad esempio la temperatura di un componente o la potenza generata dai pannelli solari variano sulla base della posizione e dell'orientamento del satellite. Gli stessi valori possono essere nominali se il satellite si trova esposto alla luce del Sole ma anomali se esso si trova nella zona d'ombra della Terra.

Le anomalie puntuali così come quelle contestuali, se legate al tempo, dovrebbero essere adeguatamente catturate dai modelli nominali della variabile, poiché essi già contengono informazioni sui range e sugli andamenti nel tempo. Le anomalie contestuali non legate al tempo, invece, potrebbero non essere quasi mai riconosciute. Ciò è causato dal fatto che la variabile andrebbe considerata in relazione ad un insieme di altre variabili e l'attuale processo di costruzione degli agenti considera ogni variabile in modo indipendente. Per riuscire a definire il "contesto" sarebbero necessari o una adeguata domain knowledge o un dataset sufficientemente ricco da permettere un'analisi statistica significativa per individuare associazioni esistenti tra i comportamenti delle variabili. Il problema che affrontiamo è caratterizzato dalla mancanza di domain knowledge e da dataset non sufficientemente estesi. Per sopperire alla mancanza del dataset, potrebbe essere comunque possibile integrare il sistema con un metodo di apprendimento che operi online e consenta di effettuare *novelty detection*. Riconoscendo nell'anomalia contestuale un nuovo comportamento non previsto dal modello, con il contributo di un esperto la si può integrare nel modello stesso. Si tratta di una forma di apprendimento basata su errori. Avendo la possibilità

di individuare la struttura ottima, da essa si possono ricavare informazioni relativamente alla dinamica del sistema e quindi potenzialmente ricavare spunti per determinare il contesto, ma ciò richiede comunque un intervento da parte di personale specializzato. A questo punto deve essere identificata un modalità per integrare nel sistema il riconoscimento delle anomalie contestuali, ad esempio con ulteriori agenti diagnostici, al costo di un possibile aumento drastico nel loro numero.

Intervallo di tempo Alcuni tipi di anomalie richiedono di analizzare l'andamento per un certo intervallo di tempo:

- **Collettive** - la variabile esibisce una sequenza di valori che non sono anomali di per sé, considerandone ciascuno indipendentemente, ma lo sono se occorrono in sequenza. Questo tipo di anomalia può presentarsi, ad esempio, poiché una certa sequenza di valori rappresenta un'alterazione al normale pattern o è noto, tramite domain knowledge, che una specifica sequenza sia associata ad una particolare anomalia.

Le anomalie collettive legate all'alterazione del normale pattern dovrebbero essere adeguatamente riconosciute dai modelli nominali delle variabili. Più difficile invece è riconoscere le anomalie relative ad una specifica sequenza di valori, ossia quelle che richiedono una precisa conoscenza del funzionamento del satellite, in quanto nei dati utilizzati per addestrare i modelli esse potrebbero non comparire.

3.4 Formalizzazione

Di seguito si propone una possibile formalizzazione del problema di ricerca della struttura gerarchica ottima.

3.4.1 Formalizzazione di alto livello

Questo tipo di formalizzazione non richiede alcuna assunzione in merito a quale strategia si utilizzerà per risolvere il problema, si fa uso soltanto di un set di funzioni a rappresentare ciascun cluster di agenti che saranno aggregati:

- $i \in [1, a_{tot}]$, indice univocamente associato ad un agente.
- $u_i(t) \in [0, 1]$, funzione che rappresenta il valore assunto dalla variabile i al tempo t.
- $a_i(t) : u_i(t) \mapsto p$ con $p \in [0, 1]$, funzione associata all'agente A_i. Sulla base dei valori che ha assunto determina il relativo coefficiente di anomalia.

- $j \in [1, f_{tot}]$, indice associato univocamente ad una funzione di aggregazione.
- $f_j(p_{b1}, p_{b2}, \ldots) : p^n \mapsto p$ con $n, b1, b2, \ldots \in [1, a_{tot} + f_{tot}]$ tale che $\forall b \in [1, a_{tot}], p_b = a_i$ e $\forall b \in [a_{tot} + 1, a_{tot} + f_{tot}], p_b = f_{b-a_{tot}}$, funzione di aggregazione. Essa può aggregare probabilità a prescindere dal fatto che provengano da agenti o da altre funzioni al livello gerarchico inferiore.
- $f_{f_tot}(\cdot)$ funzione di aggregazione più esterna, la probabilità calcolata rappresenta il risultato finale.
- $f_v = (f_j(\cdot), j = 1, ..., f_{tot})$ vettore contenente tutte le funzioni f_j, rappresenta la struttura gerarchica nel suo complesso.
- $f_v(t) : input \mapsto p^{f_{tot}}$, vettore contenente le probabilità associate ad ogni funzione dato un certo istante di tempo.
- $K(t) = k(f_v(t), u(t), v(t))$, cifra di merito che permetta di calcolare la bontà della predizione ottenuta tramite f_v dato un certo input $u(t)$ confrontandola con il risultato effettivo $v(t)$.

Individuare la soluzione ottima richiederà di costruire f_v e quindi tutte le f_j, avendo a disposizione $u_i(t)$, $a_i(t)$ e definita una opportuna funzione $k(f_v(t), u(t), v(t))$.

3.4.2 Formalizzazione tramite grafo

La formalizzazione generale può essere ricondotta ad un problema di ottimizzazione su grafo orientato, definito come segue:

- $G = (N, A)$, un grafo.
- $N = (n_z \mid z \in [1, a_{tot} + f_{tot}], \forall z \in [1, a_{tot}], n_z = a_z$ e $\forall z \in [a_{tot} + 1, a_{tot} + f_{tot}], n_z = f_{z-a_{tot}})$, un nodo per ogni agente ed ogni funzione di aggregazione.
- $n_{a_tot+f_tot}$ unico nodo del grafo avente solo archi entranti, rappresenta il risultato finale.
- $(n_x, n_y) \in A \Leftrightarrow n_x, n_y \in N, \exists f_y(\ldots, p_x, \ldots)$, un set di archi che contenga un arco ogniqualvolta il risultato di un agente o una funzione sia necessario ad un'altra funzione.
- $f_j(\cdot)$ funzione di aggregazione per i nodi costituenti il suo taglio entrante.
- $K(t) = k(G, u(t), v(t))$, cifra di merito definita sul grafo.

Sono dati i nodi $n_z, z \in [1, a_{tot}]$. Individuare la soluzione ottima richiederà di determinare quali archi debbano essere introdotti e quante funzioni di aggregazione, ossia nodi $n_z, z \geq a_{tot}$, siano necessari.

3.5 Funzione di aggregazione e spazio delle soluzioni

Nel contesto di un sistema multiagente, la scelta della funzione di aggregazione utilizzata per l'unione dei risultati degli agenti diagnostici ha un impatto marcato sulla difficoltà del problema. Essendo una soluzione una struttura gerarchica di aggregazione dei risultati, a seconda della funzione di aggregazione e della cifra di merito utilizzate, può essere possibile definire una relazione di equivalenza tra due soluzioni differenti. In questo caso, lo spazio delle soluzioni non sarà più composto da tutte le strutture possibili, bensì dalle diverse classi di equivalenza. Nelle sezioni seguenti si vuole studiare l'impatto di una specifica funzione di aggregazione sullo spazio delle soluzioni. Si cercherà di fornire una procedura costruttiva che consenta di generare una soluzione sfruttando le caratteristiche di tale spazio.

3.5.1 Spazio delle strutture possibili

Poichè le soluzioni sono espresse sotto forma di albero o grafo, la massima dimensionalità dello spazio delle soluzioni si avrà considerando tutti i possibili alberi e grafi di un certo numero di nodi. Secondo la formula di Cayley il numero di alberi di supporto che si possono costruire su un grafo con $n > 1$ vertici è pari a n^{n-2}, il che rende il numero di possibili configurazioni rapidamente intrattabile. Si osservi che n sarà strettamente maggiore del numero di agenti diagnostici, poichè è necessario introdurre degli agenti mediatori.

Se la soluzione cercata è un grafo, il problema diventa ancora più complesso, in quanto il numero di possibili grafi etichettati di n nodi è pari a $2^{\binom{n}{2}}$.

	$n = 5$	$n = 10$	$n = 15$	$n = 40$
Albero	$1.2 \cdot 10^2$	$1.0 \cdot 10^8$	$1.9 \cdot 10^{15}$	$7.5 \cdot 10^{60}$
Grafo	$1.0 \cdot 10^3$	$3.5 \cdot 10^{13}$	$4.0 \cdot 10^{31}$	$6.3 \cdot 10^{234}$

Tabella 3.1: Numero di possibili soluzioni per dimensione del grafo

3.5.2 Funzione di aggregazione massimo

La funzione di aggregazione massimo avrà come risultato il valore massimo tra quelli associati agli argomenti. Poichè il risultato della funzione non è le-

gato all'ordine col quale gli argomenti sono considerati, calcolare il massimo di un insieme di valori e suddividere tale insieme in partizioni, calcolare il massimo per ciascuna di esse e poi aggregare nuovamente i valori, produce lo stesso risultato. Da ciò consegue che, dato il nodo di interesse, la struttura dell'albero di cui è radice o del sottografo di cui è pozzo sia ininfluente sul risultato. L'unica discriminante potrà essere l'eventuale componente della cifra di merito legata alla complessità della soluzione, la quale può essere ottimizzata in base al numero di nodi a prescindere dai dati disponibili nello specifico caso applicativo. Applicare unicamente questa funzione di aggregazione nel caso in cui si voglia ottenere un solo risultato complessivo per l'intero sistema è quindi privo di utilità. Qualora allo scopo di aumentare il livello di dettaglio si volesse affiancare il risultato complessivo a due o più risultati intermedi, calcolati in base a diversi insiemi di agenti diagnostici, la struttura interna torna ad avere un impatto. Il risultato calcolato dalla radice continuerà a non risentire della struttura, il risultato calcolato dai mediatori invece ne risentirà poichè è ottenuto aggregando un sottoinsieme degli agenti diagnostici. Più precisamente, dato un nodo intermedio, si tratta di ricavare quali foglie abbia il suo sottoalbero.

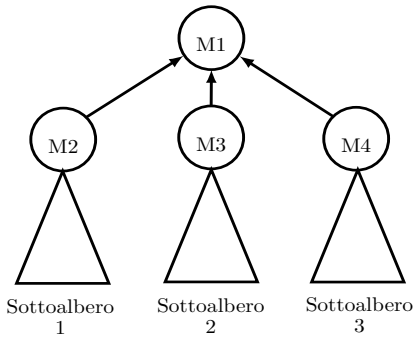

Figura 3.1: Struttura di una soluzione con tre nodi intermedi di interesse.

Posto che M1 è associato al risultato complessivo, supponiamo di voler utilizzare tre risultati intermedi, i quali sono associati ai tre agenti mediatori M2, M3 e M4, ciascuno con un sottoalbero. Si veda la Figura 3.1. Se essi si trovano sullo stesso livello, le soluzioni compatibili con questa configurazione sono date dalle possibili ripartizioni non ordinate degli agenti reali tra i tre sottoalberi. Posto a 2 il minimo numero di foglie per sottoalbero ed $n \geq 6$ il numero degli agenti reali, nel primo sottoalbero si dovranno considerare tutte le combinazioni di almeno due agenti e al massimo $n - 4$.

Per ciascuna di esse, si dovranno similmente considerare le possibili ripartizioni degli agenti rimanenti tra secondo e terzo sottoalbero. Poichè nel terzo confluiscono tutti quelli non utilizzati nel primo e nel secondo, la sua configurazione è univocamente determinata da quella dei primi due. Siano f_1 ed f_2 il numero di foglie del rispettivo sottoalbero. Le possibili foglie per il secondo albero sono le combinazioni semplici senza ripetizioni degli agenti mediatori rimanenti, quindi $n - f_1$, con un numero compreso fra 2 e $n - f_1 - 2$.

$$\sum_{f_1=2}^{n-4}\left(\binom{n}{f_1}\sum_{f_2=2}^{n-f_1-2}\binom{n-f_1}{f_2}\right)$$

	$n=10$	$n=15$	$n=20$	$n=25$	$n=30$	$n=40$
Albero	$4.0 \cdot 10^4$	$1.3 \cdot 10^7$	$3.4 \cdot 10^9$	$8.4 \cdot 10^{11}$	$2.0 \cdot 10^{14}$	$1.2 \cdot 10^{19}$
Grafo	$1.0 \cdot 10^9$	$3.5 \cdot 10^{13}$	$1.1 \cdot 10^{18}$	$3.7 \cdot 10^{22}$	$1.2 \cdot 10^{27}$	$1.3 \cdot 10^{36}$

Tabella 3.2: Numero di possibili soluzioni per numero di foglie nel caso di funzione di aggregazione massimo e tre sottoalberi.

Come si può osservare nella Tabella 3.2 l'uso di questa funzione di aggregazione riduce drasticamente la difficoltà del problema. La ricerca di soluzioni sotto forma di albero beneficia del fatto che il numero di agenti candidati ad entrare in un sottoalbero si riduce progressivamente. Per una soluzione sotto forma di grafo questo non avviene. L'impatto sulla dimensione massima trattabile è però contenuto, rimangono ingestibili casi con $n \geq 30$.

3.5.3 Funzione di aggregazione media

La funzione di aggregazione media avrà come risultato la media dei valori associati agli argomenti, quindi ai coefficienti di anomalia. La media presenta diverse complicazioni rispetto al massimo, in quanto la struttura dell'albero eha un impatto sul risultato. Come nel caso precedente si consideri la radice dell'albero come associata ad una predizione di interesse, ad esempio legata ad un solo sottosistema, e non necessariamente al risultato complessivo per l'intero sistema.

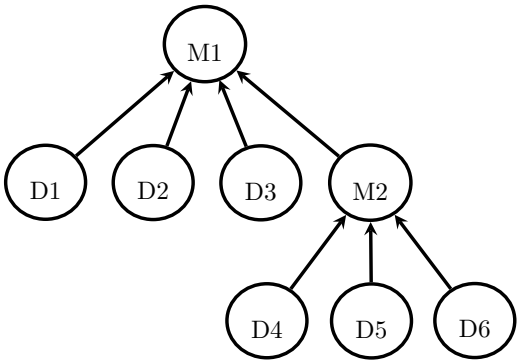

Figura 3.2: Una possibile struttura.

Consideriamo come esempio la configurazione in Figura 3.2. Per poter ottenere M1 occorre calcolare il valore degli agenti mediatori a partire da quelli più vicini alle foglie, in questo caso M2. Se l'agente di cui ci interessa il valore è solo M1, è possibile evitare di calcolare le aggregazioni seguendo la struttura dell'albero, utilizzando una media ponderata.

Nell'esempio in questione, avremo:

$$
\begin{aligned}
M2 &= \frac{D4 + D5 + D6}{3} \\
M1 &= \frac{D1 + D2 + D3 + M2}{4} = \frac{D1 + D2 + D3 + \frac{1}{3}\left(D4 + D5 + D6\right)}{4} \\
&= \frac{1}{4}D1 + \frac{1}{4}D2 + \frac{1}{4}D3 + \frac{1}{12}D4 + \frac{1}{12}D5 + \frac{1}{12}D6
\end{aligned}
$$

Il valore associato a M1 è quindi la combinazione lineare degli agenti diagnostici suoi discendenti. Chiamiamo w i coefficienti utilizzati nella combinazione lineare. Grazie a questa osservazione, possiamo fare due considerazioni:

- Focalizzando l'attenzione meramente sul risultato in $M1$, la soluzione è univocamente determinata dai coefficienti w.

- Il coefficiente w è il reciproco della produttoria del numero di fratelli dell'agente stesso e di ogni suo ascendente. Non dipende quindi dagli specifici ascendenti o fratelli del nodo. Pertanto esistono più soluzioni strutturalmente diverse, ma equivalenti. Si veda la Figura 3.3.

Individuare correttamente i set di soluzioni equivalenti può permettere di ridurre di diversi ordini di grandezza la complessità computazionale del problema, evitando di esplorare soluzioni che certamente non porteranno a risultati migliori. La ricerca della soluzione può essere eventualmente suddivisa in due fasi. Individuata una soluzione in base all'accuratezza, è possibile sceglierne una equivalente in base ad una seconda cifra di merito legata esclusivamente alla struttura.

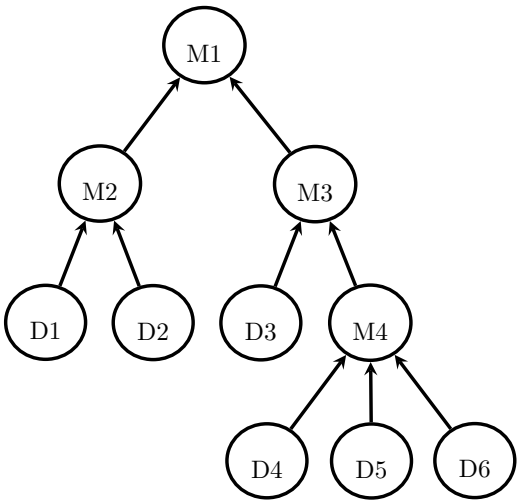

Figura 3.3: Soluzione strutturalmente diversa ma equivalente a quella in Figura 3.2.

Sebbene strutturalmente diversa, il risultato di $M1$ in questa soluzione è identico a quello della Figura 3.2.

$$
\begin{aligned}
M2 &= \frac{1}{2}D1 + \frac{1}{2}D2 \\
M4 &= \frac{1}{3}D4 + \frac{1}{3}D5 + \frac{1}{3}D6 \\
M3 &= \frac{1}{2}D3 + \frac{1}{2}M4 \\
&= \frac{1}{2}D3 + \frac{1}{6}D4 + \frac{1}{6}D5 + \frac{1}{6}D6 \\
M1 &= \frac{1}{2}M2 + \frac{1}{2}M3 \\
&= \frac{1}{4}D1 + \frac{1}{4}D2 + \frac{1}{4}D3 \\
&+ \frac{1}{12}D4 + \frac{1}{12}D5 + \frac{1}{12}D6
\end{aligned}
$$

Ulteriori 5 soluzioni equivalenti possono essere trovate sulla base dei diversi insiemi di 2 elementi formabili con i nodi D1, D2, D3 e M4, sullo stesso livello gerarchico. Poichè, rispetto alla funzione di aggregazione massimo, in questo caso le soluzioni sono distinte non solo dalla presenza di un certo agente in un sottoalbero, ma anche dal suo w, il numero di soluzioni esistenti sarà superiore.

Ammissibilità di una soluzione Definiamo $\bar{w}$ come il vettore contenente i w degli agenti diagnostici, secondo un qualsiasi ordinamento degli agenti stessi. Naturalmente non tutti i possibili $\bar{w}$ sono accettabili. Definiamo:

- **soluzione ammissibile** - un vettore $\bar{w}$ che possa essere generato da almeno una struttura. Ad esempio, con riferimento alla Figura 3.2:

$$\bar{w} = [4, 4, 4, 12, 12, 12]$$

- **struttura ammissibile per** $\bar{w}$ - una specifica struttura tale che i w associati agli agenti diagnostici siano corrispondenti a quelli contenuti in $\bar{w}$. Una struttura può essere rappresentata in diversi modi, direttamente sotto forma di albero oppure come sequenza ordinata dei fattori per ogni w. Dato un albero, esso appartiene allo spazio delle soluzioni se rispetta il seguente vincolo. Sia associato un coefficiente w anche agli agenti mediatori e posto $w_{radice} = 1$. Sia definita inoltre la funzione $\delta^+(nodo)$ che restituisce la cardinalità del taglio uscente a quel nodo, ossia il numero dei suoi figli. Per ogni nodo interno deve

valere la seguente relazione:

$$w_{padre} \cdot \delta^{+}(padre) = w_{figlio}$$

Soluzioni e strutture ammissibili possono essere espresse sotto forma di *constraint satisfaction problem.* Un CSP è definito da una tripletta $\langle X, D, C \rangle$, essendo X il set delle variabili, D il dominio per ciascuna variabile e C il set dei vincoli.

Constraint satisfaction problem In questo paragrafo propongo una formalizzazione sotto forma di CSP per il problema di ricerca di una soluzione ammissibile e per quello di ricerca di una struttura ammissibile data una soluzione.

La scelta della struttura può essere vista come un problema di allocazione dei figli agli agenti mediatori. Si può quindi esprimere tramite una matrice booleana S, si veda la Tabella 3.3. Sulle righe si troveranno gli agenti diagnostici e successivamente quelli mediatori, esclusa la radice. Sulle colonne si troverà la radice seguita da tutti gli altri agenti mediatori. La presenza di un 1 indicherà che l'agente in quella riga è figlio dell'agente in quella colonna.

	M1	M2	M3	M4
D1	0	1	0	0
D2	0	1	0	0
D3	0	0	1	0
D4	0	0	0	1
D5	0	0	0	1
D6	0	0	0	1
M2	1	0	0	0
M3	1	0	0	0
M4	0	0	1	0

Tabella 3.3: Matrice struttura per l'albero in Figura 3.3.

Posto d il numero di agenti diagnostici e m quello degli agenti mediatori,

si avrà:

$$\begin{aligned} Diagnostici &= 1, \ldots, d \\ Mediatori &= d+1, \ldots, d+m \\ Agenti &= Diagnostici \cup Mediatori \\ Colonne &= 1, \ldots, m \end{aligned}$$

Il taglio uscente è rappresentato in un vettore $\bar{\delta}^+$, calcolato per ogni colonna di S, sommando le righe:

$$\delta^+_{col} = \sum_{row \in Agenti} S(row, col) \quad \forall col \in Colonne \tag{3.2}$$

Ogni agente mediatore dovrà avere almeno due figli. Poichè è necessario il vincolo valga anche per la radice, lo si esprime sulle colonne.

$$\delta^+_{col} \geq 2 \quad \forall col \in Colonne \tag{3.3}$$

Si vuole che la matrice S rappresenti un albero, pertanto un qualsiasi agente non può avere più di un padre:

$$\sum_{col \in Colonne} S(row, col) = 1, \quad \forall row \in Agenti \tag{3.4}$$

Per evitare la costituzione di cicli, si può imporre che un agente virtuale possa essere figlio solo di agenti mediatori con indice minore.

$$\sum_{col \in Colonne} S(row, col) = 1, \quad \forall row \in Mediatori, col \leq row - 1 \tag{3.5}$$

$$\sum_{col \in Colonne} S(row, col) = 0, \quad \forall row \in Mediatori, col \geq row \tag{3.6}$$

Il coefficiente w è associato a tutti gli agenti e calcolato a partire dalla radice fino alle foglie. Con questa condizione si può calcolare w nel seguente modo:

$$w_{row} = \sum_{col \in Colonne} S(row, col) \cdot \delta^+_{col} \cdot w_{col} \quad \forall row \in Agenti \tag{3.7}$$

Si sfrutta il fatto che l'unico prodotto $\delta^+ \cdot w$ che non sarà annullato dallo zero in S, sarà quello corrispondente al padre dell'agente. Data infine una soluzione ammissibile $\bar{w}$, che come descritto in precedenza rappresenta una classe di equivalenza di soluzioni relativamente all'accuratezza, imponendo l'uguaglianza con i valori di w per gli agenti diagnostici, è possibile ottenere tutte le strutture associate a $\bar{w}$.

$$w_{ag} = \bar{w}_{ag} \quad \forall ag \in Diagnostici \tag{3.8}$$

A causa dell'Equazione 3.7 il problema non è lineare, quindi non è risolvibile tramite programmazione lineare intera.

3.5.4 Funzione di aggregazione negoziazione cooperativa

Questa funzione aggregherà i risultati dei nodi applicando un algoritmo di negoziazione cooperativa. L'algoritmo iterativo è stato descritto nella Sezione 2.3. A causa della non linearità dell'algoritmo non si è individuato uno spazio delle soluzioni specifico, la difficoltà del problema di ricerca della struttura ottima quindi insisterà sullo spazio di tutte le strutture possibili.

Altre tipologie di negoziazione, come il *One-Step Protocol* descritto successivamente, non si dimostrano adatte ad affrontare il problema. Si utilizzerà pertanto la negoziazione cooperativa iterativa.

One-Step Protocol

Così come descritto nella Sezione 2.3, gli agenti che partecipano alla negoziazione devono effettuare un'offerta o proposta. Il protocollo one-step prevede che ciascun agente determini il set delle proposte che massimizzano la produttoria di tutte le utilità degli agenti coinvolti nella negoziazione. Tra di esse, l'agente proporrà quella che massimizza la sua utilità.

Determinato il risultato della negoziazione tra gli agenti diagnostici, si pone il problema di definire il comportamento dell'agente mediatore quando negozierà con altri agenti.

Sia δ_{vi} il risultato della negoziazione il cui mediatore è l'agente Vi. Si indichino con $j \in figli(Vi)$ gli agenti figli di Vi.

$$\delta_{vi} = \arg\max_{\delta} \prod_{j \in figli(Vi)} u_j(\delta)$$

Si pone il problema di determinare quale sarà la $u(\delta)$ associata all'agente mediatore, da utilizzarsi nella negoziazione al livello gerarchico superiore. Poniamo che la $u_v(\delta)$ dei mediatori sia definita come la produttoria delle utilità degli agenti figli.

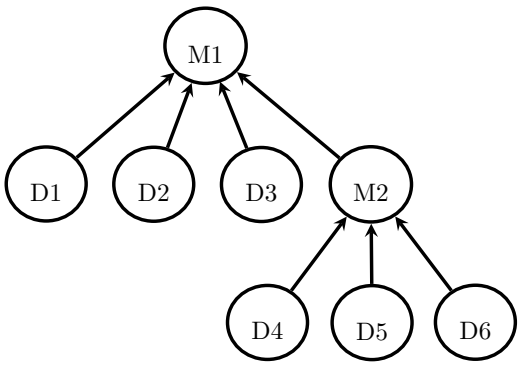

Figura 3.4: Una struttura possibile.

Consideriamo come esempio questa configurazione. Definiamo il set *Diagnostici*, contenente tutti gli agenti diagnostici. Di seguito sono calcolati i risultati della negoziazione svolta secondo la struttura di quest'albero.

$$
\begin{aligned}
\delta_{M2} &= \arg\max_{\delta} \left(u_{D4}(\delta) \cdot u_{D5}(\delta) \cdot u_{D6}(\delta)\right) \\
\delta_{M1} &= \arg\max_{\delta} \left(u_{D1}(\delta) \cdot u_{D2}(\delta) \cdot u_{D3}(\delta) \cdot u_{M2}(\delta)\right) \\
&= \arg\max_{\delta} \left(u_{D1}(\delta) \cdot u_{D2}(\delta) \cdot u_{D3}(\delta) \cdot u_{D4}(\delta) \cdot u_{D5}(\delta) \cdot u_{D6}(\delta)\right) \\
&= \arg\max_{\delta} \prod_{j \in Diagnostici} u_j(\delta)
\end{aligned}
$$

Quindi, posto che l'agente virtuale di cui ci interessa il risultato sia $M1$, si evince che:

- Il risultato della negoziazione mediata da $M2$ non è effettivamente utilizzato da $M1$. Nel caso generale, non sarà necessario calcolare esplicitamente le negoziazioni per i vari agenti mediatori nel sottoalbero dell'agente di interesse.
- Il risultato finale è calcolato tramite una produttoria, dunque l'ordine delle funzioni non è rilevante. Non compaiono coefficienti legati al livello gerarchico degli agenti reali. Da ciò deriva che l'effettiva struttura dell'albero sottostante l'agente di interesse non abbia impatto sulla negoziazione.

Quindi lo spazio delle soluzioni è coincidente con quello nel caso della funzione di aggregazione *massimo*.

3.6 Apporti innovativi

Relativamente all'ambito spaziale, l'utilizzo di un sistema multiagente consentirà di semplificare notevolmente lo sviluppo di sistemi di anomaly detection. Rispetto ad approcci più tradizionali model-based, addestrare i modelli degli agenti diagnostici basandosi sui dati evita la lunga fase di formalizzazione della domain knowledge legata al satellite. Il sistema di anomaly detection quindi può essere implementato in completa assenza di domain knowlege. Inoltre, l'uso di un sistema multagente conferisce la flessibilità necessaria ad adattarsi alle mutate condizioni operative in cui può trovarsi il satellite nel corso della sua vita. Essendo difficile o impossibile una sua riparazione, nel corso degli anni delle componenti potrebbero deteriorasi, eventualmente mutando il loro funzionamento nominale, o dover essere disattivate. Se il modello del sistema fosse espresso in modo quantitativo tramite un insieme di equazioni, sarebbe necessario un considerevole sforzo per modificare il modello adattandolo alle nuove condizioni operative. Un approccio agent-based invece si può adattare molto più velocemente semplicemente riaddestrando gli agenti diagnostici interessati facendo uso di

dati nominali più recenti. La disattivazione di un sistema può essere gestita rimuovendo gli agenti diagnostici associati. Infine, qualora la disponibilità di dati permetta di addestrare un modello per una nuova variabile, un nuovo agente diagnostico può essere facilmente integrato nel modello precedente.

La ricerca della struttura di aggregazione ottima può permettere di rappresentare la struttura fisica del sistema. In questo modo si possono fornire informazioni più dettagliate agli addetti alla sua supervisione.

Il sistema multiagente inoltre fornisce molta flessibilità in merito alla distribuzione degli agenti e dei task, permettendo di implementare un'architettura distribuita. Un approccio distribuito è adatto all'ambito spaziale in quanto esistono diversi casi in cui un sistema centralizzato a terra sia poco efficiente. Un primo motivo può essere il ritardo di comunicazione, ossia il tempo che è necessario ad un messaggio inviato dalla sonda per raggiungere il centro di controllo sulla Terra, essere elaborato e la risposta tornare alla sonda. A causa delle distanze interplanetarie questo ritardo può essere elevato. Il rover Curiosity, atterrato su Marte nell'agosto 2012, ha un ritardo di comunicazione di circa 20 minuti. La sonda Juno, il cui incontro con Giove è previsto per l'agosto 2016, ha un ritardo di oltre un'ora, la sonda New Horizons, sita poco oltre Plutone dopo il sorvolo del luglio 2015, oltre 9 ore [70], e la sonda Voyager 1, lanciata nel 1977 ed attualmente l'unica missione ad aver raggiunto lo spazio interstellare, circa 37 ore [71]. Questi lunghi lassi di tempo rendono difficile o impossibile intervenire prontamente in presenza di un'anomalia intervenendo dalla Terra. A causa delle limitate risorse computazionali disponibili su tali sonde o rover l'uso di un sistema di anomaly detecion centralizzato on board potrebbe essere ugualmente non praticabile. Un sistema distribuito permette una ripartizione del carico computazionale più flessibile, svolgendo le attività più critiche direttamente on board o su un orbiter e le altre, meno prioritarie, computazionalmente pesanti o che richiedano analisi su lunghi periodi di tempo e quindi voluminosi dataset, sulla Terra. Un approccio distribuito consente anche di ri-bilanciare la distribuzione dei task a seguito di mutate condizioni operative, per esempio trasferendo sulla sonda ulteriori compiti in quanto la distanza rende la comunicazione sempre meno efficiente oppure svolgendone un maggior numero a Terra per compensare il progressivo calo nelle risorse energetiche della sonda o un'avaria a dei banchi di memoria o ad un processore.

Ulteriore ragione per implementare un sistema distribuito può essere l'inaffidabilità del canale di comunicazione, che può essere soggetto ad interruzioni e disturbi causati dalla posizione della sonda rispetto agli altri corpi celesti ed alle stazioni di ricezione sulla terra. Il rover Curiosity per la comunicazione con la Terra si appoggia a due satelliti, il Mars Reconnaissance Orbiter ed Odissey. A causa della loro traiettoria questi satelliti possono comunicare con il rover per soli otto minuti al giorno [72]. Difficoltà di comunicazione potrebbero sorgere anche a causa di conflitti con le attività di altre missioni, tali per cui si debba contingentare il tempo di tra-

smissione e non sia possibile avere un contatto continuo e costante con tutte le missioni. Sempre nel caso di Curiosity, i due satelliti hanno traiettorie tali da permettere la comunicazione con la Terra per 16 ore al giorno, ma solo una frazione di questo tempo è utilizzata effettivamente [72]. In tutti questi casi l'uso di un sistema multiagente permette di distribuire efficacemente i task, in modo che quelli più critici o computazionalmente meno impegnativi possano essere svolti on board, offrendo un sistema di anomaly detection in grado di monitorare senza interruzioni il funzionamento della sonda o del rover.

Capitolo 4

Algoritmi di ricerca

Considerando la formalizzazione tramite grafo presentata nel capitolo precedente, si vuole ora proporre un metodo per risolvere il problema di ottimizzazione, ossia il trovare una struttura ottima per il sistema multiagente preposto all'individuazione delle anomalie in sistemi spaziali. In questo capitolo si descrive come il problema è stato affrontato. Partendo dalle caratteristiche che si desidera la soluzione abbia, si propongono due diverse famiglie di algoritmi di ricerca, locale e globale. Per i diversi algoritmi si offre una descrizione e un inquadramento generale delle loro caratteristiche e limitazioni.

4.1 Caratteristiche della struttura degli agenti

Prima di avviarsi alla descrizione degli algoritmi è necessario esprimere più nel dettaglio le caratteristiche che si vuole la struttura degli agenti possieda. Innanzitutto occorre determinare se un agente possa partecipare a più negoziazioni, quindi se ogni nodo possa avere taglio uscente superiore ad uno.

- **Aggregazione singola** - gli agenti diagnostici e mediatori partecipano ad una sola o nessuna negoziazione. Si basa sull'assunzione che una variabile non incida in modo significativo su più di un insieme di altre variabili. In questo caso il grafo avrà una struttura ad albero in quanto un nodo potrà avere un solo padre. Le foglie dell'albero saranno gli agenti diagnostici ed i nodi intermedi rappresenteranno gli agenti mediatori responsabili dell'aggregazione dei risultati dei loro figli.

- **Aggregazione multipla** - gli agenti diagnostici e mediatori possono partecipare ad un numero arbitrario di aggregazioni con diversi insiemi di altri agenti. Una variabile può incidere su più sottoinsiemi non identici. Le interdipendenze tra variabili possono essere rappresentate

in modo più accurato ma lo spazio delle possibili soluzioni è notevolmente più ampio. Inoltre, poiché un nodo può essere collegato a più agenti mediatori, i quali a loro volta sono collegati ad una sola radice, il grafo deve essere orientato per garantirne l'aciclicità.

Il secondo aspetto significativo è il tipo di ottimo che si vuole identificare:

- **Ottimo locale** - soluzione buona ma per la quale non ci siano garanzie in merito a quanto sia distante dall'ottimo globale. Si può applicare un algoritmo di ricerca locale, eventualmente coadiuvato da approcci che esplorino più ottimi locali scegliendone il migliore, avvantaggiandosi della rapidità con la quale si può raggiungere una soluzione.
- **Ottimo globale** - soluzione certamente preferibile a livello di funzione obiettivo ma potrebbe essere raggiungibile nel caso generale solo applicando un algoritmo dalla maggiore complessità computazionale. Ciò comporterebbe una dimensione del problema trattabile inferiore.

Il terzo aspetto riguarda l'introduzione di domain knowledge qualora sia disponibile:

- **Ex ante** - fornita dagli esperti sotto forma di configurazione iniziale del grafo o raggruppamenti delle variabili in sottosistemi, formalizzata prima che sia avviato l'algoritmo di ricerca.
- **In itinere** - informazioni di carattere più generale sotto forma di pattern strutturali consolidati, frutto dell'operatività di altri sistemi o definiti da esperti. Tali pattern non saranno probabilmente identici a quelli esistenti nel satellite in questione, ma potrebbero comunque contenere informazioni utili per guidare la costruzione del grafo.
- **Ex post** - archi aggiunti o tolti dagli esperti successivamente alla creazione del grafo o ad un suo aggiornamento.

Caratteristiche obiettivo per questa tesi

Tra le diverse possibilità elencate nella sezione precedente, in questa tesi si è scelto di focalizzarsi sull'aggregazione singola, in quanto si ritiene possa offrire un buon compromesso tra comprensibilità della soluzione, accuratezza e difficoltà del problema di ricerca.

Si sono proposti algoritmi sia di ricerca locale che globale, per mostrare quale sia il miglior risultato raggiungibile e verificare se il vantaggio in termini di rapidità della ricerca locale possa giustificare la minor qualità della soluzione, nonchè valutare l'entità di questa differenza.

L'introduzione della domain knowledge può essere effettuata ex-ante specificando tramite un opportuno file di configurazione quale topologia di albero si desideri ed indicando quali insiemi di agenti debbano essere aggregati

separatamente dagli altri. In questo modo, se si è a conoscenza del fatto che alcuni agenti appartengono a precisi sottosistemi è possibile separarli ed effettuare per ciascuno dei sottosistemi una ricerca indipendente della soluzione ottima. Gli alberi così ottenuti saranno poi automaticamente aggregati in una struttura complessiva che rappresenterà la struttura del sistema multiagente. Successivamente alla creazione della rete è possibile intervenire manualmente sulla struttura ottima individuata, per modificarla integrando ulteriori informazioni disponibili.

4.2 Algoritmi di ricerca globale

Applicare un algoritmo di ricerca che garantisca si raggiunga una soluzione ottima globale è certamente preferibile, a livello di cifra di merito, ad uno che si limiti ad un ottimo locale. Tuttavia l'elevata numerosità delle possibili soluzioni rende difficile applicare un algoritmo di ricerca globale. Oltre all'accuratezza anche la complessità, legata alla comprensibilità, della soluzione ha significativa importanza in problemi il cui numero di agenti diagnostici sia elevato. La cifra di merito deve quindi considerare sia l'accuratezza della predizione sia la complessità dell'albero risultante. Si vogliono scoraggiare soluzioni molto accurate ma così complesse da risultare poco comprensibili per un essere umano. Una cifra di merito che si componga di queste due parti rende possibile l'esistenza di più soluzioni ottime, pressochè equivalenti nella cifra di merito, ma strutturalmente diverse. Poichè nel caso analizzato in questa tesi il numero di agenti diagnostici è limitato, le soluzioni hanno un numero di nodi abbastanza contenuto e quindi non è stato necessario introdurre una cifra di merito relativamente alla loro complessità.

4.2.1 Algoritmo enumerativo

La più semplice tecnica per raggiungere una soluzione ottima globale è enumerare tutte le possibili soluzioni, rappresentandole esplicitamente e determinando per ciascuna la cifra di merito. In virtù delle considerazioni sulla dimensione dello spazio delle possibili soluzioni riportate in Sezione 3.5, si tratta di una strada percorribile solo per piccole istanze del problema.

E' possibile enumerare le soluzioni non limitandosi solo alla struttura, ma anche considerando le possibili funzioni di aggregazioni associabili ad un mediatore. In questo modo per ogni struttura esisteranno numerose varianti, caratterizzate ciascuna da un diverso assegnamento di funzioni di aggregazione ai mediatori.

4.2.2 Algoritmo branch and bound

La tecnica del branch and bound è utilizzabile per risolvere problemi di ottimizzazione combinatoria, quindi può ben prestarsi al caso in questione.

L'algoritmo sviluppa un albero i cui nodi rappresentano ciascuno un insieme di soluzioni che abbiano certe caratteristiche. La radice contiene tutte le soluzioni possibili. Ogni nodo figlio contiene un sottoinsieme delle soluzioni associate al nodo padre, alle quali è stata applicata la restrizione rappresentata dall'arco che li congiunge. Le restrizioni corrispondono all'introduzione di uno specifico arco nella soluzione. Scegliendo opportunamente la politica con cui generare le restrizioni applicabili ad un dato nodo del branch and bound è possibile alterare la struttura degli insiemi delle soluzioni associate ad uno stato, ad esempio albero o grafo, ed imporre l'assenza di cicli e l'esistenza di un'unica radice.

Una volta generato un nodo figlio, prima di espandere i suoi discendenti, si verifica se il suo sottoalbero possa produrre una soluzione preferibile a quella fino ad ora migliore. In caso affermativo il sottoalbero viene esplorato, in caso negativo viene scartato e l'algoritmo procede valutando un altro nodo, oppure terminando. L'accurata scelta della funzione euristica utilizzata per stimare i limiti superiore ed inferiore, o bound, della cifra di merito, con la quale è valutata una soluzione generabile in un sottoalbero, assume notevole importanza.

Poichè i bound devono essere calcolati per ogni nodo, occorre che la funzione euristica sia rapida, diversamente il branch and bound diverrebbe inefficiente. Inoltre, l'euristica pur basandosi su una descrizione solo parziale della soluzione, deve essere in grado di calcolare una stima sufficientemente stringente.

Ogni nodo può essere rappresentato come un grafo non connesso nel quale compaiano solo gli archi corrispondenti alle restrizioni applicate lungo il percorso dalla radice fino al nodo. Le soluzioni associate a quel nodo saranno quindi rappresentate in modo implicito tramite i vincoli che devono soddisfare. Nel caso in cui il nodo sia associato ad un grafo connesso, allora esso conterrà una sola soluzione. A seconda del tipo di struttura desiderata, l'algoritmo deve adottare un comportamento diverso:

- **Soluzione sotto forma di albero -** i nodi contenenti soluzioni ammissibili non possono produrre ulteriori discendenti. Il grafo è orientato, per esprimerlo sotto forma di albero è necessario trasformare gli archi orientati in lati non orientati preservando l'ordinamento dei nodi sfruttando i livelli gerarchici dell'albero stesso. Alla perdita dell'orientamento dei lati consegue che nessuna ulteriore azione possa essere applicata, in quanto l'aggiunta di un lato ad un albero comporta l'introduzione di un ciclo. Le soluzioni ammissibili potranno quindi trovarsi esclusivamente nelle foglie dell'albero del branch and bound.

- **Soluzione sotto forma di grafo -** i nodi contenenti le prime soluzioni ammissibili, sotto forma di albero, potranno a loro volta avere figli. Le soluzioni ammissibili si troveranno quindi in tutti i nodi a partire dal livello in cui è presente un grafo connesso, sino alle foglie del

branch and bound. Tali foglie conterranno tutte le soluzioni possibili nelle quali l'aggiunta di un qualunque ulteriore arco causi la creazione di un ciclo. Esse saranno replicate tante volte quante sono le possibili permutazioni nell'ordine di scelta delle restrizioni. Esisteranno quindi più percorsi con i quali raggiungere la stessa soluzione e quindi dei sottoalberi duplicati. Per evitare duplicazioni, o all'interno della funzione di pruning o nella politica di generazione dei discendenti, dovrà essere considerato anche il fatto che un certo insieme di vincoli possa già esistere in un altro nodo.

La funzione euristica è strettamente legata alla funzione di aggregazione ed alla struttura prodotta dalla politica di generazione dei discendenti di un nodo. Sfortunatamente l'unica funzione di aggregazione per la quale è stato possibile concepire un'euristica non banale è il massimo. Tuttavia, come riportato nella Sezione 3.5.2, in questo caso la struttura non ha rilevanza sul risultato e quindi l'applicazione di un algoritmo di ricerca è priva di utilità. Similmente anche nel caso della ricerca effettuata potendo scegliere quale funzione di aggregazione assegnare al mediatore non è stato possibile sviluppare un'euristica efficace per fare pruning tra negoziazione cooperativa e media. In questa tesi, quindi, in assenza di una funzione euristica ragionevole non è stato possibile implementare una ricerca tramite branch and bound.

4.3 Algoritmi di ricerca locale

Utilizzare un algoritmo di ricerca locale presenta qualche vantaggio rispetto ad uno di ricerca globale, in quanto può permettere di raggiungere una buona soluzione in un lasso di tempo contenuto e con limitate risorse. Questa caratteristica assume significativa importanza nel nostro caso, in quanto a causa delle caratteristiche delle funzioni di aggregazione non è stato possibile proporre un algoritmo di ricerca globale efficiente.

Non si può, nel caso generale, fare alcuna assunzione in merito alla effettiva bontà della soluzione ottenuta tramite ricerca locale rispetto all'ottimo globale. Approcci metaeuristici come il *simulated annealing* consentono di mitigare questo problema espandendo lo spazio di ricerca ed esplorando più possibili soluzioni. L'ampiezza di tale spazio può essere determinata sulla base del tempo a disposizione, quindi si può agevolmente adattare alle necessità del caso applicativo.

4.3.1 Simulated annealing

Il *simulated annealing* [73] è una tecnica probabilistica utilizzata per approssimare la soluzione ottima globale in problemi con uno spazio di stati discreti molto vasto. Si tratta di una metaeuristica, in quanto permette di

accettare soluzioni intermedie peggiori, consentendo di uscire da ottimi locali e realizzando un'esplorazione più esaustiva nello spazio delle soluzioni. Si applica laddove raggiungere la soluzione ottima globale sia proibitivo o difficile e si preferisca una buona soluzione in un lasso di tempo contenuto. Si tratta di una versione dell'algoritmo di Metropolis–Hastings [74], a sua volta della classe dei metodi di Monte Carlo basati su catena di Markov [75]. Lo pseudocodice del simulated annealing è riportato nell'Algoritmo 1.

Nel nostro caso l'algoritmo si muoverà sul grafo delle possibili soluzioni. Per ogni iterazione il metodo genera, a partire dalla soluzione corrente, una nuova soluzione chiamata *vicino* e decide probabilisticamente se accettarla o meno. Ciò col passare delle iterazioni porta il sistema a raggiungere stati dal minore costo. I vicini di uno stato sono ottenuti applicando ad esso un certo tipo di alterazione, differente a seconda del contesto. Tale alterazione rappresenta uno spostamento ad altro nodo nel grafo delle possibili soluzioni. Applicare una buona strategia per determinare i vicini di uno stato è molto importante, in quanto la soluzione finale sarà prodotta da una sequenza di tali operazioni. Il loro scopo è di preservare le parti migliori della soluzione corrente, modificando le altre. Nel nostro caso quindi sarà applicato ad una struttura gerarchica già completa e proverà a cambiare la collocazione degli agenti. Come struttura di partenza si sceglie la configurazione flat, in cui tutti gli agenti diagnostici sono figli della radice.

La probabilità di accettare il nuovo vicino è data da una funzione apposita che considera i costi dei due nodi e la *temperatura* T, variabile che tende a zero con il progredire dell'algoritmo. Per valori di T elevati l'algoritmo accetterà più facilmente soluzioni peggiori, per valori prossimi a zero invece accetterà quasi esclusivamente vicini migliori. Ponendo $T = 0$ si ottiene un algoritmo greedy che sceglie solo vicini migliori.

Si è dimostrato che con un *periodo di raffreddamento* sufficientemente lungo l'algoritmo sarà in grado di convergere sull'ottimo globale. Esso potrebbe però essere raggiunto più lentamente rispetto all'enumerazione di tutte le soluzioni a causa della componente stocastica.

Per poter applicare il simulated annealing sono necessari:

- Una funzione di costo $Cost(Solution) : solution \mapsto R$.
- Una procedura per determinare i vicni $Sol_{neighbor} = CreateNeighborSolution(Sol_{current})$.
- Una funzione di accettazione $AcceptanceProbability(Cost_Sol_current, Cost_Sol_neigh, Temp_current) \mapsto [0, 1]$ tale che

$$\begin{cases} AcceptanceProbability > 0, & \text{se } Cost_Sol_current < Cost_Sol_neigh \\ AcceptanceProbability \to 0, & \text{se } T \to 0 \end{cases}$$

- La funzione di raffreddamento monotona decrescente $CalculateNewTemperature(i, Temp_{max}, cooling_coefficient) \mapsto R \geq 0$.

- La temperatura iniziale $Temp_{max} > 0$.

- Il numero di iterazioni massimo $Iterations_{max}$.

Scelta dei vicini

Il comportamento dell'algoritmo può essere ricondotto al *random walk* su un grafo i cui nodi rappresentino ciascuno una possibile soluzione e i lati rappresentino le possibili azioni. La funzione che determina i vicini è efficace se permette di definire su di esso un sottografo avente gli stessi nodi ma un sottoinsieme dei lati, tale per cui esista un cammino sufficientemente breve tra la soluzione iniziale e un ottimo globale. Tanto più breve sarà questo cammino tanto più rapidamente e quindi probabilmente, dato un numero fissato di iterazioni, sarà possibile raggiungere l'ottimo locale.

Per raggiungere qualsiasi albero a partire dalla configurazione iniziale scelta, la flat, è sufficiente una funzione che scelga un mediatore esistente, vi aggiunga un nuovo figlio mediatore e gli assegni alcuni dei suoi figli. Per raggiungere un qualsiasi albero a partire da un albero generico, è sufficiente introdurre anche la funzione inversa, che elimini un dato mediatore unendo il suo sottoalbero a quello del nodo padre. Tale approccio però è poco efficiente in quanto tende a visitare soluzioni a pochi livelli con probabilità nettamente superiore. Inoltre, per modificare l'allocazione delle foglie dato uno scheletro è necessario eliminare e poi reintrodurre almeno un mediatore. Si è optato quindi per una variante che fa uso di quattro funzioni, scelte casualmente secondo la probabilità indicata. Di seguito è descritto il loro comportamento rispetto ad un albero generico rappresentato in Figura 4.1.

- Scambio di due foglie tra mediatori diversi, Figura 4.2. Probabilità 30%;

- Spostamento di una foglia da un mediatore ad un altro, Figura 4.3. Probabilità 30%;

- Aggiunta di un nuovo mediatore, assegnandogli due foglie del padre, Figura 4.4. Probabilità 25%;

- Rimozione di un mediatore, Figura 4.5. Probabilità 15%.

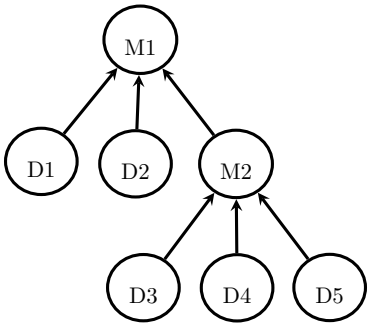

Figura 4.1: Struttura iniziale.

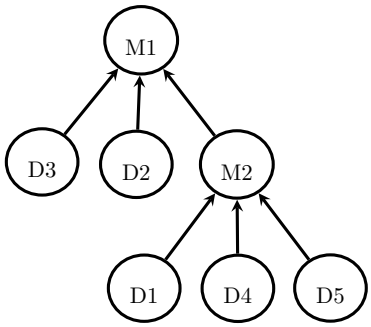

Figura 4.2: Scambio tra le foglie D1 e D3.

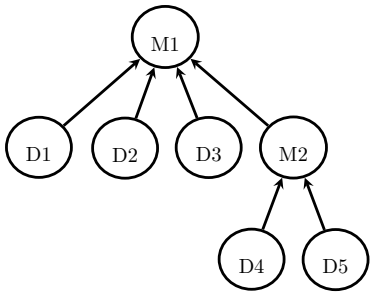

Figura 4.3: Spostamento della foglia D3 da M2 a M1.

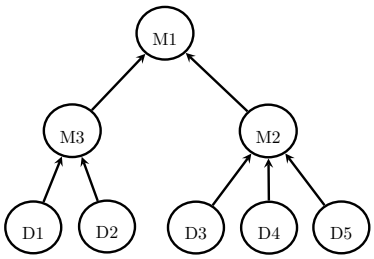

Figura 4.4: Aggiunto mediatore M3, inglobando D1 e D2.

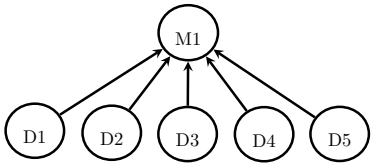

Figura 4.5: Rimosso il mediatore M2, le sue foglie sono acquisite da M1.

Con questa variante della neighbor function lo sbilanciamento nella probabilità di esplorare le soluzioni viene notevolmente ridimensionato. Eseguendo la funzione finchè ogni soluzione nel problema con 6 agenti diagnostici sia stata esplorata, si può osservare che la maggior parte degli alberi è generata mediamente decine di volte, alcuni poche centinaia, la configurazione flat circa mille. Si è così verificato che la neighbor function proposta esplora completamente lo spazio delle soluzioni.

Termperatura e raffreddamento

Si tratta di parametri strettamente legati al problema da risolvere, esistono delle tecniche per ridurre la dipendenza dai dati inseriti dall'utente chiamate *adaptive simulated annealing* [76]. Nel caso in questione è stato sufficiente stabilire dei valori fissi: la temperatura iniziale $Temp_{max}$ è stata posta a 1 000 e il coefficiente di raffreddamento $cooling_coefficient$ a 0.01. La funzione per il calcolo della temperatura utilizzata nella iterazione i è definita come segue:

$$CalculateNewTemperature(i, Temp_{max}, cooling_coefficient) = \\ Temp_{max} \cdot (1 - cooling_coefficient)^i$$

La probabilità di accettare una soluzione peggiore rispetto a quella corrente è calcolata come:

$$AcceptanceProbability = \exp\left(\frac{Cost_Sol_current - Cost_Sol_neigh}{Temp_{current}}\right)$$

Restart

Nell'algoritmo può essere implementata anche una policy di restart che intervenga se per un certo numero di iterazioni si sono considerate soluzioni peggiori, oltre una certa soglia, rispetto alla migliore trovata. In questo caso si torna alla soluzione migliore trovata fino a quel momento e si fa ripartire l'esecuzione. Per uscire da ottimi locali in fasi ormai avanzate della computazione si può applicare un *tunneling stocastico* [77], che porta l'algoritmo a saltare verso altre soluzioni con una certa probablità, superando la limitazione di un valore di temperatura corrente ormai basso. L'implementazione del simulated annealing proposta non applica policy di restart. L'opportunità di applicare queste tecniche, o altre già disponibili e descritte in letteratura, dipenderà dai risultati che saranno conseguiti in un caso reale. Per questo è utile confrontare il risultato conseguito con quello di un algoritmo di ricerca globale, in modo da studiare la natura e disposizione dei massimi nonchè la qualità delle soluzioni ottime locali effettivamente raggiunte.

4.3.2 Algoritmo greedy

Sia enumerazione che simulated annealing esplorano lo spazio delle soluzioni alla ricerca dell'ottimo, rendendo difficile gestire problemi con un numero di agenti diagnostici elevato. Si è quindi provato l'approccio inverso, ossia costruire la struttura anzichè cercarla, applicando un basilare algoritmo greedy con una componente stocastica simile a quanto si usa nelle random forest. Ad ogni iterazione si sceglie casualmente una tra le n azioni localmente migliori, dove n è specificato come parametro. Lo pseudocodice dell'algoritmo greedy è riportato nell'Algoritmo 2.

Algorithm 1 Simulated annealing

```
procedure SimulatedAnnealing(ProblemData, Iterations_max, Temp_max)
    Sol_current = CreateInitialSolution(ProblemData)
    Sol_best = Sol_current
    Cost_Sol_current = Cost(Sol_current)
    Cost_Sol_best = Cost_Sol_current

    for i = 0 to Iterations_max do
        Sol_neigh = CreateNeighborSolution(Sol_current)
        Cost_Sol_neigh = Cost(Sol_neigh)
        Temp_current = CalculateNewTemperature(i, Temp_max)

        if Cost_Sol_neigh ≤ Cost_Sol_current then
            Sol_current = Sol_neigh
            Cost_Sol_current = Cost_Sol_neigh

            if Cost_Sol_neigh ≤ Cost_Sol_best then
                Sol_best = Sol_neigh
                Cost_Sol_best = Cost_Sol_neigh
            end if

        else if AcceptanceProbability(Cost_Sol_current,
    Cost_Sol_neigh, Temp_current) > Rand(0, 1) then
            Sol_current = Sol_neigh
            Cost_Sol_current = Cost_Sol_neigh
        end if
    end for

    Return Sol_best
end procedure
```

L'algoritmo è inizializzato a partire da un albero vuoto. Tutte le azioni coinvolgono un insieme di agenti non ancora inseriti nella struttura di cardinalità z compresa tra 1 e un valore massimo, in questo caso 4. Le prime azioni disponibili prevedono di inserire un insieme di agenti diagnostici come figli della radice. Dalla seconda iterazione le azioni disponibili sono di due tipi. Di seguito è descritto il loro comportamento rispetto ad un albero generico rappresentato in Figura 4.6.

- Aggiungere un set di agenti diagnostici ai figli di un mediatore già esistente, Figura 4.7.
- Aggiungere ai figli di un mediatore un nuovo mediatore avente per figli un set specifico di agenti diagnostici, Figura 4.8.

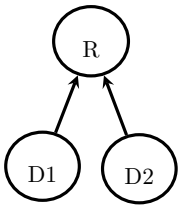

Figura 4.6: Aggiunto alla radice l'insieme formato da D1 e D2. Ancora da inserire D3, D4 e D5.

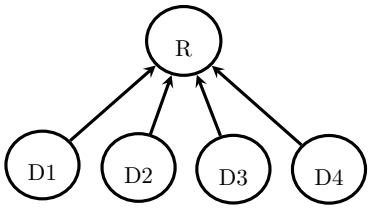

Figura 4.7: Aggiunto ai figli della radice l'insieme di agenti diagnostici composto da D3 e D4. Ancora da inserire D5.

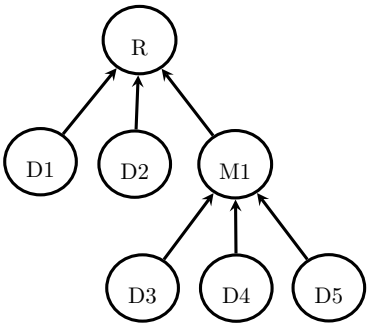

Figura 4.8: Creato un nuovo agente mediatore figlio della radice, avente per figli D3, D4 e D5. Tutti gli agenti diagnostici sono stati inseriti.

L'algoritmo termina quando tutti gli agenti diagnostici sono stati inseriti nella soluzione. Poichè ad ogni iterazione sono mantenute in memoria solo le n migliori soluzioni, l'occupazione della memoria è molto bassa. Assume notevole importanza invece il numero di possibili azioni da valutare.

Per compensare la possibilità che si raggiunga un ottimo locale, l'algoritmo è eseguito molteplici volte, sfruttando il fatto che il risultato è ottenibile molto velocemente e la componente stocastica dell'algoritmo potrebbe permettere di individuare una soluzione migliore.

Le azioni previste per questo algoritmo non sono sufficienti a raggiungere qualsiasi soluzione. Sono irraggiungibili, nello specifico, soluzioni in cui un mediatore abbia solo figli mediatori.

Si mostrano di seguito le 3 soluzioni raggiungibili nel caso di albero perfettamente sbilanciato con 3 agenti diagnostici. In questo caso è disponibile solo l'azione di creazione di un nuovo mediatore, la prima azione di default assegna un figlio alla radice, le successive introducono un nuovo livello gerarchico, si veda Figura 4.9.

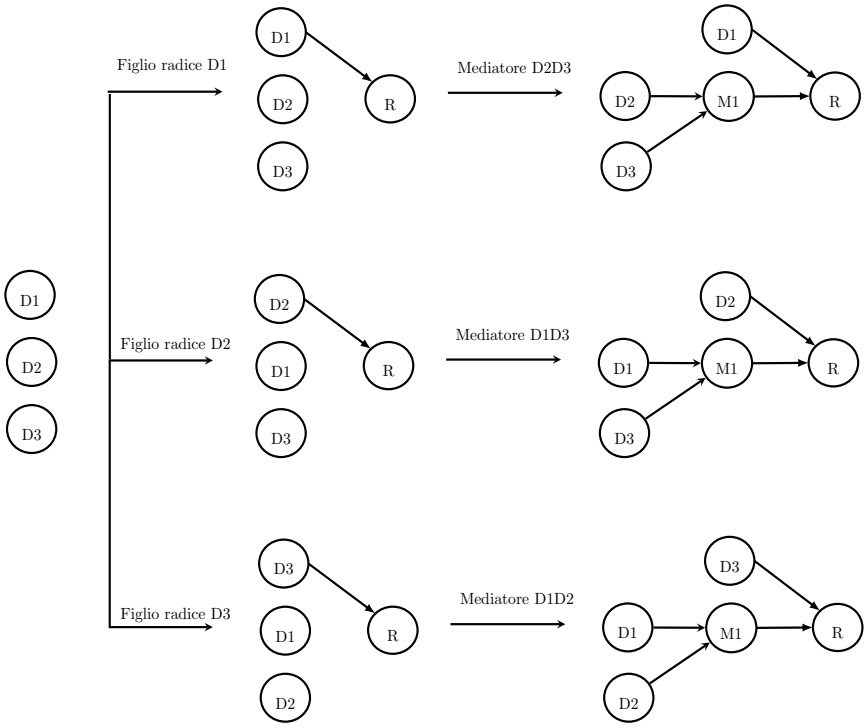

Figura 4.9: Soluzioni raggiungibili con struttura ad albero perfettamente sbilanciato.

Nel caso di più agenti diagnostici per ogni livello, l'azione che creerà un nuovo mediatore sceglierà tra i set di z elementi. Si riporta un possibile risultato in Figura 4.10.

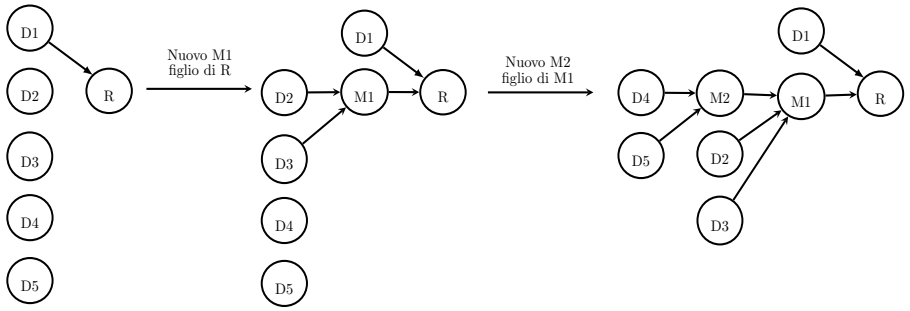

Figura 4.10: Soluzioni raggiungibili con struttura ad albero sbilanciato $z = 2$

Alberi di diversa topologia si possono raggiungere assegnando il nuovo mediatore in modo che sia fratello di un altro mediatore. In questo modo l'algoritmo greedy è in grado di esplorare soluzioni di struttura variegata non limitate agli alberi perfettamente sbilanciati.

Algorithm 2 Pseudocodice algoritmo greedy - Funzione principale e inizializzazione.

```
1: function BuildTree(availableAgentsList, maxSetSize, numAlternatives)
2:     declare: possibleActionsList
3:     declare: finalStructure

4:     possibleActionsList = InitializeActionsList(availableAgentsList, maxSetSize)

5:     repeat
6:         Genera un numero casuale tra 0 e numAlternatives
7:         Sceglie in possibleActionsList l'azione con quell'indice
8:         Assegna a finalStructure la struttura scelta
9:         Toglie da availableAgentsList gli agenti coinvolti nell'azione
10:        terminationCondition = TerminationCondition(cifra di merito)
11:        if Creato nuovo mediatore con un solo figlio then
12:            Forza allocazione di un figio a quel mediatore nella prossima iterazione
13:        end if
14:        if not terminationCondition then
15:            addDiagnosticAgentActions
16:            newMediatorAgentActions
17:        end if
18:    until not terminationCondition
19: end function

20: function InitializeActionsList(availableAgentsList, maxSetSize)
21:    for setSize = 1 to maxSetSize do
22:        Nuovo iteratore delle combinazioni semplici senza ripetizioni di setSize elementi
23:        repeat
24:            Crea una struttura in cui la radice ha come figli solo quelli contenuti nel set corrente
25:            Calcola lo score della struttura
26:            Aggiunge a possibleActions la struttura e il suo score
27:        until esistono altre combinazioni
28:    end for
29: end function
```

Algorithm 3 Pseudocodice algoritmo greedy - Possibili azioni.

```
30: procedure newMediatorAgentActions
31:     Mediatore iniziale è la radice
32:     if è prevista l'allocazione forzata per un mediatore then
33:         Mediatore iniziale è il mediatore forzato
34:     end if
35:     repeat
36:         Alloco un figlio mediatore al mediatore corrente
37:         for setSize = 1 to maxSetSize do
38:             Nuovo iteratore delle combinazioni semplici senza ripetizioni
    di setSize elementi
39:             repeat
40:                 Assegna come figli del nuovo mediatore gli agenti contenuti
    nel set corrente
41:                 Calcola lo score della struttura
42:                 Aggiunge a possibleActions la struttura e il suo score
43:             until esistono altre combinazioni
44:         end for
45:         Mediatore corrente successivo
46:     until tutti i mediatori esplorati or esiste un mediatore forzato
47: end procedure

48: procedure addDiagnosticAgentActions
49:     Mediatore iniziale è la radice
50:     if è prevista l'allocazione forzata per un mediatore then
51:         Mediatore iniziale è il mediatore forzato
52:     end if
53:     repeat
54:         for setSize = 1 to maxSetSize do
55:             Nuovo iteratore delle combinazioni semplici senza ripetizioni
    di setSize elementi
56:             repeat
57:                 Aggiungi ai figli del mediatore corrente gli agenti contenuti
    nel set corrente
58:                 Calcola lo score della struttura
59:                 Aggiunge a possibleActions la struttura e il suo score
60:             until esistono altre combinazioni
61:         end for
62:         Mediatore corrente successivo
63:     until tutti i mediatori esplorati or esiste un mediatore forzato
64: end procedure
```

4.4 Ricerca su grafo degli agenti

Nelle sezioni precedenti si è proposta una tecnica di ricerca della struttura ottima del sistema multiagente che esplori lo spazio delle soluzioni. Si tratta quindi di navigare su un grafo i cui nodi rappresentano le possibili soluzioni. Questo approccio ha dei vantaggi in termini di elasticità, in quanto permette di definire politiche articolate relative alla struttura della soluzione finale. Per contro ha un numero di nodi, o possibili soluzioni, molto alto e quindi la sua applicabilità è soggetta a limiti. Un approccio alternativo è quello di abbandonare la generalità del grafo di soluzioni, utilizzando un altro grafo i cui nodi rappresentino gli agenti stessi. Nel primo caso i vicini di un nodo rappresentano altre soluzioni complete, nel secondo rappresentano gli agenti con i quali tale nodo potrà essere aggregato. Nel primo caso l'algoritmo può operare su molteplici soluzioni diverse contemporaneamente, nel secondo invece su una sola, ignorando possibili alternative. La differenza è visibile confrontando le Figure 4.11 e 4.12.

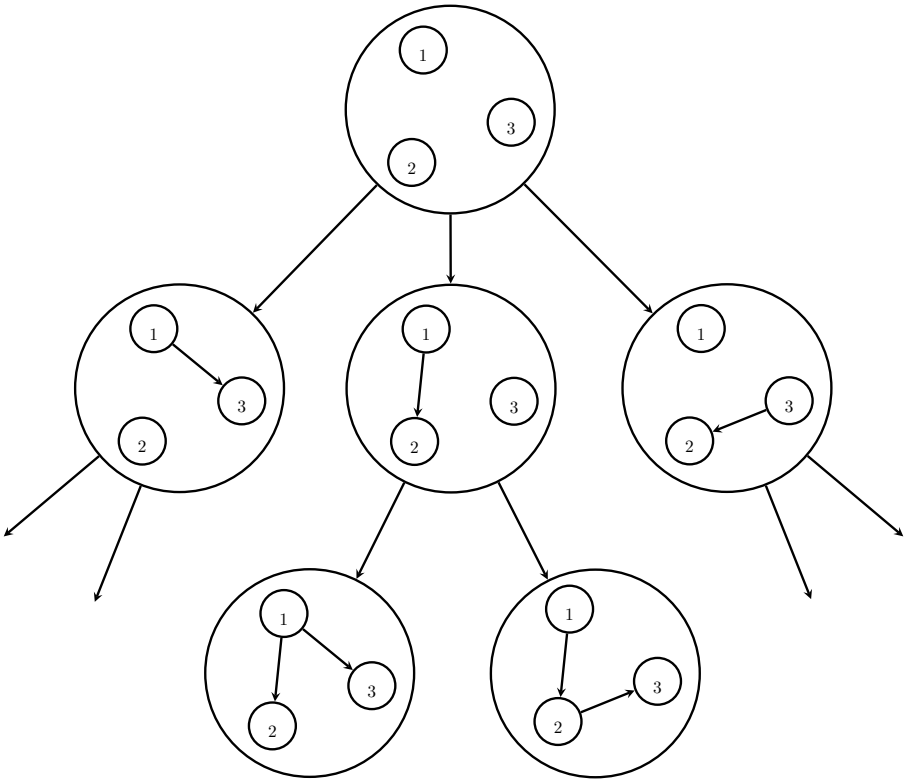

Figura 4.11: Grafo delle soluzioni

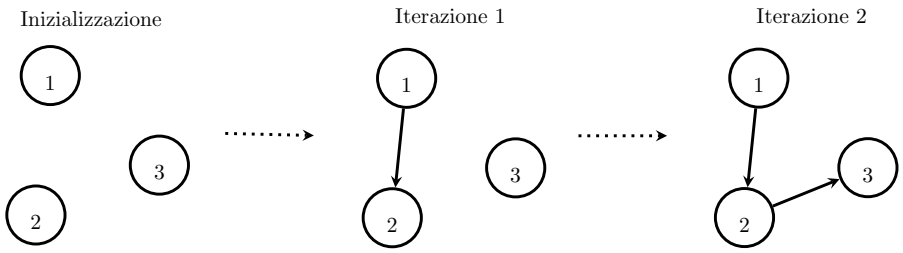

Figura 4.12: Grafo degli agenti

Nella Figura 4.11 soluzioni diverse si trovano in nodi diversi, nella Figura 4.12 invece in iterazioni diverse. Scegliendo la formalizzazione con grafo di agenti si possono utilizzare algoritmi pensati per costruire alberi di supporto, ossia dei sottografi senza cicli. L'uso di questo approccio quindi restringerà lo spazio delle soluzioni ai soli alberi.

Il numero di agenti diagnostici è, come sempre, noto a priori. Quello degli agenti mediatori invece, nel caso generale, no. Qualora si fosse in possesso di questa informazione è possibile applicare uno degli algoritmi disponibili per la costruzione del *minimum spanning tree*, MST [78]. Diversamente, si potranno utilizzare gli *alberi di Steiner* [79]. E' necessario il grafo sia orientato per poter determinare l'ordinamento dei nodi e definire la corretta gerarchia.

Per applicare tali algoritmi è necessario sia definita su ogni arco una funzione di costo che permetta di prendere una decisione locale in merito a quale arco aggiungere alla soluzione. Inoltre un'azione può sia migliorare che peggiorare la soluzione, i costi sugli archi dovranno appartenere all'insieme dei numeri reali. Questo quindi indirizza verso l'algoritmo di *Johnson* [78], efficiente nel caso di grafi sparsi, o quello di *Floyd-Warshall* [80], adatto per grafi densi. Principale difficoltà in questo caso è quella di definire una funzione di costo locale che sia in grado di rispecchiare in modo sufficientemente accurato la variazione nella bontà globale della soluzione. Poichè non è stato possibile individuare una funzione di costo con tale caratteristica, si è privilegiato l'approccio a grafo delle soluzioni.

Capitolo 5

Architettura del sistema software

In questo capitolo viene presentata l'architettura del modulo di ricerca della struttura del sistema multiagente oggetto di questa tesi e del sistema in cui esso si integra. Il sistema già esistente è stato sviluppato in Java nell'ambito del progetto FCTMAS [14], esso svolge preprocessing dei dati, addestramento degli agenti diagnostici e anomaly detection su una struttura ad albero predefinita. Il modulo sviluppato per questa tesi si colloca tra la fase di preprocessing e quella di analisi dei dati online intervenendo a modificare tale struttura predefinita. Per informazioni più dettagliate in merito all'implementazione si rimanda all'Appendice A.

5.1 Struttura di alto livello

Il sistema FCTMAS, descritto nella Sezione 3.2.2, nel suo complesso si compone di tre package, come mostrato in Figura 5.1. Di seguito si offre per ciascuno una breve descrizione.

MCS: *Mission Control System*, si occupa di avviare il sistema. Esso utilizza il modulo di preprocessing, il modulo di ricerca ed infine avvia JADE tramite l'AppMainController. JADE si occupa di leggere il log proveniente dal satellite e calcolare online la probabilità di anomalia del sistema, come descritto nella Sezione 3.2.2.

Preprocessing: Si occupa di analizzare i dati del log, isolare i diversi eventi e costruire un modello per ogni variabile tramite reti neurali o catene di Markov, come descritto nella Sezione 3.2.2. Se i dati disponibili per una variabile sono sufficienti ed il modello è abbastanza accurato, lo salva in un file. Ciascun modello sarà poi associato ad un agente diagnostico. Salva

inoltre la configurazione di default flat, nella quale tutti gli agenti diagnostici negoziano come figli della radice.

OptimalStructure: Legge il file contenente i modelli salvati e svolge la ricerca della struttura ottima. Tale struttura è salvata in un file di configurazione affinchè sia letta dal package MCS ed utilizzata dall'AppMainController. La scelta di quale algoritmo utilizzare e di una serie di altri parametri può essere specificata in un file di configurazione. Questo modulo fa uso di JADE per calcolare le probabilità di anomalia degli agenti diagnostici utilizzando il file di log.

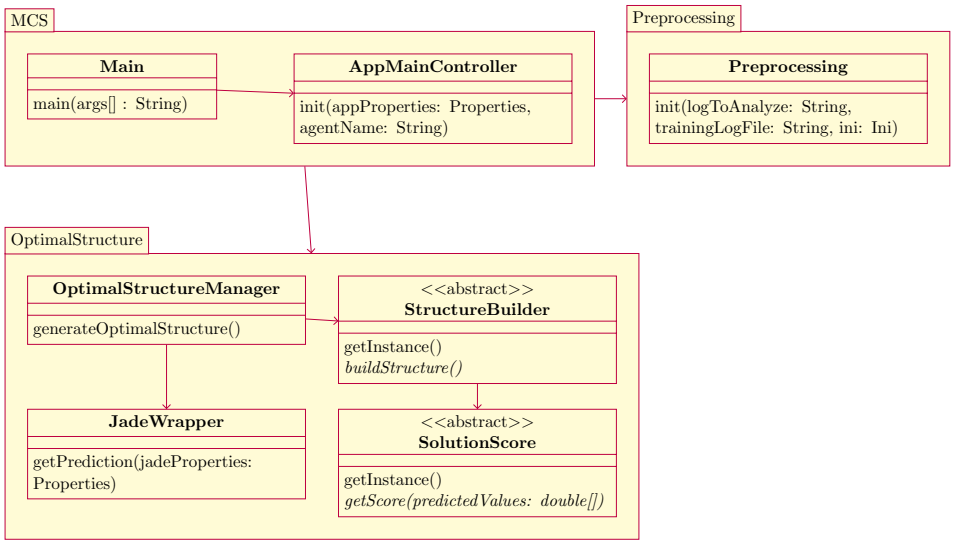

Figura 5.1: Diagramma dei package e delle classi principali del sistema FCTMAS.

5.2 Package OptimalStructure

Il modulo OptimalStructure implementato in questa tesi è di semplice estendibilità relativamente agli algoritmi implementati, alle funzioni di aggregazione, alle modalità del calcolo dello score di una soluzione ed alle topologie di struttura desiderata. In Figura 5.2 è riportato un diagramma UML di alto livello che riporta le classi e le dipendenze principali.

L'OptimalStructureManager inizializza ed avvia la costruzione della struttura istanziando un oggetto di tipo StructureBuilder, classe astratta estesa da un'implementazione diversa per ogni possibile algoritmo. Informazioni quali la struttura di soluzione desiderata, l'algoritmo da applicare, la neighbor function e la funzione di aggregazione sono scritti su un file di configurazione.

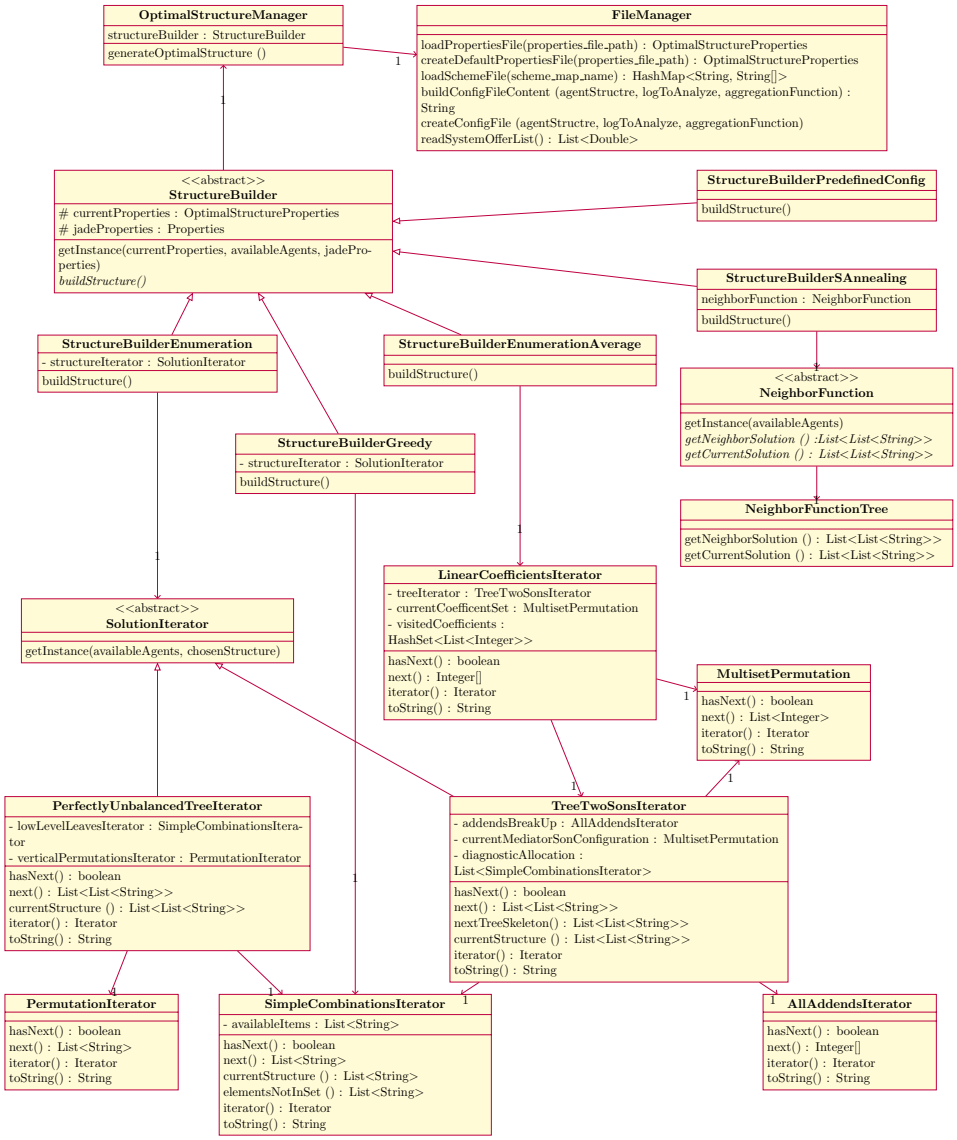

Figura 5.2: UML di alto livello del package StructureBuilder

Le classi relative agli algoritmi implementati sono:

- **StructureBuilderEnumeration -** Esplora l'intero spazio delle strutture di alberi aventi un certo numero di foglie. Questo'algoritmo è descritto nella Sezione 4.2.1.

- **StructureBuilderEnumerationAverage -** Esplora lo spazio dei coefficienti, operando quindi direttamente nello spazio delle classi di equivalenza definite nel caso della funzione di aggregazione media. Questo'algoritmo è descritto nella Sezione 4.2.1 applicato allo spazio delle soluzioni descritto nella Sezione 3.5.3.

- **StructureBuilderSAnnealing -** Applica il simulated annealing, descritto nella Sezione 4.3.1, sullo spazio delle possibili strutture.

- **StructureBuilderGreedy -** Applica l'algoritmo greedy, descritto nella Sezione 4.3.2, sullo spazio delle possibili strutture.

- **StructureBuilderPredefinedConfig -** Scrive la configurazione di default, ossia tutti gli agenti diagnostici negoziano contemporaneamente essendo figli della radice.

Gli algoritmi fanno uso di diversi iteratori, necessari all'esplorazione delle possibili strutture. Si può notare come alcuni iteratori facciano a loro volta uso di altri iteratori, ciò è strettamente legato all'algoritmo usato per esplorare lo spazio dei coefficienti e degli alberi.

La classe FileManager offre una serie di metodi statici necessari per l'input/output tra i vari sistemi e per la lettura dei file di configurazione.

Per valutare le soluzioni è utilizzata la classe astratta SolutionScore implementata in diversi modi, si veda Figura 5.3. Sono attualmente implementate la confusion matrix, quindi una misura di accuratezza, e la cost matrix, una misura di costo.

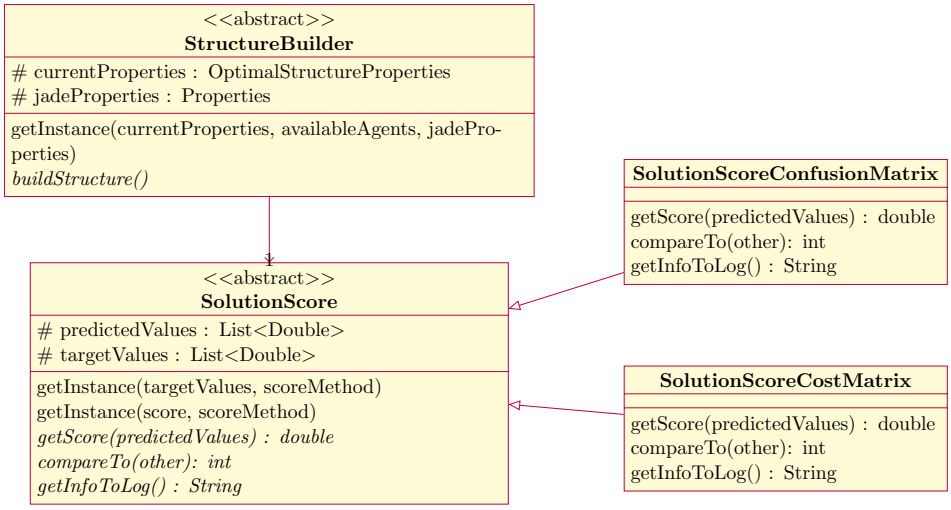

Figura 5.3: Gerarchia delle classi per la valutazione di una soluzione

Capitolo 6

Risultati sperimentali

In questo capitolo sono riportati i risultati ottenuti applicando gli algoritmi di ricerca proposti a diversi casi di test. Si verificano le performance degli algoritmi relativamente all'accuratezza, alla dimensione del problema gestibile ed alla effettiva corrispondenza tra la struttura della soluzione e la struttura del sistema.

E' stato possibile addestrare i modelli utilizzando dei dati reali prodotti dal funzionamento del satellite Cryosat-2 dal 23/07/2013 al 22/08/2013. Un caso di test è generato direttamente da questi dati, sottoponendo agli agenti diagnostici un log forgiato in modo da contenere anomalie.

6.1 Cost matrix e timestamp significativi

Le possibili strutture per il sistema multiagente sono valutate a partire da un log i cui timestamp sono corredati da un'annotazione che distingue quelli nominali da quelli anomali. Il log è analizzato dal sistema FCTMAS, in modo da calcolare le probabilità di anomalia associate a ciascun agente diagnostico. Tali probabilità sono in seguito utilizzate dal modulo di ricerca per valutare tutte le soluzioni individuate. La probabilità di anomalia calcolata dalla radice dell'albero delle aggregazioni sarà utilizzata per determinare lo stato complessivo del sistema. Qualora questa probabilità superi il 20% il sistema sarà classificato come anomalo, diversamente sarà nominale. La soglia è stata fissata al 20% in virtù di alcuni risultati sperimentali [14]. L'obiettivo di questa tesi è sviluppare un sistema di anomaly detection, perciò vogliamo che siano preferite soluzioni in grado di identificare correttamente il maggior numero di anomalie possibile. Poichè le anomalie sono un'occorrenza rara cifre di merito quali l'accuratezza sono poco efficaci, in quanto su un log che contenga decine di migliaia di timestamp e poche anomalie, come riportato in Tabella 6.2, la differenza tra l'accuratezza delle diverse soluzioni sarà nell'ordine dello 0.01%. Optiamo quindi per un diverso approccio, più flessibile, che possa meglio adattarsi a questo caso applicativo, una cost

matrix [81]. Applicare una *cost matrix* richiede di distinguere quattro possibili esiti associati alla classificazione di ogni timestamp: *true negative* se il sistema è correttamente classificato come nominale, *true positive* se il sistema è correttamente classificato come anomalo, *false positive* se il sistema è nominale ma classificato anomalo e *false negative* se il sistema è anomalo ma è classificato come nominale.

A ciascuno dei quattro esiti è associato un costo dal valore reale. Il costo di una soluzione sarà dato dalla sommatoria del numero di timestamp associati a ciascuno dei quattro esiti moltiplicati per il relativo costo. Poichè vogliamo che l'attenzione sia focalizzata sulla capacità di una soluzione di identificare correttamente le anomalie, i costi per le predizioni corrette, TP e TN, sono posti a zero. I timestamp classificati in modo erroneo invece non hanno tutti la stessa gravità, in quanto quella attribuibile ad un falso allarme è minore rispetto a quella di un'anomalia non individuata. Ai falsi positivi quindi si attribuisce costo 20 ed ai falsi negativi 50. La struttura della cost matrix è riportata in Tabella 6.1.

	Realmente nominale	Realmente anomalo
Classificato nominale	TN: 0	FN: 50
Classificato anomalo	FP: 20	TP: 0

Tabella 6.1: Struttura della cost matrix utilizzata per valutare le soluzioni.

Dopo aver svolto l'analisi dell'intero log applicando una specifica struttura e calcolato quanti timestamp sono associati ai quattro possibili esiti, il costo di una soluzione sarà calcolato nel modo seguente:

$$\text{Costo} = FP \cdot 20 + FN \cdot 50$$

Considerando funzioni di aggregazione che restituiscono un valore compreso fra il massimo ed il minimo degli argomenti, non tutti i timestamp sono significativi. Se per un dato istante nessun agente è singolarmente anomalo, certamente la funzione di aggregazione non potrà classificare il sistema nel suo complesso come anomalo. Poichè la bontà di una soluzione si determina in base alla sua capacità di predire correttamente le anomalie nel sistema, considerare i timestamp che sicuramente non saranno anomali non ci fornisce nessuna informazione aggiuntiva. E' quindi possibile filtrare i dati in modo da selezionare, tra tutti i timestamp relativi a variabili di cui è disponibile un modello nominale, quelli nei quali almeno un agente sia anomalo.

Log	Timestamps		
	Originali	Evento	Significativi
Un giorno	9 518	111	0
Un mese	336 549	5 940	0

Tabella 6.2: In tabella sono riportati i numeri di timestamp esistenti nei due log disponibili del satellite Cryosat-2. Si confronta il numero totale di timestamp presenti nel log, quelli relativi ad un evento per cui si è potuto costruire un modello e, di questi, quelli nei quali esista almeno un agente anomalo.

Come si può osservare in Tabella 6.2, il numero di timestamp necessari per valutare una soluzione cala drasticamente fino ad azzerarsi passando dal log originale alla sua versione filtrata. Risulta evidente come le anomalie siano un'occorrenza rara, inesistente nel caso dei due log disponibili, per le variabili di cui è stato addestrato il modello nominale. Questo ha delle conseguenze importanti:

- **Problema poco vincolato:** si rende evidente come le anomalie siano un'evenienza molto rara e quindi, con log anche di qualche mese sarà difficile discriminare tra possibili soluzioni. Ci si aspetta quindi un gran numero di soluzioni pressochè equivalenti.
- **Valutazione soluzione più rapida:** una riduzione drastica nel numero di timestamp da considerare per ogni soluzione, sfruttando la rarità delle anomalie, consente di abbattere di diversi ordini di grandezza il tempo necessario a valutare una soluzione.
- **Minore difficoltà del problema:** incrementare la dimensione del problema aggiungendo agenti diagnostici non ha solo un impatto legato alla crescita esponenziale del numero di soluzioni possibili, ma incide anche sul costo di valutazione di ogni soluzione. Inserire un nuovo agente implica aggiungere tutti i timestamp relativi ai suoi eventi.

6.2 Dimensione del problema gestibile

6.2.1 Algoritmo enumerativo

La ricerca enumerativa esplora tutti i possibili alberi appartenenti allo spazio delle soluzioni. In Tabella 6.3 sono riportati i numeri di possibili alberi e di vettori di coefficienti, legati alla funzione di aggregazione media. Nel caso generale devono essere esplorati tutti gli alberi, mentre applicando la funzione media è possibile esplorare i vettori dei coefficienti con una significativa riduzione nel numero di soluzioni. Si considera il caso a tutti i mediatori sia applicata la stessa funzione di aggregazione e quindi le soluzioni si

differenzino solo per la loro struttura. E' inoltre riportata una stima del tempo necessario ad esplorare tutto lo spazio delle soluzioni. Se si considera la negoziazione cooperativa è possibile calcolare la probabilità di anomalia complessiva del sistema utilizzando JADE, come implementato nel sistema FCTMAS già esistente e descritto nella Sezione 3.2.2. In questo caso sono necessari 23 secondi per valutare ogni soluzione utilizzando il log di un giorno, 84 se si fosse utilizzato quello di un mese. Il modulo sviluppato contestualmente a questa tesi è in grado di valutare 260 000 soluzioni al secondo. Si rileva quindi un incremento prestazionale sostanziale. Considerando un tempo di calcolo accettabile inferiore alla settimana, la dimensione del problema trattabile aumenta da 6 a 10 agenti, potenzialmente 11 con qualche ottimizzazione.

Applicando la media si possono esplorare 2 000 000 di soluzioni al secondo, rendendo gestibili problemi con 12 agenti diagnostici.

I test sono stati effettuati su un processore Intel Core i7-4770K 3.50 GHz.

Diagnostici	Alberi	Durata		Coefficienti	Durata
		Jade	Attuale		
5	$3.3 \cdot 10^2$	2 h	0 s	$1.8 \cdot 10^2$	0 s
6	$5.6 \cdot 10^3$	36 h	0 s	$1.8 \cdot 10^3$	0 s
7	$1.3 \cdot 10^5$	35 d	0 s	$2.1 \cdot 10^4$	0 s
8	$4.0 \cdot 10^6$	2.9 y	15 s	$3.0 \cdot 10^5$	0 s
9	$1.5 \cdot 10^8$	-	9 m	$4.9 \cdot 10^6$	2 s
10	$7.6 \cdot 10^9$	-	8 h	$9.1 \cdot 10^7$	45 s
11	$4.4 \cdot 10^{11}$	-	19 d	$1.8 \cdot 10^9$	15 m
12	$3.1 \cdot 10^{13}$	-	3.7 y	$4.0 \cdot 10^{10}$	5 h

Tabella 6.3: Numero di possibili soluzioni per dimensione del problema e tempo necessario ad esplorarle.

Restrizione nel livello massimo dell'albero

All'aumentare del numero di agenti diagnostici saranno esplorati anche alberi di elevata profondità. Considerando l'impatto che le progressive aggregazioni possono avere sulla probabilità di anomalia, "appiattendone" il valore e degradando l'informazione trasmessa al livello superiore, è ragionevole prevedere un limite alla profondità dell'albero. In Tabella 6.4 si riportano il numero di alberi esistenti per due possibili limiti e il rispettivo tempo di elaborazione.

Diagnostici	Livello 2		Livello 3	
	Num Alberi	Durata	Num Alberi	Durata
9	$1.3 \cdot 10^5$	0 s	$1.3 \cdot 10^7$	50 s
10	$1.1 \cdot 10^6$	4 s	$2.8 \cdot 10^8$	17 m
11	$1.1 \cdot 10^7$	42 s	$6.8 \cdot 10^9$	7 h
12	$1.1 \cdot 10^8$	7 m	$1.8 \cdot 10^{11}$	8 g
13	$1.3 \cdot 10^9$	1 h	$5.8 \cdot 10^{12}$	258 g
14	$1.6 \cdot 10^{10}$	17 h	$2.0 \cdot 10^{14}$	-
15	$2.1 \cdot 10^{11}$	9 g	-	-
16	$3.0 \cdot 10^{12}$	133 g	-	-

Tabella 6.4: Il valore del livello si riferisce al numero massimo di archi tra la radice e la foglia più lontana. Nel caso di livello 2 radice e foglie sono quindi separate al massimo da un mediatore.

Lo spazio delle soluzioni mantiene una crescita esponenziale rispetto al numero degli agenti diagnostici, tuttavia con una significativa riduzione rispetto al caso degli agli alberi non ristretti. Applicando un limite al livello 2 si passa da n^n alberi, senza restrizioni, a 10^{n-4} alberi, rendendo gestibile $n = 14$. Da menzionare che il numero di soluzioni esplorabili al secondo sarà sempre più basso a causa del maggior numero di agenti diagnostici coinvolti in ogni timestamp.

E' possibile applicare una restrizione anche agli alberi associati ai vettori di coefficienti per la funzione di aggregazione media. Poichè uno stesso vettore di coefficienti può essere generato da soluzioni strutturalmente diverse, non è possibile limitare con precisione il livello degli alberi ma è possibile limitare il valore massimo dei coefficienti. Un vettore $\bar{w}$ è accettabile se il valore di tutti i suoi elementi è inferiore ad un valore di soglia. Determinare un opportuno valore di soglia richiede di considerare le sue implicazioni sul risultato finale. Poichè la probabilità di anomalia calcolata dall'agente diagnostico viene divisa per il relativo coefficiente w, se il suo valore fosse superiore a 10 quell'agente diagnostico risulterebbe pressochè irrilevante nel determinare la probabilità di anomalia globale per l'intero sistema. Una soluzione associata ad un vettore di coefficienti contenente valori superiori a 10 per alcuni agenti diagnostici non permetterebbe di individuare un'anomalia nemmeno se questi esibissero una probabilità di anomalia superiore al 90%.

Il coefficiente w è calcolato come produttoria tra i fratelli dell'agente e quelli di ogni suo antenato, ciò significa che all'aumentare della dimensione

del problema anche il valore massimo che può essere assunto dai coefficienti tenderà a crescere. Applicare una restrizione ai vettori di coefficienti accettabili in base al coefficiente di valore massimo pone un importante limite alla dimensione massima del problema trattabile usando la funzione di aggregazione media. Dato un numero di agenti diagnostici e il suo relativo spazio delle soluzioni, si osserva che all'aumentare del livello dell'albero tende ad aumentare anche il valore del coefficiente massimo contenuto nel vettore $\bar{w}$. La struttura associata ad un vettore il cui coefficiente più alto sia il minimo su tutto lo spazio delle soluzioni è la flat. Poichè nella struttura flat tutti gli agenti sono posti come figli della radice, il coefficiente di tutti sarà identico e pari al numero di agenti diagnostici stessi. Qualora il valore massimo desiderato per i coefficienti sia 10, l'ultimo caso gestibile sarà quello con 10 agenti, con l'unica configurazione ammissibile flat. Si è quindi reso evidente come la funzione media, a fronte di uno spazio delle soluzioni notevolmente più piccolo ed una velocità di valutazione delle soluzioni sette volte superiore, sia probabilmente inadatta per affrontare problemi oltre quella dimensione a causa dell'eccessiva degradazione a cui l'informazione è soggetta.

Funzioni di aggregazioni differenziate

E' possibile effettuare una ricerca non solo sullo spazio degli alberi, ma considerando anche diverse possibili funzioni di aggregazione associabili ad ogni mediatore. In questa tesi la ricerca è effettuata con le funzioni *negoziazione cooperativa* e *massimo*. Il massimo è introdotto affinchè possa evidenziare eventuali anomalie che sarebbero nascoste dall'aggregazione di molti agenti, oppure anomalie deboli. Nel caso generale, ogni soluzione ha un numero di diverse configurazioni delle funzioni di aggregazione pari a f^m, con f il numero di funzioni disponibili ed m il numero di mediatori. La difficoltà del problema pertanto aumenta in modo significativo rispetto al caso in cui si usi una sola funzione di aggregazione per tutti i mediatori. A titolo di esempio si consideri il caso in cui si usino 2 funzioni di aggregazione, 10 agenti diagnostici ed alberi limitati al livello 2. Lo scheletro con il maggior numero di agenti mediatori ne contiene 5, ciascuno con due figli agenti diagnostici, più la radice. Il numero di possibili varianti per le funzioni di aggregazione in quello scheletro è $2^6 = 64$. Così come nel caso del branch and bound descritto nella Sezione 4.2.2 non è stato possibile individuare una funzione euristica non banale per effettuare pruning.

6.2.2 Simulated annealing

Il simulated annealing è molto più adatto dell'enumerazione ad esplorare spazi di stati molto vasti. L'unico problema è la qualità della soluzione effettivamente raggiunta se lo spazio delle soluzioni dovesse diventare troppo vasto e l'algoritmo non riuscisse ad esplorarlo adeguatamente. Poichè nei

log del satellite Cryosat-2 disponibili non vi sono dati sufficienti per addestrare un numero abbastanza alto di modelli, e quindi di agenti diagnostici, non si può fornire un'indicazione in merito alla dimensione trattabile. Lo pseudocodice del simulated annealing è riportato nell'Algoritmo 1.

Simulated annealing ibrido

Si vuole verificare se sia possibile combinare i vantaggi del simulated annealing e dell'algoritmo enumerativo, ad esempio applicando la neighbor function sullo scheletro dell'albero e poi valutando tutte le allocazioni delle foglie, ossia il loro essere figli di uno specifico agente mediatore.

Sfortunatamente è proprio il numero di possibili allocazioni delle foglie a crescere in modo più rapido. Data una sequenza ordinata di agenti diagnostici, una specifica allocazione può essere rappresentata indicando per ciascuno il nome del nodo padre. Esplorare tutte le allocazioni equivale ad effettuare delle permutazioni multiset sul vettore contente i nomi dei nodi padre, ossia permutazioni nelle quali esistono elementi equivalenti, i figli dello stesso nodo. Essendo possibile calcolare il numero di permutazioni senza generarle esplicitamente, è stato possibile calcolare il numero di soluzioni per $n > 11$ sommando le permutazioni per ogni scheletro. Come si può osservare in Tabella 6.5 il numero di permutazioni delle foglie, dato uno scheletro, è circa uguale allo spazio di tutte le soluzioni per il problema con 5 foglie in meno. Il valore riportato delle permutazioni per scheletro è medio sul totale degli scheletri e delle possibili strutture. In virtù dell'analogia con le permutazioni multiset, scheletri con un ridotto numero di mediatori avranno un minor numero di permutazioni delle foglie, scheletri i cui mediatori abbiano pochi figli invece avranno un maggior numero di insiemi, dalla dimensione inferiore, quindi molte allocazioni in più.

Diagnostici	Num Alberi	Num scheletri	Alberi/Scheletri
11	$4.4 \cdot 10^{11}$	$2.5 \cdot 10^{5}$	$1.7 \cdot 10^{6}$
12	$3.1 \cdot 10^{13}$	$1.8 \cdot 10^{6}$	$1.6 \cdot 10^{7}$
13	$2.5 \cdot 10^{15}$	$1.5 \cdot 10^{7}$	$1.7 \cdot 10^{8}$
14	$2.4 \cdot 10^{17}$	$1.3 \cdot 10^{8}$	$1.8 \cdot 10^{9}$

Tabella 6.5: Confronto tra il numero di alberi e il numero di scheletri.

Restringendo lo spazio delle soluzioni agli alberi con livello massimo 2, il rapporto tra alberi e scheletri si riduce abbastanza da rendere gestibile l'analisi di un singolo scheletro fino a $n = 16$.

Diagnostici	Num Alberi	Num scheletri	Alberi/Scheletri
11	$1.1 \cdot 10^7$	$1.4 \cdot 10^2$	$7.9 \cdot 10^4$
12	$1.1 \cdot 10^8$	$2.3 \cdot 10^2$	$5.1 \cdot 10^5$
13	$1.3 \cdot 10^9$	$3.7 \cdot 10^2$	$3.5 \cdot 10^6$
14	$1.6 \cdot 10^{10}$	$6.0 \cdot 10^2$	$2.7 \cdot 10^7$
16	$3.0 \cdot 10^{12}$	$1.5 \cdot 10^3$	$1.8 \cdot 10^9$
18	$7.0 \cdot 10^{14}$	$4.1 \cdot 10^3$	$1.9 \cdot 10^{11}$
20	$2.0 \cdot 10^{17}$	$1.0 \cdot 10^4$	$1.7 \cdot 10^{12}$

Tabella 6.6: Confronto tra il numero di alberi e il numero di scheletri di livello massimo 2.

Alla luce di questi dati l'approccio simulated annealing ibrido proposto, presentando un basso numero di scheletri ciascuno con un alto costo di valutazione, sarebbe poco efficiente. Inoltre la progressione fattoriale nel numero di allocazioni comunque limita il problema trattabile a poco oltre quanto raggiungibile con l'enumerazione.

Sfruttando il limitato numero di scheletri, si può concepire una diversa tecnica ibrida. Ossia un approccio stocastico che enumeri gli scheletri, generando per ciascuno di essi un numero fissato di diverse allocazioni delle foglie. In virtù dei dati della Tabella 6.6, si potrebbe ottenere un significativo aumento nella dimensione del problema gestibile valutando un massimo di 10^7 permutazioni per ogni scheletro. Tale valore sarà decrescente all'aumentare del numero di scheletri. Sono esplorabili in tempi ragionevoli 10^6 permutazioni con 16 agenti e 10^5 con 20. Nuovamente, in assenza di dati reali tali da permettere di disporre di un sufficiente numero di agenti diagnostici non è possibile verificare l'efficacia di questa tecnica.

6.2.3 Algoritmo greedy

Come descritto successivamente l'algoritmo greedy genera un numero di possibili azioni pari a circa n^4, con n il numero di agenti diagnostici. In virtù della complessità computazionale polinomiale il problema è trattabile sino ad un numero di variabili molto più alto rispetto a quanto possibile con l'enumerazione. Ponendo un numero di agenti diagnostici pari a 10 l'algoritmo greedy può essere eseguito migliaia di volte al secondo, applicando multithreading.

L'ampiezza dello spazio di ricerca può essere modulata scegliendo la dimensione massima del set di agenti diagnostici oggetto di ogni azione.

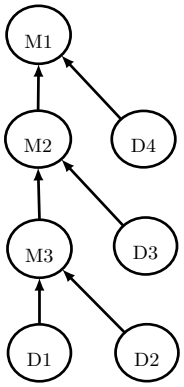

Figura 6.1: Albero perfettamente sbilanciato con almeno due figli per nodo

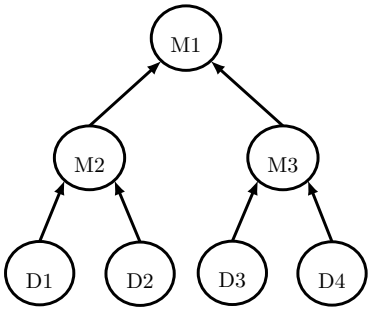

Figura 6.2: Albero binario perfettamente bilanciato

Complessità computazionale

In questa sezione si offre una stima pessimistica della complessità computazionale dell'algoritmo greedy proposto, si veda l'Algoritmo 2.

In merito al numero massimo di agenti mediatori, non esiste, in generale, un modo per calcolare quanti saranno i nodi interni di un albero dato il numero delle foglie. Tuttavia nel nostro caso ogni nodo intermedio dovrà avere almeno due figli, ciò permette di escludere, in particolare, gli alberi *perfettamente sbilanciati* aventi una sola foglia e una sequenza arbitrariamente lunga di nodi intermedi con un solo padre e un solo figlio, che termina con il nodo radice. Poichè ogni nodo intermedio deve avere almeno due discendenti, il numero complessivo degli agenti mediatori non potrà mai eguagliare quello delle foglie, essendo al più $num_{foglie} - 1$.

Sono qui proposte due configurazioni nelle quali il numero di nodi intermedi è il massimo possibile:

- **Alberi perfettamente sbilanciati** con almeno due figli per ogni nodo, come in Figura 6.1, per i quali varrà sempre $num_{interni} = num_{foglie} - 1$.

- **Alberi perfettamente bilanciati** i cui nodi intermedi abbiano esattamente k figli, come in Figura 6.2. Il numero di nodi intermedi è dato dalla relazione $num_{interni} = \frac{num_{foglie}-1}{k-1}$ il cui massimo si ha per $k = 2$, quindi per gli alberi binari.

Pertanto il numero massimo di agenti mediatori è approssimabile con il numero di agenti diagnostici.

Si calcola ora una stima pessimistica della complessità computazionale tramite il numero di azioni generate da ogni procedura dell'algoritmo greedy.

- **InitializePossibleActionsList** - La procedura sviluppa tutte le *combinazioni semplici senza ripetizioni* da 2 a *maxSetSize* elementi nell'insieme contenente tutti gli agenti. Il numero di possibili insiemi è:

$$T_{initialize}(num_{ag}) = \sum_{setSize=2}^{maxSetSize} \binom{num_{ag}}{setSize} < maxSetSize \cdot \binom{num_{ag}}{2}$$
$$\simeq maxSetSize \cdot num_{ag}^2 \simeq O(num_{ag}^2)$$

- **newMediatorAgentActions** - La procedura si compone di tre cicli annidati. Il massimo numero di mediatori è pari a num_{ag} e il coefficiente binomiale dal valore maggiore ha complessità $O(num_{ag}^2)$ e si ottiene con $setSize = 2$.

$$T_{new_mediator} < num_{ag} \cdot maxSetSize \cdot \binom{num_{ag}}{setSize}$$
$$< num_{ag} \cdot maxSetSize \cdot num_{ag}^2 \simeq O(num_{ag}^3)$$

- **addDiagnosticAgentActions** - Anche questa procedura si compone di tre cicli annidati, identici a newMediatorAgentActions. La complessità computazionale è quindi identica.

$$T_{add_diagnostic} \simeq O(num_{ag}^3)$$

- **TerminationCondition** - Si può implementare come semplice soglia, o sul costo raggiunto dalla soluzione o sulla disponibilità di altri agenti diagnostici ancora da allocare.

$$T_{termination} = O(k)$$

- **BuildTree** - La funzione chiama in sequenza le procedure e sceglie l'azione da applicare. Poichè ogni azione è legata ad almeno un agente, rimuovendolo dalla lista di agenti disponibili, il massimo numero di iterazioni sarà pari al numero di agenti diagnostici.

$$T_{BuildTree} = T_{initialize} + num_{ag} \cdot (T_{new_mediator} + T_{add_diagnostic} + T_{termination})$$
$$= num_{ag}^2 + num_{ag} \cdot \left(num_{ag}^3 + num_{ag}^3 + k\right)$$
$$\simeq num_{ag}^2 + num_{ag}^4 \simeq O(num_{ag}^4)$$

Anche se il valore di complessità computazionale stimato è pessimistico, è comunque evidente che il problema sia trattabile con istanze anche di grande dimensione.

6.3 Proprietà del sistema multiagente per l'identificazione delle anomalie

6.3.1 Che tipologie di anomalie si possono riconoscere

Le capacità di identificazione delle anomalie del modello multiagente proposto in questa tesi sono molto "verticali" in quanto un mediatore può essere classificato come anomalo solo in due casi:

- Un agente suo figlio ha un alto valore di anomalia.
- Più agenti suoi figli hanno un medio/basso valore di anomalia.

Il modello attuale non permette di riconoscere anomalie legate alla concordanza tra valori di agenti diversi.

6.3.2 Propagazione dell'anomalia verso la radice

Aggregare un sottoinsieme di agenti diagnostici e mediatori permette di riconoscere meglio anomalie che li riguardino collettivamente. La progressiva aggregazione dei vari livelli gerarchici tuttavia può rendere anomalie rilevate da agenti posti in profondità nell'albero poco significative, se non irrilevanti, rispetto al sistema nel suo complesso. Questo è legato al fatto che ad ogni livello il risultato di una aggregazione tra più agenti è riportato al livello successivo da un solo mediatore, riducendone l'impatto. L'uso di pesi nell'aggregazione, come nella negoziazione cooperativa, può mitigare l'effetto ma non annullarlo. Ciò comporta un ulteriore limite alla dimensione massima del problema trattabile, un numero di agenti troppo alto renderebbe sempre meno significativo il risultato conseguito rendendo una accurata identificazione difficile.

6.4 Prove sperimentali

Nelle seguenti sezioni si presentano i risultati delle prove sperimentali. Per ogni caso analizzato è riportata una descrizione e una tabella con le probabilità di anomalia calcolate dagli agenti diagnostici. Per agevolare la lettura, sono riportati solo i valori diversi da zero.

Le soluzioni sono cercate sotto forma di albero, dapprima non sottoposto ad alcun vincolo, poi limitato a due soli livelli. Infine, si cerca la struttura ottima anche introducendo la possibilità di differenziare le funzioni di aggregazione applicate ai vari agenti mediatori.

Tra gli insiemi delle soluzioni individuate nei tre casi sussiste la seguente relazione di inclusione:

$$\text{AlberiDueLivelli} \subset \text{Alberi} \subset \text{AlberiMultiFunzione} \tag{6.1}$$

Se definiamo una funzione $bestSolCost : solutionSet \mapsto R$ che, dati gli insiemi delle soluzioni precedentemente indicati, restituisce il costo associato alla migliore soluzione, si ha la seguente relazione:

$$\begin{aligned} bestSolCost\,(\text{AlberiDueLivelli}) &\geq bestSolCost\,(\text{Alberi}) \\ &\geq bestSolCost\,(\text{AlberiMultiFunzione}) \end{aligned} \tag{6.2}$$

Lo stesso comportamento si ottiene applicando la funzione di aggregazione media su alberi non vincolati e su alberi il cui coefficiente massimo, o peso dell'agente nella combinazione lineare, sia limitato a 10. Ossia:

$$\text{AlberiCoeffMinore 10} \subset \text{Alberi} \tag{6.3}$$

e quindi:

$$bestSolCost\,(\text{AlberiCoeffMinore 10}) \geq bestSolCost\,(\text{Alberi}) \tag{6.4}$$

Definiamo inoltre una funzione $bestSolSet : solutionSet \mapsto solutionSet$ che, dato un solution set, restituisce un suo subset composto dalle soluzioni ottime equivalenti.

Dati due solution set A e B, con A $\subset$ B:

$$\begin{cases} bestSolSet\,(\text{A}) \subseteq bestSolSet\,(\text{B})\,, & \text{se } bestSolCost\,(\text{A}) = bestSolCost\,(\text{B}) \\ bestSolSet\,(\text{A}) \nsubseteq bestSolSet\,(\text{B})\,, & \text{se } bestSolCost\,(\text{A}) > bestSolCost\,(\text{B}) \end{cases} \tag{6.5}$$

$$\text{Inoltre: } bestSolSet\,(\text{B}) \nsubset bestSolSet\,(\text{A})$$

6.4.1 Log di un mese

Per questo test si è utilizzato il log originale di un mese del satellite Cryosat-2, dal 23/07/2013 al 22/08/2013, modificato in modo da includere dei timestamp in cui il log contenga valori non nominali. I dati riportati in Tabella 6.7 sono le probabilità di anomalia calcolate dal sistema FCTMAS sulla base del log modificato. Affinchè il sistema nel suo complesso sia considerato anomalo, la probabilità di anomalia calcolata dalla radice tramite l'aggregazione delle probabilità dei suoi figli deve essere $\geq 20\%$. Questi dati non costituiscono un buon caso di test in quanto è stato possibile costruire solo cinque modelli nominali e 8 dei 12 timestamp sono sostanzialmente identici, ma le anomalie che è stato possibile introdurre non permettono di costruire un caso di test più variegato.

Timestamp	A14896	A10063	A20995	A13957	A14502	Target
01	0.06	0.02	1.0			A
02	0.06	0.34				A
03	0.28	0.34				A
04	0.28	0.34				A
05	0.07	0.34				A
06	0.06	0.34				A
07	0.06	0.34				A
08	0.06	0.34				A
09	0.06	0.34				A
10	0.07	0.34				A
11	0.37	0.02				A
12	0.06	0.47				A

Tabella 6.7: Dati utilizzati per il test con il log di un mese. Sulle colonne è riportato il nome degli agenti diagnostici.

Algoritmo enumerativo

Nella Figura 6.3 è mostrato il costo ottenuto dalle 336 soluzioni sotto forma di albero completo trovate tramite enumerazione. Circa il 10% delle soluzioni ha costo pari a zero e circa il 30% ha costo pari ad un solo falso negativo, che equivale ad un errore su 12 timestamp significativi, ossia un'accuratezza del 91%. Questi risultati non sono inaspettati sia in virtù del fatto che la soluzione è poco vincolata sia poichè se il problema ha un numero di agenti diagnostici basso ed uno di essi esibisce livello di anomalia alto, riuscirà ad influenzare il risultato finale a prescindere dalla sua collocazione.

Le soluzioni con un costo pari a zero pongono almeno uno dei due agenti che non esibiscono mai probabilità diverse da zero al secondo livello dell'albero, si veda la Figura 6.4. Ciò suggerisce quindi che le anomalie causate da una probabilità pari a 0.34 o 0.37 non riescano a "sopravvivere" la negoziazione con altri quattro valori prossimi a zero, ma vi riescano con tre.

Simulated annealing

Si riportano in Figura 6.5 i costi delle soluzioni generate in ogni iterazione del simulated annealing tramite la neighbor function, allo scopo di compararlo con l'algoritmo enumerativo. Le soluzioni possono essere ripetute. Il

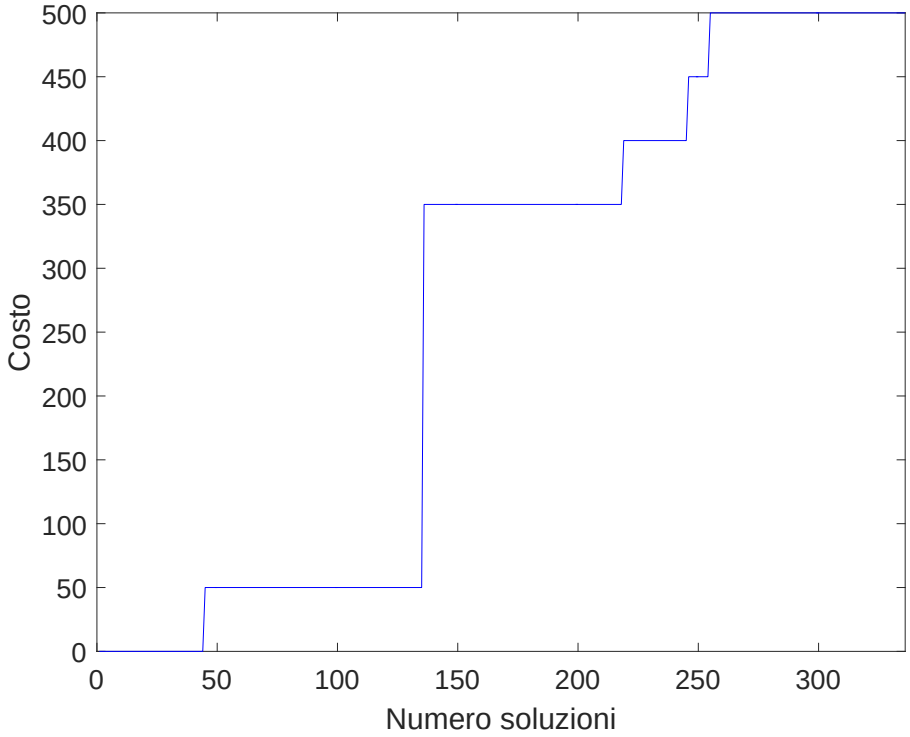

Figura 6.3: Soluzioni individuate tramite enumerazione per il test con il log di un mese, disposte in ordine di costo.

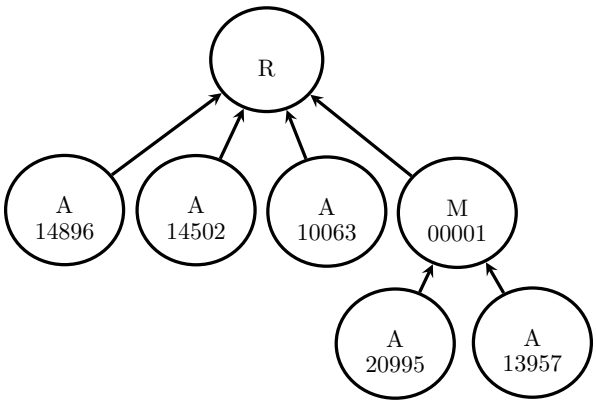

Figura 6.4: Soluzione ottima in cui l'agente A13957 è figlio di M0001.

numero di iterazioni è stato posto uguale al numero di soluzioni esistenti, per facilità di comparazione con l'algoritmo enumerativo. Si può osservare che la numerosità dei diversi gruppi di soluzioni equivalenti è differente, rispetto all'algoritmo enumerativo il simulated annealing ha generato più soluzioni dal costo molto basso e meno dal costo elevato. Ciò suggerisce che la neighbor function, usata per generare le soluzioni, riesce a ridurre leggermente la distanza media tra una qualsiasi soluzione ed una ottima, condizione essenziale per rendere il simulated annealing mediamente più efficiente rispetto all'enumerazione, in quanto potrà raggiungere la soluzione ottima con un minor numero di iterazioni.

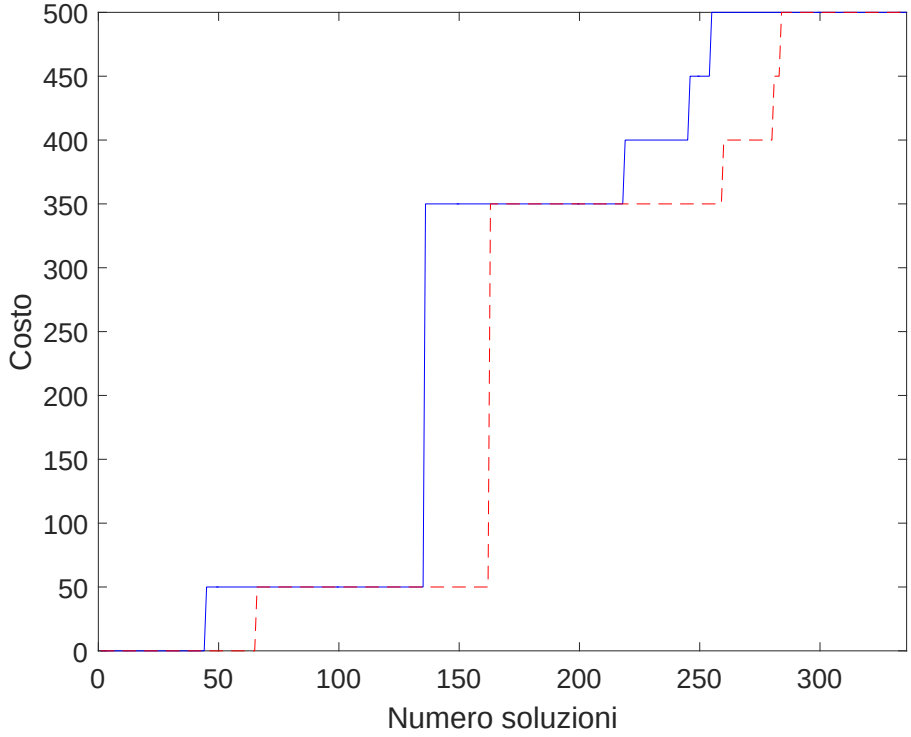

Figura 6.5: Soluzioni individuate per il test con il log di un mese, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. I valori sono medi su 10 esecuzioni.

Le soluzioni generate presentano sì alcuni duplicati ma in generale esibiscono strutture variegate, alcune di esse sono riportate in Figura 6.6. Alcune delle soluzioni ottime raggiunte tramite enumerazione non vengono generate, ma essendo la neighbor function stocastica si tratta di un comportamento accettabile.

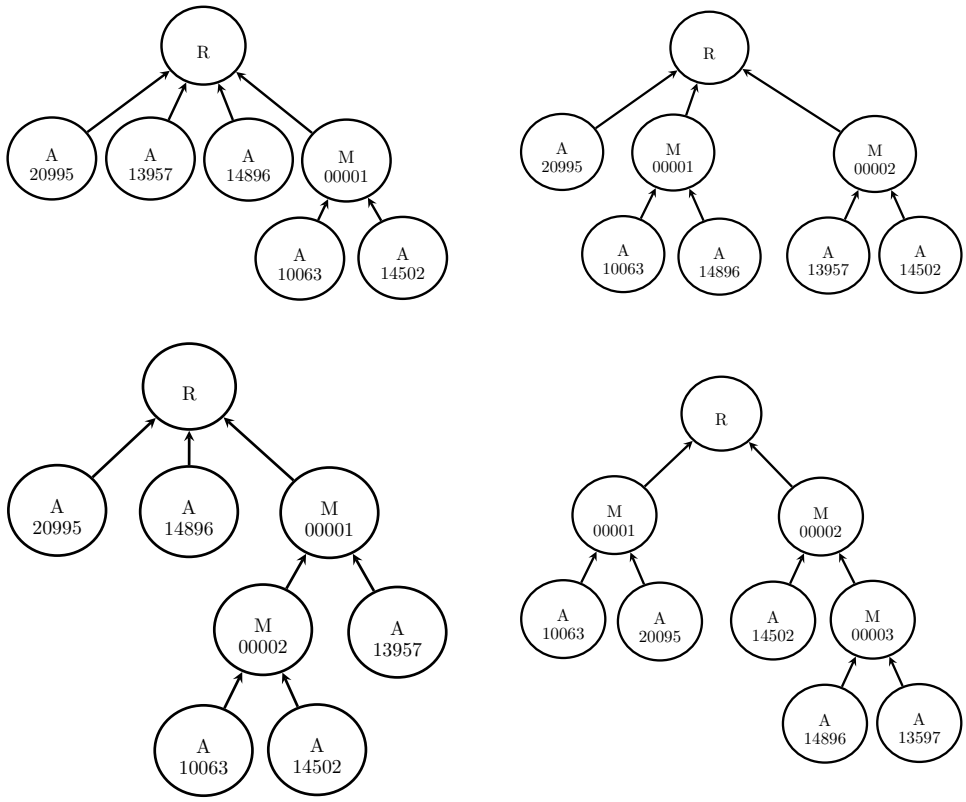

Figura 6.6: Alcune strutture visitate tramite simulated annealing.

Algoritmo greedy

L'algoritmo greedy riesce ad offrire delle performance intermedie fra quelle dell'algoritmo enumerativo e quelle del simulated annealing in termini di qualità della soluzione. Come si osserva in Figura 6.7 il numero di soluzioni ottime è drasticamente superiore rispetto ad entrambi gli altri algoritmi. Inoltre non vengono generate soluzioni dal costo superiore a 350 laddove invece gli altri due algoritmi raggiungono soluzioni dal costo pari a 500. Le soluzioni sono variegate a livello di struttura e allocazione degli agenti diagnostici.

Si può osservare un comportamento diverso dell'algoritmo a seconda del valore massimo scelto per la dimensione dell'insieme di agenti diagnostici che possono essere oggetto di un'azione. Un valore elevato tende a privilegiare la costruzione di soluzioni dalla bassa profondità. In questo caso la dimensione media dei sottoinsiemi di agenti associati alle azioni sarà superiore, portando a completare la struttura in un numero di mosse inferiore e quindi tendendo a introdurre meno mediatori. Per contro, una dimensione massima dell'insieme inferiore richiede più azioni per il completamento del-

la struttura e quindi comporta una maggiore probabilità di aggiungere un mediatore.

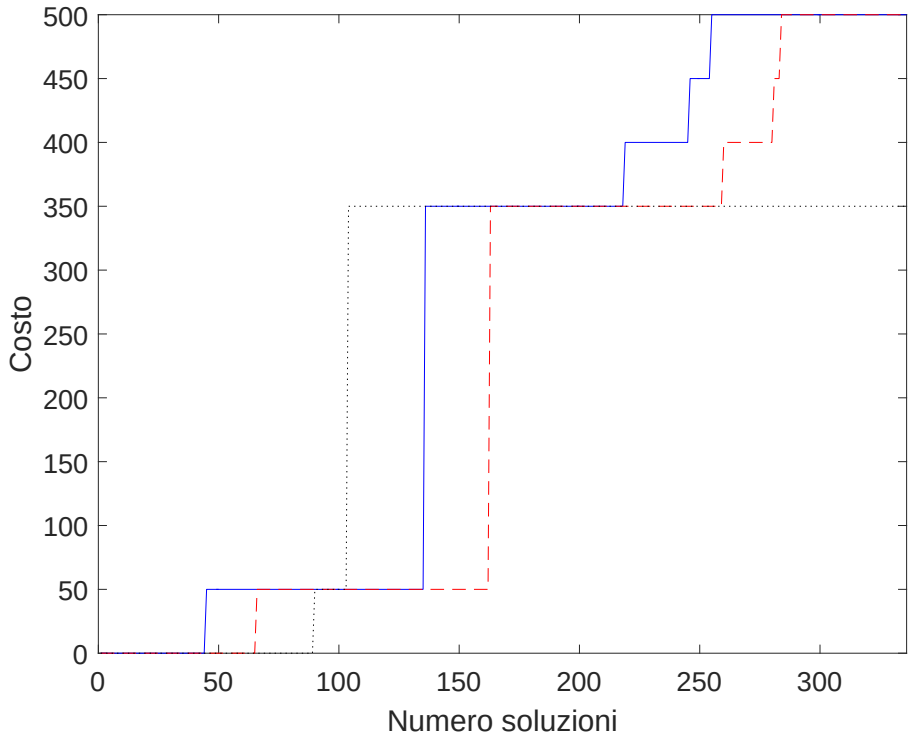

Figura 6.7: Soluzioni individuate per il test con il log di un mese, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni.

6.4.2 Forti anomalie, agenti indipendenti

In questo dataset si è considerato il caso in cui il sistema sia anomalo in tutti i timestamp significativi. Ogni agente causa almeno un'anomalia ma non ve ne sono mai due anomali contemporaneamente. I dati del test sono riportati in Tabella 6.8. In questo caso per ottenere la massima accuratezza sarebbe preferibile utilizzare la configurazione flat applicando alla radice la funzione di aggregazione massimo. Si vuole invece mostrare il comportamento della negoziazione cooperativa.

Timestamp	A1	A2	A3	A4	A5	Target
01	0.5					A
02	0.5					A
03		0.3				A
04		0.4				A
05		0.7				A
06		0.5				A
07			0.3			A
08			0.8			A
09			0.9			A
10				0.7		A
11				0.7		A
12					0.8	A
13					0.4	A
14					0.4	A
15					0.3	A
16	0.4					A
17	0.8					A
18	0.5					A
19		0.5				A
20		0.5				A

Tabella 6.8: Dati utilizzati per il test con forti anomalie ed agenti indipendenti.

Le soluzioni ottime equivalenti sono due, per tutte la confusion matrix è riportata in Tabella 6.9. I timestamp corrispondenti ai due falsi negativi cambiano a seconda della struttura della soluzione, come riportato nelle Figure 6.8 e 6.9.

TP: 18	FN: 2
FP: 0	TN: 0

Tabella 6.9: Confusion matrix delle due soluzioni ottime. Si possono osservare due falsi negativi.

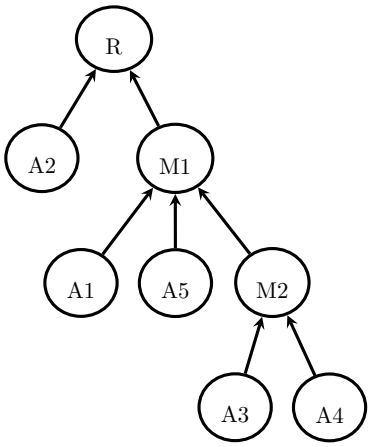

Figura 6.8: Albero di livello 3, FN ai timestamp 7 e 15.

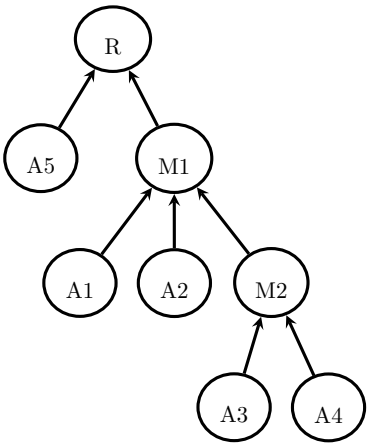

Figura 6.9: Albero di livello 3, FN ai timestamp 3 e 7.

I timestamp per i quali si hanno FN: 3, 7 e 15, corrispondono tutti a probabilità di anomalia di un agente pari a 0.30, ciò sta a significare che il sistema è poco sensibile a deboli anomalie di un singolo agente.

Timestamp	A1	A2	A3	A4	A5	Target
03		0.3				N
07			0.3			N
15					0.3	N

Tabella 6.10: Ai timestamp indicati in tabella è associato uno stato del sitema nominale.

Se si associa a tali timestamp uno stato del sistema nominale, come in Tabella 6.10, è possibile trovare una moltitudine di soluzioni ottime, 65 su 336. Le strutture sono le più variegate, dalla flat ad alberi aventi 1, 2 e 3 mediatori. La maggior parte delle strutture sono di livello 2 o 3, alcune di livello 4.

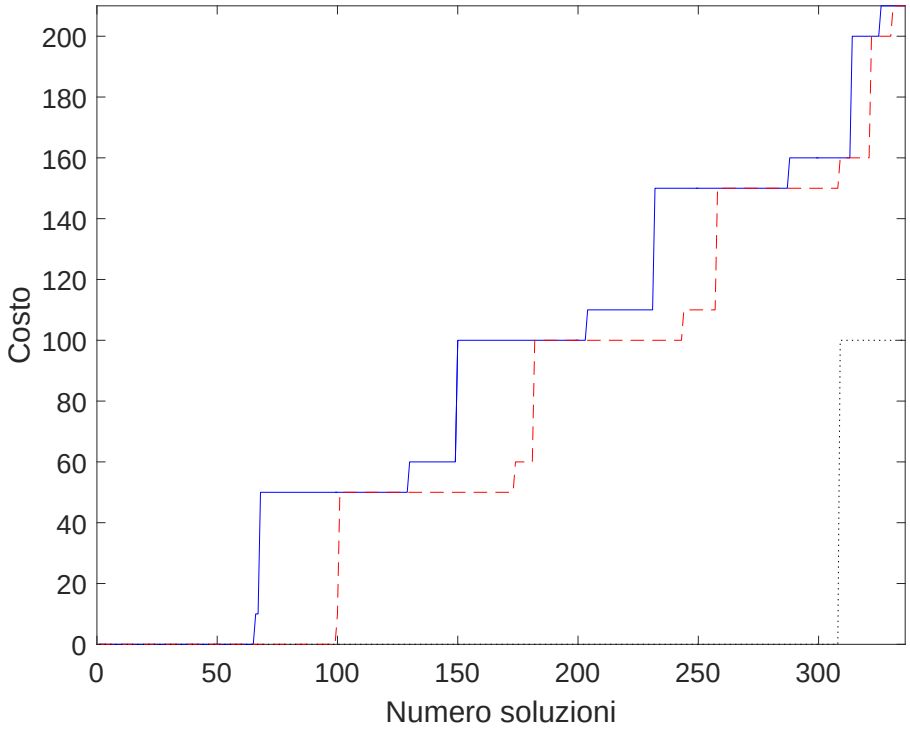

Figura 6.10: Soluzioni individuate per il test con forti anomalie ed agenti indipendenti, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni.

Come si può osservare in Figura 6.10 l'algoritmo greedy offre prestazioni superiori in termini di costo delle soluzioni. Tuttavia la struttura della soluzione è quasi sempre flat, con sporadiche occorrenze di alberi a due livelli. Il simulated annealing mostra un lieve miglioramento rispetto all'enumerazione su quasi tutti i costi.

Alberi di livello 2 Limitando la soluzione ad alberi di livello due, quindi con radice e massimo un livello di mediatori, il risultato rimane pressochè invariato. Le soluzioni equivalenti sono 52 su un totale di 76, tutte con tre FN.

Modificando il dataset come in Tabella 6.10, si eliminano gli unici timestamp causa di errore e le 52 soluzioni raggiungono costo 0.

Funzioni di aggregazione multiple Nella configurazione originale, in Tabella 6.8, rendere disponibile la funzione di aggregazione massimo consente di ottenere risultati migliori. Si veda l'Equazione 6.2. Le soluzioni sono limitate ad alberi di due livelli. Nuovamente il numero di soluzioni

ottime è alto, 171 su 502. Si osserva che quasi sempre in esse il massimo è applicato alla radice. La negoziazione cooperativa talvolta è applicata in un mediatore.

Modificando il dataset come in Tabella 6.10, il numero di soluzioni ottime rimane invariato. Cambia la funzione di aggregazione applicata alla radice, quasi sempre negoziazione cooperativa.

Il massimo è applicato solo in pochi casi, si vedano le Figure 6.11 e 6.12.

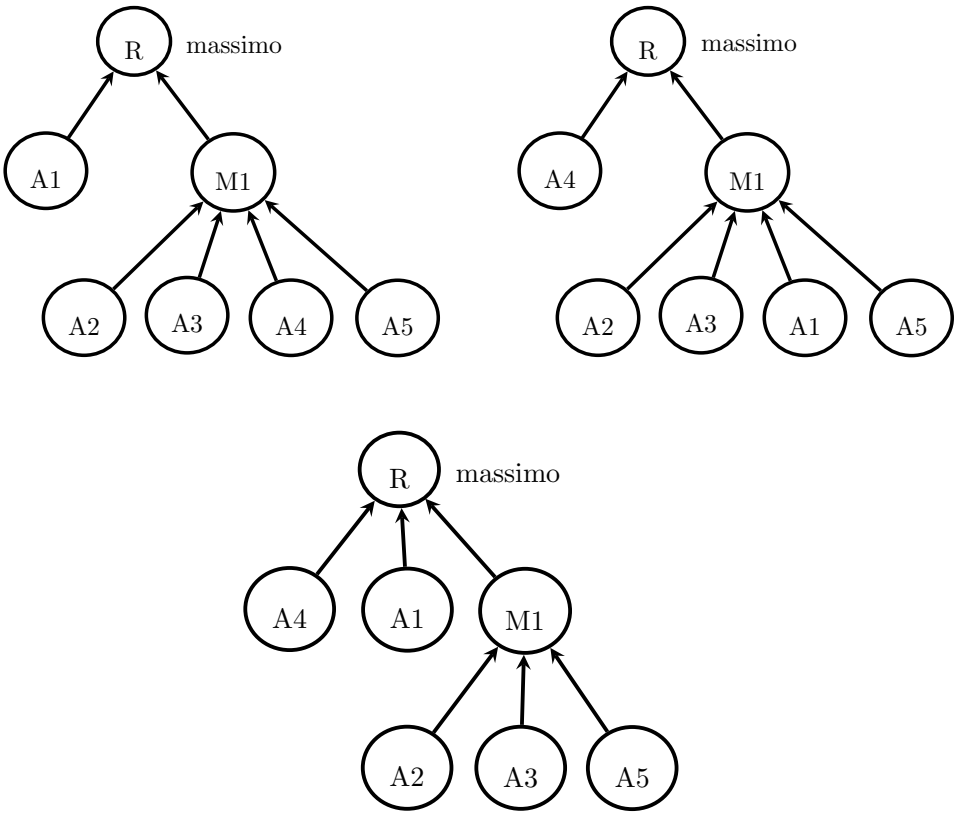

Figura 6.11: In queste soluzioni gli agenti A2, A3 ed A5 sono sempre collocati al secondo livello. Al primo invece si trovano le diverse combinazioni di agenti che esibiscono forti anomalie. Alla radice è applicata la funzione di aggregazione massimo, agli altri mediatori la negoziazione cooperativa.

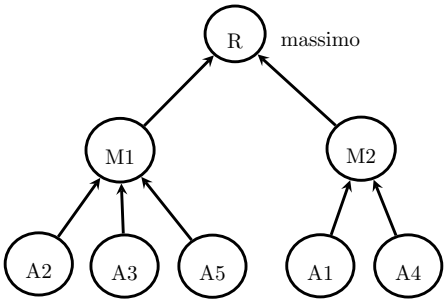

Figura 6.12: Soluzione ad albero perfettamente bilanciato. Alla radice è applicata la funzione di aggregazione massimo, agli altri mediatori la negoziazione cooperativa.

Si noti in particolare la soluzione in Figura 6.12. In essa si ha un albero perfettamente bilanciato, la radice ha come figli due mediatori. Il mediatore 1 ha per foglie tutti gli agenti in cui compare la probabilità 0.30, il mediatore 2 quelli che esibiscono forti anomalie. Entrambi i mediatori utilzzano la negoziazione cooperativa come funzione di aggregazione. Una variante di questa soluzione associa la funzione massimo al mediatore 2.

Da segnalare che questa struttura è resa possibile dal fatto che in presenza della debole anomalia di un agente, gli altri due esibiscano una probabilità pari a zero e quindi possano portare il risultato della negoziazione fra tre agenti sotto la soglia di anomalia. Qualora gli agenti non anomali esibissero probabilità maggiori di zero probabilmente ottenere un buon risultato sarebbe più difficile.

Osservando la Figura 6.13 si può notare come ampliare lo spazio delle soluzioni introducendo la possibilità di scegliere la funzione di aggregazione consenta di ottenere un numero maggiore di buone soluzioni.

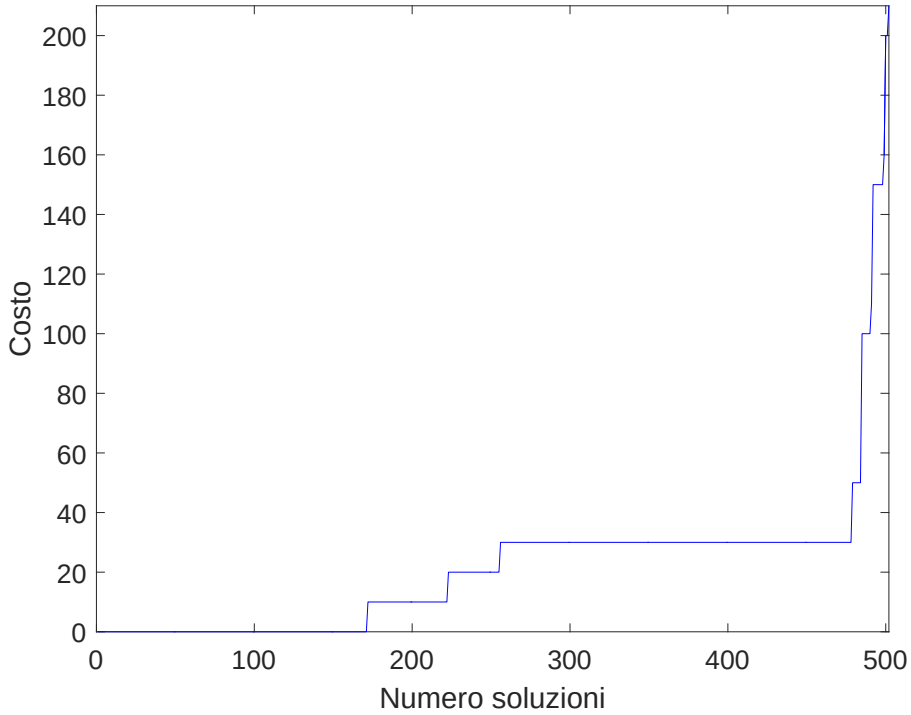

Figura 6.13: Soluzioni individuate tramite enumerazione per il test con forti anomalie ed agenti indipendenti, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa.

Funzione di aggregazione media Applicando a tutti gli agenti mediatori esclusivamente la funzione media, si ottengono otto soluzioni ottime equivalenti, tutte con un pessimo costo, 650, equivalente a 13 falsi negativi. Le combinazioni lineari sono riportate in Figura 6.14.

$A1\frac{1}{2}$	$+A2\frac{1}{8}$	$+A3\frac{1}{4}$	$+A4\frac{1}{16}$	$+A5\frac{1}{16}$
$A1\frac{1}{2}$	$+A2\frac{1}{12}$	$+A3\frac{1}{4}$	$+A4\frac{1}{12}$	$+A5\frac{1}{12}$
$A1\frac{1}{2}$	$+A2\frac{1}{16}$	$+A3\frac{1}{4}$	$+A4\frac{1}{8}$	$+A5\frac{1}{16}$
$A1\frac{1}{2}$	$+A2\frac{1}{16}$	$+A3\frac{1}{4}$	$+A4\frac{1}{16}$	$+A5\frac{1}{8}$
$A1\frac{1}{8}$	$+A2\frac{1}{2}$	$+A3\frac{1}{4}$	$+A4\frac{1}{16}$	$+A5\frac{1}{16}$
$A1\frac{1}{12}$	$+A2\frac{1}{2}$	$+A3\frac{1}{4}$	$+A4\frac{1}{12}$	$+A5\frac{1}{12}$
$A1\frac{1}{16}$	$+A2\frac{1}{2}$	$+A3\frac{1}{4}$	$+A4\frac{1}{8}$	$+A5\frac{1}{16}$
$A1\frac{1}{16}$	$+A2\frac{1}{2}$	$+A3\frac{1}{4}$	$+A4\frac{1}{16}$	$+A5\frac{1}{8}$

Figura 6.14: Le soluzioni con coefficiente massimo 12 sono di livello 3, le altre di livello 4. Tutte queste soluzioni hanno costo 650, con 13 FN la cui posizione cambia a seconda della specifica soluzione.

Modificando il dataset per eliminare le anomalie in presenza dei coefficienti 0.30, come riportato in Tabella 6.10, si ottengono le stesse soluzioni. Il costo è ridotto a 500 poichè scompaiono i tre falsi negativi, valutati ciascuno con costo 50 come descritto nella Sezione 6.1.

Per valutare l'impatto dei coefficienti sulla bontà della soluzione, consideriamo la specifica soluzione in Figura 6.15.

$A1\frac{1}{12}$	$+A2\frac{1}{2}$	$+A3\frac{1}{4}$	$+A4\frac{1}{12}$	$+A5\frac{1}{12}$

Figura 6.15: Soluzione di livello 3. Il suo costo è 500, i FN si trovano in corrispondenza dei timestamp: 1, 2, 10, 11, 12, 13, 14, 16, 17, 18.

I falsi negativi 1, 2, 16, 17, 18 sono associati ad A1, 10 e 11 ad A4 e 12, 13, 14 ad A5. Il coefficiente 12 impedisce di riconoscere persino le anomalie più marcate, con probabilità 0.80.

Sono correttamente riconosciute solo le anomalie di A2 ed A3, grazie al fatto che il loro coefficiente è basso.

Le altre soluzioni presentano le stesse dinamiche, solo associate ad agenti diversi. Applicando la funzione di aggregazione media si ottengono facilmente coefficienti di valore alto. La soluzione migliore sembra essere sostanzialmente determinata in base a quali agenti abbiano il minor numero di anomalie e possano quindi essere resi irrilevanti associandoli a coefficienti alti. La soluzione individuata quindi potrebbe cambiare molto a seconda del numero di anomalie presenti per un certo agente.

6.4.3 Gruppo correlato di agenti

In questo dataset si è considerato il caso in cui esistano tre agenti il cui andamento sia simile: A1, A2 ed A3. A3 tuttavia si colloca sopra la soglia di anomalia dello 0.20 anche in casi in cui il sistema non sia anomalo. I dati del test sono riportati in Tabella 6.11. Si vuole verificare se l'algoritmo riesce ad individuare una configurazione tale da non produrre dei falsi positivi.

Timestamp	A1	A2	A3	A4	A5	Target
01				0.60		A
02		0.20				N
03	0.10	0.15	0.35			N
04	0.10	0.12	0.35			N
05	0.10	0.10	0.35	0.70		A
06			0.20			N
07				0.50		A
08	0.10	0.09	0.25			N
09	0.20	0.17	0.35			N
10	0.25	0.18	0.35			A
11			0.20			N
12				0.40		A
13			0.10	0.50		A
14	0.10	0.10	0.25	0.60		A
15	0.10	0.08	0.25			N
16	0.05	0.07	0.20			N
17	0.30	0.28	0.45			A
18	0.35	0.30	0.55			A
19	0.25	0.10	0.35			A
20	0.10	0.12	0.25			N

Tabella 6.11: Dati utilizzati per il test con un gruppo di agenti correlati.

Applicando la ricerca su alberi non vincolati si individuano 30 soluzioni ottime, su 336. Nuovamente esse sono notevolmente variegate, con molteplici diverse allocazioni degli agenti diagnostici e scheletri.

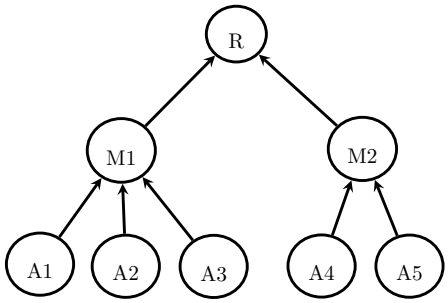

Figura 6.16: Soluzione ad albero perfettamente bilanciato.

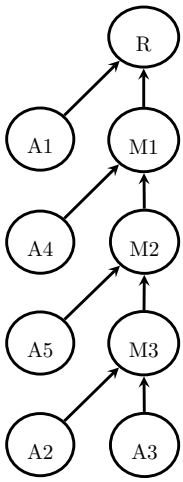

Figura 6.17: Configurazione ottima con dataset modificato come in Tabella 6.12.

Da segnalare la configurazione in Figura 6.16, nella quale gli agenti A1, A2 ed A3 sono raggruppati nello stesso mediatore.

Modificando le probabilità di anomalia in due timestamp come specificato in Tabella 6.12, e rendendo A3 lievemente più anomalo, il risultato cambia drasticamente.

Timestamp	A1	A2	A3	A4	A5	Target
03	0.10	0.15	*0.40*			N
09	0.20	0.17	*0.40*			N

Tabella 6.12: Ai timestamp indicati in tabella è associata un'anomalia di A3 più elevata rispetto a quanto riportato in Tabella 6.11.

Esiste solo una soluzione ottima, con 4 livelli, riportata in Figura 6.17. Altre 46 soluzioni presentano uno o due falsi positivi in corrispondenza dei timestamp modificati. E' quindi difficile evitare falsi positivi per valori di anomalia dei singoli agenti diagnostici superiori a 0.35, in quanto è necessario "spingere" l'agente diagnostico molto in profondità.

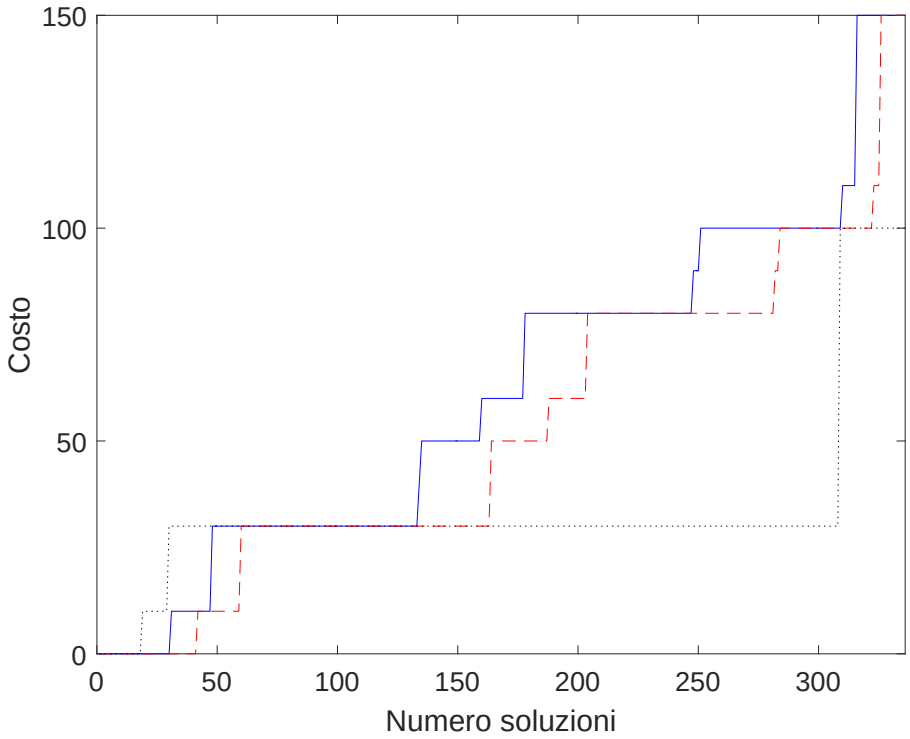

Figura 6.18: Soluzioni individuate per il test con un gruppo di agenti correlati, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori solo medi su 10 esecuzioni.

Osservando la Figura 6.18 si può notare che l'algoritmo greedy mostra un andamento intermedio. In questo caso il numero massimo di agenti diagnostici coinvolti in un'azione, chiamato $maxSetSize$ nella Sezione 6.2.3, è fissato a 2, le soluzioni raggiunte sono tutte di livello due e presentano uno o due mediatori. Se $maxSetSize$ è posto a 4, l'algoritmo non riesce a raggiungere l'ottimo globale ma genera stabilmente soluzioni di costo pari a 30. Tali soluzioni sono o flat o alberi a due livelli con un solo agente mediatore avente due figli.

Anche in questo caso il simulated annealing mostra un andamento leggermente migliore dell'enumerazione, ma ha soluzioni ripetute.

Alberi di livello 2 Imponendo la limitazione sui livelli dell'albero, le soluzioni ottime per il dataset riportato in Tabella 6.11 scendono a 4. Si segnala che gli agenti A2 ed A3 compaiono sempre aggregati come figli di un mediatore, eventualmente con anche A1.

Nel caso della configurazione modificata di Tabella 6.12, nessuna soluzione riesce ad evitare il falso positivo ai timestamp 3 e 9. Il costo della soluzione ottima è pertanto peggiore di quello ottenibile con alberi non vincolati, in accordo all'Equazione 6.2.

Funzioni di aggregazione multiple Introdurre la funzione massimo come possibile funzione di aggregazione in questo caso è di scarsa utilità, in quanto è causa di falsi positivi. Se l'albero non ha una limitazione nel numero di livelli si ottengono, tra le altre, le soluzioni in Figura 6.19, 6.20, 6.21 e 6.22.

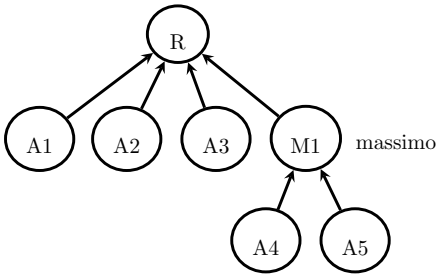

Figura 6.19: Massimo applicato al livello 1, il gruppo di agenti correlati nel dataset è direttamente figlio della radice.

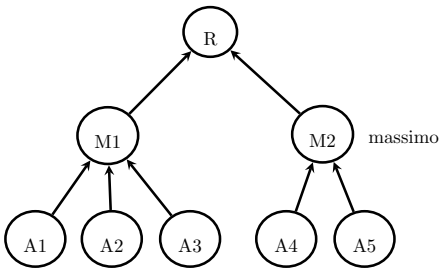

Figura 6.20: Configurazione simile a quella di Figura 6.19, il gruppo di agenti correlati è raggruppato nel mediatore 1.

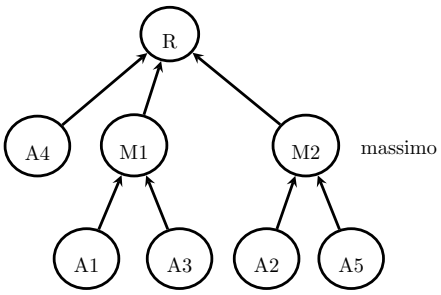

Figura 6.21: Il gruppo di agenti correlati in questo caso è separato. Poichè A5 non esibisce nessuna anomalia, le probabilità di anomalia esisbite da M2 sono uquali a quelle di A2.

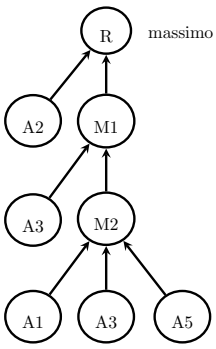

Figura 6.22: Talvolta la funzione massimo è applicata alla radice, ma nel qual caso A3 si trova sempre a livello 3 o inferiore.

Introducendo la limitazione a 2 livelli, l'unico caso in cui compare il massimo è come funzione per aggregare A4 ed A5. Viene riproposta la soluzione in Figura 6.20, quindi nuovamente i tre agenti con andamento

simile sono raggruppati. Dalla Figura 6.23 si può osservare come le soluzioni ottime in questo caso siano molto rare.

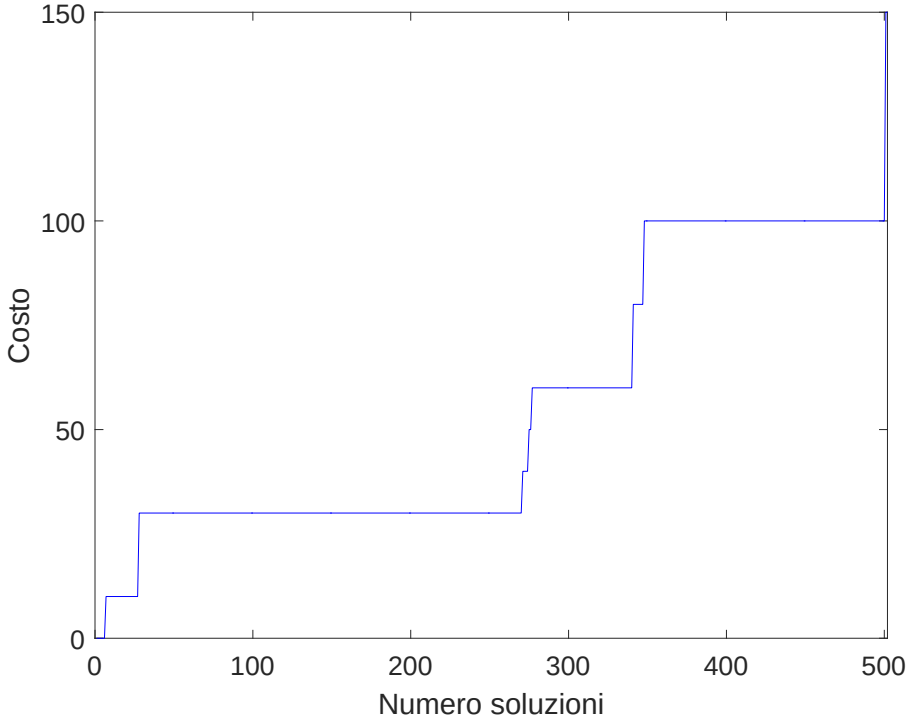

Figura 6.23: Soluzioni individuate tramite enumerazione per il test con un gruppo di agenti correlati, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa.

Funzione di aggregazione media Utilizzare esclusivamente la funzione di aggregazione media anche in questo caso porta ad ottenere soluzioni di minore qualità, il costo minimo è 150 a causa di tre falsi negativi. Esiste una sola soluzione ottima, di livello 4.

$$\boxed{A1\frac{1}{8} \quad +A2\frac{1}{16} \quad +A3\frac{1}{4} \quad +A4\frac{1}{2} \quad +A5\frac{1}{16}}$$

Figura 6.24: Soluzione di livello 4. Il suo costo è 150 ed i FN sono relativi ai timestamp: 10, 17, 19. Una sola anomalia per il gruppo A1, A2, A3 è riconosciuta, quella più marcata nel timestamp 18.

Applicando la variante di Tabella 6.12 il risultato non cambia. Per essere riconosciuta correttamente l'anomalia dovrebbe essere molto più marcata.

Limitando il coefficiente massimo a 10, impedendo quindi che venga ignorato l'agente A2, la soluzione individuata è di livello 2. Il costo minimo è 200, peggiore a causa di quattro falsi negativi, in accordo con l'Equazione 6.4.

$$\boxed{A1\frac{1}{8} \quad +A2\frac{1}{8} \quad +A3\frac{1}{8} \quad +A4\frac{1}{2} \quad +A5\frac{1}{8}}$$

Figura 6.25: Soluzione di livello 2. Il suo costo è 200 ed i FN sono relativi ai timestamp: 10, 17, 18, 19. Si tratta dell'unica soluzione ottima. Nessuna anomalia per i gruppo A1, A2, A3 è riconosciuta. Nel timestamp 18 la probabilità di anomalia associata al sistema è 0.15.

Nuovamente applicando la variante del dataset in Tabella 6.12 il risultato non cambia.

6.4.4 Soglie di anomalia differenziate

In questo dataset si vuole rappresentare l'esistenza di due sottosistemi di agenti diagnostici indipendenti, ciascuno composto da agenti diagnostici con andamento correlato. I sottosistemi presentano anomalie in timestamp indipendenti. I due sottosistemi rendono il sistema globale anomalo con soglie di anomalia diverse. Il primo, composto da A1, A2, A3 ha soglia 0.30, il secondo, composto da A4 ed A5 ha soglia 0.50. I dati del test sono riportati in Tabella 6.13

Timestamp	A1	A2	A3	A4	A5	Target
01	0.15	0.20	0.25			N
02	0.10	0.15	0.20			N
03	0.25	0.28	0.32			N
04	0.30	0.32	0.35			A
05	0.25	0.35	0.40			A
06	0.35	0.40	0.50			A
07	0.45	0.55	0.60			A
08	0.10	0.09	0.25			N
09	0.20	0.17	0.35			N
10	0.25	0.18	0.35			N
11				0.30	0.20	N
12				0.40	0.30	N
13				0.50	0.40	N
14				0.60	0.45	A
15				0.65	0.55	A
16				0.70	0.40	A
17				0.80	0.70	A
18	0.05	0.10	0.15	0.20	0.30	N
19	0.15	0.20	0.25	0.30	0.40	N
20	0.20	0.25	0.30	0.40	0.50	N

Tabella 6.13: Dati utilizzati per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata.

Le soluzioni ottenute tramite algoritmo enumerativo hanno strutture estremamente variegate. Esistono 30 soluzioni ottime su 336, con costo pari a 50 a causa di cinque falsi positivi.

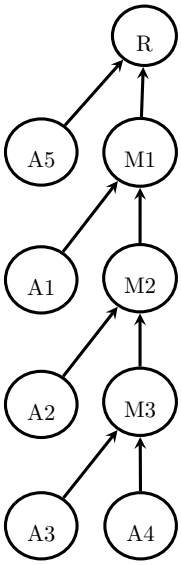

Figura 6.26: Soluzione ad albero perfettamente sbilanciato. Il suo costo è 50 a causa di FP nei timestamp: 12, 13, 18, 19, 20.

Per esempio, la soluzione di Figura 6.26 non riesce a riconoscere la soglia di anomalia più alta per il secondo sistema, producendo un FP nei timestamp 12 e 13. Questo problema è in parte causa dei FP nei timestamp 19 e 20, che esibiscono probabilità di anomalia ancora superiori. Quello relativo al timestamp 18 invece è legato al fatto che, esibendo tutti gli agenti una probabilità di anomalia maggiore di zero, le varie componenti si sommano impedendo di compensare il valore 0.30 di A5.

Per mitigare questo fenomeno, è sufficiente introdurre nel dataset un ulteriore agente, A6, la cui probabilità di anomalia sia sempre pari a zero. Risulta una sola soluzione ottima di costo 10, quella di Figura 6.27, che presenta un FP relativo al timestamp 20 in cui, come descritto sopra, la probabilità di anomalia media è troppo alta per poter essere ridotta. Tuttavia, il sistema è ora in grado di riconoscere correttamente le soglie di anomalia differenziate. La struttura dell'albero pone ad una maggiore profondità A4 ed A5. L'agente nuovo è posto direttamente come figlio della radice.

Sorge spontaneo chiedersi cosa succederebbe aggiungendo un ulteriore agente nullo, A7. In questo caso si riescono ad ottenere 20 soluzioni ottime su 132 294. La loro struttura è sempre ad albero perfettamente sbilanciato. Se ne osservi una in Figura 6.28. Essendo l'albero di livello 7 è evidente come la struttura sia poco rappresentativa dell'effettivo legame tra le variabili e quindi possa essere molto soggetta ad overfitting.

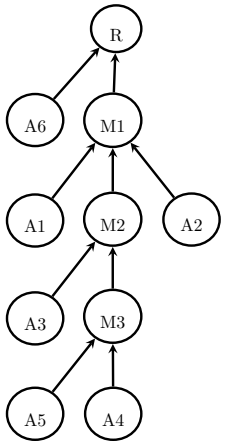

Figura 6.27: Soluzione ottima ottenuta aggiungendo al dataset l'agente A6. Il suo costo è pari a 10 a causa di un FP nel timestamp 20.

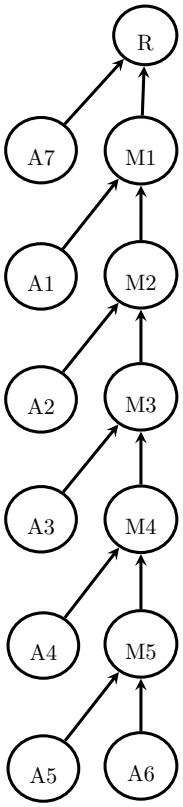

Figura 6.28: Soluzione ottima ottenuta aggiungendo al dataset gli agenti A6 ed A7. Il suo costo è pari a 0.

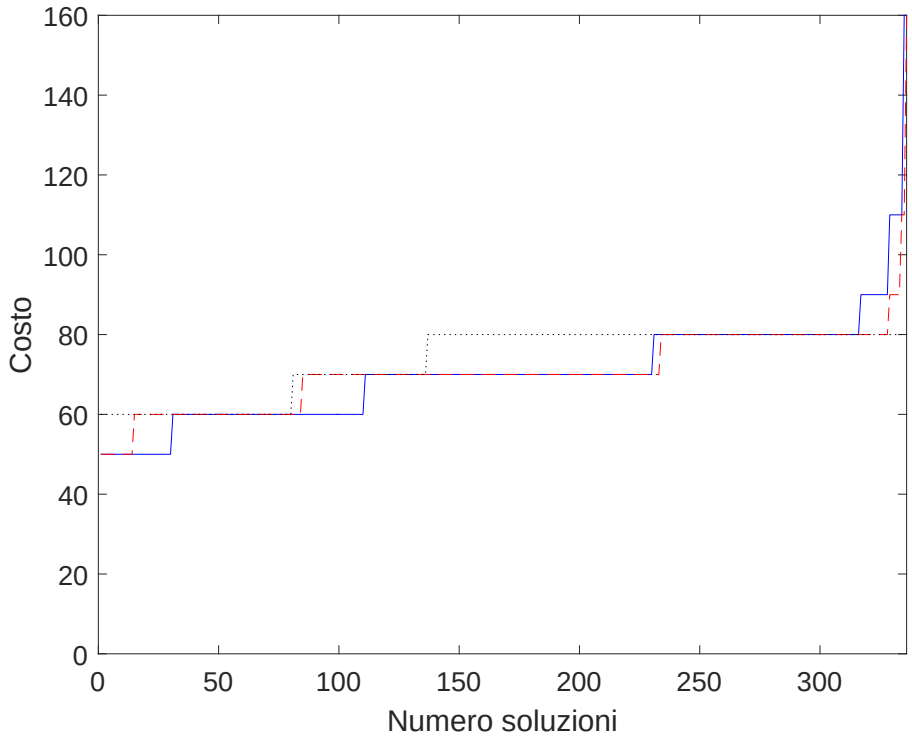

Figura 6.29: Soluzioni individuate per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata, disposte in ordine di costo. In blu (linea continua), le soluzioni individuate tramite enumerazione, in rosso (linea tratteggiata) tramite simulated annealing. In nero (linea a puntini) le soluzioni individuate tramite algoritmo greedy. I valori sono medi su 10 esecuzioni.

A differenza degli altri casi, nella Figura 6.29 si può osservare come il simulated annealing abbia prestazioni leggermente peggiori rispetto all'enumerazione. Questo è legato a fatto che costi bassi si riescono ad ottenere per soluzioni ad alta profondità o alberi sbilanciati, che l'attuale implementazione della neighbor function usata nel simulated annealing esplora con minore probabilità. Un simile risultato si ottiene anche per l'algoritmo greedy, che non riesce a raggiungere l'ottimo locale e raggiunge con bassa probabilità altre soluzioni buone.

Alberi di livello 2 La ricerca su alberi di livello massimo 2 consente di ottenere una sola soluzione ottima di costo 50, riportata in Figura 6.30. La struttura è differente ma il costo è identico a quello precedente. Si noti che assieme ad A1, A2 e A3 è posto anche A4. Esso è, rispetto ad A5, quello che esibisce probabilità di anomalia più alta nei timestamp che originerebbero

falsi negativi. Posizionarlo ad un livello inferiore consente di classificare correttamente il primo tra i timestamp 11, 12, 13.

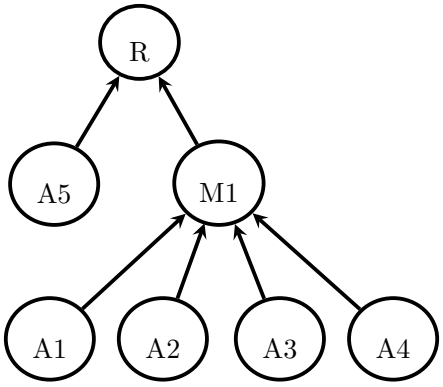

Figura 6.30: Soluzione con costo 50 a causa di FP nei timestamp: 12, 13, 18, 19, 20.

Introducendo l'agente nullo A6, la soluzione migliora, evitando i FP meno marcati, si veda la Figura 6.31.

Introducendo l'agente nullo A7, viene riconosciuto correttamente un altro FP, ma la struttura della soluzione è nuovamente lontana da quella del sistema reale, si veda Figura 6.32.

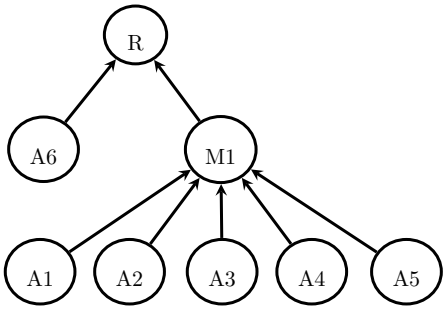

Figura 6.31: Soluzione ottima ottenuta aggiungendo al dataset l'agente A6. Il suo costo è 30 a causa di FP nei timestamp: 13, 19, 20.

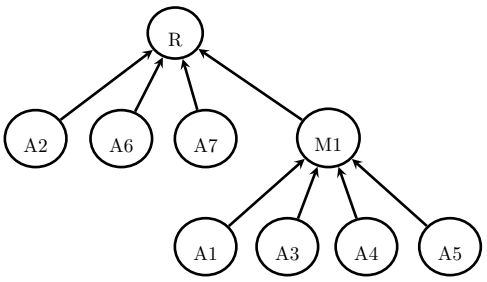

Figura 6.32: Soluzione ottima ottenuta aggiungendo al dataset gli agenti A6 ed A7. Il suo costo è 20 a causa di FP nei timestamp: 13, 20.

Funzioni di aggregazione multiple Introdurre la possibilità di applicare la funzione massimo non porta ad alcun miglioramento del risultato, in quanto non permette di risolvere i FP. I costi delle soluzioni si possono osservare in Figura 6.33. Si osserva inoltre che la funzione massimo non compare in alcuna soluzione ottima, nè con il dataset originario, nè introducendo gli agenti nulli A6 ed A7.

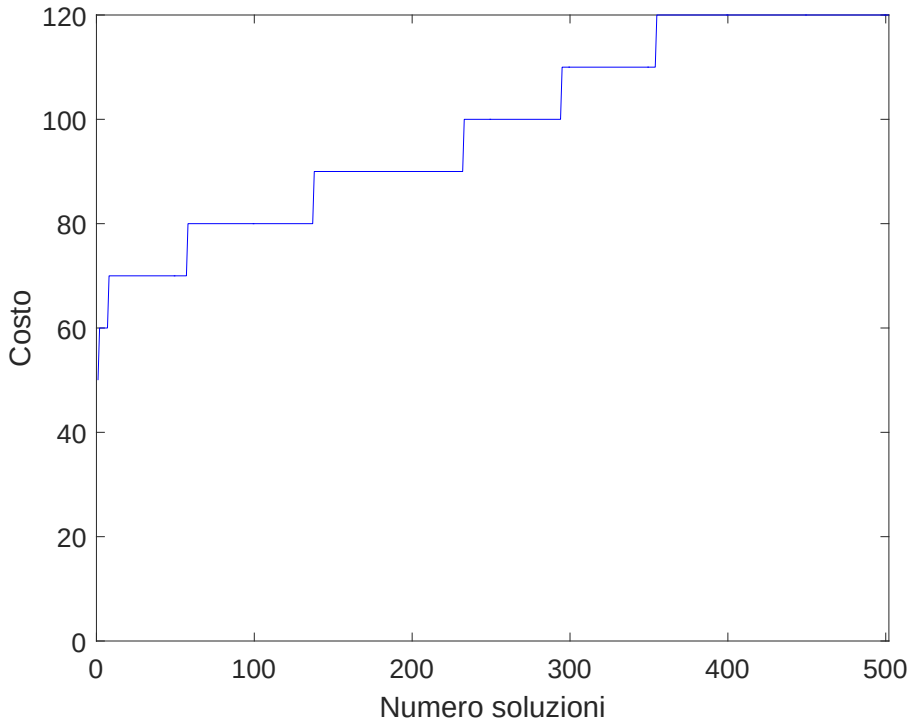

Figura 6.33: Soluzioni individuate tramite enumerazione per il test con sottosistemi di agenti diagnostici dalla soglia di anomalia differenziata, disposte in ordine di costo. Può essere attribuita agli agenti mediatori la funzione di aggregazione massimo o negoziazione cooperativa.

Funzione di aggregazione media La capacità della funzione media di abbattere velocemente i valori di anomalia in questo caso consente di ottenere dei buoni risultati. Le strutture di agenti individuate dall'algoritmo enumerativo riconoscono correttamente le soglie di anomalia differenziate. Gli unici falsi positivi sono presenti nei timestamp 19 e 20, laddove tutti gli agenti diagnostici mostrano lievi anomalie. Le soluzioni ottime sono riportate in Figura 6.34.

$A1\frac{1}{4}$	$+A2\frac{1}{8}$	$+A3\frac{1}{4}$	$+A4\frac{1}{4}$	$+A5\frac{1}{8}$
$A1\frac{1}{8}$	$+A2\frac{1}{4}$	$+A3\frac{1}{4}$	$+A4\frac{1}{4}$	$+A5\frac{1}{8}$
$A1\frac{1}{16}$	$+A2\frac{1}{2}$	$+A3\frac{1}{16}$	$+A4\frac{1}{4}$	$+A5\frac{1}{8}$

Figura 6.34: Soluzioni ottime ottenute applicando la funzione di aggregazione media, il loro costo è 200. Le soluzioni con coefficiente massimo 8 sono di livello 2, le altre di livello 4.

Introdurre gli agenti nulli A6 ed A7 peggiora il risultato in modo significativo. Nel caso di albero con coefficienti limitati a 10, il costo minimo per il dataset contenente gli agenti originali ed A6 è di 120.

Funzioni di aggregazione multiple media e negoziazione cooperativa In virtù dei risultati ottenuti con la media, si è provato a combinarla con la negoziazione cooperativa in una ricerca con due funzioni di aggregazione. Si ottengono 6 soluzioni ottime su 502, con due soli falsi positivi e di livello 2. Si può osservare che gli agenti diagnostici relativi ai due sottosistemi sono stati mescolati, a titolo di esempio una soluzione è riportata in Figura 6.35.

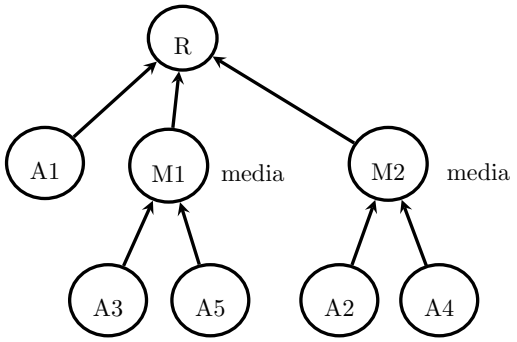

Figura 6.35: Il costo di questa soluzione è 20 a causa di FP nei timestamp: 19, 20. La funzione di aggregazione media è applicata ad M1 e M2, R applica negoziazione cooperativa.

6.5 Conclusioni

In base a quanto descritto nelle sezioni precedenti, è possibile trarre alcune conclusioni sotto forma di linee guida per scegliere la tipologia di ap-

proccio adatta ad affrontare l'individuazione delle anomalie sulla base dei comportamenti che le variabili mostrano nel dataset.

Se il dataset contiene solo forti anomalie, senza apparenti legami o correlazioni tra gli agenti, si possono ottenere ottimi risultati agevolmente, anche con strutture contenenti pochi agenti mediatori o solo la radice. La sensibilità a deboli anomalie si riduce all'aumentare del numero di agenti diagnostici. Affiancare la funzione di aggregazione massimo alla negoziazione cooperativa può mitigare questo effetto. L'algoritmo greedy si dimostra il più efficiente a produrre soluzioni dal basso costo, ma esse tendono ad essere poco rappresentative della struttura intuitiva del sistema. Il simulated anneling offre un miglior compromesso tra costo e significatività della soluzione.

Qualora un gruppo di agenti diagnostici mostri un andamento correlato, alberi di livello massimo due riescono a rappresentare l'effettiva struttura del sistema. La resistenza a falsi positivi è limitata, essendo legata alla presenza di altri agenti che nello stesso timestamp mostrino probabilità uguale a zero. Algoritmo enumerativo e simulated annealing si dimostrano pressochè equivalenti mentre l'algoritmo greedy per produrre buoni risultati richiede i suoi parametri siano diversi da quelli usati negli altri test.

Nel caso in cui il dataset mostri la presenza di sottosistemi la cui soglia di anomalia è differenziata, la negoziazione cooperativa non riesce a produrre buoni risultati. Utilizzare più funzioni di aggregazione consente di ottenere risultati migliori. La media riduce il livello di anomalia dell'aggregazione, quindi è utile se applicata al sistema che mostra soglia più elevata. I risultati migliori si ottengono combinando media e negoziazione cooperativa. Non è di nessuna utilità la funzione massimo. L'algoritmo enumerativo e il simulated annealing nuovamente producono risultati simili, l'algoritmo greedy invece ha prestazioni inferiori.

Se gruppi di agenti mostrano collettivamente probabilità di anomalia diversa da zero, è in ogni caso difficile evitare che il sistema sia classificato come anomalo. L'uso della funzione media mitiga questo fenomeno ma al costo di una peggiore accuratezza in altri casi.

Soluzioni dall'elevata accuratezza non sempre riescono a rappresentare l'effettiva struttura del sistema.

La difficoltà nel gestire correttamente i falsi positivi costituisce una limitazione significativa per l'attuale sistema.

Capitolo 7

Conclusioni e sviluppi futuri

In questa tesi sono stati studiati diversi metodi per la ricerca della struttura ottima di un sistema multiagente che svolga anomaly detection in applicazioni spaziali senza l'ausilio di domain knowledge e con limitati dati a disposizione. Abbiamo cercato di sviluppare metodi che consentano di trattare problemi con un numero di variabili sufficiente per un caso reale, ossia il caso del satellite Cryosat-2.

Lo spazio di ricerca per identificare la struttura ottima del sistema multiagente è stato modellato come uno spazio di grafi, con una restrizione agli alberi, e si sono presentate tre funzioni di aggregazione, descrivendone le caratteristiche e l'impatto sullo spazio delle strutture possibili. La funzione *massimo* può essere efficacemente applicabile a delle aggregazioni intermedie, riduce lo spazio delle soluzioni, ma non può essere utilizzata da sola per tutte le aggregazioni di un sistema multiagente in quanto la struttura perderebbe di senso. La funzione *negoziazione cooperativa* non permette di restringere lo spazio delle soluzioni ma offre un risultato che considera le diverse probabilità di anomalia calcolate dagli agenti diagnostici. La funzione *media*, infine, mostra delle caratteristiche intermedie tra il massimo e la negoziazione cooperativa. La qualità del risultato è migliore rispetto a quella della funzione massimo ed il valore ottenuto mantiene una dipendenza dalla struttura delle aggregazioni e quindi dalla specifica soluzione.

L'algoritmo *enumerativo* è l'unico proposto che operi una ricerca di tipo globale, enumerando esplicitamente tutte le soluzioni. In questo caso si è osservato che la dimensione dello spazio mostra una crescita esponenziale rispetto al numero di variabili, e quindi l'applicazione dell'algoritmo è limitata a problemi di piccola dimensione. Relativamente gli algoritmi di ricerca locale abbiamo proposto il *simulated annealing*, in quanto efficiente per svolgere ricerche su spazi di stati discreti molto vasti. Si è definita una neighbor function che esplora lo spazio delle soluzioni in modo completo e

bilanciato. Un ulteriore algoritmo di ricerca locale che abbiamo proposto è quello *greedy*, che costruisce la soluzione iterativamente scegliendo tra un insieme di possibili azioni che aggiungono progressivamente a una struttura parziale tutti gli agenti diagnostici. Questo algoritmo è caratterizzato da una velocità di generazione delle soluzioni elevata ed una limitata complessità computazionale ma, a causa di come sono definite le azioni, non esplora l'intero spazio delle strutture.

Il sistema modulare che abbiamo implementato è facilmente configurabile ed estendibile sia relativamente agli algoritmi sia per i metodi di valutazione delle soluzioni. Gli algoritmi sono stati implementati in modo da offrire adeguata velocità ed equilibrato sfruttamento delle risorse computazionali disponibili.

Tutti gli algoritmi sono stati testati rispetto a quattro diversi casi di test ciascuno focalizzato su un particolare comportamento delle variabili. Il primo caso è costruito sfruttando i dati contenuti in un log reale del satellite Cryosat-2 nel quale sono stati introdotti degli artefatti per simulare delle anomalie. Gli altri casi di test sono invece sintetici.

I risultati sperimentali evidenziano che l'approccio proposto in questa tesi è promettente. Tutti e tre gli algoritmi consentono di raggiungere buoni risultati, con il simulated annealing tendenzialmente migliore rispetto all'enumerazione. Relativamente a specifici comportamenti delle variabili, si osserva che individuare correttamente anomalie molto marcate esibite da un singolo agente sia relativamente facile. Qualora un insieme di agenti diagnostici evidenzi un andamento correlato nelle anomalie mostrate, l'algoritmo enumerativo riesce a proporre una soluzione che rappresenti correttamente la struttura del sistema che idealmente avrebbe generato quei valori. Si evidenzia una limitazione del metodo proposto nella sua scarsa robustezza nei confronti di agenti che esibiscano stabilmente probabilità di anomalia elevata, a fronte di un andamento delle variabili nominale. In questo caso si osserva che espandere lo spazio di ricerca, includendovi anche la scelta di quali funzioni di aggregazione utilizzare per ciascun insieme di agenti diagnostici, permette di migliorare il risultato. Si è osservato infine che alberi con molti livelli offrono scarse prestazioni in quanto le progressive aggregazioni degradano l'informazione proveniente dagli agenti più in profondità.

Possono essere individuate diverse direzioni di sviluppo future. Innanzitutto gli strumenti utilizzati per creare i modelli nominali delle variabili potrebbero essere ampliati per rendere modellabili fenomeni più complessi, ad esempio le vibrazioni in rotori. Inoltre, potrebbe essere utile svolgere un'analisi dell'andamento dei valori di certe variabili su lassi di tempo lunghi allo scopo di individuare trend poco visibili in brevi intervalli di tempo. Un esempio di trend può essere la riduzione nell'energia immagazzinabile

dalle batterie, che potrebbe decrescere più rapidamente del previsto e l'anomalia essere riconosciuta ben prima che la capacità delle batterie esca dalla regione nominale di funzionamento.

Similmente, anche lo studio di nuove funzioni di aggregazione potrebbe permettere di conseguire risultati migliori nelle due circostanze in cui l'attuale sistema evidenzia le maggiori limitazioni. La prima di esse è la scarsa robustezza ai falsi positivi, la seconda è la degradazione dell'informazione in problemi con numero di agenti diagnostici elevato o alberi particolarmente profondi.

Importante direzione di miglioramento può essere l'introduzione di meccanismi per aggiungere ad una struttura già completa ulteriori agenti diagnostici per i quali sia stato possibile addestrare un modello nominale. Poichè il contesto in cui si inserisce questa tesi prevede scarsa disponibilità di dati, è realistico ritenere che l'accumularsi di nuovi dati nel tempo permetta di disporre di sufficienti informazioni per addestrare nuovi modelli. Inoltre, l'esperienza stessa acquisita dal personale potrebbe essere efficacemente integrata nella struttura del sistema multiagente. Infine, si ravvisa che l'attuale impossibilità di verificare la concordanza tra i valori assunti da diverse variabili costituisca una significativa limitazione, alla luce del fatto che fra variabili legate ad aspetti diversi dello stesso sistema o processo sia molto probabile esistano legami di correlazione dalla cui analisi si possono scoprire eventuali anomalie. Questo tipo di legame potrebbe essere monitorato tramite ulteriori agenti diagnostici oppure attraverso l'introduzione negli agenti diagnostici stessi di modelli più complessi, le cui predizione siano combinate in una probabilità di anomalia complessiva.

Appendice A

Documentazione della programmazione

In questo appendice si descrive più nel dettaglio l'implementazione del modulo di ricerca della soluzione ottima, fornendo una descrizione delle classi e diagrammi di sequenza dei metodi principali. Si descrivono inoltre i file di configurazione.

A.1 Inizializzazione e avvio ricerca

L'oggetto *OptimalStructureManager* si occupa dell'acquisizione delle informazioni legate alla configurazione del modulo, si veda Figura A.1, e di avviare l'algoritmo di ricerca, si veda Figura A.2. In sintesi svolge quindi le seguenti operazioni:

- Legge il file *SchemeMap* contenente i modelli costruiti durante la fase di preprocessing.
- Legge il file di configurazione *OptimalStructure.properties* contenente l'algoritmo scelto, la struttura che si desidera la soluzione abbia, la funzione di aggregazione per gli agenti mediatori, la neighbor function per il simulated annealing ecc... Per una descrizione dettagliata si rimanda alla Sezione A.9.
- Raggruppa gli agenti associati a modelli multivariati e carica il file di configurazione in cui sono specificati i subset di agenti che si vuole siano aggregati separatamente. La sintassi del file è riportata nella Sezione A.8.
- Ottiene da *StructureBuilder* un'istanza che implementa l'algoritmo di ricerca specificato.
- Avvia la ricerca della struttura ottima su ogni subset di agenti.

- Combina le strutture tra loro in un'unica struttura globale.

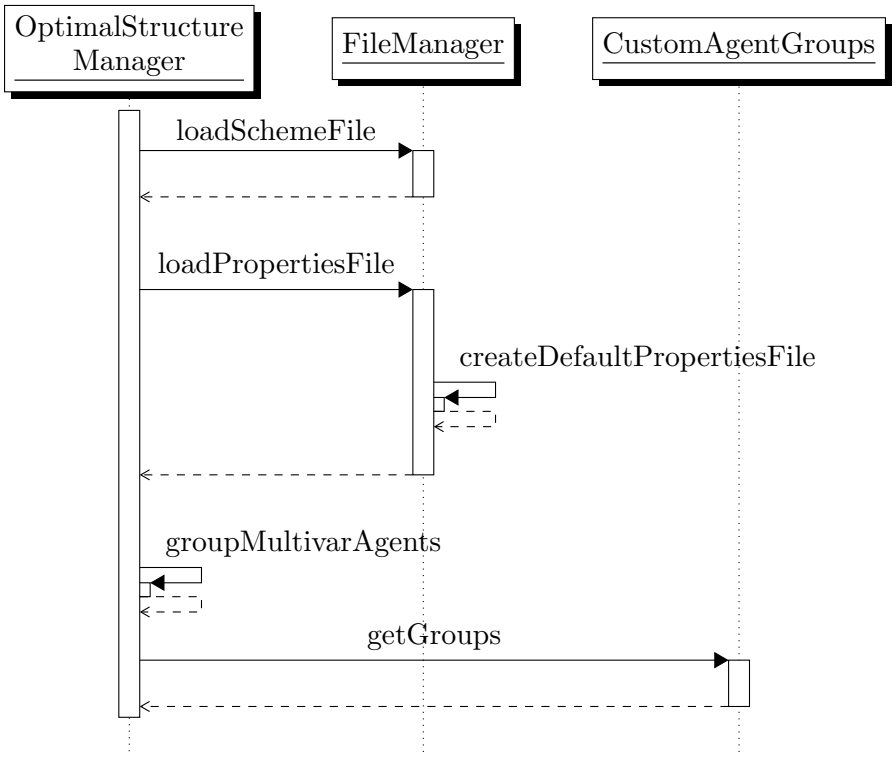

Figura A.1: Diagramma di sequenza del costruttore di OptimalStructure-Manager.

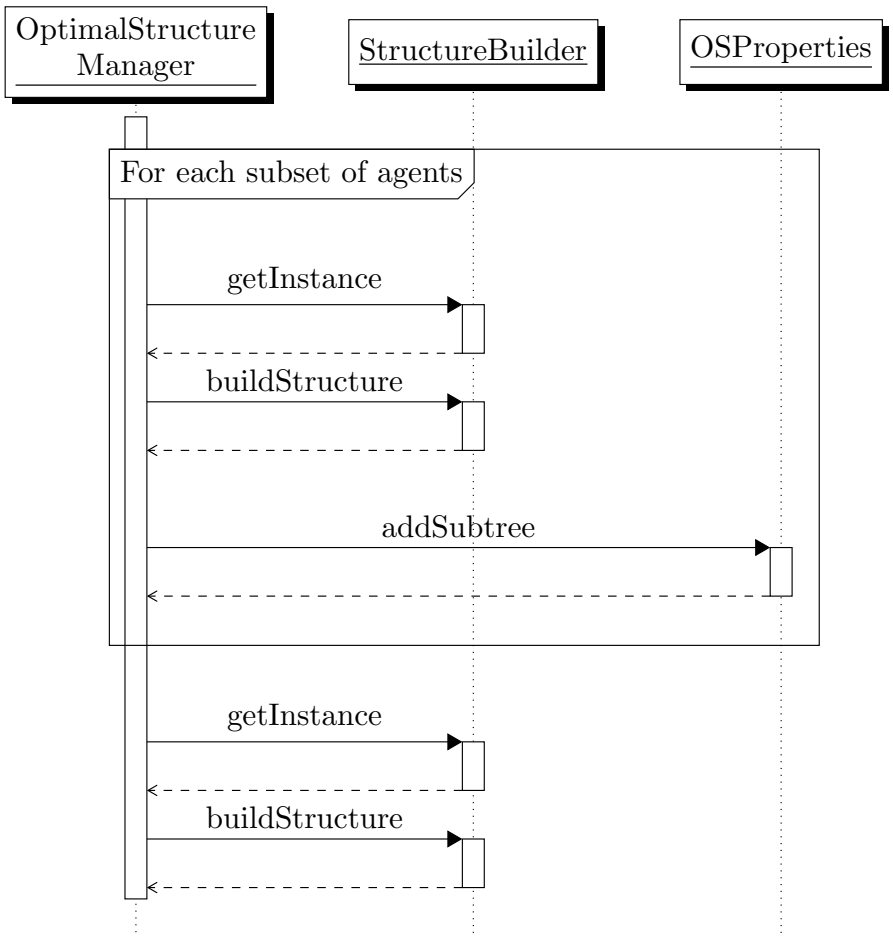

Figura A.2: Diagramma di sequenza del metodo generateOptimalStructure in OptimalStructureManager.

A.1.1 StructureBuilder

L'oggetto StructureBuilder si occupa di istanziare l'implementazione dell'algoritmo di ricerca desiderato e di caricare i dati necessari alla sua esecuzione.

In base all'algoritmo scelto, viene istanziata una specifica sottoclasse, si veda Figura A.3. Al momento sono disponibili quattro opzioni: algoritmo enumerativo, simulated annealing, algoritmo greedy e la scrittura di una struttura predefinita. Per aggiungere un nuovo algoritmo è sufficiente introdurre un ulteriore valore nell'enumerazione che li elenca, implementare l'algoritmo stesso come sottoclasse di StructureBuilder ed aggiornare il metodo getInstance.

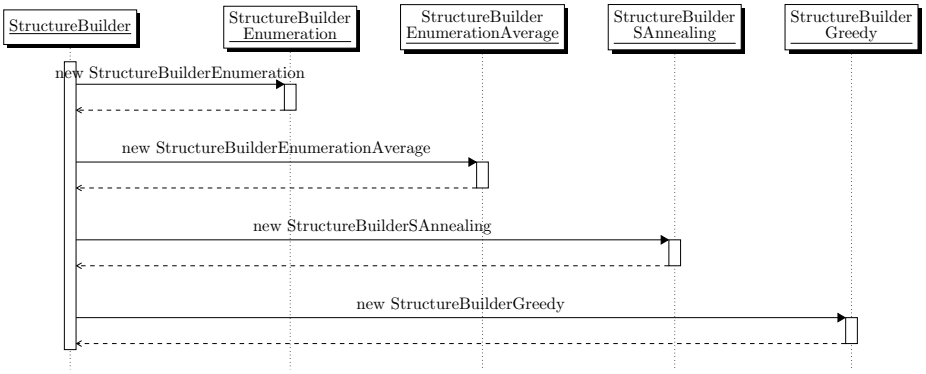

Figura A.3: Metodo getInstance in StructureBuilder per tre dei quattro algoritmi, istanzia l'oggetto relativo all'algoritmo specificato nel file di configurazione.

L'oggetto offre anche un metodo incaricato di caricare i dati necessari all'esecuzione, ossia le probabilità di anomalia e i valori su cui determinare lo score della soluzione, si veda Figura A.4. Se le probabilità di anomalia non sono ancora state calcolate, viene invocato JADE ed esse sono salvate su un file in modo da poter essere ricaricate in un secondo tempo. Ciò consente di eseguire la ricerca con diversi algoritmi e configurazioni senza dover rianalizzare il log ogni volta.

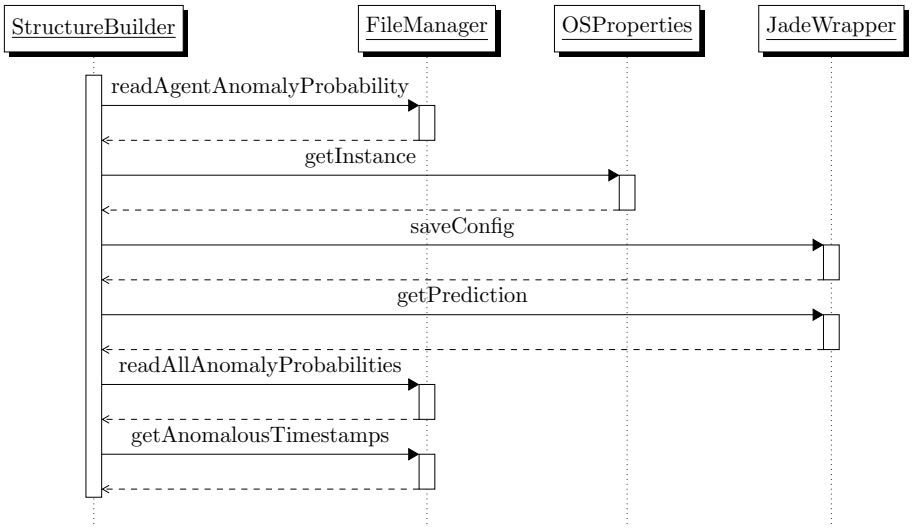

Figura A.4: Metodo loadAnomalyAndTargetProbabilities in StructureBuilder.

La classe offre infine il metodo *logBestSolution* utilizzato per inserire in

un log XML le soluzioni ottime individuate durante la ricerca, si veda Figura A.5. Il metodo integra la soluzione con gli eventuali sottoalberi frutto di iterazioni precedenti dell'algoritmo di ricerca.

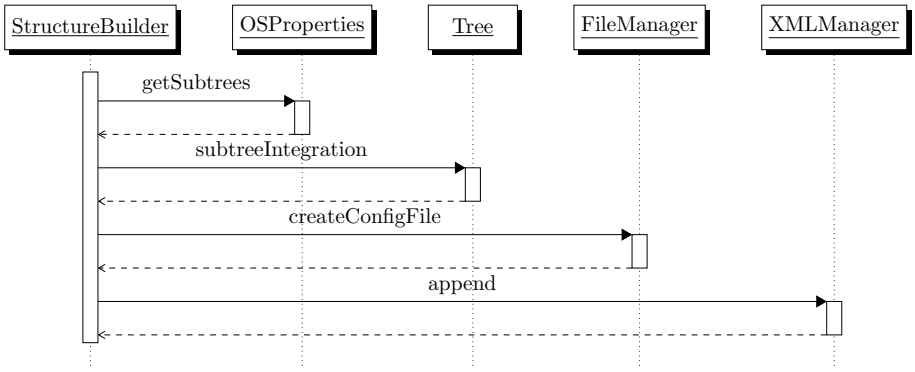

Figura A.5: Metodo logBestSolution in StructureBuilder. Si integra la soluzione ottima trovata con i sottoalberi, si crea il contenuto de file di configurazione e lo si salva nel log XML.

A.2 Algoritmo enumerativo

La ricerca enumerativa è definita come un *for each* su un iteratore. Esso viene istanziato nel costruttore dell'oggetto StructureBuilderEnumeration tramite un metodo della classe astratta SolutionIterator. E' restituito l'iteratore corrispondente alla struttura della soluzione desiderata, si veda Figura A.6. Attualmente sono implementati iteratori per gli alberi albero aventi almeno due figli per ogni nodo interno e gli alberi perfettamente sbilanciati. Gli iteratori esplorano tutti gli alberi di una specifica topologia aventi numero di foglie pari agli agenti diagnostici. Per aggiungere una nuova struttura è sufficiente introdurre un ulteriore elemento nell'enumerazione che le elenca, implementare l'iteratore come sottoclasse di SolutionIterator ed aggiornare il metodo.

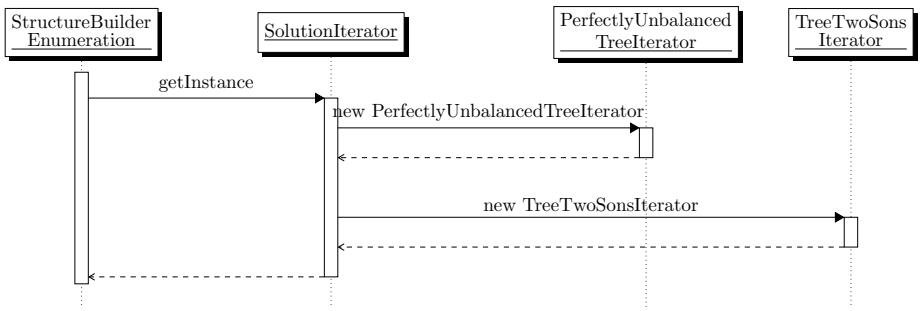

Figura A.6: Metodo getInstance nella classe SolutionIterator, restituisce l'iteratore associato alla struttura di soluzione desiderata.

Il metodo buildStructure è implementato sfruttando il multithreading. Esiste un iteratore globale delle soluzioni che è clonato e passato ad un thread, il quale eseguità un numero di iterazioni prefissato. L'iteratore globale è poi spostato in avanti dello stesso numero di iterazioni per poter essere nuovamente clonato e passato al thread successivo senza che la stessa soluzione sia esplorata più volte. Ogni thread si configura come una ricerca enumerativa su uno spazio di soluzioni più piccolo, il suo risultato sarà la soluzione migliore all'interno del sottospazio di sua competenza. Tali soluzioni sono poi progressivamente aggregate in modo da ottenere, alla terminazione dell'ultimo thread, quella ottima globale. In Figura A.7 è riportato il diagramma di sequenza del metodo.

A.3 Simulated Annealing

Il simulated annealing è definito come un *for* su un numero di iterazioni fissato, in ciascuna delle quali è necessario scegliere una nuova soluzione vicina. In Figura A.8 è riportato il diagramma di sequenza del metodo. Per offrire flessibilità nella scelta di come generare tali vicini, anche in questo caso sono disponibili diverse implementazioni della neighbor function, definite come estensioni della classe astratta NeighborFunction. Attualmente sono disponibili due funzioni. La prima opera su alberi generici i cui mediatori abbiano almeno due figli ed è stata definita nella Sezione 4.3.1, la seconda opera su alberi perfettamente sbilanciati, scambiando una coppia di foglie e garantendo di non scambiare tra loro agenti diagnostici appartenenti al set aggregato dallo stesso mediatore. Per aggiungere una nuova neighbor function è sufficiente introdurre un ulteriore valore nell'enumerazione che le elenca, implementare l'oggetto come sottoclasse di NeighborFunction ed aggiornare il metodo.

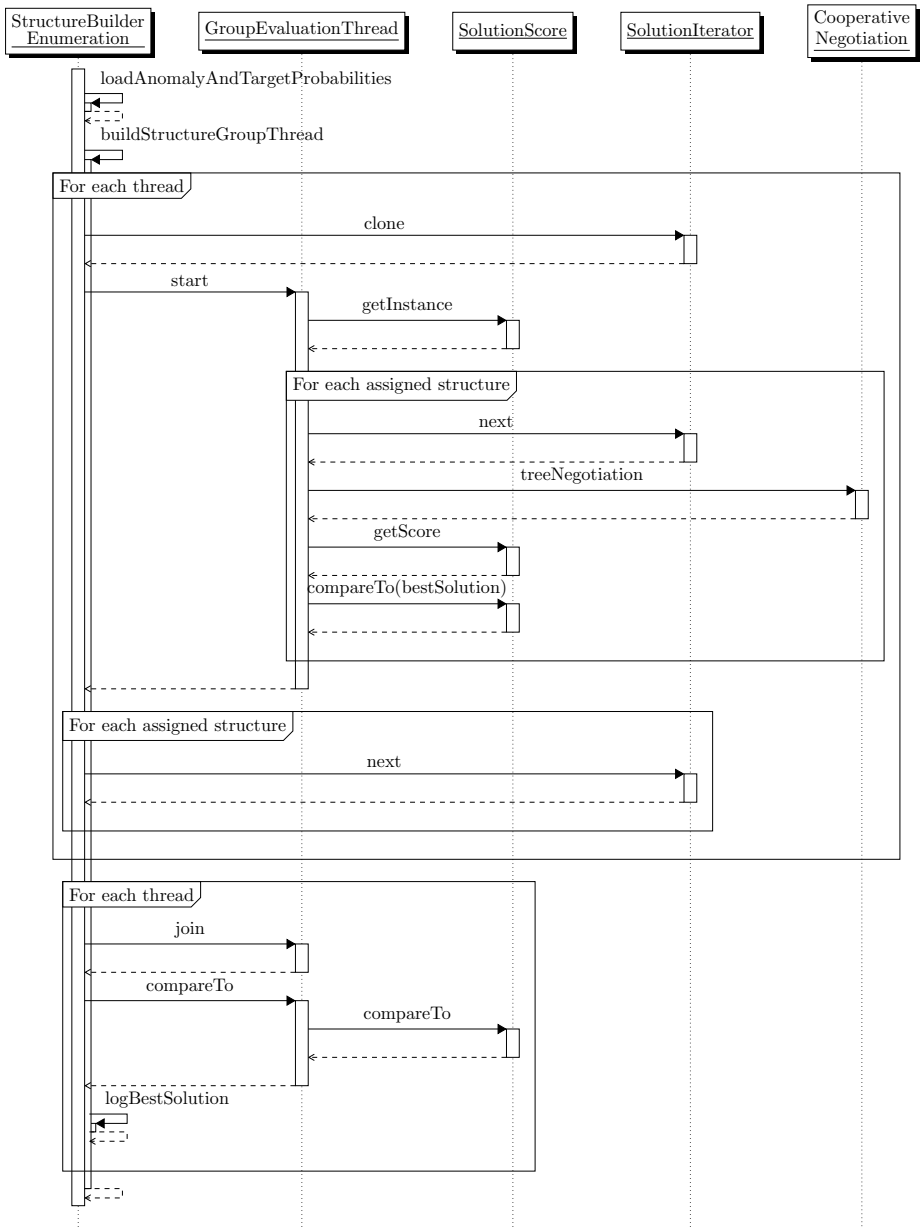

Figura A.7: Metodo buildStructure della classe BuildStructureEnumeration. Ogni thread ha assegnato un certo numero di soluzioni tra le quali sceglie la migliore. Al thread è passato come parametro un clone dell'iteratore posizionato sulla prima soluzione a lui assegnata.

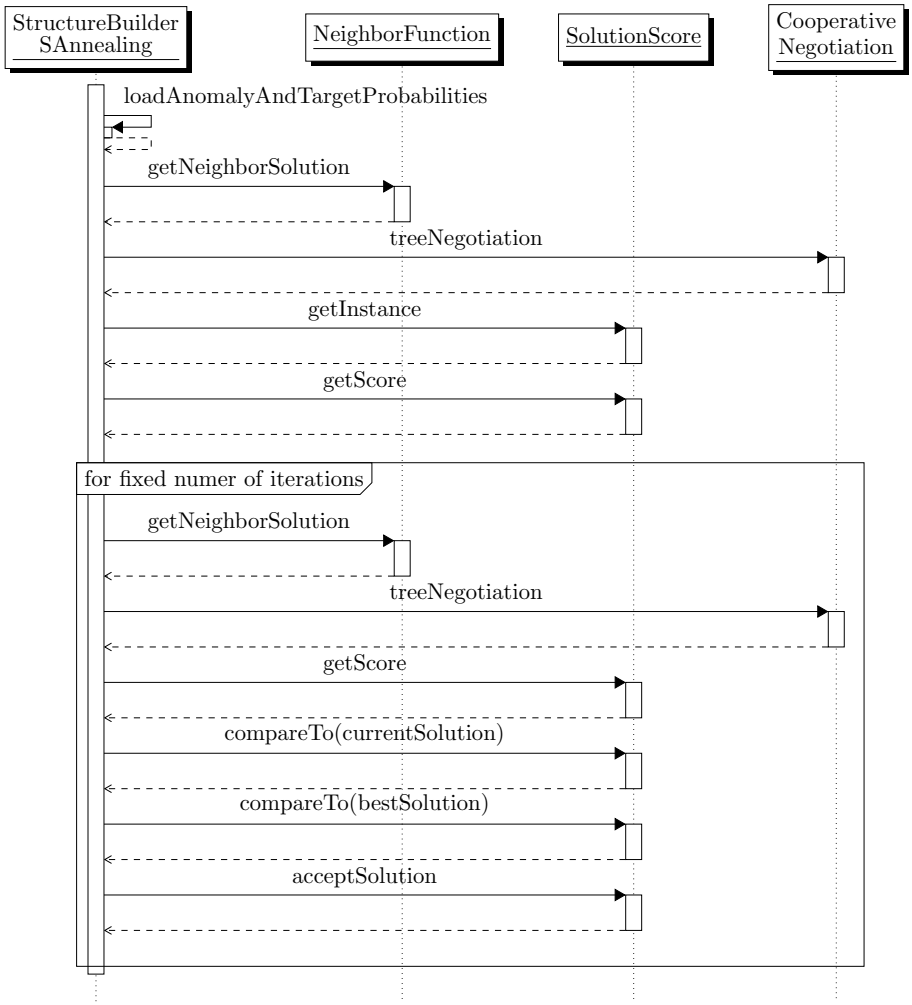

Figura A.8: Metodo buildStructure della classe BuildStructureSAnnealing. L'algoritmo è implmementato in modo sequenziale.

A.4 Enumerazione per funzione media

La ricerca enumerativa sullo spazio delle soluzioni della funzione di aggregazione media è definita come un *for each* su un iteratore di tipo LinearCoefficientsIterator, istanziato nel costruttore dell'oggetto StructureBuilderEnumerationAverage. Esso fa uso di altri iteratori: uno per generare gli scheletri dei possibili alberi ed uno per effettuare permutazioni multiclasse dei coefficienti w. L'iteratore permetterà di esplorare tutti i vettori di coefficienti associati ai possibili alberi. A differenza degli altri algoritmi il calcolo della predizione è effettuato direttamente utilizzando il vettore dei coefficienti senza ricorrere ad un oggetto dedicato. L'algoritmo sfrutta il

multithreading in modo simile alla sua versione che esplora l'intero spazio delle soluzioni, si veda la Figura A.9.

A.5 Algoritmo greedy

L'algoritmo greedy sfrutta il multithreading in modo simile agli algoritmi enumerativi. Ciascun thread eseguirà l'algoritmo greedy ottenendo una soluzione ottima locale. Il risultato di ogni thread sarà poi comparato col migliore sino ad ora ottenuto ed eventualmente la soluzione ottima sarà aggiornata. Il diagramma di sequenza dell'algoritmo è riportato in Figura A.10. Le funzioni che generano le azioni, descritte nella Sezione 4.3.2, sono invece riportate in Figura A.11.

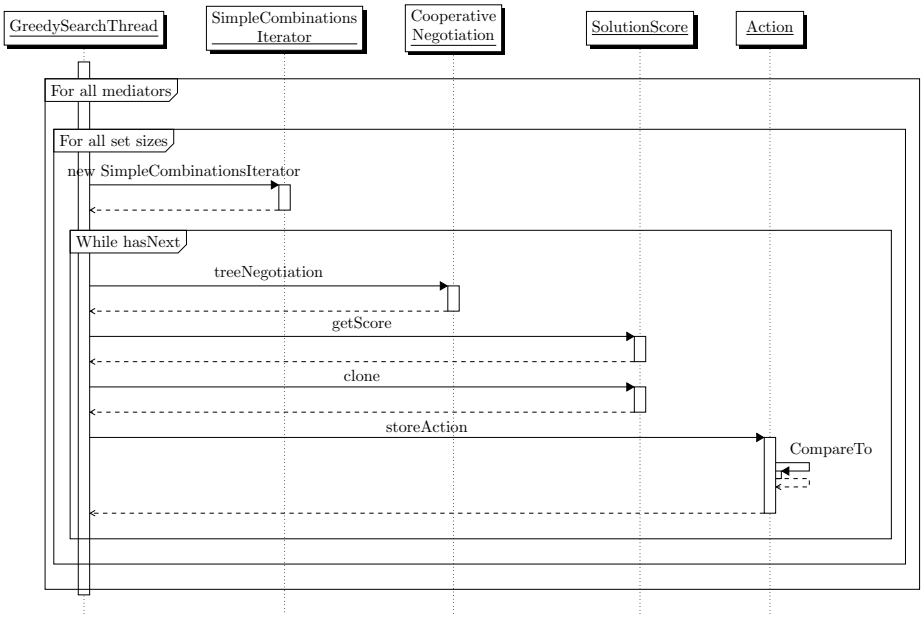

Figura A.11: I metodi generateAddDiagnosticAgentActions e generateNewMediatorAgentActions in BuildStructureGreedy si comportano in modo simile. Sono composti da tre cicli annidati, uno sui nodi mediatori esistenti, uno sulla dimensione del set di agenti da coinvolgere nell'azione ed uno sullo specifico set di agenti.

A.6 Calcolo della predizione tramite Jade

La classe JadeWrapper offre metodi statici per interagire con JADE, ossia salvare la struttura, la configurazione e ottenere le probabilità di anomalia degli agenti diagnostici. Il metodo *saveConfig* necessita due parametri: una

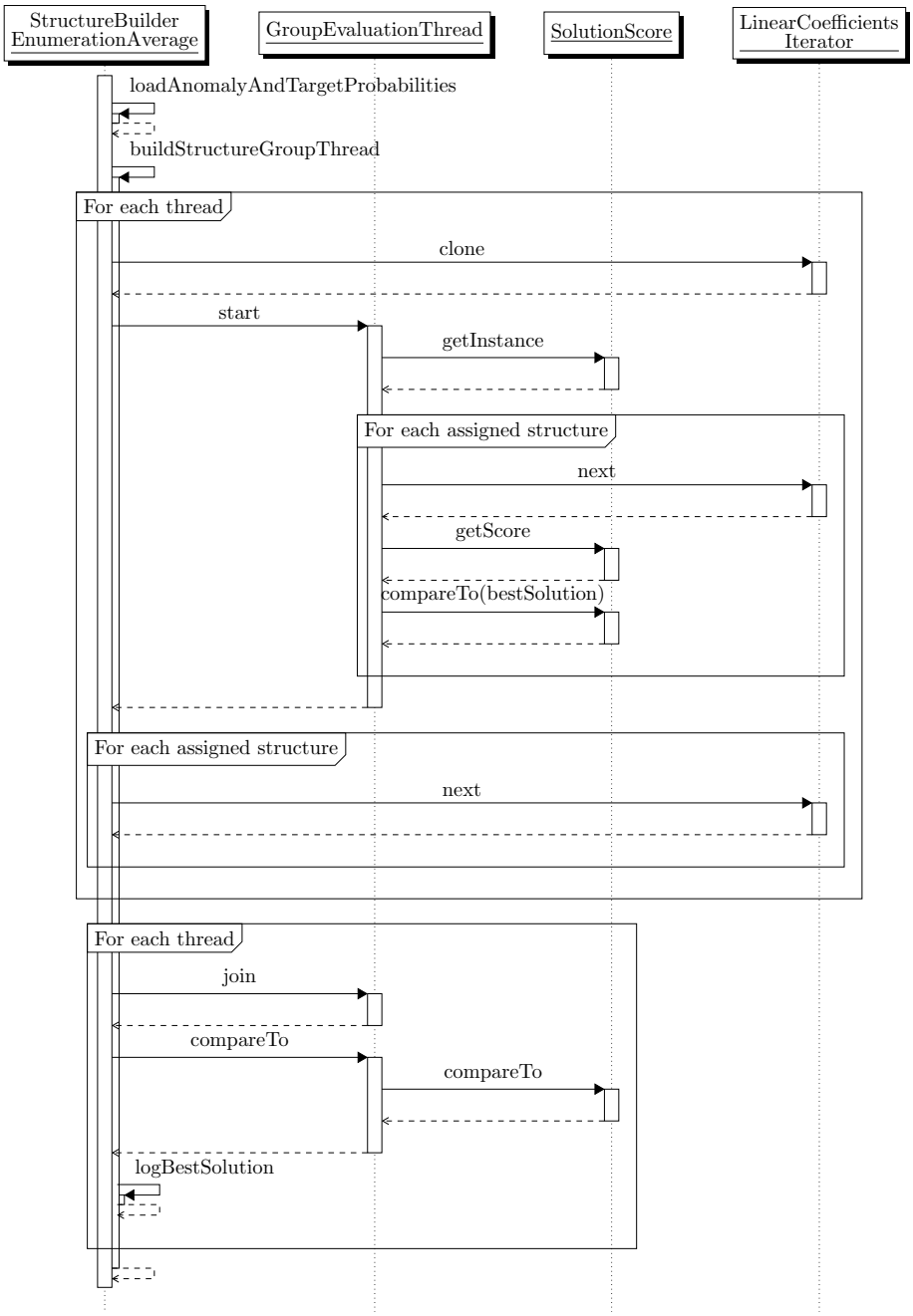

Figura A.9: Metodo buildStructure della classe BuildStructureEnumerationAverage. Ogni thread ha assegnato un certo numero di soluzioni tra le quali sceglie la migliore. Al thread è passato come parametro un clone dell'iteratore posizionato sulla prima soluzione a lui assegnata.

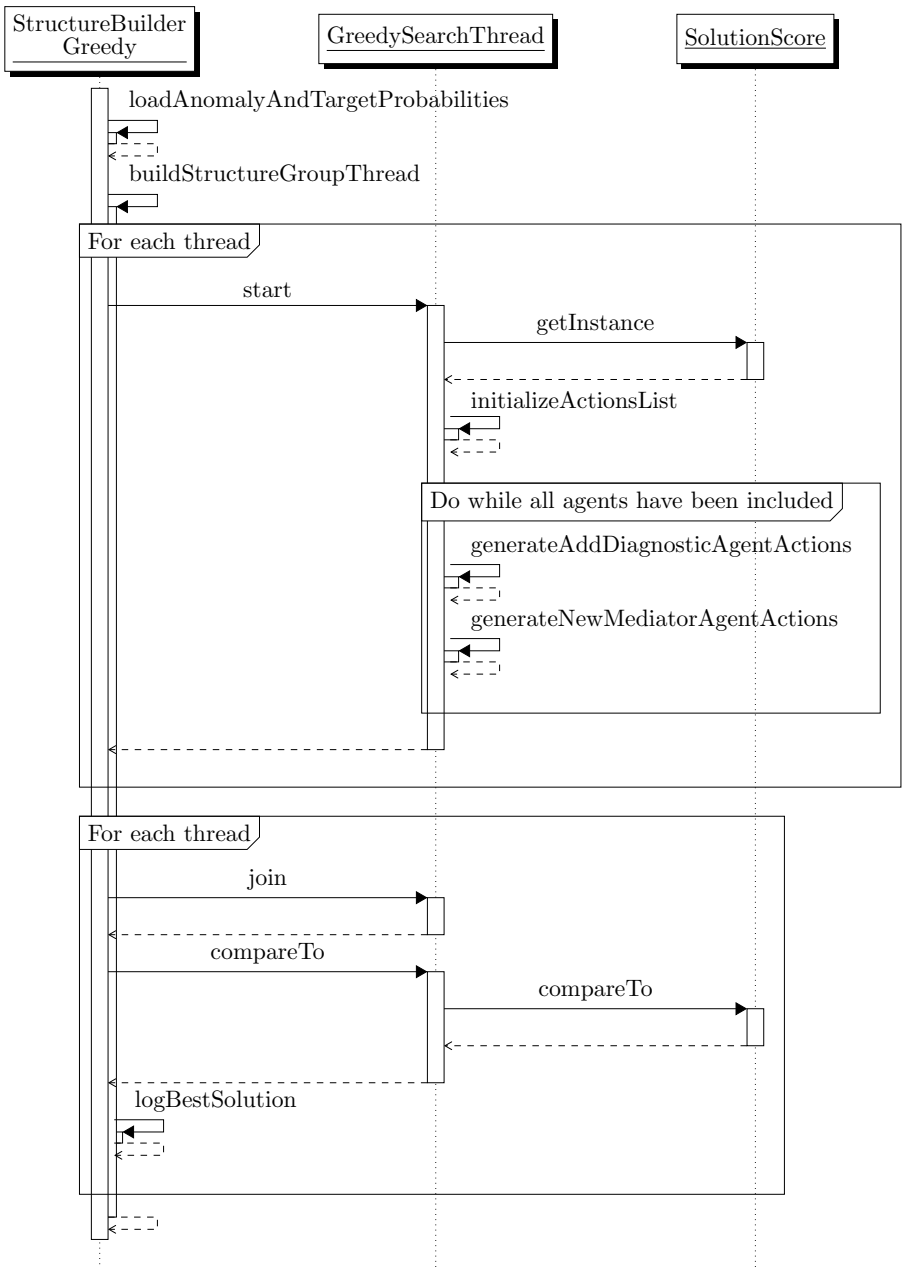

Figura A.10: Metodo buildStructure della classe BuildStructureGreedy. Ogni thread esegue l'algoritmo greedy sino a raggiungere una soluzione ottima locale.

struttura, che sarà integrata con i sottoalberi che già sono stati definiti, e il nome del log da utilizzare. Con questi dati è possibile scrivere la configurazione necessaria a JADE nel file *config.txt*. Il metodo *getPrediction*, in Figura A.12, ottiene l'istanza dell'AppMainController ed avvia JADE. L'AppMainController non è bloccante, ma si auto-termina quando ha concluso l'analisi del log e il calcolo delle probabilità di anomalia. Sfruttando un thread, ogni qualche secondo si verifica se l'AppMainController è ancora in esecuzione. Alla sua terminazione riprende il flusso di esecuzione del metodo statico ed è invocato un secondo metodo per la lettura delle probabilità.

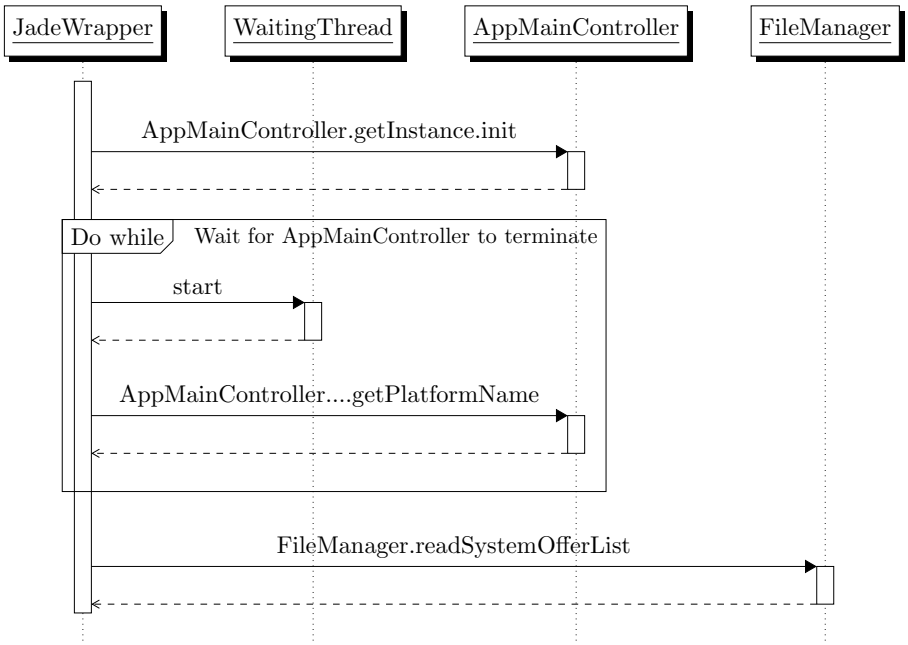

Figura A.12: Metodo getprediction della classe JadeWrapper, avvia JADE e legge i risultati.

A.6.1 File di configurazione della struttura

Il file di configurazione della struttura contiene tutti i dati necessari a JADE legati all'organizzazione gerarchica degli agenti corredata di alcuni parametri. Una configurazione di esempio è riportata in Figura A.13.

Le righe comprese tra le istruzioni *SystemStructure:* e *StartTheSystem:* contengono ciascuna informazioni su un agente. Sono indicati il suo nome, il suo tipo, ed eventuali informazioni accessorie. Dopo il simbolo *:* sono elencati gli agenti i cui risultati sono presi in input e, dopo il simbolo *, quelli che utilizzeranno l'output dell'agente.

Nella prima parte sono riportati i *FilterAgent*, ossia gli agenti che effettuano il parsing dei log disponibili. In questo caso si analizza solo il log MCS

```
SystemStructure:
MCSFilterAgent,FilterAgent:null*A4,A5,A1,A2,A3
A4,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A5,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A1,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A2,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*M00001
A3,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*M00001
M00001,MediatorAgent,Max:A2,A3*GlobalMediator
GlobalMediator,MediatorAgent,CooperativeNegotiation:A4,A5,A1,M00001*null
StartTheSystem:
MCSFilterAgent!Start,./data/logDaAnalizzare.txt,true
```

Figura A.13: Esempio del file di configurazione contenente una struttura.

e pertanto esiste solo il MCSFilterAgent. Seguono i cinque agenti diagnostici, per ciascuno sono indicati i parametri w_{min}, w_{max}, p_{min} e p_{max} utilizzati per l'aggregazione tramite negoziazione cooperativa, come indicato nella Sezione 2.3. Gli agenti diagnostici accetteranno input solo da un FilterAgent e potranno avere in output o il *GlobalMediator* o un altro mediatore.

Successivamente agli agenti diagnostici compaiono i mediatori. Per ogni mediatore è riportata la funzione di aggregazione utilizzata: CooperativeNegotiation, Max o Average. Un agente mediatore accetta input da agenti diagnostici o altri mediatori. Infine è indicato il *GlobalMediator*, che costituisce la radice dell'albero e conterrà il risultato finale. Poichè non avrà nessun ulteriore agente in output, è riportato null.

Dopo l'istruzione *StartTheSystem:* si riportano i FilterAgent con il nome del log che dovranno analizzare ed un flag ad indicare l'ordine di lettura degli eventi.

A.7 Log delle soluzioni esplorate

Nel file di configurazione *OptimalStructure.properties* è possibile richiedere che durante l'esecuzione degli algoritmi di ricerca ogni soluzione ottima venga inserita in un log XML contenente la sua struttura, come da formato config.txt, e il corrispondente score. Questo file offre una garanzia contro possibili terminazioni impreviste preservando parte del risultato fino ad ora raggiunto.

La classe *XMLManager* offre tre metodi statici: uno per creare o svuotare il file ed inizializzarlo, si veda la Figura A.14, uno per l'aggiuta dell'elemento soluzione con tutte le informazioni necessarie, si veda Figura A.15, ed uno per la chiusura dell'elemento radice. Il file è scritto come un semplice file di testo ma seguendo il formato XML, in modo da poter essere facilmente aggiornato in modo incrementale.

Nella classe è implementato anche un ContentHandler che consente di leggere il log ed estrarre la soluzione migliore trovata e l'ultima esplorata. E' quindi possibile, nel caso di algoritmo enumerativo, proseguire la ricerca dall'ultima soluzione salvata nel log. La ripresa dell'esecuzione va specificata nel file di configurazione. Il ContentHandler è in grado di gestire anche due casi di XML malformati causati da un'interruzione imprevista dell'esecuzione:

- L'elemento soluzione è ben formato ma manca la chiusura della radice, quindi la terminazione è avvenuta durante la valutazione di una soluzione successiva. Può trattarsi di una terminazione volontaria;

- L'elemento soluzione stesso è malformato quindi la terminazione è avvenuta durante la scrittura, ad esempio per un errore hardware. L'ultima soluzione valida sarà la precedente.

Il log viene validato, verificando che la struttura letta sia compatibile con quella ottenuta tramite l'iteratore utilizzato dall'enumerazione. Pertanto non potrà essere riavviata l'esecuzione da un log relativo ad una diversa topologia della soluzione, diverso set di agenti, diverso file log o diversa funzione di aggregazione predefinita. Si riconosce parzialmente incompatibilità tra le funzioni di costo.

```
<currentParameters>
<Algorithm>ENUMERATION</Algorithm>
<Solution_structure>ALL_TREES</Solution_structure>
<Aggregation_function>COOPERATIVE_NEGOTIATION</Aggregation_function>
<Evaluation_method>COST_MATRIX</Evaluation_method>
<Neighbor_function>ALL_TREES</Neighbor_function>
<solution_level_is_limited>false</solution_level_is_limited>
<solution_level_limit_value>2</solution_level_limit_value>
</currentParameters>
<agentList>
<agent>A1</agent>
<agent>A2</agent>
<agent>A3</agent>
<agent>A4</agent>
<agent>A5</agent>
</agentList>
```

Figura A.14: Intestazione del file XMLLog. Si possono osservare tutti i parametri che compaiono nel file di configurazione e l'elenco degli agenti coinvolti.

```
<newSolution><configuration>SystemStructure:
MCSFilterAgent,FilterAgent:null*A1,A2,A4,A5,A3
A1,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A2,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A4,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A5,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
A3,DiagnosticAgent,0.1,0.9,0.1,0.9:MCSFilterAgent*GlobalMediator
GlobalMediator,MediatorAgent,CooperativeNegotiation:A1,A2,A4,A5,A3*null
StartTheSystem:
MCSFilterAgent!Start,./data/logDaAnalizzare.txt,true</configuration>
<score>30.0</score>
<confusionMatrix>[ TP: 10, FN: 0], [ FP: 3, TN: 7]</confusionMatrix>
<falsePositiveList>
<FP>3</FP>
<FP>4</FP>
<FP>9</FP>
</falsePositiveList>
</newSolution>
```

Figura A.15: Elemento *newSolution* nel file XMLLog. Oltre alla struttura sono riportati lo score, la confusion matrix e la lista di falsi positivi e falsi negativi, se presenti.

A.8 File di configurazione dei sottoalberi

Qualora si voglia suddividere il set degli agenti diagnostici svolgendo la ricerca della struttura ottima per alcuni di loro separatamente, è possibile farlo tramite il file *CustomAgentGroups*. La sintassi prevede di indicare un'etichetta per il raggruppamento ed elencare gli agenti che vi facciano parte, si veda la Figura A.16.

```
<groupsList>
<newGroup label="SOTTOSISTEMA_A">
<agent>A0001</agent>
<agent>A0002</agent>
<agent>A0003</agent>
<agent>A0004</agent>
</newGroup>
<newGroup label="SOTTOSISTEMA_B">
<agent>B0001</agent>
<agent>B0002</agent>
<agent>B0003</agent>
</newGroup>
</groupsList>
```

Figura A.16: Sono elencati gli agenti appartenenti a due sottosistemi.

A.9 File di configurazione OptimalStructureManager

Nel file di configurazione *OptimalStructure.properties* sono riportate le opzioni che determineranno il comportamento del modulo sviluppato contestualmente a questa tesi e dell'algoritmo di ricerca. Nel file sono elencati diversi parametri, sotto il nome di ciascuno è specificato il valore di default e la sequenza di tutti i valori disponibili. I campi non booleani sono enumerazioni. E' sufficiente che la riga corrispondente al valore scelto non sia commentata. Nel caso di stringa non riconosciuta viene utilizzato il valore di default e nel caso di file non trovato esso viene ricostruito.

Nella prima parte del file si trovano i parametri legati all'algoritmo ed alla struttura della soluzione, ossia la sua topologia ed eventuali limiti nel livello, si veda la Figura A.17.

Nella seconda parte del file invece si trovano parametri legati alle funzioni di aggregazione, alla neighbor function e all'impiego del log XML, si veda Figura A.18.

```
#### PROPERTIES FILE FOR OPTIMAL STRUCTURE MODULE

#### Available options are listed below, uncomment
#### the row corresponding to the desired value.
#### If no value is specified, default will be used.


#### ALGORITHM
## Default Algorithm = ENUMERATION
#Algorithm = ENUMERATION
#Algorithm = SIMULATED_ANNEALING
#Algorithm = PREDEFINED_CONFIG
#Algorithm = ENUMERATION_FAST_AVERAGE
Algorithm = GREEDY

#### TARGET SOLUTION STRUCTURE
## Default Solution_structure = ALL_TREES
Solution_structure = ALL_TREES
#Solution_structure = ALL_TREES_MIXED_AGGREGATION
#Solution_structure = PERFECTLY_UNBALANCED_TREES

#### IS TREE LEVEL LIMITED
## Default solution_level_is_limited = true
#solution_level_is_limited = true
solution_level_is_limited = false

#### MAX TREE LEVEL
## Default solution_level_limit_value = 2
#solution_level_limit_value = Any positive integer value

#### MAX COEFFICIENT FOR AVERAGE AGGREGATION FUNCTION
## Default solution_coefficient_limit_value = 10
#solution_coefficient_limit_value = Any positive integer value
```

Figura A.17: Prima parte del file OptimalStructure.properties. Sono riportate informazioni legate all'algoritmo ed alla tpopologia desiderata per la soluzione.

```
#### AGGREGATION FUNCTION FOR MEDIATOR AGENTS
## Default Aggregation_function = COOPERATIVE_NEGOTIATION
Aggregation_function = COOPERATIVE_NEGOTIATION
#Aggregation_function = MAX

#### SOLUTION EVALUATION METHOD
## Default Evaluation_method = COST_MATRIX
#Evaluation_method = EUCLIDEAN_DISTANCE
#Evaluation_method = CONFUSION_MATRIX
Evaluation_method = COST_MATRIX

#### NEIGHBOR FUNCTION FOR SIMULATED ANNEALING
## Default Neighbor_function = ALL_TREES
#Neighbor_function = SWAP_LEAVES
Neighbor_function = ALL_TREES

#### LOG ALL SOLUTIONS EXPLORED TO XML FILE
## Default xml_solution_log = true
xml_solution_log = true
#xml_solution_log = false

#### RESUME EXECUTION FROM XML LOG FILE - ONLY AVAILABLE FOR ENUMERATION
## Default resume_from_xml_solution_log = false
#resume_from_xml_solution_log = true
resume_from_xml_solution_log = false
```

Figura A.18: Seconda parte del file OptimalStructure.properties. Sono riportate informazioni legate alla funzione di aggregazione, alla neighbor function e all'uso del log XML.

Bibliografia

[1] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

[2] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[3] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 313–326, 2003.

[4] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, 2014.

[5] T. Escobet, D. Feroldi, S. De Lira, V. Puig, J. Quevedo, J. Riera, and M. Serra, "Model-based fault diagnosis in pem fuel cell systems," *Journal of Power Sources*, vol. 192, no. 1, pp. 216–223, 2009.

[6] J. Shiozaki, H. Matsuyama, E. O'shima, and M. Iri, "An improved algorithm for diagnosis of system failures in the chemical process," *Computers & Chemical Engineering*, vol. 9, no. 3, pp. 285–293, 1985.

[7] R. Reiter, "A theory of diagnosis from first principles," *Artificial intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[8] L. W. Chen and M. Modarres, "Hierarchical decision process for fault administration," *Computers & Chemical Engineering*, vol. 16, no. 5, pp. 425–448, 1992.

[9] C. Rojas-Guzman and M. A. Kramer, "Comparison of belief networks and rule-based expert systems for fault diagnosis of chemical processes," *Engineering Applications of Artificial Intelligence*, vol. 6, no. 3, pp. 191–202, 1993.

[10] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[11] H. Hotelling, "Multivariate quality control," in *Techniques of statistical analysis*, pp. 113–184, McGraw-Hill, 1947.

[12] S. Wold, C. Albano, W. Dunn III, U. Edlund, K. Esbensen, P. Geladi, S. Hellberg, E. Johansson, W. Lindberg, and M. Sjöström, "Multivariate data analysis in chemistry," in *Chemometrics*, pp. 17–95, Springer, 1984.

[13] A. Cinar, S. Perk, F. Teymour, M. North, E. Tatara, and M. Altaweel, "Monitoring, analysis, and diagnosis of distributed processes with agent-based systems," in *Proceedings of the International IFAC Symposium on Advanced Control of Chemical Processes (ADCHEM)*, https://www.researchgate.net/profile/Sinem_Perk/publication/228745284_Monitoring_Analysis_and_Diagnosis_of_Distributed_Processes_with_Agent-based_Systems/links/09e41509c08f8ee3e8000000.pdf, 2009.

[14] M. Garza and M. Gallo, "Un sistema ad agenti cooperativi per la fault detection in applicazioni spaziali," Tesi di Laurea Magistrale, Politecnico di Milano, 2013.

[15] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, pp. 1329–1360, 2012.

[16] A. K. Jalan and A. Mohanty, "Model based fault diagnosis of a rotor–bearing system for misalignment and unbalance under steady-state condition," *Journal of Sound and Vibration*, vol. 327, no. 3, pp. 604–622, 2009.

[17] Y. Amirat, M. E. H. Benbouzid, E. Al-Ahmar, B. Bensaker, and S. Turri, "A brief status on condition monitoring and fault diagnosis in wind energy conversion systems," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 9, pp. 2629–2636, 2009.

[18] F. Aguilera, P. de la Barrera, C. De Angelo, and D. E. Trejo, "Current-sensor fault detection and isolation for induction-motor drives using a geometric approach," *Control Engineering Practice*, vol. 53, pp. 35–46, 2016.

[19] Z. Hameed, Y. Hong, Y. Cho, S. Ahn, and C. Song, "Condition monitoring and fault detection of wind turbines and related algorithms:

A review," *Renewable and Sustainable energy reviews*, vol. 13, no. 1, pp. 1–39, 2009.

[20] P. M. Frank, "On-line fault detection in uncertain nonlinear systems using diagnostic observers: a survey," *International journal of systems science*, vol. 25, no. 12, pp. 2129–2154, 1994.

[21] N. H. Ulerich and G. J. Powers, "On-line hazard aversion and fault diagnosis in chemical processes: the digraph+ fault-tree method," *IEEE Transactions on Reliability*, vol. 37, no. 2, pp. 171–177, 1988.

[22] F. E. Finch and M. A. Kramer, "Narrowing diagnostic focus using functional decomposition," *AIChE Journal*, vol. 34, no. 1, pp. 25–36, 1988.

[23] M. Lind, "Reasoning about causes and consequences in multilevel flow models," in *Advances in Safety, Reliability and Risk Management: ESREL 2011*, pp. 18–22, CRC Press, 2011.

[24] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[25] D. Leung and J. Romagnoli, "An integration mechanism for multivariate knowledge-based fault diagnosis," *Journal of Process Control*, vol. 12, no. 1, pp. 15–26, 2002.

[26] W. R. Becraft and P. Lee, "An integrated neural network/expert system approach for fault diagnosis," *Computers & Chemical Engineering*, vol. 17, no. 10, pp. 1001–1014, 1993.

[27] E. E. Tarifa and N. J. Scenna, "Fault diagnosis, direct graphs, and fuzzy logic," *Computers & Chemical Engineering*, vol. 21, pp. S649–S654, 1997.

[28] K. Patan, M. Witczak, and J. Korbicz, "Towards robustness in neural network based fault diagnosis," *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 4, pp. 443–454, 2008.

[29] D. Leung and J. Romagnoli, "An integration mechanism for multivariate knowledge-based fault diagnosis," *Journal of Process Control*, vol. 12, no. 1, pp. 15–26, 2002.

[30] D. Lieftucht, U. Kruger, and G. W. Irwin, "Improved reliability in diagnosing faults using multivariate statistics," *Computers & Chemical Engineering*, vol. 30, no. 5, pp. 901–912, 2006.

[31] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application*, vol. 104. Prentice Hall Englewood Cliffs, 1993.

[32] M.-D. Ma, D. S.-H. Wong, S.-S. Jang, and S.-T. Tseng, "Fault detection based on statistical multivariate analysis and microarray visualization," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 18–24, 2010.

[33] Y. Yao, T. Chen, and F. Gao, "Multivariate statistical monitoring of two-dimensional dynamic batch processes utilizing non-gaussian information," *Journal of Process Control*, vol. 20, no. 10, pp. 1188–1197, 2010.

[34] V. Choqueuse, M. E. H. Benbouzid, Y. Amirat, and S. Turri, "Diagnosis of three-phase electrical machines using multidimensional demodulation techniques," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 4, pp. 2014–2023, 2012.

[35] C.-M. Vong, P.-K. Wong, and W.-F. Ip, "A new framework of simultaneous-fault diagnosis using pairwise probabilistic multi-label classification for time-dependent patterns," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 3372–3385, 2013.

[36] R. Muradore and P. Fiorini, "A pls-based statistical approach for fault detection and isolation of robotic manipulators," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3167–3175, 2012.

[37] X. He, Z. Wang, Y. Liu, and D. Zhou, "Least-squares fault detection and diagnosis for networked sensing systems using a direct state estimation approach," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1670–1679, 2013.

[38] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 30, no. 1, pp. 179–196, 1995.

[39] W. Li, H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive pca for adaptive process monitoring," *Journal of process control*, vol. 10, no. 5, pp. 471–486, 2000.

[40] S. W. Choi, C. Lee, J.-M. Lee, J. H. Park, and I.-B. Lee, "Fault detection and identification of nonlinear processes based on kernel pca," *Chemometrics and intelligent laboratory systems*, vol. 75, no. 1, pp. 55–67, 2005.

[41] Z. Guo, H. Yue, and H. Wang, "A modified pca based on the minimum error entropy," in *Proceedings of the American Control Conference.*, pp. 3800–3801, 2004.

[42] J.-M. Lee, C. Yoo, and I.-B. Lee, "Statistical process monitoring with independent component analysis," *Journal of Process Control*, vol. 14, no. 5, pp. 467–485, 2004.

[43] J.-M. Lee, C. Yoo, and I.-B. Lee, "Statistical monitoring of dynamic processes based on dynamic independent component analysis," *Chemical engineering science*, vol. 59, no. 14, pp. 2995–3006, 2004.

[44] S. J. Qin, "Recursive pls algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4, pp. 503–514, 1998.

[45] P. Smyth, "Hidden markov models for fault detection in dynamic systems," *Pattern recognition*, vol. 27, no. 1, pp. 149–164, 1994.

[46] J. A. Leonard and M. A. Kramer, "Diagnosing dynamic faults using modular neural nets," *IEEE Expert*, vol. 8, no. 2, pp. 44–53, 1993.

[47] B. R. Bakshi and G. Stephanopoulos, "Wave-net: A multiresolution, hierarchical neural network with localized learning," *AIChE Journal*, vol. 39, no. 1, pp. 57–81, 1993.

[48] D. Cayrac, D. Dubois, and H. Prade, "Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 251–269, 1996.

[49] M. Lavagna, G. Sangiovanni, and A. Da Costa, "Modelization, failures identification and high-level recovery in fast varying non-linear dynamical systems for space autonomy," in *Dynamics and Control of Systems and Structures in Space (DCSSS), 6th Conference*, pp. 451–550, 2004.

[50] N. Holsti and M. Paakko, "Towards advanced FDIR components," *Data Systems in Aerospace, DASIA*, https://www.researchgate.net/profile/Niklas_Holsti/publication/228484215_Towards_Advanced_FDIR_Components/links/09e4150a54bd8ecb7a000000.pdf, 2001.

[51] K. C. Daly, E. Gai, and J. V. Harrison, "Generalized likelihood test for FDI in redundant sensor configurations," *Journal of Guidance, Control, and Dynamics*, vol. 2, no. 1, pp. 9–17, 1979.

[52] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[53] P. Massioni, G. Sangiovanni, and M. Lavagna, "Innovative software for autonomous fault detection and diagnosis on space systems," in *9th ESA workshop on Advanced*

Space Technologies for Robotics and Automation (ASTRA), http://robotics.estec.esa.int/ASTRA/Astra2006/Papers/ASTRA2006-2.4.1.03.pdf, 2006.

[54] D. Codetta-Raiteri, L. Portinale, A. Guiotto, and Y. Yushstein, "Evaluation of anomaly and failure scenarios involving an exploration rover: a bayesian network approach," in *Proceedings of the 11th international symposium on artificial intelligence, robotics, and automation in space (iSAIRAS-2012)*, http://robotics.estec.esa.int/i-SAIRAS/isairas2012/Papers/Session 3C/03C_04_codetta.pdf, 2012.

[55] E. E. Smith and D. J. Korsmeyer, "Intelligent systems technologies for human space exploration mission operations," in *Fourth International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, pp. 169–176, IEEE, 2011.

[56] D. L. Iverson, "Data mining applications for space mission operations system health monitoring," in *Proceedings of the SpaceOps Conference*, https://ti.arc.nasa.gov/m/pub-archive/1463h/1463 (Iverson).pdf, 2008.

[57] I. Avrekh, D. Iverson, and B. McDermott, "Odvec monitoring system certified and deployed in jsc mission control center." http://www.nasa.gov/centers/ames/orgs/exploration-tech/tech-updates/tech-update-odvec.html, 2012.

[58] S. J. Russell and P. Norvig, *Intelligenza artificiale. Un approccio moderno*, vol. 1. Pearson Italia Spa, 2005.

[59] M. Wooldridge, *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[60] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[61] Fipa, "Fipa acl message structure specification," *Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00061/SC00061G.html*, 2002.

[62] U. Endriss, "Monotonic concession protocols for multilateral negotiation," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 392–399, ACM, 2006.

[63] "ESA EO missions - cryosat." https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/cryosat.

[64] "ESA EO missions - cryosat mission and data description." http://esamultimedia.esa.int/docs/Cryosat/Mission_and_Data_Descrip.pdf.

[65] "Cross support reference model - part 1: Space link extension services," in *Blue Book*, CCSDS - Consultative Committee for Space Data Systems, http://public.ccsds.org/publications/archive/910x4b2e1.pdf, 2005.

[66] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*, vol. 7. John Wiley & Sons, 2007.

[67] A. Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing.* John Wiley & Sons, Inc., 1993.

[68] J. D. Hamilton, *Time series analysis*, vol. 2. Princeton university press, 1994.

[69] N. Ye *et al.*, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166, pp. 169–174, 2000.

[70] "New horizons - NASA's mission to Pluto." http://pluto.jhuapl.edu/Mission/Where-is-New-Horizons/index.php.

[71] "Voyager - the interstellar mission. where are the voyagers?." http://voyager.jpl.nasa.gov/where/.

[72] "Mars science laboratory - Curiosity rover. Data rates/returns." http://mars.jpl.nasa.gov/msl/mission/communicationwithearth/data/.

[73] C.-R. Hwang, "Simulated annealing: theory and applications," *Acta Applicandae Mathematicae*, vol. 12, no. 1, pp. 108–111, 1988.

[74] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[75] W. R. Gilks, *Markov chain monte carlo.* Wiley Online Library, 2005.

[76] L. Ingber *et al.*, "Adaptive simulated annealing (ASA): Lessons learned," *Control and cybernetics*, vol. 1, pp. 33–54, 1996.

[77] W. Wenzel and K. Hamacher, "Stochastic tunneling approach for global minimization of complex potential energy landscapes," *Physical Review Letters*, vol. 82, no. 15, pp. 3003–3017, http://arxiv.org/pdf/physics/9903008.pdf, 1999.

[78] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[79] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner tree problem*, vol. 53. Elsevier, 1992.

[80] S. Hougardy, "The floyd–warshall algorithm on graphs with negative cycles," *Information Processing Letters*, vol. 110, no. 8, pp. 279–281, 2010.

[81] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, pp. 973–978, Lawrence Erlbaum Associates LTD, 2001.