Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Process mining and process discovery sui processi di programmazione di robot lego

Corso di Big Data Analytics and Machine Learning

Professori

Prof. Domenico Potena
Prof.ssa Claudia Diamantini

Studenti

Michele Pasqualini
Denil Nicolosi

Anno accademico 2021-2022

Indice

1 Introduzione

In questo progetto ci si è posti l'obiettivo di analizzare la metodologia di programmazione di robot di marca *LEGO* da degli studenti di diverse scuole e di diverso ordine e grado. I ragazzi formando dei gruppi dovevano svolgere due esercizi diversi:

- Esercizio A: programmare il robot in modo da percorrere esattamente un metro di distanza dal punto iniziale.
- Esercizio B: programmare il robot in modo tale da raggiungere una determinata distanza da un obiettivo (sfruttando un sensore ad ultrasuoni già equipaggiato).

La programmazione è avvenuta attraverso il software LEGO® MINDSTORMS® EV3, un software che permette di programmare i robot in modo visuale. Vengono riportati di seguito in figura due esempi di programmazione dei due esercizi.

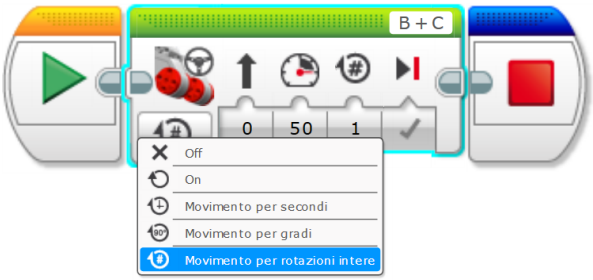

Figura 1: Esempio esercizio A

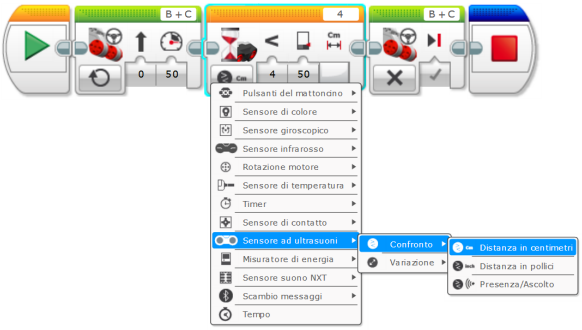

Figura 2: Esempio esercizio B

Questa analisi è stata realizzata a partire da dati di log ottenuti dai medesimi robot. Ovvero, ogni volta che un gruppo effettuava un tentativo di esecuzione, venivano descritti tutti i blocchi utilizzati in un file di testo (uno per ogni gruppo) in formato .rtf (Rich Text Format), separando ogni tentativo dall'altro mediante l'istruzione "`STOP PROGRAM;`". Sono stati forniti inoltre i risultati ottenuti da ogni gruppo per ciascun esercizio, cioè se hanno raggiunto o meno l'obiettivo e con quale errore.

1.1 Obiettivi dell'analisi

L'obiettivo dell'analisi è quello di intercettare dei pattern significativi nelle esecuzioni dei diversi gruppi, ovvero confrontare i comportamenti caratteristici dei gruppi che raggiungono l'obiettivo rispetto a quelli che non lo raggiungono. Questo è utile per fare analisi sull'apprendimento dei ragazzi, ovvero scoprire qual'è il miglior approccio da adottare che molto più probabilmente porterà a concludere l'esercizio con successo.

1.2 Descrizione del task

Per raggiungere l'obiettivo dell'analisi, vengono utilizzate tecniche di process discovery, al fine di trovare appunto il modello di processo che descrive le varie esecuzioni per i due gruppi diversi, ovvero chi ha raggiunto l'obiettivo e chi non ci è riuscito. Per arrivare ad ottenere il modello, è stato necessario prima effettuare un preprocessing, ovvero tradurre il dataset delle diverse esecuzioni in un dataset di azioni eseguite dai ragazzi, cercando di ricavare queste informazioni fra le differenze tra due esecuzioni. Questo processo viene descritto in sezione 3. Una volta ottenute le varie azioni eseguite per ogni gruppo, il nuovo dataset potrà essere utilizzato dal task di process discovery che costruirà un modello di processo utilizzando anche diversi parametri per regolare il filtraggio delle attività e dei path. Questo processo viene descritto in sezione 4.

2 Dataset

Il dataset completo è composto da diversi file di testo (Rich Text Format) dove ognuno di essi contiene i dati e i parametri inseriti dai ragazzi per la programmazione dei robot con l'obiettivo di raggiungere l'esercizio A e B. Vi sono inoltre due file .csv (uno per ogni esercizio) che indicano se ogni gruppo ha raggiunto l'obiettivo e con quale errore.

2.1 Struttura del dataset

Come già anticipato, i dati si dividono in due directory il quale si riferiscono alla tipologia di esercizio. Ogni sotto directory è suddivisa in due categorie: grandi e piccoli. La struttura del dataset viene descritta nella figura sottostante:

Nome campo	**Descrizione**
Blockname	Indica il tipo di blocco che viene inserito
Type	Canale che ha pubblicato il video
1st-param	Il primo parametro su cui si va ad agire
2st-param	Il secondo parametro su cui si va ad agire
3st-param	Il terzo parametro su cui si va ad agire
4st-param	Il quarto parametro su cui si va ad agire

Tabella 1: Descrizione del dataset

Ogni file contiene una lista di esecuzioni effettuate dai ragazzi che terminano con il comando "`STOP PROGRAM;`".

```
Blockname, Type, 1st-param, 2nd-param, 3rd-param, 4th-param
MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;

MoveSteering, OnForRotations, Rotations = 5.6338, Speed = 50, Steering = 0, MotorPorts = 123;
STOP PROGRAM;
```

Figura 3: Esempio file che descrive l'esecuzione di un gruppo

Inoltre, vengono forniti due file csv, uno per ogni tipologia di esercizio, in cui è possibile identificare i gruppi che hanno completato il task insieme ad un margine errore. In particolare, se il gruppo ha raggiunto l'obiettivo, questo viene denotato con "1", altrimenti è assegnato il valore "0".

```
ID,Y,Errore
2019_TALENTcamp_Osimo_g2,0,39
2019_IC_NovelliNatalucci_G9,0,24
2019_IC_Badaloni_Recanati_G2,1,2.5
2019_IC_Lotto_Jesi_sec_g4,1,0.5
2018_ICMontecassiano2,1,3.7
2019_ICLargoCocconi4E_G5,1,2
2018_LiceoVoltaFellini4A,1,1.5
2019_IC_Alighieri_G3,1,0.5
2018_ICNovelliNatalucci3,0,21
2019_ICLargoCocconi4E_G4,0,15
2019_ICLucaDellaRobbia_G3,0,55
2018_LiceoVoltaFellini4F,1,2
2019_IC_Alighieri_G1,0,4.5
2019_IC_NovelliNatalucci_G11,0,50
2018_LiceoVoltaFellini1A,1,3
2019_IC_FalconaraCentro_g3,1,1
2018_LiceoVoltaFellini7A,0,4
2018_ICCorinaldo7,1,0.5
2019_ICSerraSanQuirico3,1,1.5
2019_IC_Leopardi_Gualdo_g6,1,3.5
2019_TALENTcampJesi_g1,1,0
2019_IC_Lotto_Jesi_prim_G1,0,21
2019_IC_Leopardi_Gualdo_g3,1,0.5
2018_ICCorinaldo6,1,2
2018_LiceoVoltaFellini5F,0,6
2019_IC_Lotto_Jesi_prim_G3,0,20
2019_IC_Alighieri_G6,0,19
2019_ICLargoCocconi4F_G1,1,1
2019_ICLargoCocconi4F_G5,1,0.3
2019_ICLucaDellaRobbia_G4,0,18
2018_ICAlighieri5,0,8
2018_ICAlighieri3,1,1.3
2019_IC_Leopardi_Gualdo_g4,1,2.5
2019_TALENTcampJesi_g2,1,0
2019_TALENTcamp_Osimo_g1,1,3
2019_IC_Lotto_Jesi_sec_g6,1,0
2019_IC_Leopardi_Gualdo_g5,1,0.5
2018_LiceoGalilei7,1,0.3
2018_LiceoVoltaFellini9G,1,2
2019_IC_Alighieri_G2,1,3.5
2019_ICLargoCocconi5C_G3,1,1
```

Figura 4: Struttura del file con margine di errore

3 Preprocessing

Nella fase di preprocessing, dopo aver importato le opportune librerie, ci siamo soffermati su come manipolare e gestire i nostri dati, in modo tale da avere un output strutturato su cui fare process discovery. L'algoritmo implementato prende in input l'intero dataset, costituto da tutte le esecuzioni, in base alla directory dell'esercizio che si vuole analizzare. Nel dettaglio, esso confronta ogni singola esecuzione con la successiva, separate sempre dal comando "`STOP PROGRAM;`". Nel caso in cui le esecuzioni sono identiche, significa che

non è stata eseguita nessuna variazione sulla programmazione del robot ed è stato rieseguito lo stesso codice. Nel caso contrario, quando le esecuzioni differiscono, sono state implementate diverse funzioni che ci permettono di capire se è stato rimosso o aggiunto un blocco, se è stato cambiato il tipo, se sono aumentati o diminuiti alcuni parametri. In questo modo è possibile costruire un event log che contiene le differenza tra i diversi tentativi compiuti dai ragazzi. Una delle funzioni su cui va posta particolare attenzione la funzione findScore(). Essa è utile per confrontare le varie istruzioni fra due esecuzioni, e intercettare quali sono più simili tra loro in modo da capire quale istruzione è stata modificata. Inizialmente confronta due righe che corrispondono a due esecuzioni, se queste sono uguali attribuisce uno score pari a 100. Altrimenti se cambia il valore in prossimità della prima posizione della riga, significa che è stato sostituito il blocco con un altro. Quando le due esecuzioni differiscono, ma il nome del blocco resta invariato, lo score assegnato aumenterà di una quantità pari a 10. Successivamente si va ad analizzare se compare il simbolo "=" nella medesima posizione, perché questo significherebbe che si tratta di campo un "parameters" anziché "type". In questo caso lo score aumenterà di una quantità pari allo score attuale più il valore assoluto della differenza tra il valore del primo parametro della seconda esecuzione e il valore del primo parametro della prima esecuzione. Questo approccio utilizzato per calcolare lo score viene replicato anche sugli altri parametri rimanenti.

```
def find_score(row1, row2):
    score=0
    if(np.array_equal(row1,row2)):
        score=100
    elif row1[0]!=row2[0]: #if blockname change
        if (row1[0]!="" and row2[0]!=""):
            score=1
        else:
            score=0
    else:
        score+=10 #perchè il blockname è uguale
        #controllo se non ha il "=", altrimenti non è tipo ma parametro
        if(row1[1]==row2[1] and str(row2[1]).find('=')<0):
            score+=10
        if(str(row2[1]).find('=')>0):
            score+=diff_param_value(row1[1],row2[1])

        score+=diff_param_value(row1[2],row2[2])
        score+=diff_param_value(row1[3],row2[3])
        score+=diff_param_value(row1[4],row2[4])
        score+=diff_param_value(row1[5],row2[5])
    return score
```

Figura 5: Funzione per il calcolo degli score

```python
def diff_param_value(par1,par2):
    try:
        score=20
        if(par1!=par2):
            name_param=par1.split(" = ")[0]
            val1=par1.split(" = ")[1]
            val2=par2.split(" = ")[1]

            score-=abs(float(val1)-float(val2))
            if score<0:
                score=0

        return score
    except :
        return 0
```

Figura 6: Funzione per calcolare la differenza tra i valori dei parametri

Questa funzione ci consentirà di andare a popolare una matrice dei punteggi, in cui andremo ad individuare gli indici dell'elemento che rappresenta il valore massimo della matrice. Il valore massimo della matrice identifica quali istruzioni sono più simili tra di loro e trova quindi delle corrispondenze fra le istruzioni dell'esecuzione precedente e quella successiva. Viene implementata un'iterazione in cui si cattura l'elemento con il massimo punteggio nella matrice e i suoi indici rappresentano le istruzioni utilizzate nel confronto. Quindi, l'intera riga e colonna vengono eliminati (per comodità si inserisce il valore -1, in modo che non possano essere riselezionati gli stessi elementi più volte).
Un'altro passo fondamentale del preprocessing è stato quello di introdurre, nella funzione che calcola le differenze tra i parametri, un filtro sul parametro `DISTANCE CM`. Esso prende in considerazione solo i cambiamenti del prima citato parametro solo quando esso differisce tra due esecuzioni diverse di una quantità superiore a 5 cm. Inoltre, sono stati rimossi tutti i duplicati del parametro `Ultrasonic Sensor Compare` perchè questo creava implicitamente dei cicli. A fronte di questa considerazione, abbiamo rimosso anche tutti gli altri cicli. Viene eliminato in tutto il dataset la variabile `CONDITION` perché rappresenta l'esito della condizione e non una attività di programmazione. Infine, per agevolare la fase di process discovery, vengono aggiunte delle attività fittizie tra le diverse esecuzioni denominate `START_CASE_ID` e `END_CASE_ID`. Questo significa che all'inizio e alla fine di ogni esecuzione troveremo queste due istruzioni come riportato nell'esempio in figura 7.

```
ExA_piccoli_2018_ICAlighieri3, 0, START CASE_ID,,,,,,
ExA_piccoli_2018_ICAlighieri3, 0, Add blockname MoveSteering, MoveSteering, OnForSeconds, Seconds = 5, Speed = 50, Steering = 0, MotorPorts = 123
ExA_piccoli_2018_ICAlighieri3, 3, Decrease Seconds, MoveSteering, OnForSeconds, Seconds = 3.5, Speed = 50, Steering = 0, MotorPorts = 123
ExA_piccoli_2018_ICAlighieri3, 4, Increase Seconds, MoveSteering, OnForSeconds, Seconds = 4, Speed = 50, Steering = 0, MotorPorts = 123
ExA_piccoli_2018_ICAlighieri3, 5, Increase Seconds, MoveSteering, OnForSeconds, Seconds = 4.1, Speed = 50, Steering = 0, MotorPorts = 123
ExA_piccoli_2018_ICAlighieri3, 26, Increase Seconds, MoveSteering, OnForSeconds, Seconds = 4.15, Speed = 50, Steering = 0, MotorPorts = 123
ExA_piccoli_2018_ICAlighieri3, 28, END CASE_ID,,,,,,
```

Figura 7: Esempio di esecuzione con aggiunta delle attività fittizie

L'output ottenuto dalla fase di preprocessing è suddiviso in tre file:

- All.csv: contiene tutte le esecuzioni presenti nell'intero dataset;
- Good.csv: contiene tutte le esecuzione che hanno raggiunto l'obiettivo;
- Wrong.csv: contiene tutte le esecuzioni che non hanno raggiunto l'obiettivo.

Questa suddivisione viene eseguita a partire dal file degli esiti "y_tesi_completo_exA.csv" per l'esercizio A, e il file "y_tesi_completo_exB.csv" per l'esercizio B. In questi file per ogni gruppo è riportato nella colonna

"Y" se il gruppo ha raggiunto o meno l'obiettivo. Quindi al momento della scrittura in output della traccia di esecuzione, si esegue prima un controllo sul nome del gruppo all'interno di questo file, discriminando cosi in quale categoria appartiene. Tutti gli output avranno la stessa intestazione composta dai seguenti campi:

Nome campo	**Descrizione**
Case_Id	Indica il tipo di esercizio e il gruppo scolastico
Timestamp	Indica l'istante in cui viene compiuta l'activity
Activity	Indica il tipo di attività compiuta
Blockname	Indica il tipo di blocco
Type	Indica il tipo di azione sul blocco
1st-param	Il primo parametro su cui si va ad agire
2st-param	Il secondo parametro su cui si va ad agire
3st-param	Il terzo parametro su cui si va ad agire
4st-param	Il quarto parametro su cui si va ad agire

Tabella 2: Descrizione del dataset

Per quanto riguardo il campo Timestamp, nel dataset non è presente un vero e proprio timestamp che scandisce l'ordine delle esecuzioni. E' stato ritenuto opportuno costruire un indicatore di tempo che ci permettesse di schedulare le istruzioni all'interno di uno stesso Case_Id. Questo significa che quando processiamo un Case_Id diverso, il timestamp ripartirà da zero. Dallo screenshot sopra riportato si può notare che il timestamp è presente ad intervalli, ciò perché in quell'arco temporale vengono eseguite una serie di "no-action", ovvero i ragazzi hanno ripetuto la stessa esecuzione sul robot. L'approccio adottato ci ha permesso poi di avere una chiara visione sulla successione delle attività e sui tentativi impiegatati dai vari gruppi nel raggiungere l'obiettivo.

Una possibile variante, è quella di effettuare una sorta di "Roll up" delle azioni di "Increase" e "Decrease" dei parametri, per avere una minore complessità del log e di conseguenza del modello. In pratica siamo andati a sostituire, nei file di output, le variazioni "Increase" e "Decrease" dei parametri con "Change" seguito dal nome del parametro che cambia.

4 Process discovery

Nella fase di process discovery, a partire dai dati ottenuti in output dalla fase di preprocessing, si cerca di identificare un modello di processo. Per fare questo sono stati effettuati due tentativi, uno utilizzando la libreria pm4py (scaricabile al seguente link pm4py.fit.fraunhofer.de) e l'altro utilizzando il software Disco (scaricabile al seguente link: fluxicon.com/disco)

4.1 Utilizzo della libreria pm4py

Per utilizzare la libreria pm4py è stato sviluppato uno script in linguaggio python. A seguito dell'importazione di librerie fondamentali (come pandas, numpy e tutte quelle riguardanti a pm4py) sono state definite delle variabili globali che definiscono gli argomenti di esecuzione del software:

- `input_path`: variabile in cui viene inserito il percorso del file di input come stringa. Per specificare più file, si può utilizzare anche il carattere jolly "*", ad esempio per processare tutti i file della carella "input" si può scrivere: "input/*.csv".
- `output_path`: variabile in cui viene inserito il percorso della cartella di output come stringa.

Come primo step, viene letto il file csv e formattato il dataframe con lo standard di pm4py. Poi è stato prodotto un file xes di log, un particolare file xml che descrive l'event log. A questo punto, utilizzando

l'algoritmo inductive miner, sono stati realizzati 9 modelli di processo diversi incrementando con step di 0.1 il parametro "NOISE_THRESHOLD".

```
for i in range (0,number_step,1):
    noise=i/10+0.1
    #applico l'algoritmo inductive miner con i diversi noise threshold
    net, initial_marking, final_marking = inductive_miner.apply(log, variant=inductive_miner.Variants.IMf,
                                                                parameters={inductive_miner.Variants.IMf.value.Parameters.NOISE_THRESHOLD: noise})
    nets.insert(i,net)
    im.insert(i,initial_marking)
    fm.insert(i,final_marking)
    path=output_dir+"/images_net/"+group_name
    os.makedirs(path, exist_ok=True)
    path+="/noise "+str(round(noise,2)).replace(".",",") +".png"
    #salvataggio immagine rete di petri sul percorso definito
    pm4py.save_vis_petri_net(nets[i], initial_marking, final_marking, path)
```

Figura 8: Ciclo di esecuzione dell'algoritmo di mining con i diversi noise threshold

I diversi modelli ottenuti sono stati memorizzati come foto delle reti di petri all'interno della cartella "images_net" nel percorso di output. In figura 9 è riportato un esempio di modello di processo come petri net ottenuto per un determinato noise threshold.

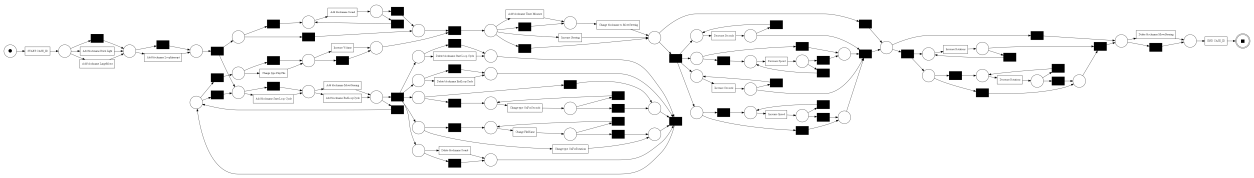

Figura 9: Modello di processo come rete di petri ottenuto con pm4py

Successivamente, a partire dai diversi modelli sono state calcolate le metriche di fitness, precision, generalization, simplicity e sound.

```
def calcola_metriche(net, noise, log, initial_marking, final_marking, return_dict,i):
    metrica={}
    metrica['noise']= round(noise,2)
    metrica['fitness'] = replay_fitness_evaluator.apply(log, net, initial_marking, final_marking, variant=replay_fitness_evaluator.Variants.ALIGNMENT_BASED)["averageFitness"]
    metrica['precision'] = precision_evaluator.apply(log, net, initial_marking, final_marking, variant=precision_evaluator.Variants.ALIGN_ETCONFORMANCE)
    metrica['generalization'] = generalization_evaluator.apply(log, net, initial_marking, final_marking)
    metrica['simplicity'] = simplicity_evaluator.apply(net)
    metrica['is_sound'] = woflan.apply(net, initial_marking, final_marking, parameters={woflan.Parameters.RETURN_ASAP_WHEN_NOT_SOUND: True,
                                                                                        woflan.Parameters.PRINT_DIAGNOSTICS: False,
                                                                                        woflan.Parameters.RETURN_DIAGNOSTICS: False})
    return_dict[i]= metrica
```

Figura 10: Funzione di calcolo delle metriche

Una volta avviato lo script, si può notare da subito come il calcolo della fitness sia molto impegnativo in termini di risorse e quindi anche in tempo di esecuzione. Per migliorare le performance è stato implementato il multithreading che ha consentito di elaborare le metriche in tempi molto più rapidi, quasi 5 volte più veloce rispetto all'esecuzione single-thread (il test è stato effettuato su un processore Intel I5 9600K, con 6 core). Questo ha portato notevoli vantaggi, perché un processo dalla durata di 20 ore in single-thread, nella nostra configurazione impiega solo 4 ore potendo eseguire molti più processi contemporaneamente.

```
manager = multiprocessing.Manager()
return_dict = manager.dict()
p=[]
#eseguo un thread per il calcolo delle metriche per ogni step di noise threshold
for i in range (0,number_step,1):
    p.insert(i, Process(target=calcola_metriche, args=(nets[i],i/10+0.1, log, im[i], fm[i], return_dict, i)))
    p[i].start()

#attendo che i vari thread terminino
for i in range (0,9,1):
    p[i].join()
```

Figura 11: Implementazione multithreading per il calcolo delle metriche

Infine l'output viene memorizzato su un file csv, costruendo una tabella con le metriche calcolate per ogni processo con ogni noise threshold, come riportato in figura 12.

Noise	Fitness	Precision	Generalization	Simplicity	Sound
0,1	0,99	0,18	0,61	0,6	True
0,2	0,93	0,43	0,59	0,61	True
0,3	0,93	0,43	0,6	0,62	True
0,4	0,88	0,63	0,53	0,63	True
0,5	0,87	0,64	0,54	0,63	True
0,6	0,72	0,51	0,58	0,66	True
0,7	0,66	0,51	0,55	0,67	True
0,8	0,63	0,51	0,57	0,67	True
0,9	0,67	0,53	0,6	0,66	True

Figura 12: Tabella in output per il calcolo delle metriche

A partire da questi valori, sono stati poi realizzati dei grafici per comprendere meglio l'andamento delle metriche al variare del noise threshold, in modo da selezionare il modello che rappresenta il miglior compromesso fra questi valori.

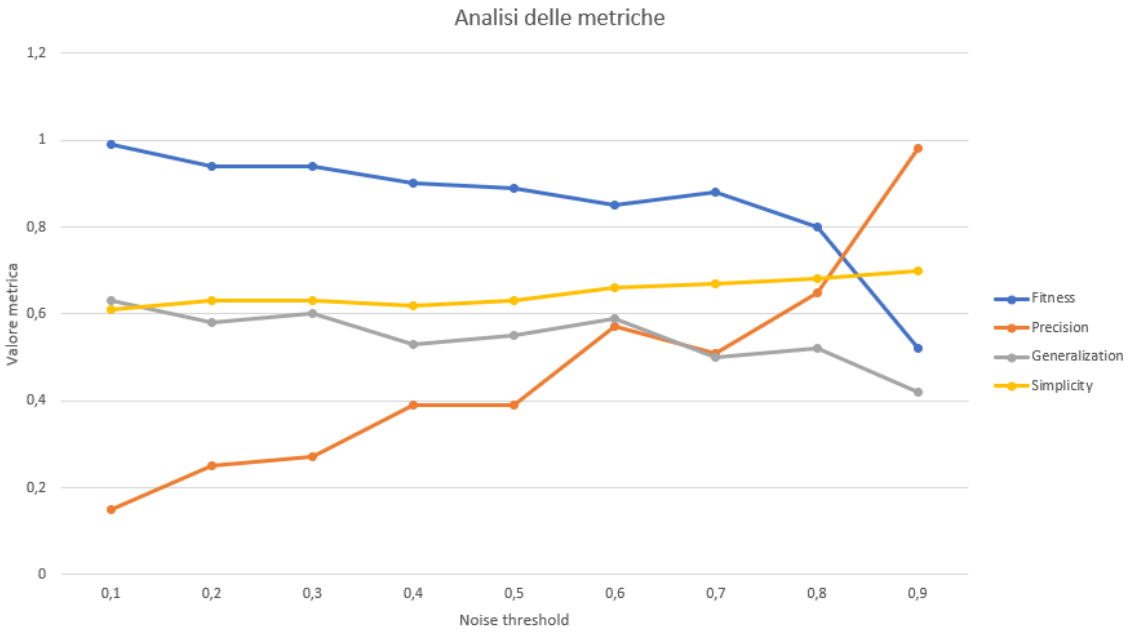

Figura 13: Grafico sull'andamento delle metriche al variare del noise threshold

4.2 Utilizzo del software Disco

Per l'implementazione dell'algoritmo Fuzzy Miner è stato utilizzato il software Disco. Questo software è stato progettato per importare i dati in modo semplice ed intuitivo, rilevando automaticamente i timestamp e attività consentendo di caricare i dati molto velocemente.

Figura 14: Logo del software Disco

Inizialmente, vengono importati i diversi file di output del preprocessing in formato csv, come visibile nella figura 15.

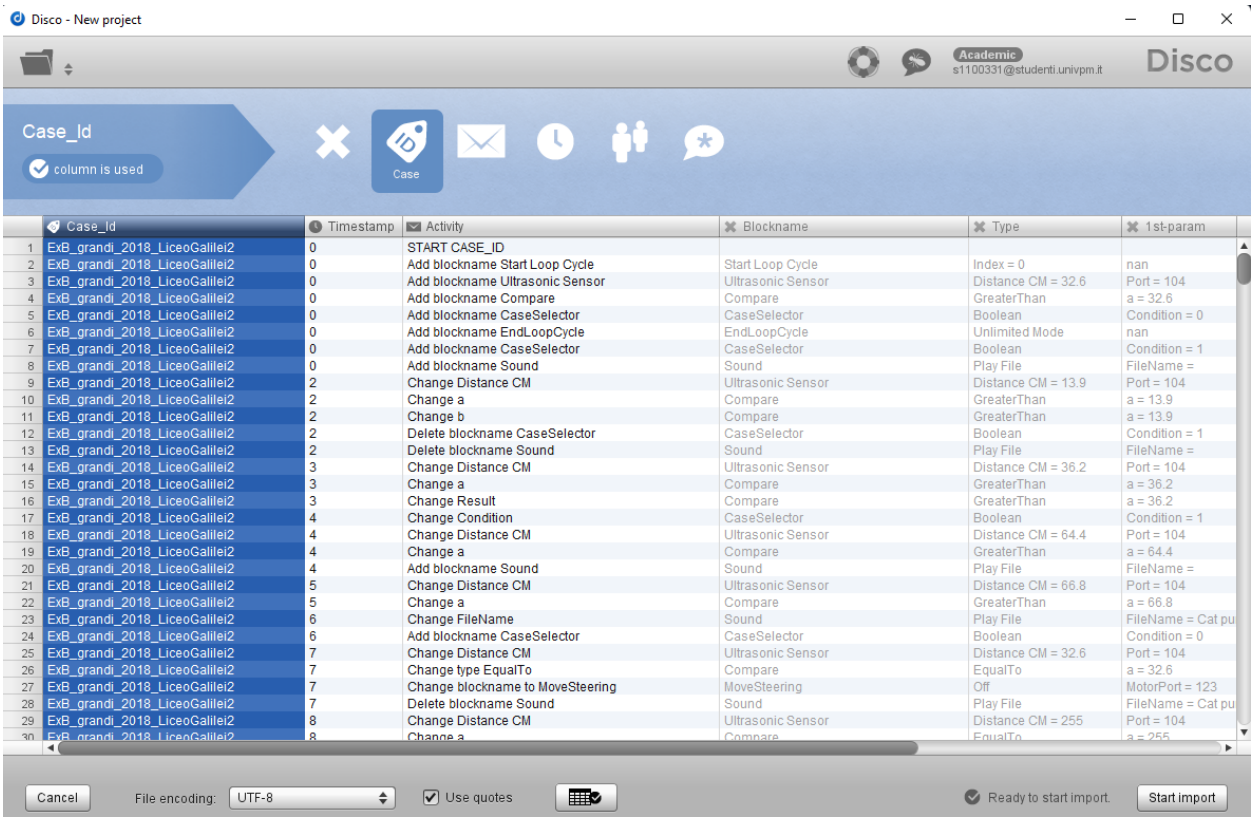

Figura 15: Finestra di importazione del file csv su disco

Dopo aver controllato di aver importato tutti i dati correttamente, viene settato il timestamp. In particolare, Disco permette di definire un pattern del timestamp, attraverso cui va a schedulare le diverse azioni. Definire il pattern significa specificare il formato dell'orario e l'unità con cui vengono sequenzializzate le azioni, ovvero secondi (s), minuti (m), etc. Questo permette di avere in output un modello di processo che identifica i tentativi che vengono effettuati tra una azione e l'altra.

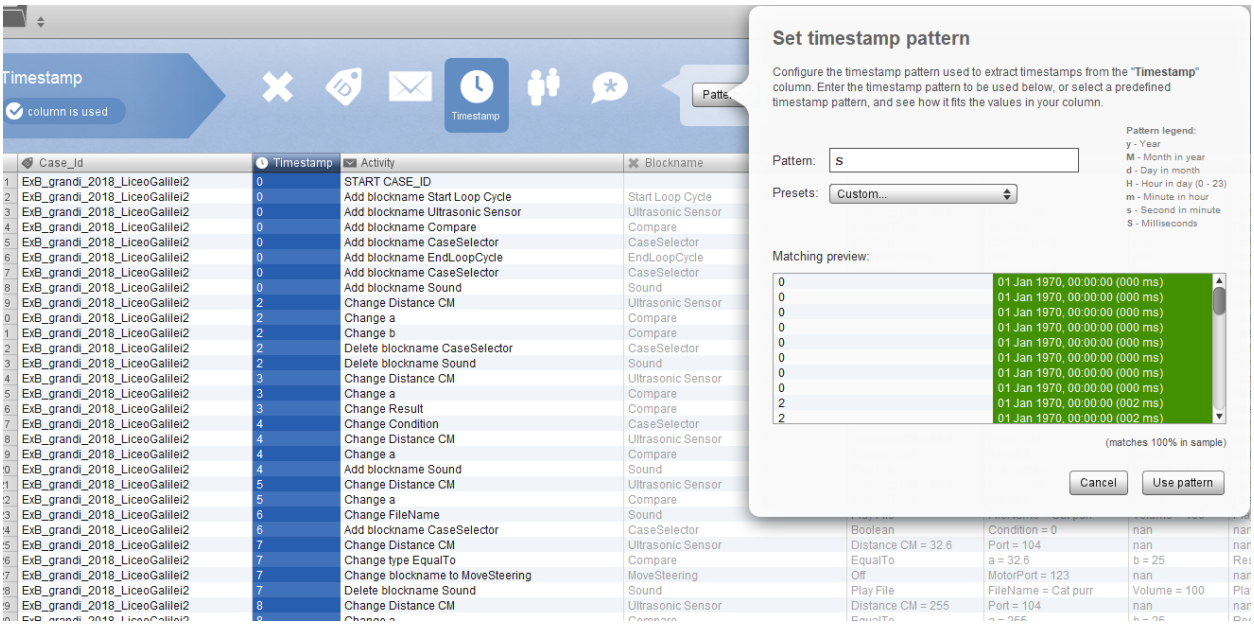

Figura 16: Impostazione del pattern del timestamp

Infine, confermando l'importazione mediante il pulsante in basso a destra si visualizzerà il modello di processo ottenuto attraverso l'algoritmo Fuzzy miner come riportato in figura 17. Sulla destra si trovano i parametri che il software utilizza per generare il modello, ovvero la percentuale di attività e percorsi che si vogliono visualizzare, filtrando tutto il resto in base alla frequenza con cui compaiono questi elementi sull'intero dataset. Il software Disco imposta già questi parametri con dei valori predefiniti auto calcolati, ma si può interagire in tempo reale per valutare il giusto compromesso tra leggibilità e completezza del modello. Sotto i parametri discussi vi è anche la possibilità di specificare come deve essere espressa la frequenza delle attività e percorsi, ad esempio se con il numero assoluto o con la percentuale.

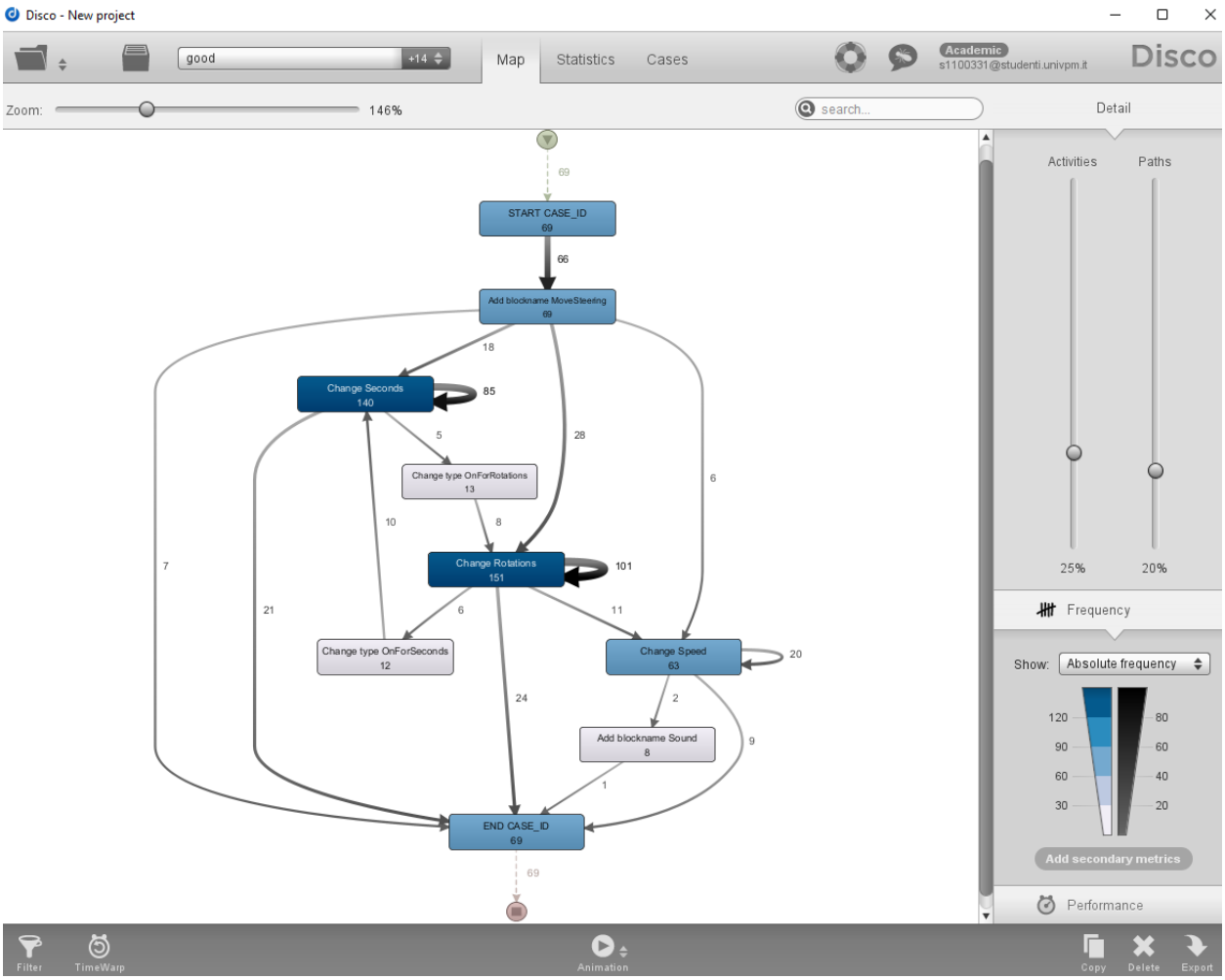

Figura 17: Modello di processo ottenuto da Disco

La figura 18 mostra la schermata performance del modello di processo. Dal menu in basso a destra è possibile scegliere quale tipo di performance si vuole mostrare, ad esempio la durata totale, la durata media o la durata massima e minima di ogni azione presente nel modello. Questo evidenzierà, in questo caso, le azioni con una durata totale notevole e quelle che hanno via via valori meno significativi. Inoltre, Disco permette anche di aggiungere un'ulteriore metrica che si vuole valutare.

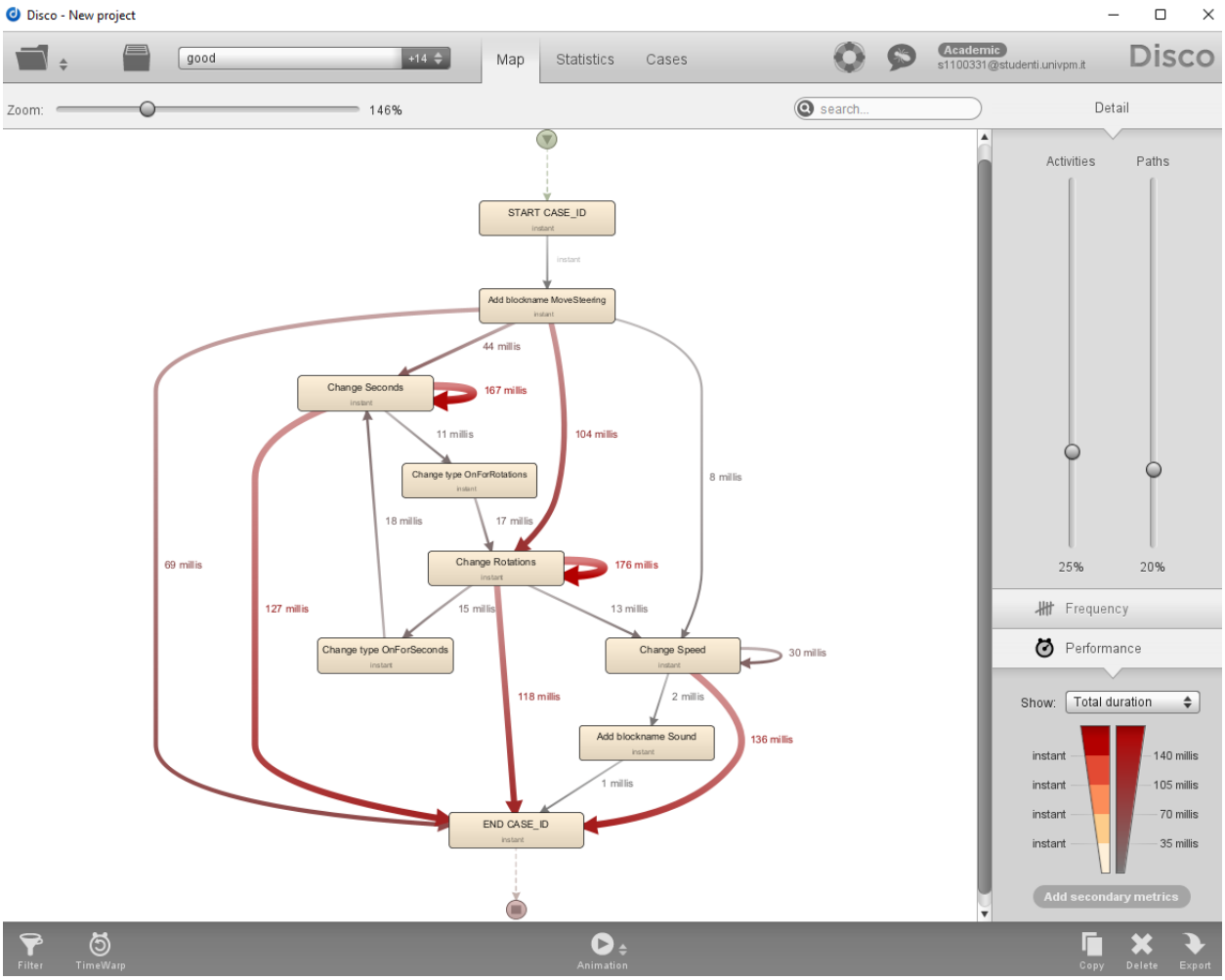

Figura 18: Vista performance di Disco

Nella figura 19 viene visualizzato la schermata raggiungibile mediante la tab "Statistics" del software Disco. Da questa pagina è possibile analizzare le singole attività indipendentemente dal percorso, per capire quante volte queste si verificano a confronto di tutte le altre. E' possibile visualizzare anche tutte le attività iniziali e finali, che in questo caso sono le attività fittizie.

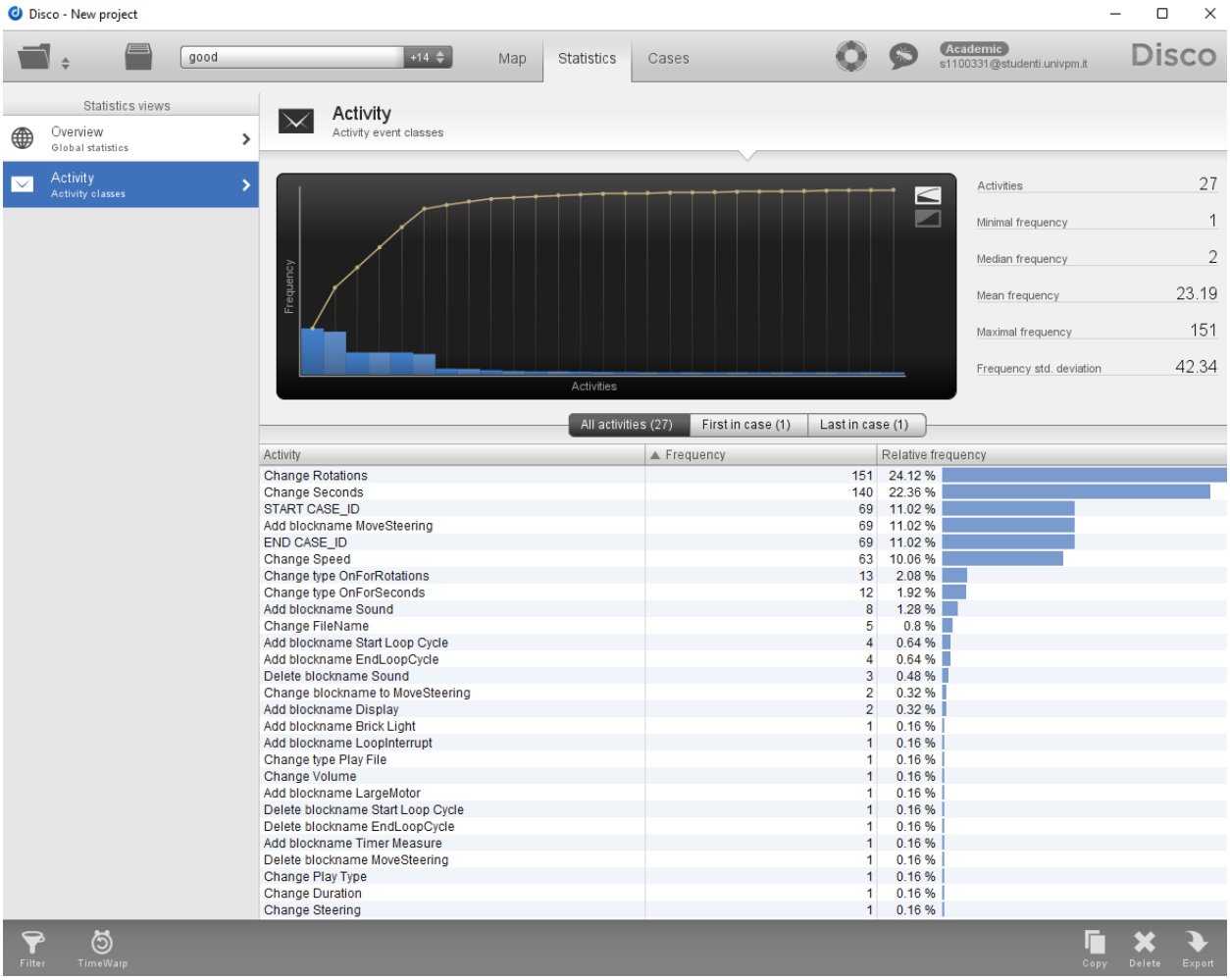

Figura 19: Vista statistics di Disco

Nella figura 20 viene mostrata la vista "Cases" di Disco. Essa mostra, per un particolare "Case_ID", quali sono le azioni compiute. Durante la fase di osservazione del modello, sono emersi alcuni percorsi interessanti da analizzare. Grazie all'opzione di filtraggio su Disco, siamo andati a vedere maggiormente nel dettaglio queste esecuzioni. Questa feature è stata particolarmente utile perché ci ha permesso di visualizzare il comportamento di coloro che hanno raggiunto l'obiettivo, vedendo l'ordine con cui sono state eseguite le azioni e i parametri impostati. Al contempo, ciò ci ha anche consentito di visionare ed interpretare i dati di chi non raggiunto l'obiettivo, cercando di capire le motivazioni di impostazione di certi parametri e l'ordine con cui le azioni venivano eseguite.

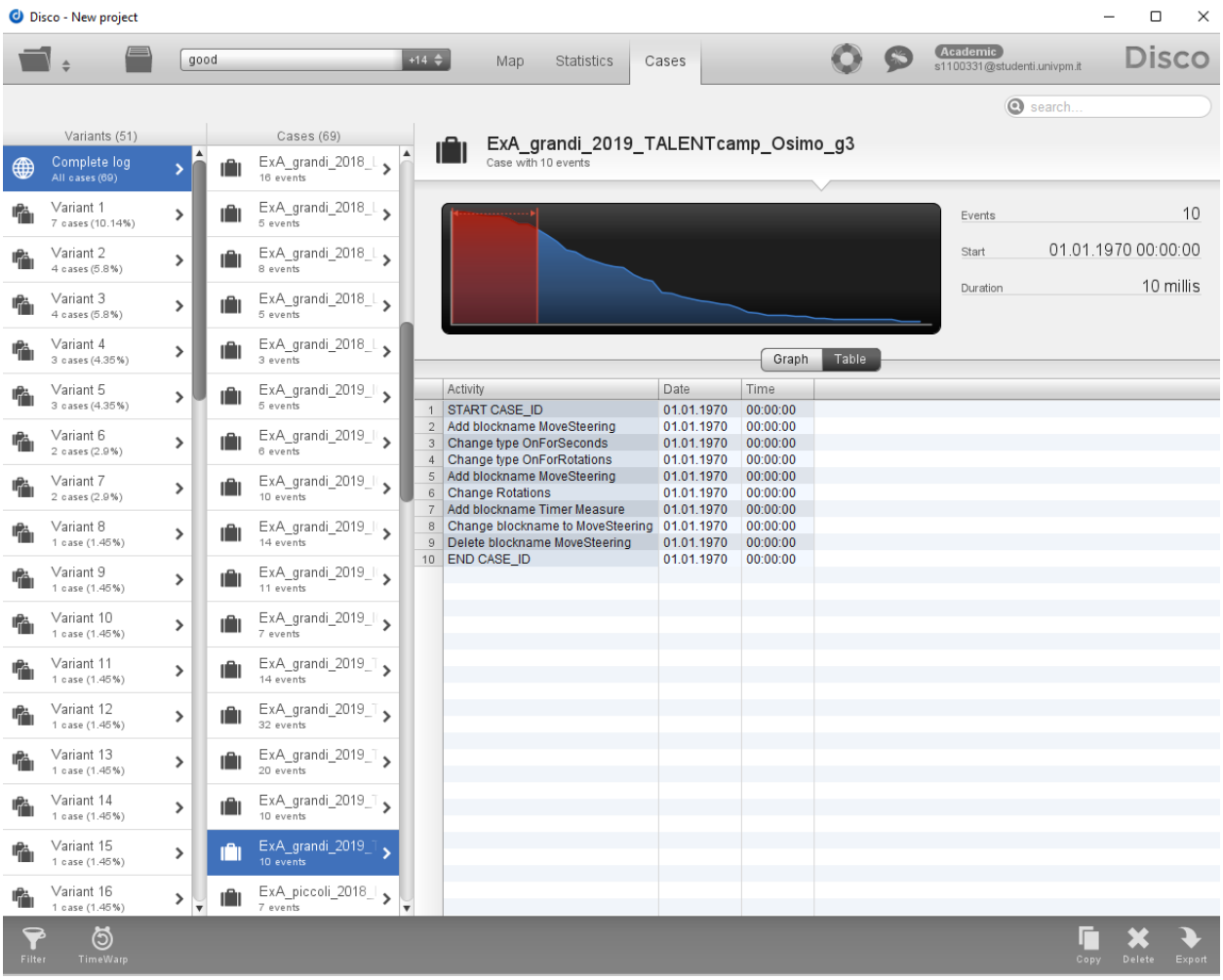

Figura 20: Vista cases di Disco

5 Risultati dell'analisi

In questa sezione verranno analizzati i risultati ottenuti con i diversi algoritmi e i diversi software. Il risultato che ci si aspetta di ottenere da queste analisi è quello di identificare dei pattern di metodologie di programmazione utilizzate per raggiungere lo scopo dell'esercizio. Quindi mediante l'attività di preprocessing il dataset è stato diviso su due gruppi diversi, chi ha raggiunto l'obiettivo e chi non lo ha raggiunto. Ottenendo cosi due dataset diversi, è stata condotta una analisi per ognuno di essi e poi confrontati per evidenziarne le differenze.

5.1 Risultati analisi con pm4py

Come specificato in sezione 4.1, lo sviluppo dello script python con la libreria pm4py e l'algoritmo inductive miner, ci ha consentito di ottenere 9 modelli con diversi noise threshold di filtraggio e le relative metriche. Le metriche ci hanno poi consentito di valutare quale modello scegliere secondo le diverse performance.

5.1.1 Esercizio A

5.1.1.1 Good

Alla fine del processo di data mining eseguito sul log "good" abbiamo ottenuto una cartella contenente i 9 modelli (uno per ogni noise threshold) e un file csv contenente delle metriche per individuare il miglior modello da scegliere. La tabella delle metriche ottenuta è la seguente:

Noise	Fitness	Precision	Generaliz	Simplicity	Sound
0,1	0,97	0,45	0,58	0,63	True
0,2	0,95	0,41	0,56	0,62	True
0,3	0,92	0,46	0,59	0,64	True
0,4	0,91	0,61	0,51	0,65	True
0,5	0,91	0,62	0,52	0,65	True
0,6	0,9	0,64	0,58	0,68	True
0,7	0,9	0,64	0,58	0,68	True
0,8	0,79	0,92	0,53	0,69	True
0,9	0,75	0,95	0,49	0,7	True

Figura 21: Metriche del modello "good" relativo all'esercizio A

Nella tabella vengono riportate le metriche di fitness (calcolata in modalità alignment based), precision (calcolata sempre in modalità alignment), generalization, simplicity e sound. Da questa tabella, è stato realizzato anche un grafico (in figura 22) in cui è molto più chiaro come i valori delle metriche cambiano al variare del noise threshold.

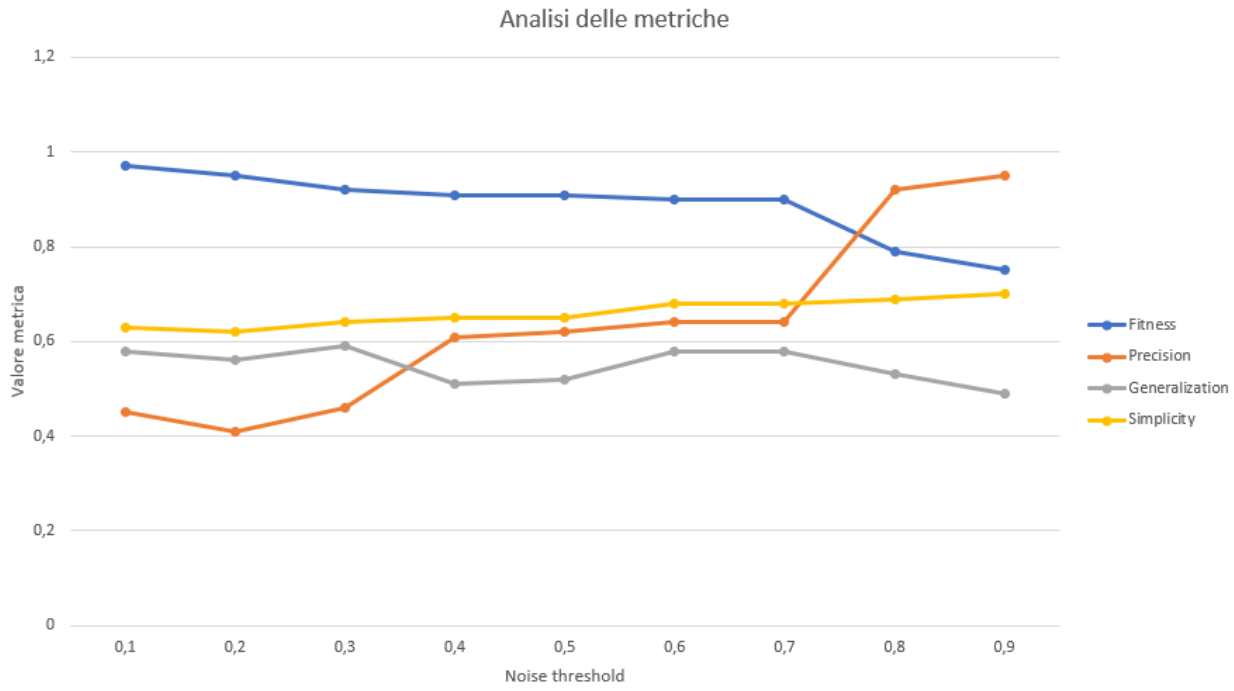

Figura 22: Grafico delle metriche del modello "good" relativo all'esercizio A

A partire da questi valori è stato scelto il modello con il noise threshold pari a 0.7 perché questo modello è quello che ci permette di avere il miglior compromesso tra tutte le metriche. Sempre dalla tabella, si può notare che anche con un valore di noise threshold pari a 0.8, i risultati sulla fitness e sulla precision risultano essere accettabili. Ma, avendo privilegiato la fitness anzichè la precision, la nostra scelta è ricaduta sul valore 0.7. Il modello scelto viene riportato in figura 23 e si può vedere come esso sia comunque abbastanza complesso e ricco di transizioni fittizie (hidden transition).

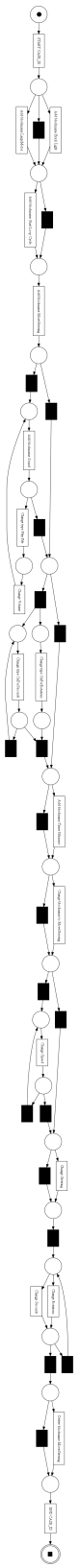

Figura 23: Modello "good" relativo all'esercizio A

Dal modello ottenuto in output possiamo vedere che:

- nella parte iniziale della rete vediamo che non ci sono transizioni fittizie in corrispondenza di "Add Blockname MoveSteering". Questo ci dice che tutti i gruppi hanno sicuramente eseguito quell'azione. Prima di quest'azione, i gruppi potrebbero aver aggiunto blocchi come "Brick Light", "Large Motor" e "Start Loop Cycle".

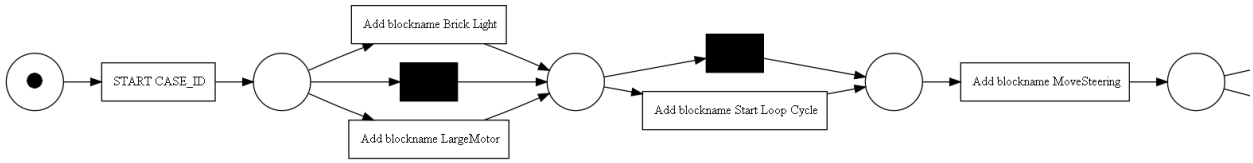

Figura 24: Osservazione sulla prima parte

- successivamente abbiamo due diramazioni, in una di queste viene aggiunto un blocco musicale che non prendiamo in considerazione per la nostra analisi. Invece, sull'altro path si arriva ad un punto in cui è possibile eseguire un cambiamento, il "Change Type OnForRotations" oppure il "Change Type OnForSeconds", entrambi riferiti al blocco "MoveSteering".

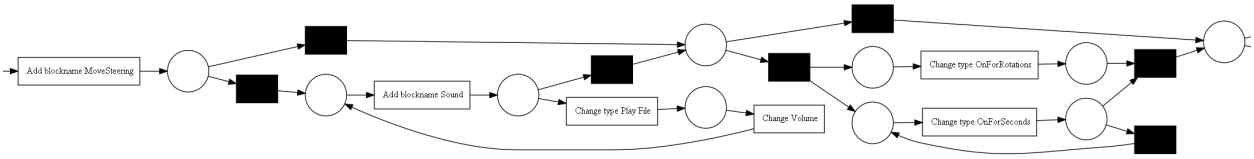

Figura 25: Osservazione sulla seconda parte

- successivamente, esiste la possibilità di effettuare un ciclo su "Change Speed", per poi andare ad eseguire nuovamente un ciclo su "Change Rotations" e "Change "Seconds".

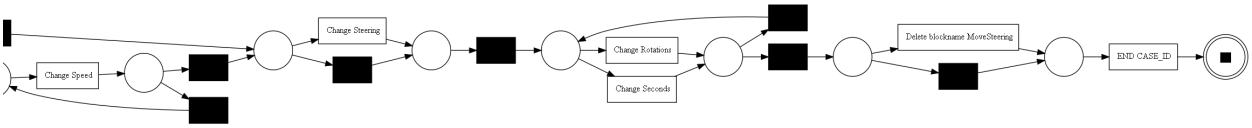

Figura 26: Osservazione sulla terza parte

Da questa analisi quindi emerge che vi sono delle azioni mandatorie che hanno dovuto eseguire tutti in qualche modo. Queste azioni sono: "Add blockname MoveSteering" insieme all'esecuzione di una o entrambe le attività di "Change Rotations" e "Change Seconds".

5.1.1.2 Wrong

Lo stesso procedimento è stato adottato anche per chi non è riuscito a raggiungere l'obiettivo. Quindi le metriche ottenute sono riportate in figura 27.

Noise	Fitness	Precision	Generaliz	Simplicity	Sound
0,1	1	0,41	0,61	0,67	True
0,2	1	0,41	0,61	0,67	True
0,3	0,96	0,47	0,59	0,66	True
0,4	0,96	0,47	0,59	0,66	True
0,5	0,91	0,55	0,58	0,67	True
0,6	0,91	0,55	0,58	0,67	True
0,7	0,86	0,8	0,54	0,68	True
0,8	0,86	0,8	0,54	0,68	True
0,9	0,75	0,49	0,47	0,72	True

Figura 27: Metriche del modello 'Wrong' relative all'esercizio A

Anche in questo caso abbiamo fatto un grafico (in figura 28 per comprendere l'andamento delle metriche al variare del noise threshold.

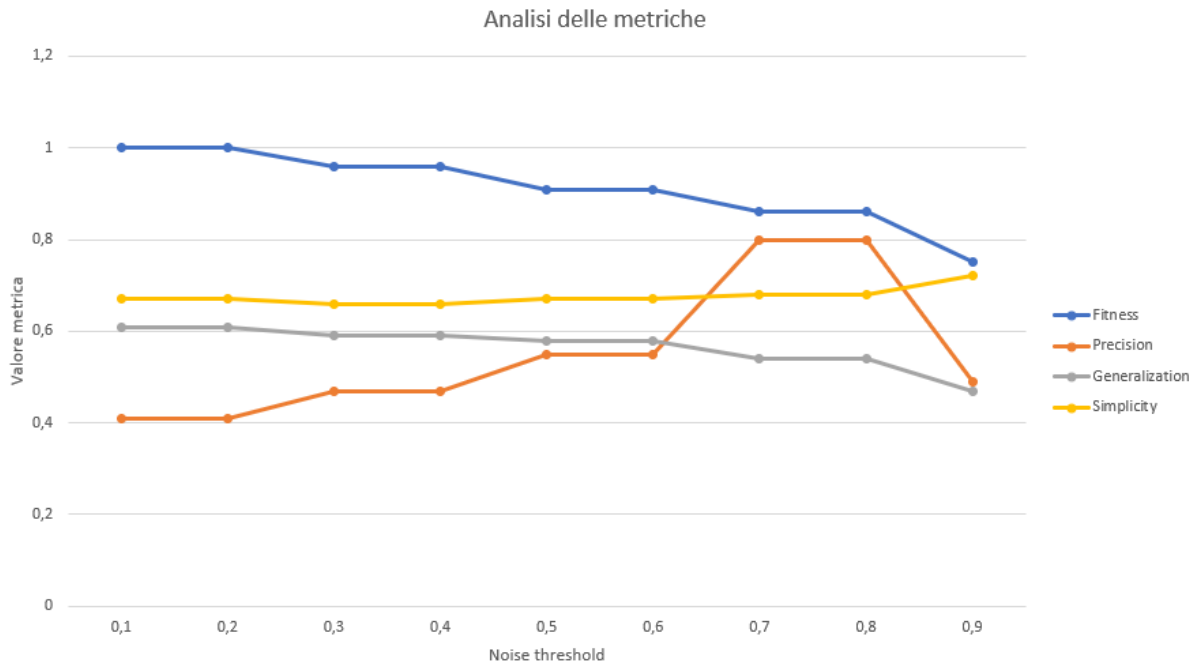

Figura 28: Grafico delle metriche del modello 'Wrong' relativo all'esercizio A

A partire da questi valori è stato scelto il modello con il noise threshold pari a 0.8 perché questo modello è quello che ci permette di avere il miglior compromesso tra tutte le metriche. In questo caso abbiamo accettato il compromesso di perdere 0.05 sulla fitness, ma guadagnare un buona quota, quasi del 50% in più, sulla precision.
Il modello scelto viene riportato in figura 29 e si può vedere come anch'esso sia comunque abbastanza complesso e ricco di transizioni fittizie (hidden transition).

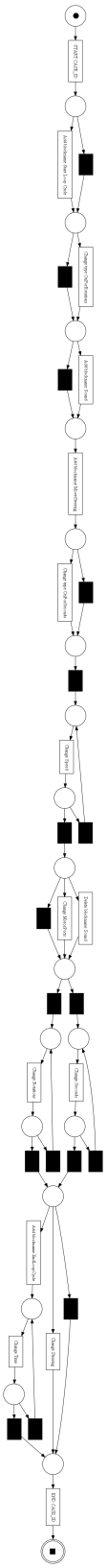

Figura 29: Modello 'Wrong' relativo all'esercizio A

Dal modello ottenuto in output possiamo vedere che:

- nella parte iniziale della rete vediamo che non ci sono transizioni fittizie in corrispondenza di "Add Blockname MoveSteering". Questo, ancora una volta, ci dice che tutti i gruppi hanno eseguito quell'azione. Prima di quest'azione, i gruppi potrebbero aver aggiunto blocchi come "Sound" e "Start Loop Cycle".

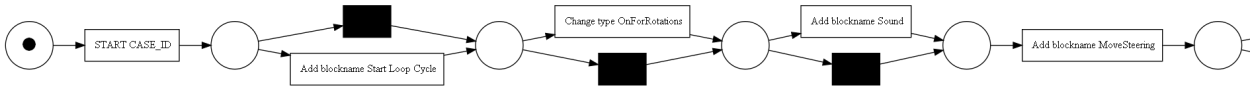

Figura 30: Osservazione sulla prima parte

- proseguendo nella rete, troviamo il possibile cambiamento "Change Type OnForSeconds" sulla modalità di movimento del blocco "MoveSteering". Invece nel modello non compare il cambio di movimento a "OnForRotations". In seguito tutti i gruppi hanno eseguito almeno una volta l'attività "Change Speed", avendo anche la possibilità di eseguire l'operazione più volte.

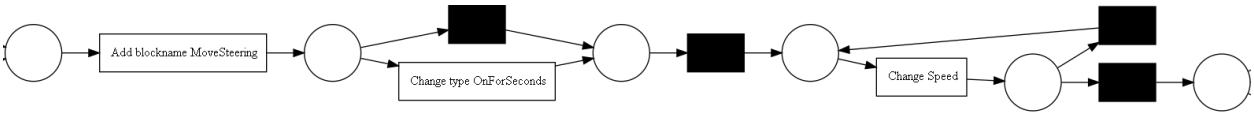

Figura 31: Osservazione sulla seconda parte

- successivamente, devono essere eseguite almeno una delle due attività fra "Change Rotations" e "Change Seconds", in cui per ognuna vi è la possibilità di rieseguire l'operazione più volte.

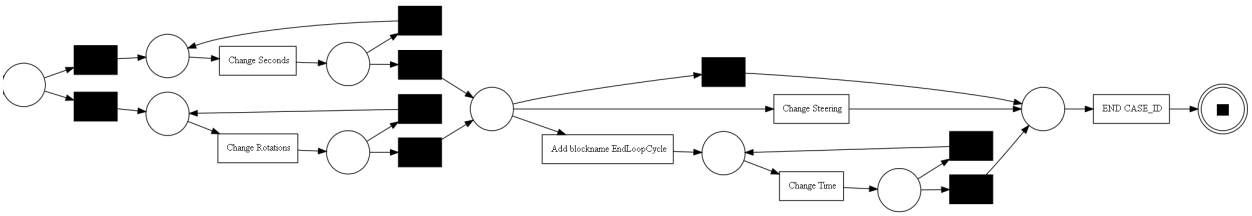

Figura 32: Osservazione sulla terza parte

Da questa analisi quindi emerge che vi sono delle azioni mandatorie che hanno dovuto eseguire tutti in qualche modo. Queste azioni sono: "Add blockname MoveSteering", "Change Speed", insieme all'esecuzione di una tra le attività di "Change Rotations" e "Change Seconds" (potendo comunque effettuare cicli sul parametro che si è scelto da modificare).

5.1.1.3 Confronto

A questo punto dell'analisi andiamo a confrontare i due modelli ottenuti tra chi raggiunge il l'obiettivo e chi no, cercando di evidenziare le principali differenze tra i comportamenti relativi ai gruppi. In particolare, possiamo affermare:

- Nel caso di chi non raggiunge l'obiettivo, a seguito dell'inserimento del blocco "MoveSteering", il modello presenta solamente "Change Type OnForSeconds", mentre nel modello di chi raggiunge l'obiettivo abbiamo la presenza sia di "Change Type OnForSeconds" che "Change Type OnForRotations". Questo potrebbe indicare, nel caso di "wrong", un possibile motivo per cui i gruppi non riescono a completare il task perché viene utilizzata maggiormente la modalità "OnForSeconds".
- Nel caso di "wrong", abbiamo che l'attività "Change Speed" deve essere eseguita obbligatoriamente almeno una volta, mente nel caso di "good" questo non è obbligatorio. Questo potrebbe indicare, nel caso di "wrong", un altro possibile motivo per cui i gruppi non sono riusciti a raggiungere l'obiettivo, ovvero che modificando anche la velocità si va a rendere più difficile la regolazione, quando in realtà si potrebbe fissare questo parametro e agire solamente sui secondi.

5.1.2 Esercizio B

Anche per l'esercizio B è stata eseguita l'attività di process discovery attraverso la libreria pm4py. Con il medesimo codice utilizzato anche per l'esercizio A è stato possibile generare i diversi modelli per ogni valore del noise threshold, ma non è stato possibile ottenere nessuna metrica per essi. Inizialmente è stato eseguito un test con l'esecuzione di tutte le metriche, ma dopo circa 6 ore di esecuzione il caricamento non è arrivato nemmeno all'1% per un singolo modello e questa operazione deve essere eseguita per un totale di 18 modelli. Per questo motivo è stato interrotto il processo perché l'esecuzione avrebbe richiesto delle tempistiche non accettabili. Successivamente è stato eseguito un nuovo tentativo, escludendo questa volta il calcolo della fitness (che dai test precedenti sull'esercizio A è stata quella più impegnativa), ma mantenendo comunque tutte le altre metriche (precision, generalization, simplicity e sound). Anche in questo caso è stato constatato che il tempo di esecuzione non era accettabile (dopo circa 12 ore è stato eseguito il calcolo per 3 modelli, quindi per eseguirne 18 dovrebbe impiegare 2-3 giorni di elaborazione continua).
Osservando comunque i modelli ottenuti, visibili nelle figure 33 e 34 possiamo supporre i motivi di questa problematica:

- Cicli: all'interno del modello troviamo cicli il quale a loro volta presentano altri sotto cicli. Inoltre, osservando ad alto livello, già il numero totale di cicli che compaiono costituiscono una complicazione notevole. A testimoniare questo fatto, in qualcuno di essi abbiamo riscontrato la presenza di circa 40 cicli.
- Hidden transition: nel modello, rispetto al numero di attività, vi è un gran numero di hidden transition. Questo aspetto, sebbene in minima parte, porta comunque ad avere un modello più variabile e quindi più complesso.
- Complessità del modello: la presenza combinata di un elevata quantità di cicli ricorsivi e hidden transition aumenta notevolmente la complessità generale del modello. In particolare, nell'eseguire algoritmi di calcolo delle metriche, ad esempio eseguire l'alignment based replay, devono essere testati tutti i possibili path. Quindi, in presenza di cicli ricorsivi, i path possibili aumentano a dismisura, rendendo l'operazione ad elevata complessità computazionale.

5.1.2.1 Good

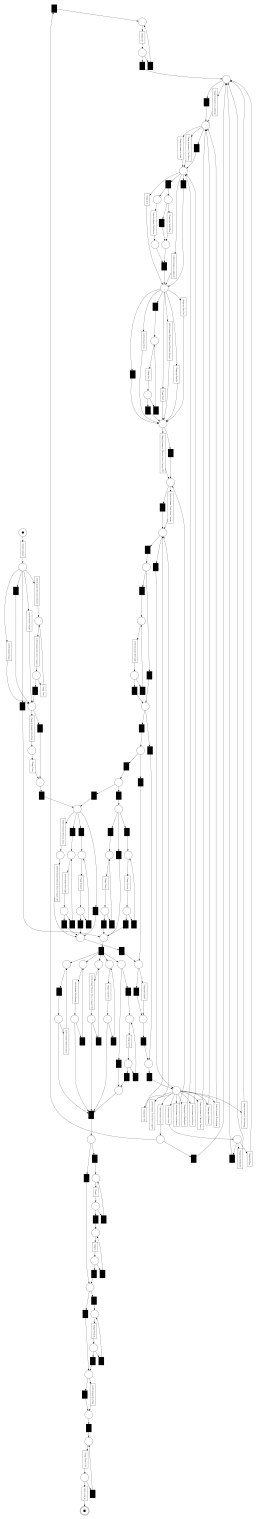

Figura 33: Modello "good" relativo all'esercizio B

5.1.2.2 Wrong

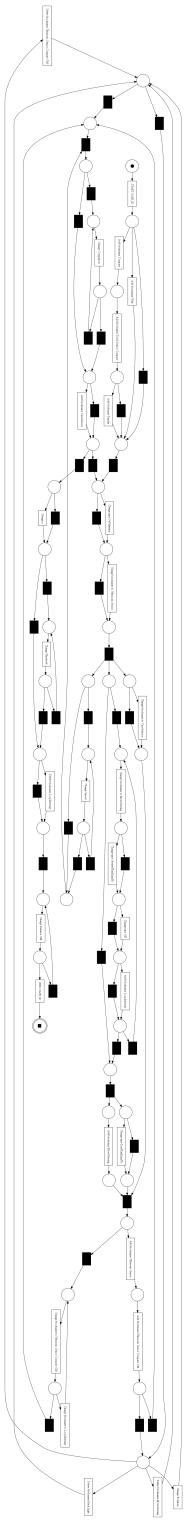

Figura 34: Modello "good" relativo all'esercizio B

5.2 Risultati analisi con Disco

5.2.1 Esercizio A

5.2.1.1 Good

Per eseguire questo tipo di analisi sul primo esercizio, Disco viene impostato con un valore di activities pari al 25% e un valore di paths pari al 20%.

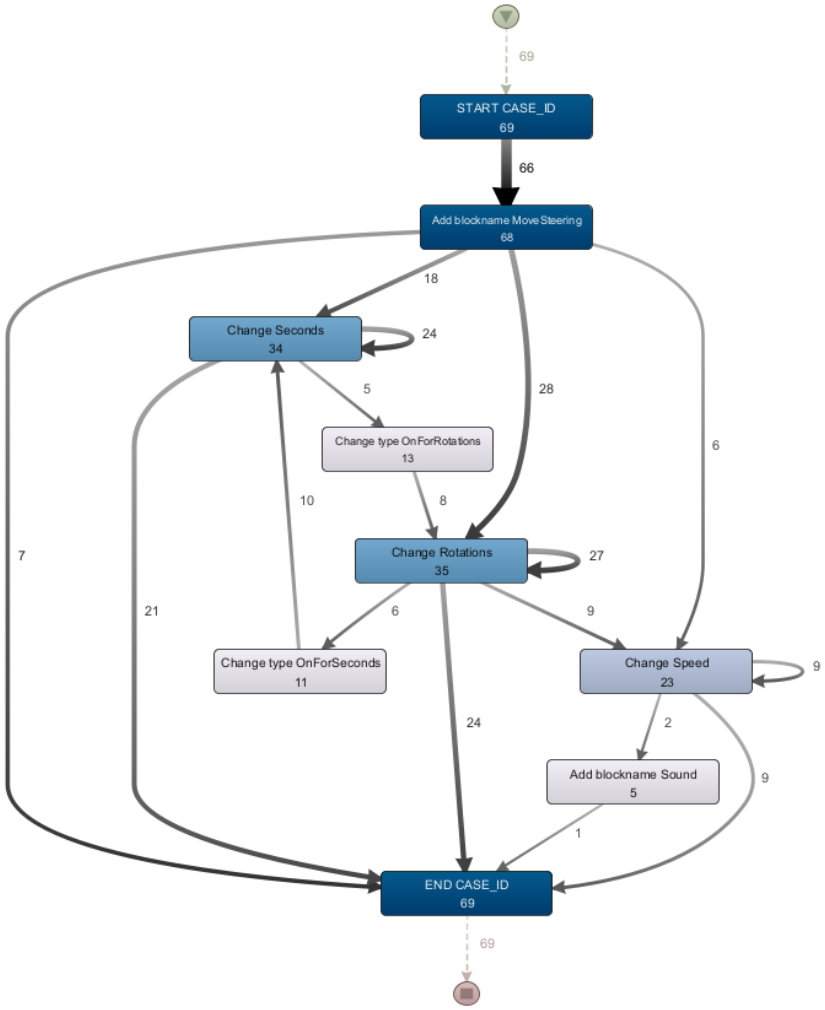

Figura 35: Modello good esercizio A

Osservando il modello di processo prodotto da coloro che hanno raggiunto l'obiettivo, si evince:

- Sette gruppi hanno raggiunto immediatamente l'obiettivo. In particolare essi iniziano l'esecuzione con l'aggiunta del blocco MoveSteering, sei gruppi impostano il tipo come OnForRotations mentre solamente un gruppo utilizza il tipo OnForSecond. Successivamente il primo parametro, in tutti e sette i gruppi, viene impostato con un valore che varia tra 5,6 e 5,9. Tra i sette gruppi varia il parametro legato alla velocità, ma questo non influisce sul risultato.
- Attraverso il case coverage, notiamo che 66 gruppi su 69 aggiungono come primo blocco il blocco MoveStering. Di questi, il 26% dei gruppi, decide di agire sui secondi, mentre il 40% dei gruppi decide di agire sul parametro Rotations e solamente l'8% dei gruppi inizia cambiando la velocità. Fra

quelli che agivano sui secondi, il 15% decide di cambiare il tipo in OnForRotations. Tra quelli che agivano sul parametro Rotations, il 17% ha cambiato modalità a OnForSeconds. Il 34% dei gruppi ha concluso l'esecuzione con un aggiustamento del parametro Rotations, il 30% invece ha concluso con l'aggiustamento del parametro Seconds e infine il 13% ha concluso con la regolazione del parametro Speed.

- Abbiamo notato che i gruppi il quale agiscono sui secondi e che cambiamo il parametro Speed sono in minoranza rispetto a quelli che agiscono sempre sul parametro Speed passando per il cambiamento delle rotazioni.
- Attraverso la vista sulle performance, è possibile notare che a seguito dell'attività di cambiamento della velocità vengono effettuati in media 15 tentativi di esecuzione prima della conclusione dell'esercizio. In generale si può comunque affermare che la maggior parte dei tentativi vengano effettuati prima del termine della prova, come è anche normale intuire. E' interessante notare che a seguito dell'aggiunta del blocco MoveSteering vengano effettuati subito diversi tentativi prima della conclusione.

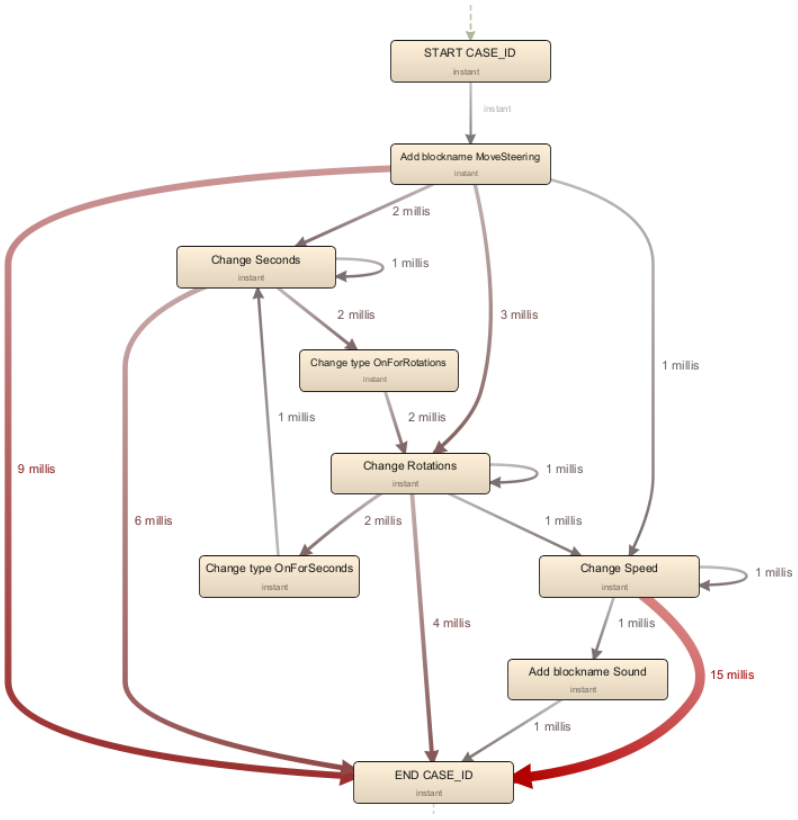

Figura 36: Vista performance good esercizio A

5.2.1.2 Wrong

Anche per eseguire l'analisi su gruppi che non hanno raggiunto l'obiettivo del primo task, Disco viene impostato con un valore di activities pari al 25% e un valore di paths pari al 40%.

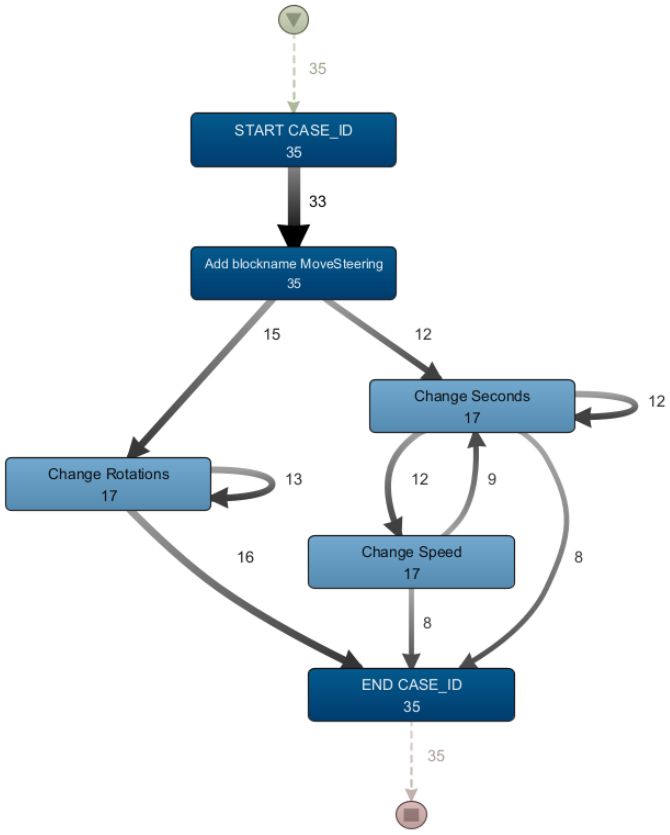

Figura 37: Modello wrong esercizio A

Osservando il modello di processo prodotto da coloro che non hanno raggiunto l'obiettivo, si evince:

- i gruppi si possono chiaramente suddividere in due categorie quasi equipollenti, ovvero chi decide di cambiare le rotazioni e chi decide di cambiare i secondi. Fra coloro che agiscono sui secondi, una buona parte decide di cambiare anche il parametro Speed. Questo può significare che il medesimo gruppo sbagli l'esecuzione, non raggiungendo l'obiettivo purché la regolazione di due parametri contemporaneamente è più difficile rispetto alla regolazione di un solo parametro.
- Utilizzando la visualizzazione del frequenza assoluta, è possibile notare che il numero di volte in cui vengono modificati i secondi è maggiore, di circa un terzo, del numero di volte in cui vengono modificate le rotazioni.
- Utilizzando la vista sulle performance riportata in figura 38, notiamo che, in media, vengono eseguiti quasi lo stesso numero di tentativi sia dopo il cambio delle rotazioni che il cambio dei secondi.

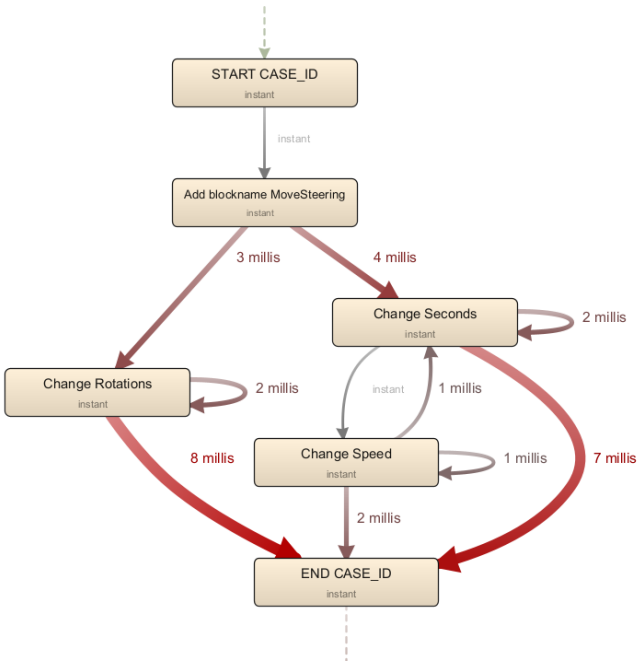

Figura 38: Vista performance wrong esercizio A

5.2.1.3 Confronto

A questo dell'analisi andiamo a confrontare i due modelli ottenuti, ovvero tra chi raggiunge l'obiettivo e chi no, per evidenziare le differenze tra i comportamenti. In particolare, possiamo affermare:

- Mentre su "wrong", il cambiamento della velocità viene eseguito in parallelo con il cambiamento dei secondi, in "good", il cambiamento della velocità avviene in parallelo al cambiamento delle rotazioni. Mentre nella modalità del movimento in secondi, la velocità è un parametro significativo, nella modalità del movimento in rotazioni questo non lo è. Questo si traduce in una maggiore difficoltà di perseguire la giusta regolazione del robot se si utilizzano due parametri piuttosto che uno, al fine di raggiungere l'obiettivo prefissato.

- Mentre caso di "wrong" abbiamo una divisione più equa tra chi agisce sulle rotazioni e sui secondi, nel caso di "good" ci troviamo di fronte ad una suddivisione dei gruppi avente un divario più significativo. Quindi, è possibile dedurre, che è più semplice raggiungere l'obiettivo regolando le rotazioni del robot piuttosto che i secondi.

- Dalla vista delle performance, è possibile notare che nel caso di "wrong" il numero dei tentavi totali prima di giungere alla conclusione è piuttosto omogeneo sulla regolazione dei secondi e rotazioni, mentre nel caso di "good" numero dei tentativi eseguiti dopo il cambiamento della velocità è molto più alto rispetto agli altri parametri.

5.2.2 Esercizio B

Anche con Disco abbiamo analizzato i due file principali Good e Wrong per l'esercizio B.

5.2.2.1 Good

In questo caso, Disco viene impostato con i valori di activity e path differenti. In particolare, nel caso di "good" avremo un valore di activity pari al 14% e un valore di path pari al 8%. Queste considerazioni vengono riportate avendo a disposizione 78 casi su cui effettuare analisi.

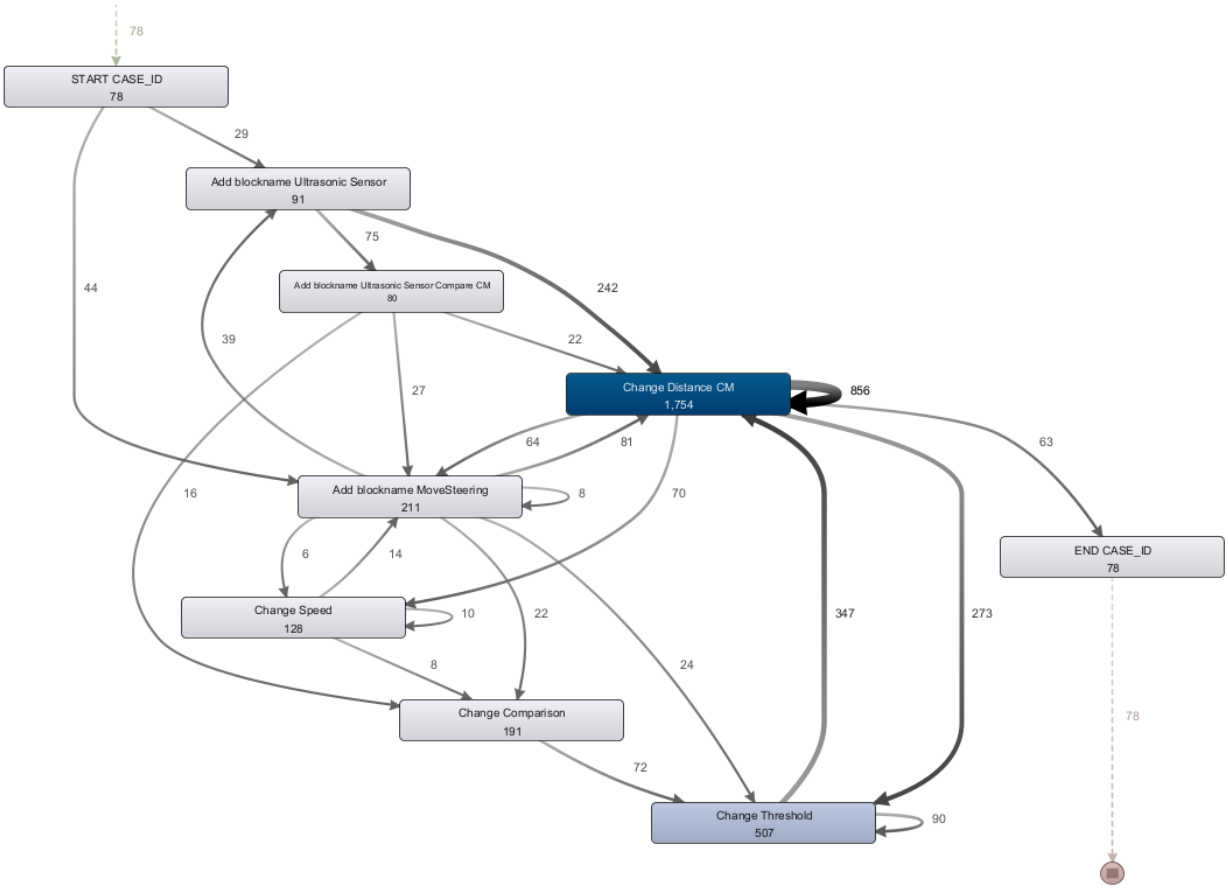

Figura 39: Modello good esercizio B

Osservando il modello di processo prodotto da coloro che hanno raggiunto l'obiettivo (Good.csv) si evince:

- nella parte iniziale, circa il 50% rispetto al totale dei casi, aggiunge il blocco "MoveSteering". In seguito, la maggior parte di questi aggiungono il blocco "Ultrasonic Sensor" e successivamente poi il blocco "Ultrasonic Sensor compare CM".
- successivamente, si può notare una divisone abbastanza equa tra chi decide di modificare il parametro "Threshold" e chi decide di modificare il parametro "Comparison".
- si può notare la presenza di un ciclo tra "Add BlockName MoveSteering" e "Change Speed". Abbiamo cercato di analizzare questa anomalia e ci siamo accorti che 13 gruppi aggiungono un nuovo blocco "MoveSteering" impostando il parametro type ad "Off", perché in precedenza hanno inserito un altro blocco "MoveSteering" con il parametro type impostato ad "OnUnlimited".
- inoltre, possiamo notare la presenza di cicli in prossimità di "Change threshold" e "Change distance CM", nel dettaglio coloro che rieseguono queste operazioni sono rispettivamente il 35% e il 91% rispetto al totale dei gruppi.
- attraverso la vista "Case Coverage" offerta da Disco, possiamo vedere che "Add Blockname MoveSteering" e "Change Distance CM" vengono utilizzate dal 100% dei gruppi. Altre attività come "Add Blockname Ultrasonic Sensor" e "Add Blockname Ultrasonic Sensor Compare CM" vengono utilizzati molto spesso, rispettivamente dal 96% e 91% dei gruppi.

- invece, con la vista "Absolute Frequency", possiamo vedere che esiste un ciclo sull'attività "Change Distance CM" che viene effettuato 856 volte. La sola attività, invece, viene eseguita ben 1754 volte.
- dalla figura 40 sulle performance del modello non emergono considerazioni interessanti ai fini di individuare pattern caratteristici.

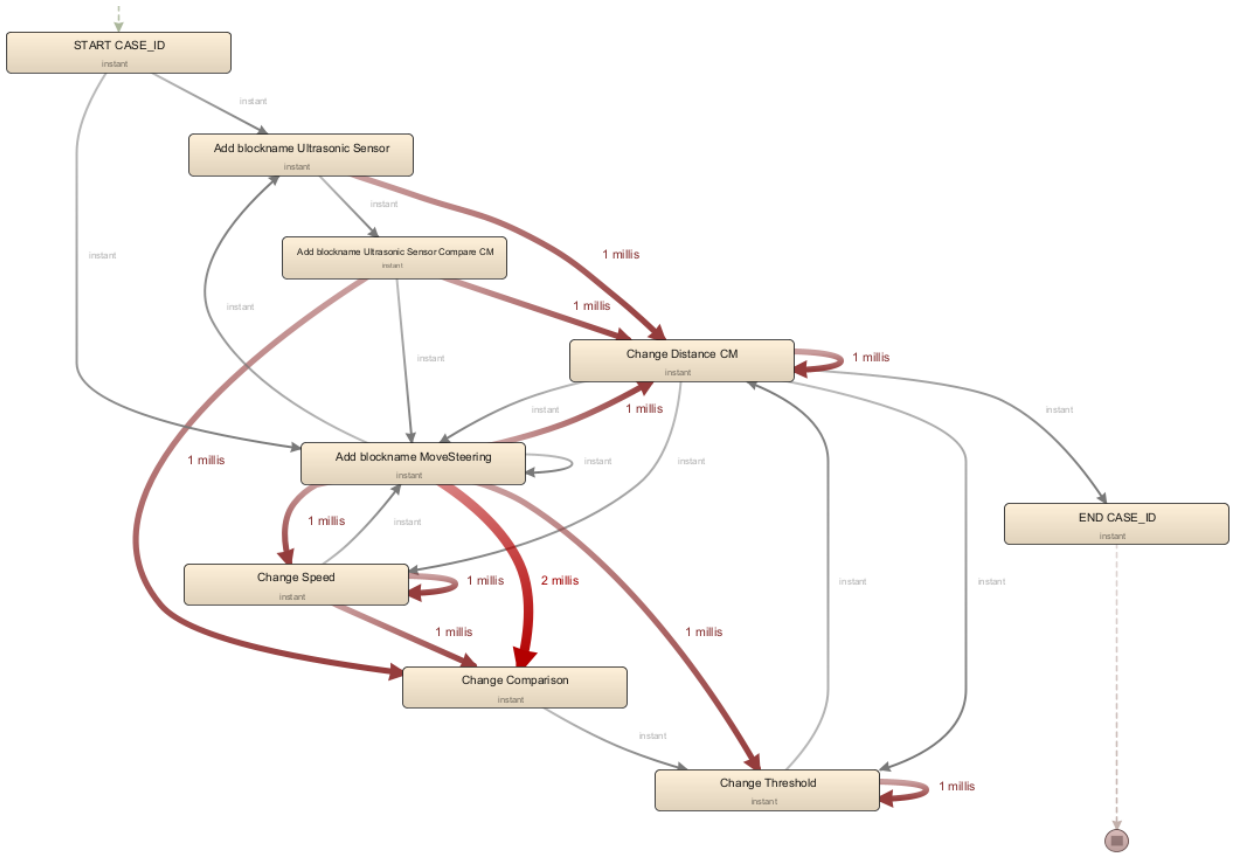

Figura 40: Performance good esercizio B

5.2.2.2 Wrong

Invece, per analizzare il file "wrong", i valori di activity e path vengono impostati rispettivamente a 20% e 8%. Queste considerazioni vengono riportate avendo a disposizione 29 casi su cui effettuare analisi.

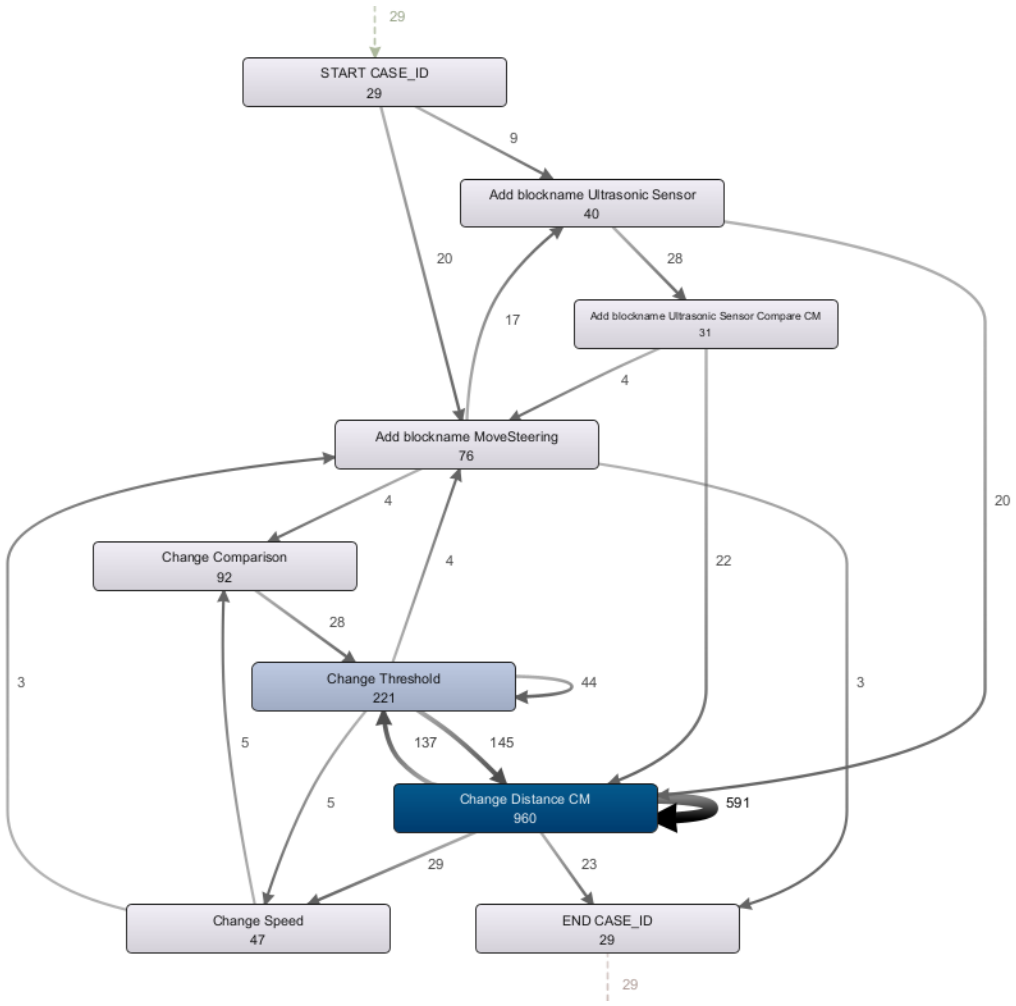

Figura 41: Modello wrong esercizio B

Osservando il modello di processo prodotto da coloro che non hanno raggiunto l'obiettivo (Wrong.csv) si evince:

- nella parte iniziale, a differenza di "good", circa il 70% rispetto al totale dei casi, aggiunge il blocco "MoveSteering". In seguito, la maggior parte di questi aggiungono il blocco "Ultrasonic Sensor" e successivamente poi il blocco "Ultrasonic Sensor compare CM".
- dopo l'aggiunta dei blocchi iniziali, nella maggior parte dei casi segue l'attività "Change Distance CM".
- successivamente, si può notare una divisone quasi equa tra chi decide di modificare il parametro "Speed" e chi decide di modificare il parametro "Threshold".
- inoltre, possiamo notare la presenza di cicli in prossimità di "Change threshold", e "Change distance CM", nel dettaglio coloro che rieseguono queste operazioni sono rispettivamente il 24% e l'86% rispetto al totale dei gruppi.
- attraverso la vista "Case Coverage" offerta da Disco, possiamo vedere che "Add Blockname MoveSteering","Change Distance CM" e "Add Blockname Ultrasonic Sensor" vengono utilizzate dal 100% dei gruppi. Altre attività come "Add Blockname Ultrasonic Sensor Compare CM" e "Change comparison" vengono utilizzati molto spesso, entrambi dall'83% dei gruppi.

- invece, con la vista "Absolute Frequency", possiamo vedere che esiste un ciclo sull'attività "Change Distance CM" che viene effettuato 591 volte. La sola attività, invece, viene eseguita ben 960 volte.
- dalla figura 42 sulle performance del modello non emergono considerazioni interessanti ai fini di individuare pattern caratteristici.

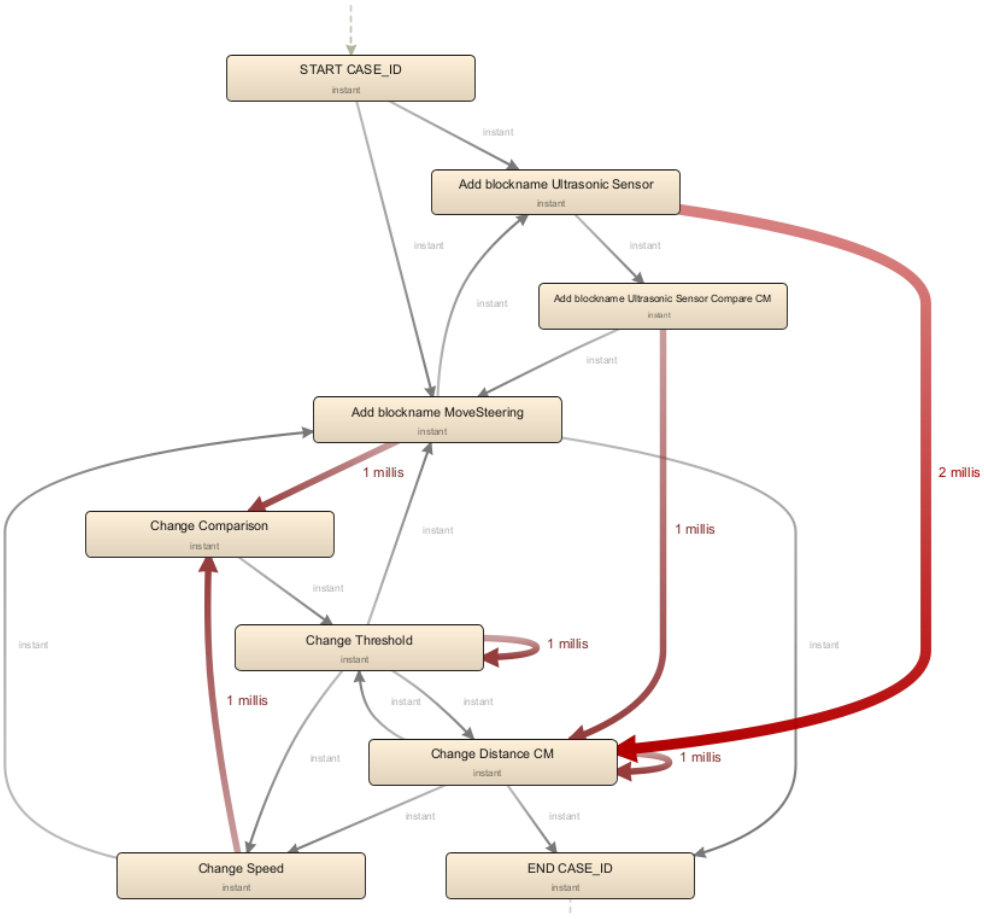

Figura 42: Performance wrong esercizio B

5.2.2.3 Confronto

Anche in questo esercizio, andiamo a confrontare i due modello ottenuti per evidenziare le caratteristiche simili e le differenze tra i comportamenti. In particolare possiamo affermare:

- analizzando entrambi i casi, si nota che i ragazzi all'inizio applicano tutti le stesse modalità, ovvero prima inseriscono il blocco "MoveSteering", poi aggiungono il blocco "UltraSonic Sensor" e successivamente il blocco "Ultrasonic Sensor compare CM".
- successivamente, in entrambi i casi, l'azione che viene eseguita più spesso, subito dopo l'aggiunta dei blocchi, è quella di "Change Distance CM".
- nel caso "wrong" abbiamo che i ragazzi decidono di cambiare i parametri "speed" e "threshold", mentre nel caso di "good" abbiamo che viene modificato i parametri "threshold" e "comparison".

- osservando la frequenza assoluta del ciclo su "Change Distance CM", è possibile osservare che nel caso di "good", ogni gruppo ripete mediamente circa 11 volte l'azione "Change Distance", mentre nel caso di "wrong" il numero di gruppi che esegue questo passo risulta essere il doppio. Invece, se soffermiamo l'attenzione solamente sull'attività e non sull'inter ciclo, abbiamo che su "good" ogni gruppo effettua quest'attività circa 22 volte in media, mentre nel caso di "wrong", quest'azione viene compiuta da ogni gruppo mediamente 33 volte. Quindi, non vi è un raddoppiamento come nel ciclo descritto in precedenza, ma aumenta di una quantità pari circa ad un terzo.

- infine, analizzando entrambi i modelli, non emergono particolari differenze di comportamento tra chi ha raggiunto l'obiettivo e chi no, quindi possiamo dedurre che le differenze sostanziali risiedono nell'impostazioni dei parametri dei blocchi.

6 Conclusioni

Lo scopo della analisi era quello di individuare dei pattern comportamentali di programmazione fra dei gruppi di ragazzi, al fine di individuare quei comportamenti che in genere portano al raggiungimento dell'obiettivo, confrontandoli invece con i comportamenti sbagliati.
Per fare questo sono state testate due diverse metodologie di process discovery. Nel primo caso utilizzando la libreria python pm4py abbiamo ottenuto dei modelli con delle reti di petri molto dettagliate ma conseguentemente anche complesse. La libreria mette a disposizione un filtraggio attraverso il parametro "noise threshold" ma, a causa anche della complessità dei dati e l'assenza di pattern significativi, il modello è sempre risultato poco intuitivo, specialmente nell'esercizio B. La libreria ci ha consentito però di ottenere delle metriche che descrivono dettagliatamente il modello.
Effettuando invece sempre la stessa analisi con il software Disco è stato possibile impostare un filtro molto più stringente sia sui path che sulle activity. Questo ha consentito di ottenere modelli molto più leggibili (con circa una decina di attività e pochi path) e quindi di effettuare analisi e confronti in modo molto più semplice. In questo modo, il modello sarà più chiaro da comprendere ma evidenzierà molto meno i comportamenti "anomali" o che capitano molto raramente. Il focus della nostra analisi è stato quello di confrontare i comportamenti e quindi i pattern di programmazione, invece che effettuare una ricerca dei modelli anomali. In qualche caso è stato analizzato anche un comportamento atipico, notato grazie al fatto che è stato eseguito da un numero considerevole di gruppi.
Infine, possiamo dire che estrarre conoscenza dal lavoro realizzato non è stato semplice perché avendo a disposizione dati altamente variabili (soprattutto nel caso dell'esercizio B), non è stato immediato riconoscere path significativi. Per quanto riguarda l'esercizio A, abbiamo potuto identificare un possibile comportamento che ha portato alla corretta esecuzione dell'esercizio: utilizzare la modalità del blocco "MoveSteering" in "OnForRotations" fa si che lo spostamento del robot dipende unicamente dalla variabile "Rotations", mentre chi sceglie la modalità "OnForSeconds" deve regolare in combinazione sia il parametro "Speed" che il parametro "Seconds" aumentando cosi la difficoltà. Invece, per quanto riguarda l'esercizio B non sono stati identificate particolari differenze nei comportamenti tra chi raggiunge l'obiettivo e chi no. Questo può essere dovuto al fatto che essendo molto più complesso e variabile, l'esercizio poteva essere eseguito in tanti modi. Tutte le metodologie che venivano eseguite solamente da pochi gruppi, sono state quindi filtrate dall'analisi. Quindi dal modello ottenuto possiamo concludere che le differenze risiedono maggiormente nel settaggio dei parametri sui blocchi. Questo rappresenta un aspetto cruciale in quanto, se un parametro viene impostato con una differenza rilevante dal valore ottimo, il gruppo non riuscirà nel portare a termine l'esercizio. Una differenza sostanziale è stata rilevata invece sul numero di tentativi effettuati. In genere, per entrambi gli esercizi, è stato notato che chi ha raggiunto l'obiettivo ha eseguito molti più tentativi rispetto a chi non ci è riuscito.