Tesi di Laurea Magistrale in Ingegneria Informatica

Testing automatico di applicazioni Android utilizzando algoritmi di Deep Reinforcement Learning

Anno accademico 2019/2020

Relatore
Ch.ma Prof. Anna Rita Fasolino

Correlatore
Ch.mo Prof. Porfirio Tramontana

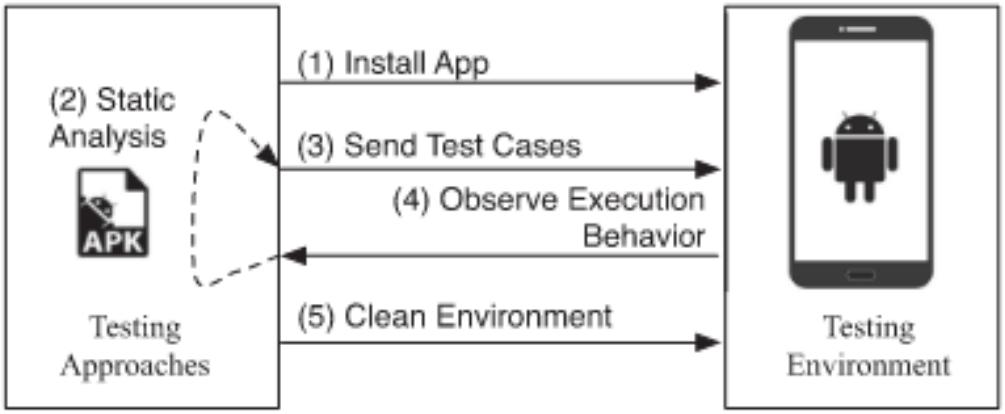
Candidato
Giovanni Pasanisi
Matr. M63/859

Contesto e contributo

- **Contesto:**
  - Utilizzo di tecniche di Deep Reinforcement Learning (Deep RL) applicate al testing di applicazioni Android

- **Contributo:**
  - Analisi dei tool esistenti per la generazione automatica dei test-case su dispositivi Android
  - Ampliamento di una sperimentazione precedente con l'introduzione di nuovi modelli di applicazioni Android «sintetiche» per l'addestramento degli algoritmi di Deep RL.

Modello generale delle fasi del testing mobile



Strategie convenzionali per il testing automatico

- **Random Testing**
  - Tecnica di testing nella quale i programmi vengono testati generando una sequenza casuale e indipendente di input.
- **Model-based Testing**
  - Generazione automatica di test case basata su modelli, che descrivono le funzionalità dell'AUT (Application Under Test).
- **Search-based Testing**
  - Utilizzo delle tecniche di ricerca metaeuristiche per la generazione di test, con l'obiettivo di rilevare il maggior numero di bug possibili.

Tool per la generazione automatica di test case

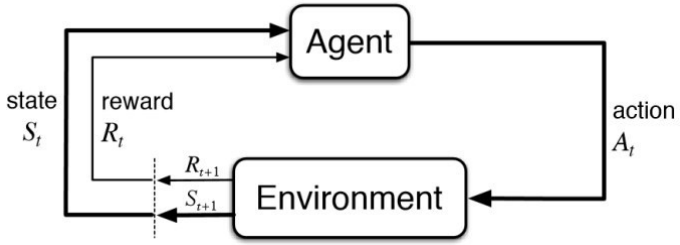
- **Monkey**
  - Generatore di test Android puramente randomizzato
  - Strumento da riga di comando
  - Può essere eseguito su qualsiasi emulatore o dispositivo

- **Sapienz**
  - Tool di tipo evolutivo per la generazione di casi ti test
  - Utilizza un algoritmo genetico per generare input ottimi che massimizzano la copertura del codice.

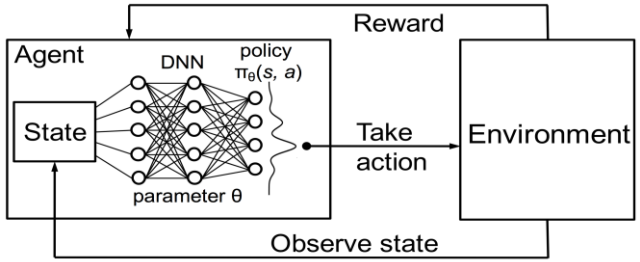
- **SwiftHand**
  - Incorpora un algoritmo di apprendimento attivo per creare un modello dell'app sotto test.

Reinforcement Learning e Deep Reinforcement Learning

- **Reinforcement Learning (RL)**
  - Sviluppa sistemi che apprendono direttamente dall'esperienza passata.
  - Lo scopo è quello di trovare un giusto compromesso tra l'esplorazione di nuovo codice e la valorizzazione di nuova conoscenza



- **Deep Reinforcement Learning (Deep RL)**
  - Combina il RL e il deep learning, permettendo di risolvere problemi che il RL non può risolvere.



Algoritmi di Deep RL utilizzati

DDPG (Deep Deterministic Policy Gradient)

E' un algoritmo di tipo *Actor-Critic:*

- *Actor:* propone un'azione dato uno stato.
- *Critic:* valuta se l'azione proposta è buona (premio positivo) o cattiva (premio negativo), dati uno stato e un'azione.

TD3 (Twin Delayed DDPG)

E' la versione migliorata dell'algoritmo DDPG:

- Corregge la sensibilità al tuning degli iperparametri.

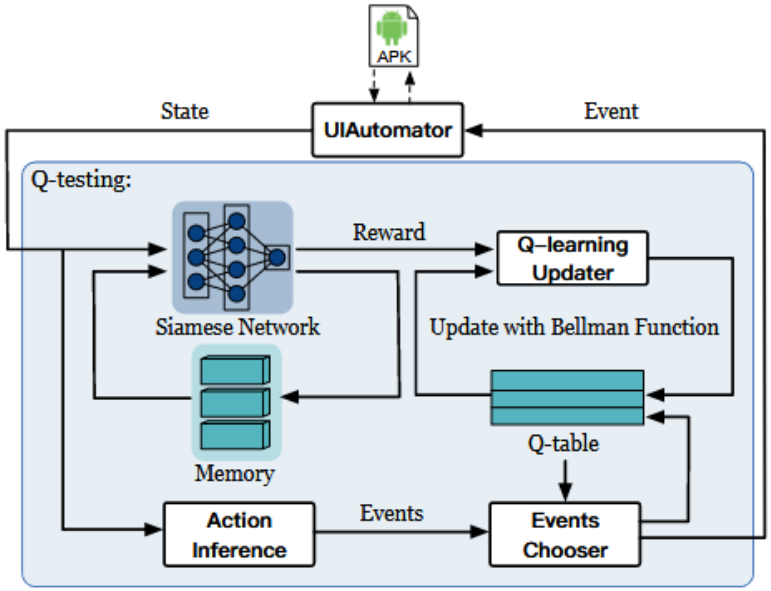
SAC (Soft Actor Critic)

La caratteristica principale è la regolarizzazione dell'entropia.

- L'agente riceverà un premio ad ogni passo di esecuzione che sarà proporzionale all'entropia della «*policy*» in quell'istante.

Applicazione del Q-learning al testing Android

- Il Q-testing interagisce con l'applicazione sotto test.
- Osserva lo stato corrente dell'applicazione utilizzando UIAutomator.
- Successivamente lo stato è confrontato con parte dei precedenti stati osservati.
- Gli stati vengono immagazzinati in un buffer.



Miglioramenti introdotti dall'utilizzo di una rete neurale:

- Copertura del codice
- Rilevazione dei fallimenti

Fast Android Test Environment (FATE)

Gli algoritmi di Deep RL richiedono «*fine-tuning*» che è dispendioso, in termini di tempo, sulle app reali.

FATE è un tool sviluppato da un gruppo di ricerca italiano con l'obiettivo di fornire un ambiente di simulazione per il testing rapido di applicazioni Android.

FATE modella esclusivamente i vincoli di navigazione delle app.
In questo modo permette di confrontare diversi algoritmi di testing e ricavare i migliori iperparametri.

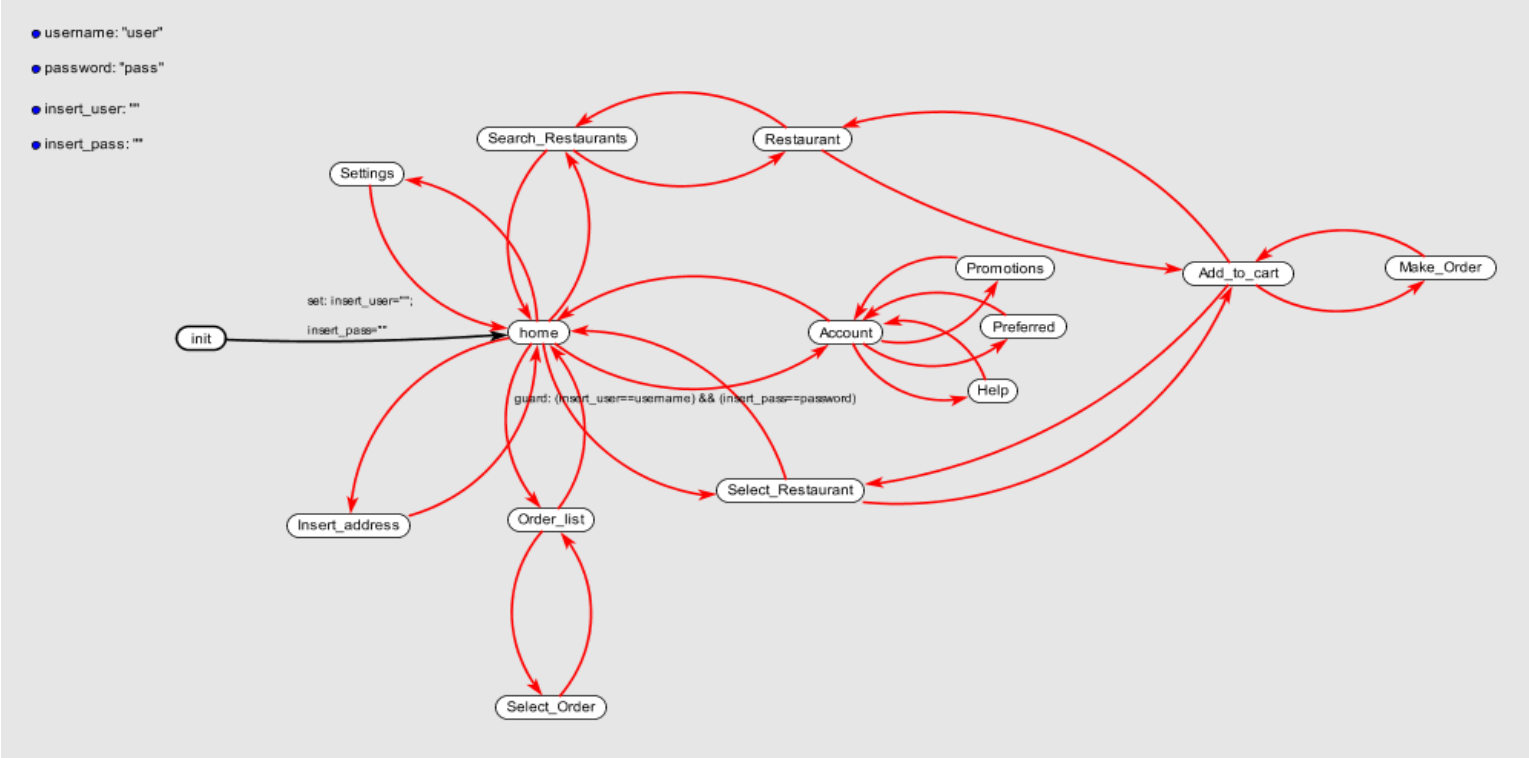
Fast Android Test Environment (FATE)

Il testing di FATE è stato effettuato considerando le seguenti categorie di applicazioni mobili:

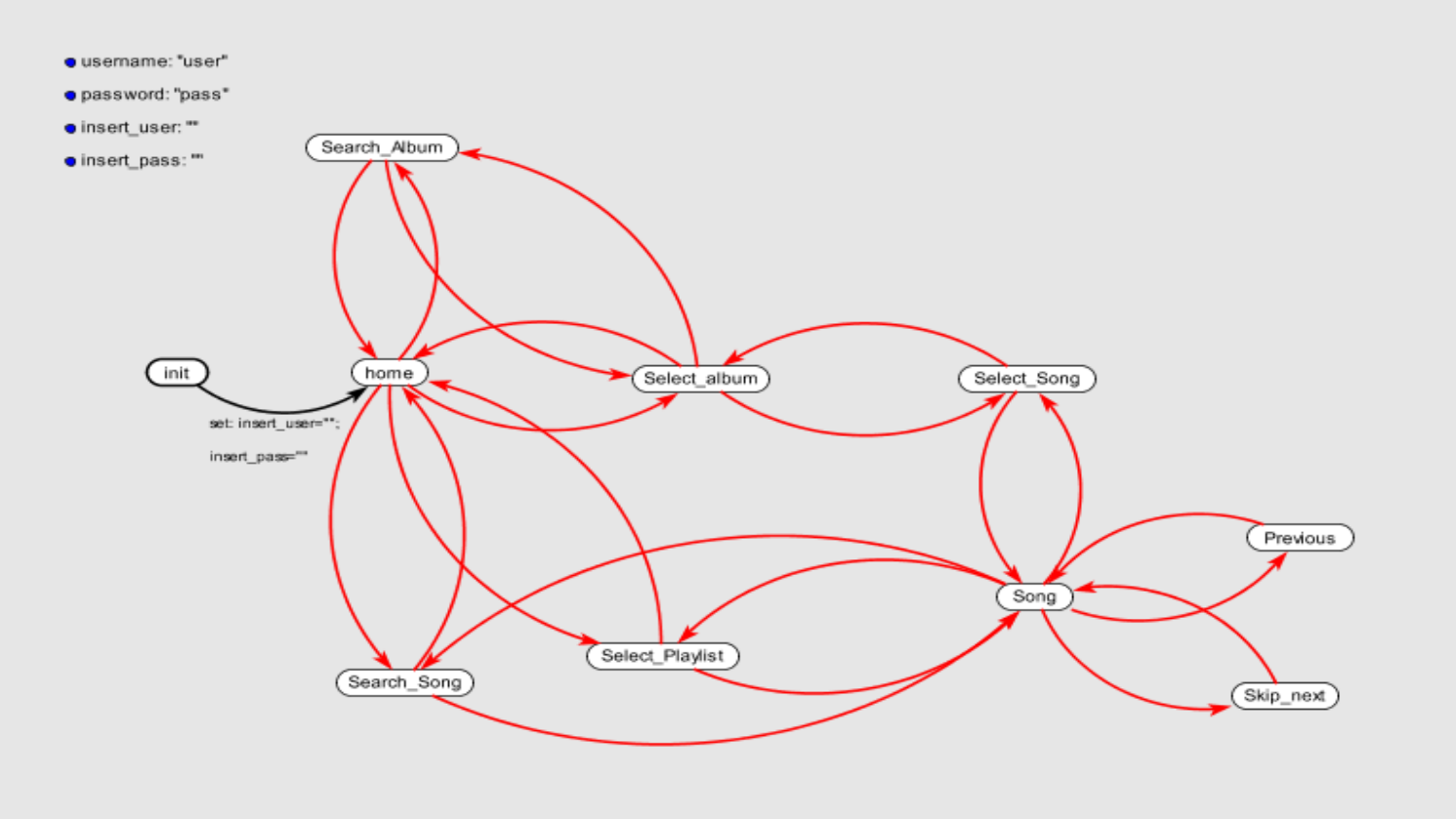
- Health and Fitness
- Food and Drink
- Music and Audio
- Bank

In questa tesi si sono valutate le performance di FATE considerando queste nuove categorie

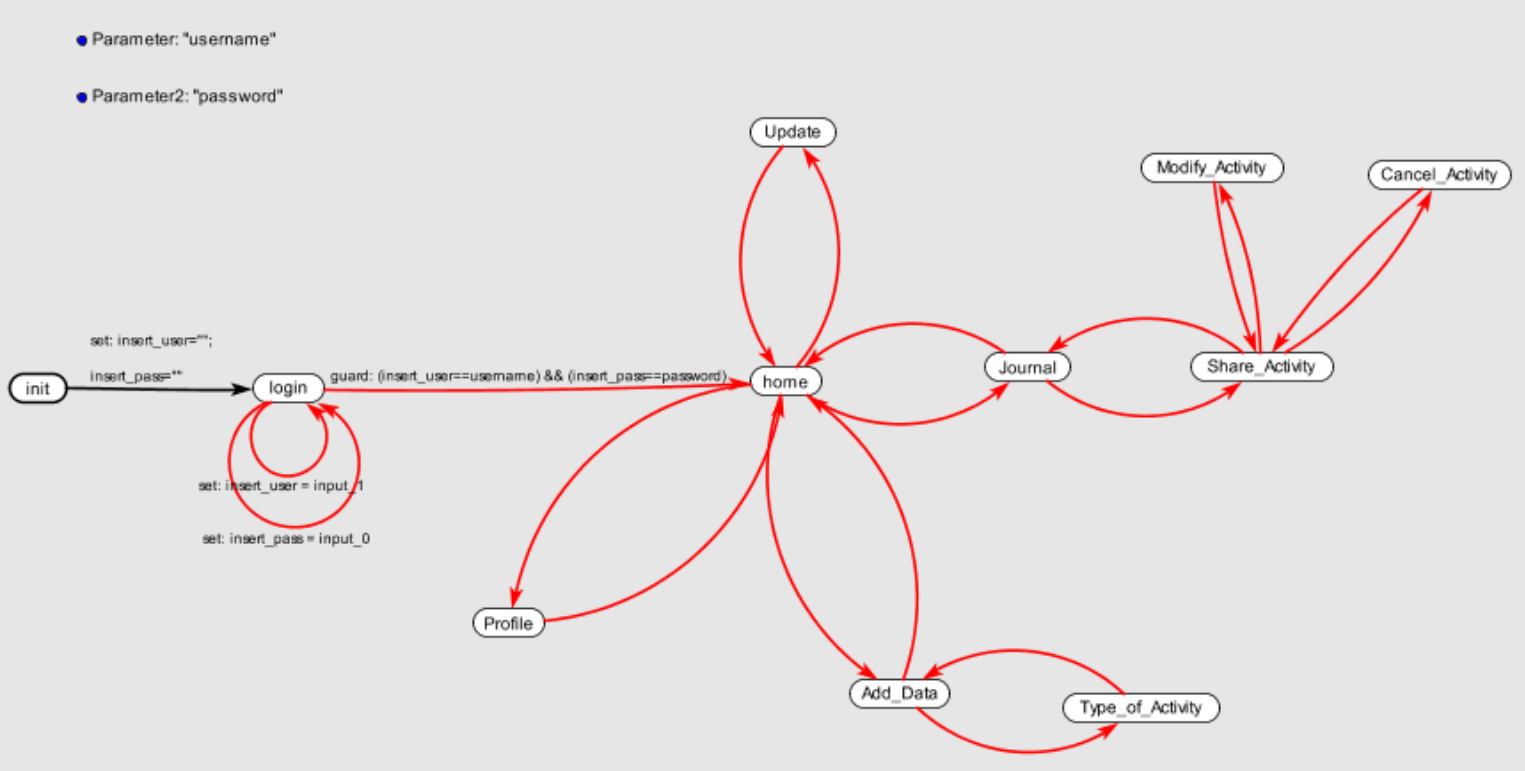
FSM per la categoria «Food and Drink»



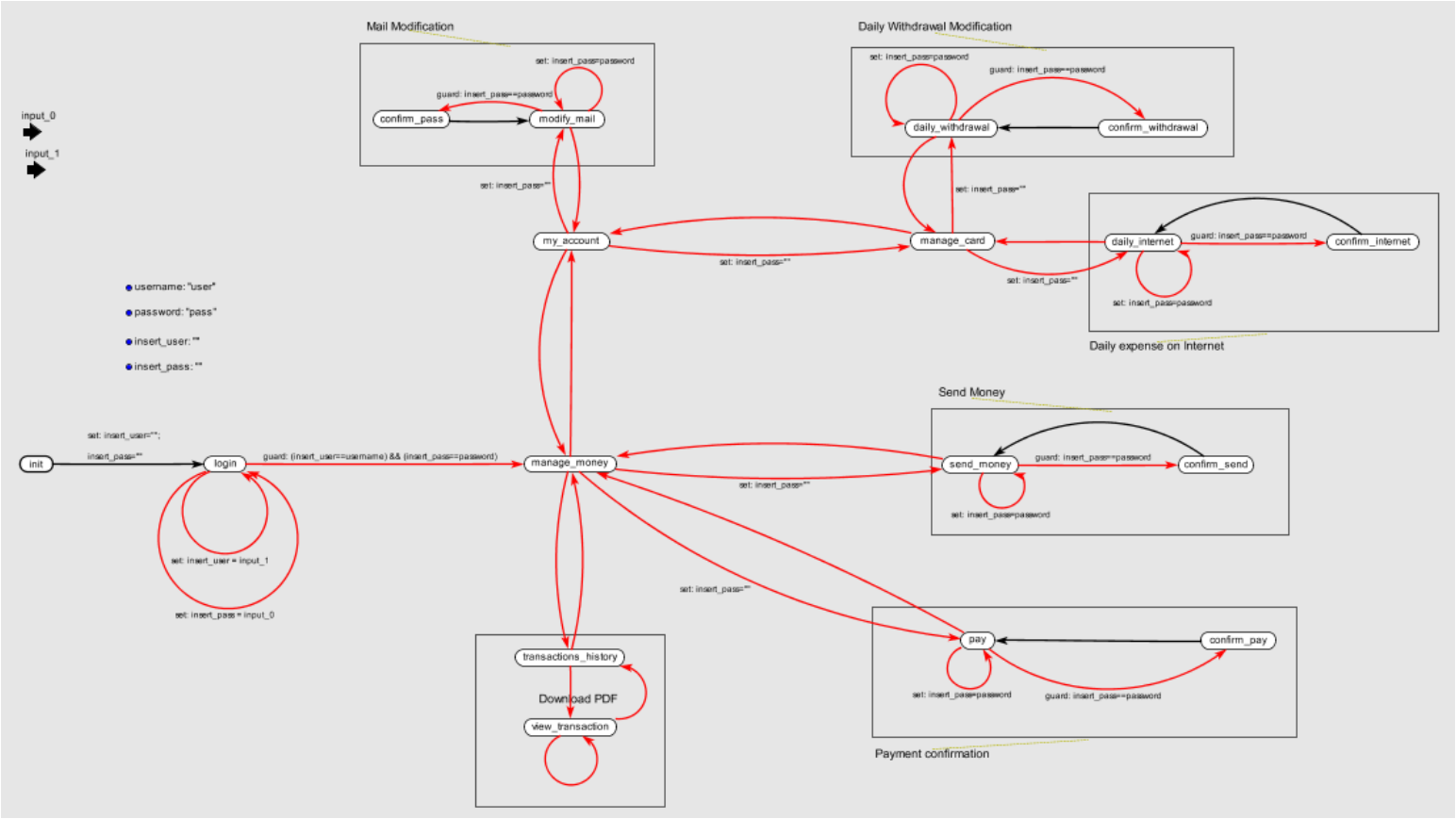
FSM per la categoria «Music and Audio»



FSM per la categoria «Health and Fitness»



FSM per la categoria «Bank»



Area Under Curve delle nuove applicazioni sintetiche

Categoria app	Dimensione stringhe	Random	TD3	SAC	DDPG
Health and Fitness	20_str	15244	20568	23587	28963
	40_str	5647	8547	9215	7658
	80_str	6854	7458	8324	4257
Music & Audio	20_str	21658	26547	24578	30214
	40_str	8235	6215	12548	3269
	80_str	4215	3689	5698	4632
Food and Drink	20_str	18659	19685	20369	23745
	40_str	9356	6547	7265	11365
	80_str	3698	4326	5324	4835
Bank Mod	20_str	15698	17325	19347	20314
	40_str	13547	14325	17324	16471
	80_str	10432	12035	15342	11974

Conclusioni

- In questo lavoro di tesi si è dunque ampliata una sperimentazione precedente, condotta con il tool FATE, introducendo descrizioni di nuovi automi a stati finiti per categorie di applicazioni Android.
- Si è dovuto provvedere, prima di tutto, a ricavare gli iperparametri che producessero le migliori prestazioni dei tre algoritmi di Deep RL del tool FATE per ogni nuova tipologia di automa a stati finiti.
- Dalle prove eseguite si conclude che all'aumentare della copertura l'algoritmo che produce le migliori prestazioni è il Soft-Actor-Critic.