Rielaborazione di algoritmi di generazione automatica di coreografie con Robot NAO

Chiara Cippitelli
Benedetta Rogato

Anno accademico 2022/2023 – Corso di Fondamenti di Intelligenza Artificiale M

Introduzione

- **Obiettivo**: rielaborazione di 2 algoritmi di generazione automatica di coreografie robotiche, dati dei vincoli riguardanti le mosse da eseguire e una traccia audio in input.
- Sono stati scelti 2 progetti e rielaborati sotto 4 punti di vista:
  - **portabilità** del codice sul sistema operativo Windows
  - upgrade da Python2 a **Python3** per l'esecuzione dei programmi
  - modifica del codice per il **campionamento e l'analisi musicale**
  - **nuove features** per personalizzare le scelte coreografiche e musicali. In particolare:
    - ✓scegliere una traccia musicale da analizzare
    - ✓scegliere la durata della coreografia
    - ✓scegliere quali mosse obbligatorie eseguire
    - ✓scegliere se si vuole rimuovere delle mosse non obbligatorie

Operazioni Preliminari

- **Analisi** del codice dei progetti scelti (ANaoRhythm e Silk-Sonic):
  - Entrambi risultavano funzionanti esclusivamente su macchina virtuale. È stato quindi modificato il codice rendendolo funzionante anche su Windows (10 e 11).
- **Reingegnerizzazione del main** file in 3 processi:
  1. Per la scelta e all'analisi della traccia musicale
  2. Per la riproduzione della traccia musicale
  3. Per l'esecuzione della coreografia generata.
- Scelta di un **pool di mosse condiviso** per confrontare le coreografie.
- **Categorizzazione delle mosse** in 3 sottoinsiemi in base alla velocità di esecuzione: lente, normali e veloci.

Portabilità su Windows

- Entrambi i progetti erano scarsamente portabili e non era garantita una buona riuscita nei test con il robot simulato su Choregraphe. Per questo è stato necessario modificare il codice.
- All'avvio è possibile riconoscere il sistema operativo su cui si esegue il programma (Windows o Unix), invocando di conseguenza funzioni differenti. Il tutto risulta trasparente all'utente.

```
def playSong(song):
    if sys.platform.startswith('linux'):
        os.chdir('..')
        bashCommand = "play " + song
        process = subprocess.Popen(bashCommand.split(), stdout=subprocess.PIPE)
        output, error = process.communicate()
    else:
        import winsound
        os.chdir('..')
        song = os.path.join(os.getcwd(), song)
        winsound.PlaySound(song, winsound.SND_FILENAME)
```

Gestione della Musica

- **Eliminazione della dipendenza** dell'algoritmo **da un'unica traccia musicale** cablata nel codice:
  - L'utente specifica la traccia musicale desiderata.
  - Nel caso in cui si volesse scegliere una traccia non presente nell'elenco, sarà sufficiente aggiungerla in formato .wav nella cartella Music.
- Invocazione dell'**algoritmo di analisi** di PartyNAO sulla traccia scelta:
  - campiona in ogni istante di tempo l'intensità della musica
  - la durata della traccia, su cui si basa il campionamento, non è più cablata nel codice, ma è la reale durata della traccia.

Generazione della Coreografia

- L'algoritmo di ricerca varia in base al progetto, ma la generazione della coreografia sfrutta in entrambi i casi una **funzione di valutazione delle mosse**: *value_BC()*.
- **Ad ogni mossa** è associato un **valore** compreso tra 0 e 1, dove 1 rappresenta una perfetta corrispondenza tra l'intensità relativa alla traccia musicale in un dato istante e la velocità della mossa. In particolare:
  - Match **buono** (0.9 - 1)
  - Match **accettabile** (0.3)
  - Match **errato** (0 - 0.1)

ANaoRhythm - 1

- L’algoritmo di ricerca è un’implementazione della **Iterative Deepening Search**.
- Si basava su un’euristica definita come la combinazione lineare tra la durata normalizzata della soluzione e il numero normalizzato di mosse compatibili con il beat (beat matching):

$$h = \alpha \cdot ST + \beta \cdot BM$$

- Dove:
  - $\beta = 1 - \alpha$,
  - ST = Total Time / 180 (Durata normalizzata della soluzione)
  - BM = Numero di mosse che partono su un beat / Numero totale di mosse (Beat Matching)

ANaoRhythm -2

- Questo comportava una **stretta dipendenza** tra la traccia musicale e l'euristica. Inoltre non era possibile generare una coreografia con una traccia di durata diversa da 3 minuti.
- Il valore BM viene sostituito con un nuovo parametro AV (**Average Value**) ottenuto come la somma del punteggio di ciascuna mossa / il numero totale di mosse. Il valore è ottenuto attraverso la funzione value_BC().
- È stato aggiunto un **nuovo parametro D**, personalizzabile dall'utente all'avvio, che permette di definire la **durata della coreografia**.

$$h = \alpha \cdot ST + \beta \cdot AV$$

- Dove:
  - ST = Total Time / D
  - AV = Somma del punteggio di ciascuna mossa / il numero totale di mosse

Esempi di Funzionamento - ANaoRhythm

```
Insert Robot IP: 127.0.0.1
Insert Robot Port: 62313

AVAILABLE SONGS:
1 : Arctic_Monkeys_Do_I_Wanna Know_3MIN.wav
2 : Muse_Feeling_Good_3MIN.wav

Which song would you like to play? Choose the number: 1

Set the duration of the choreography (in seconds): 30

You chose the song || Arctic_Monkeys_Do_I_Wanna Know_3MIN.wav || NAO will dance for:   30  seconds!

AVAILABLE MANDATORY POSITIONS:
0 : M_Hello
1 : M_Stand
2 : M_StandZero
3 : M_Sit
4 : M_WipeForehead
5 : M_SitRelax
Choose the mandatory position you want (integer): 0 1 2


AVAILABLE OPTIONAL POSITIONS:
0 : StandUp
1 : AirGuitar
2 : ArmDance
3 : BlowKisses
4 : Bow
5 : DiagonalRight
6 : DanceMove
7 : SprinklerL
8 : SprinklerR
9 : TheRobot
10 : ComeOn
11 : StayingAlive
12 : Rhythm
13 : PulpFiction
14 : Wave
15 : Glory
16 : Clap
17 : Joy
Choose the optional position you want to remove (integer): 6

Starting moves: ► I_StandInit ► M_Hello ► M_Stand ► M_StandZero ► F_Crouch
Computing Sequence:
```

```
Starting moves: ► I_StandInit ► M_Hello ► M_Stand ► M_StandZero ► F_Crouch
Computing Sequence:

Computing Time: 0.012689352035522461 seconds
Number of epochs: 7


STATS:

Best Move Sequence:
I_StandInit ► StandUp ► TheRobot ► M_Hello ► M_Stand ► BlowKisses ► M_StandZero ► F_Crouch

Total Time: 30.009999999999998s
Nodes in the last Tree: 13
Dance!
Move: I_StandInit
Move: StandUp
Move: TheRobot
Move: M_Hello
Move: M_Stand
Move: BlowKisses
Move: M_StandZero
Move: F_Crouch
```

Esempi di Funzionamento – Silk Sonic

```
IP: 127.0.0.1
Port: 54431

AVAILABLE SONGS:
1 : Arctic_Monkeys_Do_I_Wanna Know_3MIN.wav
2 : Muse_Feeling_Good_3MIN.wav

Which song would you like to play? Choose the number: 1

You chose the song || Arctic_Monkeys_Do_I_Wanna Know_3MIN.wav || let's dance nao!
AVAILABLE MANDATORY POSITIONS:
1 : I_StandInit
2 : M_WipeForehead
3 : M_Stand
4 : M_Hello
5 : M_Sit
6 : M_SitRelax
7 : M_StandZero
8 : F_Crouch
Choose the mandatory position you want to remove (integer): 6

AVAILABLE OPTIONAL POSITIONS:
0 : StandUp
1 : AirGuitar
2 : ArmDance
3 : BlowKisses
4 : Bow
5 : DiagonalRight
6 : DanceMove
7 : SprinklerL
8 : SprinklerR
9 : TheRobot
10 : ComeOn
11 : StayingAlive
12 : Rhythm
13 : PulpFiction
14 : Wave
15 : Glory
16 : Clap
17 : Joy
Choose the optional position you want to remove (integer): 15
```

```
Move: I_StandInit
Move: ArmDance
Move: BlowKisses
Move: SprinklerL
Move: Glory
Move: Rhythm
Move: M_WipeForehead
Move: StayingAlive
Move: ComeOn
Move: Bow
Move: Wave
Move: ArmDance
Move: M_Stand
Move: PulpFiction
Move: TheRobot
Move: Glory
Move: StayingAlive
Move: DanceMove
Move: M_Hello
Move: Wave
Move: Clap
Move: PulpFiction
Move: Rhythm
Move: BlowKisses
Move: M_Sit
Move: StandUp
Move: M_StandZero
Move: Rhythm
Move: DanceMove
Move: AirGuitar
Move: SprinklerR
Move: Wave
Move: F_Crouch
```