NAO CHALLENGE

UniBO FAIKR course – project by Stefano Andriolo

How it works

- The sequence of moves is defined using a search algorithm
- In this case, I used the AIMA Python library to explore the search space of **some** of the combination of moves among all the possible ones
- Total duration and number of repeated occurencies of moves are taken into account when evaluating those in the context of a choreography
- Each chosen move carries a *penalty value* which states whether it has been used in the previous **n** steps in order to prefer moves not seen in such steps and avoid repetition as much as possible
- The actual search step is performed by the **A*** search algorithm

Project structure

- **NAOproblem:** is the class with the problem definition which extends the **Problem** class of the **AIMA search library**
- **NAOsearch:** performs the actual search and returns the results ready to be used
- **NAOperform:** sends the desired moves to the NAO instance (physical or virtual)
- **NAOsong:** sends the chosen song to NAO to play it
- **NAOmove:** class describing a possible NAO move
- **NAOmain:** main class which ties all toghether, writes the choreography to the *choreography.txt* file as well to debug the sequence of chosen moves
- **constants:** contains constants used in the entire project, here parameters can be customized to try different configurations

Problems and possible improvements

- Not searching the entire solution space: for simplicity, only a random subspace of the solution space is explored
- Not accounting for low compatibility of moves: the compatibility between successive moves is not considered
- Add some further criteria to evaluate the “cost” of a move, i.e. preferring monothonically increasing or decreasing moves between one mandatory move and the following one
- Try other search strategies

THANK YOU