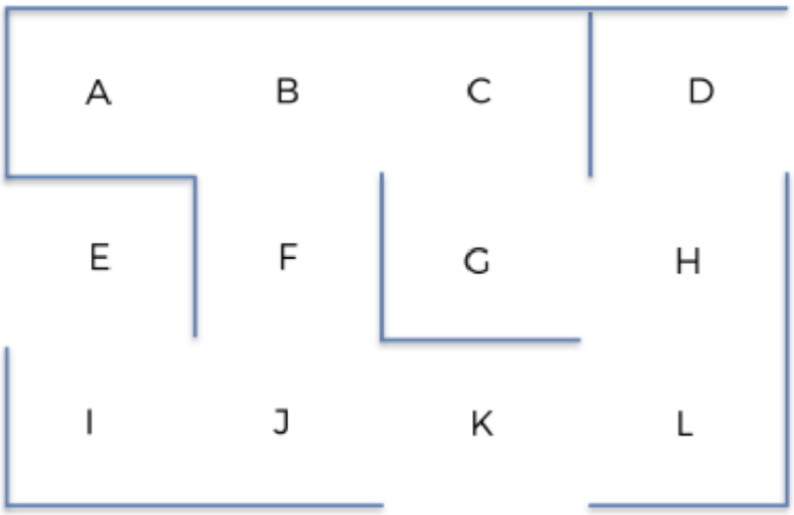
Reinforcement Learning (Q-Learning) per l'ottimizzazione del flusso di magazzino

L'obiettivo di questo lavoro è di trovare, dato un punto X e un punto Y, lo shortest path da X a Y mediante l'uso del reinforcement learning. In particolare, si vuole definire il percorso ottimale che deve effettuare un AGV (Veicolo a guida autonoma), che porta gli scaffali direttamente agli operatori di prelievo.
La figura seguente mostra la configurazione di magazzino sulla quale viene computato questo lavoro.



L'algoritmo viene implementato attraverso il principio ricorsivo del Q-Learning in modo da calcolare la sequenza ottimale per raggiungere l'obiettivo. Inoltre, nel caso in cui si vogliono includere delle fermate intermedie, l'algoritmo può essere facilmente modificato allo scopo di condurre il robot ad effettuarle, definendo, quindi, tale fermate, come vincoli, e trovare comunque il percorso ottimale.
Bisogna definire gli stati, le azioni e le rewards.

Stati

Ogni stato (le lettere che caratterizzano la figura) viene codificato con un numero:

{ A:0, B:1, C:2, D:3, E:4, F:5, G:6, H:7, I:8, J:9, K:10, L:11 }

Azioni

Le azioni vengono definite utilizzando lo stato corrente in cui si trova l'agente.
Quindi, si implementano le azioni in base allo stato poiché tutte le azioni non possono essere riprodotte in ogni stato. Quindi, come per gli stati, anche le azioni vengono rappresentate come un array:

actions = [0,1,2,3,4,5,6,7,8,9,10,11]

Rewards

Bisogna costruire un sistema di rewarding. Il processo viene costituito da un input: <Stato, Azione> e in output si avrà una reward.
In particolare, si va a delineare una matrice di adiacenza, dove si avrà una reward = 0 per le azioni che l'operatore non può svolgere, mentre si avrà una reward = 1 per le azioni che può svolgere.

Inoltre, il luogo che si vuole raggiungere avrà la più alta di tutte le rewards in modo che il training converga alla reward finale più alta, ovvero la destinazione prefissata.

Algoritmo

FASE 1 – Si seleziona uno stato random $\boldsymbol{S_t}$ dai 12 possibili stati:

Random(0,1,2,3,4,5,6,7,8,9,10,11)

FASE 2 – Si definisce un'azione random che può portare ad un prossimo stato possibile t.c. la funzione di ricompensa sia >0.3.

FASE 3 - Si raggiunge lo stato successivo e si ottiene la reward.

FASE 4 – Si va a computare la differenza temporale (TD)in accordo alla seguente:

$$\boldsymbol{TD_t(s_t, a_t) = R(s_t, a_t) + \gamma \max_a (Q(s_{t+1}, a)) - Q(s_t, a_t)}$$

Con TD = Reward immediata dell'azione selezionata + discount rate per valore massimo di tutte le azioni possibili nella posizione successiva.

FASE 5 – Si aggiorna il Q-value mediante l'equazione di bellman:

$$\boldsymbol{Q_t(s_t, a_t) = Q_{t-1}(s_t, a_t) + \alpha TD_t(s_t, a_t)}$$

L'equazione permette di calcolare il valore di ogni azione in funzione delle azioni future. Il valore di un'azione è scomposto in una reward immediata per l'azione (i.e. movimento da una locazione X ad una locazione Y) e il discount rate per valore massimo di tutte le azioni possibili nella posizione successiva. Come parte del processo di apprendimento, un AGV itererà più volte esplorando l'ambiente, consentendo il calcolo ricorsivo del valore di tutte le azioni (i valori Q). Questi valori alla fine convergono, consentendo di determinare il percorso ottimale dato un punto di partenza e uno di arrivo. Si usa, quindi, la matrice Q-value per determinare il percorso ottimale da una posizione di partenza ad una di destinazione. Questo si ottiene in primo luogo, assegnando una grande reward alla posizione target e, in secondo luogo, calcolando la matrice dei valori Q attraverso le iterazioni dell'AGV (esplorazione). Il passaggio finale richiede di seguire il valore massimo delle azioni possibili in ciascuna posizione in modo iterativo. L'azione con il valore più alto indica il passaggio alla posizione successiva, e così via, fino a quando non viene determinato passo dopo passo il percorso ottimale.

.