Algoritmi Genetici per la Soluzione di Labirinti

Progetto per il corso di Sistemi Intelligenti Avanzati dell'Università degli Studi di Milano

A.A. 2023-2024

Di Gianluca Iacchini

Introduzione

Questo report ha lo scopo di dettagliare l'implementazione di un progetto mirato alla soluzione di labirinti utilizzando gli algoritmi genetici.
Il progetto è stato ideato per il corso di *Sistemi Intelligenti Avanzati* dell'Università degli Studi di Milano.

Algoritmi Genetici

Gli algoritmi genetici sono una classe di algoritmi solitamente usati per risolvere problemi di ottimizzazione. Essi sono ispirati al processo della selezione naturale teorizzato da Charles Darwin.

Popolazione

Gli algoritmi genetici sono costituiti da una *popolazione* di *individui*. Ogni individuo rappresenta una possibile soluzione al problema.

Cromosomi e Geni

Ogni individuo possiede al suo interno una rappresentazione genetica, chiamata *cromosoma*, la quale è composta da una serie di *geni*.
I geni rappresentano le caratteristiche della soluzione.

Processo iterativo

Una popolazione di individui costituisce una *generazione*. Periodicamente alcuni individui vengono selezionati dalla generazione corrente e combinati tra loro per produrre dei nuovi individui, i *figli,* che costituiranno una nuova generazione.
Solitamente i figli rappresentano delle soluzioni migliori rispetto ai padri, pertanto dopo tante generazioni avremo degli individui la cui soluzione si avvicinerà di molto a quella richiesta dal problema.
Per valutare la bontà di una soluzione viene usata una funzione chiamata *funzione di fitness.*
Il processo iterativo è formato da tre step

1. Selezione

    - Alcuni individui vengono scelti dalla generazione attuale per costituire quella successiva. Di solito questi vengono scelti in base al valore assegnato loro dalla funzione di fitness.
2. Combinazione
    - Gli individui scelti nel passo precedente vengono combinati tra loro per produrre i figli della generazione successiva
3. Mutazione
    - I figli ottenuti dal passo precedente possono subire delle mutazioni che cambiano il valore di uno o più geni.

Il progetto

Il labirinto

Il labirinto è costituito da una serie di percorsi intrecciati. Ognuno di questi percorsi termina in un punto cieco; l'uscita del labirinto è piazzata alla fine del percorso più lungo.

Il labirinto è generato in maniera casuale utilizzando un algoritmo *DFS* con backtracking[1].

Individui

Gli individui rappresentano possibili soluzioni al labirinto, ossia possibili percorsi che portano dall'entrata all'uscita.

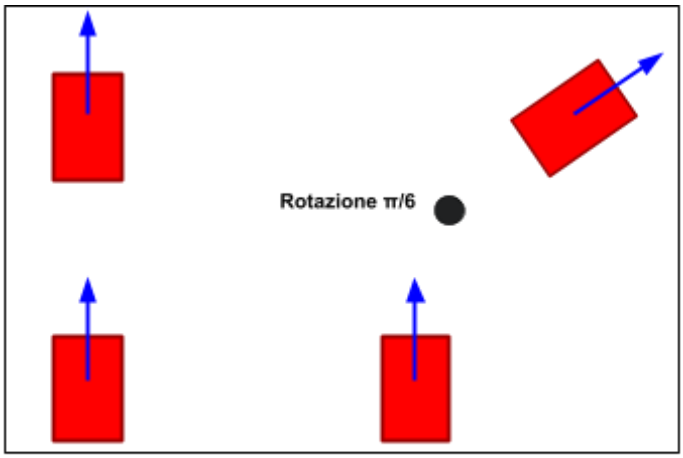
[1] *https://www.youtube.com/watch?v=_aeYq5BmDMg*

Il loro movimento è ristretto al piano del labirinto. Su di loro è applicata una velocità la cui direzione coincide con il loro orientamento, ossia l'*asse z* rispetto al sistema di riferimento locale di ciascun agente, e terminano se entrano in contatto con una qualsiasi delle mura del labirinto o dopo un tempo massimo di vita prefissato.
Gli individui sono liberi di ruotare su se stessi ad ogni angolo, a differenza del labirinto, i cui percorsi sono costruiti parallelamente agli assi del piano.

Gli individui non conoscono la posizione dell'uscita ma sono in grado di riconoscerla se la vedono. Inoltre sono capaci di capire la direzione verso la quale le mura si trovano e ricordano le zone che sono già state visitate in passato.

Cromosomi e geni

I cromosomi degli individui sono composti da una lista di valori decimali che rappresentano delle rotazioni in radianti.
Ciascun agente applica periodicamente queste rotazioni su se stesso. Poiché gli agenti si muovono a velocità fissa verso la direzione nella quale sono orientati, le rotazioni causano delle variazioni nel percorso che l'agente traccia.



Nel caso a sinistra non vi sono rotazioni, l'individuo si muove nella direzione verso la quale sta guardando.
Nel secondo caso l'individuo viene affetto da una rotazione, di conseguenza il suo orientamento verrà modificato.

Timestep

Il *timestep* regola la frequenza con la quale gli individui applicano le rotazioni contenute nei loro cromosomi.
Le rotazioni non vengono applicate istantaneamente ma sono applicate gradualmente durante l'arco del timestep.
Per esempio, se un individuo possiede una rotazione iniziale di 0 *rad*, il prossimo gene indica una rotazione di +1 *rad* ed il timestep è di *1s,* allora l'individuo avrà una rotazione di *0.25 rad* a *0.25s*, *0.5 rad* a *0.5s* e raggiungera' la rotazione completa di *1 rad* solamente allo scadere del timestep, ossia dopo *1s*.

L’algoritmo

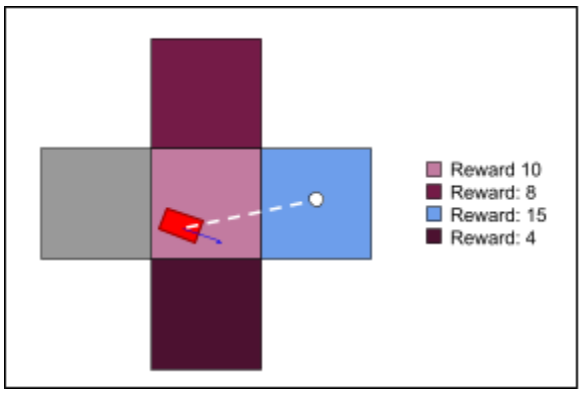
Funzione di fitness

La funzione di fitness tiene conto di diversi fattori. Poiché gli individui non conoscono la posizione dell’uscita finché non la scoprono, abbiamo che la funzione di fitness inizialmente favorisce gli agenti che esplorano il labirinto. Questo viene ottenuto tenendo traccia delle zone che sono già state visitate dagli individui e della distanza percorsa dall’individuo di cui si sta calcolando il valore di fitness.
Una volta che un individuo trova l’uscita esso viene ricompensato con una grossa ricompensa che incrementa all’avvicinarsi dell’individuo verso l’uscita.

Esplorazione

Per prediligere l’esplorazione il labirinto viene diviso in celle della stessa dimensione. Queste celle posseggono un valore individuale che indica quante volte sono state visitate.
Ogni volta che un individuo termina il suo percorso esso registra in quale cella si trova e ne incrementa il valore; celle con un valore più basso offrono una ricompensa maggiore in quanto esse sono state visitate meno.

Questo ragionamento viene esteso anche alle celle vicine (a distanza 1) a quella dove l’individuo ha terminato il percorso. Esso controlla tra le celle vicine quale fornisce la ricompensa maggiore (ossia quale è stata visitata meno volte) ed ottiene una ricompensa aggiuntiva data dal valore di tale cella e dalla vicinanza tra l’individuo e tale cella.



Per esempio, nell’immagine precedente abbiamo che l’agente termina il percorso nello spazio rosa, ottiene quindi un reward di 10.
A questo punto controlla quale tra le celle vicine ha il reward maggiore e scopre che è quella azzurra, ottiene quindi un reward aggiuntivo di 15 pesato per la distanza tra l’agente ed il centro di tale cella.

Vicoli ciechi

Nel caso l'individuo si renda conto di essere in un vicolo cieco viene incrementato di molto il valore della cella (che di conseguenza fornirà una ricompensa molto bassa).

Percorsi lunghi

Per favorire l'esplorazione, gli agenti che percorrono percorsi più lunghi unici (ossia senza tornare sui propri passi) vengono ricompensati ulteriormente.

Selezione

Gli algoritmi di selezione implementati sono

- Selezione proporzionale
- Selezione a torneo

Essi non differiscono dalle implementazioni usuali. Per quanto riguarda questo progetto, la selezione a torneo solitamente produce risultati migliori

Combinazione

Gli algoritmi di combinazione implementati sono

- Combinazione ad un punto
- Combinazione a due punti
- Combinazione aritmetica (media pesata)
- Combinazione uniforme

Gli algoritmi di combinazione ad un punto, due punti presentano le seguenti modifiche rispetto all'implementazione classica

- Ogni coppia di genitori produce un solo figlio
- La combinazione è eseguita in maniera tale da far sì che il figlio eredita la parte finale del genoma dal genitore con il valore di fitness più alto.

Questo perché i genitori con il valore di fitness più alto probabilmente sono arrivati più lontano o abbiano scoperto delle zone inesplorate quando hanno terminato la computazione.
Di conseguenza ereditare la parte finale del genoma di tale genitore favorisce l'esplorazione.

Mutazione

Dopo la combinazione, il figlio ottenuto ha una piccola probabilità di mutare, se questo è il caso la mutazione può avvenire in due modi.
Il primo controlla se il genitore che ha fornito a questo figlio la parte terminale del suo genoma ha terminato la computazione per via di una collisione con un muro.
In questo caso il figlio muta gli ultimi geni per tentare di evitare una nuova collisione.

Il secondo semplicemente muta uno qualsiasi dei geni del figlio scelto in maniera casuale.
Questa probabilità è pesata a favore degli ultimi geni, in quanto questo favorisce l'esplorazione.

Risultati e problemi riscontrati

Risultati

L'algoritmo riesce a trovare l'uscita nella maggior parte dei casi. Il numero medio di generazioni necessarie ed il tasso di successo dipendono da diversi fattori come le impostazioni dell'algoritmo, la grandezza del labirinto, il numero di vicoli ciechi, la distanza dell'uscita.

Problemi noti

Al fine di velocizzare la simulazione di ogni generazione, gli agenti si muovono velocemente all'interno del labirinto. Tuttavia l'alta velocità degli individui può creare risultati inconsistenti con il sistema di collisioni, per esempio alcuni individui registrano una collisione mentre altri no, anche se il percorso tracciato è lo stesso.
Nel caso in cui una collisione sia registrata erroneamente la funzione di fitness verrà aggiornata con un valore non veritiero, ed in base alla situazione attuale della simulazione potrebbe causare la regressione dell'algoritmo.

Struttura del progetto

- *AgentData.cs*
    - Contiene i dati che compongono gli individui come il DNA e la velocità di partenza.
- *Agent.cs*
    - Regola il comportamento dei GameObject che rappresentano gli individui: ne controlla la velocità e rotazione, regola il loro comportamento quando vengono creati e distrutti
- *AgentManager.cs*
    - Contiene il cuore dell'algoritmo, inizializza e gestisce la popolazione, implementa le funzioni di selezione, combinazione, mutazione e la funzione di fitness.
- *AgentPooler.cs*
    - Gestisce la creazione e distruzione dei GameObject degli agenti attraverso il pooling per aumentare le prestazioni.
- *GaussianDistribution.cs*
    - Implementa alcune funzioni di utility per la generazione casuale di numeri.
- *MazeCell.cs*
    - Rappresenta una cella del labirinto e contiene funzioni per la sua gestione
- *MazeGenerator.cs*
    - Gestisce la creazione e distruzione del labirinto
- *CameraControl.cs*
    - Gestisce la camera