Alma Mater Studiorum · Università di Bologna

Dipartimento di Informatica - Scienza e Ingegneria (DISI)
Corso di Laurea Magistrale in Informatica

Collaborative Robbies

Relazione di FSC

18 dicembre 2015

Giulio Biagini
0000705715
giulio.biagini@studio.unibo.it

Daniele Baschieri
0000688992
daniele.baschieri@studio.unibo.it

Indice

Capitolo 1

Introduzione

In questo capitolo andremo a descrivere *Robby: The Soda-Can-Collecting Robot* [1], il progetto a cui questo lavoro si ispira, riportando alcuni cenni teorici di Intelligenza Artificiale che useremo per descrivere in maniera più formale la struttura dell'intero progetto.

1.1 Robby: The Soda-Can-Collecting Robot

Robby è un robot il cui compito consiste nel muoversi in uno spazio sporco e cercare di ripulirlo. Lo spazio può essere visto in modo astratto come una griglia suddivisa in celle. Robby possiede solamente una vista parziale dello spazio nel quale si trova: è in grado di vedere cosa è presente nella cella a nord, in quella a sinistra, nella casella in cui si trova, in quella a destra ed in quella a sud rispetto alla propria posizione. Le celle possono essere "pulite" (vuote), "sporche" (nelle quali è presente una lattina da raccogliere) oppure rappresentare un ostacolo (un muro).
Robby attua un'azione in base alla vista che ha in quel momento dell'ambiente. Se, ad esempio, vede una lattina nella cella a nord, un muro nella cella a sinistra e niente nella cella da lui occupata, in quella a destra e nella cella a sud, questo non si muoverà negli spazi vuoti, né tantomeno sbatterà contro il muro muovendosi a sinistra, ma si muoverà di un passo verso l'alto, in direzione della lattina, per poi raccoglierla. Le azioni che Robby può effettuare sono: la mossa verso nord, verso sinistra, verso destra e verso sud, può decidere di rimanere fermo, raccogliere una lattina oppure effettuare una mossa casuale scelta fra le precedenti. Tutte le mosse hanno l'effetto immaginato a parte la raccolta della lattina. Questa toglierà la lattina dalla mappa nella cella in cui il robot si trova, se presente, altrimenti lascerà l'ambiente invariato.

Inizialmente Robby possiede un dna casuale, ovvero, ad ogni vista è associata un'azione casuale. Robby è però in grado di apprendere come muoversi correttamente nell'ambiente e come raccogliere lattine. La sua evoluzione è infatti guidata da un Algoritmo Genetico.

1.2 Entità Intelligenti

Lo scopo dell'*Intelligenza Artificiale* è quello di cercare di capire come le *entità intelligenti* funzionano ed, in particolare, quello di cercare di costruire, creare, entità intelligenti.
Negli anni sono state date varie definizioni di entità intelligenti: alcune fanno paragoni con gli esseri umani, altre usano il principio della razionalità, alcune definiscono queste entità in base a come pensano, altre a come agiscono. Nel nostro particolare caso, andremo a definire un'entità come intelligente se *agisce in modo razionale*, ovvero se "fa la cosa giusta".
Affinché **Robby**, l'entità intelligente che abbiamo il compito di creare, possa essere definita intelligente, dovrà quindi "fare la cosa giusta". Intuitivamente, siccome il compito del robot sarà quello di muoversi in uno spazio cercando di pulirlo, "fare la cosa giusta" significherà, ad esempio, non sbattere contro i muri durante il proprio movimento, raccogliere le lattine durante il proprio passaggio e non tentare di raccogliere lattine dove queste non sono presenti.

1.3 Agenti Intelligenti

Un *agente intelligente* è un'entità intelligente in grado di percepire l'ambiente tramite dei *sensori* e compiere delle azioni tramite degli *attuatori*.
Un agente ha delle *percezioni* dell'ambiente, ovvero l'insieme di tutti gli input percettivi che provengono dai propri sensori in un dato istante.
Il comportamento di un agente intelligente è descritto matematicamente da una *funzione agente*, ovvero l'insieme di tutte le sequenze percettive e da tutte le relative azioni. L'implementazione di una funzione agente prende il nome di *programma agente*.
Analizzando la definizione di agente intelligente, possiamo notare come **Robby** sia un'entità che appartenga a questa categoria: esso infatti è in grado di percepire l'ambiente tramite dei sensori che gli permettono di vedere il contenuto delle celle che lo circondano e possiede degli attuatori che gli consentono di muoversi e raccogliere lattine.
L'insieme di tutte le viste (percezioni) che Robby può avere dell'ambiente e le rispettive azioni sono descritte da una funzione agente. Il nostro compi-

to sarà quello di fornirne un'implementazione attraverso la scrittura di un programma agente.

1.4 Agenti Razionali

Il nostro obiettivo, però, non è semplicemente quello di creare agenti intelligenti, ovvero entità in grado di percepire l'ambiente e compiere azioni, ma, piuttosto, creare *agenti razionali*, ovvero entità sì intelligenti, ma che "facciano la cosa giusta". Questo significa che il programma agente, per ogni sequenza percettiva, produce un'azione che va a massimizzare una determinata *misura di prestazione*: se le azioni attuate dall'agente portano l'ambiente ad attraversare una sequenza di stati che può essere definita "desiderabile", allora la misura di prestazione è massimizzata.
Nel caso di **Robby**, la misura di prestazione da massimizzare sarà sia il numero di lattine che debbono essere raccolte in un dato ambiente, sia il numero di passi impiegato per raccoglierle.

1.5 Agenti Reattivi Semplici

Esistono varie tipologie di programmi agente in base alle tipologie di agenti di cui questi hanno il compito di descrivere il comportamento. I più semplici di tutti sono gli *Agenti Reattivi Semplici*, ovvero agenti che basano le proprie azioni solamente sulla percezione corrente. Questi agenti, dunque, analizzano tutti gli input percettivi che provengono dai sensori in un dato istante e computano quale azione compiere.
Robby è un agente reattivo semplice in quanto la scelta dell'azione da attuare è guidata solamente dalla vista che ha in quel momento dell'ambiente.

1.6 Agenti in Grado di Apprendere

Esistono particolari tipologie di agenti che possono essere programmati in modo che siano *in grado di apprendere*.
Un agente, infatti, può calcolare la scelta delle proprie azioni basandosi su conoscenze pregresse che ha dell'ambiente, che sono ad esempio state inserite a priori da un programmatore: in questo caso si dice che l'agente manca di autonomia. Al contrario un *agente autonomo* è in grado di apprendere per compensare la presenza di conoscenza parziale o erronea. Un agente in grado di apprendere ha il vantaggio di poter operare in ambienti all'inizio sconosciuti, diventando col tempo via via più competente.

Questa tipologia di agenti possiedono, oltre ad un *elemento esecutivo*, che gli permette di selezionare le azioni da compiere, anche un *elemento di apprendimento*, responsabile del miglioramento interno, il quale usa le informazioni provenienti dall'*elemento critico* riguardo le prestazioni correnti dell'agente e determina se e come modificare l'elemento esecutivo affinché in futuro si comporti meglio. Infine, le entità in grado di apprendere possiedono un *generatore di problemi*, il cui scopo è quello di suggerire azioni che portino ad esperienze nuove e significative dalle quali apprendere.
Nel caso di **Robby**, quello che vogliamo è un agente autonomo in grado di apprendere, ovvero un agente che non si basi su conoscenza pregressa da noi inserita. Per fare questo ci appoggeremo sugli *Algoritmi Genetici*.

Capitolo 2

Obiettivi

In questo capitolo sono descritti gli obiettivi di questo lavoro. In particolare, quello che ci interessa non è tanto re-implementare il lavoro proposto da Melanie Mitchell [1], quanto piuttosto realizzare un sistema che ci permetta di studiare come, inserendo più entità nell'ambiente, queste si influenzino a vicenda.
Lo scopo principale, dunque, è analizzare se può esservi **collaborazione spontanea** o meno fra le entità che si trovano a nella stessa mappa.

2.1 Entità Invisibili - Vista a Croce

Come prima cosa andremo a cercare di stabilire quali sono le performance di due agenti che si muovono sulla mappa senza vedersi, ovvero ignorando la posizione dell'altro robot. Sarà dunque ammesso trovarsi nella stessa cella e raccogliere la stessa lattina.
I due robot avranno vista a croce, ovvero come quella descritta nel capitolo introduttivo: potranno vedere la casella a nord, quella a sinistra, quella nella quale si trovano, quella a destra e quella a sud.
I risultati ottenuti saranno poi usati come "caso base", per fare un confronto con le simulazioni lanciate successivamente.

2.2 Entità Invisibili - Vista Quadrata

In questo secondo test, così come nel caso precedente, le due entità non saranno in grado di vedersi ma avranno una percezione maggiore della mappa nella quale si troveranno. Potranno infatti vedere uno spazio di 3x3 celle dove il robot occupa la posizione centrale (seconda riga e seconda colonna).
Questa prova ci permetterà di capire se potendo vedere una porzione di

mappa maggiore, i robot saranno avvantaggiati nella pulizia della stessa, ovvero, se disponendo di più risorse (sensori più potenti) saranno in grado di massimizzare maggiormente la misura di prestazione: sia essa la raccolta di un numero maggiore di lattine, sia essa la raccolta dello stesso numero di lattine ma con un numero minore di passi.

2.3 Entità Visibili - Vista a Croce

Con quest'ultima simulazione vedremo cosa succede dando la possibilità ai due robot di vedersi. In questo caso non sarà tollerata la compresenza nella medesima cella.
Vedremo così se i due robot impareranno a collaborare o se invece la presenza di uno ostacolerà l'altro.

Capitolo 3

Progettazione

In questo capitolo parleremo di come è stato implementato l'algrotimo genetico che ha permesso l'apprendimento di una strategia di pulizia alle coppie che saranno impegnate sulla mappa.

3.1 Algoritmi Genetici: schema generale

Gli *Algoritmi Genetici* sono algoritmi spesso usati in tutte quelle situazioni nelle quali risulta difficile andare a progettare una determinata strategia che ci permetta di arrivare ad una soluzione. Questi algoritmi sono difatti usati per far sì che siano loro stessi ad evolvere il sistema in modo che esso arrivi autonomamente ad una soluzione. Uno dei campi nei quali sono maggiormente impiegati, infatti, sono i *Sistemi Complessi*.
Per far sì che **Robby** possa muoversi e pulire l'ambiente raccogliendo il maggior numero di lattine con il minor numero di mosse, non conoscendo a priori la posizione del robot stesso nella mappa, né delle lattine, non rimane che progettare un algoritmo genetico che permetta ai robot di apprendere una buona strategia.
La struttura utilizzata nel paper di riferimento [1] è la seguente:

- dapprima sono generati *POPULATION_SIZE* individui con un dna casuale, ovvero, ad ogni possibile vista è associata una mossa scelta a caso fra quelle possibili;
- dopodiché, per *NUM_GENERATIONS*:
  - viene calcolato il *valore di fitness* per ogni individuo assegnando un punteggio 10 nel caso in cui un robot raccolga una lattina, -1 se il robot tenta di raccogliere una lattina in una cella dove questa

non sia presente e -5 punti se il robot tenta di muoversi in una casella occupata da un ostacolo (muro). Questa operazione viene effettuata per *NUM_ACTIONS_PER_SESSIONS* passi e ripetuta per *SESSIONS_NUMBER* diverse sessioni. Ogni sessione prevede la generazione di una mappa nella quale le lattine sono posizionate casualmente nelle celle (ogni cella ha la medesima probabilità di ospitarne una) così come la posizione del robot è scelta casualmente. La mappa ha dimensione 10x10 con 10 lattine. Alla fine, ogni robot ottiene un valore di fitness mediato sul numero di sessioni. Il numero di azioni su ogni mappa è fissato a 200, così come il numero di sessioni;

- si procede poi ordinando gli individui in base al proprio valore di fitness;
- sono scelti due individui nella popolazione con una probabilità che varia linearmente in base all'ordinamento (l'individuo con il maggiore valore di fitness avrà probabilità più alta di essere scelto rispetto al secondo e così via...) e viene applicato il *crossover*. Questo procedimento genererà due nuovi figli ed è ripetuto fino alla generazione della nuova popolazione;
- una volta ottenuta la nuova generazione, ogni gene (azione corrispondente ad una vista) può essere mutato con una probabilità pari a *MUTATION_PROBABILITY*: 0.5%.

3.2 Collaborazione

Siccome il nostro obiettivo è quello di studiare la collaborazione fra più robot, nella mappa saranno presenti contemporaneamente 2 agenti. Questo farà sì che la grandezza POPULATION_SIZE non indichi il numero di robot nellpopolazione ma il numero di coppie. Dunque, il numero di entità sarà pari al popolazione ma il numero di coppie. Dunque, il numero di entità sarà pari al doppio.

3.3 Funzione di Fitness

La presenza di due robot nella mappa comporta anche la modifica dei come la funzione di fitness valuta gli individui. Il valore di fitness non riguarderà un singolo agente, ma la coppia.
Nel caso in cui i due robot abbiano la possibilità di vedersi, poi, sarà assegnato un valore negativo di -5 punti anche nel caso le due entità cerchino di occupare

la stessa cella, scontrandosi, così come avviene nel caso in cui un robot tenti di muoversi contro il muro. Nel caso in cui non si vedono ed entrambi occupano la stessa cella nella quale è presente una lattina e cercano di raccoglierla, il secondo robot non prenderà punteggio negativo in quanto la lattina è già stata raccolta dal primo. Non si avrà però neanche l'incremento del valore di fitness di 20 punti (10 per l'azione di raccolta del primo e 10 per l'azione di raccolta del secondo). In un caso simile sarà incrementato il valore di fitness di soli 10 punti in quanto entrambi hanno fatto l'azione corretta ma solo una lattina è stata raccolta.

3.4 Selezione dei Genitori

La tecnica da noi usata per generare la nuova popolazione non è esattamente quella descritta nel paper. Difatti, a seconda delle varie simulazioni, saranno copiati alcuni individui dalla vecchia alla nuova generazione senza passare tramite il crossover.

3.5 Crossover

La tecnica di crossover da noi usata è quella descritta nel paper, ovvero: è scelto un punto di taglio casuale nel dna dei due genitori per generare i figli. Avendo però delle coppie la strategia cambia lievemente: sono scelte due coppie come genitori dove, il primo Robby della prima coppia ed il primo Robby della seconda generano il primo figlio, mentre il secondo Robby della prima coppia ed il secondo della seconda coppia generano il secondo figlio. Per ognuno dei due figli, il punto di taglio nel dna dei genitori varia.

3.6 Mutazione Genetica

La mutazione genetica è fatta nella maniera standard, mutando i geni dei due robot generati in accordo con una percentuale fissata. Anche nel caso in cui siano copiati degli individui dalla vecchia alla nuova generazione (senza dunque l'uso del crossover), questi subiranno mutazioni.

Capitolo 4

Implementazione

TODO

4.1 ANSI C

TODO

4.2 MPI

TODO -¿ parlare dei seed

Capitolo 5

Risultati

In questo capitolo saranno presentati i risultati ottenuti nelle varie simulazioni.

5.1 Entità Invisibili - Vista a Croce

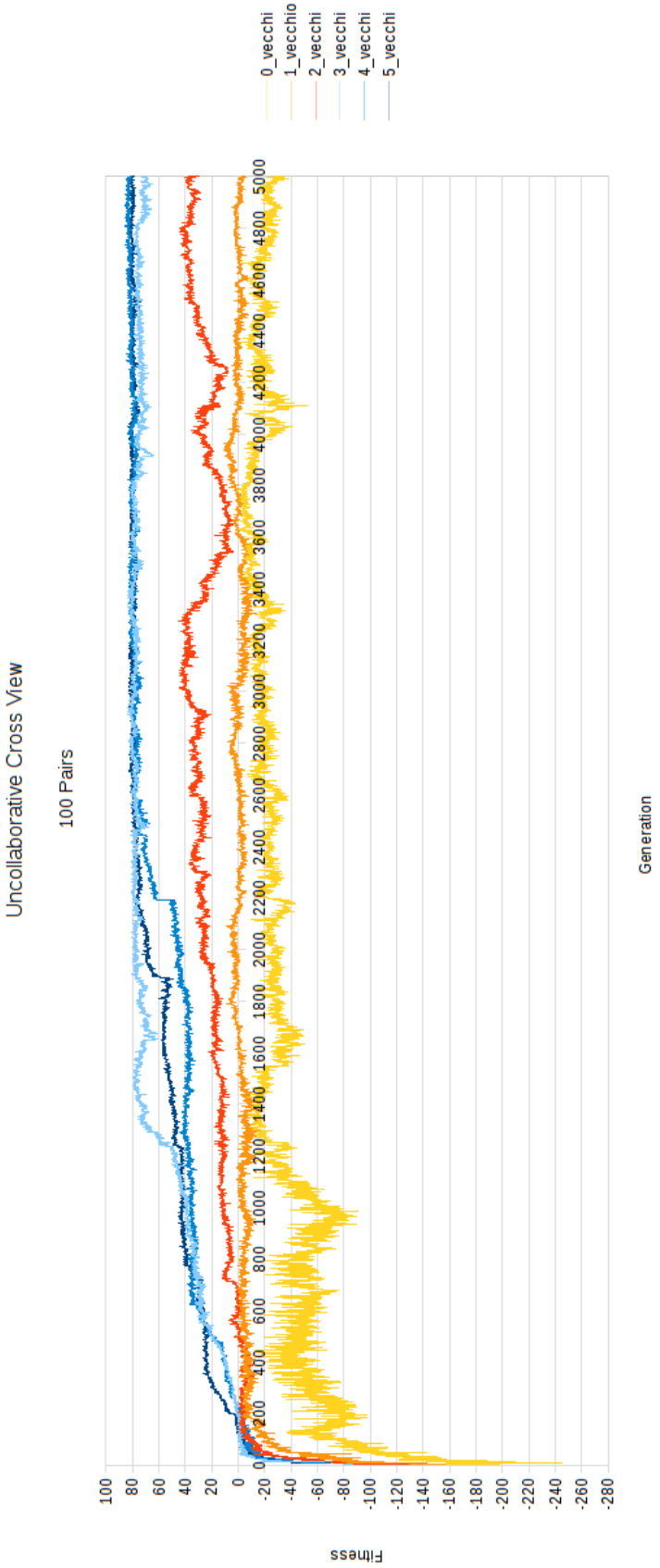

Figura 5.1: Vista a croce non collaborativa, 100 coppie

Capitolo 6

Conclusioni

Bibliografia

[1] Melanie Mitchell,
"*Robby, The Soda-Can-Collecting Robot*".
http://web.cecs.pdx.edu/~mm/ArtificialIntelligenceFall2008/Homework/Homework6.pdf.