Università di Bologna
Campus di Cesena

Corso di Laurea Magistrale in Ingegneria
e Scienze Informatiche

Sistemi Intelligenti Robotici

# An overview on RoboGen

*Giuseppe Pisano*
*Luca Passeri*
*Davide Giacomini*

Ottobre 2019

# Indice

# 1 Introduzione

Esistono numerose forme di adattamento, una di queste è l'adattamento di specie, dove vi è una specie che evolve mettendo in luce le caratteristiche più adatte all'ambiente in cui si trova. Questo è il principio su cui si basano gli algoritmi genetici. Questo tipologia di algoritmi prevede che vi siano individui di una popolazione caratterizzati da una specifica capacità di adattamento all'ambiente (fitness) che competono per ottenere risorse. Si interpretano gli individui come la soluzione al problema, la fitness rappresenta la prestazione del robot, mentre l'ambiente è il contesto dove gli individui vengono valutati. Non si conosce la descrizione del comportamento di ogni individuo, si sa solamente che tale descrizione produce un risultato che può essere valutato da un'apposita funzione.

Schematicamente la struttura di tali algoritmi può essere riassunta in tal modo:

1. codifica della struttura degli individui in modo formale (genotipo);
2. generazione casuale di una popolazione iniziale;
3. selezione degli individui migliori sulla base di varie valutazioni (fitness, etc);
4. manipolazione del materiale genetico mediante ricombinazioni e mutazioni per ottenere nuovi individui;
5. aggiornamento della popolazione.

Tale procedimento viene iterato fino al raggiungimento del risultato voluto (in termini di prestazioni), o di un limite di tempo.

Nel contesto della robotica l'idea è di avere un genoma che codifica una rete neuronale in un qualche modo (si possono modificare solo i pesi o anche la struttura della rete), per esempio nel caso di struttura fissa ciascuna configurazione di pesi è associata ad un individuo robot. La rete è collegata nei suoi input ai sensori del robot, mentre l'output ne controlla gli attuatori.

La ricerca dei parametri ottimali di questa rete viene fatta utilizzando un algoritmo genetico, con la sola particolarità che la valutazione (fitness) non è data da una semplice funzione, ma viene stimata a fronte della simulazione

del comportamento del robot nell'ambiente di riferimento.

Data una breve descrizione dell'ambito teorico si può introdurre lo strumento in esame: Robogen.

## 1.1 Robogen

RoboGen è una piattaforma open source mirata all'evoluzione dei robot sia dal punto di vista morfologico che da quello del software di controllo. Il suo fine primario è costruire dei robot facilmente realizzabili con l'ausilio della stampa 3D e di un piccolo insieme di componenti elettronici predefiniti. È composto principalmente da un motore evoluzionistico, responsabile del processo di evoluzione (generazione della popolazione, selezione, riproduzione, ...), e da un motore di simulazione fisico, incaricato invece della valutazione della fitness di un individuo.

Per quanto riguarda la composizione del corpo del robot (obbligatoria nel caso di sola evoluzione della mente) è possibile utilizzare una serie di elementi predefiniti. Tramite una sintassi specifica è possibile specificare come questi elementi sono connessi fra loro, andando a creare una struttura ad albero dove ogni nodo rappresenta una parte del corpo.

La mente invece è rappresentata da una rete neuronale composta da tre strati: input, nascosto e output. Nel primo tutti i neuroni sono collegati a un sensore e vanno in output a tutti i neuroni nascosti e a quelli di output. Quelli nascosti, attivati in base a una funzione sigmoide come quelli in output, si collegano a tutti i neuroni nascosti e a quelli in output, mentre gli ultimi fungono da collegamento agli attuatori del robot. Tutte le connessioni sono caratterizzate da un peso che è target del processo di evoluzione. Come lo è pure il bias dei neuroni con funzione di attivazione sigmoide. I neuroni inoltre possono essere definiti come oscillatori, cioè fare in modo che senza nessun input diano in output una oscillazione sinusoidale in funzione del tempo.

Il motore evoluzionistico ha due modalità di funzionamento: sola mente, che muta i pesi della rete neuronale, e corpo e mente, che contemporaneamente evolve anche la morfologia del robot. L'evoluzione del corpo viene attuata andando ad eseguire delle mutazioni all'albero morfologico quali per

esempio inserimento/rimozione di nodi e scambio di sotto alberi. Per quanto riguarda la mente, i vari parametri (pesi, errori, etc) sono mappati come una stringa di numeri reali. Su questa stringa sono applicate poi le varie variazioni come **single-point crossover**, possibile solo con una topologia del corpo fissa e con la probabilità di aggiungere neuroni nascosti a 0, e **mutazione**, applicata secondo il metodo della mutazione Gaussiana.

In fase di selezione (questo vale in caso di utilizzo dell'algoritmo di evoluzione semplice, non per HyperNEAT) possono essere utilizzate due strategie a scelta ( +/, ), seguite da una selezione a torneo fra la popolazione risultante. La fase di selezione è legata alla valutazione della fitness di un individuo. Questa è calcolata dal simulatore che, presa in input la descrizione del cervello e del corpo del robot, esegue la simulazione(completamente parametrizzabile secondo vari parametri che definiscono ambiente, rumore, time step, etc.) e procede alla valutazione della fitness secondo lo scenario stabilito. Questo scenario può essere definito dall'utilizzatore tramite una porzione di codice JavaScript.

È possibile eseguire la configurazione di tutti questi aspetti e visualizzare i risultati delle simulazioni attraverso un ambiente grafico definito appositamente.

Per la documentazione completa si rimanda al sito ufficiale.

# 2 L'esperimento

Dopo avere introdotto il contesto teorico e le principali funzionalità della piattaforma in esame non resta che definire il compito su cui poter effettuare la valutazione di tale strumento.

Scopo principale è la realizzazione di un robot capace di spostarsi il più velocemente possibile all'interno di un ambiente estremamente semplice: un piano infinito senza nessuna irregolarità od ostacolo. Non si pone nessun limite sulla struttura morfologica del robot. Unico vincolo è sull'utilizzo dell'algoritmo di evoluzione semplice piuttosto che HyperNEAT (disponibile all'interno di Robogen). Questa scelta è stata fatta per limitare lo spazio delle possibilità in fase di configurazione dell'algoritmo e potersi concentrare

sui meccanismi studiati durante il corso. Per quanto riguarda la funzione di fitness è stata scelta quella presente come impostazione predefinita all'interno delle piattaforma.

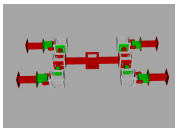
```
{
  distances : [],

  setupSimulation: function() {
    this.startPos = this.getRobot()
        .getCoreComponent().getRootPosition();
    return true;
  },

  endSimulation: function() {
    var minDistance = Number.MAX_VALUE;
    var bodyParts = this.getRobot().getBodyParts();
    for (var i = 0; i < bodyParts.length; i++) {
        var xDiff =
            (bodyParts[i].getRootPosition().x -
            this.startPos.x);
        var yDiff =
            (bodyParts[i].getRootPosition().y -
            this.startPos.y);
        var dist = Math.sqrt(Math.pow(xDiff,2) +
              Math.pow(yDiff,2));
        if (dist < minDistance) minDistance = dist;
    }
    this.distances.push(minDistance);
    return true;
  },

  getFitness: function() {
      var fitness = this.distances[0];
      for (var i = 1; i < this.distances.length; i++) {
        if (this.distances[i] < fitness)
          fitness = this.distances[i];
      }
      return fitness;
  },

}
```

Come si può vedere si vuole massimizzare la distanza percorsa, ottenuta come distanza fra il punto di partenza (definito all'inizio della simulazione) e

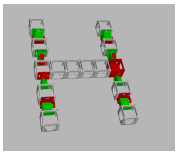
la distanza fra il punto più vicino a questo dopo lo spostamento. In caso di simulazioni multiple viene considerato come effettivo il valore di fitness peggiore fra quelli calcolati.

Dopo aver valutato le prestazioni dei vari robot proposti si procederà ad effettuare dei tentativi di evoluzione morfologica a partire dal migliore in modo da esplorare questa funzionalità.

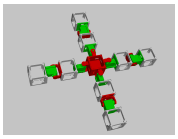
Vengono ora riportate le descrizioni delle tre morfologie create. Nella realizzazione si è dovuto tenere conto di un limite importante posto da Robogen: l'utilizzo di un numero massimo di otto componenti attivi. Questo limite impone la realizzazione di corpi con una complessità non molto elevata.



(a) Robot 1



(b) Robot 2



(c) Robot 3

Figura 1: I tre robot testati

## 2.1 Robot 1

Si è cercato in questo caso di ottenere un corpo che ricordasse quello di un quadrupede con le spalle snodabili.

```
1
2  0 CoreComponent Core 0
3    0 ParametricJoint Hinge1 0 0.05 0 0
4      0 FixedBrick Body1 1
5        1 ActiveHinge HipHHinge1 0
6          0 FixedBrick HipBrick1 1
7            2 ActiveHinge HipHinge1 1
8              0 ParametricJoint Leg1 1 0.04 0 0
9        2 ActiveHinge HipHHinge2 0
10         0 FixedBrick HipBrick2 1
11           1 ActiveHinge HipHinge2 1
12             0 ParametricJoint Leg2 1 0.04 0 0
13   1 ParametricJoint Hip2 0 0.05 0 0
```

```
    0 FixedBrick Body2 1
      1 ActiveHinge Hip3 0
        0 FixedBrick HipBrick3 0
          2 ActiveHinge HipHinge3 1
            0 ParametricJoint Leg3 1 0.04 0 0
      2 ActiveHinge Hip4 0
        0 FixedBrick HipBrick4 0
          1 ActiveHinge HipHinge4 1
            0 ParametricJoint Leg4 1 0.04 0 0
```

## 2.2 Robot 2

Anche questo robot è ispirato alla struttura di un quadrupede.

```
0 CoreComponent Core 0
  0 FixedBrick Body1 1
    0 FixedBrick Body2 1
      0 FixedBrick Body3 1
        0 FixedBrick Body4 1
          0 FixedBrick Body5 1
            1 ActiveHinge BackRightHip 0
              0 FixedBrick BackRightUpperLeg 1
                0 ActiveHinge BackRightKnee 0
                  0 FixedBrick BackRightLowerLeg 0
            2 ActiveHinge BackLeftHip 0
              0 FixedBrick BackLeftUpperLeg 1
                0 ActiveHinge BackLeftKnee 0
                  0 FixedBrick BackLeftLowerLeg 0
  2 ActiveHinge FrontRightHip 0
    0 FixedBrick FrontRightUpperLeg 1
      0 ActiveHinge FrontRightKnee 0
        0 FixedBrick FrontRightLowerLeg 0
  3 ActiveHinge FrontLeftHip 0
    0 FixedBrick FrontLeftUpperLeg 1
      0 ActiveHinge FrontLeftKnee 0
        0 FixedBrick FrontLeftLowerLeg 0
```

## 2.3 Robot 3

Anche in questo caso si è cercato di riprodurre una morfologia abbastanza nota, quella di una stella marina.

```
0 CoreComponent Core 0
  0 ActiveHinge Hip1 1
```

```
    0 FixedBrick UpperLeg1 1
      0 ActiveHinge Knee1 0
        0 FixedBrick LowerLeg1 0
  1 ActiveHinge Hip2 3
    0 FixedBrick UpperLeg2 1
      0 ActiveHinge Knee2 0
        0 FixedBrick LowerLeg2 0
  2 ActiveHinge Hip3 0
    0 FixedBrick UpperLeg3 1
      0 ActiveHinge Knee3 0
        0 FixedBrick LowerLeg3 0
  3 ActiveHinge Hip4 2
    0 FixedBrick UpperLeg4 1
      0 ActiveHinge Knee4 0
        0 FixedBrick LowerLeg4 0
```

# 3 Risultati

Nell'esecuzione dei test ci si è concentrati su cinque parametri principali. La dimensione della popolazione ($\mu$), la quantità dei figli generati ad ogni iterazione dell'algoritmo ($\lambda$), il numero di individuo che gareggiano in ogni torneo in fase di selezione della nuova popolazione (dimensione torneo), la probabilità di mutazione e quella di crossover.

Come principio di selezione si è visto come l'aggiunta ($\mu + \lambda$) performasse quasi sempre meglio della sostituzione e per tale motivo questo parametro non è riportato fra quelli indicativi. Anche il numero di generazioni è stato mantenuto fisso (le performance migliorano al suo aumentare quindi non è stato ritenuto significativo).

| $\mu$ | $\lambda$ | **dimensione torneo** | **mutazione** | **crossover** | **fitness** |
|---|---|---|---|---|---|
| 40 | 40 | 2 | 0.7 | 0.2 | 3.996 |
| 40 | 40 | 2 | 0.2 | 0.7 | 3.82264 |
| 40 | 40 | 2 | 0.7 | 0.7 | 4.55228 |
| 40 | 40 | 30 | 0.2 | 0.7 | 3.86783 |
| 40 | 40 | 10 | 0.2 | 0.7 | 2.71147 |

Tabella 1: Risultati Robot 1

| $\mu$ | $\lambda$ | **dimensione torneo** | **mutazione** | **crossover** | **fitness** |
|---|---|---|---|---|---|
| 40 | 40 | 2 | 0.7 | 0.2 | 3.24374 |
| 40 | 40 | 2 | 0.2 | 0.7 | 1.74421 |
| 40 | 40 | 2 | 0.7 | 0.7 | 2.06734 |
| 40 | 40 | 30 | 0.2 | 0.7 | 2.75865 |
| 40 | 40 | 10 | 0.2 | 0.7 | 2.48127 |

Tabella 2: Risultati Robot 2

| $\mu$ | $\lambda$ | **dimensione torneo** | **mutazione** | **crossover** | **fitness** |
|---|---|---|---|---|---|
| 40 | 40 | 2 | 0.7 | 0.2 | 4.78 |
| 40 | 40 | 2 | 0.2 | 0.7 | 4.779 |
| 40 | 40 | 2 | 0.7 | 0.7 | 5.37 |
| 40 | 40 | 30 | 0.2 | 0.7 | 5.416 |
| 40 | 40 | 10 | 0.2 | 0.7 | 3.956 |

Tabella 3: Risultati Robot 3

Si può notare come la mutazione sia un parametro importantissimo nel processo di evoluzione, in quanto porta della diversità nel genoma degli individui. Questo parametro assume ancora più rilevanza in caso di popolazione ridotta. Si è notato come il crossover invece porti ad una crescita lenta ma costante nel valore di fitness.

In modo da migliorare i risultati ottenuti è stata sfruttata una importante funzionalità offerta da RoboGen: la possibilità di riaddestrare una mente precedentemente addestrata modificando i parametri di configurazione dell'algoritmo. Infatti, specificando i parametri di configurazione iniziali della rete neuronale di controllo nel file di descrizione della morfologia, è possibile, semplicemente impostando il parametro **useBrainSeed** a **true** nel file di configurazione dell'algoritmo evolutivo, utilizzare questa configurazione nel processo di generazione della popolazione iniziale. Questa possibilità si è dimostrata estremamente efficace per superare le fasi di stagnazione nell'addestramento. Ha necessitato comunque di un lavoro di sviluppo in modo da creare uno script che convertisse automaticamente i risultati della simulazione (formattati in *json*) nel formato di descrizione della morfologia.

## 3.1 Evoluzione Morfologica

Essendosi il **Robot 3** rivelato il migliore in fase di apprendimento, questo è stato usato come base per il processo di evoluzione morfologica.

In RoboGen la modifica della morfologia del robot è resa possibile grazie al fatto che viene usato un insieme limitato di componenti standard ed esistono regole precise per assemblare un robot. È possibile quindi specificare alcuni parametri che regolano delle operazioni di base che agiscono sulla modifica della struttura con una certa probabilità, tali operazioni sono:

- pNodeInsert: probabilità di inserire un nuovo nodo
- pSubtreeRemove: probabilità di rimuovere una porzione del corpo
- pSubtreeDuplicate: probabilità di duplicare una porzione del corpo
- pSubtreeSwap: probabilità di scambiare due porzioni del corpo
- pNodeRemove: probabilità di rimuovere un nodo
- pParameterModify: probabilità di modificare un parametro di un componente parametrico

Nelle simulazioni effettuate il valore di tutti questi parametri era pari a 0.1, mentre la probabilità della mutazione della mente era 0.2. Per quanto riguarda il crossover esso deve essere mantenuto a 0 nel caso di evoluzione della morfologia, perciò è stato utilizzato, come alternativa, il parametro relativo alla probabilità do aggiunta di un neurone nel livello nascosto (pAddHiddenNeuron) impostato a 0.05.

I risultati di questo esperimento sono stati sorprendenti, il robot in questo caso ha aumentato notevolmente le performance raggiungendo un fitness value pari a **7.924**.

Ciò che più stupisce in questo caso è il fatto che il robot abbia modificato in modo sostanziale la propria struttura eliminando ogni vincolo di simmetria di partenza (figura 2), mentre il suo movimento risulta particolarmente efficace nonostante sia privo di semantica.

In questo caso sia il corpo che la mente del robot sono il risultato del processo di adattamento di tale specie all'ambiente e da questo esperimento emerge che avendo un grado di libertà maggiore (potendo modificare la morfologia) è più facile che vi siano individui che meglio si adattano a tale ambiente.

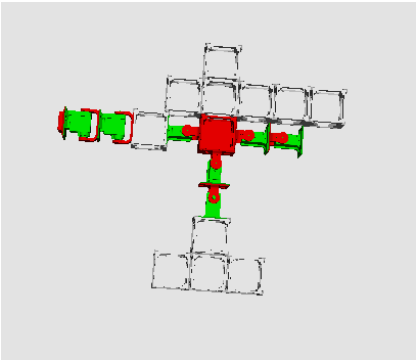


Figura 2: Individuo più performante risultante dall'esperimento di evoluzione con modifica della morfologia

# 4 Conclusioni

RoboGen si è rivelato un ottimo strumento da un punto di vista puramente didattico, essendo capace di rendere l'ambito della robotica genetica facilmente sperimentabile. Viene infatti distribuito con un ambiente grafico estremamente intuitivo, attraverso il quale in poco tempo è possibile avere i primi risultati. Anche da un punto di vista della configurabilità del processo evolutivo si è dimostrato uno strumento ottimo, permettendo di sperimentare con tutti i meccanismi più noti in letteratura e offrendo anche delle alternative più avanzate (HyperNEAT) con cui migliorare i risultati del processo. Unica nota negativa è la stabilità di questo tool. Infatti è possibile sperimentare spesso delle interruzioni nell'esecuzione o dei rallentamenti che lo rendono inusabile. Anche il processo di apprendimento è inficiato da questo problema, infatti non è stato possibile superare un certo numero di generazioni evolutive (circa 300/400) senza incorrere in un collasso del programma.

Altro problema riscontrato è la mancanza di feedback per l'utente per eventuali errori nella definizione della morfologia del robot. Infatti, nonostante il formato di scrittura sia banale, in caso di errore risulta abbastanza complicato localizzare l'errore. Il consiglio quindi sarebbe di utilizzare un formato più strutturato (es. yaml o json) oppure fornire maggiori informazioni sui problemi rilevati. Sarebbe poi comodo avere la possibilità tramite l'interfaccia di riprendere una simulazione interrotta precedentemente.