

Progetto Fondamenti di Intelligenza Artificiale

2020/2021

|  |  |
| --- | --- |
| Felice De Chiara | 0512105866 |
| Vincenzo Emanuele Martone | 0512105758 |
| Antonio Russo | 0512106058 |
| Alfonso Graziano | 0512105776 |

Sommario

[Definizione del problema 3](#_Toc59640357)

[Specifica dell’ambiente 3](#_Toc59640358)

[Soluzione 4](#_Toc59640359)

[Il confronto con OPT 4](#_Toc59640360)

[Come trovare la soluzione ottima? 4](#_Toc59640361)

[Come valutare una configurazione dell’algoritmo genetico con **OPT**? 4](#_Toc59640362)

[Parametri 5](#_Toc59640363)

[Selezione degli individui 5](#_Toc59640364)

[Mutazione 5](#_Toc59640365)

[Crossover 5](#_Toc59640366)

[Funzione di fitness 5](#_Toc59640367)

[Stopping condition 7](#_Toc59640368)

[Passi dell’algoritmo 7](#_Toc59640369)

[Reiterazione 7](#_Toc59640370)

[Testing dei parametri scelti 7](#_Toc59640371)

[Quante volte eseguire l’algoritmo genetico? 7](#_Toc59640372)

[Quanto deve durare ogni iterazione? 7](#_Toc59640373)

[Perché abbiamo scelto 6 come dimensione dell’individuo? 8](#_Toc59640374)

[Size della popolazione 8](#_Toc59640375)

[Perché abbiamo scelto il single point crossover? 8](#_Toc59640376)

## Definizione del problema

Sito che permette all'utente di navigare tra le schede tecniche di vari cellulari, e che dispone di un tool che consiglia un cellulare all'utente, sulla base di un budget e sulla base di quanto all'utente importino determinate caratteristiche (ad esempio l'utente dovrà dire su una scala da 1 a 10 quanto gli importano batteria, fotocamera, display e prestazioni). Il dataset è facilmente reperibile grazie ad un semplice tool di scraping. Per quanto riguarda il metro di paragone per confrontare le caratteristiche dei cellulari, abbiamo pensato di introdurre un sistema di punteggi che i recensori (che nel nostro dominio applicativo sono un team di professionisti) assegnano alle caratteristiche stesse e sono dunque parte integrante della scheda tecnica del prodotto; in caso di pareggio su determinate caratteristiche, si terrà conto delle recensioni degli utenti.

## Specifica dell’ambiente

* Performance
  + L’algoritmo considera i seguenti parametri:
    - Minimizzare lo scarto di:
      * Battery
      * Performance
      * Display
      * Camera
    - Massimizzare la data di uscita
      * Es: a parità di altri parametri, uno smartphone uscito nel 2019 sarà scelto al posto di uno uscito nel 2016
* Environment
  + Completamente osservabile
    - L’agente conosce in ogni momento tutto lo spazio degli stati
    - Nello specifico in ogni momento tiene in memoria la lista delle specifiche degli smartphone
  + Agente singolo
  + Deterministico
    - Lo stato è deterministico in quanto non ci sono azioni esterne che influenzano il prossimo stato. Lo stato successivo infatti deriva dall’azione e dallo stato corrente
  + Discreto
    - Il numero di percezioni dell’agente è limitato
    - Nel nostro caso abbiamo un numero discreto di specifiche, azioni e percezioni
  + Statico
    - L’ambiente è statico in quanto l’agente elabora su un’istanza dell’insieme delle schede tecniche aggiornate fino a quel momento
  + Episodico
* Actuators
  + Mostrare a video un set di “specs” conformi
* Sensors
  + Input di campi => range float da 1 a 10
  + Database
    - Noi leggiamo i dati da un form e il sistema manda il dataset dal db

## Soluzione

Dopo aver formulato il problema abbiamo capito che ci serviva un algoritmo di ottimizzazione. Abbiamo scartato gli algoritmi che lavorano sui grafi, in quanto tra gli elementi del nostro dataset non ci sono connessioni evidenti e questo ci ha fatto scartare l’ipotesi di rappresentare il problema come un grafo. Dato che il problema è offline abbiamo scartato gli algoritmi di ricerca online. Non avendo un ambiente multi agente abbiamo scartato anche le soluzioni con la teoria dei giochi.

L’opzione più promettente ci è sembrata quella di utilizzare un **algoritmo genetico** proprio perché le sue caratteristiche sono simili alle caratteristiche del problema.

Il nostro obiettivo infatti è quello di restituire un insieme di smartphone che meglio approssimano le caratteristiche richieste in input dall’utente. Nello specifico le caratteristiche sono: **battery, performance, display, camera**.

Pertanto un **individuo** non è altro che un insieme di schede tecniche (banalmente un array formato da n schede tecniche).

## Il confronto con OPT

Durante lo sviluppo dell’algoritmo genetico abbiamo cercato un modo per stabilire quanto fosse buona una configurazione di parametri. Abbiamo quindi pensato che nel caso specifico ci è possibile calcolare la soluzione ottima senza troppe difficoltà dato che il dataset si presta per un’operazione di ricerca lineare.

### Come trovare la soluzione ottima?

La soluzione ottima è composta da un individuo che minimizza lo scarto rispetto ai parametri presi in input. Sarà quindi composta dalle x schede tecniche che più si avvicinano a quei parametri.   
Procediamo quindi scorrendo tutta la lista e calcoliamo il **valore di similitudine** in base ai parametri e ordiniamo la lista in maniera non decrescente basandoci su questo valore. La soluzione ottima (OPT) è formata dal set dei primi x individui della lista ottenuta.

### Come valutare una configurazione dell’algoritmo genetico con **OPT**?

Data una configurazione dell’algoritmo genetico, lo eseguiamo un numero y di volte. Da ogni iterazione otteniamo un valore minimo di fit (in base ai parametri passati) e al termine ne effettuiamo la **media**.

Questa media verrà confrontata con il valore di fit di OPT ottenuto sommando tutti i valori di fit delle schede tecniche all’interno del set di OPT.

Per calcolare il grado di ottimalità di una determinata configurazione dell’algoritmo genetico, non facciamo altro che valutare la sua vicinanza al valore di OPT.

Es. OPT = 3.12 | G1 = 4.7   
La vicinanza è G1-OPT = 4.7-3.12 = 1.58

OPT = 3.12 | G2 = 5.3   
La vicinanza è G2-OPT = 5.3-3.12 = 2.18

Viene preferita la configurazione G1, in quanto lo scarto è quello che più si avvicina a OPT

Nella prossima sezione del documento analizzeremo tutti i parametri utilizzati e tutti i test effettuati

## Parametri

### Selezione degli individui

Al momento l’unico algoritmo di selezione utilizzato è il Truncation. Vengono scelti i primi n individui ordinati in base alla minimizzazione della funzione di fitting.

Per aggiungere una componente di casualità eventualmente si potrebbe valutare l’utilizzo di un algoritmo Roulette Wheel

#### Elitismo

Dato che facciamo una selezione in base alla funzione di fit non decrescente, questo ci fa implementare l’elitismo in quanto gli x migliori genitori vengono propoagati nelle generazioni successive.

### Mutazione

La mutazione avviene tramite Random Resetting. All’interno dell’individuo viene iniettato una scheda tecnica randomica in una posizione random dopo aver effettuato il crossover

### Crossover

Il crossover può essere di diversi tipi. Nell’algoritmo genetico al momento è implementato un crossover di tipo Single Point.

### Funzione di fitness

Originariamente il problema era pensato come multiobiettivo. Vogliamo infatti minimizzare lo scarto di tutti i parametri. Per semplificare abbiamo utilizzato un whole approach per cui abbiamo fuso tutte le funzioni in una singola funzione di fitness.

La prima versione della funzione di fitness era una semplice somma lineare di tutti i parametri (performance, battery, camera, display). Per individuare la funzione che più si avvicini a OPT, abbiamo svolto diversi test modificando la funzione stessa. Sorge, a questo punto, il problema di confrontare diverse funzioni tra di loro, in quanto, ad esempio, un paragone tra il risultato di una funzione lineare e il risultato di una funzione esponenziale non risulta immediato. Per ovviare a questo problema al termine della computazione delle diverse funzioni, abbiamo effettuato il confronto tra i risultati (quindi gli individui generati) “normalizzandoli” con una funzione lineare (se ad esempio si utilizza una funzione lineare ed una funzione esponenziale, dopo aver calcolato i risultati si applica la funzione lineare sull’output della funzione esponenziale in modo tale da ottenere un valore confrontabile con quello generato dalla funzione lineare). Di seguito sono riportati diversi test:

Una somma lineare è fallimentare in quanto non scarta abbastanza velocemente i risultati più “scarsi”.

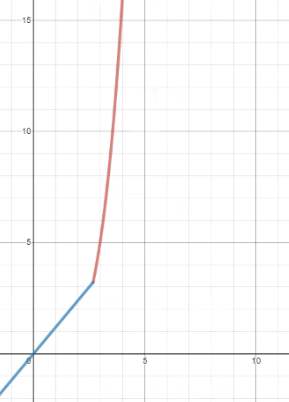
F(n) = scartoPerformance + scartoBattery + scartoCamera + scartoDisplay

Abbiamo poi pensato ad una somma di tutti i parametri a cui moltiplichiamo una costante:  
F(n) = scartoPerformance\*c + scartoBattery\*c + scartoCamera\*c + scartoDisplay\*c

A questo punto la funzione di fitness approssimava meglio ma ancora i risultati non venivano divisi abbastanza bene. Abbiamo quindi utilizzato una funzione esponenziale:

F(n) = scartoPerformance^d + scartoBattery^d + scartoCamera^d + scartoDisplay^d

Questo ha portato ad un bug in quanto per valori di scarto minori di 1 il quadrato è minore dello scarto (es 0.2^2 = 0.04). Abbiamo risolto quindi rendendo la funzione composta. La funzione al momento è quindi la seguente:   
scarto = x>2.67 ? x\*1.2 : (x/2)^4 (funzione expand che opera sugli scarti di batteria, performance, display, camera)

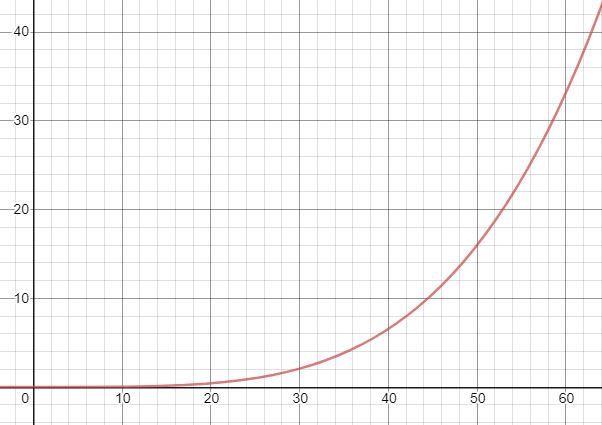


Una volta fatte queste considerazioni abbiamo riflettuto sull’aggiungere un’altra misura importante: la data d’uscita. Nello specifico abbiamo a disposizione il mese e l’anno di uscita del device. Procediamo quindi a sottrarre la data attuale da quella d’uscita per sapere da quanti mesi è uscito lo smartphone.

Anche in questo caso bisogna capire come integrare questa nuova variabile all’interno della funzione di fitting per implementare l’aging. A parità di prestazioni infatti, un dispositivo uscito qualche anno fa oggi ha sicuramente prestazioni inferiori.

Proprio in merito all’inserimento della componente “Data” all’interno della funzione di fitting, abbiamo notato un degrado dei risultati dell’algoritmo nel momento in cui è stata implementata, in quanto l’algoritmo vincolava in maniera troppo stretta i risultati alla data d’uscita. Ci siamo, dunque, resi conto del fatto che fosse necessario capire quale fosse il peso da dare alla “Data” sulla scrematura dei nostri risultati.

Tra i diversi test effettuati siamo arrivati a valutare la seguente funzione di aging che ci consente di scartare i dispositivi più vecchi: y = (x/25)^4



Il valore di fit complessivo dell’individuo è la media dei valori di fit ottenuti dalle singole schede tecniche.  
Inizialmente utilizzavamo una somma lineare degli scarti, ma abbiamo deciso di utilizzare la media in modo da slegare il valore dal numero di geni presenti nell’individuo. In questo modo in una futura implementazione gli individui potranno anche avere un numero variabile di geni.

### Stopping condition

Abbiamo testato diverse stopping condition. Alla fine abbiamo optato per monitorare il tempo di esecuzione e stoppare l’algoritmo dopo x millisecondi. Dato il numero molto limitato presente nel dataset (al più circa un migliaio di device) l’algoritmo arriva molto velocemente a convergenza.

Abbiamo scelto questo parametro anche basandoci su un requisito non funzionale all’interno del progetto di Ingegneria del Software per cui il tool DoraIA deve ritornare un valore entro max 5 secondi.

### Passi dell’algoritmo

1. Costruire una popolazione iniziale contenente tutti gli individui del DataSet
2. Calcolare il loro valore di fitness
3. Selezionare un sottoinsieme di individui della popolazione iniziale per ammetterli nel Mating Pool tramite Truncation (ad esempio usare un valore di soglia per far morire gli individui che superano quel valore)
4. Gli individui precedentemente selezionati vengono fatti accoppiare tramite Single Point Crossover
5. Gli individui ottenuti dal Crossover vengono mutati tramite una Random Resetting (il gene da sostituire viene preso da un pool di spec che si ha all’inizio e ogni gene che viene sostituito viene messo nel pool. Ciò risolve il problema della validità degli individui generati dal crossover) e si ritorna al passo 3
6. Quando si arriva a terminazione viene ritornato l’individuo che minimizza il valore di fit

### Reiterazione

Avendo effettuato vari test abbiamo notato che l’algoritmo performa meglio se ne vengono eseguite più istanze. Abbiamo quindi scelto di eseguire l’algoritmo n volte e scegliere l’individuo che minimizza la varianza. Utilizziamo la varianza come parametro di valutazione in quanto vogliamo ottenere un individuo che al suo interno contenga risultati omogenei (le funzioni di fit devono ritornare valori simili). Preferiamo una soluzione subottima rispetto ad un’altra soluzione con media minore ma varianza maggiore.

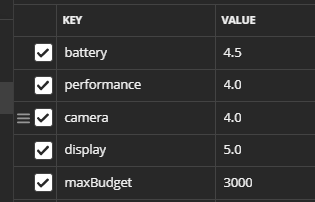
## Testing dei parametri scelti

### Quante volte eseguire l’algoritmo genetico?

Abbiamo effettuato diversi testi in merito, sapevamo che far girare l’algoritmo genetico diverse volte era la scelta giusta. Non sapevamo però quale sarebbe stato il numero migliore che ci dava il giusto equilibrio tra velocità dell’algoritmo e precisione. Siamo partiti da un valore di **5 run** e abbiamo ottenuto un valore medio di fit di 120.86. Abbiamo incrementato a **10 run** ottenendo un valore di fit medio di 113.05. Il miglior individuo ha un valore di fit di 99.09. Con **20 iterazioni** abbiamo ottenuto un valore medio di fit 109.09 e un individuo migliore con un valore di 98.96. L’individuo “vincitore” è quello che tra tutti i candidati **minimizza la varianza** tra i valori di fit di ogni scheda tecnica. Possiamo dirci soddisfatti dei risultati raggiunti fino a questo momento.

### Quanto deve durare ogni iterazione?

I dati di test utilizzati per questo esperimento sono i seguenti:



Come metro di paragone usiamo il valore di fit medio degli individui che sono stati selezionati come migliori.

Per prima cosa abbiamo testato delle **iterazioni di lunghezza 200ms**. Questi hanno portato ad una media di 112.181. Abbiamo quindi portato le iterazioni a **400ms**, il valore medio (calcolato su 20 iterazioni) è 109.35. Aumentando il tempo a **600ms** non otteniamo miglioramenti (anzi, si peggiora leggermente andando a 111.92). Testiamo quindi un valore molto basso di **100ms** per verificare l’eventuale presenza di problemi. Otteniamo un valore di 114. Procediamo abbassando ancora il valore fino a 50ms e otteniamo un valore di 116.51. Dopo altri test di questo tipo siamo arrivai al valore di 200ms che ci garantisce un buon equilibrio tra prestazioni e qualità dei risultati. Aumentando infatti il tempo, non abbiamo riscontrato miglioramenti sostanziali. L’algoritmo arriva molto velocemente a convergenza dato il numero molto ridotto di device presenti.

N.B. I dati riportati sono una media di più run effettuate dall’algoritmo

### Perché abbiamo scelto 6 come dimensione dell’individuo?

La scelta della grandezza dell’individuo deriva da una decisione presa in fase di design del progetto di ingegneria del software dove appunto mostravamo 6 risultati alla volta. Tale numero ci è sembrato un buon compromesso per offrire all’utente una scelta variegata ma non confusionaria.

### Size della popolazione

Data la grandezza molto ridotta del dataset abbiamo testato 3 tipi di popolazione: con al più 40 individui, con al più 100 individui, con al più 140 individui. Abbiamo notato che la popolazione da max 40 individui molte volte dava risultati più scarsi delle altre due. La popolazione da max 140 individui però era più “lenta” nell’arrivare ad un buon risultato. Abbiamo quindi valutato di scegliere la popolazione da max 100 individui che è una via di mezzo.

### Perché abbiamo scelto il single point crossover?

Abbiamo scelto un single point crossover (e non ad esempio un double point crossover) data la natura non posizionale del problema. Dato che alla singola scheda tecnica all’interno dell’individuo non corrisponde un peso, o comunque non è attribuito un valore particolare in base alla posizione, e dato che le schede tecniche non sono legate tra di loro da qualche parametro, la scelta di un algoritmo di crossover anziché di un altro è indifferente.