Sommario

[1. Introduzione e contesto del problema 1](#_Toc67580454)

[2. Come approcciarsi al problema della scelta degli utenti verificatori? Una proposta. 2](#_Toc67580455)

[3. Algoritmo genetico: un possibile setup. 2](#_Toc67580456)

[3.1. Codifica degli individui 3](#_Toc67580457)

[3.2. Funzione di fitness 3](#_Toc67580458)

[3.3. Operatori genetici 4](#_Toc67580459)

[3.3.1. Selezione 4](#_Toc67580460)

[3.3.2. Crossover 4](#_Toc67580461)

[3.3.3. Mutazione 5](#_Toc67580462)

[3.4. Condizione di terminazione 5](#_Toc67580463)

[4. Appendice – Il problema della posizione geografica e strategie di preprocessing 5](#_Toc67580464)

# Introduzione e contesto del problema

Shall we go è una piattaforma che fa del suo basarsi sulla community degli utenti dei suoi servizi la sua base. Data la sua natura, questa è stata una scelta quasi obbligata da compiere, descritta in maniera sommaria anche nel documento contenente il problem statement. Se è vero che esistono servizi come Moovit o lo stesso Google Maps che forniscono un’ottima mole di dati riguardante il *topic* che Shall we Go approccia, ovvero quello dei trasporti e tutto ciò che riguarda questo mondo, è altrettanto vero che questi ultimi presentano dei limiti che si fanno evidenti soprattutto quando si vanno a trattare aziende di trasporto medio-piccole che non comunicano (o non possono comunicare) i dati sulla loro organizzazione delle fermate e delle corse che espletano.

Nel caso specifico dell’Agro Nocerino-Sarnese, per presentare un esempio pratico, operano diverse aziende di trasporto private (come la ex *Buonotourist* o la *Leonetti & Gallucci*). Nelle due applicazioni sopracitate, tuttavia, i loro punti di fermata (che molto spesso differiscono da quelli delle aziende statali, come Busitalia Campania e la SITA) e lo scheduling delle loro corse non sono minimamente menzionati in quanto queste ultime verosimilmente non li comunicano ai gestori delle piattaforme. Ed è in questo caso che entra in gioco il concetto di crowdsourcing su cui Shall we Go si basa.

Gli utenti, di conseguenza, giocano un ruolo fondamentale per il funzionamento ed il successo della piattaforma, perché possono applicare le loro conoscenze al servizio degli altri segnalando la presenza di fermate in un determinato punto e le linee che le sfruttano.

Tuttavia, anche questo approccio non è esente da problemi di natura pratica. Quando si mette nelle mani del pubblico la possibilità di contribuire a dei dati che poi sono destinati ad essere consultati anche da altre persone, non è sempre possibile assumere la buona fede di chi effettua delle segnalazioni, in quanto potrebbero verificarsi episodi di *trolling* o di vero e proprio vandalismo, come accade di tanto in tanto sulle voci dell’enciclopedia online Wikipedia. È stato quindi da subito necessario pensare ad un qualche tipo di contromisura a questo fenomeno. La prima feature che è stata presa in considerazione è quella che consiste nel prevedere una sorta di “carriera” all’interno della piattaforma che funziona in modo simile al social *Reddit* in cui gli utenti sono caratterizzati da un certo livello di “Karma”, che certifica la loro attività derivata dalla permanenza sulla piattaforma ed il contributo che esso apporta (nel caso di Reddit, in termini di post, upvote o commenti ai post nelle varie boards). Vien da sé che quanto più un utente contribuisce alla piattaforma, tanto più il suo *karma* aumenta. Un approccio del genere potrebbe essere adottato anche in Shall we Go. Ad esempio, quando un utente effettua la segnalazione di una fermata in un determinato punto, essa verrà analizzata da uno (o più) utenti “fidati” in base a diversi fattori come karma, tempo di permanenza sulla piattaforma e posizione geografica (uno dei temi principali che saranno affrontati più avanti in questo documento). Nel caso questa segnalazione si riveli corretta, essa verrà integrata nella piattaforma, chi segnala verrà notificato e “premiato” con aumento del proprio karma.

Il karma di un utente, quindi, può fungere da discriminante sulla sua possibilità di avere “potere decisionale” sulle segnalazioni che arrivano, garantendo quindi il mantenimento di un certo livello di qualità sui dati presenti sulla piattaforma.

# Come approcciarsi al problema della scelta degli utenti verificatori? Una proposta.

Fino ad ora, tuttavia, si è discusso molto sul problema di come garantire la qualità dei dati ma poco dell’infrastruttura “under the hood”. In particolare, sorge immediatamente la questione prettamente tecnica del come andare a scegliere chi devono essere gli incaricati ad esaminare le segnalazioni di ciascun utente. Banalmente, un utente di Milano ha una probabilità decisamente minore di saper analizzare la segnalazione di una fermata che si trova nel comune di Fisciano rispetto ad un utente che abita in quel comune o comunque poco lontano da lì.

Oltre ad un criterio prettamente tecnico (che è quello della posizione geografica del verificatore), c’è poi da affrontare tutto il discorso sul karma e sul periodo di appartenenza alla piattaforma. Una situazione del genere, specialmente se ci si proietta in un contesto con molti utenti e quindi su larga scala, non sempre è approcciabile con una semplice ricerca lineare tra tutti gli utenti della piattaforma, al netto di un possibile lavoro di preprocessing (che sarà discusso in un’appendice di questo documento e che comunque non sempre può andare a buon fine). Nel contesto del corso di *Fondamenti di Intelligenza Artificiale*, tenuto per la prima volta nel corrente anno accademico (2020/2021) dai prof. Fabio Palomba e Fabio Narducci, sono stati discussi vari tipi di algoritmi di ottimizzazione del processo di ricerca, compresa una classe di algoritmi che si presta molto bene a questo tipo di problema: gli Algoritmi Genetici (da qui in avanti, *GA*).

I GA, più precisamente, sono una *metaeuristica* generica, ovvero forniscono una serie di “regole” che permettono di definire degli algoritmi facilmente adattabili ad un numero molto alto di tipologie di problemi e che si basano sul concetto di Evoluzione come inteso da Darwin nel XIX secolo.

L’idea alla base dei GA è quella di andare a costruire inizialmente un insieme di soluzioni ad un problema, i cosiddetti “individui” di una popolazione, che man mano verranno fatti “evolvere” usando sostanzialmente tre operatori genetici: **selezione**, **crossover** e **mutazione**, ottenendo man mano soluzioni sempre diverse e migliori in termini di una metrica chiamata **fitness**, che esprime in termini quantitativi la bontà di una soluzione.

Il vantaggio del paradigma GA, essendo una metaeuristica, risiede proprio nella sua adattabilità a molti problemi, tra cui vi è proprio quello affrontato da Shall we Go in questo contesto. Di seguito verrà proposto un possibile setup, fase che rappresenta una delle difficoltà tipiche della progettazione dell’algoritmo, altrimenti molto intuitivo da applicare

# Algoritmo genetico: un possibile setup.

Si è detto che i principali *step* da affrontare durante la progettazione di un GA sono:

* Come codificare gli individui che compongono una popolazione
* Come stabilire una *funzione di fitness*
* Come applicare i tre *operatori genetici* descritti in precedenza (selezione, crossover e mutazione di un individuo)
* Come stabilire una *condizione di terminazione,* ovvero quando far terminare il processo di “evoluzione”.

## 3.1. Codifica degli individui

Nel contesto degli algoritmi genetici, possiamo interpretare un individuo come un insieme di geni, che lo caratterizzano e permettono la creazione di nuovi individui tramite l’applicazione dei tre operatori genetici. Calandoci nel contesto del problema che si vuole affrontare, si nota subito che un individuo, contrariamente a quanto ci si potrebbe aspettare, non può coincidere con un singolo utente “candidato” alla verifica delle segnalazioni, in quanto composto da caratteristiche che non possono essere mutate o incrociate tra di loro (banalmente, si creerebbe un utente che nella realtà non esiste, di fatto vanificando lo sforzo dell’algoritmo). È necessario quindi “salire di livello”. Si potrebbe provare a passare dalla relazione *singolo utente = individuo* a quella, più generale, *gruppo di utenti = individuo*. Questo approccio permette di poter effettuare in tutta libertà (e senza perdita di coerenza) le operazioni di mutazione e crossover. Ognuno degli individui, quindi, avrà un numero fissato *n* di geni che consistono in singoli utenti, che sono caratterizzati da quanto discusso in precedenza.

## 3.2. Funzione di fitness

La funzione di fitness, nel contesto degli algoritmi genetici, fornisce una stima quantitativa della bontà della soluzione di un certo problema. Nel caso di Shall we Go sarebbe desiderabile andare a creare una funzione che permetta di capire, sulla base delle caratteristiche di un gruppo di utenti, quanto questi ultimi possano essere reputati adatti a verificare una certa segnalazione ed eventualmente assegnargliela.

Come accennato in precedenza, i tratti distintivi degli utenti nel contesto di una certa segnalazione (importante) sono sostanzialmente tre:

* Posizione geografica (e, di conseguenza, distanza dal luogo della segnalazione)
* Permanenza sulla piattaforma
* Karma accumulato con le segnalazioni di fermate, linee o avvenimenti vari

La funzione di fitness, dunque, deve tenere conto di queste tre caratteristiche ma allo stesso tempo assegnare un peso maggiore a quelle che vengono reputate più importanti. Ad esempio, potrebbe essere preferibile assegnare più peso alla posizione geografica in cui risiede un utente (e quindi alla sua distanza approssimativa dal luogo della segnalazione) e al suo karma piuttosto che alla sua permanenza sulla piattaforma, che di conseguenza avrà un’importanza marginale (ma non nulla) sulla valutazione della fitness totale. C’è comunque da tener presente che la fitness deve essere calcolata sull’individuo e non sul singolo utente. Per ovviare a questo potenziale problema, si potrebbe pensare di verificare la fitness di ogni utente singolarmente ed eventualmente mediare tra i singoli valori. Questo approccio, tuttavia, non è esente da difetti. Si assuma che all’interno di un individuo ci siano due geni molto molto forti ed uno molto debole. In questo caso, la sua fitness sarà parecchio alta e quindi questo individuo sarà un ottimo candidato per essere la soluzione al problema, pur avendo al suo interno un utente che risulta poco adatto al suo compito.

Una soluzione a questa problematica risiederebbe nel compiere un lavoro di preprocessing iniziale sui geni degli individui e prevedere un sistema di priorità delle valutazioni delle segnalazioni.

Nello specifico, se due utenti X ed Y danno due valutazioni contrastanti ad una segnalazione (X dice Sì e Y dice No), potrebbe essere accettata l’opinione di chi dei due ha la fitness più alta. Questo lavoro, tuttavia, va al di fuori del lavoro dell’Algoritmo Genetico ed è stato menzionato solo per completezza.

Dal punto di vista matematico, la “fitness” **di un gene** viene calcolata come una media pesata di valori che sono in funzione delle tre caratteristiche sopra elencate. La crescita di queste funzioni, nello specifico, varia a seconda del valore di una determinata caratteristica.

* Nel caso della distanza geografica, più il valore è piccolo più la “fitness parziale” sarà alta. (Si preferiscono senz’altro persone che risiedono nella stessa zona del punto in cui è stata effettuata la segnalazione). Dal punto di vista matematico, questo termine della funzione di fitness potrebbe essere tradotto in una curva del tipo

.

Il parametro n non è stato ancora stabilito in quanto deve essere tale da rendere il valore di f(x) grande per x piccoli e piccolo per x grandi, senza incorrere una decrescita eccessiva per valori “intermedi”

* Nel caso del Karma, si potrebbe pensare ad una funzione che cambia comportamento: crescita lineare della funzione fino ad un determinato valore soglia, oltre la quale la funzione tende a crescere molto velocemente (più il karma è alto, maggiore sarà il valore: una sorta di “premio” per utenti con karma alto)
* Nel caso della durata della permanenza sulla piattaforma, non si prevedono particolari accorgimenti, quindi si potrebbe pensare ad una funzione che cresce linearmente. (Il valore sarà più alto se un utente è iscritto da più tempo). Tuttavia, questo fattore non ha peso particolare nel contesto del calcolo della fitness. Potrebbe servire solo come discriminante in caso di situazioni dubbie

I valori soglia saranno oggetto di verifica approfondita nel corso del tempo.

## 3.3. Operatori genetici

I tre operatori genetici sono il vero e proprio cuore di un algoritmo genetico perché forniscono un metodo per andare a diversificare la popolazione di individui, in modo del tutto simile a quanto avviene in natura.

### 3.3.1. Selezione

Durante la fase di **selezione,** si ha una sorta di “selezione naturale” degli individui: solo i più forti sopravvivono (in termini di fitness, nel nostro caso). Durante le lezioni sugli algoritmi genetici sono stati proposti diversi approcci per implementare questa fase: tra questi vi era la cosiddetta “**Roulette Wheel**”, dove chi ha la fitness più alta ha una più alta possibilità di accedere alla fase successiva. La si pensi come una vera e propria “roulette” che gira e sceglie gli individui ammessi nel mating pool, preludio della fase di crossover. Vien da sé che l’individuo che occupa un’area più grande di questa ipotetica roulette ha più possibilità di essere scelto.

### 3.3.2. Crossover

La fase di **crossover** prevede l’incrocio di due individui per crearne altri. Ciò permette alla popolazione di variare ed ottenere individui sempre migliori in termini di fitness. Questa fase, nello specifico, si ottiene andando a scambiare i geni di un individuo con quelli di un altro, utilizzando diversi approcci. Uno di questi consiste nel ***single point crossover:*** i geni di due individui vengono divisi in un punto fissato e le due parti vengono incrociate. Nello specifico:

L’individuo X viene diviso in due parti: X1X2 e l’individuo Y in Y1Y2. I due individui risultanti saranno quindi X1Y2 e Y1X2. Di questi due individui poi verrà calcolata la fitness, ottenendo una stima della loro “performance”.

### 3.3.3. Mutazione

L’ultimo operatore che caratterizza questo tipo di algoritmi consiste in quello di **mutazione.** La mutazione di un individuo consiste nel cambiamento pseudo-casuale di uno dei suoi geni con un altro. Ciò solitamente avviene dopo la fase di crossover. Nel caso di codifica degli individui sotto forma di stringhe binarie ciò si ottiene usando diverse tecniche quali bit flip (inversione degli 0 con gli 1 e viceversa) o magari anche permutazioni dei geni (rilevante in quanto il sistema binario è posizionale). Nel caso di Shall we Go, tuttavia, ciò non è possibile, in quanto i geni sono appunto composti da entità con delle caratteristiche ben precise: si andrebbero a creare degli individui inesistenti!

Uno tra gli approcci possibili consisterebbe nel sostituire un gene con un altro preso in maniera casuale tra quelli disponibili globalmente, preservando quindi l’integrità dell’individuo. Ciò però fa sorgere un altro problema. Se in un individuo si trovano due geni identici, la fitness potrebbe essere falsata (sia in positivo che in negativo, naturalmente). In questo caso si aprono due possibili strade: ripetere la mutazione fino a che non si trova un gene che non va in “conflitto” con gli altri oppure si decide di rendere la codifica di un individuo di lunghezza variabile.

Nel contesto di Shall we Go quest’ultimo approccio, seppur generalmente scoraggiato, potrebbe funzionare in quanto non è necessario che i valutatori siano sempre considerati in gruppi della stessa grandezza. La cosa importante è che sia presente un numero sufficiente di “opinioni” su una singola segnalazione (nello specifico, sarebbe desiderabile averne almeno **3**). Introducendo la variabilità della codifica sarà quindi naturalmente necessario andare a modificare di conseguenza l’operatore di crossover, rendendolo in grado di gestire individui del genere.

## 3.4. Condizione di terminazione

L’evoluzione, in qualche modo, deve terminare.

La caratteristica di questo tipo di algoritmi, tuttavia, è che potrebbero continuare all’infinito se non viene imposto un qualche tipo di discriminante che permette la terminazione del processo. Nello specifico, sarebbe desiderabile che l’evoluzione termini nel momento in cui ci si accorge che la fitness degli individui che vengono generati è minore rispetto a quella delle popolazioni precedenti. Il numero di generazioni che vengono generate senza che la fitness aumenti è stato fissato a **3**, superato il quale la computazione termina e viene restituito il miglior individuo della generazione immediatamente precedente all’inizio del calo. In questo modo, si evita che la computazione richieda troppo tempo e risorse anche nel caso ci si renda conto che non si riesce a fare meglio rispetto ad un determinato livello.

# Appendice – Il problema della posizione geografica e strategie di preprocessing

Nel contesto di Shall we Go, all’atto di registrarsi, all’utente è richiesto di indicare alla piattaforma il Comune, la Provincia e la Regione in cui vive (per il momento si prevede di supportare solamente l’Italia da questo punto di vista). Ciò, oltre a permettere alla piattaforma di notificargli avvisi o variazioni di servizio nella sua zona, rende possibile anche effettuare confronti tra il luogo geografico associato ad una segnalazione e la zona di residenza di un possibile *reviewer.* Infatti, tramite un servizio di geocoding come Nominatim appositamente predisposto su una macchina dedicata, un toponimo può essere convertito in una coppia (Latitudine, Longitudine) che tramite una formula matematica possono essere confrontate tra di loro (e comunque i linguaggi di programmazione come Java forniscono delle discrete API per compiere la stessa operazione in maniera semplice). Non essendo richiesta grossa precisione per questo scopo, il confronto può essere fatto sul nome del Comune. Questo dato sarà utilizzato dall’algoritmo genetico per computare una delle variabili che determinano la fitness, come descritto in precedenza. Per migliorare le prestazioni della ricerca dei reviewer si potrebbe pensare a due strategie euristiche di preprocessing. La prima consiste nel bypassare (quasi) del tutto l’esecuzione dell’algoritmo andando a considerare, nel caso la segnalazione che deve essere verificata venga effettuata in un comune in cui risiede un utente della piattaforma idoneo al compito. In questo caso, di conseguenza, lo si va a considerare in maniera preferenziale e se il suo livello di karma risulti superiore ad una certa soglia, lo si include tra i reviewer. In caso risultino più di tre utenti idonei residenti nel comune della segnalazione, l’algoritmo non viene eseguito e la computazione termina. La seconda strategia presa in considerazione consiste nel “facilitare” la vita all’algoritmo andando a ridurre il suo dominio ai soli utenti residenti nella provincia della segnalazione da verificare. Questo permette di tagliare fuori una grossa parte di utenza che con grossa probabilità non sarebbe in grado di valutare in maniera adeguata quella determinata segnalazione.