Documentazione

iTeam

« L'intelligenza è la capacità di evitare di fare il lavoro, pur riuscendo a farlo.»

- Linus Torvalds

Sommario

1. Premessa

iTeam si basa sul progetto di Ingegneria Del Software: Agency Formation che migliora i processi di:

- Reclutamento Candidati
- Formazione Dipendenti
- Gestione Materiale di Formazione

Inoltre permette la creazione di *Team* di dipendenti a cui è affidato un progetto. La creazione viene agevolata dall'intelligenza artificiale: iTeam.

2. Breve proposta di progetto

Sviluppo di un agente intelligente che sulla base dei requisiti di un progetto, riesca a formare un team adatto per quest'ultimo.

3. Sistema

Definizioni, acronimi e abbreviazioni:

- *TM*: Team Manager.
- *Tm*: Team member.
- *Soft-skill:* Skill interpersonali.
- *Geni:* Dipendenti.
- *Individuo:* Team.
- *Avg_Skill:* valore che indica la soluzione ottima, cioè 5.0.

3.1. Sistema Attuale

Allo stato corrente non è presente alcun sistema che gestisce la creazione automatica di un Team in base ai requisiti espressi dal TM. Quindi attualmente per creare un team, bisogna controllare le singole skill di ogni dipendente e scegliere i dipendenti migliori.

Con questo tipo di approccio sorgono diversi svantaggi come:

- Nel caso si abbia una lunga lista di dipendenti, avremo lunghi periodi per l'esaminazione di ogni futuro *Tm*.
- Possibilità di non scegliere correttamente il dipendente più adatto a lavorare in quel team.

3.2. Sistema Proposto

iTeam è un'AI che supporta l'attività di creazione di un Team, in modo tale da avere tempi molto più brevi per l'esaminazione del singolo dipendente. iTeam è capace di creare team di dipendenti in base ai requisiti del progetto da svolgere. I *Tm* verranno scelti in base alle skills richieste dal Progetto. Nel caso in cui venga richiesta una skill che un dipendente non possiede, ma avrà altre skills richieste, verrà comunque processato dall'AI. Il livello delle skill di un dipendente è segnato con un valore auto-

valutativo compreso tra: 0 come il più basso e 5 come il più alto. Nel caso in cui il livello della skill sia 0, il dipendente non verrà ritenuto in possesso di quella competenza, in quanto ciò sta ad indicare che il dipendente non ha più dimestichezza ("ha dimenticato") con quella skill.

4. Specifiche PEAS

4.1 Performance

Le performance dell'agente sono valutate attraverso il valore/punteggio del team restituito dall'AI.

Ex: Prendiamo in considerazione due team, A con valutazione **4.0** e B con valutazione **2.5**. iTeam sceglierà il team con valutazione più alta, in questo caso il team A.

4.2 Environment

Completamente osservabile:

L'AI è sempre a conoscenza di ogni specifica di ogni singolo dipendente.

Agente singolo:

L'ambiente presenta un singolo agente.

Stocastico:

Lo stato successivo del team non è completamente determinato dallo stato corrente ma è condizionato da una componente randomica.

Statico:

Quando l'agente è in funzione l'ambiente rimane invariato.

Episodico:

Le modifiche apportate all'ambiente sono indipendenti fra loro, quindi appartengono a due episodi atomici diversi.

Discreto:

Il numero di percezioni dell'agente è limitato in quanto ha un numero discreto di specifiche, azioni e percezioni possibili.

4.3 Actuators

Team suggerito sulle skill richieste e relativa pagina web.

4.4 Sensors

I sensori dell'agente consistono nei bottoni della pagina web.

5. Dataset

Il dataset che è stato utilizzato è stato creato artificialmente con l'utilizzo di uno script in Python:

```python
fake = Faker()

records = 1000000
print("Sto eseguendo %d records\n" % records)

fieldnames = ['id', 'name', 'surname', 'email', 'skill1', 'skill2', 'skill3', 'level1', 'level2', 'level3']
writer = csv.DictWriter(open("dataset.csv", "w"), fieldnames=fieldnames)

skill = ['Java', 'C', 'HTML', 'Python', 'React', 'Node', 'C++', 'javascript',
         'CSS', 'Ruby', 'C#', 'Android', 'SQL', 'PHP']

names = []
level = []
emails = []
for n in range(records):
    names.append(fake.name())
    emails.append(fake.email())
writer.writerow(dict(zip(fieldnames, fieldnames)))
for i in range(0, records):
    add1 = random.choice(skill)
    add2 = random.choice(skill)
    while add1 == add2:
        add2 = random.choice(skill)
    add3 = random.choice(skill)
    while add1 == add3 or add2 == add3:
        add3 = random.choice(skill)
    gener = random.choice(names).split()
    nome = gener[0]
    cognome = gener[1]
    email = nome[0].lower()+"." + cognome.lower()+str(i+6) + "@gmail.com"

    writer.writerow(dict([
        ('id', i+6),
        ('name', nome),
        ('surname', cognome),
        ('email', email),
        ('skill1', add1),
        ('skill2', add2),
        ('skill3', add3),
        ('level1', str(random.randint(0, 5))),
        ('level2', str(random.randint(0, 5))),
        ('level3', str(random.randint(0, 5)))]))

print("finito")
```

La scelta progettuale di generare il dataset artificialmente è dovuta alla mancanza di un dataset appropriato al corretto funzionamento di iTeam.

Lo script genera in maniera randomica: nome, cognome, e-mail e tre skill con i rispettivi livelli. Le skill vengono scelte casualmente da una lista prestabilita mentre il livello è scelto in modo randomico con un valora che va da 0 a 5.

In totale vengono generati 1.000.000 dipendenti; questo valore è stato scelto perché iTeam andrà a lavorare su una popolazione ridotta in quanto vengono processati solo i dipendenti che hanno le skill richieste.

Ex: Dimensione dataset= 1.000.000 geni;

Skill richieste = (Java, HTML, CSS);

Dipendenti effettivi = 300.000* geni.

Popolazione = 75.000 individui.

*I valori sono unicamente indicativi.

6. Soluzione

Data la composizione del problema – ovvero un problema di **ottimizzazione –** abbiamo capito che:

- La teoria dei giochi – ricerca con avversari - non è applicabile in quanto abbiamo un singolo agente;
- Non è possibile rappresentare il problema sotto forma di grafo, non essendoci un forte legame tra gli individui che formano il dataset.

L'obiettivo di iTeam è quello di restituire un team con la maggiore valutazione possibile; quindi, abbiamo optato per l'utilizzo di un **algoritmo genetico**, appropriato per i problemi di ottimizzazione.

6.1. Soluzione ottima

La soluzione ottima è rappresentata dal team che ha come valutazione **5.0**, ovvero il massimo valore assegnabile a un team.

$$\text{avg}(\sum_{d=1}^{4} dip_d)$$

Per ottenere la valutazione totale della soluzione candidata andremo a calcolare la media dei singoli dipendenti che formano il team, che si ottiene mediante la media della somma delle singole skill del dipendente.

$$\text{dip} = \text{avg}(\sum_{s=1}^{3} skill_s)$$

Ex. OPT = 5.0,

fit(Team1) = 3.70,

fit(Team2) = 4,50.

Verrà scelto il team Team2 poiché **OPT – fit(Team2) < OPT – fit(Team1)**

Se il team scelto ha valutazione 5.0, significa che avremo la seguente situazione:

$$\text{OPT - fit(team)} = 0$$

$$5.0 \text{ - avg}(\sum_{d=1}^{4} dip_d) = 0$$

Quindi il team preso in esame è **ottimo**.

7. Operatori algoritmo

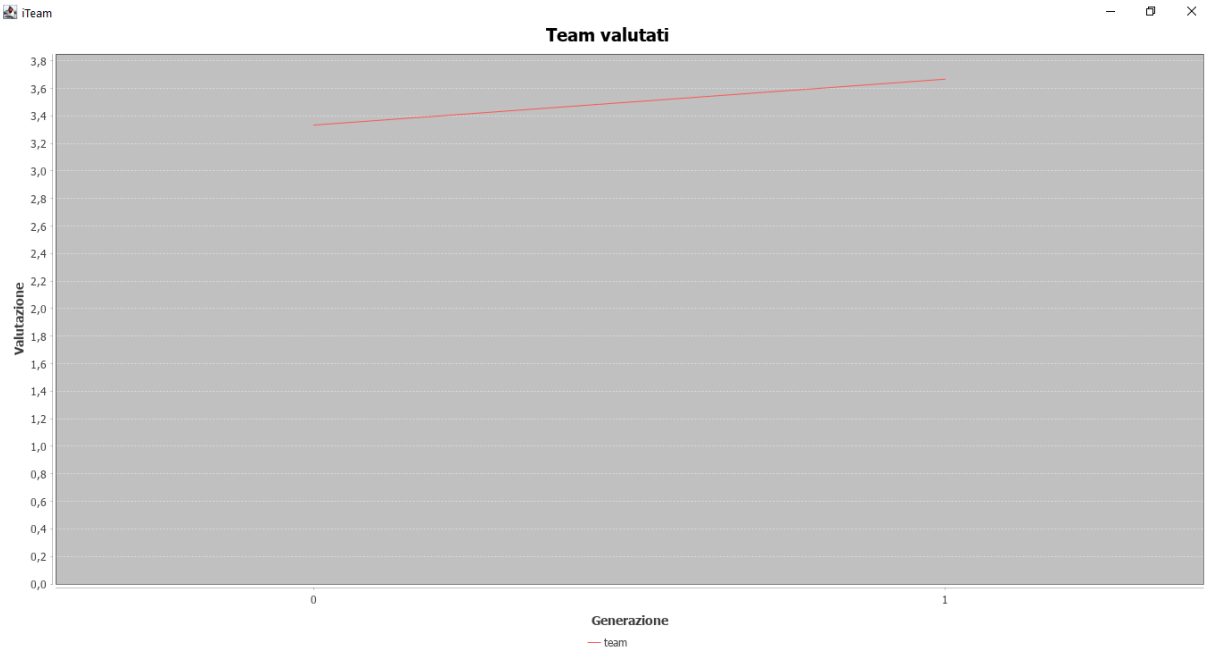
Non sono stati utilizzati framework come JMetal o Moea. Gli operatori sono stati implementati da 0.

7.1. Selection

Sono stati implementati vari tipi di selezione:

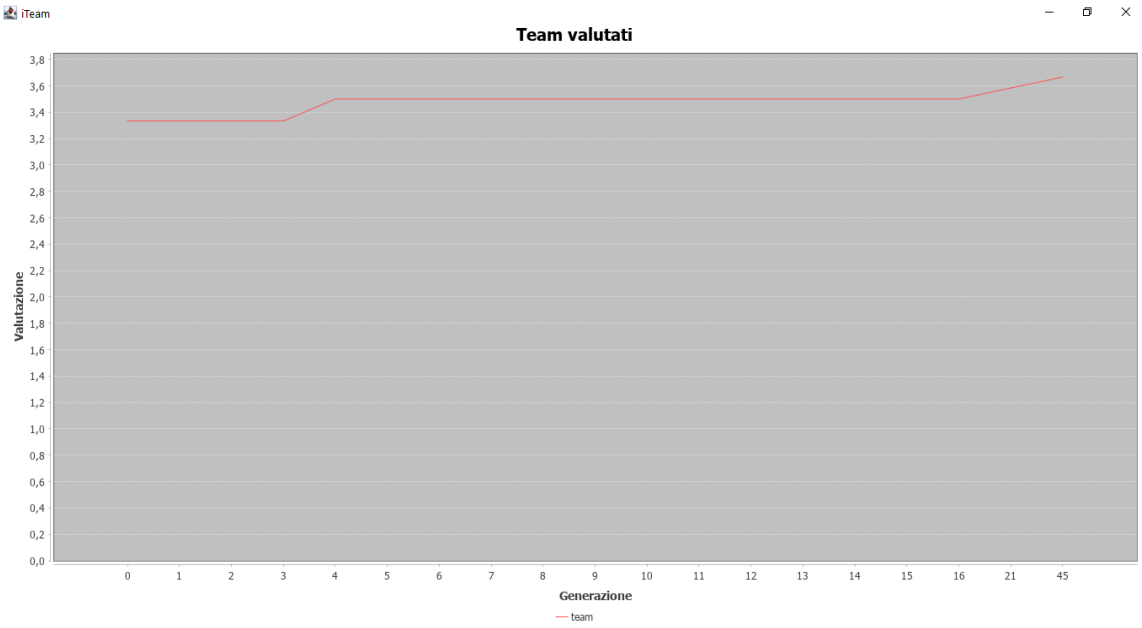
Premessa: per tutti e 3 i casi sono state eseguite 100 iterazioni.

7.1.1 Truncation



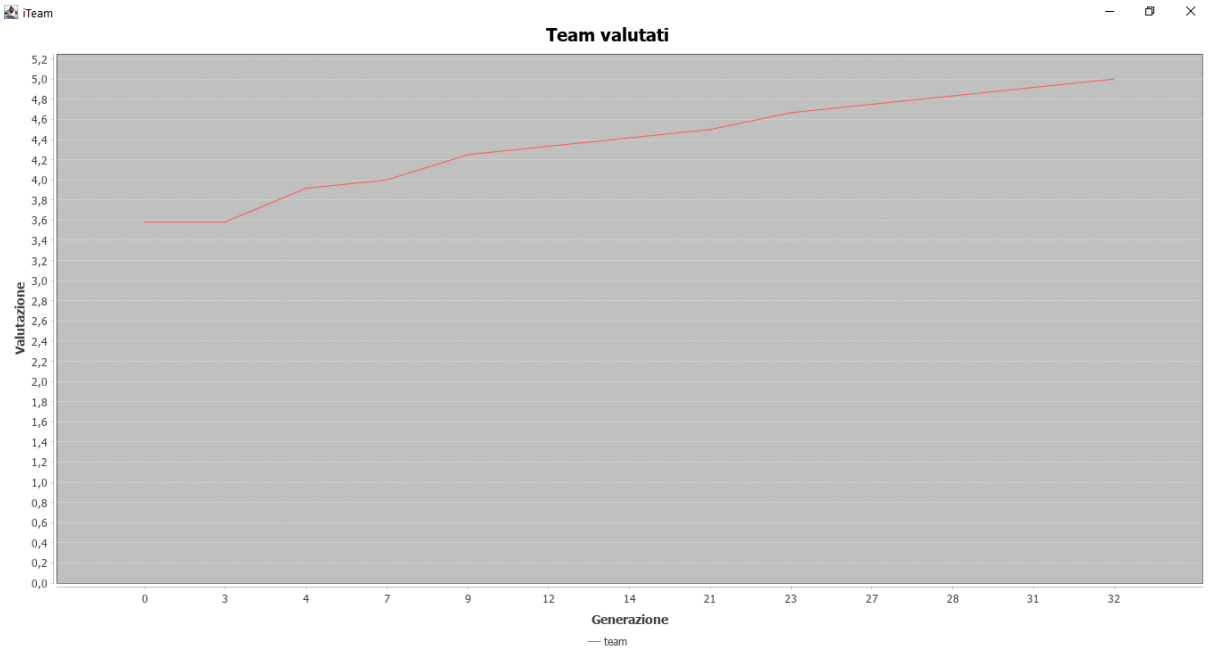
Viene scelto un valore M casuale, questo valore prende i primi M individui migliori (cioè con valore di fit più alto). Non viene utilizzato questo metodo di selezione perché porta alla generazione di pochi (in questo caso pochissimi) individui e una riduzione della popolazione. Gli individui scelti non hanno neanche il risultato migliore.

7.1.2. Roulette-Wheel



Questo metodo di selezione, anche se porta alla generazione di più individui promettenti, non è migliore del metodo che andremo ad utilizzare, in quanto non porta alla restituzione del risultato migliore.

7.1.3. Simple Sorting



Il metodo utilizzato non è proprio una selezione, ma semplicemente un ordinamento non decrescente degli individui. Questo tipo di selezione può sembrare la *Truncation,* ma la differenza sostanziale sta nella non riduzione della popolazione. Come possiamo vedere il metodo adottato ritorna risultati soddisfacenti.

7.2. Elitism

Visto che il "**metodo di selezione**" utilizzato, ordina gli individui per valore non decrescente, è stato utilizzato l'operatore di elitismo per salvare i migliori individui della generazione corrente, in modo da non perderli per le generazioni successive.

La popolazione sulla quale viene effettuato l'elitismo equivale al 60% della popolazione totale.

7.3. Crossover

L'operatore crossover è un **Single-Point Crossover** (dove il punto di crossover è generato randomicamente) che ci permette di avere una prole molto diversificata di generazione in generazione.

La probabilità di crossover è del 80%.

7.4. Mutation

L'operatore di mutazione utilizzato è il **Random Resetting** che ci permette di modificare i geni dell'individuo corrente con geni di un individuo scelto casualmente dalla popolazione.

La probabilità di mutazione è del 50%.

7.5. Evaluate

L'obiettivo di questo operatore è quello di scegliere gli individui migliori per la generazione corrente e ritorna un insieme di individui il cui valore, sottratto a quello di *avg_skill*, si avvicina di più allo 0.

8. Stopping Condition

Le condizioni di arresto di iTeam sono due:

- Il numero totale di iterazioni: 100;
- Il raggiungimento di una soluzione ottima.

Il tempo di esecuzione per 100 iterazioni è di circa 15 secondi sull'intero dataset (se la dimensione del dataset diminuisce, anche il tempo di esecuzione lo farà). È stato scelto come trade-off quello della qualità del valore di ritorno dell'algoritmo, a discapito del vincolo descritto nel design goal, "tempo di risposta", di Agency Formation.

La seconda stopping condition si basa sul raggiungimento della soluzione ottima. È stata scelta questa condizione in quanto l'algoritmo, se trova un individuo con valutazione massima, ha raggiunto il suo scopo e quindi non abbiamo bisogno di farlo continuare.

9. Semplificazioni Apportate

Per semplificare lo svolgimento dell'algoritmo sono state apportate le seguenti semplificazioni:

- Ogni dipendente preso in considerazione ha esattamente 3 skills associate;
- Per la creazione di team è stato scelto un numero di dipendenti uguale a 4 e un numero di skills richieste uguale a 3;
- Per ogni skill è stato scelto un livello di conoscenza che va da 0 a 5, scelto casualmente;