Università di Perugia
Dipartimento di Matematica e Informatica

Computational Intelligence

Traveling sales man with Particle Sworm Optimization using random key

Professore
Prof. Marco Baioletti

Studenti
Nicolò Posta

Anno Accademico 2022-2023

Indice

1 Introduzione

Il Problema del Commesso Viaggiatore (TSP) è un problema classico di ottimizzazione combinatoria che consiste nel trovare il percorso più breve che visita un insieme di città una sola volta e ritorna alla città di partenza. Risolvere il TSP è fondamentale in numerosi contesti, tra cui la logistica, la pianificazione dei percorsi e l'ottimizzazione dei trasporti.
In questa relazione, esplorerò l'applicazione dell'algoritmo Particle Swarm Optimization (PSO) con la tecnica della Random Key al TSP. L'algoritmo PSO è un algoritmo di ottimizzazione stocastico ispirato al comportamento sociale degli uccelli e dei pesci. La Random Key è una rappresentazione alternativa delle soluzioni del TSP, dove le soluzioni sono rappresentate da sequenze di numeri casuali tra 0 e 1.

2 Metodologia

2.1 Algoritmo Particle Swarm Optimization (PSO)

L'algoritmo PSO è ispirato al comportamento sociale degli uccelli e dei pesci. Nel PSO, un insieme di particelle si muove nello spazio delle soluzioni, cercando di trovare la soluzione ottimale. Ogni particella ha una posizione e una velocità, che vengono aggiornate durante le iterazioni dell'algoritmo.

Aggiornamento della Velocità della Particella:

$$\mathbf{v}_i^{t+1} = w \cdot \mathbf{v}_i^t + c_1 \cdot r_1 \cdot (\mathbf{pbest}_i - \mathbf{x}_i^t) + c_2 \cdot r_2 \cdot (\mathbf{gbest} - \mathbf{x}_i^t)$$

dove:

$\mathbf{v}_i^{t+1}$: Nuova velocità della particella i al tempo $t + 1$.

$\mathbf{x}_i^t$: Posizione corrente della particella i al tempo t.

$\mathbf{pbest}_i$: Miglior posizione individuale raggiunta dalla particella i fino a quel momento.

gbest : Miglior posizione globale tra tutte le particelle.

w : Fattore di inerzia che regola l'importanza della velocità precedente.

c_1, c_2 : Coefficienti di accelerazione cognitiva e sociale, rispettivamente.

r_1, r_2 : Numeri casuali nell'intervallo [0, 1].

Aggiornamento della Posizione della Particella:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$$

dove:

$\mathbf{x}_i^{t+1}$: Nuova posizione della particella i al tempo $t+1$.

$\mathbf{v}_i^{t+1}$: Nuova velocità della particella i al tempo $t+1$.

$\mathbf{x}_i^t$: Posizione corrente della particella i al tempo t.

Durante ogni iterazione, ogni particella aggiorna la sua velocità in base alle sue esperienze passate e alla migliore soluzione trovata dallo stormo fino a quel momento. Inoltre, la posizione di ogni particella viene aggiornata in base alla sua velocità corrente. Questa cooperazione tra particelle favorisce l'esplorazione e lo sfruttamento dello spazio delle soluzioni.

Il successo dell'algoritmo PSO dipende dalla scelta dei parametri, come il numero di particelle, il numero di iterazioni dell'algoritmo, il fattore di inerzia ed i coefficienti di accelerazione cognitiva e sociale (in questa implementazione il vicinato è tutto lo stormo quindi possiamo parlare di soluzione Globale). Trovare un insieme di parametri efficace può richiedere un'ottimizzazione specifica per il problema trattato, ma grazie allo studio di Maurice Clerc e al suo libro "Particle Swarm Optimization", ho apportato notevoli miglioramenti all'algoritmo PSO per risolvere il Problema del Commesso Viaggiatore (TSP). Attraverso un'analisi attenta, sono stati identificati valori ottimali per i parametri dell'algoritmo: $w = 0.7$, $c_1 = 1.43$, e $c_2 = 1.43$.

Inoltre, l'aggiornamento globale della migliore soluzione trovata dallo stormo consente di mantenere una traccia della soluzione ottimale complessiva, migliorando così la convergenza dell'algoritmo verso la soluzione migliore possibile.

2.2 Rappresentazione Random Key

La rappresentazione Random Key nel contesto del Problema del Commesso Viaggiatore (TSP) è una tecnica che utilizza sequenze di numeri casuali tra 0 e 1 per codificare le soluzioni del problema. Ogni sequenza di numeri casuali costituisce una

chiave univoca che sarà successivamente mappata alle città da visitare.

Il vantaggio di utilizzare questa rappresentazione nel contesto di PSO per risolvere il TSP è legato alla natura continua delle chiavi. Mentre altre rappresentazioni potrebbero richiedere la gestione di vincoli combinatori, la rappresentazione con random key offre una struttura continua e ordinata che facilita l'applicazione degli operatori di PSO, come l'aggiornamento della velocità e della posizione delle particelle.

Inoltre, l'utilizzo di una rappresentazione continua può contribuire a una maggiore esplorazione dello spazio delle soluzioni, consentendo al PSO di attraversare più agevolmente diverse regioni dello spazio di ricerca. Questo aspetto può essere particolarmente vantaggioso quando si affrontano problemi complessi come il TSP.

Nel contesto del PSO, le particelle rappresentano potenziali soluzioni del TSP e hanno una posizione associata (la lista di chiavi) e una velocità. L'aggiornamento della posizione della particella è influenzato dalla sua attuale posizione, velocità, dalla migliore soluzione globale e dalla migliore soluzione locale (la migliore trovata dalla particella stessa).

Per chiarire correttamente l'uso della random key nel TSP, facciamo un esempio più accurato:

Supponiamo di avere 5 città e perciò creeremo una random key con 5 valori compresi tra 0 e 1:

$$[0.8, 0.2, 0.5, 0.1, 0.6]$$

La random key viene mappata all'array ordinato delle città e create delle tuple nella forma (random key, città) (che a loro volta vengono ordinate, ma ora in base al valore della random key). Dopo la mappatura, otteniamo:

$$[(0.1, Città4), (0.2, Città2), (0.5, Città3), (0.6, Città5), (0.8, Città1)]$$

Data questa codifica, il venditore visiterà le città in base all'ordine crescente delle random key. Quindi, il venditore inizierà dalla Città 4 (con il valore della random key più piccola, 0.1), poi visiterà la Città 2, seguita dalla Città 3, la Città 5 e infine la Città 1 (con il valore della random key più grande, 0.8).

2.3 Implementazione

L'implementazione dell'algoritmo PSO con la Random Key è stata realizzata utilizzando il linguaggio di programmazione Python. Il codice sfrutta la libreria TSPLIB per l'importazione di problemi TSP presi dal sito TSPLIB. Per visionare il codice basta andare nella Prepository GitHub nella quele lo ho caricato:
Link alla Repository GitHub.

3 Test

Durante la fase di sperimentazione, ho condotto una serie di test su diversi problemi estratti dalla libreria TSPLIB al fine di valutare le prestazioni dell'algoritmo implementato. La TSPLIB rappresenta un insieme di istanze di problemi del commesso viaggiatore che fornisce una base standardizzata per la valutazione delle soluzioni. I test sono stati condotti considerando varie dimensioni e complessità dei problemi, al fine di ottenere una valutazione esaustiva delle capacità dell'algoritmo. I risultati di tali test sono riportati nella tabella sottostante, che evidenzia le prestazioni dell'algoritmo in termini di qualità delle soluzioni ottenute:

Iterazioni per risoluzione	**Numero risoluzioni per problema**
5000	20

Nome del Problema	**Soluzione ottima**	**Media soluzioni**	**Errore medio dall'ottimo**	**Iterazione media ultima miglioria**	**w variabile**
a280.tsp	2579	29380.7	1039.2%	2306.9	no
berlin52.tsp	7542	17570	133%	3501.2	no
att48.tsp	10628	23395.5	120.1%	3582.2	no
bays29.tsp	2020	2958.4	46.5%	3250.4	no
fri26.tsp	937	1306.9	39.5%	3904.0	no
gr24.tsp	1272	1688.8	32.8%	3340.5	no
ulysses16.tsp	6859	7348.3	7.1%	1838.5	no

Nome del Problema	**Soluzione ottima**	**Media soluzioni**	**Errore medio dall'ottimo**	**Iterazione media ultima miglioria**	**w variabile**
a280.tsp	2579	29552.0	1045.9%	2607.8	si
berlin52.tsp	7542	20264.6	168.7%	2194.2	si
att48.tsp	10628	27971.3	163.2%	1819.8	si
bays29.tsp	2020	3055.3	51.3%	940.1	si
fri26.tsp	937	1357.7	44.9%	1266.7	si
gr24.tsp	1272	1784.8	40.3%	1031.8	si
ulysses16.tsp	6859	7336.8	7.0%	1017.1	si

Dall'analisi della tabella sopra riportata emergono chiaramente alcune evidenze riguardo all'algoritmo PSO con Random Key applicato al problema del commesso viaggiatore (TSP). Risulta evidente che all'aumentare delle dimensioni del problema, le soluzioni generate diventano progressivamente subottimali. Tale tendenza si manifesta in maniera significativa, con errori che possono raggiungere addirittura il 1000%. Questo suggerisce che l'algoritmo in questione potrebbe non essere tra i più efficaci per risolvere il TSP su grandi istanze, richiedendo possibili ottimizzazioni o alternative per affrontare con successo casi di maggiore complessità. Inoltre è interessante notare come l'aggiunta della diminuzione progressiva del parametro w (fattore di inerzia) per bilanciare dinamicamente la exploitation e l'exploration del problema porti a delle performance peggiori in tutti i problemi trattati, traducendosi in una convergenza troppo rapida (si può osservare dal fatto che mediamente l'ultima miglioria avviene bel a di sotto della metà delle iterazioni utilizzate dall'algoritmo PSO).