Traveling Salesman Problem

19ª Coppa di Algoritmi

Studente: Luca Ambrosio

Data: 03/05/2019

Sommario

Problema

Il problema del commesso viaggiatore consiste nella ricerca del ciclo hamiltoniano più breve all'interno di un grafo pesato completo.
Questo tipo di problema appartiene alla classe dei problemi NP-Completi.
In Figura 1 è mostrato uno dei problemi TSP con soluzione, sottoinsieme dei problemi NP.

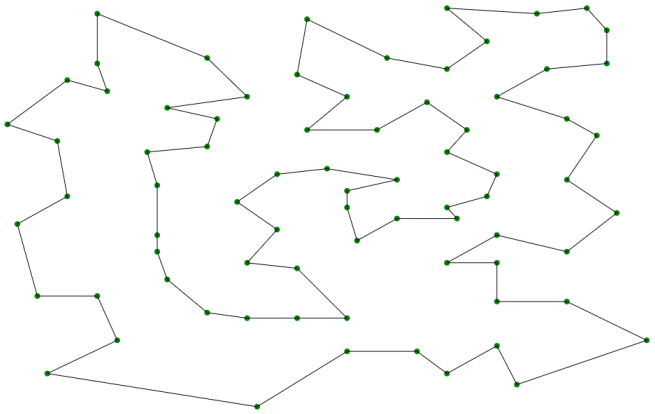


Figura 1: Eil76, problema di TSP da 76 città

Il compito assegnato è di generare una soluzione ammissibile e quanto più possibile vicina a quella ottimale (ovvero la migliore) in 3 minuti di esecuzione del programma. Il programma deve poter lavorare su 10 problemi diversi:

- *ch130;*
- *d198;*
- *eil76;*
- *fl1577;*
- *kroA100;*
- *lin318;*
- *pcb442;*
- *pr439;*
- *rat783;*
- *u1060.*

Dove il numero che segue la sigla indica il numero di città presenti.

Soluzioni implementate

Algoritmo costruttivo

L'algoritmo costruttivo è quel algoritmo il quale una volta letto il file in input ci permetterà di avere un tour delle città iniziale ed ammissibile in modo da poter poi procedere con gli algoritmi di ottimizzazione del percorso.

Per quanto riguarda questa categoria di algoritmi si è deciso di implementare una versione dell'algoritmo Nearest-Neighbor il quale è molto semplice da implementare e ci poterà a produrre delle soluzioni migliori rispetto che ad un algoritmo il quale fornirà un percorso costruito in maniera totalmente stocastica.

La prima città dalla quale si parte a costruire il percorso viene scelta in maniera randomica e ci permetterà di costruire un percorso basandoci sulla distanza minore tra la città attuale e quella successiva che si intende visitare.
Per la costruzione di un percorso ammissibile è stato dunque necessario introdurre una lista di città visitate grazie al quale sarà possibile capire se la città che si sta prendendo in considerazione è già stata visitata in precedenza.

Questo algoritmo produce una soluzione molto distante dalla soluzione ottimale del problema.

Nome file	BEST KNOW	Mio Algoritmo	%
ch130.tsp	6110	47797	87,21677093
d198.tsp	15780	22498	43204
eil76.tsp	538	1969	72,67648553
fl1577.tsp	22249	51304	56,63301107
kroA100.tsp	21282	191387	88,88012247
lin318.tsp	42029	119872	64,93843433
pcb442.tsp	50778	221440	77,06918353
pr439.tsp	107217	270646	60,38478307
rat783.tsp	8806	72134	87,79216458
u1060.tsp	224094	260174	13,86764242
		Media errori	63,931903

Percentuale errore algoritmo NN

Questa immagine infatti ci mostra che gli errori commessi da questo algoritmo sono molto alti e non possono considerarsi delle soluzioni soddisfacenti quelle prodotte da questo algoritmo. Per questo motivo si è deciso di integrare un ottimizzatore locale all'interno dell'algoritmo in modo da poter trovare soluzioni più vicine alla migliore.

Algoritmi di ottimizzazione locale

L'algoritmo implementato nella soluzione di questo tipo è l'algoritmo 2-opt il quale mi consente di trovare il minimo locale data una soluzione.

Nella versione vista in classe di questo algoritmo lo scambio avveniva trovando il best gain di tutta l'esecuzione del 2-opt ma testando l'algoritmo si è potuto constatare che se lo scambio viene fatto ogni qualvolta si trova un gain il nostro algoritmo produrrà soluzioni migliori.

Questo algoritmo viene utilizzato per trovare il minimo locale di un percorso fornito in input. L'algoritmo viene principalmente utilizzato durante l'esecuzione dell'algoritmo Simulated Annealing in quanto esso ci garantisce di spostarci nello spazio delle soluzioni ma per l'ottimizzazione di una soluzione trovata utilizzeremo appunto l'algoritmo 2-opt.

Nome file	BEST KNOW	Mio Algoritmo	%
ch130.tsp	6110	7028	13,06203756
d198.tsp	15780	17186	8,181077621
eil76.tsp	538	561	4,099821747
fl1577.tsp	22249	24816	10,34413282
kroA100.tsp	21282	24200	12,05785124
lin318.tsp	42029	49625	15,30680101
pcb442.tsp	50778	54154	6,234073199
pr439.tsp	107217	117612	8,838383838
rat783.tsp	8806	9610	8,36628512
u1060.tsp	224094	261388	14,2676787
		Media errori	10,07581429

percentuale di errore calcolata applicando alla soluzione NN e 2-opt

Algoritmi meta-euristici

Per quanto riguarda questa categoria di è deciso di implementare un algoritmo chiamato Simulated Annealing il quale ci permette di muoverci nello spazio delle soluzioni in modo da riuscire ad uscire dai minimi locali, questo sarà possibile grazie all'algoritmo double bridge, e per la ricerca dei minimi locali di una soluzione si utilizzerà l'algoritmo sopra citato.

L'algoritmo ha come parametri la temperatura di partenza la quale come visto in classe è fissata a 100 e un valore alpha fissato a 0.95 utilizzato per il lento abbassamento della temperatura che questo algoritmo richiede ed un numero di iterazioni a temperatura costante il quale nel nostro caso sarà pari a 100.

Questo algoritmo ha permesso di dare un ottimo miglioramento alle soluzioni generate e con un tempo di esecuzione di circa 150 secondi ha permesso di scende sotto l'1% di errore come media di tutte le istanze di problemi del TSP.

Esecuzione del programma

Piattaforma

La piattaforma usata è un computer portatile Asus n552vx. Nella Tabella 1 sono mostrati alcuni parametri rilevanti.

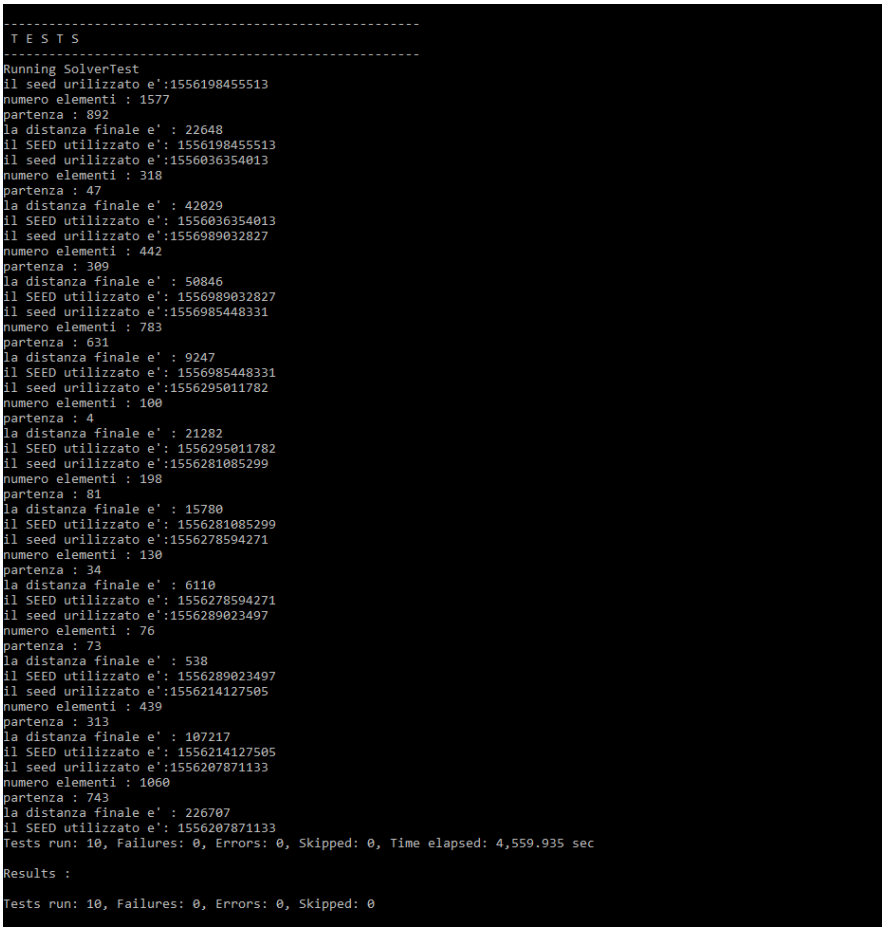
Piattaforma usata per i test

Sistema Operativo	Windows 10 Home
Processore	Intel(R) Core(TM) i7-6700HQ 2.60GHz
RAM	8 GB

Questa piattaforma è stata utilizzata per lo sviluppo e per il test di tutto il codice ed anche per la ricerca del seed ottimale per ogni istanza del problema del TSP.
Il codice infine è stato testa nella macchina di test e non vi sono stato grosse differenze tra le esecuzioni sul mio pc e quello di testing vi erano solo due file i quali hanno generato lunghezza diverse mantenendo inalterato il seed i quali sono il file “rat783.tsp” il quale è passato da essere 9124 sul mio pc ad essere 9247 sul computer fornito per i test e il miglioramento per quanto riguarda il file “pcb442.tsp” è da 50923 sul mio pc a 50846 sul pc di testing.

Compilazione ed esecuzione

Per la compilazione e l'esecuzione durante lo sviluppo di questo algoritmo si è utilizzato l'IDE IntelliJIDEA.
Per quanto riguarda la gestione delle versioni del codice sviluppato mi sono appoggiato a un repository creato su GitHub e pubblico il quale è possibile consultare al seguente link: https://github.com/lucaambrosio/TSP.
Il progetto in questione è un progetto Maven ed è stato predisposto in modo tale da rendere possibile l'esecuzione di tutte le 10 istanze di problemi da risolvere in modo automatizzato attraverso 10 test nel quale ognuno scriverà il file .tsp.tour relativo alla propria istanza.
Per fare ciò bisognerà dopo aver installato il pacchetto Maven in locale recassi nella cartella contente il file pom.xml e digitare il comando: **mvn test** e grazie a questo far partire in sequenza le 10 istanze le quali ci forniranno i tour calcolati e li andranno a scrivere su file.
Un esempio di esecuzione di questo comando è il seguente:



Esempio esecuzione di tutte le 10 istanze con test

Risultati

Nella tabella seguente sono riportate le soluzioni migliori trovate con relativo seed e percentuale di errore ad essa associati.
sono mostrati i risultati migliori per ogni problema, la relativa percentuale d'errore e il seed dal quale partire per ottenere i migliori risultati.
Nella cartella di esecuzione saranno generati a fine esecuzione i file relativi all'istanza del problema, questi file avranno .tsp.tour come estensione e ci permetteranno di essere verificati con il tool fornito tspTourChecker.py, un codice scritto in python in grado di controllare l'effettiva ammissibilità del percorso e il calcolo giusto della lunghezza del percorso.

Soluzioni migliori per ogni problema

Nome file	BEST KNOW	Mio Algoritmo	%	SEED
ch130.tsp	6110	6110	0	1556278594271
d198.tsp	15780	15780	0	1556281085299
eil76.tsp	538	538	0	1556289023497
fl1577.tsp	22249	22648	1,761744966	1556198455513
kroA100.tsp	21282	21282	0	1556295011782
lin318.tsp	42029	42029	0	1556036354013
pcb442.tsp	50778	50846	0,133737167	1556989032827
pr439.tsp	107217	107217	0	1556214127505
rat783.tsp	8806	9247	4,769114307	1556985448331
u1060.tsp	224094	226707	1,152589025	1556207871133
		Media errori	0,781718547	

Conclusioni

La percentuale finale è del 0,78% nel computer di testing fornito.
Avendo dunque ottenuto un risultato al disotto dell'1% reputo che rappresenti una buona approssimazione di un problema NP completo come quello del TSP, anche se si potrebbero integrare delle modifiche al codice in modo da abbassare l'errore medio ottenuto.

Per quanto riguarda il mio algoritmo si può notare dalla tabella degli errori che il file il quale il nostro algoritmo riesce peggio ad approssimare è il file "rat783.tsp" in quanto questo file è molto clusterizzato e rappresenta dunque una maggiore difficoltà nella sua risoluzione.

Come migliorie possibili da effettuare a questo progetto si poterebbero tenere in considerazione varie strade una possibile sarebbe quella di implementare delle candidate list le quali ci permetterebbero di avere meno iterazioni all'interno dell'algoritmo 2-opt e questo porterebbe ad ottenere risultati migliori.
Un'altra soluzione potrebbe essere quella di adattare la nostra soluzione in modo da ottenere in algoritmo di nome Lin–Kernighan il quale se implementato bene ci porterebbe ad ottenere dei risultati prossimi all'errore nullo su tutti i problemi.

In conclusione, mi ritengo molto soddisfatto del mio algoritmo e reputo che l'errore ottenuto sia accettabile per questo algoritmo.

Luca Ambrosio