Furthest Insertion: una nuova euristica per il TSP

Asaf Cohen – matr: 975599

Data termine dello stage: 21 giugno 2024

1) Ente presso cui è stato svolto il lavoro di stage: il lavoro di stage si è svolto tramite un tirocinio interno: ho contattato regolarmente il relatore professor Giovanni Righini in modo da procedere correttamente nell'implementazione del progetto e poi nella stesura dell'elaborato finale.

2) Contesto iniziale: dopo aver sostenuto l'esame di Ricerca Operativa, ho contattato il professore in quanto ero interessato ad un tirocinio interno. Tra le varie possibilità, ho scelto la traccia di tesi "Furthest Insertion" in quanto ero interessato all'argomento. Per prima cosa ho approfondito tutto quello che potevo trovare in letteratura riguardo il problema del Commesso Viaggiatore e le varie euristiche basate su inserzione.

3) Obbiettivi del lavoro:

- Implementare correttamente alcune euristiche per il TSP presenti in letteratura inclusa la nuova (Furthest Insertion) proposta nella tesi
- Confrontare le prestazioni (qualità e tempo) delle euristiche
- Determinare quanto la nuova euristica si comporti "bene" rispetto alle altre euristiche

4) Descrizione del lavoro svolto: dopo una attenta ricerca, ho deciso di implementare nel linguaggio Python le varie euristiche basate su inserzione (includendo ovviamente la nuova euristica Furthest Insertion) oltre ad altre euristiche come Nearest Neighbor e Random Insertion. Inizialmente ho scritto una classe Python che consentisse di importare le istanze TSP da TSP-LIB e che predisponesse le strutture dati fondamentali per risolvere il problema.
Successivamente ho iniziato ad implementare i metodi ausiliari della classe (distanza tra due punti, metodi di verifica della correttezza del tour…) e poi le varie implementazioni delle euristiche.
Alcune euristiche sono state implementate in varie versioni: prima di tutto ho implementato le versioni senza nessun tipo di ottimizzazione, poi le euristiche con l'utilizzo di uno heap che consente di tenere traccia delle distanze ottime / costi migliori di inserzione per ogni nodo fuori dal tour; infine dopo una preziosa indicazione da parte del relatore, ho implementato Cheapest e Furthest insertion in una terza variante che utilizza piccoli heap per ogni nodo fuori dal tour in modo da mantenere anche posizioni sub-ottime di inserzione in modo da ridurre la complessità temporale delle prime due versioni.
Inoltre sono state scritte due versioni di Random Insertion: una che sceglie casualmente il nodo da inserire nel tour (ma comunque cerca di inserirlo nel modo migliore possibile) e una seconda versione completamente casuale che sceglie casualmente il nodo da inserire nel tour e sceglie casualmente la posizione di inserzione.
Infine ho scritto anche una interessante estensione del progetto con istanze generate casualmente (ovvero con matrice delle distanze casuale ma comunque garantendo la disuguaglianza triangolare) e ottenuto delle varianti delle euristiche in cui l'inizializzazione (coppia di città iniziali del tour) avviene casualmente.
Infine ho scritto degli script Python che consentono di testare ogni singola euristica implementata su ogni istanza da TSP-LIB, sono stati ottenuti i risultati ovvero:

- Qualità delle soluzioni trovate, ovvero quanto (in percentuale) il costo totale del tour si allontana dall'ottimo
- Tempi delle euristiche (i confronti sono stati fatti a parità di istanza considerata)
- Nel caso di inizializzazione casuale, è stata analizzata la media delle qualità, la media dei tempi, la varianza delle qualità e la varianza dei tempi.

- Analizzate le prestazioni di Random Insertion e confrontati i risultati.

5) Tecnologie coinvolte: le tecnologie coinvolte e apprese sono le seguenti (in ordine cronologico):
   - Python, in particolare ho approfondito la parte OOP del linguaggio che mi ha consentito di creare una classe che includesse gli attributi (matrice di adiacenza ecc…) e i metodi utili per la manipolazione di una determinata istanza (e risoluzione della stessa), inoltre ho utilizzato in particolare i moduli:
     - heapq: che mi ha consentito di creare e gestire min e max-heap
     - matplotlib e numpy: che mi hanno consentito di generare tutti i grafici presenti nell'elaborato finale
   - Latex: durante la stesura della tesi ho impiegato ampiamente il linguaggio Latex sotto molti punti di vista (citazioni, riferimenti ecc…)

6) Competenze e risultati raggiunti:
   - Sono riuscito ad implementare correttamente alcune euristiche per il TSP presenti in letteratura
   - Sono riuscito a confrontare le prestazioni (qualità e tempo) delle euristiche
   - Sono riuscito a determinare quanto la nuova euristica si comporti "bene" rispetto alle altre euristiche
   - Ho imparato molto da questo progetto di tesi, in particolare ho imprato quanto l'ottimizzazione degli algoritmi possa comportare (a parità di risultati ottenuti) un miglioramento significativo nelle prestazioni.
   - I problemi riscontrati sono stati principalmente piccoli errori nella progettazione che sono stati risolti dopo una analisi attenta del codice.

7) Bibliografia: sono stati molti i documenti rilevanti per questa tesi, in particolare vorrei citare:
   - G. Reinelt, The Traveling Salesman: Computational Solution for TSP Applications, Heidelberg, Springer-Verlag (1994)
   - Daniel J. Rosenkrantz, Richard E. Stearns, Philip M. Lewis II, An Analysis of Several Heuristics for the Traveling Salesman Problem, SIAM Journal on Computing, 1977

In quanto hanno fornito una dettagliata analisi delle altre euristiche (Cheapest, Farthest e Nearest Insertion) che mi hanno consentito di implementare correttamente le euristiche (inclusa la nuova).