Metodi e Modelli per l’Ottimizzazione Combinatoria

Progetto d’esame, aa 2017-2018

Anna Bonaldo

# Modellazione

Il problema da affrontare può essere rappresentato in modo generico come un insieme di punti in uno spazio bidimensionale. In particolare, è modellabile come un TSP asimmetrico, in quanto è sufficiente utilizzare spigoli (e non archi orientati) per rappresentare il percorso eseguito per la foratura di una scheda. La direzione in cui viene eseguito il percorso non è influente sul tempo impiegato per percorrere l’intero tragitto.

Le istanze del problema su cui vengono testati gli algoritmi (esatti ed euristici) sono di più tipologie:

* Punti generati randomicamente in uno spazio di raggio determinato. Questo insieme di istanze del problema rappresenta un caso molto generico, e compare spesso anche in letteratura, per svariate tipologie di analisi sugli algoritmi per la risoluzione del TSP. In particolare, vengono utilizzate più misure per il raggio in cui generare casualmente i punti. In seguito una tabella che elenca tutte le possibilità esaminate.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimensione del Problema | Dataset RANDN  raggio uguale alla dimensione del problema. | CONSTRAND50  Raggio costante | CONSTRAND5000  Raggio costante | GRID  N punti disposti a griglia di lato N0.5  ­equi distanziati |
| 10 punti | Raggio 10 | Raggio 50 | Raggio 5000 | Griglia di 10 punti |
| 50 punti | Raggio 50 | Raggio 50 | Raggio 5000 | Griglia di 50 punti |
| 100 punti | Raggio 100 | Raggio 50 | Raggio 5000 | Griglia di 100 punti |
| 200 punti | Raggio 200 | Raggio 50 | Raggio 5000 | Griglia di 200 punti |

* Punti posizionati “a griglia”, tra loro uniformemente distanziati. Questo insieme di istanze del problema, invece, è probabilmente più aderente al problema in questione. E’ infatti probabile che nel caso in esame i punti in cui eseguire la foratura non siano sparsi casualmente sulla superficie dell’oggetto.

Le varie tipologie di dataset utilizzati sono state scelte per confrontare gli effetti della morfologia del problema sulle prestazioni dell’algoritmo utilizzato per la risoluzione.

# Struttura del progetto

Il progetto include:

* Una prima parte realizzata con l’utilizzo del framework **CPlex Studio**, utilizzata per la risoluzione delle istanze di TSP con metodi esatti.
* Una seconda parte, realizzata in c++, in ambiente Windows 10 (Visual Studio 2017) per la risoluzione delle istanze del TSP con metodi euristici. Il programma realizzato utilizza la libreria **OpenCV** per rappresentare graficamente (e salvare su file) le soluzioni calcolate delle istanze del problema.

# Implementazione dei metodi euristici per la soluzione del TSP

I metodi euristici implementati sono i seguenti:

* Greedy nearest-neighbour: ad ogni passo di esecuzione sceglie come nuovo passo del cammino il punto più vicino tra quelli disponibili. Viene mantenuta una lista sei nodi non ancora inclusi nel cammino. L’insieme dei nodi “liberi”, ovvero non inclusi nel cammino, è implementata attraverso la struttura dati std::set della libreria standard c++. Quest’ultima permette di verificare in tempo O(log n) se un nodo è già stato inserito nel cammino.

Poiché ad ogni passo è necessario scegliere il nodo a distanza minima tra i disponibili, questa implementazione ha un tempo di esecuzione che è al più O(n! log(n)). E’ possibile realizzare implementazioni, un po’ più complesse ma più efficienti, utilizzando strutture dati più complesse.

* 2-opt di complessità O(n2)
* Simulated Annealing
* TabuSearch

# Dataset

# Analisi dei dati