Scuola Universitaria Professionale
della Svizzera Italiana

Dipartimento
Tecnologie
Innovative

# Traveling Salesman Problem
## 19ª Coppa di Algoritmi

**Studente:** Alessandro Bianchi

**Data:** 4 maggio 2019

# Problema

Il problema è il Traveling Salesman Problem (TSP): dato un insieme di città trovare il percorso più corto che le visiti tutte senza passare due volte dalla stessa città.

Per il progetto, è stata richiesta l'implementazione di uno o più algoritmi di ricerca (vedi capitolo "Soluzioni Implementate").

# Soluzioni implementate

## Algoritmi

### Algoritmi meta-euristici

L'algoritmo metaeuristico implementato è l'**Ant Colony Optimization (ACO)**. L'algoritmo prevede un certo numero di formiche che, spostandosi tra le varie città attraverso due possibilità (*exploration* o *exploitation*), lasciano lungo la via del feromone che le formiche successive utilizzeranno per migliorare il risultato di volta in volta.

### Algoritmo costruttivo

L'algoritmo di tipo costruttivo scelto per l'implementazione è il **nearest neighbor**. Grazie a questo è possibile generare in maniera rapida una soluzione ammissibile del problema. É inoltre utilizzato per calcolare il feromone iniziale nell'algoritmo *ACO*.

### Algoritmo di ottimizzazione locale

L'algoritmo di ottimizzazione scelto è il 2-OPT. 2-OPT è utilizzato per ottimizzare il percorso ottenuto dalle formiche dell'*ACO*.

## Altre funzionalità

É possibile disegnare il percorso di ogni tour utilizzando i uno tra i metodi *getVisualTour()* e *betterDraw()* presenti nella classe Tour. Per l'esecuzione dei test, il codice relativo alla visualizzazione dei percorsi è commentato.

# Risultati

Nella Tabella 1 sono riportati i parametri utilizzati per l'esecuzione dell'ACO.
Nella Tabella 2 sono mostrati i risultati ottenuti per ogni problema.

Tabella 1: Parametri utilizzati per ogni problema

| Problema | Alpha | Beta | Exploitation | Formiche | Iterazioni | Seed 1 | Seed 2 |
|---|---|---|---|---|---|---|---|
| ch130 | 0.1 | 2 | 0.9 | 10 | 150 | 10000000 | 0 |
| d198 | 0.1 | 2 | 0.9 | 10 | 350 | 10000000 | 0 |
| eil76 | 0.1 | 2 | 0.9 | 10 | 150 | 10000000 | 0 |
| fl1577 | 0.1 | 2 | 0.9 | 3 | 44 | 10000000 | 0 |
| kroA100 | 0.1 | 2 | 0.9 | 10 | 150 | 10000000 | 0 |
| lin318 | 0.1 | 2 | 0.9 | 10 | 350 | 10000000 | 0 |
| pcb442 | 0.1 | 2 | 0.9 | 10 | 350 | 10000000 | 0 |
| pr439 | 0.1 | 2 | 0.9 | 10 | 350 | 10000000 | 0 |
| rat783 | 0.1 | 2 | 0.9 | 10 | 67 | 10000000 | 0 |
| u1060 | 0.1 | 2 | 0.9 | 10 | 23 | 10000000 | 0 |

Tabella 2: Soluzioni calcolate per ogni soluzione

| Problema | Ottimo | Risultato | Errore |
|---|---|---|---|
| ch130 | 6110 | 6110 | 0.0% |
| d198 | 15780 | 15780 | 0.0% |
| eil76 | 538 | 538 | 0.0% |
| fl1577 | 22249 | 22652 | 1,81% |
| kroA100 | 21282 | 21282 | 0.0% |
| lin318 | 42029 | 42091 | 0,15% |
| pcb442 | 50788 | 51019 | 0,47% |
| pr439 | 107217 | 107271 | 0,05% |
| rat783 | 8806 | 8955 | 1,69% |
| u1060 | 224094 | 229895 | 2723.26% |
| media | | | 0,676 % |

# Conclusioni

I risultati ottenuti, come confermato dalla media, sono piuttosto buoni: su 3 problemi gli algoritmi riescono a replicare l'ottimo e solamente due sono oltre l'1% di errore.
Lavorando ancora per trovare dei parametri migliori, è sicuramente possibile cercare di ridurre ancora l'errore.