Traveling Salesman Problem
19 Coppa di Algoritmi

Sala Nicholas

03 Maggio 2019

1 Problema

Il problema proposto durante il corso di Algoritmi nel semestre primaverile, consiste nel generare una soluzione al problema del commesso viaggiatore ammissibile e quanto più migliore in 3 minuti. Questa soluzione viene generata attraverso l'utilizzo di vari algoritmi noti che possono essere allacciati a piacimento.

2 Soluzioni implementate in uso

2.1 Algoritmo costruttivo

L'algoritmo costruttivo utilizzato è un **Nearest Neighbour**, la soluzione ottenuta da questo algoritmo viene utilizzata come soluzione di partenza dagli algoritmi meta-euristici.

2.2 Algoritmi di ottimizzazione locale

L'algoritmo di ottimizzazione locale in uso negli Algoritmi meta-euristici è un **Two-Opt**.

2.3 Algoritmi di supporto

Viene utilizzato come algoritmo di supporto per il calcolo delle Candidate List (Solo nel caso di Ant Colony) **Prim**. Il minimum spanning tree generato grazie a questo algoritmo consente di avere i collegamenti alle città che uniscono due cluster della mappa.

2.4 Algoritmi meta-euristici

Riguardo agli algoritmi meta-euristici ne sono stati implementati e sono in uso nell'algoritmo finale due: **Simulated Annealing** e **Ant Colony System**. L'algoritmo da utilizzare puo essere scelto in base al problema fornito, sfruttando

il fatto che Ant-Colony raggiunge soluzioni migliori su certi tipi di problemi rispetto a Simulated, e viceversa. Di seguito alcuni dettagli su come sono stati implementati i due algoritmi meta-euristici:

- Simulated Annealing: l'implementazione utilizza una soluzione generata dal metodo "Double bridge" ottimizzata tramite Two-Opt come soluzione candidata, ed esegue le valutazioni probabilistiche attraverso una temperatura iniziale di 100 e un valore di attenuazione alpha di 0.96.
- Ant Colony System: l'implementazione prevede l'utilizzo di candidate list, generate aggiungendo anche i nodi appartenenti al minimum spanning tree Prim, per la scelta della prossima città ad ogni passo della formica in movimento. Nel caso in cui la formica abbia già visitato tutte le città candidate, per una questione di velocità viene scelta la prima città non visitata trovata nell'array. Come algoritmo di ottimizzazione viene usato Two-Opt.

3 Esecuzione del programma

È possibile eseguire il programma passando i seguenti argomenti(In ordine):

- Nome file
- Tipo di algoritmo da utilizzare(A Ant Colony, S Simulated)
- Seed
- Tempo(millisecondi) massimo a disposizione per l'esecuzione del programma

4 Risultati

in questa sezione vengono mostrati i risultati migliori ottenuti, i file .opt.tour delle soluzioni vengono salvati nella cartella **OptTour**

Problema	**Optimum**	**Risultato**	**Errore**	**Seed**
ch130	6110	6110	0.0%	1557046674254
d198	15780	15780	0.0%	1557046815609
eil76	538	538	0.0%	1557047724198
fl1577	22249	22649	1.798%	1557048070063
kroA100	21282	21282	0.0%	1557049057362
lin318	42029	42570	1.287%	1557056244301
pcb442	50788	51512	1.445%	1557050592605
pr439	107217	107428	0.197%	1557053699610
rat783	8806	9127	3.645%	1556990197717
u1060	224094	228349	1.899%	1556991158804
media			1.0271%	

5 Conclusioni

L'algoritmo implementato è sufficientemente efficiente e consente di trovare buone soluzioni. Uno sviluppo futuro potrebbe includere l'implemetazione di un Algoritmo di Ottimizzazione in grado di sfruttare le Candidate List, in questo modo si potrebbero ottenere soluzioni ancor migliori perchè il tempo di ottimizzazione di ogni soluzione sarebbe molto minore rispetto a quello attuale.