Decision Support and Recommender Systems

Fabrizio Meniconi

Relazione progetto modulo II

ACO

La consegna del progetto consisteva nella realizzazione di un algoritmo ACO (Ant Colony Optimization) per risolvere problemi TSP (commesso viaggiatore).

ACO è un algoritmo di Swarm Intelligence, una famiglia di algoritmi che si ispirano al comportamento di gruppi di animali.

ACO tenta di emulare il comportamento naturale delle formiche al fine di trovare un percorso ottimale tra un nodo di partenza (detto colonia) e un nodo obbiettivo (cibo) basato sull'utilizzo di tecniche probabilistiche e sull'ottimizzazione metaeuristica.

Una formica è un semplice agente computazionale, che iterativamente costruisce una soluzione per il problema da risolvere. Le soluzioni parziali sono viste come stati; ogni agente si muove da uno stato i a un altro j, corrispondente a una soluzione parziale più completa. Ad ogni passo, ogni formica k calcola un insieme di espansioni ammissibili del suo stato corrente, e muove in uno di questi con probabilità $p(\mathrm{k, i, j})$.

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

Dove τ è indica quanto è stato utile nel passato fare quella specifica mossa, cioè la desiderabilità a posteriori di quella mossa e η calcolata da qualche euristica che indica la desiderabilità a priori di quella mossa. α e β regolano l'influenza di η e τ.

Alla fine di ogni iterazione, una volta che tutte le formiche hanno trovato una soluzione parziale viene aggiornata M che è la matrice dei feromoni che stabilisce l'attrattività a posteriori di ogni arco. Ogni suo elemento viene aggiornato secondo la seguente formula:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}$$

Dove ρ è il coefficiente di evaporazione che diminuisce il valore della traccia e Δτ rappresenta la somma dei contributi di tutte le formiche.

Il Codice

- Il codice è stato realizzato in linguaggio Python,
- I problemi TSP sono stati scaricati da:

  http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
- I problemi sono stati implementati nel codice mediante la libreria tsplib95
- η è stata calcolata come 1/wheight(i,j) dove wheight è la distanza tra il nodo i e il nodo j
- α e β sono state poste a 1
- Δτ è stata trovata mediante la seguente formula:

  $$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k}$$

  $$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

  Dove Q è una costante che è stata posta uguale a 10 e L(k) è la lunghezza totale del percorso dove è presente l'arco
- Il coefficiente di evaporazione ρ è stato posto uguale a 0,2

Ogni parametro è stato scelto dopo aver provato altre possibilità attraverso la fase di testing.

Risultati

Sono stati provati tre problemi TSP con 17, 21 e 24 città, attraverso le fasi di test si è cercato di rendere l'algoritmo più efficiente possibile di modo che riesca a trovare la soluzione ottima col minor numero di formiche e iterazioni possibili.

```
Il primo grafo conta 17 città, il miglior percorso è lungo 2085

Il percroso più corto trovato con 5 formiche e 10 ere è:
[ 0. 12.  3.  6.  7.  5. 16. 13. 14.  2. 10.  4.  1.  9.  8. 11. 15.]
E' lungo  2158

Il percroso più corto trovato con 25 formiche e 30 ere è:
[ 0.  3. 12.  6.  7.  5. 16. 13. 14.  2. 10.  9.  1.  4.  8. 11. 15.]
E' lungo  2085

Il percroso più corto trovato con 50 formiche e 60 ere è:
[ 0.  3. 12.  6.  7.  5. 16. 13. 14.  2. 10.  9.  1.  4.  8. 11. 15.]
E' lungo  2085

Il secondo grafo conta 21 città, il miglior percorso è lungo 2707

Il percroso più corto trovato con 5 formiche e 10 ere è:
[ 0.  5.  7.  6. 11.  3. 16. 17.  9. 19. 13. 12. 20. 14.  1.  2. 15.  4.
  8. 18. 10.]
E' lungo  4028

Il percroso più corto trovato con 25 formiche e 30 ere è:
[ 0. 11.  3. 10. 19. 18. 16.  9. 17. 12. 13. 14. 20.  1.  2.  8.  4. 15.
  5.  7.  6.]
E' lungo  2707

Il percroso più corto trovato con 50 formiche e 60 ere è:
[ 0. 11.  3. 10. 19. 18. 16.  9. 17. 12. 13. 14. 20.  1.  2.  8.  4. 15.
  5.  7.  6.]
E' lungo  2707

Il terzoo grafo conta 24 città, il miglior percorso è lungo 1272

Il percroso più corto trovato con 5 formiche e 10 ere è:
[ 0. 23.  5.  7. 20.  4.  6.  3. 11.  9. 19.  1. 14. 18. 13. 12.  8. 22.
 16. 21. 17.  2. 10. 15.]
E' lungo  1668

Il percroso più corto trovato con 25 formiche e 30 ere è:
[ 0. 11.  3. 22.  8. 12. 13. 19. 14.  1. 18. 21. 17.  2. 10.  6.  7. 20.
 16.  9.  4. 23.  5. 15.]
E' lungo  1310

Il percroso più corto trovato con 50 formiche e 60 ere è:
[ 0. 11.  3. 22.  8. 12. 13. 19.  1. 14. 18. 21. 17. 16.  9.  4. 20.  7.
 23.  5.  6.  2. 10. 15.]
E' lungo  1272
```

I risultati mostrano che con poco tempo l'algoritmo riesce a trovare la soluzione ottima di TSP con un numero non proibitivo.

La correttezza del risultato è direttamente proporzionale al numero di formiche e al numero di iterazioni.