Simulated Annealing e TSP

Simulated Annealing: implementazione e
applicazione al problema del commesso viaggiatore

Elia Innocenti

Gennaio 2024

1 Introduzione

Questa relazione descrive il lavoro svolto per l'esercizio assegnato di Intelligenza Artificiale. L'assegnazione prevede: di implementare l'algoritmo di Simulated Annealing (nella semplice versione descritta in RN 2009 §4.1.2) e di applicarlo al problema del commesso viaggiatore (Travelling Salesman Problem), ovvero della ricerca di un ciclo di costo minimo che copre tutti i vertici di un grafo assegnato. Viene fornito inoltre un esempio di applicazione classica del Simulated Annealing (V. Černý, 1985).

2 L'algoritmo

La versione dell'algorimo a cui si fa riferimento è quello riportato in RN 2009 §4.1.2.

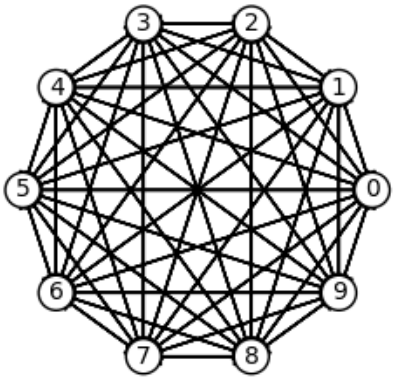
```
1  SIMULATED-ANNEALING(problema, velocità_raffreddamento)
2  corrente ←problema.STATOINIZIALE
3  for t ← 1 to inf do
4  |  T ←velocità_raffreddamento[t]
5  |  if T = 0 then
6  |  |  return corrente
7  |  end
8  |  successivo ← un successore a caso di corrente
9  |  ΔE ←VALORE(corrente) - VALORE(successivo)
10 |  if ΔE > 0 then
11 |  |  corrente ←successivo
12 |  end
13 |  else
14 |  |  corrente ←successivo con probabilità e^(ΔE/T)
15 |  end
16 end
17
```

3 Implementazione

L'algoritmo è stato implementato in python, riceve in input il problema e il cooling rate (velocità di raffreddamento). Si è voluta mantenere una struttura quanto più fedele allo pseudocodice fornito dal RN 2009, di conseguenza la permutazione delle stazioni e la generazione dei nuovi path è affidata ad un metodo del problema (implementato come classe). L'algoritmo utilizza il metodo *get_initial_state*() per ricevere lo stato iniziale, *get_random_successor*() per ricevere un successore dello stato iniziale (un random path generato a partire dallo stato corrente) e le funzioni di valutazione *calculate_value*() per valutare gli stati (calcola quindi la lunghezza dei path). Per la modellizzazione del problema è stato seguito il paper: viene utilizzata una matrice di incidenza D i cui elementi $D(i, j)$ rappresenzano il peso dell'arco che va dal nodo i al nodo j; i nodi sono invece numerati con degli interi che vanno da 0 a $N - 1$, dove N è la dimensione della matrice. Il cooling rate è implementabile in vari modi, in questo caso viene costruito un array di dimensione max_iteration con i valori della temperatura che segue un certo andamento.

4 Test e risultati

È stato effettuato un test per il funzionamento con una matrice (file matrix1.py): dimensione $10 * 10$ (le stazioni sono quindi 10) e contiene valori che vanno da 0 a 10.



	0	1	2	3	4	5	6	7	8	9
0	0	9	9	6	7	2	9	4	3	10
1	7	0	2	2	4	10	3	7	9	3
2	4	10	0	5	6	6	4	1	3	8
3	7	4	2	0	10	8	4	5	8	8
4	10	7	9	7	0	7	8	8	1	7
5	8	4	4	3	10	0	2	5	8	1
6	2	1	2	5	3	2	0	5	10	8
7	2	10	2	4	9	8	7	0	10	9
8	3	9	8	7	9	5	2	6	0	2
9	9	5	10	9	2	10	6	4	10	0

Figure 1: matrice di incidenza del problema e grafo associato.

In questo test, è stato verificato che con un tipo di cooling rate esponenziale (con $\alpha = 0.7$), una temperatura iniziale = 10000, un numero massimo di iterazioni = 100 e un numero di trials = 1000, si trova sempre il percorso di costo minimo. Il costo computazionale (il numero di iterazioni e di trials) è di molto migliorabile se si studia più approfonditamente il modello del problema specifico in questione per ottenere dei paramtri più efficaci.

5 Conclusioni

Come anche specificato nella ricerca, l'algoritmo riesce sempre a stimare il cammino ciclico di costo minimo dato un certo numero di trials, e spesso a trovarlo in maniera esatta.
Viene inoltre specificato che la correttezza dell'algoritmo non è dimostrata con teoremi rigorosi, ma il funzionamento di questo viene comunque mostrato e supportato grazie ad un'ampia serie di esempi (vedesi il paper).
C'è da dire anche che le esecuzioni dell'algoritmo per uno specifico problema sono sempre migliorabili se viene fatto uno studio approfondito sui parametri che caratterizzano la modellazione del problema stesso. Infatti se si agisce sulla velocità di raffreddamento, sulla temperatura iniziale, sul numero massimo di iterazioni, e così via, si possono ottenere risultati molto diversi. È vero anche che, se si utilizzano dei parametri che funzionano per un determinato problema, non è detto che funzionino per un problema diverso, proprio per questo è importante studiare la modellizzazione del problema ed effettuare un numero sufficiente di test.

6 Bibliografia

1. Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach (3rd ed.)

2. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, V. Černý, 1985