Esercitazione di laboratorio - Parte I

Introduzione

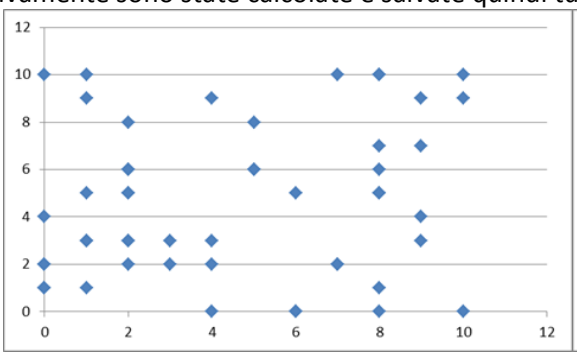
Dato un insieme di punti su una lastra da perforare, il problema consiste nel trovare il percorso per poter spostare una punta diamantata, passando per tutti questi punti una volta solamente, nel minor tempo possibile. Si deve poi tornare al punto di partenza per poter perforare la lastra successiva. Pensando alla lastra e ai punti come ad un grafo non orientato, il problema equivale a trovare il circuito hamiltoniano più breve per tale grafo.

Implementazione

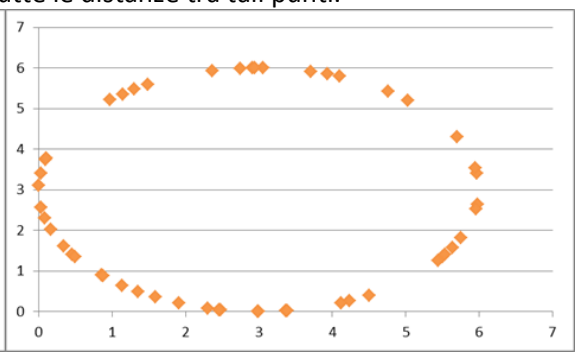
La lastra può essere vista come un piano cartesiano, dove ogni punto è rappresentato dalla coppia di valori *x* e *y* maggiori di 0.
Per una corretta esecuzione del programma è necessario conoscere alcuni parametri:

- Numero dei nodi da utilizzare
- Disposizione dei punti (0=4 punti, 1=random, 2=circonferenza, 3=clusters, 4=leggi da file)
- Numero di cluster (serve solo se il secondo parametro equivale a 3)

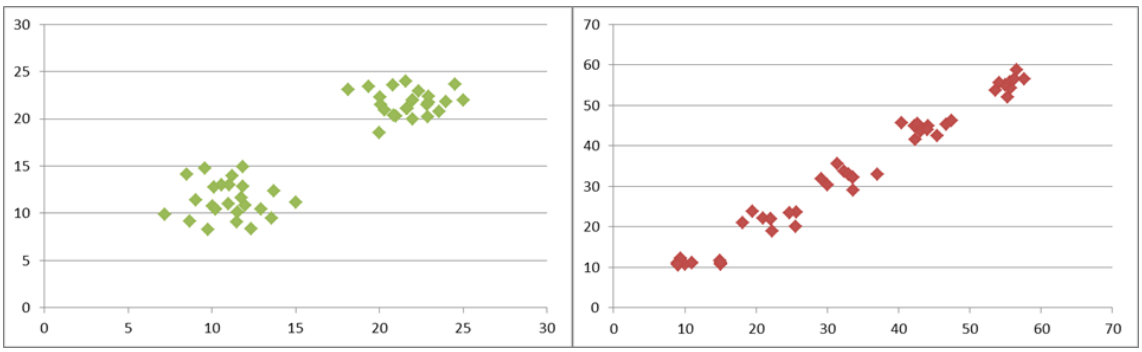
Il programma dispone perciò di diverse funzioni, chiamate a seconda del valore di un numero intero passato in ingresso al programma stesso. Tali funzioni servono per creare una certa quantità di punti da usare in una particolare disposizione. La prima crea 4 punti per poter vedere in modo semplice il risultato, la seconda crea dei punti posizionati in maniera casuale, la terza li crea lungo una circonferenza mentre la quarta li raggruppa all'interno di alcuni cerchi non sovrapposti (in questo caso c'è bisogno di un parametro aggiuntivo che indica quanti gruppi verranno usati). La quinta invece legge i punti da un file.
Successivamente sono state calcolate e salvate quindi tutte le distanze tra tali punti.



Completamente casuale



Casualmente lungo una circonferenza



Casualmente in 2 o più raggruppamenti

Per risolvere correttamente il problema, è stato usato un grafo completo non orientato. Questo perché è possibile spostare la punta da un nodo qualsiasi ad un altro nodo qualsiasi. Inoltre bisogna poter distinguere il senso di percorrenza della punta.

Sono state in seguito create le variabili *xi_j* e *yi_j* corrispondenti agli archi del grafo (le *xi_j* servono per il numero di archi da utilizzare, mentre le *yi_j* servono per vedere se un arco viene utilizzato o meno) e i vari vincoli richiesti. Sono state create anche le variabili corrispondenti ai cappi per una gestione più semplice durante le varie iterazioni, ma non sono state utilizzate nella funzione obiettivo e nei vincoli.

INSIEMI:

- N = insieme dei nodi del grafo che rappresentano le posizioni dei fori desiderati;
- A = insieme degli archi (i, j), $\forall\, i, j \in N$, che rappresentano il tragitto percorso dalla punta diamantata per spostarsi della posizione i alla posizione j.

PARAMETRI:

- c_{ij} = tempo impiegato dalla punta diamantata per spostarsi dalla posizione i alla posizione j, $\forall\, (i, j) \in A$;
- 0 = nodo di partenza del cammino, $0 \in N$.

VARIABILI DECISIONALI:

- x_{ij}= numero di unità di flusso trasportate dal nodo i al nodo j, $\forall\, (i, j) \in A$;
- y_{ij}= 1 se l'arco (i, j) viene utilizzato, 0 altrimenti, $\forall\, (i, j) \in A$.

La funzione obiettivo utilizzata è la semplice somma dei costi, utilizzando le variabili binarie relative agli archi. Queste servono sostanzialmente per rendere nulli i termini di questa somma che sono relativi agli archi che non fanno parte del ciclo hamiltoniano.
I primi due vincoli servono per gestire il numero di archi da utilizzare per costruire il ciclo. Il terzo mi dice che ogni nodo deve avere un solo arco uscente. Il quarto dice che ogni nodo deve avere un solo arco entrante. Gli ultimi servono solo per stabilire che valori possono assumere le variabili.

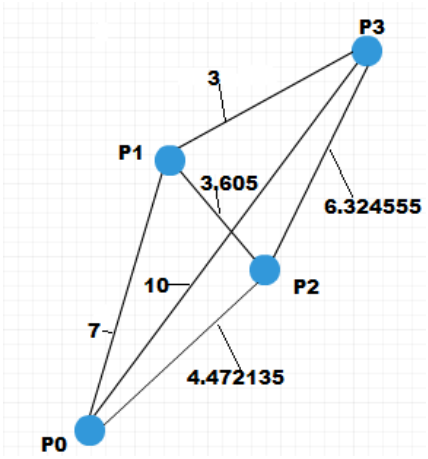
$$\min \quad \sum_{i,j:(i,j)\in A} c_{ij} y_{ij}$$

$$\begin{aligned}
s.t. \quad & \sum_{j:(0,j)\in A} x_{0j} && = |N| && \\
& \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj} && = 1 && \forall\, k \in N \setminus \{0\} \\
& \sum_{j:(i,j)\in A} y_{ij} && = 1 && \forall\, i \in N \\
& \sum_{i:(i,j)\in A} y_{ij} && = 1 && \forall\, j \in N \\
& x_{ij} && \leq |N|\, y_{ij} && \forall\, (i, j) \in A \\
& x_{ij} \in \mathbb{Z}_+ && && \forall\, (i, j) \in A \\
& y_{ij} \in \{0, 1\} && && \forall\, (i, j) \in A
\end{aligned}$$

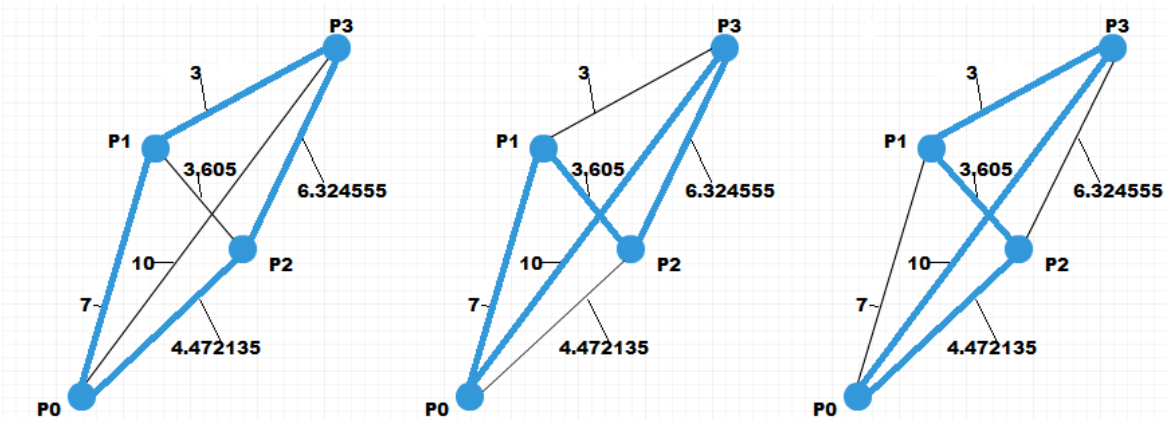
Correttezza

Per poter verificare la corretta esecuzione dell'algoritmo, si è creato un grafo con le seguenti coordinate:

	X	Y
P0	0	1
P1	1	8
P2	2	5
P3	3	11



Essendo un grafo molto semplice, si possono subito vedere quali sono i tre possibili percorsi possibili (per i motivi sopra descritti non sono stati considerati i cappi).



Sono state calcolate le distanze tra i vari nodi e quindi le lunghezze dei tre cammini, trovando rispettivamente 20.79669, 26.930106 e 21.077686. Il percorso minore è quindi il secondo. Si è perciò confrontato questo risultato con quello ottenuto con CPLEX, ovvero 20.7967 (con variabili y0_1, y1_3, y2_0 e y3_2 non nulle).

Dimensioni e tempistiche

Per vedere il tempo impiegato dall'algoritmo, si è provato a cambiare il numero dei nodi del grafo. Di seguito sono riportati i risultati del test con dei punti disposti in maniera casuale:

Numero Nodi	Tempo (sec)
3	1
4	1
10	1
20	1
30	1
40	1
50	15
60	25

70	45
80	55
90	480
100	600

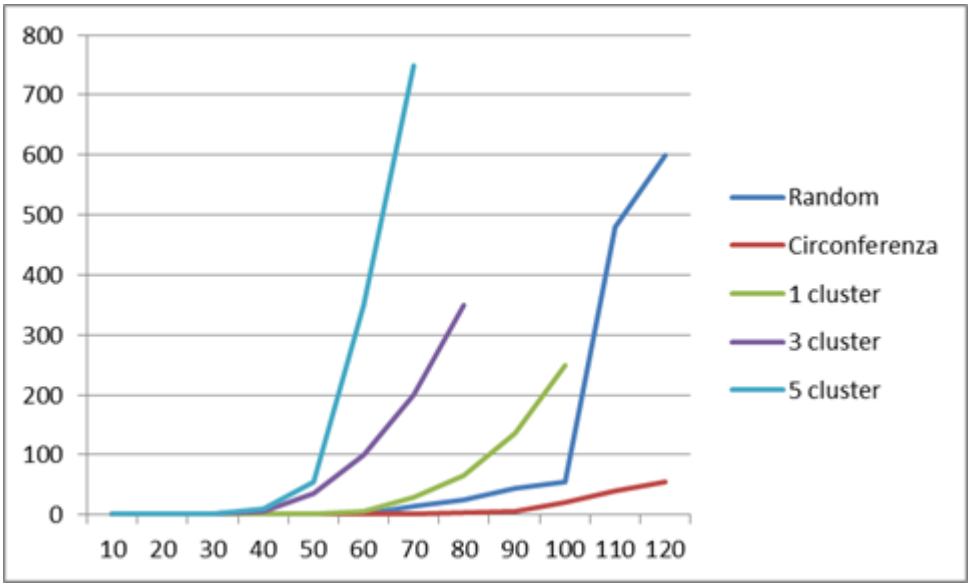
Come si può notare, l'algoritmo termina in circa un secondo se sono presenti fino a 40 nodi. Il tempo è comunque accettabile fino a 80 nodi, terminando in meno di un minuto. Aggiungendo solamente altri 10 nodi, l'algoritmo termina in 8 minuti, mentre con 100 nodi l'algoritmo è stato fermato dopo 10 minuti. Ovviamente non si è proseguito con i test visti i tempi impiegati dagli ultimi due tentativi.
Si può perciò vedere che il modello utilizzato funziona bene con un numero basso di nodi, e che comincia a perdere efficacia attorno ai 100 nodi.
Come detto in precedenza, sono state effettuate delle prove anche con i raggruppamenti.

Numero Nodi	Tempo (1 cluster)	Tempo (2 cluster)	Tempo (5 cluster)
3	1	1	1
4	1	1	1
10	2	2	2
20	2	5	10
30	2	35	55
40	7	100	350
50	30	200	750
60	65	350	-
70	136	-	-
80	250	-	-

I tempi ora sono addirittura più alti dei precedenti. Con 2 e 5 cluster sono stati fatti poche prove in quanto i tempi cominciavano a diventare molto alti e quindi il programma è stato bloccato per procedere con altre misurazioni.
Di seguito invece ci sono i risultati del test con dei punti disposti lungo una circonferenza:

Numero Nodi	Tempo (sec)
3	1
4	1
10	1
20	2
30	2
40	2
50	2
60	3
70	5
80	20
90	40
100	55

Come si può vedere in questo caso i tempi sono molto più bassi dei precedenti. Questo si può spiegare in quanto per calcolare il percorso minimo vengono presi gli archi connessi ai due "vicini", permettendo quindi di saltare molti passi.
Infine è stato creato un grafico per mostrare in maniera chiara i tempi impiegati a seconda delle varie disposizioni. L'asse X rappresenta il numero di nodi, mentre l'asse Y rappresenta il tempo espresso in secondi.


800
700
600
500
400
300
200
100
0
10 20 30 40 50 60 70 80 90 100 110 120
Random
Circonferenza
1 cluster
3 cluster
5 cluster