ELABORATO

RICERCA OPERATIVA

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Algoritmo esatto per la soluzione di problemi di clustering utilizzando una formulazione di P-Mediana.

Docente:
Prof. Boccia Maurizio

Studenti:
Coppola Vincenzo **M63/1000**
Della Torca Salvatore **M63/1011**

PROBLEMI DI LOCALIZZAZIONE

I modelli di localizzazione ed i relativi algoritmi risolutivi sono uno degli strumenti più importanti per la pianificazione territoriale dei centri di servizio.

Obiettivo → Definire le localizzazioni dei *centri di servizio* che devono soddisfare una domanda sul territorio.

LOCALIZZAZIONE PUNTUALE	***LOCALIZZAZIONE NON PUNTUALE***
Si ricercano dei punti in cui posizionare i centri di servizio.	I centri di servizio hanno un'*estensione* che impedisce di rappresentarli attraverso dei punti.

I modelli da utilizzare possono essere:

- Discreti: le localizzazioni vanno scelte in un insieme finito;
- Continui: le localizzazioni vanno scelte in un insieme continuo.

MISURA DELLE DISTANZE

Un aspetto fondamentale dei problemi di localizzazione è il concetto di *distanza* tra un centro di servizio e un punto di domanda.
Dati due punti $P_i = (x_i, y_i)$ e $P_j = (x_j, y_j)$, la distanza tra i due punti può essere espressa come:

$$d(P_i, P_j) = ((x_i - x_j)^k + (y_i - y_j)^k)^{1/k}$$

- $k = 1$ → Metrica lineare *(di Manhattan)*, efficace per descrivere spostamenti su direzioni ortogonali.
- $k = 2$ → Metrica euclidea, efficace in uno spazio che non presenta particolari ostacoli.

È stato verificato che la metrica più efficace su realtà territoriali urbane ed extraurbane è caratterizzata da valori di k compresi tra 1 e 2.

COSTI

In un problema di localizzazione è possibile distinguere due tipi di costi:

- *Costi fissi*, detti anche costi di localizzazione, che possono dipendere dal punto di localizzazione e dalla capacità del servizio (costi di apertura del centro);
- *Costi variabili*, detti anche costi di *afferenza*, che rappresentano i costi di accesso al servizio.

OBIETTIVI

Gli obiettivi di un problema di localizzazione possono essere di diverso tipo:

- Minimizzare i costi fissi;
- Minimizzare i costi di afferenza;
- Minimizzare del massimo costo di afferenza;
- Massimizzare la domanda totale coperta.

Ovviamente, è possibile considerare anche una combinazione lineare di più obiettivi.

VINCOLI

I vincoli del problema possono essere vincoli di *soddisfacimento della richiesta,* vincoli sui *costi da sostenere,* vincoli sulla *capacità dei centri di servizio,* o anche vincoli sulla *struttura della rete (distanza tra i centri, numero di centri)*.

PROBLEMA DI P-MEDIANA

Un problema di P-MEDIANA è un problema di localizzazione in cui:

- i *costi di localizzazione* sono **uguali** per ogni centro di servizio;
- il numero di centri di servizio da localizzare è fissato e pari a p.

In funzione obiettivo è possibile trascurare i costi di localizzazione essendo uguali per tutti i centri.
Il problema consiste quindi nell'individuare p punti in cui localizzare i centri di servizio con l'obiettivo di *minimizzare la somma dei costi di afferenza*.

c_{ij} → costo di afferenza in *j* della domanda del cliente *i;*
x_{ij} → 1 se il cliente *i* afferisce al centro di servizio *j*, 0 altrimenti;
y_j → 1 se nel punto *j* è localizzato un centro di servizio, 0 altrimenti;

FUNZIONE OBIETTIVO:

$$\min \sum_{\substack{i \in I \\ j \in J}} c_{ij} x_{ij}$$

- $I \subseteq V$ è l'insieme di tutti i *nodi clienti;*
- $J \subseteq V$ è l'insieme di tutti i nodi in cui può essere localizzato un centro di servizio*;*
- I e J possono anche coincidere tra loro e coincidere con tutto V.

PROBLEMA DI P-MEDIANA

VINCOLI:

$$\sum_{j \in J} y_j = p$$ → *Vincolo sul numero di centri da aprire*

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I$$ → *Vincolo sul soddisfacimento delle richieste*

$$0 \leq x_{ij} \leq y_j \qquad \forall i \in I, \forall j \in J$$ → *Vincolo di variable upper bound*

$$y_j \in \{0, 1\} \qquad \forall j \in J$$

Osservazione: *nonostante x_{ij} sia una variabile binaria non è necessario definire il vincolo di interezza ma è sufficiente che sia ≥ 0 perché, per come è definito il problema, non potrà mai assumere valore frazionario. Infatti, non essendoci vincoli di capacità dei centri, il cliente sceglierà sempre il centro aperto più vicino e afferirà completamente ad esso.*
Se dovessero esistere due centri che hanno la stessa distanza dal cliente i potrebbe capitare che il valore della x_{ij} sia frazionario, ma se ciò dovesse accadere esisterebbe sicuramente una soluzione migliore (ottima) in cui il valore di x_{ij} è intero.

PROBLEMI DI CLUSTERING

Un problema di clustering nasce quando si ha a disposizione un insieme di dati che devono essere raggruppati in più sottoinsiemi, detti *cluster,* in maniera tale che i dati che presentino una maggiore omogeneità o somiglianza siano contenuti nello stesso cluster.

Dato un insieme di oggetti di cardinalità elevata, ogni oggetto può essere:

- classificato sulla base del valore assunto da n grandezze;
- rappresentato da un punto in uno spazio n-dimensionale.

Dire che gli oggetti sono simili vuol dire che sono caratterizzati da valori vicini delle grandezze.

Per misurare la *dissimilarità* tra due oggetti è necessario calcolare la distanza euclidea:

$$d(u,v) = \sqrt{\sum_{k=1}^{n} (u_k - v_k)^2}$$

Maggiore è la distanza euclidea, più gli oggetti sono diversi tra loro.

PROBLEMA DI CLUSTERING

Il problema può essere rappresentato mediante un grafo $G(V, A)$, dove V è l'insieme dei punti dello spazio euclideo rappresentativi dei singoli oggetti, e A è l'insieme di tutti gli archi (u, v).
Ad ogni arco è associato un *costo* pari a $d(u, v)$.

L'obiettivo è trovare la partizione di V in k cluster, con k prefissato, che minimizza la somma dei pesi degli archi incidenti nei nodi appartenenti ad uno stesso cluster.

Centroide ⟹ punto dello spazio euclideo posto al centro del cluster e rappresentativo di tutti gli elementi del cluster

Il centroide può essere *reale,* cioè coincide con uno degli oggetti del cluster, o *immaginario*.
Se si desidera un centroide reale, il problema di clustering può essere modellato come un problema di *P-Mediana* in cui:

- ciascuna mediana corrisponde ad un centroide;
- il cluster rappresentato da un centroide contiene tutti e soli i nodi del grafo che afferiscono al centroide.

PROBLEMA DI CLUSTERING

d_{ij} → distanza del nodo *i* dal j;
x_{ij} → 1 se il nodo *i* afferisce al centroide *j*, 0 altrimenti;
y_j → 1 se nel punto *j* è localizzato un centroide, 0 altrimenti;

FUNZIONE OBIETTIVO:

$$\min \sum_{i,j \in V} d_{ij} x_{ij}$$

VINCOLI:

$$\sum_{j \in V} y_j = p$$ → *Vincolo sul numero di centroidi/cluster*

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in I$$ → *Vincolo sul soddisfacimento delle richieste*

$$0 \leq x_{ij} \leq y_j \qquad \forall i \in I, \forall j \in J$$ → *Vincolo di variable upper bound*

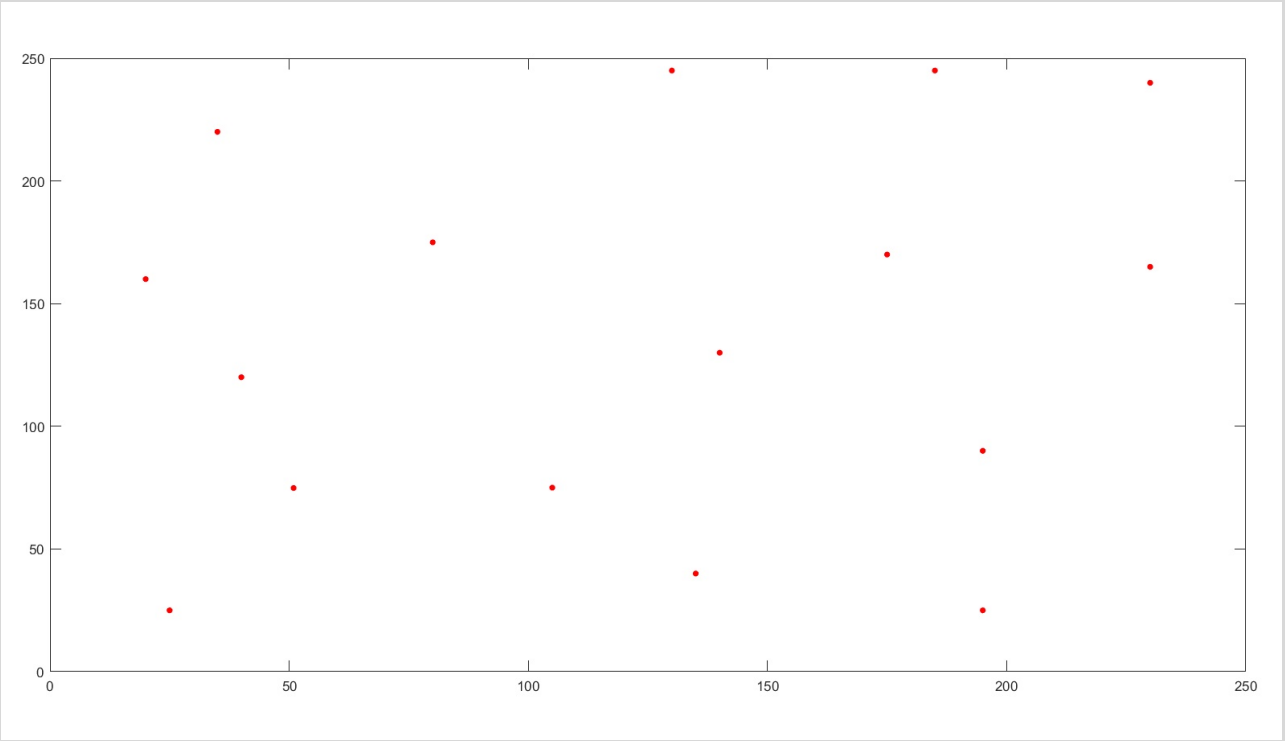
$$y_j \in \{0,1\} \qquad \forall j \in J$$

Osservazione: per $p = 1$ il valore di funzione obiettivo è il più alto possibile. Più cresce il valore di p più il valore di funzione obiettivo decresce. Il valore ottimale di p è quello a partire dal quale la funzione obiettivo inizia a decrescere più lentamente.

DATI DEL PROBLEMA

Insieme dei punti che si vogliono clusterizzare.

```
!Nodi
NODIINPUT:  [
(1)     [25     25]
(2)     [105    75]
(3)     [40     120]
(4)     [140    130]
(5)     [195    25]
(6)     [230    240]
(7)     [195    90]
(8)     [230    165]
(9)     [185    245]
(10)    [135    40]
(11)    [80     175]
(12)    [35     220]
(13)    [175    170]
(14)    [20     160]
(15)    [130    245]
]
```



ALGORITMO P-MEDIANA

L'algoritmo risolutivo del problema è stato sviluppato in MOSEL.

DICHIARAZIONE E INIZIALIZZAZIONE

```
model Clustering
uses "mmxprs", "mmsvg"

parameters
DATAFILE = 'Clustering.dat'                    !File da cui leggere i dati di input
end-parameters
forward procedure Calcola_distanze             !Procedura che calcola le distanze tra tutti i nodi
forward procedure Disegna_soluzione            !Procedura usata per rappresentare la soluzione

declarations
    p:      integer                            !Numero di centroidi
    NODI:   set of integer                     !Nodi della rete
    XCORD:  array(NODI) of real                !Coordinate x dei nodi
    YCORD:  array(NODI) of real                !Coordinate y dei nodi
    DIST:   dynamic array(NODI,NODI) of real   !Distanze tra i nodi
    x:      dynamic array(NODI,NODI) of mpvar  !Variabili per gli archi
    y:      dynamic array(NODI) of mpvar       !Variabili per i centroidi
end-declarations

initializations from DATAFILE
    [XCORD, YCORD] as "NODIINPUT"
end-initializations
```

CREAZIONE DELLE VARIABILI

```
!Creazione delle variabili
forall(i,j in NODI) do
    create(x(i,j))
end-do

forall(j in NODI) do
    create(y(j))
    y(j) is_binary                    !Vincolo y={0,1}
end-do
```

CALCOLO DELLE DISTANZE TRA I PUNTI

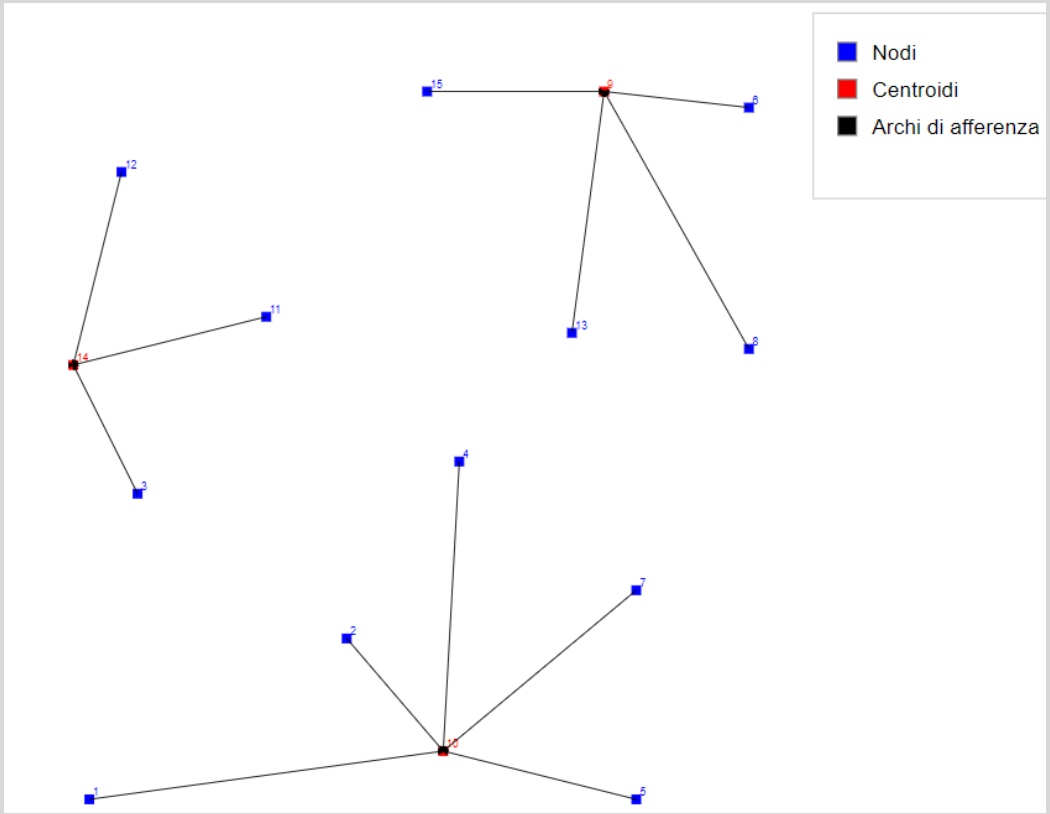
```
procedure Calcola_distanze

    writeln
    writeln("Calcolo delle distanze tra i nodi...")
    forall(i, j in NODI | i<>j) do
        DIST(i,j) := sqrt((XCORD(i)- XCORD(j))^2 + (YCORD(i)-YCORD(j))^2)
        writeln("La distanza tra i nodi ", i, " e ", j, " e' ", DIST(i,j))
    end-do
    forall(i in NODI)
        DIST(i,i) := 0
    writeln("Distanze tra i nodi calcolate")

end-procedure
```

FORMULAZIONE DEL PROBLEMA

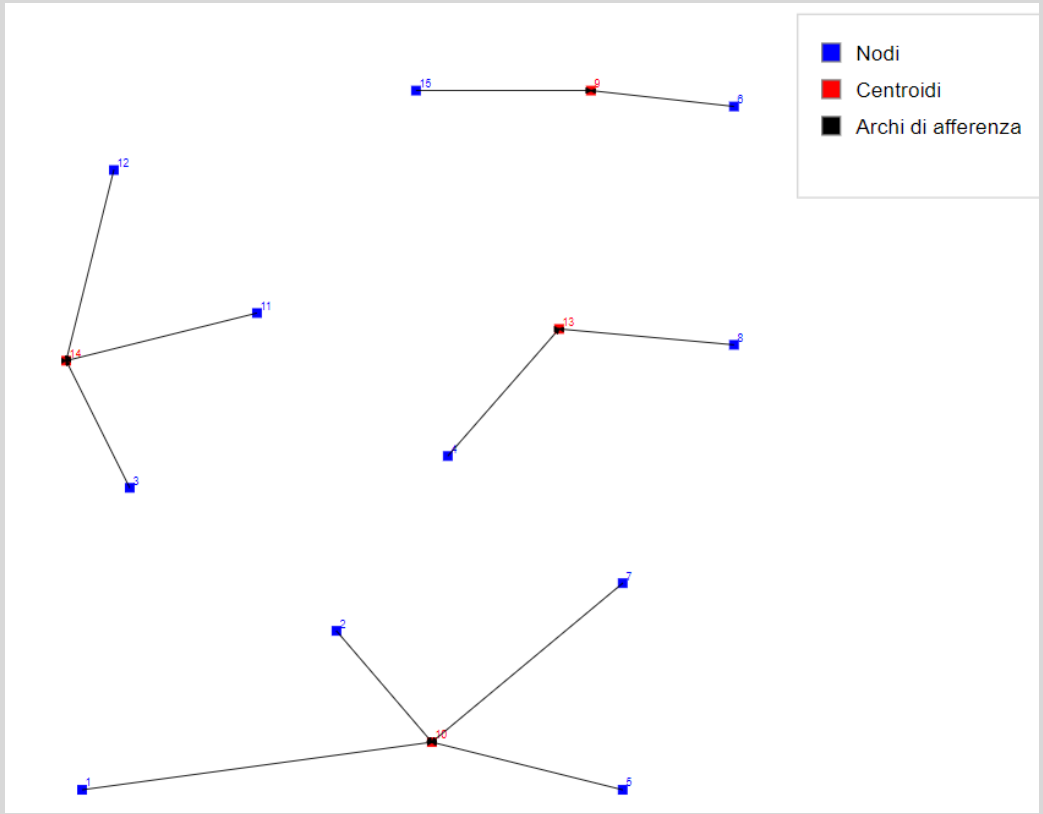
```
!Vincoli
sum(j in NODI) y(j) = p                    !numero di centroidi uguale a p

forall(i in NODI)
    sum(j in NODI) x(i,j) = 1              !ogni nodo deve afferire ad un solo centroide

forall(i,j in NODI) do
    x(i,j) <= y(j)                         !Vincolo di variable upper bound
    x(i,j) >= 0
end-do

!Funzione obbiettivo
Obiettivo := sum(i,j in NODI | i<>j and exists(DIST(i,j))) DIST(i,j)*x(i,j)

minimize(Obiettivo)     !Minimizzazione
```

ESECUZIONE ALGORITMO

$p = 3$

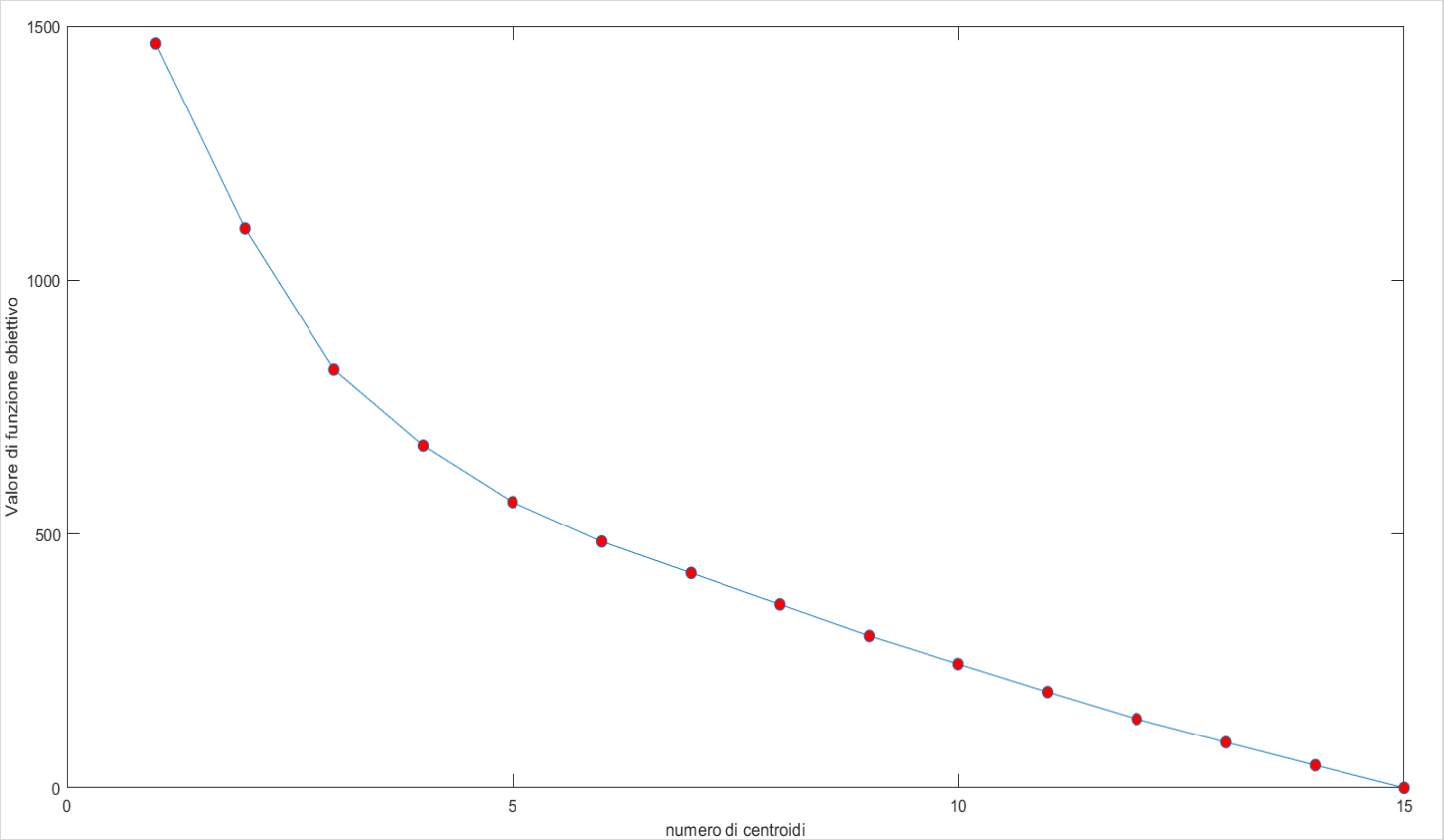


$p = 4$



ESECUZIONE ALGORITMO

L'algoritmo è stato eseguito più volte con valori diversi di p per osservare l'andamento della funzione obiettivo.



Osservazione: *dal grafico della funzione obiettivo in funzione del numero di cluster si può osservare che a partire dal valore $p = 3$ la funzione obiettivo inizia a decrescere più lentamente.*

ALGORITMO K-MEANS

Quest'algoritmo individua per ogni cluster un centroide che si trova esattamente al centro e che quindi, con molta probabilità, sarà un centroide immaginario.
Questo centroide è anche detto *baricentro*.

1. L'algoritmo seleziona k centroidi iniziali;

2. Ciascun nodo viene assegnato al cluster con il centroide più vicino; per ogni elemento x il centroide c a cui viene assegnato è dato da $c = argmin_{c_i \in C}\, dist(c_i, x)$;

3. Si aggiorna la posizione dei centroidi per ogni cluster; la posizione del nuovo centroide sarà la media delle posizioni di tutti gli elementi che appartengono al cluster.

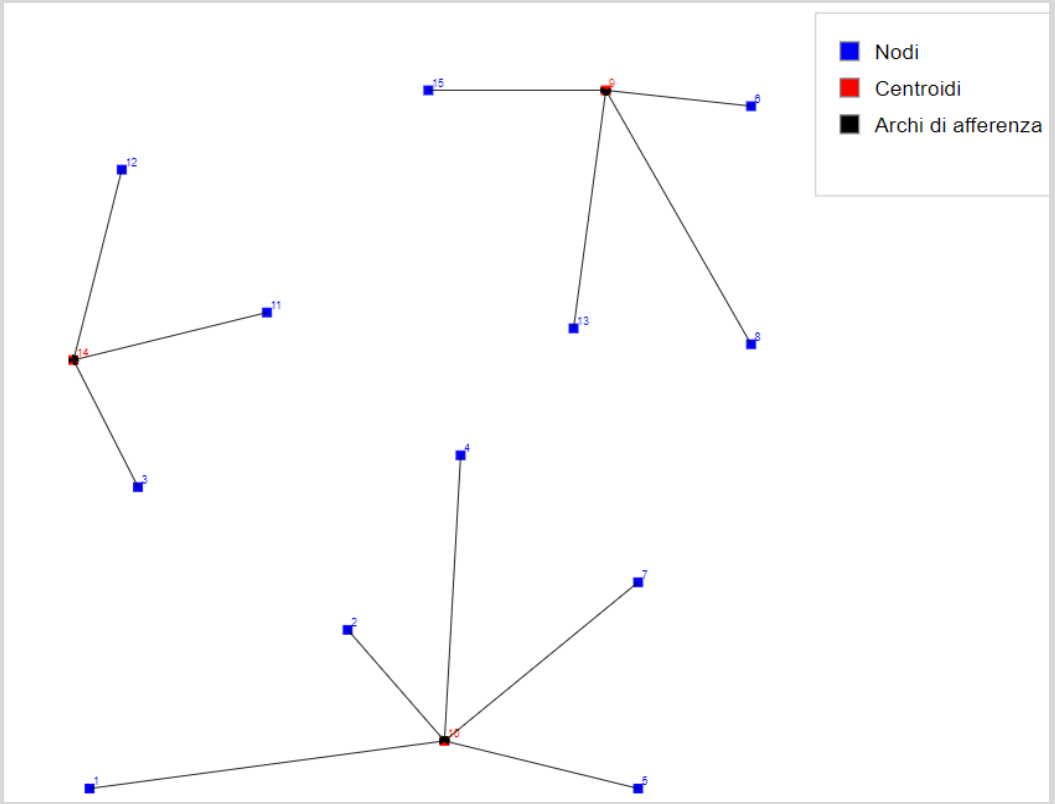
$$c_i = \frac{1}{|k_i|} \sum_{x \in k_i} x_i$$

Il procedimento viene iterato fino ad arrivare a *convergenza*, cioè da un'iterazione a quella successiva non cambia la posizione di nessun centroide.

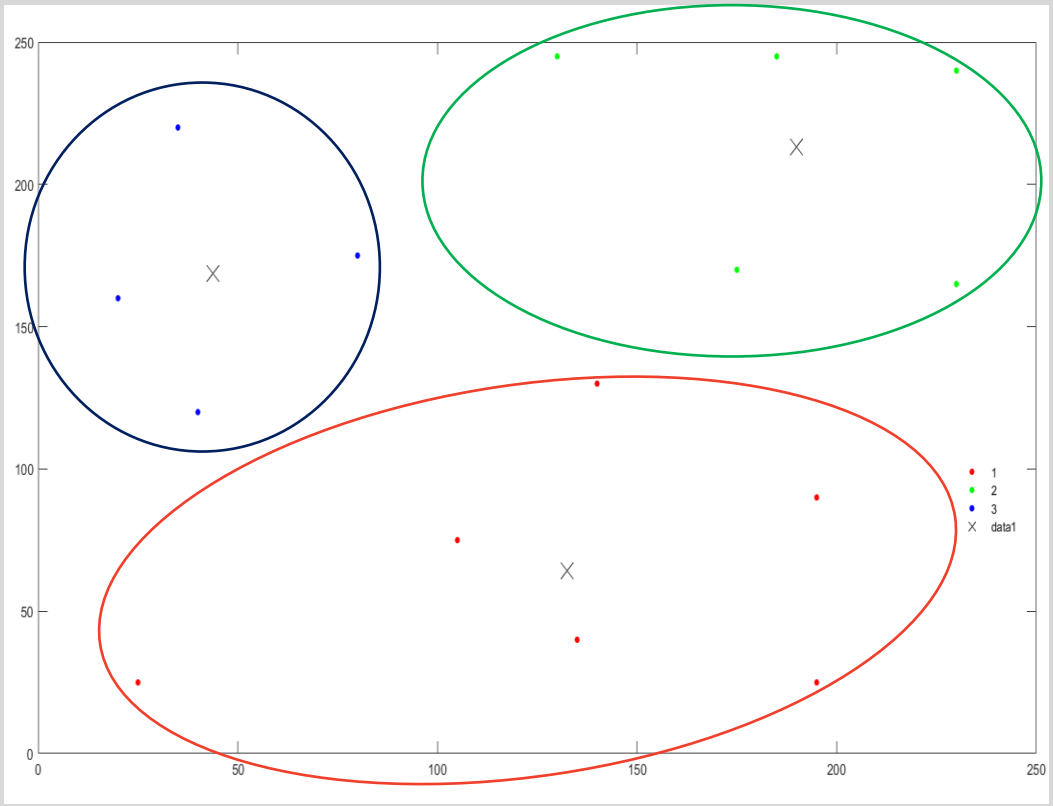
Per realizzare il confronto tra p-mediana e k-means è stata utilizzata la funzione kmeans di Matlab.

CONFRONTO: $p = 3$

P-MEDIANA



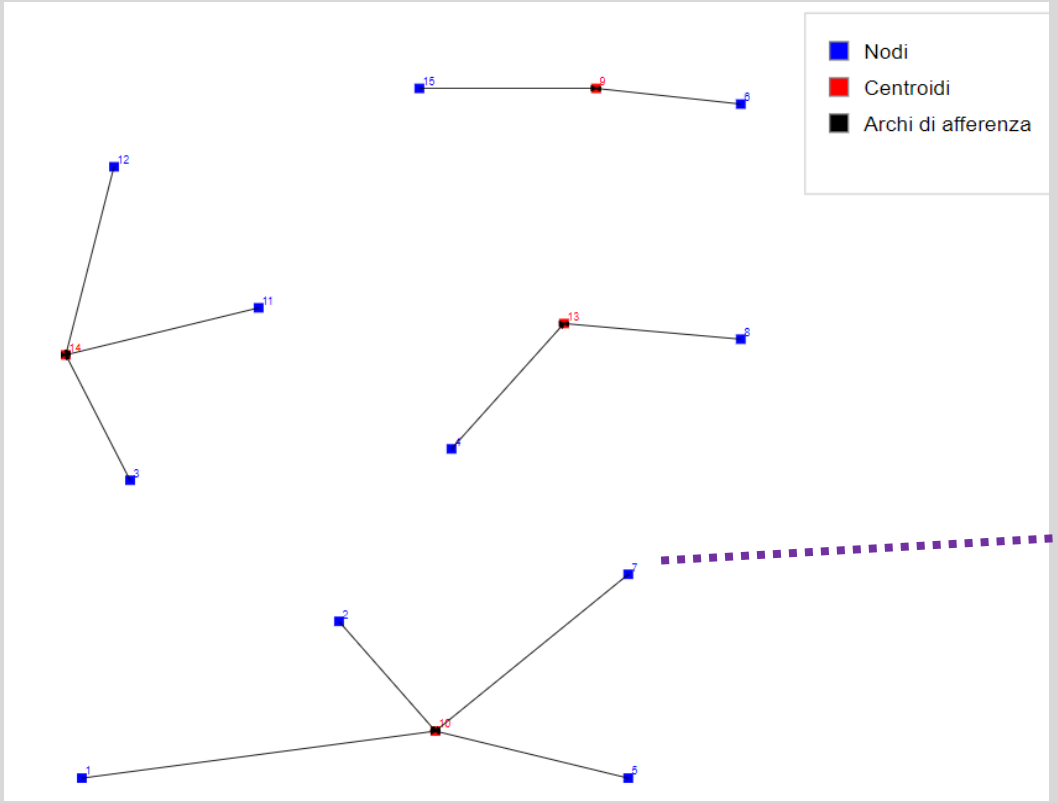
K-MEANS



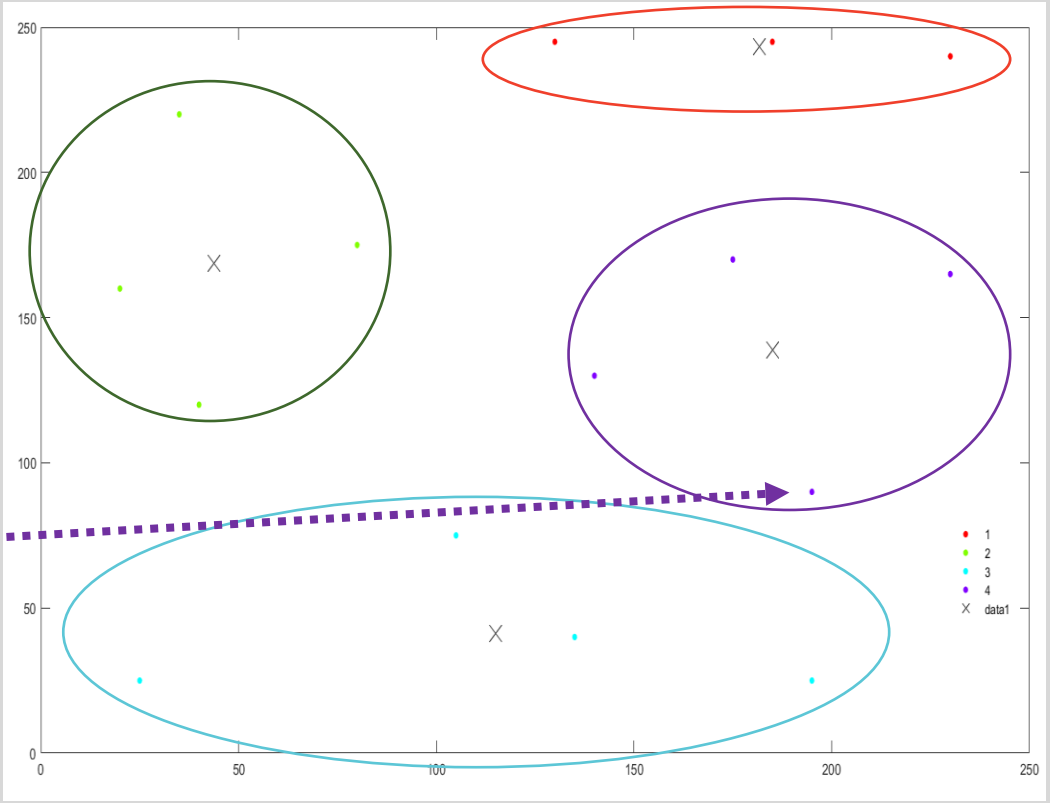
Osservazione: *in questo caso i due risultati sono uguali, ma cambiano i centroidi che nel primo caso sono reali e nel secondo caso sono immaginari.*

CONFRONTO: $p = 4$

P-MEDIANA



K-MEANS



Osservazione: *in questo caso i due risultati differiscono. Questo perché il k-means calcola come centroide il punto medio degli elementi appartenenti al cluster e assegna ogni elemento al cluster col centroide a distanza minima.*