# Clustering

Generado por Doxygen 1.8.4

Marzo de 2014

# Índice

# 1. Indice de namespaces

## 1.1. Lista de 'namespaces'

Lista de los 'namespaces', con una breve descripción:



# 2. Indice jerárquico

## 2.1. Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:



# 3. Índice de clases

### 3.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:



## 4. Indice de archivos

### 4.1. Lista de archivos

Lista de todos los archivos con descripciones breves:



## 5. Documentación de namespaces

### 5.1. Referencia del Namespace Ui

## 6. Documentación de las clases

### 6.1. Referencia de la Clase Algoritmo

```
#include <algoritmo.h>
```

### Métodos públicos

- Algoritmo ()

  *Algoritmo::Algoritmo() Contructor por defecto. Inicializa un objeto Algoritmo.*
- Algoritmo (igraph_t &grafo, int nodos, int aristas, int individuos, int generaciones, double porcentajeMatingPool, bool mutarDosBits)

  *Algoritmo::Algoritmo Constructor. Inicializa un objeto Algoritmo.*
- ~Algoritmo ()
- void algoritmo ()

  *Algoritmo::algoritmo Se realiza primero el clustering haciendo uso del algoritmo de Newman. Paso siguiente se inicializa la población de cromosomas garantizando que tengan solapamiento entre clusters. Luego de tener la población inicial, se realiza la selección de un porcentaje de cromosomas, usando la modularidad como función de aptitud. Una vez se carga el mating pool con los cromosomas seleccionados se realiza la reproducción, usando como operador la mutación.*
  *Finalmente se selecciona como solución el cromosoma que obtenga la mayor modularidad.*
- double getModularidad ()

  *Algoritmo::getModularidad Retorna la modularidad final obtenida con el mejor cromosoma encontrado.*
- vector< unordered_set< int > > getConfiguracionFinal ()

  *Algoritmo::getConfiguracionFinal Retorna la configuración final del mejor cromosoma encontrado.*
- int getCantCambios ()

  *Algoritmo::getCantCambios Retorna cuantos cambios se hicieron a la referencia del mejor cromosoma al final del algoritmo (Para elegir la solución).*

### Métodos privados

- void inicializarPoblacion ()

  *Algoritmo::inicializarPoblacion Se crean los individuos iniciales de cromosomas, se usa la configuración de clustering retornada por Newman, y se garantiza que todos los cromosomas iniciales tengan solapamiento en un nodo.*
- void seleccionar ()

  *Algoritmo::seleccionar La selección se realiza con la técnica torneo, se seleccionan dos cromosomas al azar, de cada uno se obtiene la modularidad, se elige el cromosoma con mayor modularidad. Si la modularidad llegara a ser igual en ambos cromosomas, se elige el cromosoma que tenga mayor porcentaje de clusters respecto a la cantidad de clusters encontrada al inicio con el algoritmo de Newman.*
- void reproducir ()

  *Algoritmo::reproducir Para la reproducción, se usa la mutación sobre un cromosoma, esta puede darse de dos formas, según el se halla definido. Una es mutar un solo bit de la matriz X, y la otra mutar dos bits de la matriz X. Si el cromosoma es el indicado por el indice del mejor cromosoma en el mating pool, este no se muta y se pasa directo a la siguiente generación.*
- void clusteringNewman ()

  *Algoritmo::clusteringNewman Realiza el primer clustering haciendo uso del algoritmo propuesto por Newman en **Finding community structure in very large networks** puede ver detalles en* `http://igraph.sourceforge.net/doc/html/ch22s06.html` *y* `http://www.arxiv.org/abs/cond-mat/0408187` .
- void inicializarMatrizX ()

  *Algoritmo::inicializarMatrizX Inicializa la matriz de membresia (binaria), tal y como el algoritmo de Newma configuró los clusters. La matriz tiene la forma descrita en el trabajo* `A Extraction Method of Overlapping Cluster Based on Network Structure Analysis` *Las filas corresponden a los clusters y las columnas a los nodos.*
- void inicializarConfiguracionInicial ()

  *Algoritmo::inicializarConfiguracionInicial Inicializa la configuración incial de clustering en el formato de lista de adyacencia.*
- void construirListaAdyacencia ()

  *Algoritmo::construirListaAdyacencia Crea la lista de adyacencia del grafo, la cual corresponde a una estructura de dato mucho más eficiente que la matriz de adyacencia. Se usa como estructura de dato vector<unordered_set<int>>.*

**Atributos privados**

- int nodos
- int aristas
- int individuos
- int generaciones
- Cromosoma ** poblacion
- Cromosoma ** matingPool
- int mejorIndicePoblacion
- double mejorModularidad
- Cromosoma * mejorCromosoma
- int tamMatingPool
- double porcentajeMatingPool
- bool mutarDosBits
- igraph_t grafo
- double modularidadInicial
- double modularidadFinal
- int cantClustersInicial
- int cantCambios
- igraph_vector_t tamanioClustersInicial
- igraph_vector_t membresiaInicial
- vector< vector< bool > > matrizXInicial
- vector< unordered_set< int > > configuracionInicial
- vector< unordered_set< int > > listaAdyacencia
- vector< unordered_set< int > > configuracionFinal

#### 6.1.1. Descripción detallada

Definición en la línea 19 del archivo algoritmo.h.

#### 6.1.2. Documentación del constructor y destructor

##### 6.1.2.1. Algoritmo::Algoritmo ( )

Algoritmo::Algoritmo() Contructor por defecto. Inicializa un objeto Algoritmo.

Definición en la línea 20 del archivo algoritmo.cpp.

##### 6.1.2.2. Algoritmo::Algoritmo ( igraph_t & *grafo*, int *nodos*, int *aristas*, int *individuos*, int *generaciones*, double *porcentajeMatingPool*, bool *mutarDosBits* )

Algoritmo::Algoritmo Constructor. Inicializa un objeto Algoritmo.

**Parámetros**

| | |
|---|---|
| *grafo* | igraph_t& grafo al cual se le realizará el agrupamiento. |
| *nodos* | int con el número total de nodos del grafo. |
| *aristas* | int con el número total de aristas del grafo. |
| *individuos* | int cantidad de individuos para la población inicial. |
| *generaciones* | int número de generaciones para iterar el algoritmo. |
| *porcentaje-MatingPool* | double con el porcentaje de la población inicial que se llevará al *mating pool* |

| | |
|---|---|
| *mutarDosBits* | bool activa mutar sobre dos bits de la matriz de membresia, sino se muta sobre un sólo bit. |

Definición en la línea 36 del archivo algoritmo.cpp.

#### 6.1.2.3. Algoritmo::∼Algoritmo ( )

Definición en la línea 64 del archivo algoritmo.cpp.

### 6.1.3. Documentación de las funciones miembro

#### 6.1.3.1. void Algoritmo::algoritmo ( )

Algoritmo::algoritmo Se realiza primero el *clustering* haciendo uso del algoritmo de Newman. Paso siguiente se inicializa la población de cromosomas garantizando que tengan solapamiento entre *clusters*. Luego de tener la población inicial, se realiza la selección de un porcentaje de cromosomas, usando la modularidad como función de aptitud. Una vez se carga el *mating pool* con los cromosomas seleccionados se realiza la reproducción, usando como operador la mutación.

Finalmente se selecciona como solución el cromosoma que obtenga la mayor modularidad.

Definición en la línea 102 del archivo algoritmo.cpp.

#### 6.1.3.2. void Algoritmo::clusteringNewman ( ) [private]

Algoritmo::clusteringNewman Realiza el primer *clustering* haciendo uso del algoritmo propuesto por Newman en **Finding community structure in very large networks** puede ver detalles en http://igraph.-sourceforge.net/doc/html/ch22s06.html y http://www.arxiv.org/abs/cond-mat/0408187 .

Definición en la línea 165 del archivo algoritmo.cpp.

#### 6.1.3.3. void Algoritmo::construirListaAdyacencia ( ) [private]

Algoritmo::construirListaAdyacencia Crea la lista de adyacencia del grafo, la cual corresponde a una estructura de dato mucho más eficiente que la matriz de adyacencia. Se usa como estructura de dato vector<unordered_-set<int>>.

Definición en la línea 534 del archivo algoritmo.cpp.

#### 6.1.3.4. int Algoritmo::getCantCambios ( )

Algoritmo::getCantCambios Retorna cuantos cambios se hicieron a la referencia del mejor cromosoma al final del algoritmo (Para elegir la solución).

**Devuelve**

int cantCambios

Definición en la línea 218 del archivo algoritmo.cpp.

#### 6.1.3.5. vector< unordered_set< int > > Algoritmo::getConfiguracionFinal ( )

Algoritmo::getConfiguracionFinal Retorna la configuración final del mejor cromosoma encontrado.

**Devuelve**

vector<unordered_set<int> > con la configuración final.

Definición en la línea 208 del archivo algoritmo.cpp.

#### 6.1.3.6. double Algoritmo::getModularidad ( )

Algoritmo::getModularidad Retorna la modularidad final obtenida con el mejor cromosoma encontrado.

**Devuelve**

double con la modularidad final.

Definición en la línea 198 del archivo algoritmo.cpp.

#### 6.1.3.7. void Algoritmo::inicializarConfiguracionInicial ( ) [private]

Algoritmo::inicializarConfiguracionInicial Inicializa la configuración incial de *clustering* en el formato de lista de adyacencia.

Definición en la línea 500 del archivo algoritmo.cpp.

#### 6.1.3.8. void Algoritmo::inicializarMatrizX ( ) [private]

Algoritmo::inicializarMatrizX Inicializa la matriz de membresia (binaria), tal y como el algoritmo de Newma configuró los *clusters*. La matriz tiene la forma descrita en el trabajo A Extraction Method of Overlapping Cluster Based on Network Structure Analysis Las filas corresponden a los *clusters* y las columnas a los nodos.

Definición en la línea 472 del archivo algoritmo.cpp.

#### 6.1.3.9. void Algoritmo::inicializarPoblacion ( ) [private]

Algoritmo::inicializarPoblacion Se crean los individuos iniciales de cromosomas, se usa la configuración de *clustering* retornada por Newman, y se garantiza que todos los cromosomas iniciales tengan solapamiento en un nodo.

Definición en la línea 229 del archivo algoritmo.cpp.

#### 6.1.3.10. void Algoritmo::reproducir ( ) [private]

Algoritmo::reproducir Para la reproducción, se usa la mutación sobre un cromosoma, esta puede darse de dos formas, según el se halla definido. Una es mutar un solo *bit* de la matriz X, y la otra mutar dos *bits* de la matriz X. Si el cromosoma es el indicado por el indice del mejor cromosoma en el mating pool, este no se muta y se pasa directo a la siguiente generación.

Definición en la línea 402 del archivo algoritmo.cpp.

#### 6.1.3.11. void Algoritmo::seleccionar ( ) [private]

Algoritmo::seleccionar La selección se realiza con la técnica torneo, se seleccionan dos cromosomas al azar, de cada uno se obtiene la modularidad, se elige el cromosoma con mayor modularidad. Si la modularidad llegara a ser igual en ambos cromosomas, se elige el cromosoma que tenga mayor porcentaje de *clusters* respecto a la cantidad de *clusters* encontrada al inicio con el algoritmo de Newman.

Además se guarda el indice del cromosoma en toda la población y que halla sido elegido para el *mating pool* y que sea el cromosoma con la mayor modularidad obtenida en el proceso.

Definición en la línea 309 del archivo algoritmo.cpp.

### 6.1.4. Documentación de los datos miembro

#### 6.1.4.1. int Algoritmo::aristas [private]

Definición en la línea 23 del archivo algoritmo.h.

#### 6.1.4.2. int Algoritmo::cantCambios [private]

Definición en la línea 44 del archivo algoritmo.h.

#### 6.1.4.3. int Algoritmo::cantClustersInicial [private]

Definición en la línea 43 del archivo algoritmo.h.

#### 6.1.4.4. vector<unordered_set<int> > Algoritmo::configuracionFinal [private]

Definición en la línea 51 del archivo algoritmo.h.

#### 6.1.4.5. vector<unordered_set<int> > Algoritmo::configuracionInicial [private]

Definición en la línea 49 del archivo algoritmo.h.

#### 6.1.4.6. int Algoritmo::generaciones [private]

Definición en la línea 26 del archivo algoritmo.h.

#### 6.1.4.7. igraph_t Algoritmo::grafo [private]

Definición en la línea 39 del archivo algoritmo.h.

#### 6.1.4.8. int Algoritmo::individuos [private]

Definición en la línea 25 del archivo algoritmo.h.

#### 6.1.4.9. vector<unordered_set<int> > Algoritmo::listaAdyacencia [private]

Definición en la línea 50 del archivo algoritmo.h.

#### 6.1.4.10. Cromosoma** Algoritmo::matingPool [private]

Definición en la línea 29 del archivo algoritmo.h.

#### 6.1.4.11. vector<vector<bool> > Algoritmo::matrizXInicial [private]

Definición en la línea 48 del archivo algoritmo.h.

#### 6.1.4.12. Cromosoma* Algoritmo::mejorCromosoma [private]

Definición en la línea 33 del archivo algoritmo.h.

#### 6.1.4.13. int Algoritmo::mejorIndicePoblacion [private]

Definición en la línea 31 del archivo algoritmo.h.

#### 6.1.4.14. double Algoritmo::mejorModularidad [private]

Definición en la línea 32 del archivo algoritmo.h.

#### 6.1.4.15. igraph_vector_t Algoritmo::membresiaInicial [private]

Definición en la línea 46 del archivo algoritmo.h.

#### 6.1.4.16. double Algoritmo::modularidadFinal [private]

Definición en la línea 42 del archivo algoritmo.h.

#### 6.1.4.17. double Algoritmo::modularidadInicial [private]

Definición en la línea 41 del archivo algoritmo.h.

#### 6.1.4.18. bool Algoritmo::mutarDosBits [private]

Definición en la línea 37 del archivo algoritmo.h.

#### 6.1.4.19. int Algoritmo::nodos [private]

Definición en la línea 22 del archivo algoritmo.h.

#### 6.1.4.20. Cromosoma** Algoritmo::poblacion [private]

Definición en la línea 28 del archivo algoritmo.h.

#### 6.1.4.21. double Algoritmo::porcentajeMatingPool [private]

Definición en la línea 36 del archivo algoritmo.h.

#### 6.1.4.22. igraph_vector_t Algoritmo::tamanioClustersInicial [private]

Definición en la línea 45 del archivo algoritmo.h.

#### 6.1.4.23. int Algoritmo::tamMatingPool [private]

Definición en la línea 35 del archivo algoritmo.h.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- clustering/algoritmo.h
- clustering/algoritmo.cpp

## 6.2. Referencia de la Clase Cromosoma

`#include <cromosoma.h>`

### Métodos públicos

- Cromosoma ()

  *Cromosoma::Cromosoma Constructor por defecto.*
- Cromosoma (vector< vector< bool > > &matrizXInicial, vector< unordered_set< int > > configuracionInicial, bool mutarDosBits, int nodos, int cantClustersInicial, int nodo, int cluster)

  *Cromosoma::Cromosoma Constructor. Se inicializa un objeto de la clase Cromosoma, donde se añade un solapamiento en el clustering.*
- ~Cromosoma ()

  *Cromosoma::~Cromosoma Destructor.*
- void calcularModularidad (vector< unordered_set< int > > &listaAdyacencia, int aristas)

  *Cromosoma::calcularModularidad Calcula la modularidad de la configuración de clustering que se ha realizado en un grafo.*
- void calcularProporcionClusters ()

  *Cromosoma::calcularProporcionClusters Calcula la proporción de clusters. Divide la cantidad de clusters actuales por la cantidad de clusters inicial.*
- double getModularidad ()

  *Cromosoma::getModularidad Retorna modularidad del cromosoma.*
- double getProporcionClusters ()

  *Cromosoma::getProporcionClusters Retorna la proporción de clusters respecto a la cantidad inicial.*
- void mutar (int nodosMutar[], int clustersMutar[])

  *Cromosoma::mutar Cambia a true la pertencia del nodo, si el nodo a mutar no pertenecía al cluster, o a false en caso contrario. Cuando cambia a false la pertenencia del nodo; se verifica que el nodo pertenezca al menos a un cluster, sino se deja el cromosoma sin mutar.*
- vector< unordered_set< int > > getConfiguracionClustering ()

  *Cromosoma::getConfiguracionClustering Retorna la configuracin de clustering obtenida en el cromosoma.*
- bool getCalculoModularidad ()

*Cromosoma::getCalculoModularidad* *Retorna el booleano que representa si ya se ha realizado el cálculo de la modularidad.*

- int getCantidadClustersFinal ()

  *Cromosoma::getCantidadClustersFinal* *Retorna la cantidad de clusters de la configuración final de clustering.*

#### Métodos privados

- double calcularFraccionAristasInternas (vector< unordered_set< int > > &listaAdyacencia, int clusterA, int clusterB, int aristas)

  *Cromosoma::calcularFraccionAristasInternas* *Calcula la cantidad la fracción de las aristas que conectan nodos que pertenencen al clusterA con nodos que pertenencen al clusterB.*
- double calcularFraccionAristasExternas (vector< unordered_set< int > > &listaAdyacencia, int clusterA, int aristas)

  *Cromosoma::calcularFraccionAristasExternas* *Calcula la fraccin de aristas que terminan en nodos del clusterA.*

#### Atributos privados

- vector< vector< bool > > matrizX
- vector< unordered_set< int > > configuracion
- double modularidad
- float proporcionClusters
- bool calculoModularidad
- int nodos
- int cantClustersInicial
- int cantClustersFinal
- igraph_vector_t tamanioClusters
- bool mutarDosBits

### 6.2.1. Descripción detallada

Definición en la línea 18 del archivo cromosoma.h.

### 6.2.2. Documentación del constructor y destructor

#### 6.2.2.1. Cromosoma::Cromosoma ( )

Cromosoma::Cromosoma Constructor por defecto.

Definición en la línea 18 del archivo cromosoma.cpp.

#### 6.2.2.2. Cromosoma::Cromosoma ( vector< vector< bool > > & *matrizXInicial*, vector< unordered_set< int > > *configuracionInicial*, bool *mutarDosBits*, int *nodos*, int *cantClustersInicial*, int *nodo*, int *cluster* )

Cromosoma::Cromosoma Constructor. Se inicializa un objeto de la clase Cromosoma, donde se añade un solapamiento en el *clustering*.

**Parámetros**

| | |
|---|---|
| *matrizXInicial* | vector<vector<bool> > referencia de la configuración de *clustering* que retornó el algoritmo de Newman. |
| *configuracion-Inicial* | vector<unordered_set<int> > con la configuración de *clustering* que retornó el algoritmo de Newman en formato de lista de adyacencia. |

| | |
|---|---|
| *mutarDosBits* | bool que indica si la mutación se debe realizar sobre un bit o sobre dos bits de la matriz de membresia. |
| *nodos* | int con el número total de nodos en el grafo. |
| *cantClusters-Inicial* | int con el número total de *clusters* que se obtuvieron con el algoritmo de Newman. |
| *nodo* | int con el número del nodo que hará solapamiento. |
| *cluster* | int con el número del *cluster* que tendrá un nodo mas en el grupo. |

Definición en la línea 34 del archivo cromosoma.cpp.

#### 6.2.2.3. Cromosoma::∼Cromosoma ( )

Cromosoma::∼Cromosoma Destructor.

Definición en la línea 86 del archivo cromosoma.cpp.

### 6.2.3. Documentación de las funciones miembro

#### 6.2.3.1. double Cromosoma::calcularFraccionAristasExternas ( vector< unordered_set< int > > & *listaAdyacencia*, int *clusterA*, int *aristas* ) [private]

Cromosoma::calcularFraccionAristasExternas Calcula la fraccin de aristas que terminan en nodos del *clusterA*.

**Parámetros**

| | |
|---|---|
| *listaAdyacencia* | vector<unordered_set<int> > referencia de la lista de adyacencia del grafo. |
| *clusterA* | int como identificador de un cluster. |
| *aristas* | int número total de aristas en el grafo. |

**Devuelve**

doble con la fracción de aristas externas de un *cluster*.

Definición en la línea 168 del archivo cromosoma.cpp.

#### 6.2.3.2. double Cromosoma::calcularFraccionAristasInternas ( vector< unordered_set< int > > & *listaAdyacencia*, int *clusterA*, int *clusterB*, int *aristas* ) [private]

Cromosoma::calcularFraccionAristasInternas Calcula la cantidad la fracción de las aristas que conectan nodos que pertenencen al *clusterA* con nodos que pertenencen al *clusterB*.

**Parámetros**

| | |
|---|---|
| *listaAdyacencia* | vector<unordered_set<int> > referencia de la lista de adyacencia del grafo. |
| *clusterA* | int como identificador de un cluster. |
| *clusterB* | int como identificador de un cluster. |
| *aristas* | int número total de aristas en el grafo. |

**Devuelve**

doble con la fracción de aristas internas de un *cluster*.

Definición en la línea 137 del archivo cromosoma.cpp.

#### 6.2.3.3. void Cromosoma::calcularModularidad ( vector< unordered_set< int > > & *listaAdyacencia*, int *aristas* )

Cromosoma::calcularModularidad Calcula la modularidad de la configuración de *clustering* que se ha realizado en un grafo.

**Parámetros**

| | |
|---|---|
| *listaAdyacencia* | vector<unordered_set<int> > referencia de la lista de adyacencia del grafo. |
| *aristas* | int número total de aristas en el grafo. |

Definición en la línea 98 del archivo cromosoma.cpp.

#### 6.2.3.4. void Cromosoma::calcularProporcionClusters ( )

Cromosoma::calcularProporcionClusters Calcula la proporción de *clusters*. Divide la cantidad de *clusters* actuales por la cantidad de *clusters* inicial.

Definición en la línea 120 del archivo cromosoma.cpp.

#### 6.2.3.5. bool Cromosoma::getCalculoModularidad ( )

Cromosoma::getCalculoModularidad Retorna el booleano que representa si ya se ha realizado el cálculo de la modularidad.

**Devuelve**

bool calculoModularidad.

Definición en la línea 291 del archivo cromosoma.cpp.

#### 6.2.3.6. int Cromosoma::getCantidadClustersFinal ( )

Cromosoma::getCantidadClustersFinal Retorna la cantidad de *clusters* de la configuración final de *clustering*.

**Devuelve**

int cantClustersFinal.

Definición en la línea 302 del archivo cromosoma.cpp.

#### 6.2.3.7. vector< unordered_set< int > > Cromosoma::getConfiguracionClustering ( )

Cromosoma::getConfiguracionClustering Retorna la configuracin de *clustering* obtenida en el cromosoma.

**Devuelve**

vector<unordered_set<int> > que contiene la configuración de los *clusters* en el formato de lista de adyacencia.

Definición en la línea 280 del archivo cromosoma.cpp.

#### 6.2.3.8. double Cromosoma::getModularidad ( )

Cromosoma::getModularidad Retorna modularidad del cromosoma.

**Devuelve**

double con el valor de la modularidad, para la configuración del cromosoma.

Definición en la línea 191 del archivo cromosoma.cpp.

#### 6.2.3.9. double Cromosoma::getProporcionClusters ( )

Cromosoma::getProporcionClusters Retorna la proporción de *clusters* respecto a la cantidad inicial.

**Devuelve**

double con el valor de la proporción de cantidad de *clusters* finales respecto a la cantidad de *clusters* inicial.

Definición en la línea 202 del archivo cromosoma.cpp.

#### 6.2.3.10. void Cromosoma::mutar ( int *nodosMutar[],* int *clustersMutar[]* )

Cromosoma::mutar Cambia a *true* la pertencia del nodo, si el nodo a mutar no pertenecía al *cluster*, o a *false* en caso contrario. Cuando cambia a *false* la pertenencia del nodo; se verifica que el nodo pertenezca al menos a un *cluster*, sino se deja el cromosoma sin mutar.

**Parámetros**

| | |
|---|---|
| *nodosMutar* | arreglo de int que contiene los nodos que serán mutados. |
| *clustersMutar* | arreglo de int que contiene los *clusters* que serán mutados. |

Definición en la línea 215 del archivo cromosoma.cpp.

### 6.2.4. Documentación de los datos miembro

#### 6.2.4.1. bool Cromosoma::calculoModularidad [private]

Definición en la línea 26 del archivo cromosoma.h.

#### 6.2.4.2. int Cromosoma::cantClustersFinal [private]

Definición en la línea 30 del archivo cromosoma.h.

#### 6.2.4.3. int Cromosoma::cantClustersInicial [private]

Definición en la línea 29 del archivo cromosoma.h.

#### 6.2.4.4. vector< unordered_set<int> > Cromosoma::configuracion [private]

Definición en la línea 22 del archivo cromosoma.h.

#### 6.2.4.5. vector< vector<bool> > Cromosoma::matrizX [private]

Definición en la línea 21 del archivo cromosoma.h.

#### 6.2.4.6. double Cromosoma::modularidad [private]

Definición en la línea 24 del archivo cromosoma.h.

#### 6.2.4.7. bool Cromosoma::mutarDosBits [private]

Definición en la línea 33 del archivo cromosoma.h.

#### 6.2.4.8. int Cromosoma::nodos [private]

Definición en la línea 28 del archivo cromosoma.h.

#### 6.2.4.9. float Cromosoma::proporcionClusters [private]

Definición en la línea 25 del archivo cromosoma.h.

#### 6.2.4.10. igraph_vector_t Cromosoma::tamanioClusters [private]

Definición en la línea 31 del archivo cromosoma.h.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- clustering/cromosoma.h
- clustering/cromosoma.cpp

## 6.3. Referencia de la Clase LectoraArchivo

`#include <lectoraarchivo.h>`

### Métodos públicos

- LectoraArchivo ()

  *LectoraArchivo::LectoraArchivo Constructor.*
- ∼LectoraArchivo ()

  *LectoraArchivo::∼LectoraArchivo Destructor.*
- void preprocesar (string nombreArchivo)

  *LectoraArchivo::preprocesar.*
- igraph_t getSubgrafo ()

  *LectoraArchivo::getSubgrafo Retorna igraph_t una copia exacta del subgrafo, que corresponde al componente conexo más grande del grafo original ingresado.*
- int getNodosOriginal ()

  *LectoraArchivo::getNodosOriginal Retorna la cantidad de nodos del grafo original.*
- int getNodosSubgrafo ()

  *LectoraArchivo::getNodosSubgrafo Retorna la cantidad de nodos del subgrafo obtenido(gran componente).*
- int getAristasOriginal ()

  *LectoraArchivo::getAristasOriginal Retorna la cantidad de aristas del grafo original.*
- int getAristasSubgrafo ()

  *LectoraArchivo::getAristasSubgrafo Retorna la cantidad de aristas del subgrafo obtenido(gran componente).*
- bool getFlagArchivo ()

  *LectoraArchivo::getFlagArchivo Retorna el valor de flagArchivo. si es FALSE existieron errores de lectura de archivo. Si es TRUE no hubo errores de lectura de archivo de grafo.*
- bool getFlagArchivoSolucion ()

  *LectoraArchivo::getFlagArchivoSolucion Retorna el valor de flagArchivoSolucion. si es FALSE existieron errores en la creación del archivo. Si es TRUE no hubo errores de creación del archivo de solución.*
- void crearArchivoSolucion (string nombreArchivo, igraph_t &grafo, vector< unordered_set< int > > configuracionFinal)

  *LectoraArchivo::crearArchivoSolucion Crea el archivo de solución en formato graphml que contiene toda la configuración de clustering encontrada en el algoritmo.*

### Métodos privados

- void obtenerGranComponente (const igraph_t ∗grafo, igraph_t ∗subgrafo)

  *LectoraArchivo::obtenerGranComponente.*

### Atributos privados

- string nombreArchivo
- int nodosOriginal
- int nodosSubgrafo
- int aristasOriginal
- int aristasSubgrafo
- bool flagArchivo
- bool flagArchivoSolucion
- bool flagGrafo
- bool flagSubgrafo
- igraph_t subgrafo
- igraph_t grafo

#### 6.3.1. Descripción detallada

Definición en la línea 19 del archivo lectoraarchivo.h.

#### 6.3.2. Documentación del constructor y destructor

##### 6.3.2.1. LectoraArchivo::LectoraArchivo ( )

LectoraArchivo::LectoraArchivo Constructor.

Definición en la línea 20 del archivo lectoraarchivo.cpp.

##### 6.3.2.2. LectoraArchivo::∼LectoraArchivo ( )

LectoraArchivo::∼LectoraArchivo Destructor.

Definición en la línea 33 del archivo lectoraarchivo.cpp.

#### 6.3.3. Documentación de las funciones miembro

##### 6.3.3.1. void LectoraArchivo::crearArchivoSolucion ( string *nombreArchivo*, igraph_t & *grafo*, vector< unordered_set< int > > *configuracionFinal* )

LectoraArchivo::crearArchivoSolucion Crea el archivo de solución en formato *graphml* que contiene toda la configuración de *clustering* encontrada en el algoritmo.

**Parámetros**

| | |
|---|---|
| *nombreArchivo* | string que contiene el nombre del archivo con el que se guarda. |
| *grafo* | igraph_t& que contiene el grafo que será guardado en el archivo. |
| *configuracion-Final* | vector<unordered_set<int> > que contiene la configuración final de *clustering* |

Definición en la línea 322 del archivo lectoraarchivo.cpp.

##### 6.3.3.2. int LectoraArchivo::getAristasOriginal ( )

LectoraArchivo::getAristasOriginal Retorna la cantidad de aristas del grafo original.

**Devuelve**

int aristas del grafo original.

Definición en la línea 273 del archivo lectoraarchivo.cpp.

##### 6.3.3.3. int LectoraArchivo::getAristasSubgrafo ( )

LectoraArchivo::getAristasSubgrafo Retorna la cantidad de aristas del subgrafo obtenido(gran componente).

**Devuelve**

int aristas del subgrafo.

Definición en la línea 284 del archivo lectoraarchivo.cpp.

##### 6.3.3.4. bool LectoraArchivo::getFlagArchivo ( )

LectoraArchivo::getFlagArchivo Retorna el valor de flagArchivo. si es FALSE existieron errores de lectura de archivo. Si es TRUE no hubo errores de lectura de archivo de grafo.

**Devuelve**

bool flagArchivo.

Definición en la línea 296 del archivo lectoraarchivo.cpp.

#### 6.3.3.5. bool LectoraArchivo::getFlagArchivoSolucion ( )

LectoraArchivo::getFlagArchivoSolucion Retorna el valor de flagArchivoSolucion. si es FALSE existieron errores en la creación del archivo. Si es TRUE no hubo errores de creación del archivo de solución.

**Devuelve**

bool flagArchivoSolucion.

Definición en la línea 308 del archivo lectoraarchivo.cpp.

#### 6.3.3.6. int LectoraArchivo::getNodosOriginal ( )

LectoraArchivo::getNodosOriginal Retorna la cantidad de nodos del grafo original.

**Devuelve**

int nodos del grafo original

Definición en la línea 251 del archivo lectoraarchivo.cpp.

#### 6.3.3.7. int LectoraArchivo::getNodosSubgrafo ( )

LectoraArchivo::getNodosSubgrafo Retorna la cantidad de nodos del subgrafo obtenido(gran componente).

**Devuelve**

int nodos del subgrafo.

Definición en la línea 262 del archivo lectoraarchivo.cpp.

#### 6.3.3.8. igraph_t LectoraArchivo::getSubgrafo ( )

LectoraArchivo::getSubgrafo Retorna igraph_t una copia exacta del subgrafo, que corresponde al componente conexo más grande del grafo original ingresado.

**Devuelve**

subgrafo obtenido en el proceso.

Definición en la línea 238 del archivo lectoraarchivo.cpp.

#### 6.3.3.9. void LectoraArchivo::obtenerGranComponente ( const igraph_t * *grafo*, igraph_t * *subgrafo* ) [private]

LectoraArchivo::obtenerGranComponente.

**Parámetros**

| | |
|---|---|
| *grafo* | Puntero al grafo original. |
| *subgrafo* | Puntero al grafo que se creará una vez se obtenga el gran componente del grafo original. |

**Nota**

Si el grafo fue cargado con edgelist los ids de los vertices corresponden a los números que habí en el archivo. Por el contrario si fue cargado con graphml los ids corresponderán a la configuración del archivo graphml.

Definición en la línea 145 del archivo lectoraarchivo.cpp.

#### 6.3.3.10. void LectoraArchivo::preprocesar ( string *nombreArchivo* )

LectoraArchivo::preprocesar.

**Parámetros**

| | |
|---|---|
| *nombreArchivo* | string con la ruta del archivo que contiene el grafo. Lee el archivo, extrae el grafo, si el grafo no es conexo obtiene el gran componente de éste. |

**Nota**

Si el archivo no existe, o es sintácticamente incorrecto el procesamiento no continua.

**Nota**

Si se esta trabajando con un grafo cargado con un formato de edgelist, se debe ajustar los atributos de los vertices para que estos no pierdan su id de identificación.

Definición en la línea 55 del archivo lectoraarchivo.cpp.

#### 6.3.4. Documentación de los datos miembro

##### 6.3.4.1. int LectoraArchivo::aristasOriginal [private]

Definición en la línea 26 del archivo lectoraarchivo.h.

##### 6.3.4.2. int LectoraArchivo::aristasSubgrafo [private]

Definición en la línea 27 del archivo lectoraarchivo.h.

##### 6.3.4.3. bool LectoraArchivo::flagArchivo [private]

Definición en la línea 28 del archivo lectoraarchivo.h.

##### 6.3.4.4. bool LectoraArchivo::flagArchivoSolucion [private]

Definición en la línea 29 del archivo lectoraarchivo.h.

##### 6.3.4.5. bool LectoraArchivo::flagGrafo [private]

Definición en la línea 30 del archivo lectoraarchivo.h.

##### 6.3.4.6. bool LectoraArchivo::flagSubgrafo [private]

Definición en la línea 31 del archivo lectoraarchivo.h.

##### 6.3.4.7. igraph_t LectoraArchivo::grafo [private]

Definición en la línea 34 del archivo lectoraarchivo.h.

##### 6.3.4.8. int LectoraArchivo::nodosOriginal [private]

Definición en la línea 24 del archivo lectoraarchivo.h.

##### 6.3.4.9. int LectoraArchivo::nodosSubgrafo [private]

Definición en la línea 25 del archivo lectoraarchivo.h.

##### 6.3.4.10. string LectoraArchivo::nombreArchivo [private]

Definición en la línea 22 del archivo lectoraarchivo.h.

##### 6.3.4.11. igraph_t LectoraArchivo::subgrafo [private]

Definición en la línea 33 del archivo lectoraarchivo.h.

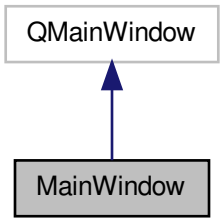
La documentación para esta clase fue generada a partir de los siguientes ficheros:

- clustering/lectoraarchivo.h
- clustering/lectoraarchivo.cpp

## 6.4. Referencia de la Clase MainWindow

`#include <mainwindow.h>`

Diagrama de herencias de MainWindow




### Métodos públicos

- MainWindow (QWidget ∗parent=0)

  *MainWindow::MainWindow(QWidget ∗parent) Constructor. Inicializa objeto MainWindow.*
- ∼MainWindow ()

  *MainWindow::∼MainWindow() Destructor.*

### Slots privados

- void ejecutar ()

  *MainWindow::ejecutar() Ejecuta el algoritmo. Y actualiza los campos con los datos obtenidos con el algoritmo.*
- void abrirArchivo ()

  *MainWindow::abrirArchivo() Este método permite cargar un archivo de grafo, como primer paso para aplicar el algoritmo.*
- void guardarSolucion ()

  *MainWindow::guardarSolucion() Guarda la solucion encontrada en formato graphml.*

### Métodos privados

- void deshabilitarAlgoritmo (bool habilitar)

  *MainWindow::deshabilitarAlgoritmo Habilita o deshabilita la opción de usar el algoritmo.*
- void limpiarDatos ()

  *MainWindow::limpiarDatos Limpia los campos de los paneles de datos de entrada y datos de salida.*
- void destruirDatos ()

#### Atributos privados

- Ui::MainWindow ∗ ui
- Algoritmo ∗ algoritmo
- LectoraArchivo ∗ lectoraDeArchivo
- bool flagLectora
- bool flagAlgoritmo

### 6.4.1. Descripción detallada

Definición en la línea 20 del archivo mainwindow.h.

### 6.4.2. Documentación del constructor y destructor

#### 6.4.2.1. MainWindow::MainWindow ( QWidget ∗ *parent* = 0 ) [explicit]

MainWindow::MainWindow(QWidget ∗parent) Constructor. Inicializa objeto MainWindow.

**Parámetros**

| | |
|---|---|
| *parent* | |

Definición en la línea 25 del archivo mainwindow.cpp.

#### 6.4.2.2. MainWindow::∼MainWindow ( )

MainWindow::∼MainWindow() Destructor.

Definición en la línea 49 del archivo mainwindow.cpp.

### 6.4.3. Documentación de las funciones miembro

#### 6.4.3.1. void MainWindow::abrirArchivo ( ) [private], [slot]

MainWindow::abrirArchivo() Este método permite cargar un archivo de grafo, como primer paso para aplicar el algoritmo.

**Nota**

Los formatos admitidos son .graphml y .txt El formato .txt corresponde al formato de edgelist de un grafo.
A continuación un ejemplo del formato .txt (edgelist):
1 2
3 6
7 8

Para conocer más acerca del formato .graphml visite http://graphml.graphdrawing.org/primer/graphml-primer.html

Definición en la línea 157 del archivo mainwindow.cpp.

#### 6.4.3.2. void MainWindow::deshabilitarAlgoritmo ( bool *habilitar* ) [private]

MainWindow::deshabilitarAlgoritmo Habilita o deshabilita la opción de usar el algoritmo.

**Parámetros**

| | |
|---|---|
| *habilitar* | bool |

Definición en la línea 73 del archivo mainwindow.cpp.

#### 6.4.3.3. void MainWindow::destruirDatos ( ) [private]

#### 6.4.3.4. void MainWindow::ejecutar ( ) [private], [slot]

MainWindow::ejecutar() Ejecuta el algoritmo. Y actualiza los campos con los datos obtenidos con el algoritmo.

Definición en la línea 108 del archivo mainwindow.cpp.

#### 6.4.3.5. void MainWindow::guardarSolucion ( ) [private], [slot]

MainWindow::guardarSolucion() Guarda la solucion encontrada en formato *graphml*.

Definición en la línea 232 del archivo mainwindow.cpp.

#### 6.4.3.6. void MainWindow::limpiarDatos ( ) [private]

MainWindow::limpiarDatos Limpia los campos de los paneles de datos de entrada y datos de salida.

Definición en la línea 88 del archivo mainwindow.cpp.

### 6.4.4. Documentación de los datos miembro

#### 6.4.4.1. Algoritmo* MainWindow::algoritmo [private]

Definición en la línea 30 del archivo mainwindow.h.

#### 6.4.4.2. bool MainWindow::flagAlgoritmo [private]

Definición en la línea 35 del archivo mainwindow.h.

#### 6.4.4.3. bool MainWindow::flagLectora [private]

Definición en la línea 34 del archivo mainwindow.h.

#### 6.4.4.4. LectoraArchivo* MainWindow::lectoraDeArchivo [private]

Definición en la línea 31 del archivo mainwindow.h.

#### 6.4.4.5. Ui::MainWindow* MainWindow::ui [private]

Definición en la línea 29 del archivo mainwindow.h.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- clustering/mainwindow.h
- clustering/mainwindow.cpp

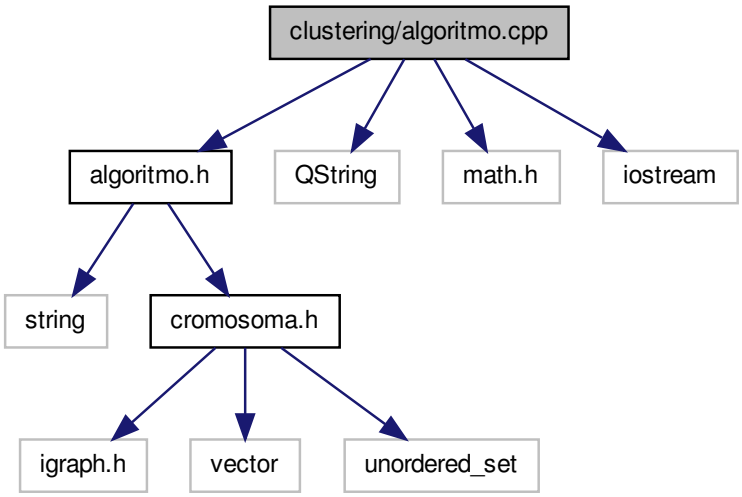
# 7. Documentación de archivos

## 7.1. Referencia del Archivo clustering/algoritmo.cpp

Implementación de los mtodos de la clase Algoritmo.

```
#include "algoritmo.h"
#include <QString>
#include <math.h>
#include <iostream>
```

Dependencia gráfica adjunta para algoritmo.cpp:




#### 7.1.1. Descripción detallada

Implementación de los mtodos de la clase Algoritmo.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo algoritmo.cpp.

## 7.2. Referencia del Archivo clustering/algoritmo.h

Definición de la clase Algoritmo.

```
#include <string>
#include "cromosoma.h"
```

Dependencia gráfica adjunta para algoritmo.h:

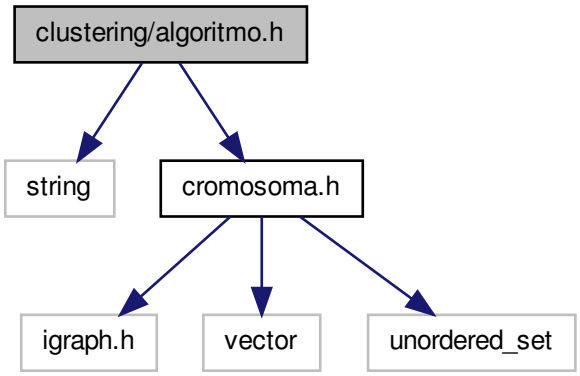
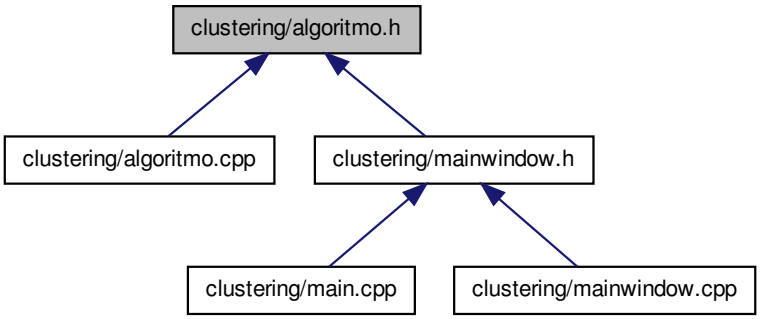



Gráfico de los archivos que directa o indirectamente incluyen a este archivo:




**Clases**

- class Algoritmo

### 7.2.1. Descripción detallada

Definición de la clase Algoritmo.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

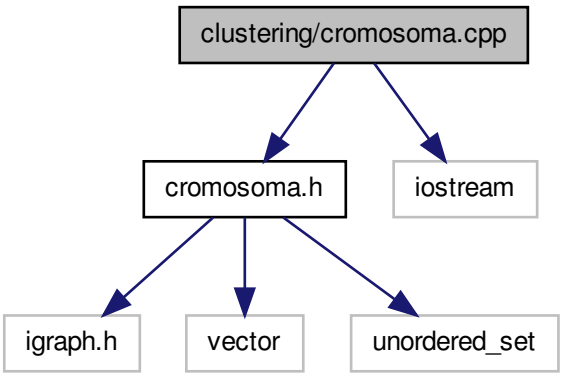
Definición en el archivo algoritmo.h.

## 7.3. Referencia del Archivo clustering/cromosoma.cpp

Implementación de los mtodos de la clase Cromosoma.

```
#include "cromosoma.h"
#include <iostream>
```

Dependencia gráfica adjunta para cromosoma.cpp:




### 7.3.1. Descripción detallada

Implementación de los mtodos de la clase Cromosoma.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo cromosoma.cpp.

## 7.4. Referencia del Archivo clustering/cromosoma.h

Definición de la clase Cromosoma.

```
#include <igraph.h>
#include <vector>
#include <unordered_set>
```

Dependencia gráfica adjunta para cromosoma.h:

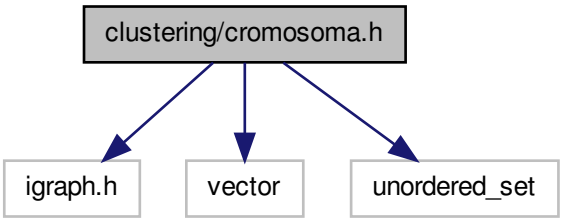
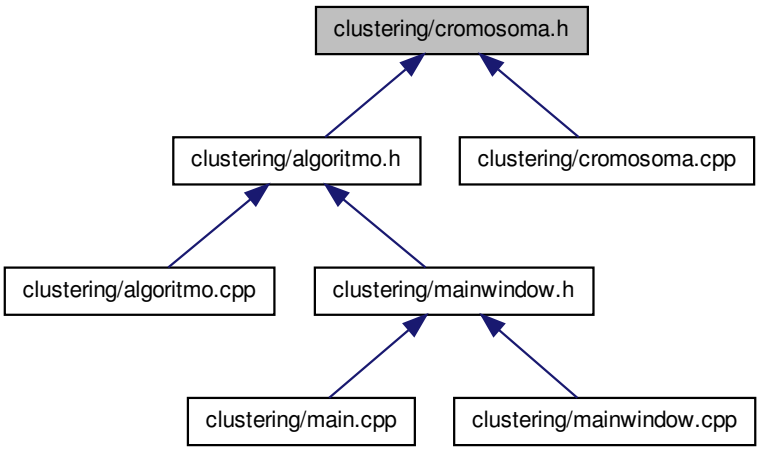



Gráfico de los archivos que directa o indirectamente incluyen a este archivo:




**Clases**

- class Cromosoma

#### 7.4.1. Descripción detallada

Definición de la clase Cromosoma.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

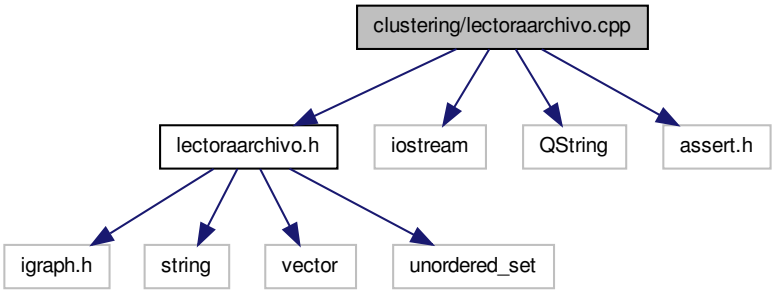
Definición en el archivo cromosoma.h.

## 7.5. Referencia del Archivo clustering/lectoraarchivo.cpp

Implementación de los mtodos de la clase LectoraArchivo.

```
#include "lectoraarchivo.h"
#include <iostream>
#include <QString>
#include <assert.h>
```

Dependencia gráfica adjunta para lectoraarchivo.cpp:




### 7.5.1. Descripción detallada

Implementación de los mtodos de la clase LectoraArchivo.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo lectoraarchivo.cpp.

## 7.6. Referencia del Archivo clustering/lectoraarchivo.h

Definición de la clase LectoraArchivo.

```
#include <igraph.h>
#include <string>
#include <vector>
#include <unordered_set>
```

Dependencia gráfica adjunta para lectoraarchivo.h:

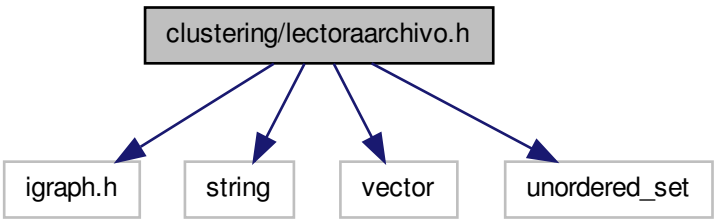
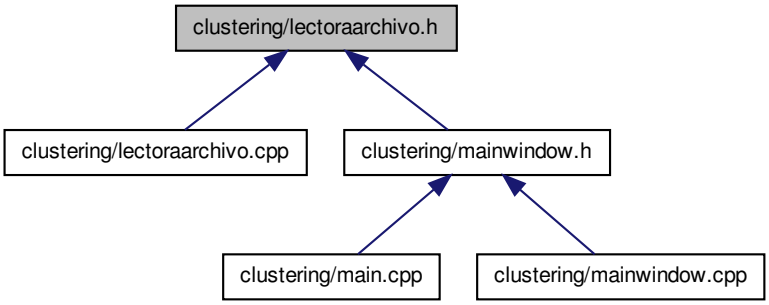



Gráfico de los archivos que directa o indirectamente incluyen a este archivo:




**Clases**

- class LectoraArchivo

#### 7.6.1. Descripción detallada

Definición de la clase LectoraArchivo.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

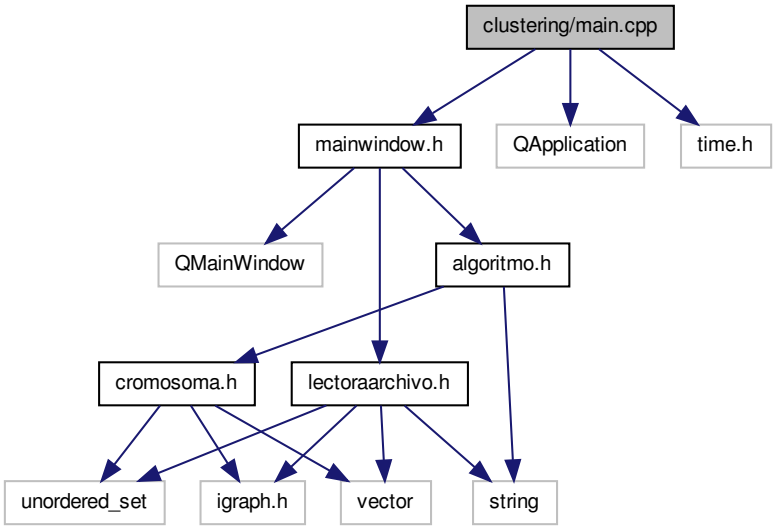
Definición en el archivo lectoraarchivo.h.

## 7.7. Referencia del Archivo clustering/main.cpp

Archivo principal del proyecto.

```
#include "mainwindow.h"
#include <QApplication>
#include <time.h>
```

Dependencia gráfica adjunta para main.cpp:




### Funciones

- int main (int argc, char ∗argv[])

  *main Se inicializa la aplicación.*

### 7.7.1. Descripción detallada

Archivo principal del proyecto.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo main.cpp.

### 7.7.2. Documentación de las funciones

#### 7.7.2.1. int main ( int *argc*, char ∗ *argv[]* )

main Se inicializa la aplicación.

**Nota**

En algoritmo geneticos es recomendables sólo ejecutar la inicializacion del Rand una vez en toda la ejecución del algoritmo.

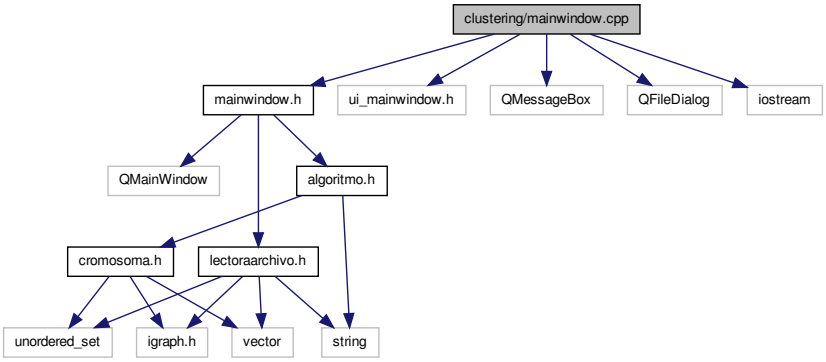
Definición en la línea 27 del archivo main.cpp.

## 7.8. Referencia del Archivo clustering/mainwindow.cpp

Implementación de los mtodos de la clase MainWindow.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <QFileDialog>
#include <iostream>
```

Dependencia gráfica adjunta para mainwindow.cpp:




### 7.8.1. Descripción detallada

Implementación de los mtodos de la clase MainWindow.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo mainwindow.cpp.

## 7.9. Referencia del Archivo clustering/mainwindow.h

Definición de la clase MainWindow.

```
#include <QMainWindow>
#include "algoritmo.h"
#include "lectoraarchivo.h"
```

Dependencia gráfica adjunta para mainwindow.h:

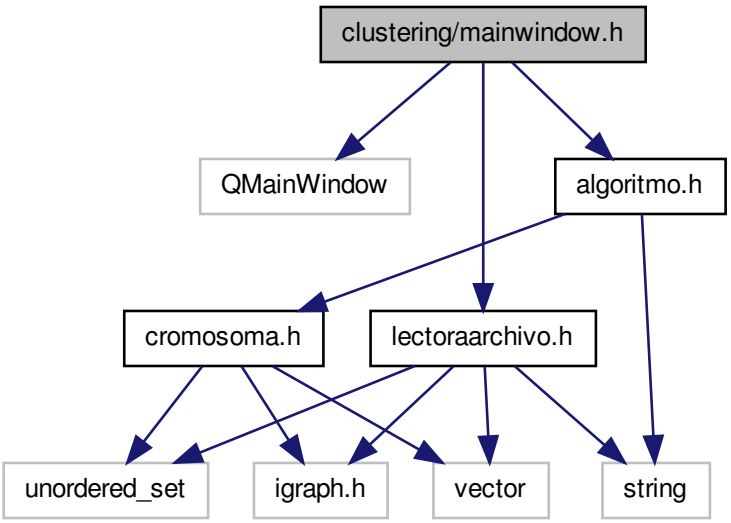
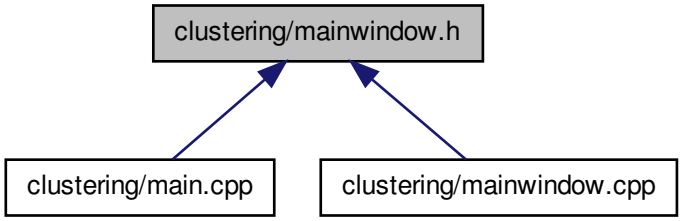



Gráfico de los archivos que directa o indirectamente incluyen a este archivo:




#### Clases

- class MainWindow

#### Namespaces

- Ui

#### Constant Groups

- Ui

#### 7.9.1. Descripción detallada

Definición de la clase MainWindow.

**Autor**

María Andrea Cruz Blandón

**Fecha**

09/2013, 11/2013

Definición en el archivo mainwindow.h.

## Índice alfabético