

4EnRatlla CNNGenètic

Generat per Doxygen 1.9.1



# Chapter 1

## Llista de coses per fer

Global **comprovarSolucioDiagonal1** (QuatreEnRatlla \*partida, int fila, int col)

Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

Global **comprovarSolucioDiagonal2** (QuatreEnRatlla \*partida, int fila, int col)

Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

Global **comprovarSolucioHoritzontal** (QuatreEnRatlla \*partida, int fila, int col)

Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

Global **comprovarSolucioVertical** (QuatreEnRatlla \*partida, int fila, int col)

Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

Classe **contextHeuristica**

potser es podria implementar la profunditat variable per aquí.

Global **iniciarPartida** (selectorDeMoviment decisioJ1, selectorDeMoviment decisioJ2, double esperaEntre↵  
Torns, void \*ctx)

Global **iteracioMinmax** (QuatreEnRatlla \*partida, signed char jugadorOriginal, int nivellNode, double  
\*puntuacioNode, int \*profunditatNode, funcioHeuristica fHeuristica, void \*ctxHeuristica)

Global **minMax** (QuatreEnRatlla \*partida, signed char jugadorOriginal, funcioHeuristica fHeuristica, void  
\*ctxHeuristica)

s'hauria de ficar la profunditat com a paràmetre



## Chapter 2

# Índex d'Estructures de Dades

### 2.1 Estructures de Dades

Aquestes són les estructures de dades acompanyades amb breus descripcions:

<a href="#">capaXarxa</a>	Estructura que conté una capa de la xarxa . . . . .	??
<a href="#">CNN</a>	Estructura que encapsula una xarxa neuronal . . . . .	??
<a href="#">contextHeuristica</a>	Estructura que conté les funcions heurístiques de la partida i altres paràmetres necessaris per a l'execució d'aquesta . . . . .	??
<a href="#">generacio</a>	Estructura que controla com es formen les generacions . . . . .	??
<a href="#">quatreEnRatlla</a>	Estructura que conté l'estat d'una partida . . . . .	??



## Chapter 3

# Índex de Fitxers

### 3.1 Llista dels Fitxers

Aquesta és la llista de tots els fitxers documentats acompanyats amb breus descripcions:

<a href="#">4enratlla.c</a>		
	Fitxer que implementa les funcions bàsiques per a poder jugar una partida del quatre en ratlla	??
<b>4enratlla.h</b>		??
<a href="#">conexioXarxa4EnRatlla.c</a>		
	Implementa un wrapper per a poder fer servir una xarxa com a funció heurística	??
<b>conexioXarxa4EnRatlla.h</b>		??
<a href="#">Entrenament.c</a>		
	Fitxer que implementa les funcions necessàries per a l'entrenament de la <a href="#">CNN</a>	??
<b>Entrenament.h</b>		??
<a href="#">funcioUtilitat.c</a>		
	Fitxer que implementa una funció heurística que valora l'adjacència	??
<b>funcioUtilitat.h</b>		??
<a href="#">minmax.c</a>		
	Fitxer que implementa l'algorisme del minMax	??
<b>minmax.h</b>		??
<a href="#">partides.c</a>		
	Fitxer que inclou totes les funcions relacionades amb la interfície de la partida i la tria de moviments	??
<b>partides.h</b>		??
<a href="#">Utilitats.c</a>		
	Fitxer que inclou funcions amb utilitats diverses sense dependències	??
<b>Utilitats.h</b>		??
<a href="#">ValidacioML.c</a>		
	Fitxer per a validar les xarxes	??
<a href="#">Xarxa.c</a>		
	Fitxer que inclou totes les funcions relacionades les operacions de les <a href="#">CNN</a>	??
<b>Xarxa.h</b>		??





## Chapter 4

# Documentació de les Estructures de Dades

### 4.1 Referència de l'Estructura capaXarxa

Estructura que conté una capa de la xarxa.

```
#include <Xarxa.h>
```

Diagrama de col·laboració per a capaXarxa:



### Camps de Dades

- int [nombreKernels](#)
- double \*\*\* [kernels](#)
- double \* [biaixos](#)
- funcioReal [funcioActivacio](#)
- int [dimKer](#)
- int [nMatrius](#)
- int [dimFil](#)
- int [dimCol](#)

### 4.1.1 Descripció Detallada

Estructura que conté una capa de la xarxa.

Definició a la línia 19 del fitxer Xarxa.h.

### 4.1.2 Documentació dels Camps

#### 4.1.2.1 biaixos

```
double* capaXarxa::biaixos
```

Biaixos de cada kernel

Definició a la línia 22 del fitxer Xarxa.h.

#### 4.1.2.2 dimCol

```
int capaXarxa::dimCol
```

Nombre de columnes que tenen les matrius a processar

Definició a la línia 27 del fitxer Xarxa.h.

#### 4.1.2.3 dimFil

```
int capaXarxa::dimFil
```

Nombre de files que tenen les matrius a processar

Definició a la línia 26 del fitxer Xarxa.h.

#### 4.1.2.4 dimKer

```
int capaXarxa::dimKer
```

Dimensió dels kernels

Definició a la línia 24 del fitxer Xarxa.h.

#### 4.1.2.5 funcioActivacio

```
funcioReal capaXarxa::funcioActivacio
```

La funció d'activació que fa servir la capa

Definició a la línia 23 del fitxer Xarxa.h.

#### 4.1.2.6 kernels

```
double*** capaXarxa::kernels
```

Nuclis de la capa

Definició a la línia 21 del fitxer Xarxa.h.

#### 4.1.2.7 nMatrius

```
int capaXarxa::nMatrius
```

Nombre de matrius a processar que coincideix amb la profunditat dels kernels

Definició a la línia 25 del fitxer Xarxa.h.

#### 4.1.2.8 nombreKernels

```
int capaXarxa::nombreKernels
```

Nombre de kernels que té la capa

Definició a la línia 20 del fitxer Xarxa.h.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

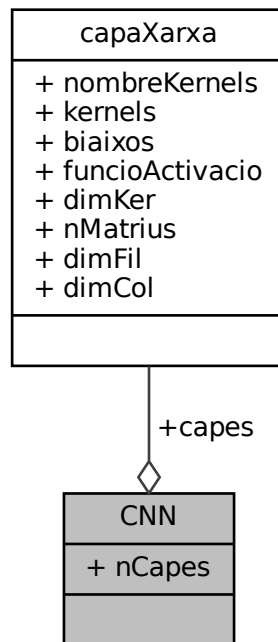
- Xarxa.h

## 4.2 Referència de l'Estructura CNN

Estructura que encapsula una xarxa neuronal.

```
#include <Xarxa.h>
```

Diagrama de col·laboració per a CNN:



### Camps de Dades

- `int nCapes`
- `CapaXarxa ** capes`

#### 4.2.1 Descripció Detallada

Estructura que encapsula una xarxa neuronal.

Sempre té una capa extra que suma tots els valors de la última capa.

Definició a la línia 37 del fitxer Xarxa.h.

#### 4.2.2 Documentació dels Camps

#### 4.2.2.1 capes

```
CapaXarxa** CNN::capes
```

Array de capes

Definició a la línia 39 del fitxer Xarxa.h.

#### 4.2.2.2 nCapes

```
int CNN::nCapes
```

Nombre de capes de la xarxa

Definició a la línia 38 del fitxer Xarxa.h.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

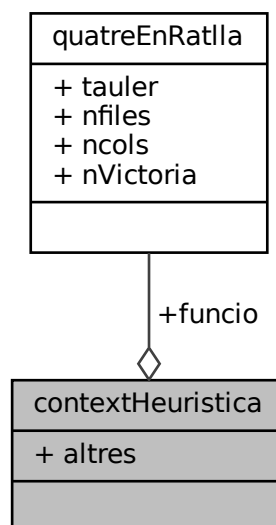
- Xarxa.h

### 4.3 Referència de l'Estructura contextHeuristica

Estructura que conté les funcions heurístiques de la partida i altres paràmetres necessaris per a l'execució d'aquesta.

```
#include <partides.h>
```

Diagrama de col·laboració per a contextHeuristica:



## Camps de Dades

- funcioHeuristica [funcio](#) [2]
- void \* [altres](#) [2]

### 4.3.1 Descripció Detallada

Estructura que conté les funcions heurístiques de la partida i altres paràmetres necessaris per a l'execució d'aquesta.

[Per fer](#) potser es podria implementar la profunditat variable per aquí.

Definició a la línia 26 del fitxer partides.h.

### 4.3.2 Documentació dels Camps

#### 4.3.2.1 altres

```
void* contextHeuristica::altres[2]
```

Un apuntador a altres paràmetres necessaris per al funcionament de les funcions

Definició a la línia 28 del fitxer partides.h.

#### 4.3.2.2 funcio

```
funcioHeuristica contextHeuristica::funcio[2]
```

Un apuntador a las funcions heurístiques del J1 y del J2

Definició a la línia 27 del fitxer partides.h.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

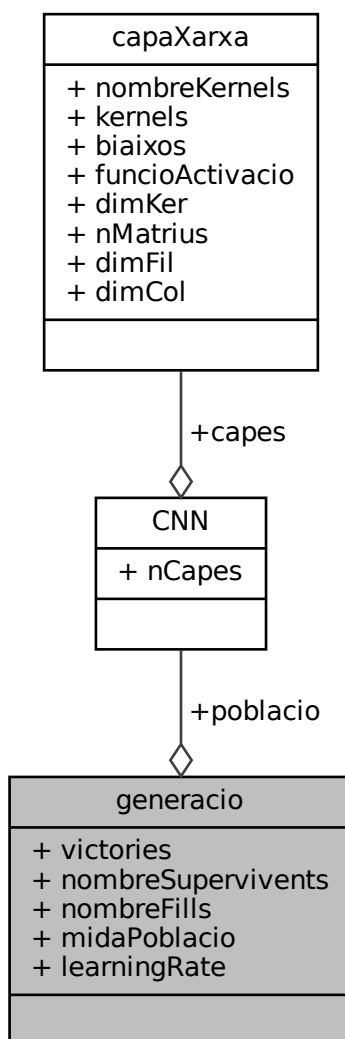
- partides.h

## 4.4 Referència de l'Estructura generacio

Estructura que controla com es formen les generacions.

```
#include <Entrenament.h>
```

Diagrama de col·laboració per a generacio:



### Camps de Dades

- `XarxaNeuronal` \*\* `poblacio`
- `int` \* `victories`
- `int` `nombreSupervivents`
- `int` `nombreFills`
- `int` `midaPoblacio`
- `double` `learningRate`

#### 4.4.1 Descripció Detallada

Estructura que controla com es formen les generacions.

Definició a la línia 12 del fitxer Entrenament.h.

#### 4.4.2 Documentació dels Camps

##### 4.4.2.1 learningRate

```
double generacio::learningRate
```

El ratio d'aprenentatge

Definició a la línia 18 del fitxer Entrenament.h.

##### 4.4.2.2 midaPoblacio

```
int generacio::midaPoblacio
```

La mida total de la població (ha de ser  $\text{nombreSupervivents} * \text{midaPoblacio}$ )

Definició a la línia 17 del fitxer Entrenament.h.

##### 4.4.2.3 nombreFills

```
int generacio::nombreFills
```

El nombre de fills que té cada supervivent

Definició a la línia 16 del fitxer Entrenament.h.

##### 4.4.2.4 nombreSupervivents

```
int generacio::nombreSupervivents
```

El nombre de supervivents d'una iteració

Definició a la línia 15 del fitxer Entrenament.h.



#### 4.4.2.5 poblacio

```
XarxaNeuronal** generacio::poblacio
```

Array d'apuntadors a les xarxes neuronals que conforma la població

Definició a la línia 13 del fitxer Entrenament.h.

#### 4.4.2.6 victories

```
int* generacio::victories
```

Array que desa a la posició i la quantitat de victòries que ha aconseguit la xarxa i

Definició a la línia 14 del fitxer Entrenament.h.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

- Entrenament.h

## 4.5 Referència de l'Estructura quatreEnRatlla

Estructura que conté l'estat d'una partida.

```
#include <4enratlla.h>
```

Diagrama de col·laboració per a quatreEnRatlla:



### Camps de Dades

- signed char \*\* **tauler**
- int **nfiles**
- int **ncols**
- int **nVictoria**

### 4.5.1 Descripció Detallada

Estructura que conté l'estat d'una partida.

Definició a la línia 19 del fitxer 4enratlla.h.

### 4.5.2 Documentació dels Camps

#### 4.5.2.1 ncols

```
int quatreEnRatlla::ncols
```

nombre de columnes del taulell

Definició a la línia 22 del fitxer 4enratlla.h.

#### 4.5.2.2 nfiles

```
int quatreEnRatlla::nfiles
```

nombre de files del taulell

Definició a la línia 21 del fitxer 4enratlla.h.

#### 4.5.2.3 nVictoria

```
int quatreEnRatlla::nVictoria
```

Nombre de peces iguals que s'han d'alinear per a guanyar la partida

Definició a la línia 23 del fitxer 4enratlla.h.

#### 4.5.2.4 tauler

```
signed char** quatreEnRatlla::tauler
```

Taulell actual amb les fitxes col·locades. Ha de ser una matriu de mida nfiles x ncols. L'índex 0,0 és la posició superior a l'esquerra

Definició a la línia 20 del fitxer 4enratlla.h.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

- 4enratlla.h

## Chapter 5

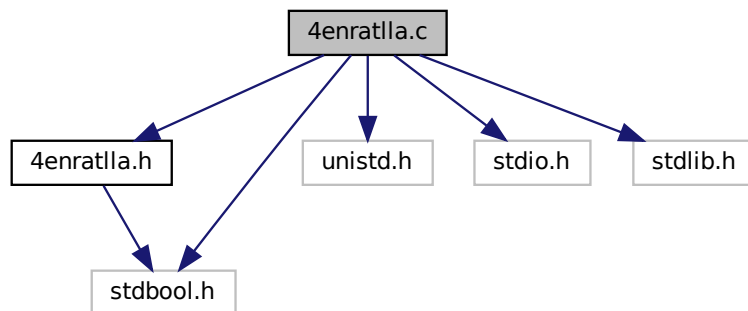
# Documentació dels Fitxers

### 5.1 Referència del Fitxer 4enratlla.c

Fitxer que implementa les funcions bàsiques per a poder jugar una partida del quatre en ratlla.

```
#include "4enratlla.h"  
#include <unistd.h>  
#include <stdio.h>  
#include <stdbool.h>  
#include <stdlib.h>
```

Inclou el graf de dependències per a 4enratlla.c:



### Funcions

- void [imprimirQuatreEnRatlla](#) ([QuatreEnRatlla](#) \*partida)  
*Imprimeix un taulell del 4 en ratlla.*
- void [reiniciarQuatreEnRatlla](#) ([QuatreEnRatlla](#) \*partida)  
*Fica tots els valors del tauler a 0.*
- void [inicialitzarQuatreEnRatlla](#) ([QuatreEnRatlla](#) \*partida, int nFils, int nCols, int nVictoria)  
*Inicia una partida del 4 en ratlla des de zero.*
- void [alliberarQuatreEnRatlla](#) ([QuatreEnRatlla](#) \*partida)

- *Allibera la partida del 4 en ratlla quan ja no es necessita.*
- bool `comprovarColumnaPlena` (`QuatreEnRatlla` \*partida, int moviment)
- *Comprova si hi ha espai en una columna per a introduir fitxes.*
- void `realitzarMoviment` (`QuatreEnRatlla` \*partida, int moviment, signed char jugador)
- *Introdueix una fitxa d'un jugador en una columna.*
- void `desferMoviment` (`QuatreEnRatlla` \*partida, int moviment)
- *Desfar un moviment d'una partida.*
- int `filaSuperior` (`QuatreEnRatlla` \*partida, int columna)
- *Donada una partida y una columna retorna la fila on és hi ha la peça més alta.*
- bool `comprovarSolucioHoritzontal` (`QuatreEnRatlla` \*partida, int fila, int col)
- *Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla horitzontal.*
- bool `comprovarSolucioVertical` (`QuatreEnRatlla` \*partida, int fila, int col)
- *Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla verical.*
- bool `comprovarSolucioDiagonal1` (`QuatreEnRatlla` \*partida, int fila, int col)
- *Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla diagonal d'esquerra a dretas.*
- bool `comprovarSolucioDiagonal2` (`QuatreEnRatlla` \*partida, int fila, int col)
- *Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla diagonal de dreta a esquerra.*
- bool `comprovarSolucio` (`QuatreEnRatlla` \*partida, int ultimMoviment)
- *Comprova si un moviment fet ha guanyat la partida.*
- bool `comprovarEmpat` (`QuatreEnRatlla` \*partida)
- *Comprova si s'ha empatat la partida.*

### 5.1.1 Descripció Detallada

Fitxer que implementa les funcions bàsiques per a poder jugar una partida del quatre en ratlla.

### 5.1.2 Documentació de les Funcions

#### 5.1.2.1 alliberarQuatreEnRatlla()

```
void alliberarQuatreEnRatlla (
    QuatreEnRatlla * partida )
```

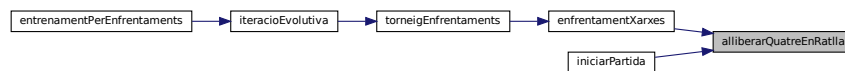
Allibera la partida del 4 en ratlla quan ja no es necessita.

#### Paràmetres

<code>partida</code>	és un apuntador a la partida que es vol alliberar.
----------------------	--

Definició a la línia 50 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.2 comprovarColumnaPlena()

```

bool comprovarColumnaPlena (
    QuatreEnRatlla * partida,
    int moviment )
  
```

Comprova si hi ha espai en una columna per a introduir fitxes.

Només té en compte si les columnes estan plenes, no considera si el nombre està fora del taulell.

#### Paràmetres

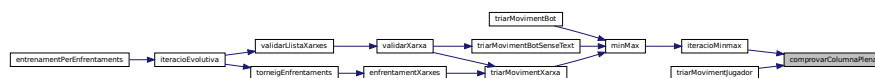
<i>partida</i>	és una apuntador a la partida que es vol comprovar si es pot fer el moviment.
<i>moviment</i>	és el número de columna que es vol comprovar si es pot introduir una fitxa.

#### Retorna

true si la columna es plena i false si hi ha espai.

Definició a la línia 56 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.3 comprovarEmpat()

```

bool comprovarEmpat (
    QuatreEnRatlla * partida )
  
```

Comprova si s'ha empatat la partida.

**Paràmetres**

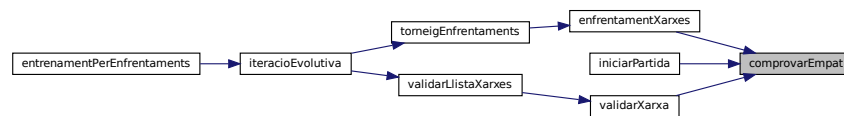
<i>partida</i>	un apuntador a la partida que es comprova.
----------------	--

**Retorna**

true si la partida s'ha empatat i false en cas contrari.

Definició a la línia 165 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:

**5.1.2.4 comprovarSolucio()**

```

bool comprovarSolucio (
    QuatreEnRatlla * partida,
    int ultimMoviment )
  
```

Comprova si un moviment fet ha guanyat la partida.

No comprova que l'últim moviment donat realment ho sigui. És important no ficar un moviment que no 'shagi fet.

**Paràmetres**

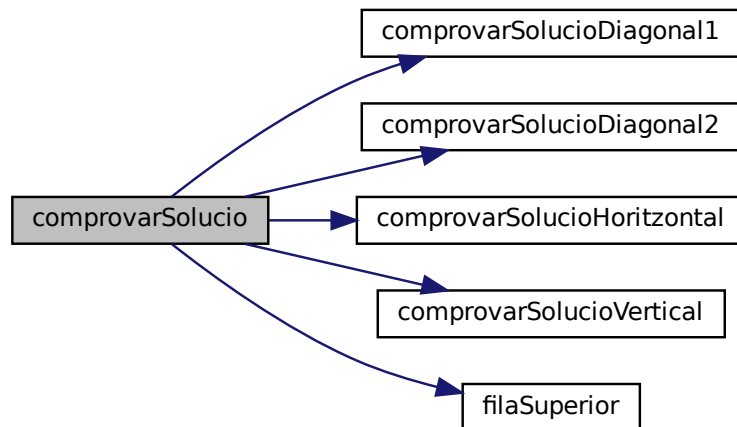
<i>partida</i>	és un apuntador a la partida que es vol comprovar.
<i>int</i>	és l'últim moviment fet on es comprovarà si forma part d'una solució.

Retorna

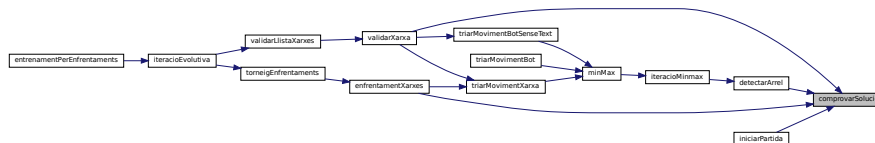
true si s'ha guanyat la partida i false si la partida continua.

Definició a la línia 160 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.1.2.5 comprovarSolucioDiagonal1()

```

bool comprovarSolucioDiagonal1 (
    QuatreEnRatlla * partida,
    int fila,
    int col )
  
```

Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla diagonal d'esquerra a dretes.

**Per fer** Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que són molt semblants.

**Paràmetres**

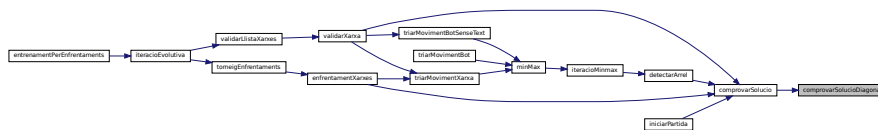
<i>partida</i>	és la partida que es vol comprovar.
<i>fila</i>	és la fila on s'ha fet el moviment.
<i>col</i>	és la columna on s'ha fet el moviment.

**Retorna**

true si s'ha guanyat la partida i false en cas de no ser la solució diagonal.

Definició a la línia 118 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:

**5.1.2.6 comprovarSolucioDiagonal2()**

```

bool comprovarSolucioDiagonal2 (
    QuatreEnRatlla * partida,
    int fila,
    int col )

```

Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla diagonal de dreta a esquerra.

**Per fer** Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

**Paràmetres**

<i>partida</i>	és la partida que es vol comprovar.
<i>fila</i>	és la fila on s'ha fet el moviment.
<i>col</i>	és la columna on s'ha fet el moviment.

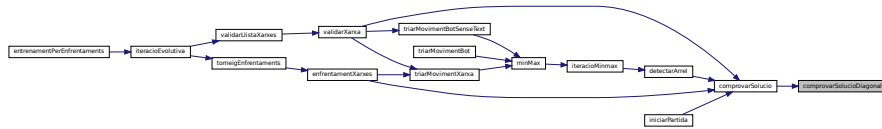
**Retorna**

true si s'ha guanyat la partida i false en cas de no ser la solució diagonal.

Definició a la línia 138 del fitxer 4enratlla.c.



Gràfic de crides a aquesta funció:



### 5.1.2.7 comprovarSolucioHoritzontal()

```

bool comprovarSolucioHoritzontal (
    QuatreEnRatlla * partida,
    int fila,
    int col )
  
```

Donat on s'ha inserit l'última peça retorna si aquest moviment provoca un k en ratlla horitzontal.

**Per fer** Segurament aquesta funció es pot fusionar d'alguna manera amb les altres comprovar solució, ja que son molt semblans.

#### Paràmetres

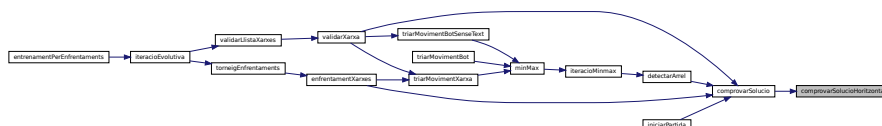
<i>partida</i>	és la partida que es vol comprovar.
<i>fila</i>	és la fila on s'ha fet el moviment.
<i>col</i>	és la columna on s'ha fet el movient

#### Retorna

true si s'ha guanyat la partida i false en cas de no ser la solució horitzontal.

Definició a la línia 85 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



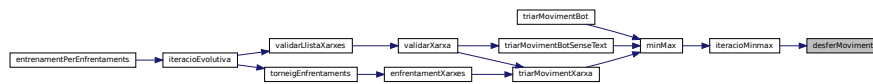


Definició a la línia 70 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



#### 5.1.2.10 filaSuperior()

```

int filaSuperior (
    QuatreEnRatlla * partida,
    int columna )
  
```

Donada una partida y una columna retorna la fila on és hi ha la peça més alta.

##### Paràmetres

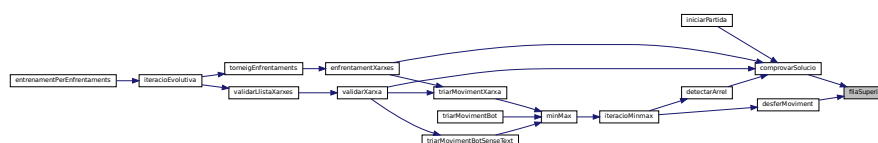
<i>partida</i>	és la partida de la qual es vol comprovar la fila superior.
<i>columna</i>	és la columna de la qual es vol comprovar la fila superior.

##### Retorna

l'índex de la fila amb la peça més alta de la columna indicada. Si la fila es buida retorna -1.

Definició a la línia 75 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.11 imprimirQuateEnRatlla()

```
void imprimirQuateEnRatlla (
    QuatreEnRatlla * partida )
```

Imprimeix un taulell del 4 en ratlla.

#### Paràmetres

<i>partida</i>	és un apuntador la partida de la que es vol imprimir l'estat
----------------	--

Definició a la línia 16 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.12 inicialitzarQuatreEnRatlla()

```
void inicialitzarQuatreEnRatlla (
    QuatreEnRatlla * partida,
    int nFils,
    int nCols,
    int nVictoria )
```

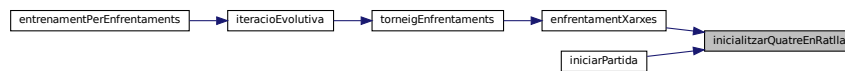
Inicia una partida del 4 en ratlla des de zero.

#### Paràmetres

<i>partida</i>	és un apuntador a la partida que es vol iniciar.
<i>nFils</i>	és el nombre de files de la taula.
<i>nCols</i>	és el nombre de columnes de la taula.
<i>nVictoria</i>	és el nombre de peces adjacents alineades que s'han de tenir per a guanyar la partida

Definició a la línia 39 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.13 realitzarMoviment()

```

void realitzarMoviment (
    QuatreEnRatlla * partida,
    int moviment,
    signed char jugador )
  
```

Introdueix una fitxa d'un jugador en una columna.

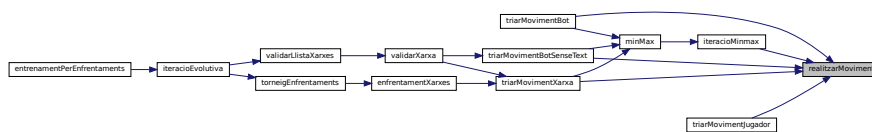
No té en compte si el moviment no es pot fer.

#### Paràmetres

<i>partida</i>	és un apuntador a la partida en la que es vol fer el moviment.
<i>moviment</i>	és el número de la columna en la que es vol introduir la fitxa.
<i>jugador</i>	és el jugador que realitza el moviment.

Definició a la línia 61 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



### 5.1.2.14 reiniciarQuatreEnRatlla()

```

void reiniciarQuatreEnRatlla (
    QuatreEnRatlla * partida )
  
```

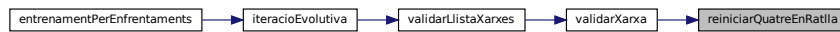
Fica tots els valors del tauler a 0.

### Paràmetres

`partida` és la partida que vols reiniciar.

Definició a la línia 33 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:

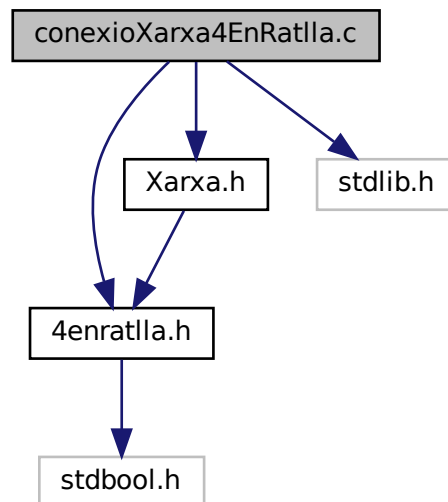


## 5.2 Referència del Fitxer conexioXarxa4EnRatlla.c

Implementa un wrapper per a poder fer servir una xarxa com a funció heurística.

```
#include "4enratlla.h"
#include "Xarxa.h"
#include "stdlib.h"
```

Inclou el graf de dependències per a conexioXarxa4EnRatlla.c:



### Funcions

- double [wrapperXarxa](#) ([QuatreEnRatlla](#) \*partida, void \*xarxaCtx)  
*Wrapper que serveix per a donar a una xarxa el tipus de funció heurística.*

## 5.2.1 Descripció Detallada

Implementa un wrapper per a poder fer servir una xarxa com a funció heurística.

## 5.2.2 Documentació de les Funcions

### 5.2.2.1 `wrapperXarxa()`

```
double wrapperXarxa (
    QuatreEnRatlla * partida,
    void * xarxaCtx )
```

Wrapper que serveix per a donar a una xarxa el tipus de funció heurística.

Aplica una xarxa neuronal al taulell d'una partida.

#### Paràmetres

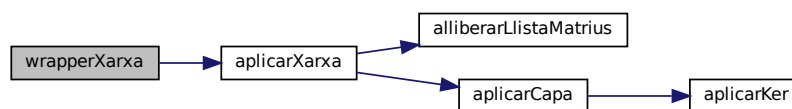
<i>partida</i>	és la partida a la que es vol aplicar la xarxa.
<i>xarxaCtx</i>	ha de ser un apuntador a una <code>xarxaNeuronal</code> , que es la que s'aplicarà a la partida.

#### Retorna

la puntuació a l'estat actual de la partida

Definició a la línia 12 del fitxer `conexioXarxa4EnRatlla.c`.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

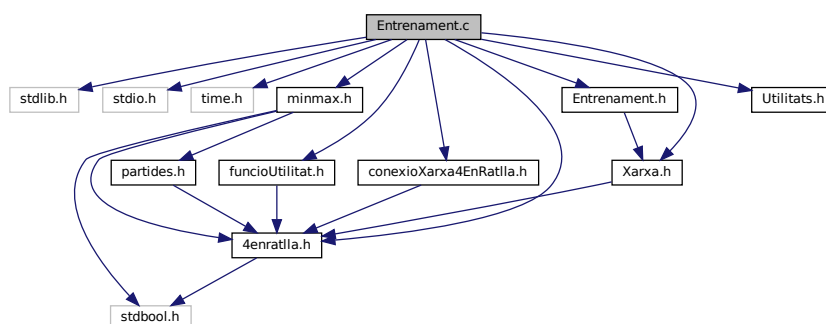


## 5.3 Referència del Fitxer Entrenament.c

Fitxer que implementa les funcions necessàries per a l'entrenament de la [CNN](#).

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "Xarxa.h"
#include "4enratlla.h"
#include "minmax.h"
#include "Utilitats.h"
#include "funcioUtilitat.h"
#include "Entrenament.h"
#include "conexioXarxa4EnRatlla.h"
```

Inclou el graf de dependències per a Entrenament.c:



## Funcions

- int [triarMovimentXarxa](#) ([QuatreEnRatlla](#) \*partida, char jugador, [XarxaNeuronal](#) \*xarxa)  
*Tria el moviment que fa una xarxa neuronal (aplican minmax) segons l'estat de la partida y el jugador al que li toqui tirar.*
- int [enfrentamentXarxes](#) ([XarxaNeuronal](#) \*J1, [XarxaNeuronal](#) \*J2)  
*Enfrenta dues xarxes neuronals i retorna el guanyador.*
- void [crearNovaGeneracio](#) ([Generacio](#) \*antigaGeneracio, int \*millorsIndividus)  
*Crea una nova generació de xarxes a partir dels millors individus, modificant aquests lleugerament.*
- void [torneigEnfrentaments](#) ([Generacio](#) \*generacioXarxesJ1, [Generacio](#) \*generacioXarxesJ2)  
*Enfrenta a la generació de xarxes per a veure quina guanya més.*
- void [validarXarxa](#) ([XarxaNeuronal](#) \*xarxa, int tornXarxa, int \*nTorns, int \*nVictories, [QuatreEnRatlla](#) \*partida)  
*Simula una partida de la xarxa contra la IA de validació indicant el nombre de moviments i qui ha guanyat.*
- void [validarLlistaXarxes](#) ([XarxaNeuronal](#) \*\*llistaXarxesJ1, [XarxaNeuronal](#) \*\*llistaXarxesJ2, int nXarxesJ1, int nXarxesJ2, [QuatreEnRatlla](#) \*partida)  
*Valida una llista de xarxes neuronals fent-les competir contra un algorisme predefinit i afegeix els resultats en un arxiu anomenat Record.csv L'arxiu Record.csv conté 4 columnes:*
- void [iteracioEvolutiva](#) ([Generacio](#) \*generacioXarxesJ1, [Generacio](#) \*generacioXarxesJ2, [QuatreEnRatlla](#) \*partida)  
*Realitza una iteració de l'algorisme evolutiu i guarda la xarxa.*
- void [entrenamentPerEnfrentaments](#) ([Generacio](#) \*generacioXarxesJ1, [Generacio](#) \*generacioXarxesJ2, [QuatreEnRatlla](#) \*partida, int iteracions)  
*Crea una generació de xarxes y les va evolucionant per a arribar a una bona funció heurística.*
- void [main](#) ()



### 5.3.1 Descripció Detallada

Fitxer que implementa les funcions necessàries per a l'entrenament de la [CNN](#).

### 5.3.2 Documentació de les Funcions

#### 5.3.2.1 crearNovaGeneracio()

```
void crearNovaGeneracio (
    Generacio * antigaGeneracio,
    int * millorsIndividus )
```

Crea una nova generació de xarxes a partir dels millors individus, modificant aquests lleugerament.

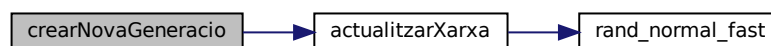
##### Paràmetres

<i>antigaGeneracio</i>	és la generació que es vol evolucionar.
<i>millorsIndividus</i>	és un array amb els index del millors individus de la generació.

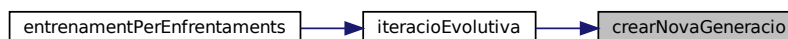
Per la implementació que s'ha dut a terme, pot ser que alguns pares tinguin menys o més fills que el que els hi pertoca. S'ha considerat que aquesta aleatorietat afegida no afecta a l'entrenament.

Definició a la línia 59 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.3.2.2 enfrontamentXarxes()

```
int enfrontamentXarxes (
    XarxaNeuronal * J1,
    XarxaNeuronal * J2 )
```

Enfronta dues xarxes neuronals i retorna el guanyador.

#### Paràmetres

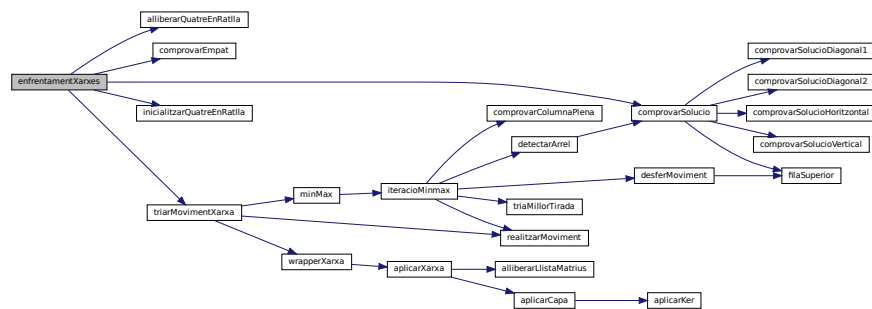
J1	és un apuntador a la xarxa neuronal que començarà tirant.
J2	és un apuntador a la xarxa neuronal que tirarà al segon torn.

#### Retorna

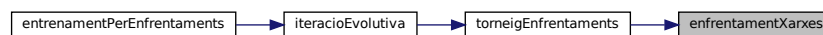
Retorna 0 si ha guanyat J1 i 1 si ha guanyat J2.

Definició a la línia 30 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.3.2.3 entrenamentPerEnfrontaments()

```
void entrenamentPerEnfrontaments (
    Generacio * generacioXarxesJ1,
    Generacio * generacioXarxesJ2,
    QuatreEnRatlla * partida,
    int iteracions )
```

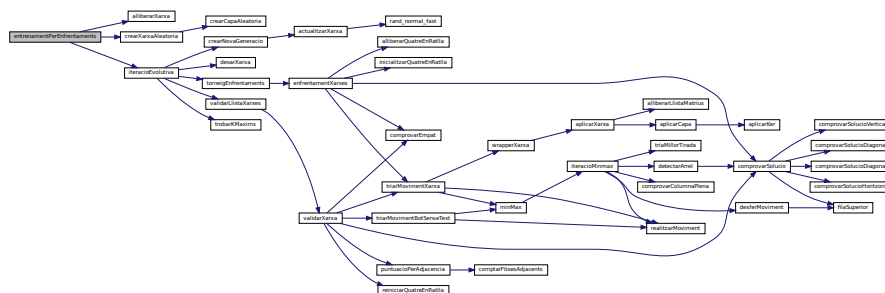
Crea una generació de xarxes y les va evolucionant per a arribar a una bona funció heurística.

## Paràmetres

<i>generacioXarxesJ1</i>	és una generació sense inicialitzar. Només ha de tenir creats els paràmtres ints, però no el *victories ni l'array de xarxes. Serà la que competirà iniciant la partida.
<i>generacioXarxesJ2</i>	és una generació sense inicialitzar. Només ha de tenir creats els paràmtres ints, però no el *victories ni l'array de xarxes. Serà la que competirà fent el segon moviemnt.
<i>partida</i>	és un apuntador a un QuatreEnRatlla inicialitzada i buida.
<i>iteracions</i>	és el nombre d'iteracions que fa l'entrenament.

Definició a la línia 214 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



## 5.3.2.4 iteracioEvolutiva()

```
void iteracioEvolutiva (
    Generacio * generacioXarxesJ1,
    Generacio * generacioXarxesJ2,
    QuatreEnRatlla * partida )
```

Realitza una iteració de l'algorisme evolutiu i guarda la xarxa.

Desa la xarxa neuronal del J1 a un arxiu anomenat "XarxaEntrenadaJ1.DrusCNN" i la J2 a un arxiu anomenat "XarxaEntrenadaJ2.DrusCNN".

## Paràmetres

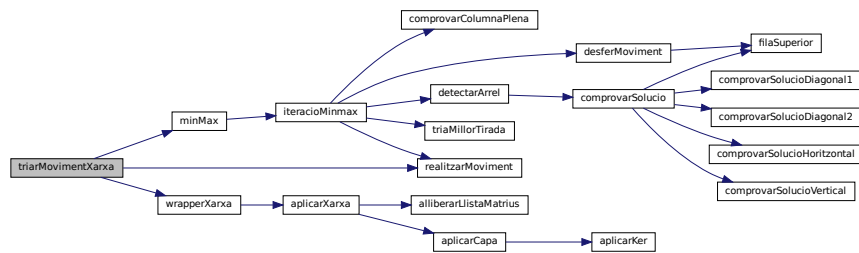
<i>generacioXarxesJ1</i>	és la generació de xarxes que es vol fer evolucionar per a millorar en el primer torn. En acabar la iteració ja retorna la generació evolucionada.
<i>generacioXarxesJ1</i>	és la generació de xarxes que es vol fer evolucionar per a millorar en el segon torn. En acabar la iteració ja retorna la generació evolucionada.
<i>partida</i>	és el taulell on competeixen les xarxes.

Definició a la línia 164 del fitxer Entrenament.c.

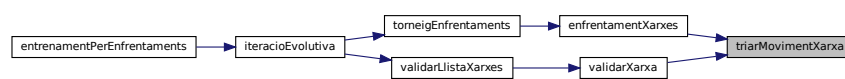




Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.3.2.7 validarLlistaXarxes()

```

void validarLlistaXarxes (
    XarxaNeuronal ** llistaXarxesJ1,
    XarxaNeuronal ** llistaXarxesJ2,
    int nXarxesJ1,
    int nXarxesJ2,
    QuatreEnRatlla * partida )
  
```

Valida una llista de xarxes neuronals fent-les competir contra un algorisme predefinit i afegeix els resultats en un arxiu anomenat Record.csv L'arxiu Record.csv conté 4 columnes:

- Columna 1: Nombre de moviments fets com a J1
- Columna 2: Nombre de moviments fets com a J2
- Columna 3: Nombre de victories com a J1
- Columna 4: Nombre de victories com a J2

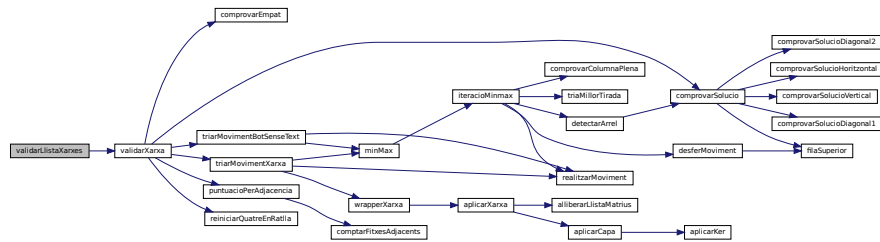
Cada fila és una nova validació.

#### Paràmetres

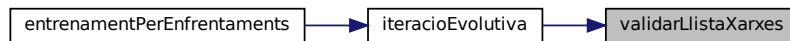
<i>llistaXarxesJ1</i>	és l'array de <b>CNN</b> que es volen avaluar en el primer torn de partida.
<i>llistaXarxesJ2</i>	és l'array de <b>CNN</b> que es volen avaluar en el segon torn de partida.
<i>nXarxesJ1</i>	és la longitud de l'array llista xarxes de primer torn.
<i>nXarxesJ2</i>	és la longitud de l'array llista xarxes de segon torn.
<i>partida</i>	és el taulell on es validaran les xarxes

Definició a la línia 136 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.3.2.8 validarXarxa()

```

void validarXarxa (
    XarxaNeuronal * xarxa,
    int tornXarxa,
    int * nTorns,
    int * nVictories,
    QuatreEnRatlla * partida )
  
```

Simula una partida de la xarxa contra la IA de validació indicant el nombre de moviments i qui ha guanyat.

#### Paràmetres

<i>xarxa</i>	és la xarxa que es vol validar.
<i>tornXarxa</i>	indica si la xarxa és el J1 o J2 (1 per a J1 i -1 per a J2)
<i>nTorns</i>	variable on es registraran els torns fets. En acabar la funció a aquest paràmetre se li ha afegit el nombre de moviments fets en total.
<i>nVictories</i>	variable on és registrarà si la xarxa ha guanyat. Se li sumarà 1 en cas de que guanyi i res en cas contrari.
<i>partida</i>	és un apuntador a la partida en la que es validaran les xarxes. Es reinicia el taulell de la partida.

Definició a la línia 104 del fitxer Entrenament.c.





## Funcions

- void [comptarFitxesAdjacents](#) ([QuatreEnRatlla](#) \*partida, int fila, int col, int \*fitxesJ1, int \*fitxesJ2)  
*Compta quantes fitxer de cada tipus hi ha al voltant d'una posició en concret.*
- double [puntuacioPerAdjacencia](#) ([QuatreEnRatlla](#) \*partida, void \*ctx)  
*Valora un estat de partida.*

### 5.4.1 Descripció Detallada

Fitxer que implementa una funció heurística que valora l'adjacència.

### 5.4.2 Documentació de les Funcions

#### 5.4.2.1 comptarFitxesAdjacents()

```
void comptarFitxesAdjacents (
    QuatreEnRatlla * partida,
    int fila,
    int col,
    int * fitxesJ1,
    int * fitxesJ2 )
```

Compta quantes fitxer de cada tipus hi ha al voltant d'una posició en concret.

#### Paràmetres

<i>partida</i>	és l'estat de partida actual
<i>fila</i>	fila on es la posició que es vol comprovar
<i>col</i>	columna on es la posició que es vol comprovar
<i>fitxesJ1</i>	direcció per on retorna el nombre de fitxes del J1
<i>fitxesJ2</i>	direcció per on retorna el nombre de fitxes del J2

Definició a la línia 11 del fitxer funcioUtilitat.c.

Gràfic de crides a aquesta funció:



### 5.4.2.2 puntuacioPerAdjacencia()

```
double puntuacioPerAdjacencia (
    QuatreEnRatlla * partida,
    void * ctx )
```

Valora un estat de partida.

#### Paràmetres

<i>partida</i>	és l'estat que es vol valorar
<i>ctx</i>	és un apuntador a void que només serveix per a que s'adapti a l'estructura de funcióHeurística per a fer el càlcul, per a cada casella es compten quantes n'hi ha de cada jugador i es resten les d'un amb les de l'altre elevades al quadrat

#### Retorna

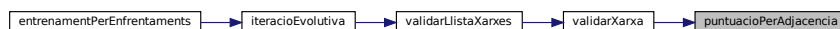
la valoració de la partida, on un nombre positiu vol dir que J1 va guanyant i negatiu que J2 va guanyant

Definició a la línia 25 del fitxer funcioUtilitat.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



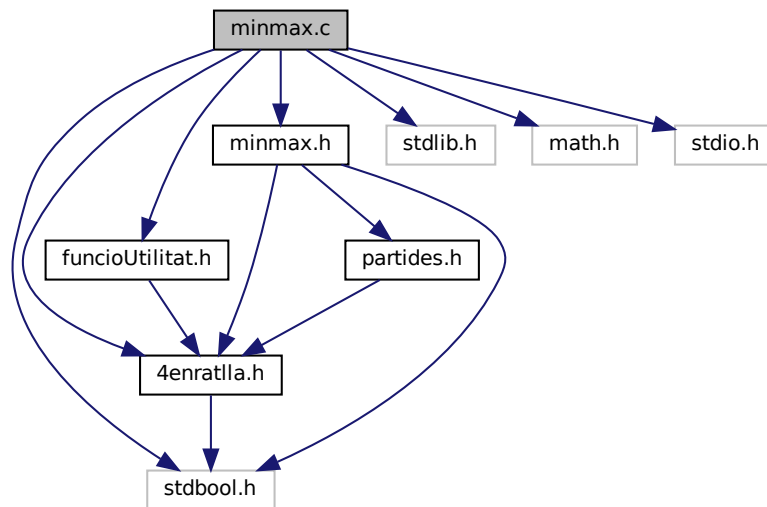
## 5.5 Referència del Fitxer minmax.c

Fitxer que implementa l'algorisme del minMax.

```
#include "4enratlla.h"
#include "funcioUtilitat.h"
#include "minmax.h"
#include <stdbool.h>
#include <stdlib.h>
#include <math.h>
```

```
#include <stdio.h>
```

Inclou el graf de dependències per a minmax.c:



## Funcions

- int **minMax** (**QuatreEnRatlla** \*partida, signed char jugadorOriginal, funcioHeuristica fHeuristica, void \*ctx↔ Heuristica)  
*Comença l'algorisme del MinMax.*
- int **iteracioMinmax** (**QuatreEnRatlla** \*partida, signed char jugadorOriginal, int nivellNode, double \*puntuacioNode, int \*profunditatNode, funcioHeuristica fHeuristica, void \*ctxHeuristica)  
*realitza l'algorisme del MinMax*
- bool **detectarArrel** (**QuatreEnRatlla** \*partida, int moviment, signed char jugadorOriginal, double \*valoracio, int profunditat, funcioHeuristica fHeuristica, void \*ctxHeuristica)  
*Gestiona les arrels de l'arbre Donat un node comprova si:*
- void **triaMillorTirada** (int \*millorMoviment, int movimentActual, double \*millorValoracio, double valoracio↔ Actual, int \*accioDesempat, int accioDesempatActual)  
*Compara el node actual amb el millor d'aquella bifuració.*

### 5.5.1 Descripció Detallada

Fitxer que implementa l'algorisme del minMax.

### 5.5.2 Documentació de les Funcions

### 5.5.2.1 detectarArrel()

```
bool detectarArrel (
    QuatreEnRatlla * partida,
    int moviment,
    signed char jugadorOriginal,
    double * valoracio,
    int profunditat,
    funcioHeuristica fHeuristica,
    void * ctxHeuristica )
```

Gestiona les arrels de l'arbre Donat un node comprova si:

- El moviment provoca una victòria (i aleshores assigna INF en cas de ser el jugador original i -INF en cas de ser l'altre)
- El moviment es de la profunditat màxima de l'arbre (i aleshores assigna la puntuació corresponent si el jugador és l'original és el primer i l'oposat si és l'altre)

En cas de detectar alguna d'aquestes coses s'assigna fulla=true al node i és retorna true En cas contrari retorna false.

No modifica l'estat actual de la partida.

#### Paràmetres

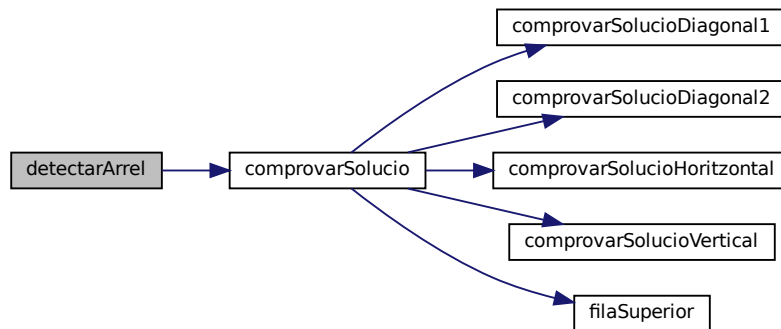
<i>partida</i>	és l'estat de la partida que es vol puntuar.
<i>moviment</i>	és el l'últim moviment fet per a arribar a aquest estat.
<i>jugadorOriginal</i>	és el jugador que està fent el moviment actual.
<i>valoracio</i>	és el un aptuntador per on es retornarà la valoració del node.
<i>profunditat</i>	és a la profunditat que ens trobem a l'arbre.
<i>fHeuristica</i>	és la funció heurística que s'usarà per a puntuar les fulles.
<i>ctxHeuristica</i>	son altres paràmetres que pot requerir la heurística.

#### Retorna

retorna cert en cas de que el node sigui una fulla i false en cas contrari.

Definició a la línia 67 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.5.2.2 iteracioMinimax()

```

int iteracioMinimax (
    QuatreEnRatlla * partida,
    signed char jugadorOriginal,
    int nivellNode,
    double * puntuacioNode,
    int * profunditatNode,
    funcioHeuristica fHeuristica,
    void * ctxHeuristica )
  
```

realitza l'algorisme del MinMax

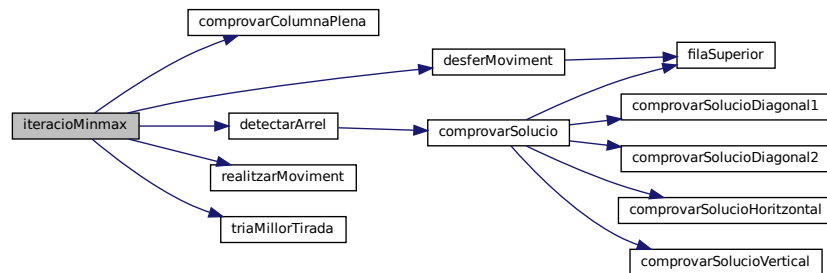
Per fer

#### Paràmetres

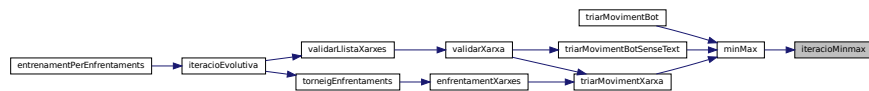
<i>partida</i>	és l'estat acutal de la partida (sense modificar el moviment actual).
<i>jugadorOriginal</i>	és el jugador al que li toca fer el moviment a l'inici del minmax.
<i>nivellNode</i>	és a quina profunditat de l'arbre ens trobem.
<i>puntuacioNode</i>	és un apuntador que conté la puntuació assignada al node.
<i>profunditatNode</i>	és un apuntador a la profunditat a la que es ha rebut el valor del minmax.
<i>fHeuristica</i>	és la funció heurística que s'usa per a valorar nodes.
<i>ctxHeuristica</i>	son altres paràmetres que pot necessitar la heurísitca.
Generat per Doxygen	el int del moviment que ha trobat l'algorisme que és el millor

Definició a la línia 27 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.5.2.3 minMax()

```

int minMax (
    QuatreEnRatlla * partida,
    signed char jugadorOriginal,
    funcioHeuristica fHeuristica,
    void * ctxHeuristica )
  
```

Comença l'algorisme del MinMax.

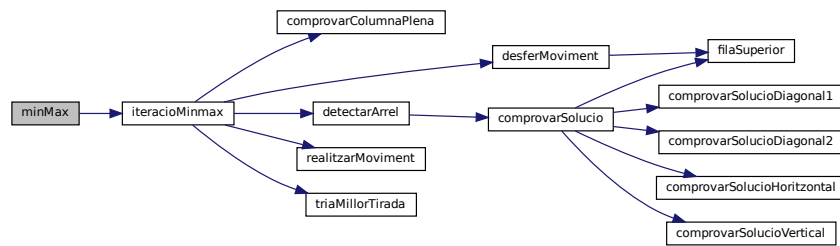
**Per fer** s'hauria de ficar la profunditat com a paràmetre

#### Paràmetres

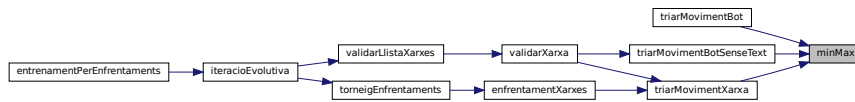
<i>partida</i>	l'estat de la partida en la que es fa el minmax.
<i>jugadorOriginal</i>	és el jugador al que li toca tirar.
<i>fHeuristica</i>	és la funció heurística que s'utilitza per a fer el minmax.
<i>ctxHeuristica</i>	és una estructura que conté dades necessàries per a la funció heurística.

Definició a la línia 18 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



#### 5.5.2.4 triaMillorTirada()

```

void triaMillorTirada (
    int * millorMoviment,
    int movimentActual,
    double * millorValoracio,
    double valoracioActual,
    int * accioDesempat,
    int accioDesempatActual )
  
```

Compara el node actual amb el millor d'aquella bifuració.

##### Paràmetres

<i>millorMoviment</i>	és el millor moviment fet en aquella bifuració. Si el moviment actual és millor, actualitza el valor
<i>movimentActual</i>	és el moviment fet per a arribar a aquell estat.
<i>millorValoracio</i>	és la valoració del millor moviment d'aquella biuració.
<i>valoracioActual</i>	és la valoració del node actual.
<i>accioDesempat</i>	és la profunditat relativa a la que s'ha arribat a la millor valoració de la bifuració.
<i>accioDesempatActual</i>	és la profunditat relativa a la que s'ha arribat a la fulla corresponent al node actual.

Definició a la línia 79 del fitxer minmax.c.

Gràfic de crides a aquesta funció:



## 5.6 Referència del Fitxer partides.c

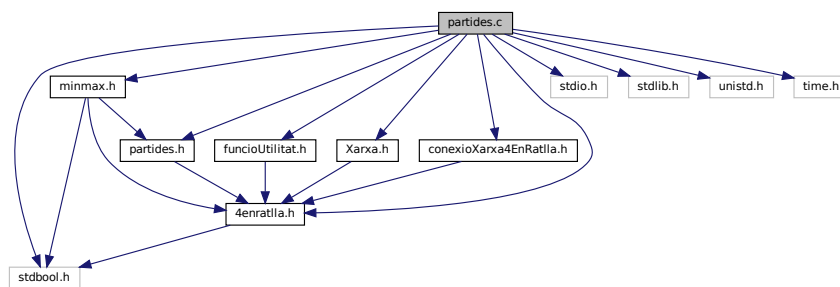
Fitxer que inclou totes les funcions relacionades amb la interfície de la partida i la tria de moviments.

```

#include "4enratlla.h"
#include "minmax.h"
#include "partides.h"
#include "funcioUtilitat.h"
#include "Xarxa.h"
#include "conexioXarxa4EnRatlla.h"
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

```

Inclou el graf de dependències per a partides.c:



## Definicions

- #define **PROBABILITAT\_ALEATORI** 0.01

## Funcions

- int **triarMovimentJugador** (**QuatreEnRatlla** \*partida, signed char jugador, void \*ctx)  
*Pregunta al jugador quin moviment vol fer.*
- int **triarMovimentBot** (**QuatreEnRatlla** \*partida, signed char jugador, void \*ctx)  
*Utilitza el min-max per a trobar la millor jugada.*
- int **triarMovimentBotAleatori** (**QuatreEnRatlla** \*partida, signed char jugador, void \*ctx)  
*Utilitza el min-max per a trobar la millor jugada amb una probabilitat de fer un moviment aleatori.*
- int **triarMovimentBotSenseText** (**QuatreEnRatlla** \*partida, signed char jugador, void \*ctx)  
*Utilitza el min-max per a trobar la millor jugada sense imprimir les decisions per pantalla.*
- void **iniciarPartida** (selectorDeMoviment decisióJ1, selectorDeMoviment decisióJ2, double esperaEntreTorns, void \*ctx)
- void **validacioXarxa** ()
- void **validacioNormal** ()



### 5.6.1 Descripció Detallada

Fitxer que inclou totes les funcions relacionades amb la interfície de la partida i la tria de moviments.

### 5.6.2 Documentació de les Funcions

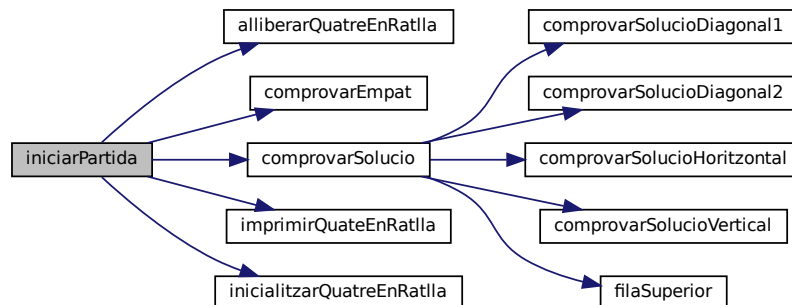
#### 5.6.2.1 iniciarPartida()

```
void iniciarPartida (
    selectorDeMoviment decisioJ1,
    selectorDeMoviment decisioJ2,
    double esperaEntreTorns,
    void * ctx )
```

##### Per fer

Definició a la línia 104 del fitxer partides.c.

Gràfic de crides d'aquesta funció:



#### 5.6.2.2 triarMovimentBot()

```
int triarMovimentBot (
    QuatreEnRatlla * partida,
    signed char jugador,
    void * ctx )
```

Utilitza el min-max per a trobar la millor jugada.

La funció per decidir el moviment s'ha de passar com a context.

## Paràmetres

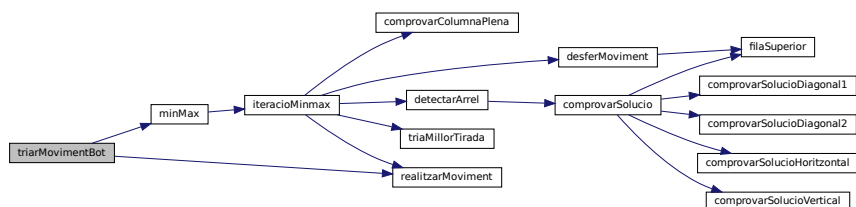
<i>partida</i>	és un apuntador a la partida que s'està jugant.
<i>jugador</i>	és el jugador que fa el moviment.
<i>ctx</i>	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

## Retorna

el número de la columna que l'algorisme ha decidit. Sempre retorna un moviment vàlid

Definició a la línia 45 del fitxer partides.c.

Gràfic de crides d'aquesta funció:



## 5.6.2.3 triarMovimentBotAleatori()

```

int triarMovimentBotAleatori (
    QuatreEnRatlla * partida,
    signed char jugador,
    void * ctx )
  
```

Utilitza el min-max per a trobar la millor jugada amb una probabilitat de fer un moviment aleatori.

La funció per decidir el moviment s'ha de pasar com a context.

## Paràmetres

<i>partida</i>	és un apuntador a la partida que s'està jugant.
<i>jugador</i>	és el jugador que fa el moviment.
<i>ctx</i>	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

## Retorna

el número de la columna que l'algorisme ha decidit. Sempre retorna un moviment vàlid

Definició a la línia 60 del fitxer partides.c.

### 5.6.2.4 triarMovimentBotSenseText()

```
int triarMovimentBotSenseText (
    QuatreEnRatlla * partida,
    signed char jugador,
    void * ctx )
```

Utilitza el min-max per a trobar la millor jugada sense imprimir les decisions per pantalla.

La funció per decidir el moviment s'ha de passar com a context.

#### Paràmetres

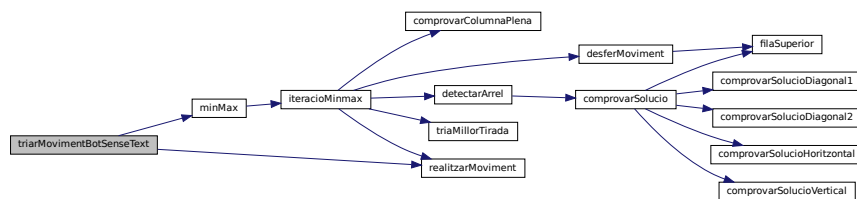
<i>partida</i>	és un apuntador a la partida que s'està jugant.
<i>jugador</i>	és el jugador que fa el moviment.
<i>ctx</i>	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

#### Retorna

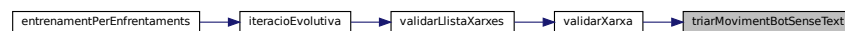
el número de la columna que l'algorisme ha decidit. Sempre retorna un moviment vàlid

Definició a la línia 87 del fitxer partides.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.6.2.5 triarMovimentJugador()

```
int triarMovimentJugador (
    QuatreEnRatlla * partida,
    signed char jugador,
    void * ctx )
```

Pregunta al jugador quin moviment vol fer.

No para de preguntar fins a rebre una jugada vàlida. En cas de no rebre una jugada vàlida imprimeix per consola si es per un error ortogràfic o per una columna de fora del taulell.

**Paràmetres**

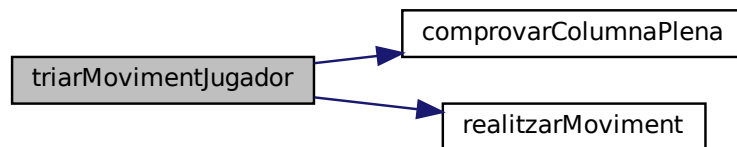
<i>partida</i>	és un apuntador a la partida que s'està jugant
<i>jugador</i>	és el jugador que fa el moviment.
<i>ctx</i>	és un apuntador que no ha de contenir res, només li dona el tipus de selectorDeMoviment.

**Retorna**

el número de la columna que el jugador ha dit que vol tirar. Sempre retorna un moviment vàlid

Definició a la línia 24 del fitxer partides.c.

Gràfic de crides d'aquesta funció:

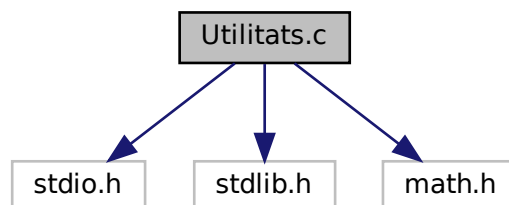


## 5.7 Referència del Fitxer Utilitats.c

Fitxer que inclou funcions amb utilitats diverses sense dependències.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Inclou el graf de dependències per a Utilitats.c:

**Definicions**

- `#define PARAM_LReLU .1`

## Funcions

- void [alliberarLlistaMatrius](#) (double \*\*\*llistaMatrius, int nMatrius, int dimFil)  
*Donat un array de matrius amb el mateix nombre de files, les allibera totes.*
- void [imprimirMatriu](#) (double \*\*matriu, int dimFil, int dimCol)  
*Imprimeix una matriu.*
- double \*\* [deepCopyMatriu](#) (double \*\*matriu, int dimFil, int dimCol)  
*Fa una còpia profunda d'una matriu de doubles.*
- int \* [trobarKMaxims](#) (int \*llista, int midaLlista, int k)  
*Trova els k índexos amb valors més grans d'una llista.*
- double [rand\\_normal\\_fast](#) ()  
*Genera un nombre aleatori amb una distribució aproximada de la normal estandar.*
- double [leakyReLU](#) (double x)  
*Rectificador Lineal Unitari Leaky.*

### 5.7.1 Descripció Detallada

Fitxer que inclou funcions amb utilitats diverses sense dependències.

### 5.7.2 Documentació de les Funcions

#### 5.7.2.1 alliberarLlistaMatrius()

```
void alliberarLlistaMatrius (
    double *** llistaMatrius,
    int nMatrius,
    int dimFil )
```

Donat un array de matrius amb el mateix nombre de files, les allibera totes.

#### Paràmetres

<i>llistaMatrius</i>	és la llista de matrius que vols alliberar
<i>nMatrius</i>	és el nombre de matrius que vols alliberar
<i>dimFil</i>	és el nombre de files que tenen les matrius

Definició a la línia 12 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



### 5.7.2.2 deepCopyMatriu()

```
double** deepCopyMatriu (
    double ** matriu,
    int dimFil,
    int dimCol )
```

Fa una còpia profunda d'una matriu de doubles.

#### Paràmetres

<i>matriu</i>	és la matriu que vols copiar.
<i>dimFil</i>	és el nombre de files que té la matriu.
<i>dimCol</i>	és el nombre de columnes que té la matriu.

#### Retorna

Retorna un apuntador a la nova matriu.

Definició a la línia 32 del fitxer Utilitats.c.

### 5.7.2.3 imprimirMatriu()

```
void imprimirMatriu (
    double ** matriu,
    int dimFil,
    int dimCol )
```

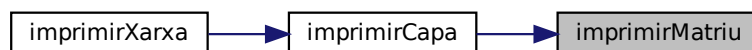
Imprimeix una matriu.

#### Paràmetres

<i>matriu</i>	és la matriu que vols imprimir
<i>dimFil</i>	és el nombre de files que té la matriu
<i>dimCol</i>	és el nombre de columnes que té la matriu

Definició a la línia 23 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



#### 5.7.2.4 leakyReLU()

```
double leakyReLU (
    double x )
```

Rectificador Lineal Unitari Leaky.

##### Paràmetres

x	és la variable de la funció.
---	------------------------------

##### Retorna

x si  $x \geq 0$  y  $a \cdot x$  si  $x < 0$ .

Definició a la línia 85 del fitxer Utilitats.c.

#### 5.7.2.5 rand\_normal\_fast()

```
double rand_normal_fast ( )
```

Genera un nombre aleatori amb una distribució aproximada de la normal estandar.

##### Atenció

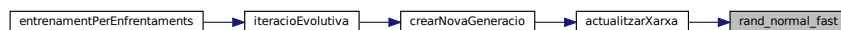
S'ha de vigilar perquè aquesta funció no es pot fer servir en paral·lel degut a que fa servir variables static.

##### Retorna

el nombre aleatori

Definició a la línia 59 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



#### 5.7.2.6 trobarKMaxims()

```
int* trobarKMaxims (
    int * llista,
    int midaLlista,
    int k )
```

Trova els k índexos amb valors més grans d'una llista.

**Paràmetres**

<i>llista</i>	és la llista de la que es volen trobar els màxims.
<i>midaLlista</i>	és la longitud de la llista.
<i>k</i>	és el nombre de valors més grans que vols trobar.

**Retorna**

una llista de mida k amb els índexos dels valors més grans.

Definició a la línia 43 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:

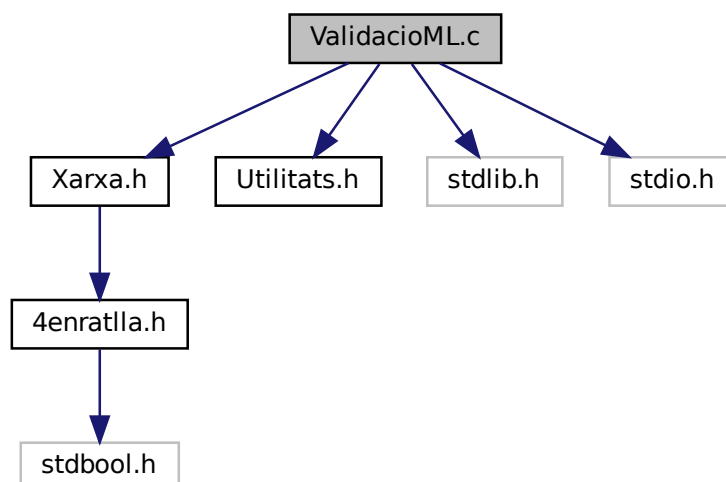


## 5.8 Referència del Fitxer ValidacioML.c

Fitxer per a validar les xarxes.

```
#include "Xarxa.h"  
#include "Utilitats.h"  
#include <stdlib.h>  
#include <stdio.h>
```

Inclou el graf de dependències per a ValidacioML.c:





## Funcions

- void **exempleConvolucio** ()
- void **exempleCapa** ()
- void **exempleCapaAleatoria** ()
- void **exempleXarxaAleatoria** ()
- void **exempleDesarXarxa** ()
- void **exempleAplicarXarxa** ()
- int **main** ()

### 5.8.1 Descripció Detallada

Fitxer per a validar les xarxes.

#### Atenció

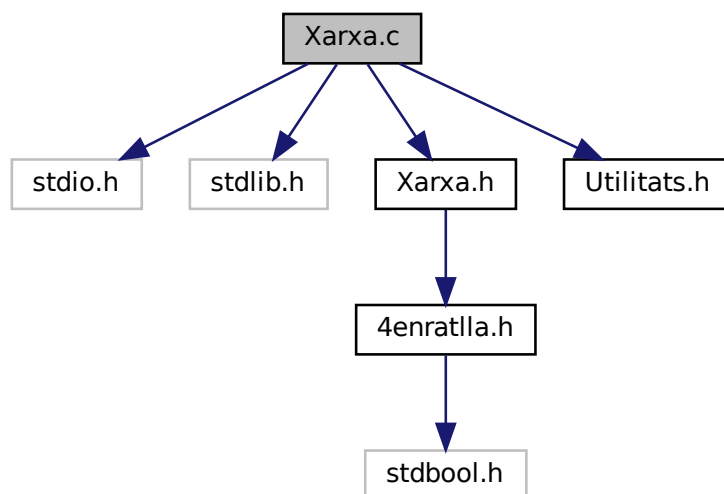
Segurament està anticuat i algunes funcions d'han d'actualitzar.

## 5.9 Referència del Fitxer Xarxa.c

Fitxer que inclou totes les funcions relacionades les operacions de les [CNN](#).

```
#include <stdio.h>
#include <stdlib.h>
#include "Xarxa.h"
#include "Utilitats.h"
```

Inclou el graf de dependències per a Xarxa.c:



## Funcions

- void `imprimirCapa` (`CapaXarxa` \*capa)  
*Imprimeix els kernels d'una capa.*
- void `imprimirXarxa` (`XarxaNeuronal` \*xarxa)  
*Imprimeix els kernels de les capes d'una xarxa.*
- double \*\* `aplicarKer` (double \*\*\*llistaMatrius, int dimFilMat, int dimColMat, int profunditat, double \*\*\*kernel, int dimKer, double biaix, funcioReal activacio)  
*Donades una llista de matrius de longitud igual a la profunditat del kernels d'una llista, aplica la convolució dels kernels.*
- double \*\*\* `aplicarCapa` (double \*\*\*llistaMatrius, `CapaXarxa` \*capa)  
*Donades una llista de matrius els hi aplica una capa d'una CNN.*
- double `aplicarXarxa` (double \*\*matriu, `XarxaNeuronal` \*xarxa)  
*Donada una matriu li aplica una xarxa neuronal.*
- `CapaXarxa` \* `crearCapaAleatoria` (int dimKer, int nKer, int profunditatKer, int dimFilln, int dimColln)  
*Crea una capa amb núclis amb valors aleatoris.*
- `XarxaNeuronal` \* `crearXarxaAleatoria` (int nCapes, int \*llistaDimKernels, int \*llistaNKernels, int dimFilln, int dimColln)  
*Crea una xarxa amb valors aleatoris.*
- void `alliberarXarxa` (`XarxaNeuronal` \*xarxa)  
*allibera una Xarxa*
- void `actualitzarXarxa` (`XarxaNeuronal` \*xarxaOriginal, `XarxaNeuronal` \*xarxaFilla, double sigma)  
*Fa una copia d'una xarxa neuronal variant els valors lleugerament seguint una distribució normal.*
- void `desarXarxa` (`XarxaNeuronal` \*xarxa, const char \*filename)  
*Desa una xarxa neuronal a un arxiu.*
- `XarxaNeuronal` \* `carregarXarxa` (const char \*filename)  
*Carrega una xarxa neuronal des d'un arxiu.*

### 5.9.1 Descripció Detallada

Fitxer que inclou totes les funcions relacionades les operacions de les CNN.

### 5.9.2 Documentació de les Funcions

#### 5.9.2.1 `actualitzarXarxa()`

```
void actualitzarXarxa (
    XarxaNeuronal * xarxaOriginal,
    XarxaNeuronal * xarxaFilla,
    double sigma )
```

Fa una copia d'una xarxa neuronal variant els valors lleugerament seguint una distribució normal.

Les dues xarxes han de tenir les mateixes dimensions.

La distribució normal s'aproxima amb l'algorisme de Marsaglia polar.

## Paràmetres

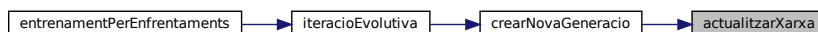
<i>xarxaOriginal</i>	és de la que és copien els valors
<i>xarxaFilla</i>	és on s'aplicaran els valors modificats. Ja ha de tenir la memòria reservada abans de passar-la.
<i>sigma</i>	és la desviació de la distribució normal que s'utilitza per a variar els valors

Definició a la línia 185 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.9.2.2 alliberarXarxa()

```
void alliberarXarxa (  
    XarxaNeuronal * xarxa )
```

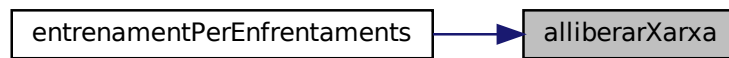
allibera una Xarxa

## Paràmetres

<i>xarxa</i>	és la xarxa que es vol alliberar.
--------------	-----------------------------------

Definició a la línia 164 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



### 5.9.2.3 aplicarCapa()

```
double*** aplicarCapa (
    double *** llistaMatrius,
    CapaXarxa * capa )
```

Donades una llista de matrius els hi aplica una capa d'una [CNN](#).

#### Paràmetres

<i>llistaMatrius</i>	és l'array de matrius al que s'els hi vol aplicar la capa. Totes han de tenir les mateixes dimensions. L'array ha de tenir la mateixa longitud que la profunditat dels kernels.
<i>capa</i>	és la capa que es vol aplicar.

#### Retorna

Retorna l'array de matrius resultant de l'aplicació dels kernels de la capa a la llista de matrius.

Definició a la línia 68 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



### 5.9.2.4 aplicarKer()

```
double** aplicarKer (
    double *** llistaMatrirus,
    int dimFilMat,
    int dimColMat,
    int profunditat,
    double *** kernel,
    int dimKer,
    double biaix,
    funcioReal activacio )
```

Donades una llista de matrius de longitud igual a la profunditat del kernels d'una llista, aplica la convolució dels kernels.

#### Paràmetres

<i>llistaMatrirus</i>	és l'array de matrius al que s'els hi vol aplicar la convolució dels kernels. Totes han de tenir les mateixes dimensions. L'array ha de tenir la mateixa longitud que la profunditat del kernel.
<i>dimFilMat</i>	nombre de files que tenen les matrius.
<i>dimColMat</i>	nombre de columnes que tenen les matrius
<i>profunditat</i>	profunditat que té el kernel (que coincideix amb el nombre de matrius de l'array llistaMatrirus).
<i>kernel</i>	és el kernel que es vol aplicar.
<i>dimKer</i>	és el nombre de files/columnes que té el kernel.
<i>biaix</i>	és el biaix que s'aplica a la convolució.
<i>activacio</i>	és la funció d'activació que s'aplica un cop afegit el viaix.

#### Retorna

Retorna la matriu resultant de l'aplicació del kernel a la llista de matrius.

Definició a la línia 36 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



### 5.9.2.5 aplicarXarxa()

```
double aplicarXarxa (
    double ** matriu,
    XarxaNeuronal * xarxa )
```

Donada una matriu li aplica una xarxa neuronal.

El valor retornat sempre és un únic double perquè al final de tot la xarxa suma tots els valors.

**Paràmetres**

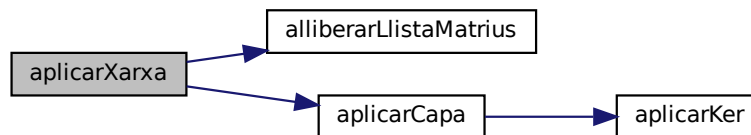
<i>matriu</i>	és la matriu inicial.
<i>xarxa</i>	és la xarxa que es vol aplicar a la matriu.

**Retorna**

Retorna el valor resultant d'aplicar la xarxa a la matriu.

Definició a la línia 83 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

**5.9.2.6 carregarXarxa()**

```
XarxaNeuronal* carregarXarxa (
    const char * filename )
```

Carrega una xarxa neuronal des d'un arxiu.

Fet per la IA.

**Paràmetres**

<i>filename</i>	és el nom del fitxer des del que es vol carregar la xarxa.
-----------------	--

**Retorna**

un apuntador a la xarxa carregada.

Definició a la línia 238 del fitxer Xarxa.c.

### 5.9.2.7 crearCapaAleatoria()

```
CapaXarxa* crearCapaAleatoria (
    int dimKer,
    int nKer,
    int profunditatKer,
    int dimFillIn,
    int dimColIn )
```

Crea una capa amb núclis amb valors aleatoris.

#### Paràmetres

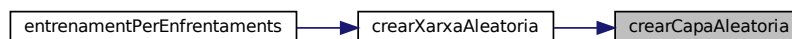
<i>dimKer</i>	és el nombre de files/columnes que tindran els kernels.
<i>nKer</i>	és el nombre de kernels que tindrà la capa.
<i>profunditatKer</i>	és la profunditat que tindran els kernels.
<i>dimFillIn</i>	dimensió de les matrius d'entrada.
<i>dimColIn</i>	dimensió de les columnes d'entrada.

#### Retorna

Retorna l'apuntador a la capa creada.

Definició a la línia 115 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



### 5.9.2.8 crearXarxaAleatoria()

```
XarxaNeuronal* crearXarxaAleatoria (
    int nCapes,
    int * llistaDimKernels,
    int * llistaNKernels,
    int dimFillIn,
    int dimColIn )
```

Crea una xarxa amb valors aleatoris.

Està ajustada per a que totes les dimensions quadrin.

Això vol dir que cada capa està preparada per a rebre tantes matrius com kernels té la capa anterior. I aquestes son de dimensió igual que l'anterior però reduïdes en un nombre igual que la meitat de la mida dels kernels més 1.

**Paràmetres**

<i>nCapes</i>	és el nombre de capes de la xarxa
<i>llistaDimKernels</i>	és un array on l'element <i>i</i> és la dimensió que tenen els kernels de la capa <i>i</i> .
<i>llistaNKernels</i>	és un array on l'element <i>i</i> és el nombre de kernels de la capa <i>i</i> .
<i>dimFilln</i>	és el nombre de files de la matriu que reb com a entrada la xarxa.
<i>dimColln</i>	és el nombre de columnes de la matriu que reb com a entrada la xarxa.

**Retorna**

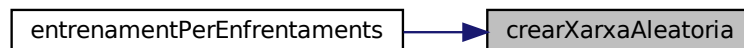
Retorna l'apuntador a la xarxa creada.

Definició a la línia 141 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

**5.9.2.9 desarXarxa()**

```

void desarXarxa (
    XarxaNeuronal * xarxa,
    const char * filename )
  
```

Desa una xarxa neuronal a un arxiu.

Fet per la IA.



## Paràmetres

<i>xarxa</i>	és la xarxa que es vol desar.
<i>filename</i>	és el nom del fitxer en el que es vol desar la xarxa.

Definició a la línia 205 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



#### 5.9.2.10 imprimirCapa()

```
void imprimirCapa (  
    CapaXarxa * capa )
```

Imprimeix els kernels d'una capa.

## Paràmetres

<i>capa</i>	és la capa que es vol imprimir
-------------	--------------------------------

Definició a la línia 15 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



#### 5.9.2.11 imprimirXarxa()

```
void imprimirXarxa (  
    XarxaNeuronal * xarxa )
```

Imprimeix els kernels de les capes d'una xarxa.

##### Paràmetres

<i>xarxa</i>	és la xarxa que es vol imprimir
--------------	---------------------------------

Definició a la línia 28 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

