4EnRatlla CNNGenètic

Generat per Doxygen 1.9.1

Chapter 1

4 en Ratlla amb IA Evolutiva (CNN + MinMax)

1.1 Descripcio

Implementacio completa del joc Quatre en Ratlla en llenguatge C. El projecte inclou una Xarxa Neuronal Convolucional (CNN) implementada des de zero (sense llibreries externes de ML) que apren a jugar mitjancant un algorisme genetic i evolucio competitiva.

El bot utilitza l'algorisme MinMax prenent com a funció heurística la xarxa neuronal per decidir el millor moviment.

1.2 Caracteristiques Principals

- Motor de Joc: Implementacio del 4 en Ratlla amb deteccio de victories.
- Intel.ligencia Artificial:
  - Algorisme MinMax per a la presa de decisions.
  - CNN propia amb capes convolucionals i activacio Leaky ReLU.
- Entrenament Evolutiu:
  - Sistema de generacio de poblacions i torneigs (Lliga tots contra tots).
  - Mutacio genetica (soroll gaussia) sobre els pesos de la xarxa.
  - Paral.lelisme amb OpenMP per accelerar l'entrenament.
- Documentacio: Codi font completament documentat amb Doxygen.

1.3 Requisits

- Compilador C: GCC o Clang.
- OpenMP: Necessari per a la paral.lelitzacio de l'entrenament.
- Llibreries Estandard: math.h, stdlib.h, stdio.h, omp.h.

1.4 Compilacio

1.4.0.1 Entrenament

Abans de compilar assegura't d'haver variat els paràmetres d'entrenament del main:

- El nombre de supervivents de cada iteració de cada xarxa.
- El nombre de fills que té cada supervivent.
- El learningRate de cada xarxa.
- Les dimensions del taulell i la longitud de k-en ratlla. A més és recomanable reduïr el PROFUNDITAT de l'arxiu minmax.h a 3.

Per compilar l'entrenament, utilitza la seguent comanda:

```
gcc -fopenmp -funroll-loops -flto -march=native Entrenament.c 4enratlla.c  minmax.c Xarxa.c Utilitats.c
      funcioUtilitat.c conexioXarxa4EnRatlla.c -O3 -o entrenament -lm
```

1.4.0.2 Partida

Abans de compilar assegura't d'haver variat els paràmetres de la funció validacioXarxa:

- El nom de la xarxa que vols carregar (en cas de voler fer servir una xarxa).
- Quina funció heurística fa servir cada jugador (en cas de voler fer servir heurística). Aquestes son wrapper↩ Xarxa (si has carregat una xarxa neuronal) i puntuacioPerAjacencia si vols ver servir la bàsica.
- la funció de decisió que fa servir el Jugador 1/2. Les disponibles son triarMovimentJugador (si vols que el jugador trii), triarMovimentBot (si vols que trii la IA especificada anteriorment) i triarMovimentBotAleatori (si vols que sigui IA, però amb possibilitat de tirar aleatoriament).

A més es pot variar la profunditat de l'algorisme minmax amb la constant PROFUNDITAT de l'arxiu minmax.c (recomanat 7) y les dimensions/longitud de la ratlla del taulell a la funció inicialitzar partida (compte amb que si fas servir una xarxa neuronal aquesta sigui compatible amb les dimensions del taullel).

Per a compilar la partida, utilitza la seguent comanda:

```
gcc -g -fopenmp -funroll-loops -flto -march=native Entrenament.c 4enratlla.c  minmax.c Xarxa.c Utilitats.c
      partides.c funcioUtilitat.c conexioXarxa4EnRatlla.c -O3 -o partida -lm
```

1.5 Com s'utilitza

1.5.1 1. Entrenament de la Xarxa

Per iniciar el proces evolutiu, executa el programa compilat:

./entrenament

El programa:

1. Generara una poblacio inicial aleatoria.
2. Realitzara tornejos entre les xarxes.
3. Seleccionara les millors i creara noves generacions amb mutacions.
4. Desara els millors models (.DrusCNN) automaticament. (A cada iteració sobreescriu la anterior i desa una de J1 i una de J2).

1.5.2 2. Validacio i Joc

Per jugar contra la IA executa ./entrenament. Assegura't de que en compilar has introduït les opcions correctes.

1.6 Detalls Tecnics de la IA

1.6.1 Estructura de la CNN

- Entrada: Matriu del tauler (files x columnes).
- Capes: Sistema flexible definit per l'usuari (actualment configurat amb capes convolucionals de kernels 3x3 i 5x5).
- Funcio d'Activacio: Leaky ReLU.
- Sortida: Un unic valor (double) que representa l'avaluacio heuristica del tauler.

1.6.2 Algorisme Genetic

- Metode de Seleccio: Torneig per parelles o lliga, jugen xarxes J1 vs J2.
- Reproduccio: Els guanyadors es clonen i els perdedors es substitueixen per versions mutades dels guanyadors.
- Mutacio: S'aplica soroll gaussia als biaixos i pesos dels kernels.

1.7 Documentacio

Aquest projecte inclou documentacio generada automaticament. Per consultar-la obrint la carpeta html o el pdf.

1.8 Autor

Drus Sentís Cahué.

Desenvolupat com a part d'un projecte d'implementacio d'IA en C pur.

1.9 Llicencia

Codi obert.

Chapter 2

Llista de coses per fer

Global comprovarSolucioDiagonal1 (QuatreEnRatlla ∗partida, int fila, int col)

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

Global ContextHeuristica

potser es podría 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)

Fitxer minmax.h

s'hauria de ficar la prodfunditat com a paràmetre.

Fitxer Xarxa.c

Crec que milloraría bastant la velocitat si s'implementes paralelització amb la gràfica. Per a fer això s'auria de canviar l'struct capaXarxa per a que uses double ∗kernel enlloc de ∗∗∗∗kerner per a així usar memòria contigua.

Chapter 3

Índex d'Estructures de Dades

3.1 Estructures de Dades

Aquestes són les estructures de dades acompanyades amb breus descripcions:

Chapter 4

Índex de Fitxers

4.1 Llista dels Fitxers

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

Chapter 5

Documentació de les Estructures de Dades

5.1 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

5.1.1 Descripció Detallada

Estructura que conté una capa de la xarxa.

Definició a la línia 24 del fitxer Xarxa.h.

5.1.2 Documentació dels Camps

5.1.2.1 biaixos

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

Biaixos de cada kernel

Definició a la línia 27 del fitxer Xarxa.h.

5.1.2.2 dimCol

```
int capaXarxa::dimCol
```

Nombre de columnes que tenen les matrius a processar

Definició a la línia 32 del fitxer Xarxa.h.

5.1.2.3 dimFil

```
int capaXarxa::dimFil
```

Nombre de files que tenen les matrius a processar

Definició a la línia 31 del fitxer Xarxa.h.

5.1.2.4 dimKer

```
int capaXarxa::dimKer
```

Dimensió dels kernels

Definició a la línia 29 del fitxer Xarxa.h.

5.1.2.5 funcioActivacio

```
funcioReal capaXarxa::funcioActivacio
```

La funció d'activació que fa servir la capa

Definició a la línia 28 del fitxer Xarxa.h.

5.1.2.6 kernels

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

Nuclis de la capa

Definició a la línia 26 del fitxer Xarxa.h.

5.1.2.7 nMatrius

```
int capaXarxa::nMatrius
```

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

Definició a la línia 30 del fitxer Xarxa.h.

5.1.2.8 nombreKernels

```
int capaXarxa::nombreKernels
```

Nombre de kernels que té la capa

Definició a la línia 25 del fitxer Xarxa.h.

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

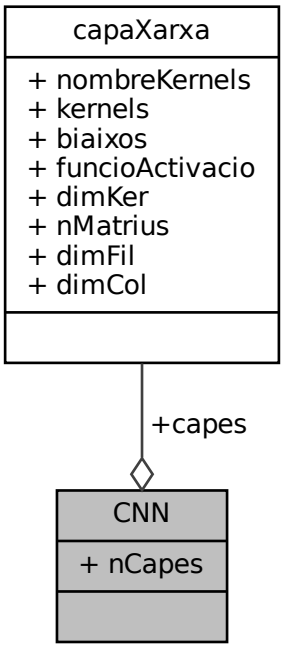
- Xarxa.h

5.2 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

5.2.1 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 42 del fitxer Xarxa.h.

5.2.2 Documentació dels Camps

5.2.2.1 capes

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

Array de capes

Definició a la línia 44 del fitxer Xarxa.h.

5.2.2.2 nCapes

```
int CNN::nCapes
```

Nombre de capes de la xarxa

Definició a la línia 43 del fitxer Xarxa.h.

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

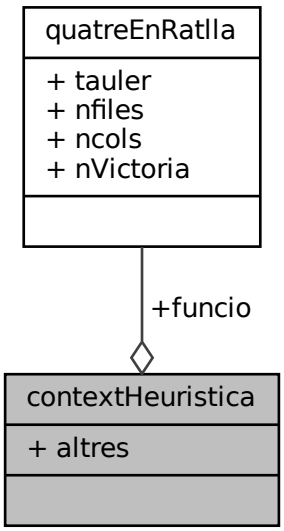
- Xarxa.h

5.3 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]

5.3.1 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 podría implementar la profunditat variable per aquí.

Definició a la línia 32 del fitxer partides.h.

5.3.2 Documentació dels Camps

5.3.2.1 altres

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

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

Definició a la línia 34 del fitxer partides.h.

5.3.2.2 funcio

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

Un apuntador a las funcions heurístiques del J1 y del J2

Definició a la línia 33 del fitxer partides.h.

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

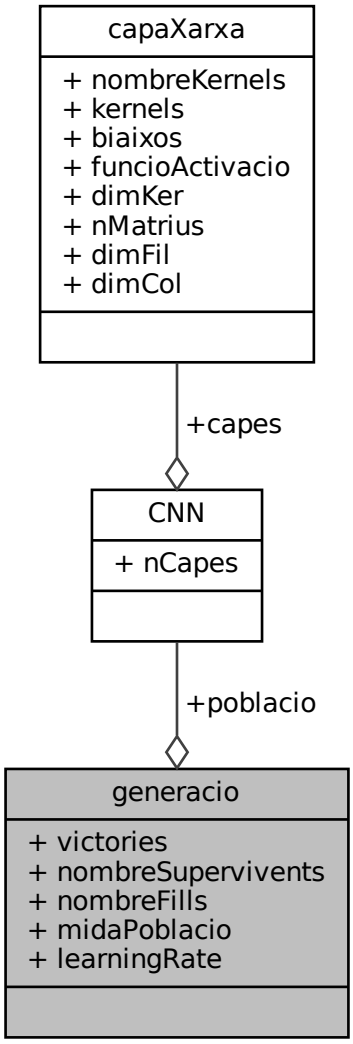
- partides.h

5.4 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

5.4.1 Descripció Detallada

Estructura que controla com es formen les generacions.

Definició a la línia 14 del fitxer Entrenament.h.

5.4.2 Documentació dels Camps

5.4.2.1 learningRate

```
double generacio::learningRate
```

El ratio d'aprenentatge

Definició a la línia 20 del fitxer Entrenament.h.

5.4.2.2 midaPoblacio

```
int generacio::midaPoblacio
```

La mida total de la població (ha de ser nombreSupervivents∗midaPoblacio)

Definició a la línia 19 del fitxer Entrenament.h.

5.4.2.3 nombreFills

```
int generacio::nombreFills
```

El nombre de fills que té cada supervivent

Definició a la línia 18 del fitxer Entrenament.h.

5.4.2.4 nombreSupervivents

```
int generacio::nombreSupervivents
```

El nombre de supervivents d'una iteració

Definició a la línia 17 del fitxer Entrenament.h.

5.4.2.5 poblacio

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

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

Definició a la línia 15 del fitxer Entrenament.h.

5.4.2.6 victories

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

Array que desa a la posicií i la quantitat de victories que ha aconseguit la xarxa i

Definició a la línia 16 del fitxer Entrenament.h.

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

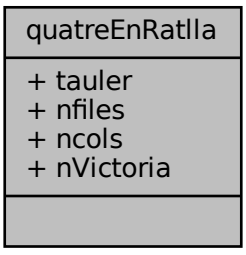
- Entrenament.h

5.5 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

5.5.1 Descripció Detallada

Estructura que conté l'estat d'una partida.

Definició a la línia 18 del fitxer 4enratlla.h.

5.5.2 Documentació dels Camps

5.5.2.1 ncols

`int quatreEnRatlla::ncols`

nombre de columnes del taulell

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

5.5.2.2 nfiles

`int quatreEnRatlla::nfiles`

nombre de files del taulell

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

5.5.2.3 nVictoria

`int quatreEnRatlla::nVictoria`

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

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

5.5.2.4 tauler

`signed char** quatreEnRatlla::tauler`

Taulell actual amb les fitxes colocades. 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 19 del fitxer 4enratlla.h.

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

- 4enratlla.h

Chapter 6

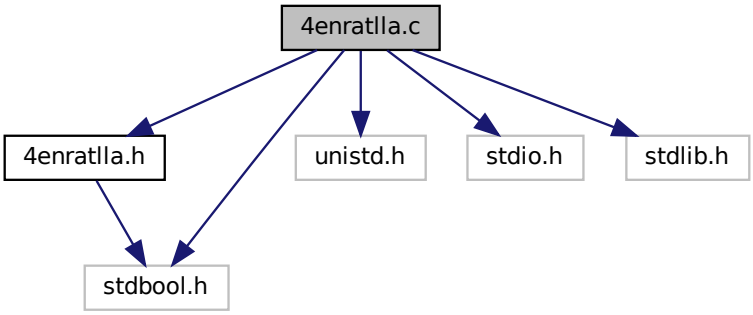
Documentació dels Fitxers

6.1 Referència del Fitxer 4enratlla.c

Fitxer que implementa les funcions básiques per a poder jugar una partida del cuatre 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 imprimirQuateEnRatlla (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 introduïr 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.

6.1.1 Descripció Detallada

Fitxer que implementa les funcions básiques per a poder jugar una partida del cuatre en ratlla.

6.1.2 Documentació de les Funcions

6.1.2.1 alliberarQuatreEnRatlla()

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

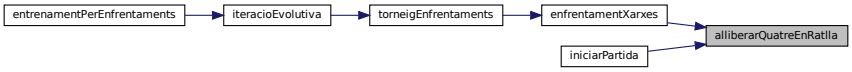
Allibera la partida del 4 en ratlla quan ja no es necessita.

Paràmetres

partida	és un apuntador a la partida que es vol alliberar.

Definició a la línia 51 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.2 comprovarColumnaPlena()

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

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

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

Paràmetres

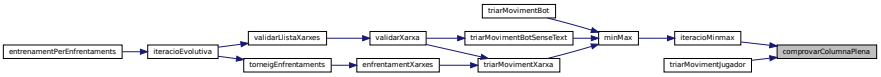
partida	és una apuntador a la partida que es vol comprovar si es pot fer el moviment.
moviment	és el número de columna que es vol comprovar si es pot introduïr una fitxa.

Retorna

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

Definició a la línia 57 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.3 comprovarEmpat()

```
bool comprovarEmpat (
            QuatreEnRatlla * partida )
```

Comprova si s'ha empatat la partida.

Paràmetres

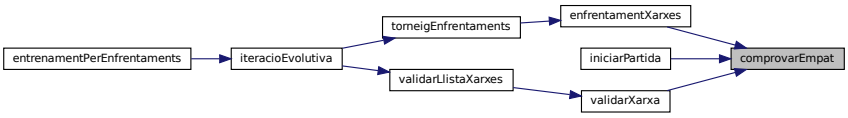
partida	un apuntador a la partida que es comprova.

Retorna

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

Definició a la línia 166 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.4 comprovarSolucio()

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

Comprova si un moviment fet ha guanyat la partida.

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

Paràmetres

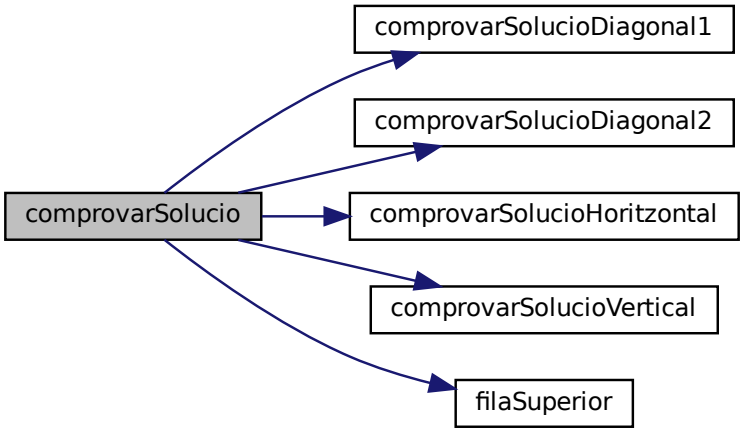
partida	és un apuntador a la partida que es vol comprovar.
int	és l'ultim moviment fet on es comprovarà si forma part d'una solució.

Retorna

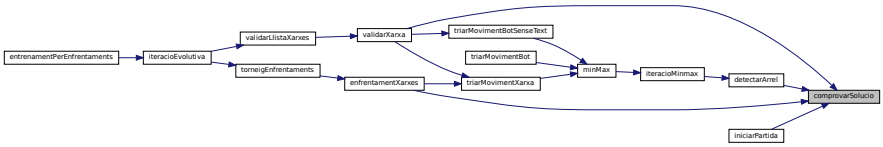
true si s'ha guanyat la partida i false si la partida continua.

Definició a la línia 161 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.1.2.5 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 dretas.

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

Paràmetres

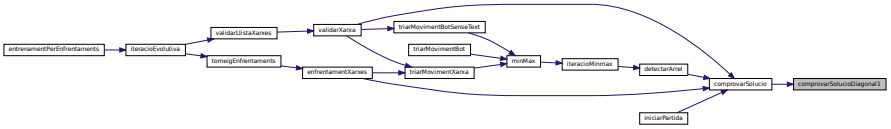
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 119 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.6 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 altes comprovar solució, ja que son molt semblans.

Paràmetres

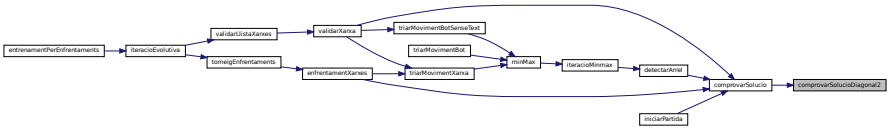
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 139 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.7 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 altes comprovar solució, ja que son molt semblans.

Paràmetres

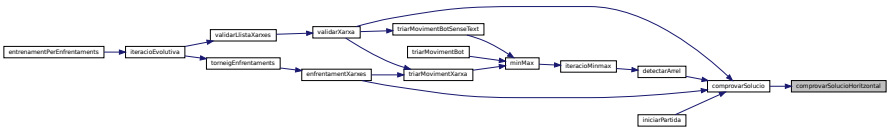
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient

Retorna

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

Definició a la línia 86 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.8 comprovarSolucioVertical()

```
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.

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

Paràmetres

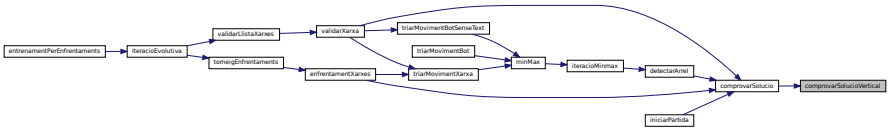
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 106 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.9 desferMoviment()

```
void desferMoviment (
            QuatreEnRatlla * partida,
            int moviment )
```

Desfar un moviment d'una partida.

El que fa és eliminar l'ultim moviment fer.

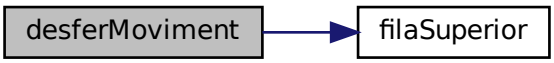
No té en compte si el movment s'ha fet anteriorment. (no verifica que l'acció sigui válida)

Paràmetres

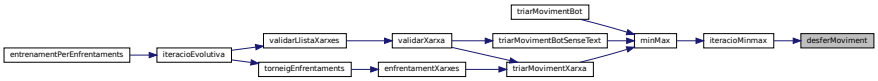
partida	és un apuntador a la partida en la que es vol desfer un moviment
moviment	és el número de columna del que es vol desfer l'últim moviment

Definició a la línia 71 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.1.2.10 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

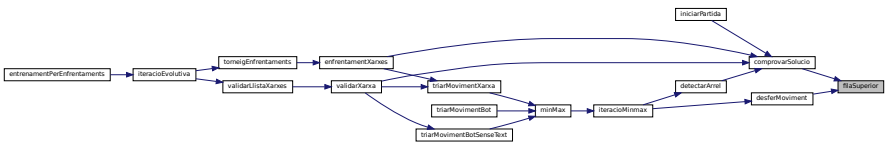
partida	és la partida de la qual es vol comprovar la fila superior.
columna	és la columna de la qual es vol comprovar la fila superior.

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 76 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.11 imprimirQuateEnRatlla()

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

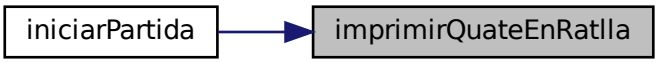
Imprimeix un taulell del 4 en ratlla.

Paràmetres

partida	és un apuntador la partida de la que es vol imprimir l'estat

Definició a la línia 17 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.12 inicialitzarQuatreEnRatlla()

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

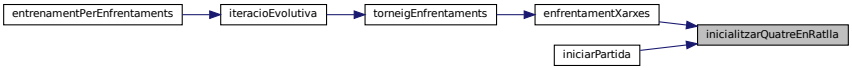
Inicia una partida del 4 en ratlla des de zero.

Paràmetres

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

Definició a la línia 40 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.13 realitzarMoviment()

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

Introdueix una fitxa d'un jugador en una columna.

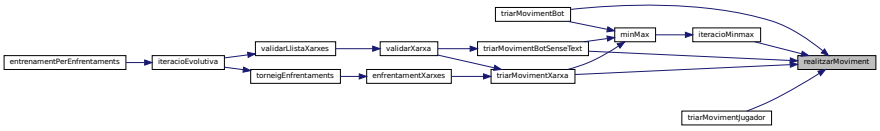
No té en compte si el moviment no es pot fer.

Paràmetres

partida	és un apuntador a la partida en la que es vol fer el moviment.
moviment	és el número de la columna en la que es vol introduïr la fitxa.
jugador	és el jugador que realitza el moviment.

Definició a la línia 62 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.1.2.14 reiniciarQuatreEnRatlla()

```
void reiniciarQuatreEnRatlla (
            QuatreEnRatlla * partida )
```

Fica tots els valors del tauler a 0.

Paràmetres

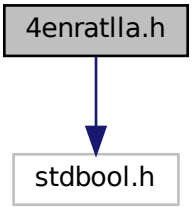
partida	és la partida que vols reiniciar.

Definició a la línia 34 del fitxer 4enratlla.c.

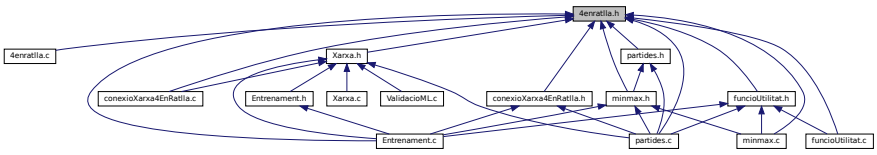
Gràfic de crides a aquesta funció:

6.2 Referència del Fitxer 4enratlla.h

`#include <stdbool.h>`
Inclou el graf de dependències per a 4enratlla.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Estructures de Dades

- struct quatreEnRatlla

  Estructura que conté l'estat d'una partida.

Definicions de Tipus

- typedef struct quatreEnRatlla QuatreEnRatlla

  Estructura que conté l'estat d'una partida.

Funcions

- void imprimirQuateEnRatlla (QuatreEnRatlla *partida)

  Imprimeix un taulell del 4 en ratlla.
- void inicialitzarQuatreEnRatlla (QuatreEnRatlla *partida, int nFils, int nCols, int nVictoria)

  Inicia una partida del 4 en ratlla des de zero.
- void reiniciarQuatreEnRatlla (QuatreEnRatlla *partida)

  Fica tots els valors del tauler a 0.
- 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 introduïr 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.

6.2.1 Documentació de les Funcions

6.2.1.1 alliberarQuatreEnRatlla()

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

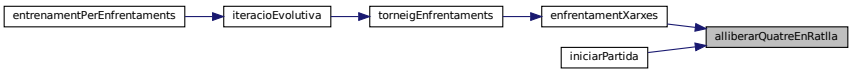
Allibera la partida del 4 en ratlla quan ja no es necessita.

Paràmetres

partida	és un apuntador a la partida que es vol alliberar.

Definició a la línia 51 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.2 comprovarColumnaPlena()

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

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

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

Paràmetres

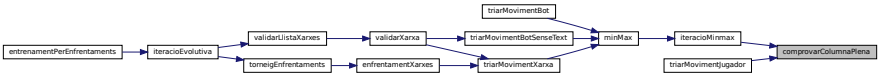
partida	és una apuntador a la partida que es vol comprovar si es pot fer el moviment.
moviment	és el número de columna que es vol comprovar si es pot introduïr una fitxa.

Retorna

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

Definició a la línia 57 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.3 comprovarEmpat()

```
bool comprovarEmpat (
            QuatreEnRatlla * partida )
```

Comprova si s'ha empatat la partida.

Paràmetres

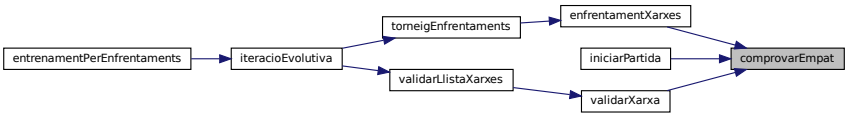
partida	un apuntador a la partida que es comprova.

Retorna

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

Definició a la línia 166 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.4 comprovarSolucio()

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

Comprova si un moviment fet ha guanyat la partida.

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

Paràmetres

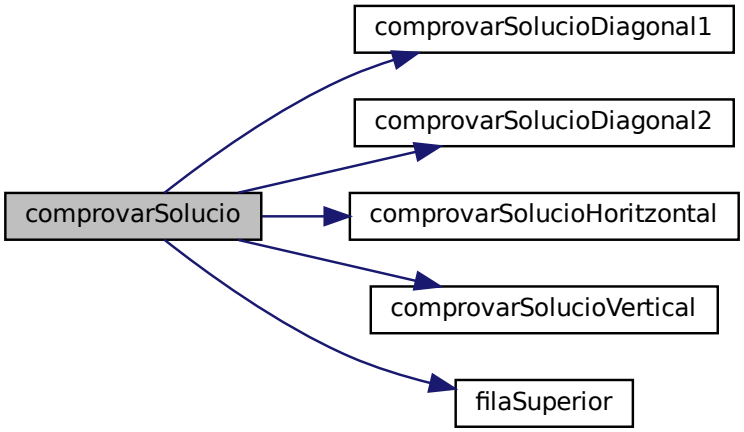
partida	és un apuntador a la partida que es vol comprovar.
int	és l'ultim moviment fet on es comprovarà si forma part d'una solució.

Retorna

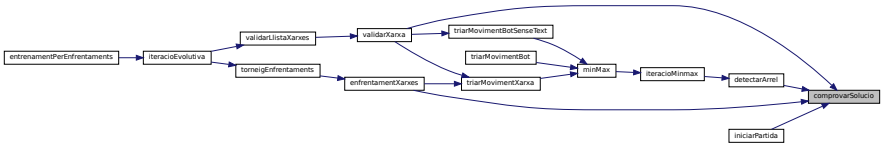
true si s'ha guanyat la partida i false si la partida continua.

Definició a la línia 161 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.2.1.5 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 dretas.

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

Paràmetres

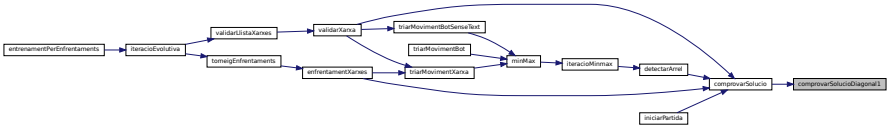
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 119 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.6 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 altes comprovar solució, ja que son molt semblans.

Paràmetres

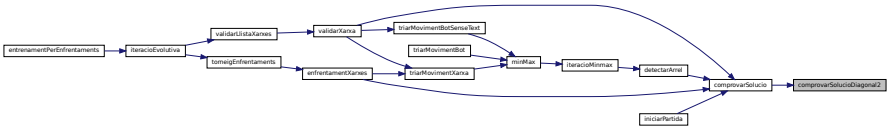
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 139 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.7 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 altes comprovar solució, ja que son molt semblans.

Paràmetres

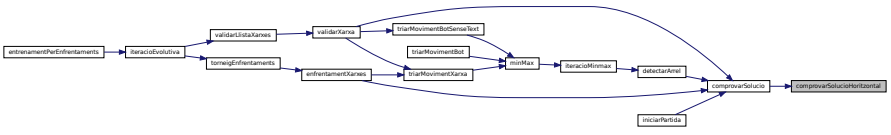
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient

Retorna

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

Definició a la línia 86 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.8 comprovarSolucioVertical()

```
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.

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

Paràmetres

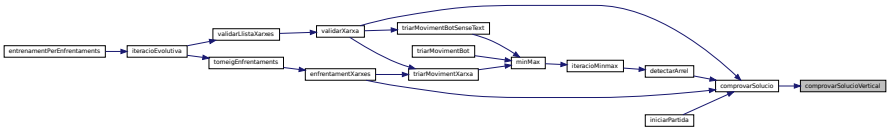
partida	és la partida que es vol comprovar.
fila	és la fila on s'ha fet el moviment.
col	és la columna on s'ha fet el movient.

Retorna

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

Definició a la línia 106 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.9 desferMoviment()

```
void desferMoviment (
            QuatreEnRatlla * partida,
            int moviment )
```

Desfar un moviment d'una partida.

El que fa és eliminar l'ultim moviment fer.

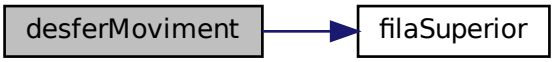
No té en compte si el movment s'ha fet anteriorment. (no verifica que l'acció sigui válida)

Paràmetres

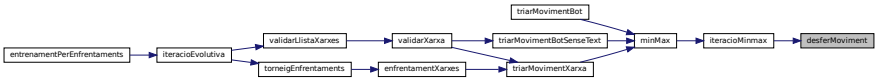
partida	és un apuntador a la partida en la que es vol desfer un moviment
moviment	és el número de columna del que es vol desfer l'últim moviment

Definició a la línia 71 del fitxer 4enratlla.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.2.1.10 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

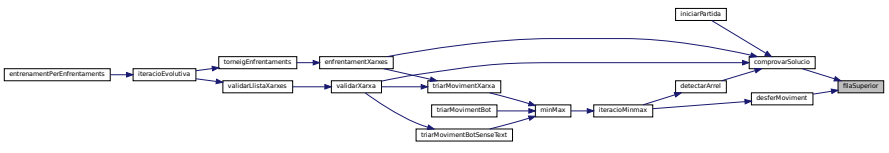
partida	és la partida de la qual es vol comprovar la fila superior.
columna	és la columna de la qual es vol comprovar la fila superior.

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 76 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.11 imprimirQuateEnRatlla()

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

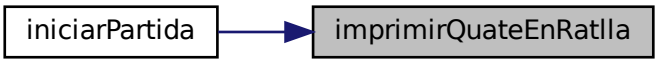
Imprimeix un taulell del 4 en ratlla.

Paràmetres

partida	és un apuntador la partida de la que es vol imprimir l'estat

Definició a la línia 17 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.12 inicialitzarQuatreEnRatlla()

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

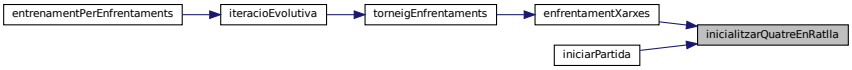
Inicia una partida del 4 en ratlla des de zero.

Paràmetres

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

Definició a la línia 40 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.13 realitzarMoviment()

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

Introdueix una fitxa d'un jugador en una columna.

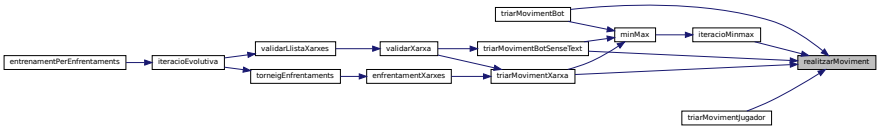
No té en compte si el moviment no es pot fer.

Paràmetres

partida	és un apuntador a la partida en la que es vol fer el moviment.
moviment	és el número de la columna en la que es vol introduïr la fitxa.
jugador	és el jugador que realitza el moviment.

Definició a la línia 62 del fitxer 4enratlla.c.

Gràfic de crides a aquesta funció:



6.2.1.14 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 34 del fitxer 4enratlla.c.

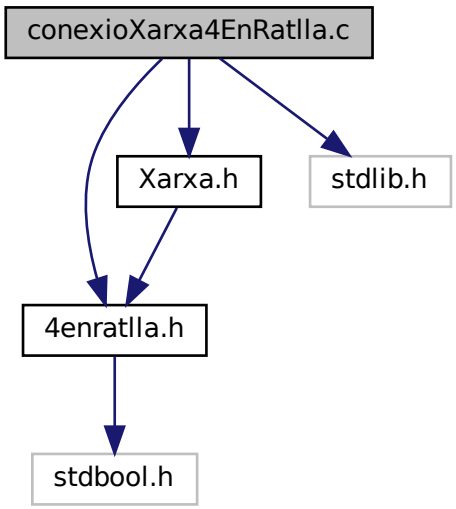
Gràfic de crides a aquesta funció:

6.3 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)

  Wraper que serveix per a donar a una xarxa el tipus de funció heurísica.

6.3.1 Descripció Detallada

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

6.3.2 Documentació de les Funcions

6.3.2.1 wrapperXarxa()

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

Wraper que serveix per a donar a una xarxa el tipus de funció heurísica.

Aplica una xarxa neuronal al taulell d'una partida.

Paràmetres

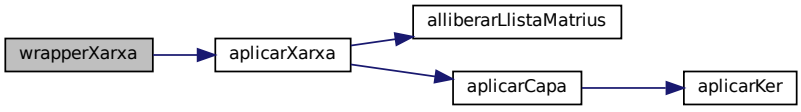
partida	és la partida a la que es vol aplicar la xarxa.
xarxaCtx	ha der ser un apuntador a una xarxaNeuronal, que es la que s'aplicarà a la partida.

Retorna

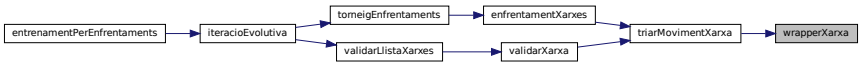
la puntuació a l'estat actual de la partida

Definició a la línia 12 del fitxer conexioXarxa4EnRatlla.c.

Gràfic de crides d'aquesta funció:

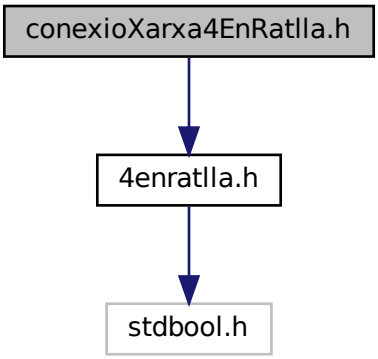


Gràfic de crides a aquesta funció:

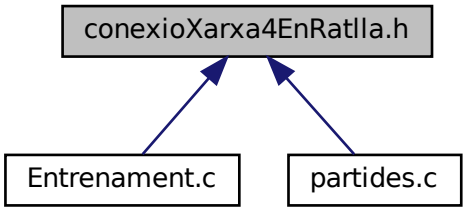


6.4 Referència del Fitxer conexioXarxa4EnRatlla.h

`#include "4enratlla.h"`
Inclou el graf de dependències per a conexioXarxa4EnRatlla.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Funcions

- double wrapperXarxa (QuatreEnRatlla ∗partida, void ∗xarxaCtx)

  Wraper que serveix per a donar a una xarxa el tipus de funció heurísica.

6.4.1 Documentació de les Funcions

6.4.1.1 wrapperXarxa()

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

Wraper que serveix per a donar a una xarxa el tipus de funció heurísica.

Paràmetres

partida	és la partida a la que es vol aplicar la xarxa.
xarxaCtx	ha der ser un apuntador a una xarxaNeuronal, que es la que s'aplicarà a la partida.

Retorna

la puntuació a l'estat actual de la partida

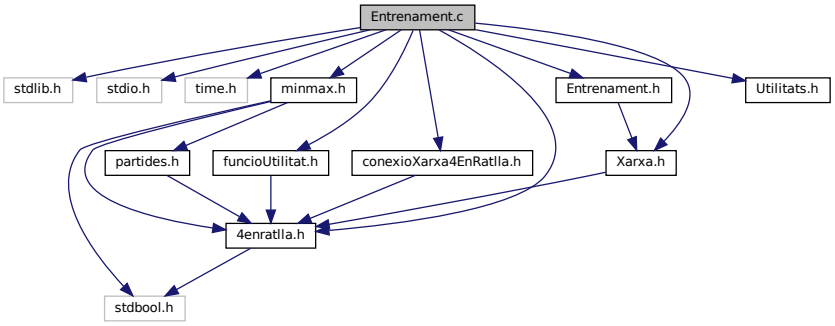
Definició a la línia 12 del fitxer conexioXarxa4EnRatlla.c.

6.5 Referència del Fitxer Entrenament.c

Fitxer que implementa les funcions necesàries 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 reultats 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** ()

6.5.1 Descripció Detallada

Fitxer que implementa les funcions necesàries per a l'entrenament de la CNN.

6.5.2 Documentació de les Funcions

6.5.2.1 crearNovaGeneracio()

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

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

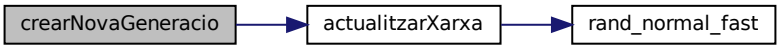
Paràmetres

antigaGeneracio	és la generació que es vol evolucionar.
millorsIndividus	és un array amb els index del millors indiviuds de la generació.

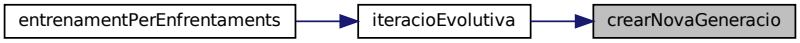
Per la implementació que s'ha dut a terme, pot ser que alguns pares tinguin menys o més fills que el que els hi pertoca. S'ha considerar que aquesta aleatorietat afegida no afecta a l'entrenament.

Definició a la línia 59 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.5.2.2 enfrentamentXarxes()

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

Enfrenta dues xarxes neuronals i retorna el guanyador.

Paràmetres

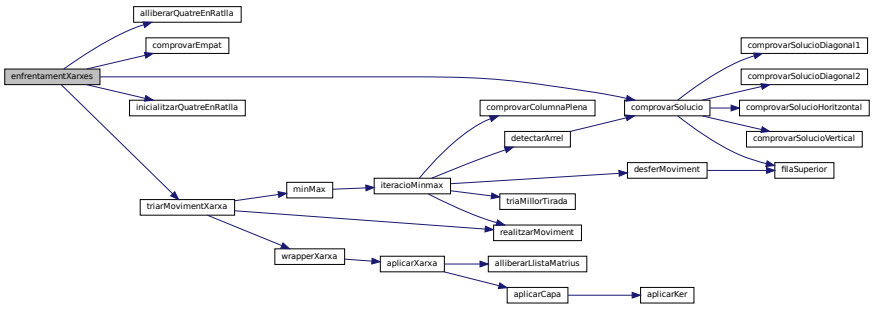
J1	és un apuntador a la xarxa neuronal que començarà tirant.
J2	és un apuntador a la xarxa neuronal que tirarà al segón torn.

Retorna

Retorna 0 si ha juanyat 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:

6.5.2.3 entrenamentPerEnfrentaments()

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

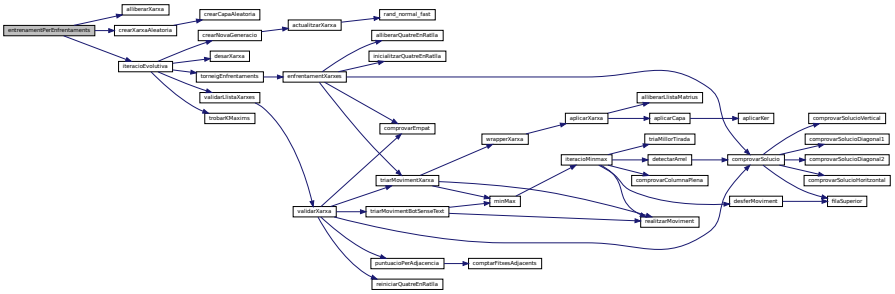
Crea una generació de xarxes y les va evolucionant per a arribar a una bona funció heurística.

Paràmetres

generacioXarxesJ1	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.
generacioXarxesJ2	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.
partida	és un apuntador a un QuatreEnRatlla inicialitzada i buida.
iteracions	és el nombre d'iteracions que fa l'entrenament.

Definició a la línia 214 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



6.5.2.4 iteracioEvolutiva()

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

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

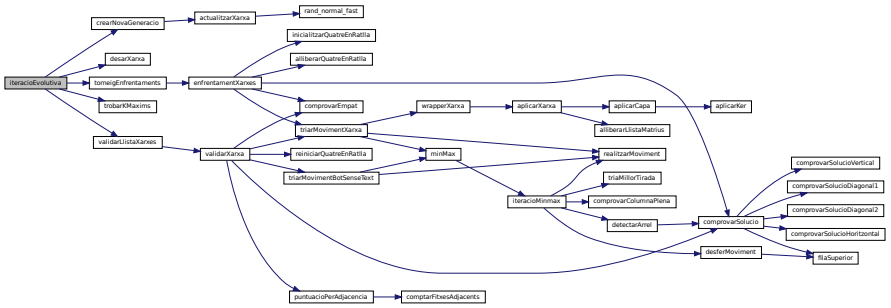
Desa la xarxa neuronal del J1 a un arxiu anomenat "XarxaEntrenadaJ1.DrusCNN" i la J2 a un arxiu anomenat "XarxaEntrenadaJ2.DrusCNN".

Paràmetres

generacioXarxesJ1	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.
generacioXarxesJ1	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.
partida	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:

6.5.2.5 torneigEnfrentaments()

```
void torneigEnfrentaments (
            Generacio * generacioXarxesJ1,
            Generacio * generacioXarxesJ2 )
```

Enfrenta a la generació de xarxes per a veure quina guanya més.

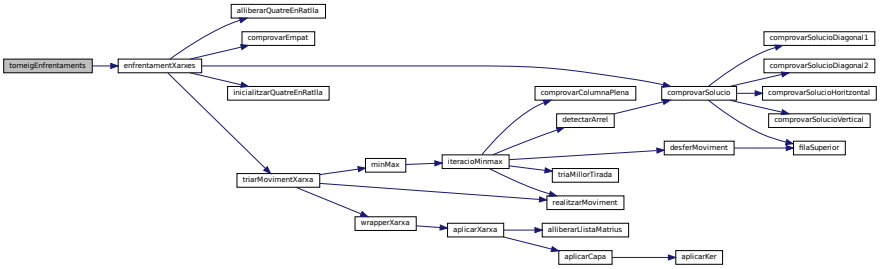
Enfrentes totes les xarxes en un format de lliga. Si la xarxa amb identifiacdor "i" guanya aleshores si li suma 1 a victories[i] de la seva generacio.

Paràmetres

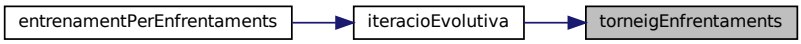
generacioXarxesJ1	és la generació de xarxes que es vol enfrentar i que començaran tirant.
generacioXarxesJ2	és la generació de xarxes que es vol enfrentar i que moura en segón lloc.

Definició a la línia 79 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.5.2.6 triarMovimentXarxa()

```
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.

Paràmetres

partida	és l'estat actual de la partida.
jugador	és al jugador que li toca tirar actualment.
xarxa	és la xarxa que farà de funció heurística

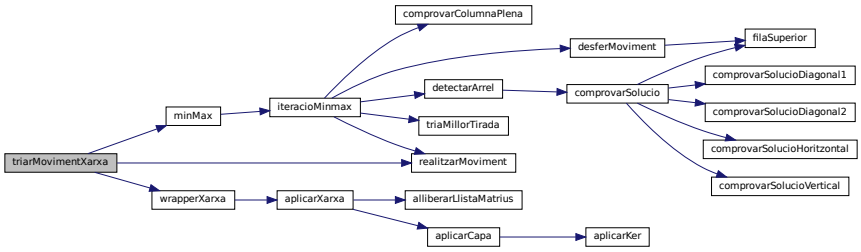
Compleix la classe selector de moviment.

Retorna

El moviment que ha decidit la xarxa. Només realitza moviments válids

Definició a la línia 22 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

6.5.2.7 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 reultats 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

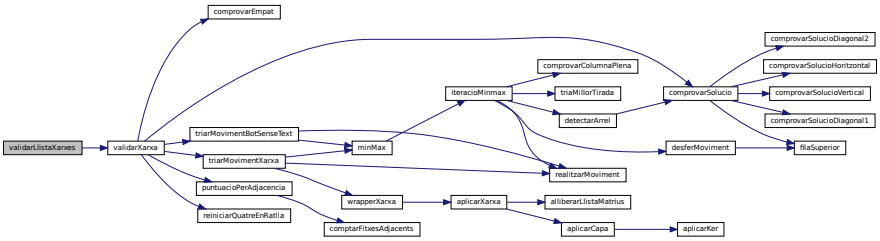
Cada fila és una nova validació.

Paràmetres

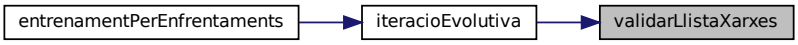
llistaXarxesJ1	és l'array de CNN que es volen avaluar en el primer torn de partida.
llistaXarxesJ2	és l'array de CNN que es volen avaluar en el segon torn de partida.
nXarxesJ1	és la longitut de l'array llista xarxes de primer torn.
nXarxesJ2	és la longitut de l'array llista xarxes de segon torn.
partida	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:



6.5.2.8 validarXarxa()

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

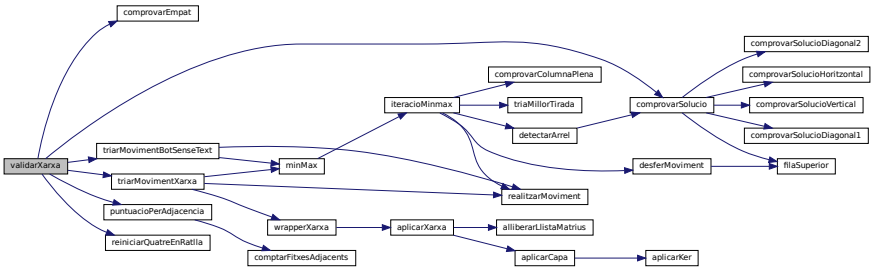
Simula una partida de la xarxa contra la IA de validació indicant el nombre de moviments i qui ha guanyat.

Paràmetres

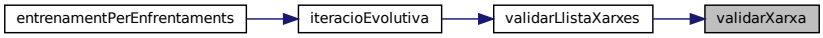
xarxa	és la xarxa que es vol validar.
tornXarxa	indica si la xarxa és el J1 o J2 (1 per a J1 i -1 per a J2)
nTorns	variable on es registraran els torns fets. En acabar la funció a aquest paràmetre se li ha afegit el nombre de moviments fets en total.
nVictories	variable on és registrarà si la xarxa ha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

Gràfic de crides d'aquesta funció:



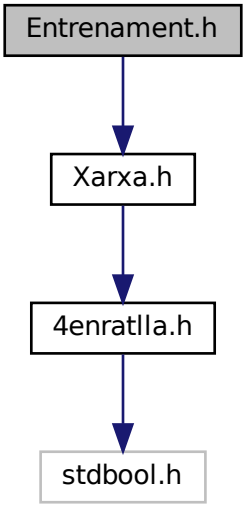
Gràfic de crides a aquesta funció:



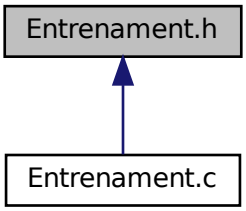
6.6 Referència del Fitxer Entrenament.h

```
#include "Xarxa.h"
```

Inclou el graf de dependències per a Entrenament.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Estructures de Dades

- struct generacio

  Estructura que controla com es formen les generacions.

Definicions de Tipus

- typedef struct generacio Generacio

  Estructura que controla com es formen les generacions.

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 reultats 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.

6.6.1 Documentació de les Funcions

6.6.1.1 crearNovaGeneracio()

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

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

Paràmetres

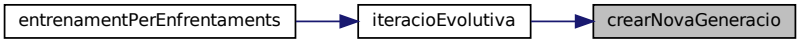
antigaGeneracio	és la generació que es vol evolucionar.
millorsIndividus	és un array amb els index del millors indiviuds de la generació.

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 considerar 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:



6.6.1.2 enfrentamentXarxes()

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

Enfrenta dues xarxes neuronals i retorna el guanyador.

Paràmetres

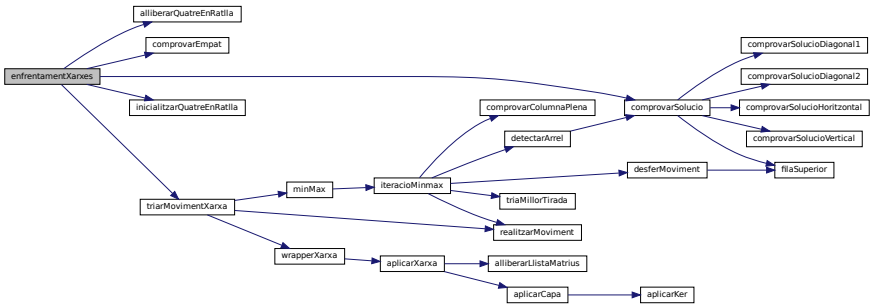
J1	és un apuntador a la xarxa neuronal que començarà tirant.
J2	és un apuntador a la xarxa neuronal que tirarà al segón torn.

Retorna

Retorna 0 si ha juanyat 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:

6.6.1.3 entrenamentPerEnfrentaments()

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

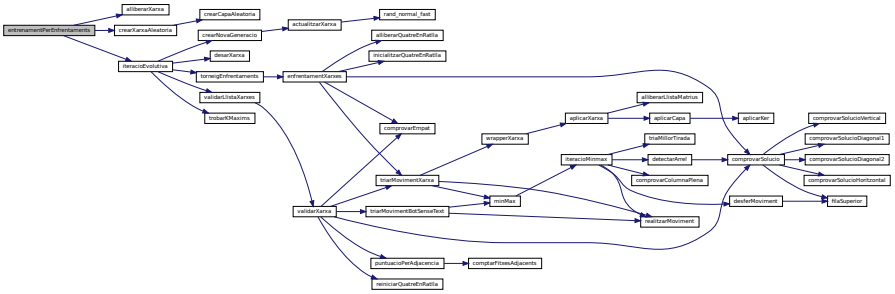
Crea una generació de xarxes y les va evolucionant per a arribar a una bona funció heurística.

Paràmetres

generacioXarxesJ1	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.
generacioXarxesJ2	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.
partida	és un apuntador a un QuatreEnRatlla inicialitzada i buida.
iteracions	és el nombre d'iteracions que fa l'entrenament.

Definició a la línia 214 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



6.6.1.4 iteracioEvolutiva()

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

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

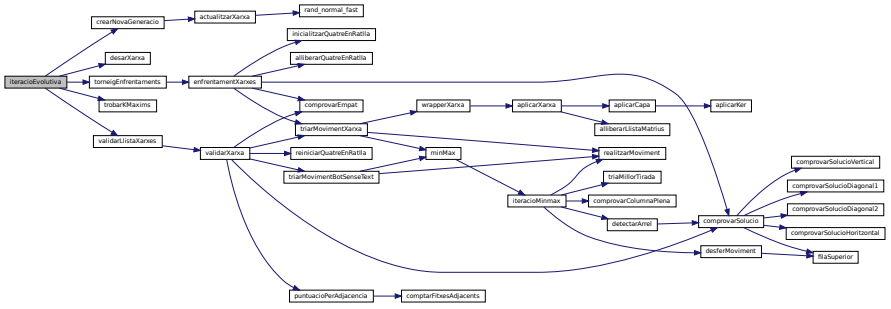
Desa la xarxa neuronal del J1 a un arxiu anomenat "XarxaEntrenadaJ1.DrusCNN" i la J2 a un arxiu anomenat "XarxaEntrenadaJ2.DrusCNN".

Paràmetres

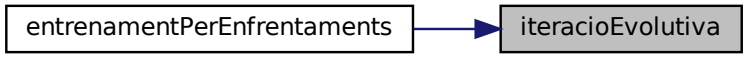
generacioXarxesJ1	és la generació de xarxes que es vol fer evolucionar per a millorar en el primer torn. En acabar la iteració ja retorna la generació evolucionada.
generacioXarxesJ1	és la generació de xarxes que es vol fer evolucionar per a millorar en el segon torn. En acabar la iteració ja retorna la generació evolucionada.
partida	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:



6.6.1.5 torneigEnfrentaments()

```
void torneigEnfrentaments (
            Generacio * generacioXarxesJ1,
            Generacio * generacioXarxesJ2 )
```

Enfrenta a la generació de xarxes per a veure quina guanya més.

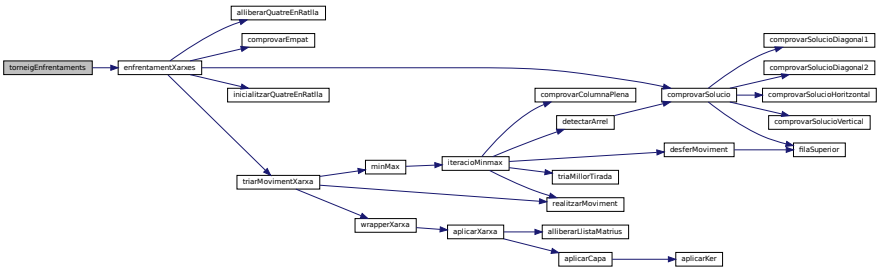
Enfrentes totes les xarxes en un format de lliga. Si la xarxa amb identifiacdor "i" guanya aleshores si li suma 1 a victories[i] de la seva generacio.

Paràmetres

generacioXarxesJ1	és la generació de xarxes que es vol enfrentar i que començaran tirant.
generacioXarxesJ2	és la generació de xarxes que es vol enfrentar i que moura en segón lloc.

Definició a la línia 79 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

6.6.1.6 triarMovimentXarxa()

```
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.

Paràmetres

partida	és l'estat actual de la partida.
jugador	és al jugador que li toca tirar actualment.
xarxa	és la xarxa que farà de funció heurística

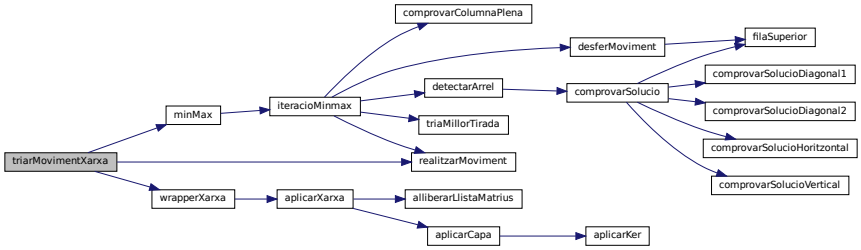
Compleix la classe selector de moviment.

Retorna

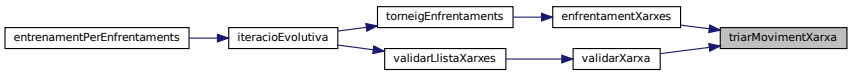
El moviment que ha decidit la xarxa. Només realitza moviments válids

Definició a la línia 22 del fitxer Entrenament.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.6.1.7 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 reultats 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

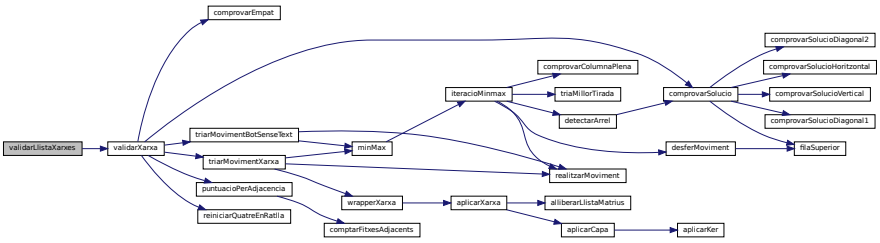
Cada fila és una nova validació.

Paràmetres

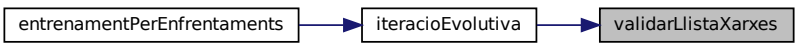
llistaXarxesJ1	és l'array de CNN que es volen avaluar en el primer torn de partida.
llistaXarxesJ2	és l'array de CNN que es volen avaluar en el segon torn de partida.
nXarxesJ1	és la longitut de l'array llista xarxes de primer torn.
nXarxesJ2	és la longitut de l'array llista xarxes de segon torn.
partida	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:



6.6.1.8 validarXarxa()

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

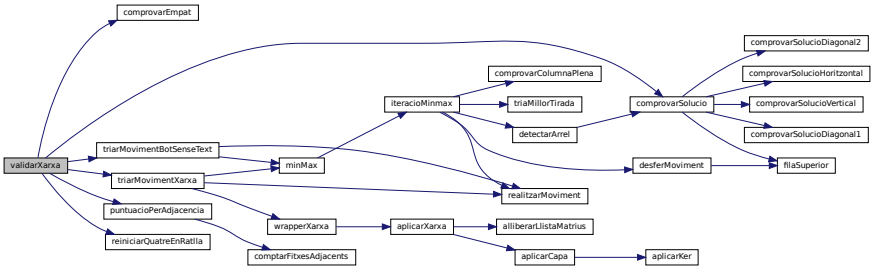
Simula una partida de la xarxa contra la IA de validació indicant el nombre de moviments i qui ha guanyat.

Paràmetres

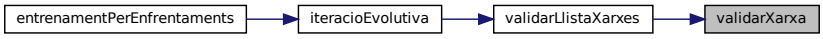
xarxa	és la xarxa que es vol validar.
tornXarxa	indica si la xarxa és el J1 o J2 (1 per a J1 i -1 per a J2)
nTorns	variable on es registraran els torns fets. En acabar la funció a aquest paràmetre se li ha afegit el nombre de moviments fets en total.
nVictories	variable on és registrarà si la xarxa ha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

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:

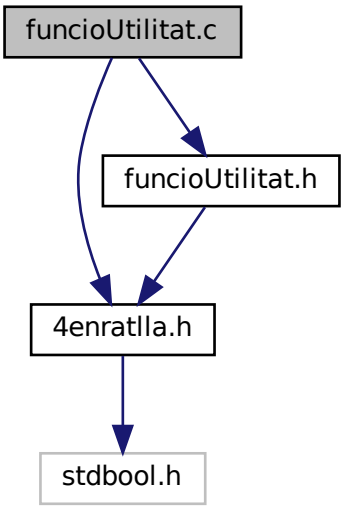


6.7 Referència del Fitxer funcioUtilitat.c

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

```
#include "4enratlla.h"
#include "funcioUtilitat.h"
```

Inclou el graf de dependències per a funcioUtilitat.c:



Funcions

- void comptarFitxesAdjacents (QuatreEnRatlla ∗partida, int fila, int col, int ∗fitxesJ1, int ∗fitxesJ2)

  Compta quantes fitxer de cada tipos hi ha al voltant d'una posició en concret.
- double puntuacioPerAdjacencia (QuatreEnRatlla ∗partida, void ∗ctx)

  Valora un estat de partida.

6.7.1 Descripció Detallada

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

6.7.2 Documentació de les Funcions

6.7.2.1 comptarFitxesAdjacents()

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

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

Paràmetres

partida	és l'estat de partida actual
fila	fila on es la posició que es vol comprovar
col	columna on es la posició que es vol comprovar
fitxesJ1	direcció per on retorna el nombre de fitxes del J1
fitxesJ1	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:

6.7.2.2 puntuacioPerAdjacencia()

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

Valora un estat de partida.

Paràmetres

partida	és l'estat que es vol valorar
ctx	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

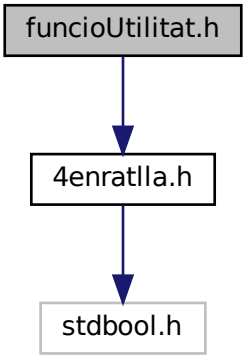
Definició a la línia 25 del fitxer funcioUtilitat.c.

Gràfic de crides d'aquesta funció:

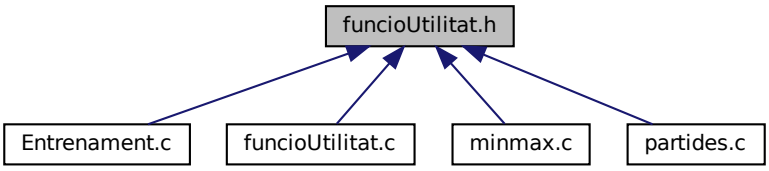
Gràfic de crides a aquesta funció:

6.8 Referència del Fitxer funcioUtilitat.h

`#include "4enratlla.h"`
Inclou el graf de dependències per a funcioUtilitat.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Funcions

- void comptarFitxesAdjacents (QuatreEnRatlla ∗partida, int fila, int col, int ∗fitxesJ1, int ∗fitxesJ2)

  Compta quantes fitxer de cada tipos hi ha al voltant d'una posició en concret.
- double puntuacioPerAdjacencia (QuatreEnRatlla ∗partida, void ∗ctx)

  Valora un estat de partida.

6.8.1 Documentació de les Funcions

6.8.1.1 comptarFitxesAdjacents()

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

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

Paràmetres

partida	és l'estat de partida actual
fila	fila on es la posició que es vol comprovar
col	columna on es la posició que es vol comprovar
fitxesJ1	direcció per on retorna el nombre de fitxes del J1
fitxesJ1	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:

6.8.1.2 puntuacioPerAdjacencia()

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

Valora un estat de partida.

Paràmetres

partida	és l'estat que es vol valorar
ctx	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:

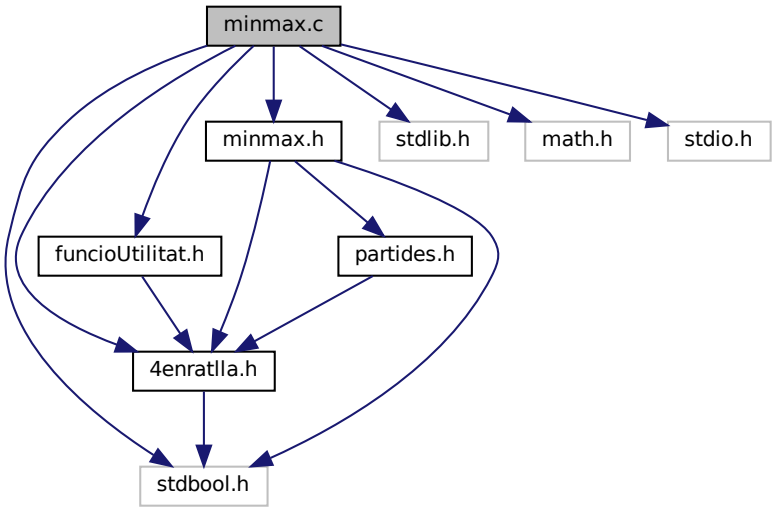
Gràfic de crides a aquesta funció:

6.9 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.

6.9.1 Descripció Detallada

Fitxer que implementa l'algorisme del minMax.

6.9.2 Documentació de les Funcions

6.9.2.1 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 victoria (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 punuació 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 acutal de la partida.

Paràmetres

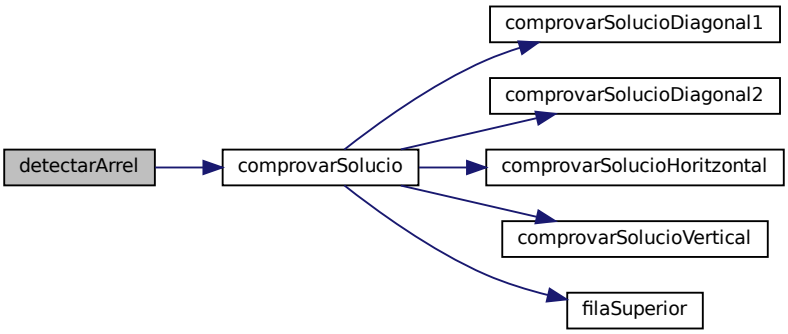
partida	és l'estat de la partida que es vol puntuar.
moviment	és el l'últim moviment fet per a arribar a aquest estat.
jugadorOriginal	és el jugador que està fent el moviment actual.
valoracio	és el un aptuntador per on es retornarà la valoració del node.
profunditat	és a la profunditat que ens trobem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

Retorna

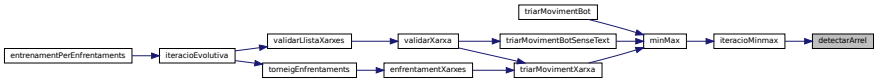
retorna cert en cas de que el node sigui una fulla i false en cas contrari.

Definició a la línia 67 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.9.2.2 iteracioMinmax()

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

realitza l'algorisme del MinMax

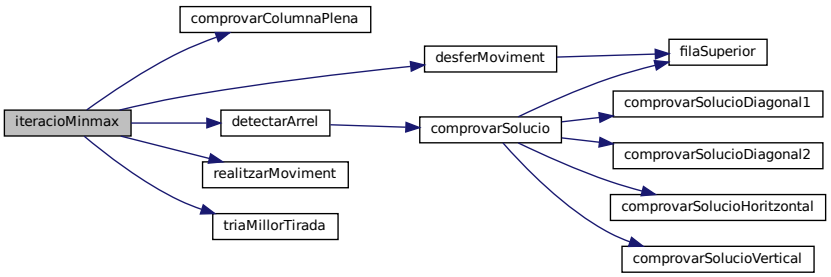
Per fer

Paràmetres

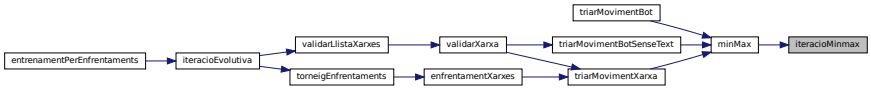
partida	és l'estat acutal de la partida (sense modificar el moviment actual).
jugadorOriginal	és el jugador al que li toca fer el moviment a l'inici del minmax.
nivellNode	és a quina profunditat de l'arbre ens trobem.
puntuacioNode	és un apuntador que conté la puntuació assignada al node.
profunditatNode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



6.9.2.3 minMax()

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

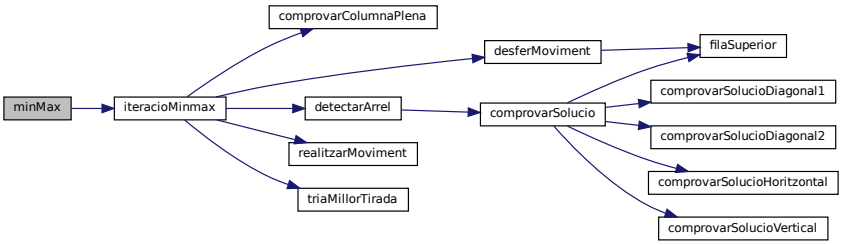
Comença l'algorisme del MinMax.

Paràmetres

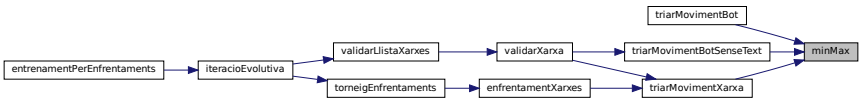
partida	l'estat de la partida en la que es fa el minmax.
jugadorOriginal	és el jugador al que li toca tirar.
fHeuristica	és la funció heurísitca que s'utilitza per a fer el minmax.
ctxHeuristica	és una estructura que conté dades necesàries per a la funció heurística.

Definició a la línia 18 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.9.2.4 triaMillorTirada()

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

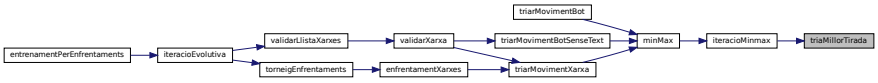
Compara el node actual amb el millor d'aquella bifuració.

Paràmetres

millorMoviment	és el millor moviment fet en aquella bifuracació. Si el moviment actual és millor, actualitza el valor
movimentActual	és el moviment fet per a arribar a aquell estat.
millorValoracio	és la valoració del millor moviment d'aquella biuració.
valoracioActual	és la valoració del node actual.
accioDesempat	és la profunditat relativa a la que s'ha arribat a la millor valoració de la bifurcació.
accioDesempatActual	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.

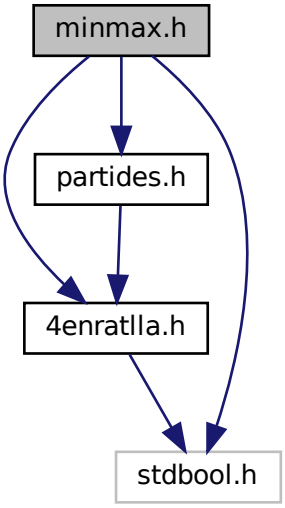
Gràfic de crides a aquesta funció:



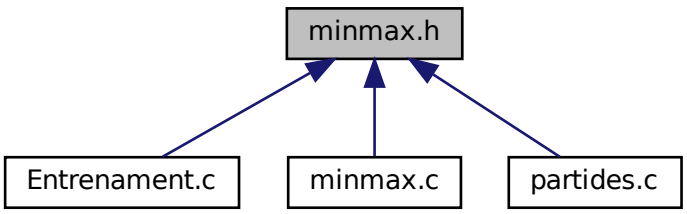
6.10 Referència del Fitxer minmax.h

```
#include "4enratlla.h"
#include "partides.h"
#include <stdbool.h>
```

Inclou el graf de dependències per a minmax.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Definicions

- #define **PROFUNDITAT** 3

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.

6.10.1 Descripció Detallada

Per fer s'hauria de ficar la prodfunditat com a paràmetre.

6.10.2 Documentació de les Funcions

6.10.2.1 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 victoria (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 punuació 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 acutal de la partida.

Paràmetres

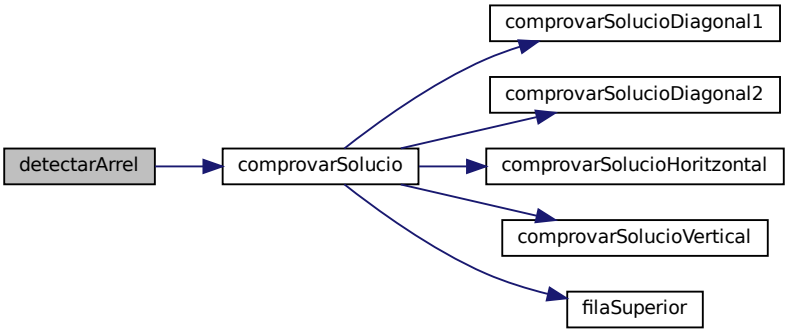
partida	és l'estat de la partida que es vol puntuar.
moviment	és el l'últim moviment fet per a arribar a aquest estat.
jugadorOriginal	és el jugador que està fent el moviment actual.
valoracio	és el un aptuntador per on es retornarà la valoració del node.
profunditat	és a la profunditat que ens trobem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

Retorna

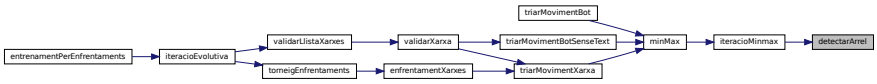
retorna cert en cas de que el node sigui una fulla i false en cas contrari.

Definició a la línia 67 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.10.2.2 iteracioMinmax()

```
int iteracioMinmax (
            QuatreEnRatlla * partida,
            signed char jugadorOriginal,
            int nivellNode,
```

```
double * puntuacioNode,
int * profunditatNode,
funcioHeuristica fHeuristica,
void * ctxHeuristica )
```

realitza l'algorisme del MinMax

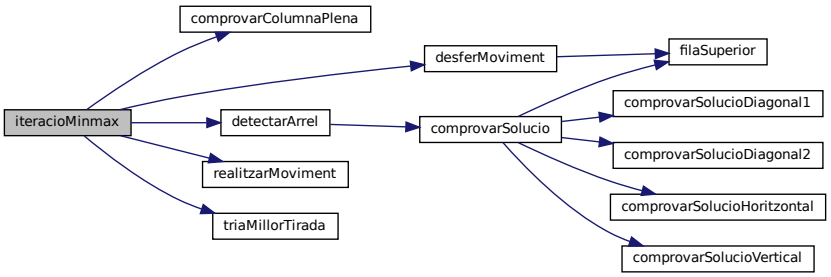
Per fer

Paràmetres

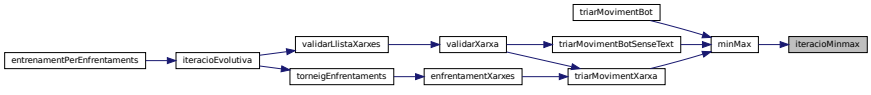
partida	és l'estat acutal de la partida (sense modificar el moviment actual).
jugadorOriginal	és el jugador al que li toca fer el moviment a l'inici del minmax.
nivellNode	és a quina profunditat de l'arbre ens trobem.
puntuacioNode	és un apuntador que conté la puntuació assignada al node.
profunditatNode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



6.10.2.3 minMax()

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

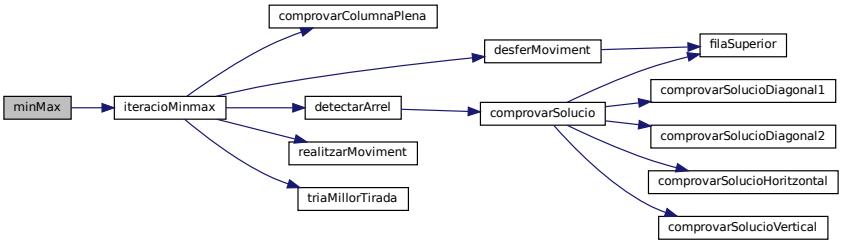
Comença l'algorisme del MinMax.

Paràmetres

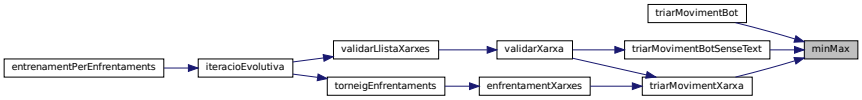
partida	l'estat de la partida en la que es fa el minmax.
jugadorOriginal	és el jugador al que li toca tirar.
fHeuristica	és la funció heurísitca que s'utilitza per a fer el minmax.
ctxHeuristica	és una estructura que conté dades necesàries per a la funció heurística.

Definició a la línia 18 del fitxer minmax.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.10.2.4 triaMillorTirada()

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

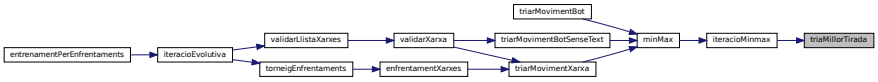
Compara el node actual amb el millor d'aquella bifuració.

Paràmetres

millorMoviment	és el millor moviment fet en aquella bifuracació. Si el moviment actual és millor, actualitza el valor
movimentActual	és el moviment fet per a arribar a aquell estat.
millorValoracio	és la valoració del millor moviment d'aquella biuració.
valoracioActual	és la valoració del node actual.
accioDesempat	és la profunditat relativa a la que s'ha arribat a la millor valoració de la bifurcació.
accioDesempatActual	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:

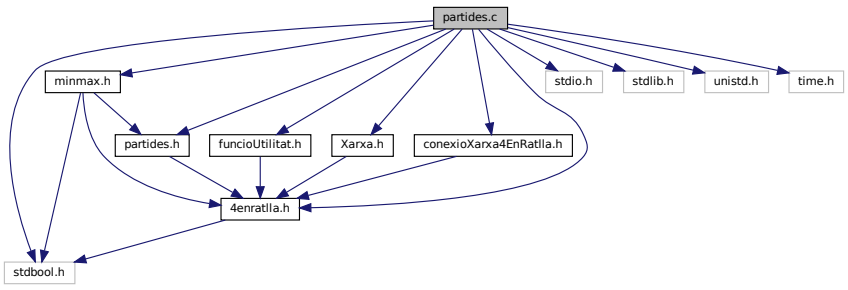


6.11 Referència del Fitxer partides.c

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 probablitat 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 decisioJ1, selectorDeMoviment decisioJ2, double esperaEntreTorns, void ∗ctx)
- void **validacioXarxa** ()
- void **validacioNormal** ()
- int **main** ()

6.11.1 Descripció Detallada

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

6.11.2 Documentació de les Funcions

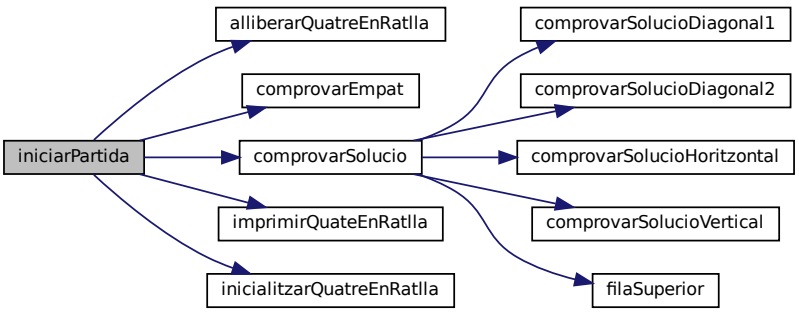
6.11.2.1 iniciarPartida()

```
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:



6.11.2.2 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 pasar com a context.

Paràmetres

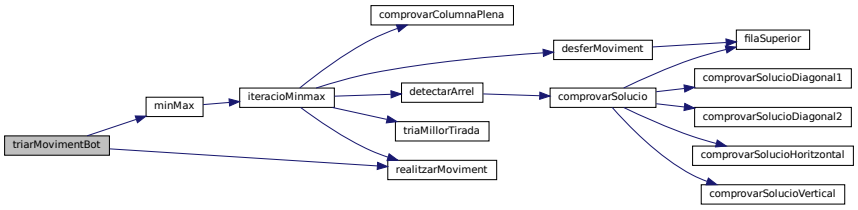
partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

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:



6.11.2.3 triarMovimentBotAleatori()

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

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

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

Paràmetres

partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	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.

6.11.2.4 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 pasar com a context.

Paràmetres

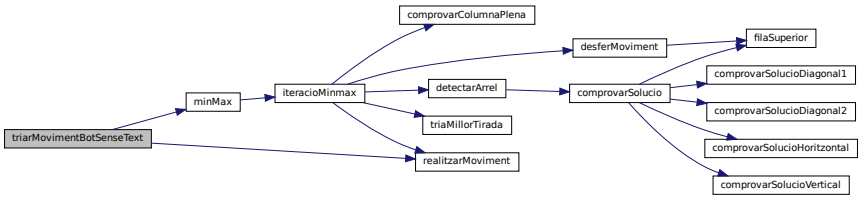
partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

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:

6.11.2.5 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 válidad. 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

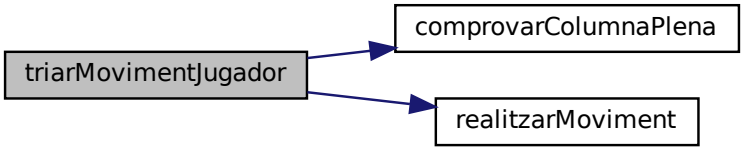
partida	és un apuntador a la partida que s'està jugant
jugador	és el jugador que fa el moviment.
ctx	és un apuntador que no ha de contenir res, només li dona el tipus de selectorDeMoviment.

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.

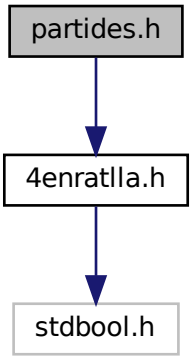
Gràfic de crides d'aquesta funció:



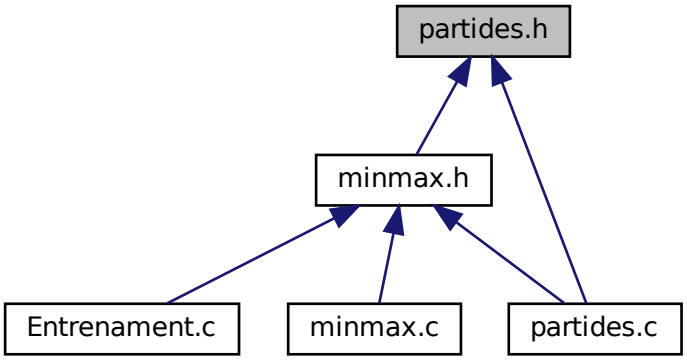
6.12 Referència del Fitxer partides.h

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

Inclou el graf de dependències per a partides.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Estructures de Dades

- struct contextHeuristica

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

Definicions de Tipus

- typedef int(∗ selectorDeMoviment) (QuatreEnRatlla ∗, signed char, void ∗)

  Tipus de dada que representa una funció que donat un QuatreEnRatlla y un jugador, decideix quin moviment es realitzarà.
- typedef double(∗ funcioHeuristica) (QuatreEnRatlla ∗partida, void ∗ctx)

  Tipus de dada que representa una funció que donada una partida assigna un double. Aquest representa lo ben posicionat que està el jugador que ha començat la partida per a guanyar-la.
- typedef struct contextHeuristica ContextHeuristica

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

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 probablitat 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.

6.12.1 Documentació de les Definicions de Tipus

6.12.1.1 ContextHeuristica

```
typedef struct contextHeuristica ContextHeuristica
```

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

Per fer potser es podría implementar la profunditat variable per aquí.

6.12.1.2 funcioHeuristica

```
typedef double(* funcioHeuristica) (QuatreEnRatlla *partida, void *ctx)
```

Tipus de dada que representa una funció que donada una partida assigna un double. Aquest representa lo ben posicionat que està el jugador que ha començat la partida per a guanyar-la.

Reb com a paràmetres l'estat de la partida actual y el context. Aquest inclou altres paràmetres que reb la funció.

Definició a la línia 26 del fitxer partides.h.

6.12.2 Documentació de les Funcions

6.12.2.1 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 pasar com a context.

Paràmetres

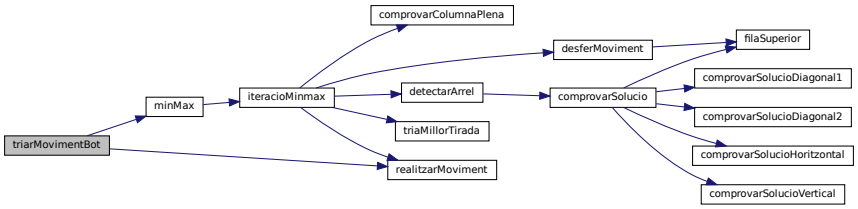
partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

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:



6.12.2.2 triarMovimentBotAleatori()

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

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

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

Paràmetres

partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	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.

6.12.2.3 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 pasar com a context.

Paràmetres

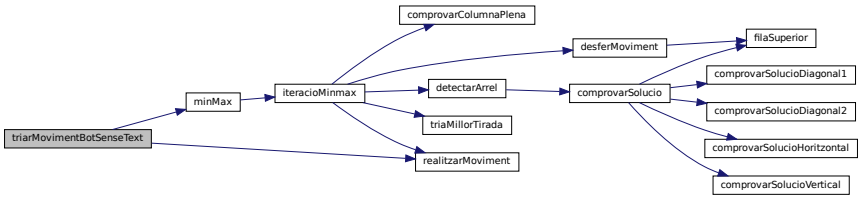
partida	és un apuntador a la partida que s'està jugant.
jugador	és el jugador que fa el moviment.
ctx	serveix per a passar paràmetres extra (con funcions heurístiques) però preservant el tipus selectorDeMoviment. Aquest ha de ser de la classe context heurística.

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:

6.12.2.4 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 válidad. 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

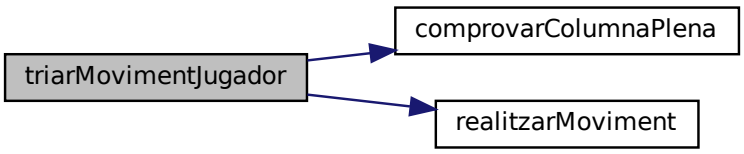
partida	és un apuntador a la partida que s'està jugant
jugador	és el jugador que fa el moviment.
ctx	és un apuntador que no ha de contenir res, només li dona el tipus de selectorDeMoviment.

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:

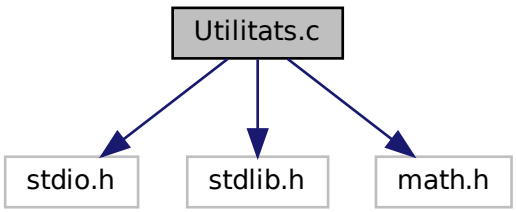


6.13 Referència del Fitxer Utilitats.c

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 copia 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.

6.13.1 Descripció Detallada

Fitxer que inclou funcions amb utilitats diverses sense dependències.

6.13.2 Documentació de les Funcions

6.13.2.1 alliberarLlistaMatrius()

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

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

Paràmetres

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

Definició a la línia 12 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:

6.13.2.2 deepCopyMatriu()

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

Fa una copia profunda d'una matriu de doubles.

Paràmetres

matriu	és la matriu que vols copiar.
dimFil	és el nombre de files que té la matriu.
dimCol	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.

6.13.2.3 imprimirMatriu()

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

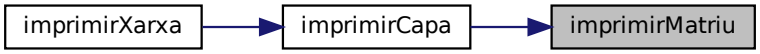
Imprimeix una matriu.

Paràmetres

matriu	és la matriu que vols imprimir
dimFil	és el nombre de files que té la matriu
dimCol	és el nombre de columnes que té la matriu

Definició a la línia 23 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



6.13.2.4 leakyReLU()

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

Rectificador Lineal Unitari Leaky.

Paràmetres

x	és la variable de la funció.

Retorna

x si x>=0 y a∗x si x<0.

Definició a la línia 85 del fitxer Utilitats.c.

6.13.2.5 rand_normal_fast()

```
double rand_normal_fast ( )
```

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

Atenció

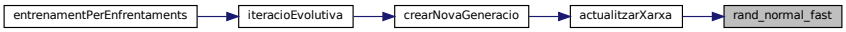
S'ha de vigilar perque aquesta funció no es pot fer servir en paral·le degut a que fa servir variables static.

Retorna

el nombre aleatori

Definició a la línia 59 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



6.13.2.6 trobarKMaxims()

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

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

Paràmetres

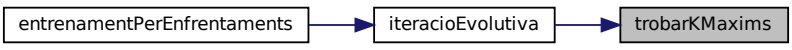
llista	és la llista de la que es volen trobar els màxims.
midaLlista	és la longitud de la llista.
k	és el nombre de valors més grans que vols trobar.

Retorna

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

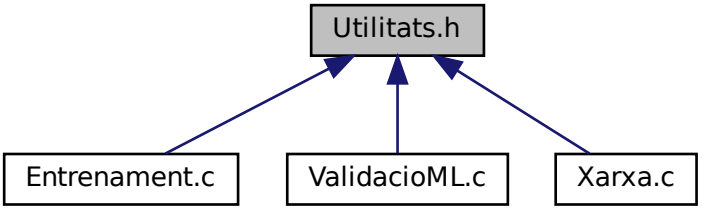
Definició a la línia 43 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



6.14 Referència del Fitxer Utilitats.h

Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



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 copia 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.

6.14.1 Documentació de les Funcions

6.14.1.1 alliberarLlistaMatrius()

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

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

Paràmetres

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

Definició a la línia 12 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:

6.14.1.2 deepCopyMatriu()

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

Fa una copia profunda d'una matriu de doubles.

Paràmetres

matriu	és la matriu que vols copiar.
dimFil	és el nombre de files que té la matriu.
dimCol	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.

6.14.1.3 imprimirMatriu()

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

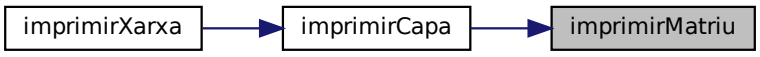
Imprimeix una matriu.

Paràmetres

matriu	és la matriu que vols imprimir
dimFil	és el nombre de files que té la matriu
dimCol	és el nombre de columnes que té la matriu

Definició a la línia 23 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



6.14.1.4 leakyReLU()

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

Rectificador Lineal Unitari Leaky.

Paràmetres

x	és la variable de la funció.

Retorna

x si x>=0 y a∗x si x<0.

Definició a la línia 85 del fitxer Utilitats.c.

6.14.1.5 rand_normal_fast()

```
double rand_normal_fast ( )
```

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

Atenció

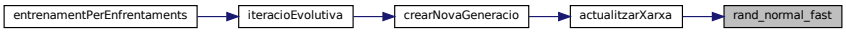
S'ha de vigilar perque aquesta funció no es pot fer servir en paral·le degut a que fa servir variables static.

Retorna

el nombre aleatori

Definició a la línia 59 del fitxer Utilitats.c.

Gràfic de crides a aquesta funció:



6.14.1.6 trobarKMaxims()

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

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

Paràmetres

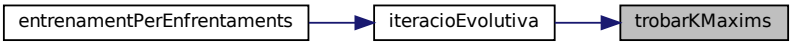
llista	és la llista de la que es volen trobar els màxims.
midaLlista	és la longitud de la llista.
k	és el nombre de valors més grans que vols trobar.

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:

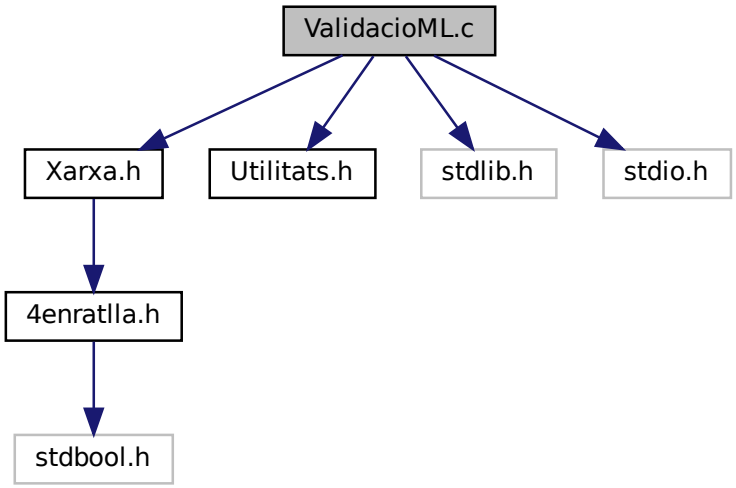


6.15 Referència del Fitxer ValidacioML.c

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** ()

6.15.1 Descripció Detallada

Fitxer per a validar les xarxes.

Atenció

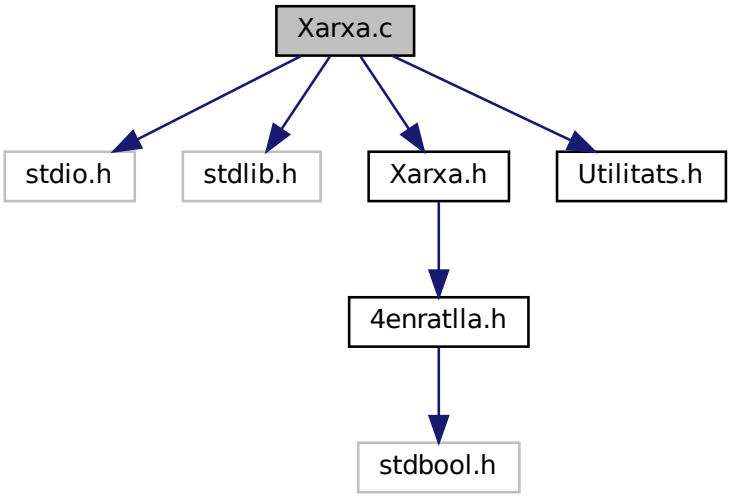
Segurament està anticuat i algunes funcións d'han d'actualitzar.

6.16 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 longitut 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 dimFilIn, int dimColIn)

  Crea una capa amb núclis amb valors aleatoris.
- XarxaNeuronal * crearXarxaAleatoria (int nCapes, int *llistaDimKernels, int *llistaNKernels, int dimFilIn, int dimColIn)

  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 neuronar 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.

6.16.1 Descripció Detallada

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

Per fer Crec que milloraría bastant la velocitat si s'implementes paralelització amb la gràfica. Per a fer això s'auria de canviar l'struct capaXarxa per a que uses double *kernel enlloc de ****kerner per a així usar memòria contigua.

6.16.2 Documentació de les Funcions

6.16.2.1 actualitzarXarxa()

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

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

Les dues xarxes han de tenir les mateixes dimensions.

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

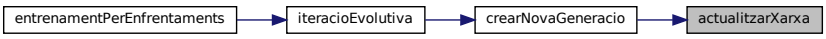
Paràmetres

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

Definició a la línia 187 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:



6.16.2.2 alliberarXarxa()

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

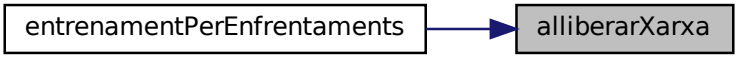
allibera una Xarxa

Paràmetres

xarxa	és la xarxa que es vol alliberar.

Definició a la línia 166 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



6.16.2.3 aplicarCapa()

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

Donades una llista de matrius els hi aplica una capa d'una CNN.

Paràmetres

llistaMatrius	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.
capa	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 70 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.16.2.4 aplicarKer()

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

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

Paràmetres

llistaMatrius	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.
dimFilMat	nombre de files que tenen les matrius.
dimColMat	nombre de columnes que tenen les matrius
profunditat	profunditat que té el kernel (que coincideix amb el nombre de matrius de l'array llistaMatrius).
kernel	és el kernel que es vol aplicar.
dimKer	és el nombre de files/columnes que té el kernel.
biaix	és el biaix que s'aplica a la convolució.
activacio	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 38 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.16.2.5 aplicarXarxa()

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

Donada una matriu li aplica una xarxa neuronal.

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

Paràmetres

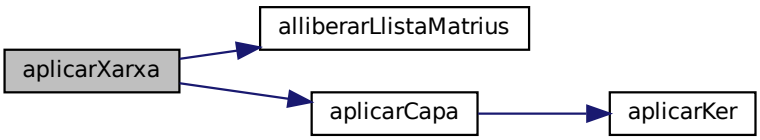
matriu	és la matriu inicial.
xarxa	és la xarxa que es vol aplicar a la matriu.

Retorna

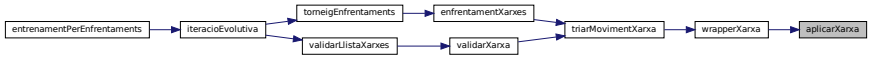
Retorna el valor resultant d'aplicar la xarxa a la matriu.

Definició a la línia 85 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.16.2.6 carregarXarxa()

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

Carrega una xarxa neuronal des d'un arxiu.

Fet per la IA.

Paràmetres

filename	és el nom del fitxer des del que es vol carregar la xarxa.

Retorna

un apuntador a la xarxa carregada.

Definició a la línia 240 del fitxer Xarxa.c.

6.16.2.7 crearCapaAleatoria()

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

Crea una capa amb núclis amb valors aleatoris.

Paràmetres

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

Retorna

Retorna l'apuntador a la capa creada.

Definició a la línia 117 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.16.2.8 crearXarxaAleatoria()

```
XarxaNeuronal* crearXarxaAleatoria (
            int nCapes,
            int * llistaDimKernels,
            int * llistaNKernels,
            int dimFilIn,
            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

nCapes	és el nombre de capes de la xarxa
llistaDimKernels	és un array on l'element i és la dimensió que tenen els kernels de la capa i.
llistaNKernels	és un array on l'element i és el nombre de kernels de la capa i.
dimFilIn	és el nombre de files de la matriu que reb com a entrada la xarxa.
dimColIn	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 143 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.16.2.9 desarXarxa()

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

Desa una xarxa neuronal a un arxiu.

Fet per la IA.

Paràmetres

xarxa	és la xarxa que es vol desar.
filename	és el nom del fitxer en el que es vol desar la xarxa.

Definició a la línia 207 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.16.2.10 imprimirCapa()

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

Imprimeix els kernels d'una capa.

Paràmetres

capa	és la capa que es vol imprimir

Definició a la línia 17 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.16.2.11 imprimirXarxa()

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

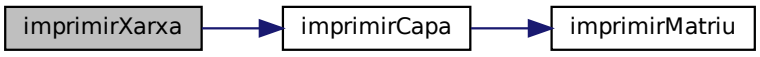
Imprimeix els kernels de les capes d'una xarxa.

Paràmetres

xarxa	és la xarxa que es vol imprimir

Definició a la línia 30 del fitxer Xarxa.c.

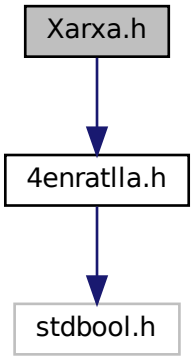
Gràfic de crides d'aquesta funció:



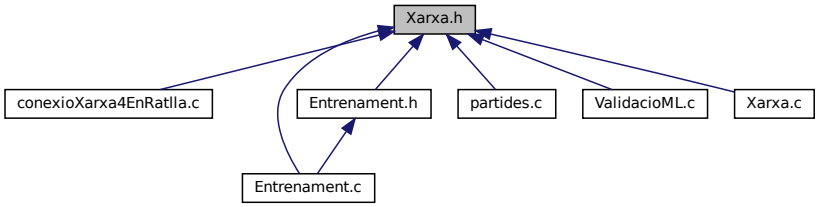
6.17 Referència del Fitxer Xarxa.h

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

Inclou el graf de dependències per a Xarxa.h:



Aquest gràfic mostra quins fitxers inclouen, de forma directa o indirecta, aquest fitxer:



Estructures de Dades

- struct capaXarxa

  Estructura que conté una capa de la xarxa.
- struct CNN

  Estructura que encapsula una xarxa neuronal.

Definicions

- #define **ACTIVACIO** leakyReLU

Definicions de Tipus

- typedef double(∗ funcioReal) (double)

  Tipus de dada que representa la funció d'activació.
- typedef struct capaXarxa CapaXarxa

  Estructura que conté una capa de la xarxa.
- typedef struct CNN XarxaNeuronal

  Estructura que encapsula una xarxa neuronal.

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 longitut 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 dimFilIn, int dimColIn)

  Crea una capa amb núclis amb valors aleatoris.
- XarxaNeuronal ∗ crearXarxaAleatoria (int nCapes, int ∗llistaDimKernels, int ∗llistaNKernels, int dimFilIn, int dimColIn)

  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 neuronar 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.
- double wrapperXarxa (QuatreEnRatlla ∗partida, void ∗xarxa)

  Aplica una xarxa neuronal al taulell d'una partida.

6.17.1 Documentació de les Definicions de Tipus

6.17.1.1 XarxaNeuronal

```
typedef struct CNN XarxaNeuronal
```

Estructura que encapsula una xarxa neuronal.

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

6.17.2 Documentació de les Funcions

6.17.2.1 actualitzarXarxa()

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

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

Les dues xarxes han de tenir les mateixes dimensions.

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

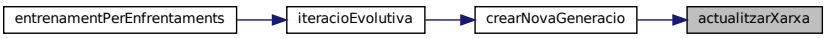
Paràmetres

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

Definició a la línia 187 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:



6.17.2.2 alliberarXarxa()

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

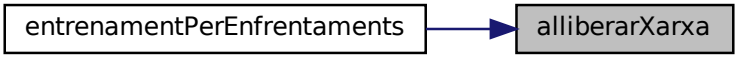
allibera una Xarxa

Paràmetres

xarxa	és la xarxa que es vol alliberar.

Definició a la línia 166 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:



6.17.2.3 aplicarCapa()

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

Donades una llista de matrius els hi aplica una capa d'una CNN.

Paràmetres

llistaMatrius	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.
capa	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 70 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.17.2.4 aplicarKer()

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

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

Paràmetres

llistaMatrius	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.
dimFilMat	nombre de files que tenen les matrius.
dimColMat	nombre de columnes que tenen les matrius
profunditat	profunditat que té el kernel (que coincideix amb el nombre de matrius de l'array llistaMatrius).
kernel	és el kernel que es vol aplicar.
dimKer	és el nombre de files/columnes que té el kernel.
biaix	és el biaix que s'aplica a la convolució.
activacio	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 38 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.17.2.5 aplicarXarxa()

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

Donada una matriu li aplica una xarxa neuronal.

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

Paràmetres

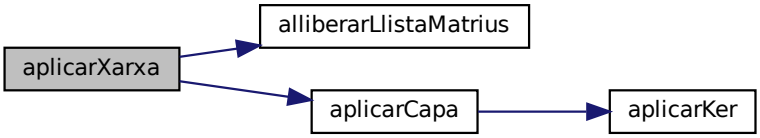
matriu	és la matriu inicial.
xarxa	és la xarxa que es vol aplicar a la matriu.

Retorna

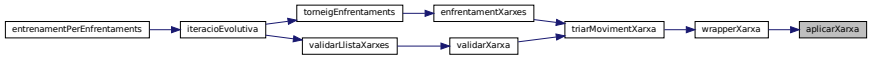
Retorna el valor resultant d'aplicar la xarxa a la matriu.

Definició a la línia 85 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



Gràfic de crides a aquesta funció:



6.17.2.6 carregarXarxa()

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

Carrega una xarxa neuronal des d'un arxiu.

Fet per la IA.

Paràmetres

filename	és el nom del fitxer des del que es vol carregar la xarxa.

Retorna

un apuntador a la xarxa carregada.

Definició a la línia 240 del fitxer Xarxa.c.

6.17.2.7 crearCapaAleatoria()

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

Crea una capa amb núclis amb valors aleatoris.

Paràmetres

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

Retorna

Retorna l'apuntador a la capa creada.

Definició a la línia 117 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.17.2.8 crearXarxaAleatoria()

```
XarxaNeuronal* crearXarxaAleatoria (
            int nCapes,
            int * llistaDimKernels,
            int * llistaNKernels,
            int dimFilIn,
            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

nCapes	és el nombre de capes de la xarxa
llistaDimKernels	és un array on l'element i és la dimensió que tenen els kernels de la capa i.
llistaNKernels	és un array on l'element i és el nombre de kernels de la capa i.
dimFilIn	és el nombre de files de la matriu que reb com a entrada la xarxa.
dimColIn	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 143 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.17.2.9 desarXarxa()

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

Desa una xarxa neuronal a un arxiu.

Fet per la IA.

Paràmetres

xarxa	és la xarxa que es vol desar.
filename	és el nom del fitxer en el que es vol desar la xarxa.

Definició a la línia 207 del fitxer Xarxa.c.

Gràfic de crides a aquesta funció:

6.17.2.10 imprimirCapa()

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

Imprimeix els kernels d'una capa.

Paràmetres

capa	és la capa que es vol imprimir

Definició a la línia 17 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:

Gràfic de crides a aquesta funció:

6.17.2.11 imprimirXarxa()

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

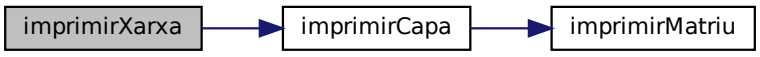
Imprimeix els kernels de les capes d'una xarxa.

Paràmetres

xarxa	és la xarxa que es vol imprimir

Definició a la línia 30 del fitxer Xarxa.c.

Gràfic de crides d'aquesta funció:



6.17.2.12 wrapperXarxa()

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

Aplica una xarxa neuronal al taulell d'una partida.

Aquesta funció és necesaria per a poder passar una xarxa com a paràmetre a les funcions que prenen com a paràmetres funcions heurístiques.

Paràmetres

partida	és la partida de la que es vol obtenir el taulell.
ctx	és un apuntador a la xarxa neuronal que es vol passar.

Retorna

la aplicació de la xarxa al taulell de la partida.

Aplica una xarxa neuronal al taulell d'una partida.

Paràmetres

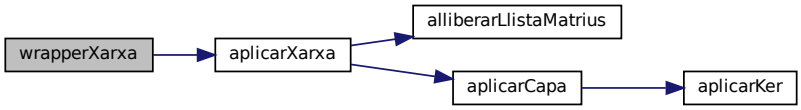
partida	és la partida a la que es vol aplicar la xarxa.
xarxaCtx	ha der ser un apuntador a una xarxaNeuronal, que es la que s'aplicarà a la partida.

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: