<<Página en blanco completamente necesaria no borrar>>

Conjunto de candidatos (C): vértices del grafo no seleccionados con grado mayor que 0 (contando sólo las aristas restantes).

Conjunto de aristas restantes (A): aristas que no han sido cubiertas todavía por ningún nodo seleccionado.

Conjunto de seleccionados (S): nodos seleccionados por la función de selección

Función solución: el conjunto de aristas restantes está vacío.

Función selección (FS): función que selecciona uno de los nodos adyacentes al nodo con menor grado (contando sólo las aristas restantes) de los candidatos excluyendo a aquellos de grado 0.

Función objetivo: Devuelve el conjunto mínimo de nodos que envuelven el grafo.

Recubrimiento minimal (N, B)

S = Ø

C = N //N es Vértices del grafo

A = B //B es el conjunto de aristas del grafo sin envolver

mientras (A != Ø) hacer //Función solución (el conjunto de aristas del grafo está vacío)

x = FS(C) //Seleccionar el adyacente al nodo de menor grado.

C = C - {x} //Lo elimina del conjunto de candidatos.

S = S U {x} //Añadir el vértice al conjunto de seleccionados.

A = A - {x.aristas} //Eliminar las aristas cubiertas del conjunto A.

C = C - {nodos de grado 0 en |C,A|}

fin mientras

devolver S

fin

Explicación:

Cuando en un grafo tenemos un nodo de grado 1, significa que ese nodo está unido únicamente a otro nodo mediante una arista. Para envolver esa arista, será necesario siempre tomar ese nodo o su vecino. Ante esta elección, no tiene sentido escoger el nodo de grado 1, pues sólo tomaría una arista, mientras que si tomamos el vecino podemos estar, potencialmente, envolviendo más aristas.

Basado en ese principio, el algoritmo buscará los nodos con menor grado para tomar a sus vecinos. Si el nodo de menor grado tiene solo grado 1, se tomará automáticamente el nodo vecino, si tiene varios vecinos, tomará uno de ellos.

A continuación, eliminará de la lista de aristas todas aquellas que queden cubiertas y eliminará de la lista de candidatos los nodos que tras este proceso se hayan quedado con grado 0.

Mientras queden aristas sin cubrir, repetirá este proceso hasta tener todo el grafo envuelto.

vector<int> recubriMin(vector<int> & N, vector< vector<int> > & B, int &aristas){

vector<int> S;

while(aristas != 0){

int x = fS(N, B);

for(int i = 0; i < N.size() ; i++){

if(N[i] == x) N.erase(N.begin()+i); break;

}

S.push\_back(x);

for( int i = 0; i < B.size(); i++){

if(B[i][x] == 1){

aristas--;

B[i][x] = 0;

if(comprobarSiEs0(i, &B)){

for(int j = 0; j < N.size(); j++){

if(N[j] == x) N.erase(N.begin()+j); break;

}

}

}

if(B[x][i] == 1){

aristas--;

B[x][i] = 0;

if(comprobarSiEs0(i, &B)){

for(int j = 0; j < N.size(); j++){

if(N[j] == x) N.erase(N.begin()+j); break;

}

}

}

}

}

}

bool comprobarSiEs0(int i, vector< vector<int> > & B){

bool Es0 = true;

for(int j = 0; i < B.size(); i++){

if(B[j][i] != 0 || B[i][j] != 0){

Es0 = false;

}

}

return Es0;

}

int funcionSeleccion(vector<int> & N, vector< vector<int> > & B){

int x;

int minGrado = 100000;

for(int i = 0; i < N.size() || minGrado == 1; i++){

int grado = 0;

int nodo = N[i];

for(int j = 0; i < B.size(); i++){

if(B[j][nodo] != 0 || B[nodo][j] != 0){

grado++;

}

}

if(grado < minGrado && grado > 0) x = nodo;

}

bool encontrado = false;

for(int i = 0; i < B.size() && !encontrado; i++){

if(B[x][i] != 0){

x = i;

encontrado = true;

}

}

return x;

}

Demostración:

Partimos de que una solución con nuestro algoritmo es S y consideramos que hay una solución S’ mejor. Esta solución cumple que tiene al menos un nodo seleccionado menos, luego |S’| < |S|

Tomemos el nodo p con menor grado del grafo. La arista o aristas que tenga ese nodo han de ser recubiertas para tener la solución y para ello solo hay dos posibilidades, tomar ese nodo o tomar a su adyacente. S tiene al adyacente y como S’ difiere de S debe tomar el propio p. Como el grado de p es menor o igual que el grado del nodo adyacente, el número de aristas cubiertas por S’ es menor o igual que el número de aristas cubiertas por S. Para compensar las aristas no cubiertas por ese nodo será necesario coger un número igual o mayor de nodos que S, lo que nos lleva a una contradicción, pues habíamos supuesto que |S’| < |S|.



