

Generalized Node Deletion

Quick Tutorial

Gruppo EUREKA

Calabrò Michele	224521
-----------------	--------

Colantonio Viviana	224473
--------------------	--------

Costa Cristian Giuseppe	227507
-------------------------	--------

1. Introduzione	3
2. Interfaccia grafica	6
3. Il menu File	7
3.1 Esegui esempio	7
3.2 Crea nuovo input	10
3.3 Carica	10
4. Il menu Code	11
5. Il menu Help	12
6. Le schermate Grafo F e Grafo G	13
7. La schermata k	14

1. Introduzione

L'esercizio della traccia è stato rielaborato nel seguente modo:

Generalized Node Deletion

Siano dati due grafi F e G ed un intero k. Il problema è stabilire se sia possibile rimuovere **esattamente** k vertici dal grafo F in modo tale che il grafo risultante F' non contenga G come sottografo.

Ossia, si è sostituito “al più” con “esattamente”, in quanto si tratta di problemi equivalenti. Indipendentemente dall'input, il codice risolutore in DLV per l'esercizio è il seguente

```
node_F(X) :- edge_F(X, Y).
node_F(Y) :- edge_F(X, Y).

node_G(X) :- edge_G(X, Y).
node_G(Y) :- edge_G(X, Y).

compl_node_Fs(X) | node_Fs(X) :- node_F(X).

edge_Fs(X, Y)      :- node_Fs(X), node_Fs(Y), edge_F(X,Y).
compl_edge_Fs(X, Y) :- compl_node_Fs(X), compl_node_Fs(Y), edge_F(X,Y).

:- not #count{ X : compl_node_Fs(X)} = k.

count_G(X) :- #count{Y : node_G(Y)} = X.

f(X, Y)      :- node_G(Y), node_Fs(X), not n_f(X, Y).
n_f(X, Y)    :- node_G(Y), node_Fs(X), not f(X, Y).

% check che il mapping vada bene
NO :- node_Fs(A), node_Fs(B), node_G(C), f(A, C), f(B, C), A != B.
NO :- node_G(A), node_G(B), node_Fs(C), f(C, A), f(C, B), A != B.
NO :- edge_G(X,Y), f(A, X), f(B, Y), not edge_Fs(A,B), not edge_Fs(B,A).
NO :- count_G(X), #count{I : f(Z, I)} < X.
NO :- count_G(X), #count{I : f(Z, I)} > X.

:- NO.
```

Verrà ora illustrato il ragionamento implementato dal programma DLV.

Nelle righe

```
node_F(X) :- edge_F(X, Y).
node_F(Y) :- edge_F(X, Y).

node_G(X) :- edge_G(X, Y).
node_G(Y) :- edge_G(X, Y).
```

si istanziano eventuali nodi non presenti nella base di conoscenza in input (ma presenti negli archi), sia per F, sia per G.

Nelle righe

```
compl_node_Fs(X) | node_Fs(X) :- node_F(X).

edge_Fs(X, Y)      :- node_Fs(X), node_Fs(Y), edge_F(X,Y).
compl_edge_Fs(X, Y) :- compl_node_Fs(X), compl_node_Fs(Y), edge_F(X,Y).

:- not #count{ X : compl_node_Fs(X) } = k.
```

si esegue un *Guess* per trovare un sottografo F' di F, con esattamente k nodi in meno di F.

Tale sottografo è chiamato Fs in DLV.

Il vincolo su “esattamente k nodi” è espresso attraverso il predicato di aggregazione **#count**, il quale verifica che effettivamente il numero di predicati nel *symbolic set* { X : **compl_node_Fs(X)** } sia pari a k.

Nelle righe

```
count_G(X) :- #count{Y : node_G(Y)} = X.

f(X, Y)      :- node_G(Y), node_Fs(X), not n_f(X, Y).
n_f(X, Y)    :- node_G(Y), node_Fs(X), not f(X, Y).
```

vengono contati i nodi in G (sarà utile nei vincoli per modelli stabili, spiegati dopo) e viene inoltre effettuato un *Guess* di un possibile isomorfismo tra F' e G. Ossia, il predicato **f** andrà ad individuare una funzione biiettiva tra l'insieme dei nodi di F' e l'insieme dei nodi di G.

Ovviamente, qualora tale funzione esistesse, l'output del programma dovrebbe essere **FALSE**.

Infine, nelle righe

```
% check che il mapping vada bene
NO :- node_Fs(A), node_Fs(B), node_G(C), f(A, C), f(B, C), A != B.
NO :- node_G(A), node_G(B), node_Fs(C), f(C, A), f(C, B), A != B.
NO :- edge_G(X,Y), f(A, X), f(B, Y), not edge_Fs(A,B), not edge_Fs(B,A).
NO :- count_G(X), #count{I : f(Z, I)} < X.
NO :- count_G(X), #count{I : f(Z, I)} > X.

:- NO.
```

vengono posti i vincoli relativi ad **f**.

In particolare, la funzione dev'essere biiettiva e deve inoltre essere tale da garantire che a archi in G corrispondano archi isomorfi in F.

Ciò è imposto tramite i vincoli:

- il primo vincolo, assieme gli ultimi 2, impone che la funzione sia iniettiva;
- il secondo, assieme gli ultimi 2, impone che la funzione sia suriettiva;
- il terzo impone il vincolo sugli archi di G: se esiste un arco in G tra due nodi, deve esistere il corrispondente arco relativo all'isomorfismo in F';
- gli ultimi 2 vincoli impongono che **f** mappi effettivamente tutti i nodi di G, ossia che l'isomorfismo sia completo (e non parziale).

Per la realizzazione dell'interfaccia grafica si è fatto uso del software Netbeans¹ e della libreria GraphStream².

¹ <https://netbeans.apache.org/>

² <https://graphstream-project.org/>

2. Interfaccia grafica

Per eseguire il software è necessario estrarre il file .zip, il quale contiene:

- l'eseguibile dlv per Windows
- l'eseguibile dlv per Linux
- Un file .jar

Una volta estratti nella stessa directory, per avviare il software basterà lanciare il file jar da riga di comando.

La schermata principale del programma è la seguente:

Generalized Node Deletion

File Code Help

Grafo F Grafo G K

Etichetta Nodo :

a Inserisci

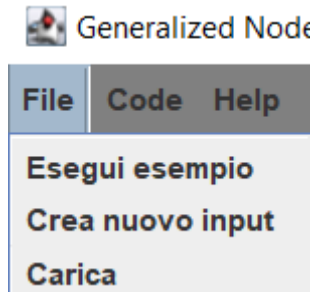
Arco Nodo1 to Nodo2 :

to Inserisci

ESEGUI

3. Il menu File

Il menu File contiene i seguenti sotto-menu:



3.1 Esegui esempio

Il menu Esegui esempio permette di eseguire direttamente un esempio di input per il problema dato. In particolare, l'esempio è fornito della seguente base di conoscenza per DLV:

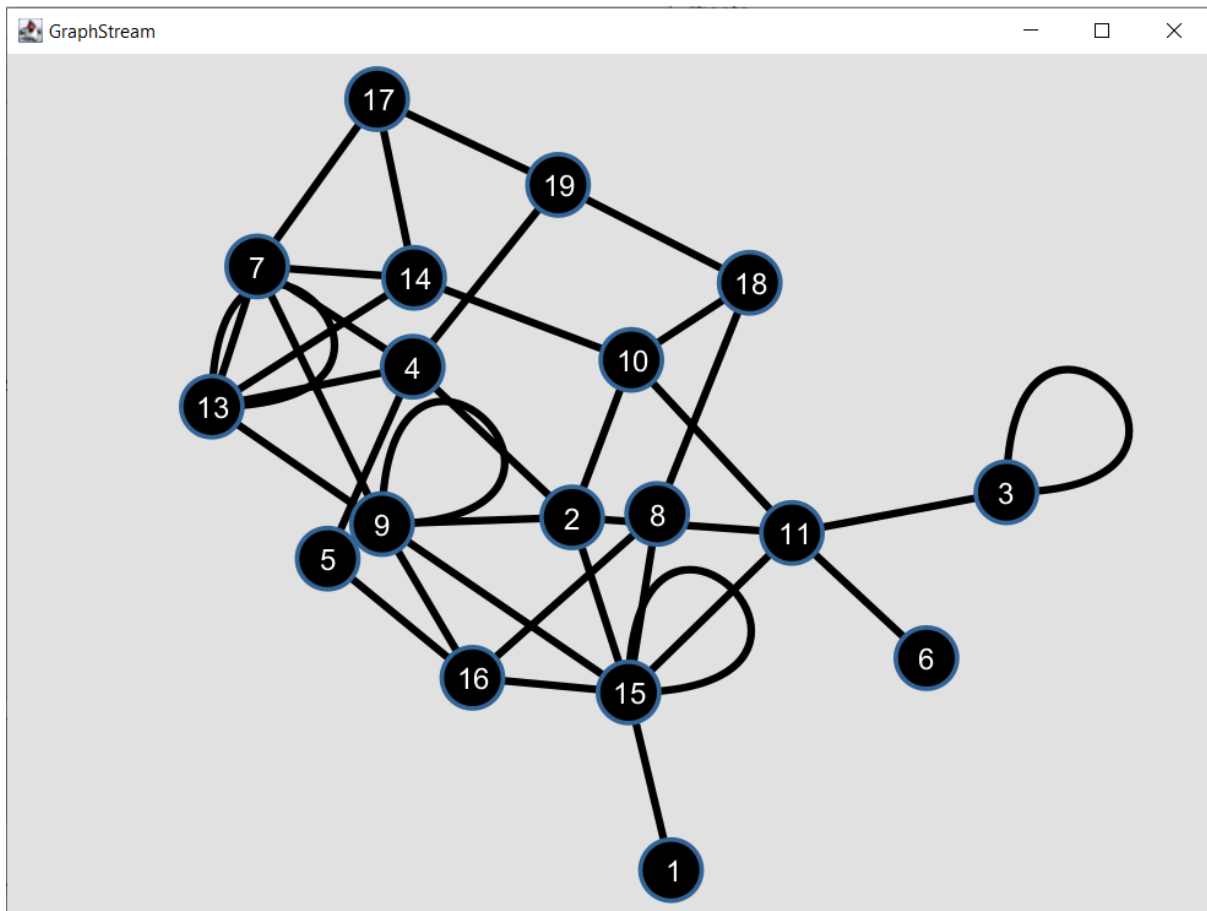
```
node_F(2).  
node_F(6).  
node_F(1).  
node_F(14).  
node_F(19).  
node_F(13).  
node_F(3).  
node_F(4).  
node_F(11).  
node_F(5).  
node_F(9).  
node_F(7).  
node_F(10).  
node_F(15).  
node_F(8).  
node_F(16).  
node_F(18).  
node_F(17).  
edge_F(10, 18).  
edge_F(10, 11).  
edge_F(14, 7).  
edge_F(14, 10).  
edge_F(5, 4).  
edge_F(9, 2).  
edge_F(9, 7).  
edge_F(9, 16).  
edge_F(9, 13).
```

```
edge_F(9, 9).
edge_F(16, 8).
edge_F(16, 15).
edge_F(16, 5).
edge_F(15, 1).
edge_F(15, 9).
edge_F(15, 15).
edge_F(2, 4).
edge_F(2, 10).
edge_F(2, 15).
edge_F(11, 2).
edge_F(11, 3).
edge_F(11, 15).
edge_F(8, 18).
edge_F(8, 15).
edge_F(13, 14).
edge_F(13, 7).
edge_F(13, 13).
edge_F(6, 11).
edge_F(3, 3).
edge_F(17, 14).
edge_F(17, 7).
edge_F(17, 19).
edge_F(4, 7).
edge_F(4, 13).
edge_F(19, 4).
edge_F(18, 19).
```

```
node_G(1).
node_G(2).
node_G(3).
node_G(4).
node_G(5).
node_G(6).
node_G(7).
node_G(8).
node_G(9).
edge_G(8, 8).
edge_G(8, 6).
edge_G(6, 2).
edge_G(1, 2).
edge_G(3, 8).
edge_G(7, 1).
edge_G(7, 8).
edge_G(2, 7).
edge_G(9, 3).
edge_G(9, 7).
edge_G(4, 3).
edge_G(4, 5).
edge_G(4, 8).
```


La KB codifica due grafi, F e G, e un valore per l'intero k .

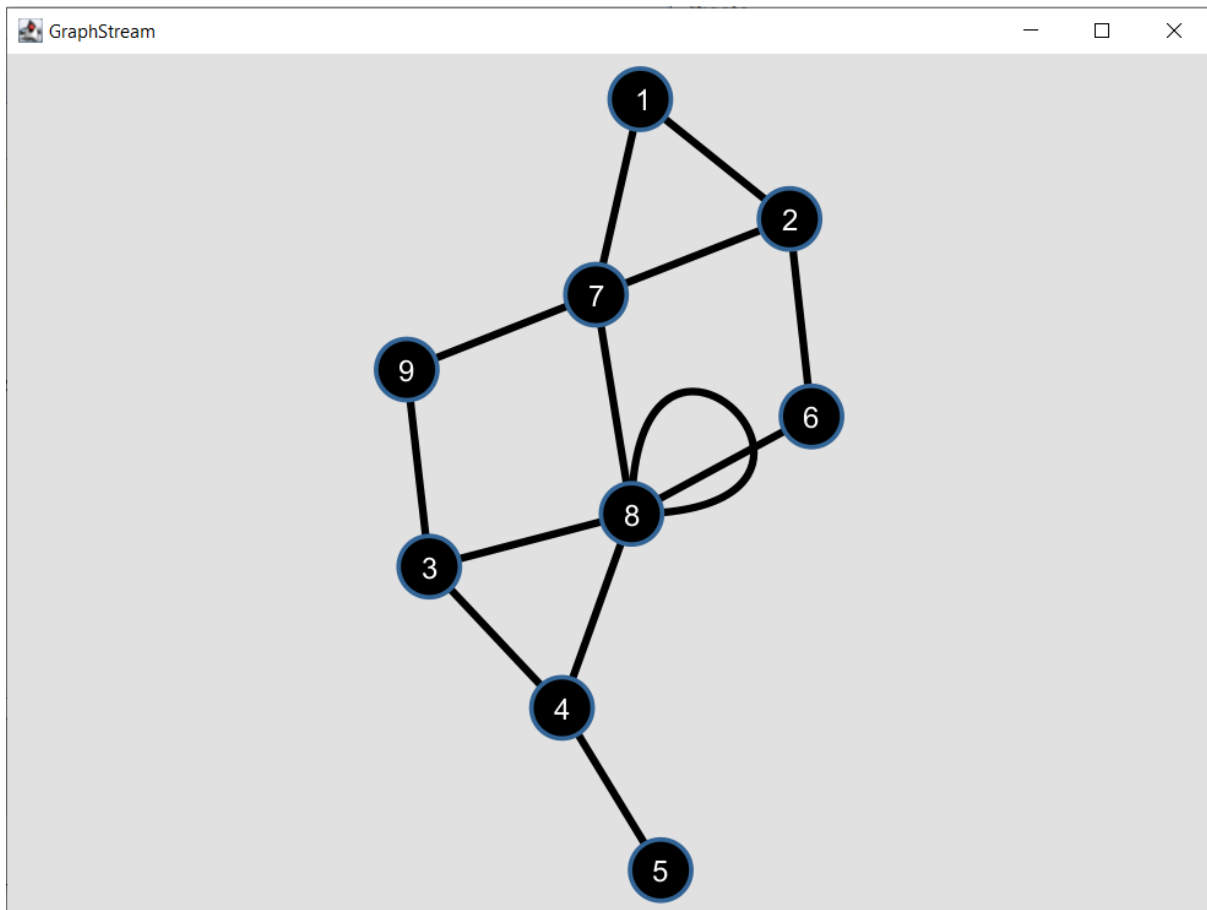
Il grafo F è il seguente:



Si tratta di un grafo composto da 18 nodi e 39 archi.

La libreria GraphStream permette la navigazione del grafo attraverso la tastiera tramite frecce (per movimento lungo le direzioni cardinali) oppure tramite zoom con i tasti Pag Up e Pag Down.

Il grafo G è invece il seguente:



Si tratta di un grafo con 9 nodi e 13 archi.

3.2 Crea nuovo input

Il menu si preoccupa di resettare lo stato dell'applicazione, di modo che sia possibile caricare manualmente un nuovo input.

3.3 Carica

Il menu permette di caricare una nuovo input per il problema da un file in formato testuale (con codifica UTF-8).

L'input deve essere conforme alla rappresentazione di grafi tramite archi.

A titolo di esempio è fornito un file chiamato `esempio.txt` che implementa in formato testuale la rappresentazione di F e G dell'esempio del gruppo.

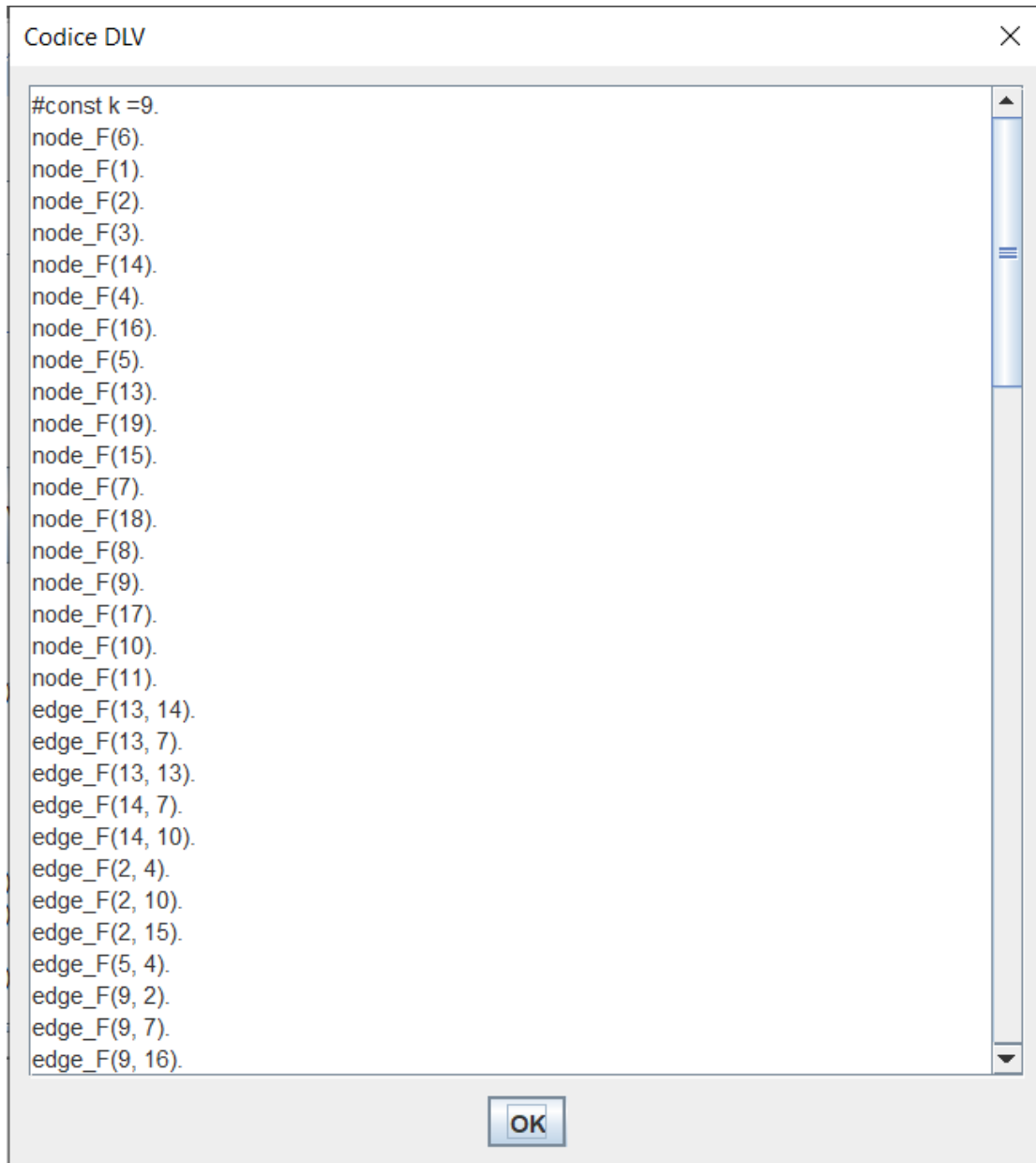
Il valore di k sarà invece impostato manualmente, e non da file, mediante interfaccia.

4. Il menu Code

Il menu permette di accedere al pulsante Codice DLV.

Il pulsante permette di visualizzare il codice DLV attualmente in attesa di essere messo in esecuzione tramite DLV, in base ai dati correntemente inseriti nell'applicazione per F, G e k.

A titolo di esempio, rispetto all'esempio proposto dal gruppo, il menu mostrerà la seguente schermata:



The screenshot shows a window titled "Codice DLV" with a close button (X) in the top right corner. The main area contains a list of DLV code statements, which are: `#const k =9.`, `node_F(6).`, `node_F(1).`, `node_F(2).`, `node_F(3).`, `node_F(14).`, `node_F(4).`, `node_F(16).`, `node_F(5).`, `node_F(13).`, `node_F(19).`, `node_F(15).`, `node_F(7).`, `node_F(18).`, `node_F(8).`, `node_F(9).`, `node_F(17).`, `node_F(10).`, `node_F(11).`, `edge_F(13, 14).`, `edge_F(13, 7).`, `edge_F(13, 13).`, `edge_F(14, 7).`, `edge_F(14, 10).`, `edge_F(2, 4).`, `edge_F(2, 10).`, `edge_F(2, 15).`, `edge_F(5, 4).`, `edge_F(9, 2).`, `edge_F(9, 7).`, and `edge_F(9, 16).`. A vertical scrollbar is on the right side of the text area. At the bottom center, there is an "OK" button.

```
#const k =9.
node_F(6).
node_F(1).
node_F(2).
node_F(3).
node_F(14).
node_F(4).
node_F(16).
node_F(5).
node_F(13).
node_F(19).
node_F(15).
node_F(7).
node_F(18).
node_F(8).
node_F(9).
node_F(17).
node_F(10).
node_F(11).
edge_F(13, 14).
edge_F(13, 7).
edge_F(13, 13).
edge_F(14, 7).
edge_F(14, 10).
edge_F(2, 4).
edge_F(2, 10).
edge_F(2, 15).
edge_F(5, 4).
edge_F(9, 2).
edge_F(9, 7).
edge_F(9, 16).
```

Attraverso le combinazioni di tastiera Ctrl + A e Ctrl + C sarà possibile copiare l'intero testo e incollarlo su Loide o altri esecutori per DLV.

5. Il menu Help

Il menu permette di accedere al pulsante About.

Il pulsante permette di visualizzare informazioni sul gruppo che ha realizzato il progetto.

6. Le schermate Grafo F e Grafo G

Le schermate mettono a disposizione la seguente interfaccia:

The screenshot shows a software interface with three tabs at the top: 'Grafo F' (selected), 'Grafo G', and 'K'. The main area is divided into two sections. The top section, titled 'Etichetta Nodo :', contains a text input field with the letter 'a' and a blue 'Inserisci' button to its right. To the right of this section is a large empty rectangular box. The bottom section, titled 'Arco Nodo1 to Nodo2 :', contains two dropdown menus separated by the text 'to'. Each dropdown menu is currently empty. To the right of these inputs is a blue 'Inserisci' button. To the right of this section is another large empty rectangular box.

Attraverso l'interfaccia, è possibile inserire nodi manualmente, con etichette composte da stringhe in minuscolo.

Una volta inserito almeno un nodo, sarà anche possibile inserire archi all'interno del grafo relativo alla schermata.

Nelle aree di testo della schermata, inoltre, verranno visualizzati i fatti corrispondenti ai nodi e agli archi del grafo in DLV rispetto allo stato dell'applicazione.

7. La schermata k

La schermata mette a disposizione la seguente interfaccia:

The interface consists of a top navigation bar with three tabs: 'Grafo F', 'Grafo G', and 'K'. The 'K' tab is currently selected. Below the tabs, there is a section for 'Valore K' with an input field containing the number '0' and a 'Conferma' button. A horizontal line separates this from the main content area. The main area is divided into two sections. The top section is titled 'Visualizzazione con GraphStream' and contains two buttons: 'Visualizza F' on the left and 'Visualizza G' on the right. The bottom section is titled 'Visualizzazione dei fatti con DLV' and contains two large, empty rectangular boxes. At the bottom right of the interface, there is a button labeled 'ESEGUI'.

Attraverso di essa è possibile:

- impostare il valore di k ;
- visualizzare F e G tramite la libreria GraphStream;
- visualizzare i fatti DLV relativi a F e G ;
- eseguire il programma presente nel menu Code -> Codice DLV tramite DLV.

Si fa presente che il bottone ESEGUI non sarà attivo a meno che non verranno specificati sia F , sia G , sia k .

k deve essere confermato tramite il bottone alla destra del campo di input.

Una volta cliccato il pulsante ESEGUI, l'applicazione sarà inutilizzabile fino a quando non sarà terminata la computazione DLV.

Una volta terminata, sarà mostrata una schermata che notifica l'esito.

Sono possibili due output nella GUI: TRUE e FALSE:

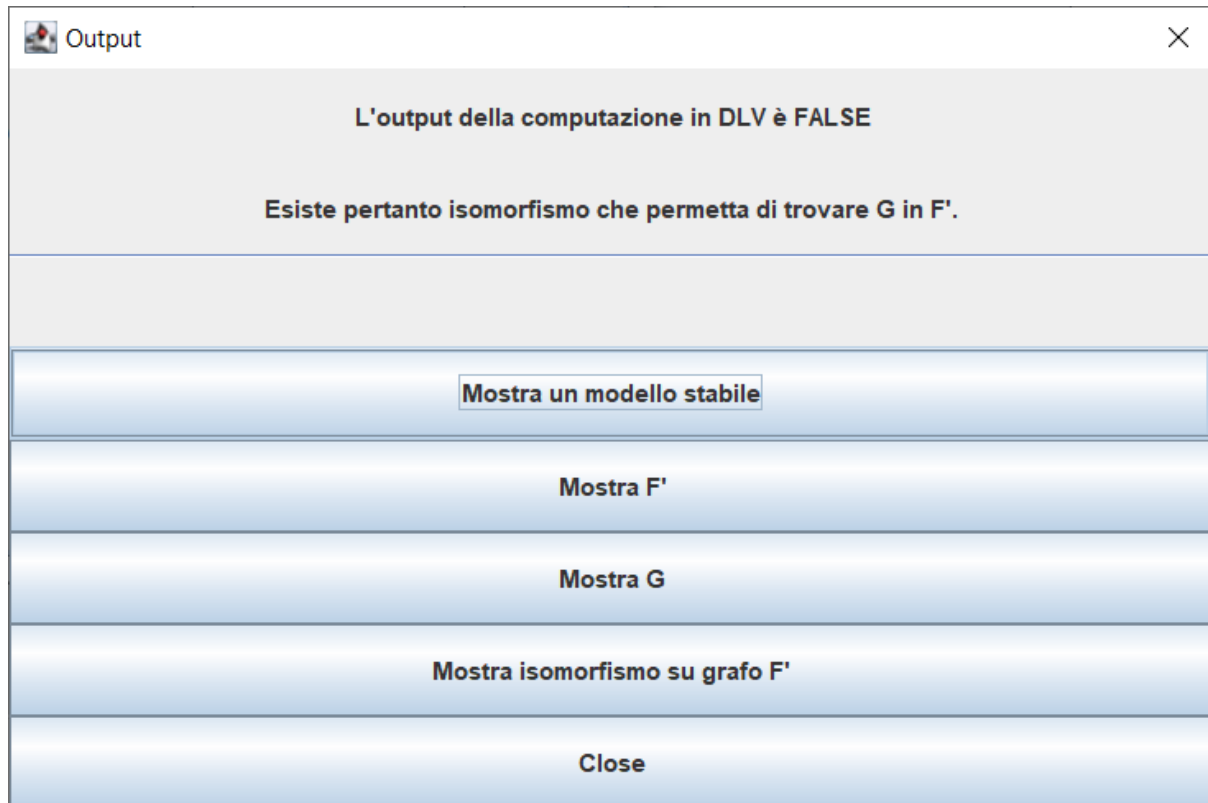
- TRUE, nel caso in cui l'istanza (F, G, k) è un'istanza SI per il problema Generalized Node Deletion (come specificato da traccia dei progetti).
- FALSE, nel caso in cui l'istanza (F, G, k) è un'istanza NO.

La risoluzione segue un approccio costruttivo, pertanto il programma restituirà FALSE se sarà riuscito a trovare un isomorfismo da un sottografo di F' a G .

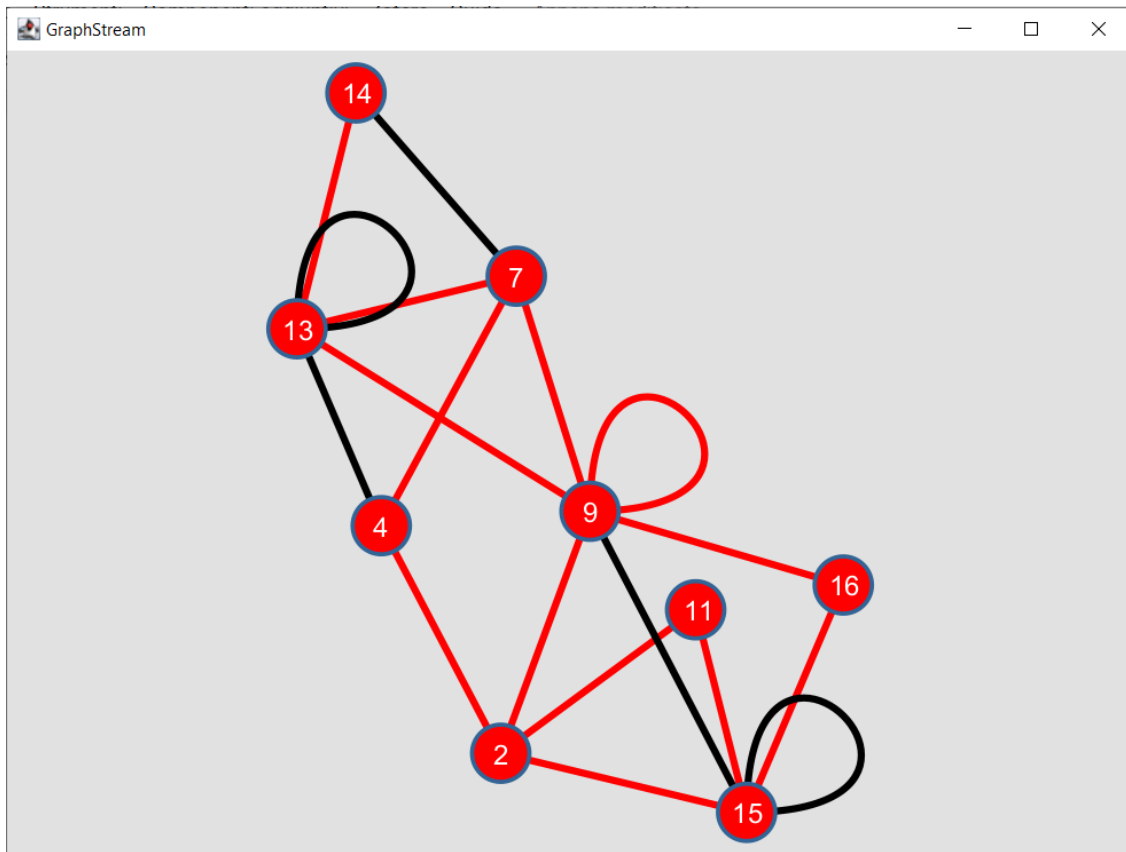
In tal caso, sarà possibile visualizzare l'isomorfismo tramite la libreria GraphStream.

Segue ora un esempio, relativo alla KB di prova caricata nel programma (per eseguirlo, si lancia il menu File -> Esegui esempio).

Lanciato l'esempio, terminata la computazione verrà visualizzata la seguente schermata:



Da cui è possibile, ad esempio, visualizzare un modello stabile (il primo restituito da DLV), oppure l'isomorfismo su F' , come segue:



Si fa notare che tale output si riferisce a un valore di k pari a 9. Ponendo $k=10$ non esisterà alcun isomorfismo, in quanto il grafo F' avrà un numero di nodi inferiore al numero di nodi in G :

