Manuale di Rocs

Tomaz Canabrava
Andreas Cord-Landwehr
Traduzione della documentazione: Valter Mura
Traduzione della documentazione: Paolo Zamponi
Traduzione della documentazione: Domenico Camasta
Traduzione della documentazione: Francesco Nigro
Traduzione di interfaccia e documentazione: Federico Zenith

Indice

1 Introduzione **6**

1.1 Obiettivi, destinatari e processi di lavoro 6

1.2 Rocs in breve 7

1.2.1 Grafi 7

1.2.2 Tipi di archi 7

1.2.3 Tipi di nodi 7

1.2.4 Proprietà 8

1.3 Esercitazione 8

1.3.1 Creazione del grafo 8

1.3.2 Creazione dei tipi di elementi 8

1.3.3 L'algoritmo 9

1.3.4 Eseguire l'algoritmo 9

2 L'interfaccia di Rocs **10**

2.1 Elementi principali dell'interfaccia 10

3 Scripting **12**

3.1 Eseguire algoritmi in Rocs 12

3.1.1 Controllare l'esecuzione degli script 12

3.1.2 Risultato script 12

3.1.3 API del motore di scripting 13

4 Importazione ed esportazione **14**

4.1 Condividere i progetti di Rocs 14

4.1.1 Importazione ed esportazione di grafi 14

4.1.1.1 Il formato di grafi banale 14

4.1.1.1.1 Specifica del formato 14

4.1.1.1.2 Esempio 15

4.1.1.2 Linguaggio DOT / Formato di grafi Graphviz 15

4.1.1.2.1 Funzionalità non supportate 15

4.1.1.2.2 Esempio 15

5 Disposizione dei grafi **16**
5.1 Disporre automaticamente i grafi in Rocs 16
5.1.1 Disposizione basata sulla forza 16
5.1.1.1 Disposizione ad albero radiale 17

6 Riconoscimenti e licenza **18**

Sommario

Rocs è uno strumento per la teoria dei grafi.

Capitolo 1

Introduzione

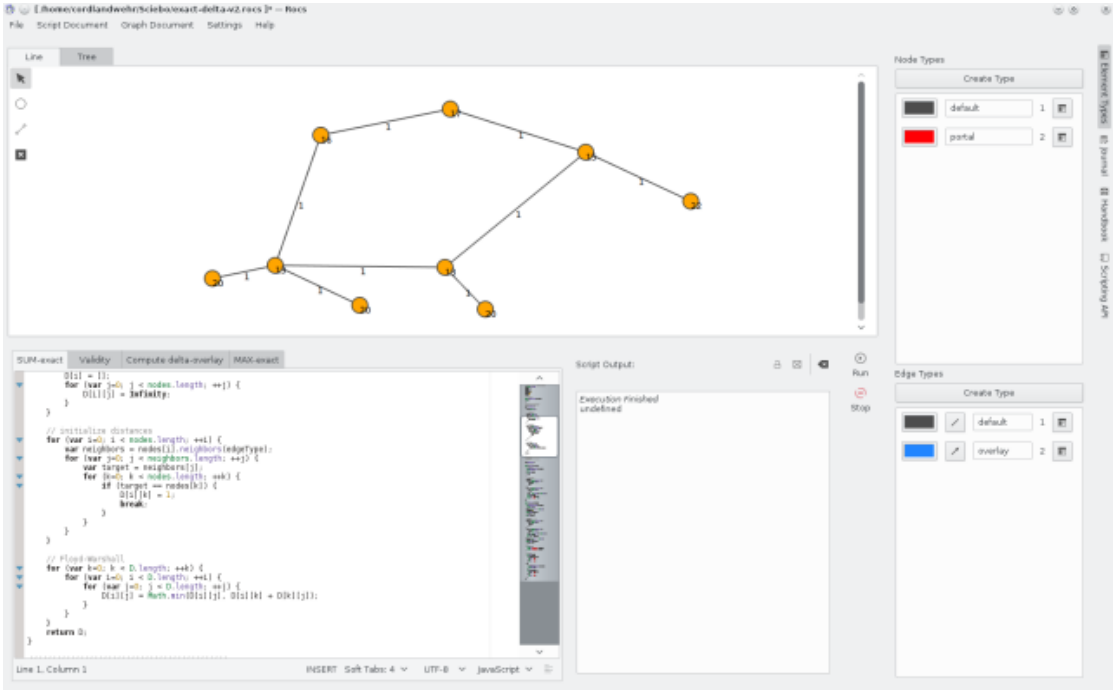
Questo capitolo fornisce una panoramica delle funzionalità principali e dei tipici processi di lavoro. Le parti più importanti sono Sezione 1.2 e capitolo 3, che insieme dovrebbero permettere al nuovo utente di iniziare ad usare subito Rocs.

1.1 Obiettivi, destinatari e processi di lavoro

Rocs è uno strumento per la teoria dei grafi pensato per chiunque sia interessato a tracciare e analizzare algoritmi per grafi. Ciò include esplicitamente:

- docenti che vogliano dimostrare gli algoritmi ai loro studenti,
- studenti e ricercatori che vogliono vedere e capire come funziona il loro algoritmo, e
- chiunque sia interessato alle strutture di dati e agli algoritmi.

Rocs fornisce a tutti loro un editor grafico per la creazione di grafi semplice da usare, un potente motore di scripting per eseguire gli algoritmi, molti strumenti di aiuto per le simulazioni e gli esperimenti, ed un esportatore di grafi. Generalmente Rocs viene utilizzato per creare grafi, sia a mano (trascinando, cioè, i nodi e gli archi sulla lavagna), sia usando uno dei generatori di grafi. Gli algoritmi per grafi possono poi essere implementati ed eseguiti sui grafi creati, e tutti i cambiamenti eseguiti dall'algoritmo sono immediatamente visibili nell'editor dei grafi.



1.2 Rocs in breve

Ogni sessione di Rocs è un progetto: quando si apre, Rocs crea un progetto vuoto, mentre quando si carica un progetto, esso diventa quello corrente. In questo contesto, un progetto consiste di *grafi*, *script/algoritmi*, e un *diario*.

1.2.1 Grafi

Un grafo rappresenta il contenuto di una lavagna dell'editor di grafi. Contiene le informazioni sui nodi definiti dall'utente e sui tipi di arco, sulle loro proprietà e sui nodi e gli archi già creati. Rocs interpreta l'insieme di tutti i nodi e gli archi di un grafo per formare un grafo (non necessariamente connesso). Tutto ciò che appartiene ad un grafo è accessibile dal motore di script per mezzo dell'oggetto globale **`Document`**.

1.2.2 Tipi di archi

In alcuni scenari i grafi sono formati da diversi tipi di archi (ad esempio un grafo non direzionale con tre archi computato da un algoritmo di ricerca in ampiezza) che dovrebbero essere trattati e visualizzati in maniera diversa. Pertanto è possibile definire arbitrariamente altri tipi di archi oltre al tipo predefinito. Ciascun tipo di arco ha una sua rappresentazione visuale individuale e delle proprietà dinamiche, inoltre può essere impostato come diretto o indiretto. L'interfaccia di scripting fornisce un metodo comodo per accedere solo agli archi di un tipo specifico.

1.2.3 Tipi di nodi

Analogamente ai tipi di archi, si possono definire diversi tipi di nodi di un grafo (ad esempio per dare ad alcuni di questi un ruolo speciale). Ciascun tipo di nodo ha la sua rappresentazione visuale e le sue proprietà dinamiche.

1.2.4 Proprietà

Ogni elemento (nodo o arco) può avere delle proprietà, che devono essere impostate nel corrispondente tipo di nodo o di arco. Le proprietà sono identificate e sono accessibili per nome, e possono contenere qualsiasi valore. Per crearne di nuove, o per cambiare quelle esistenti, si può usare la barra laterale **Tipi di elemento** e il pulsante  **Proprietà** per aprire la finestra delle proprietà.

Si può usare il motore di scripting per aver accesso alle proprietà registrate e cambiarne i valori. Nell'esempio seguente si assume che la proprietà 'weight' sia registrata per il tipo di arco predefinito.

```
var nodes = Document.nodes()
for (var i = 0; i < nodes.length; ++i){
    nodes[i].weight = i;
}
for (var i = 0; i < nodes.length; ++i){
    Console.log("weight of node " + i + ": " + nodes[i].weight);
}
```

1.3 Esercitazione

In questa sezione vogliamo elaborare un progetto di esempio per esplorare alcune tra le più importanti funzioni di Rocs. L'obiettivo è quello di creare un grafo e uno script che illustrino un semplice algoritmo di approssimazione di fattore 2, per trattare un problema di *copertura minima dei vertici*. Questo problema può essere descritto come la ricerca del più piccolo sottoinsieme C di vertici in un grafo tale che ogni arco ha almeno un'estremità in C. Si tratta di un problema NP-completo e il nostro obiettivo è quello di trovare un'approssimazione di fattore 2 attraverso la computazione di una corrispondenza all'interno di un grafo dato.

Il nostro obiettivo è di visualizzare la corrispondenza tra l'accoppiamento e la copertura minima dei vertici. Per ottenere questo obiettivo dobbiamo specificare due tipi di archi, uno per mostrare gli archi accoppiati e l'altro per mostrare gli archi 'ordinari'; dobbiamo specificare inoltre due tipi di nodi, che useremo per distinguere i nodi contenuti in C da quelli non contenuti in C.

1.3.1 Creazione del grafo

Per creare il grafo usiamo il generatore di grafi fornito da Rocs, che si trova in **Grafo** → **Strumenti** → **Genera grafo**. In questo modo selezioniamo un 'grafo casuale' composto di 30 nodi, 90 archi, e con seme di generazione 1 (il seme determina la casualità del grafo; partendo dallo stesso seme si riproduce lo stesso grafo).

1.3.2 Creazione dei tipi di elementi

Usiamo **Tipi di elementi** per creare sia un secondo tipo di nodo sia di arco. Dobbiamo aprire la finestra delle proprietà di entrambi questi due nuovi tipi facendo clic sul rispettivo pulsante delle  **Proprietà** per impostare gli ID a `2`; dobbiamo anche cambiare il colore degli elementi di questi due nuovi tipi (per distinguerli da quelli predefiniti). Infine dobbiamo impostare tutti i tipi di archi come bidirezionali, e gli ID del tipo predefinito a `1`.

1.3.3 L'algoritmo

Al minimo è necessario implementare l'algoritmo di approssimazione. Per farlo si utilizza la seguente implementazione:

```
for (var i=0; i < Document.nodes.length; i++) {
    Document.nodes[i].type = 1;
}
for (var i=0; i < Document.edges.length; i++) {
    Document.edges[i].type = 1;
}

var E = Document.edges(); // insieme degli archi non esaminati
var C = new Array();      // archi accoppiati
while (E.length
> 0) {
    var e = E[0];         // prende il primo arco e={u,v}
    var u = e.from();
    var v = e.to();
    e.type = 2;           // imposta l'arco come arco accoppiato
    E.shift();            // elimina l'arco e (i.e., E[0]) dalla lista ↩
        degli archi
    C.push(u);            // aggiunge u a C
    C.push(v);            // aggiunge v a C

    // segna u,v come nodi di C
    u.type = 2;
    v.type = 2;

    // rimuove da E tutti gli archi incidenti a u o v
    var adjacent = u.edges();
    for (var i=0; i < adjacent.length; i++) {
        var index = E.indexOf(adjacent[i]); // trova l'indice
        if (index != -1) {
            E.splice(index, 1); // elimina l'indice dopo averlo trovato
        }
    }
    var adjacent = v.edges();
    for (var i=0; i < adjacent.length; i++) {
        var index = E.indexOf(adjacent[i]); // trova l'indice
        if (index != -1) {
            E.splice(index, 1); // elimina l'indice dopo averlo trovato
        }
    }
}
Console.log("Vertex Cover contains " + C.length + " nodes.");
```

1.3.4 Eseguire l'algoritmo

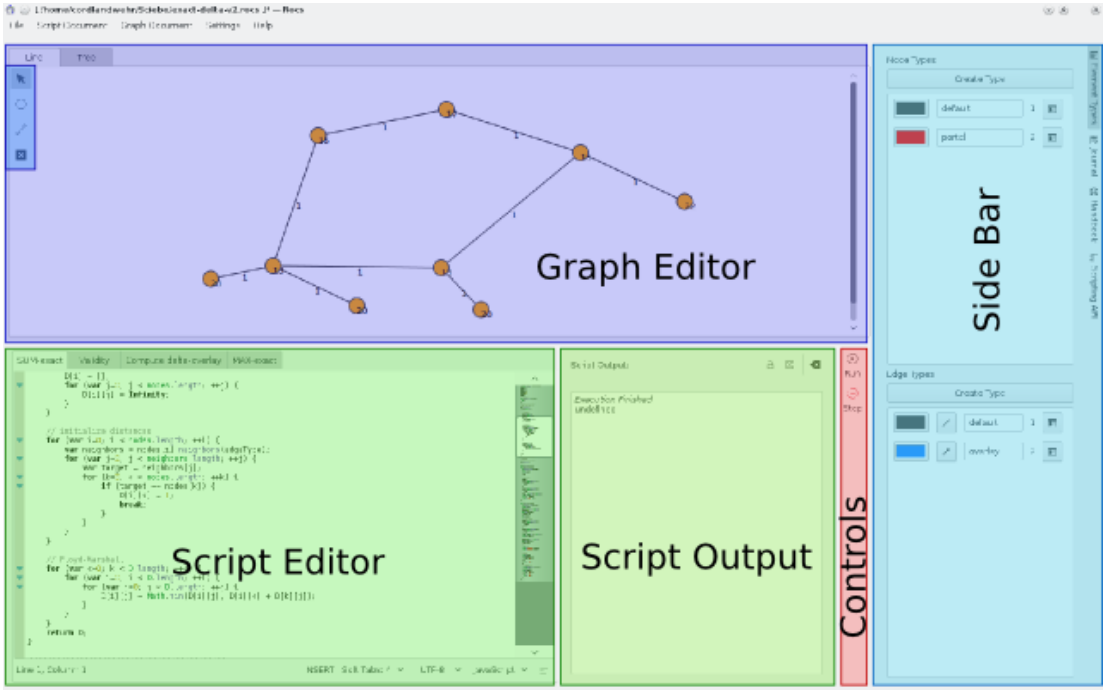
L'algoritmo può essere eseguito facendo clic sul pulsante  **Esegui** nel pannello di controllo dello script.

Capitolo 2

L'interfaccia di Rocs

2.1 Elementi principali dell'interfaccia

L'interfaccia si divide in diverse parti logiche, come presentato nella schermata sotto.



Editor dei grafi

L'editor fornisce una lavagna su cui si possono posizionare i nodi e gli archi; un doppio clic su un elemento qualsiasi apre il corrispondente menu delle proprietà. Per creare e modificare i grafi puoi usare gli strumenti delle *schede della Barra laterale*.

Strumenti disponibili:

- Nella parte superiore sinistra del riquadro troverai le icone delle azioni seguenti. Fare clic su un'azione implica che il puntatore del mouse l'applica alla lavagna dell'editor dei grafi:

-  **Seleziona e sposta**: per selezionare gli elementi, fai clic su un'area vuota della lavagna, tieni il mouse premuto e disegna un rettangolo che contenga alcuni elementi di dati o archi per selezionarli, oppure fai direttamente clic su un elemento non selezionato per selezionarlo. Se fai clic su uno o più elementi selezionati, li puoi spostare tenendo il mouse premuto e spostando il mouse. È possibile spostare gli elementi selezionati anche con i tasti freccia.
-  **Aggiungi un nodo**: facendo clic su un punto qualsiasi della lavagna dell'editor dei grafi si crea un nuovo dato che appartiene alla struttura dati attualmente selezionata. Tenendo premuto a lungo il tasto del mouse appare un menu attraverso il quale il tipo del dato appena creato può essere selezionato (ciò richiede l'esistenza di tipi di dati differenti).
-  **Crea un arco**: fai clic su un dato, tieni il mouse premuto, e traccia una linea a un altro dato a cui l'arco punterà. Questa azione ha successo solo se il grafo attuale permette l'aggiunta di questo arco (per esempio, in un grafo non direzionale non è possibile aggiungere archi multipli tra due dati). Mantenendo la pressione del puntatore sul tasto appare un menu da cui è possibile selezionare il tipo a cui assegnare l'arco appena creato (solo se esistono più tipi di archi).
-  **Elimina elemento**: fai clic su un elemento per eliminarlo. Se elimini un nodo, verranno eliminati anche tutti gli archi adiacenti.

Barra laterale

A destra puoi trovare la barra laterale, che contiene diversi strumenti per il processo di lavoro:

- **Tipi di elemento**: questo oggetto ti dà accesso immediato ai tipi di archi e ai nodi disponibili.
- **Diario**: ogni progetto ha il suo diario, che può essere usato, per esempio, per annotare i compiti, i risultati e le osservazioni.
- **API di scripting**: puoi aprire questo oggetto per avere accesso immediato alla documentazione degli script.

Editor di script

In questo editor di testo puoi scrivere algoritmi, come spiegato in dettaglio in capitolo 3. Puoi lavorare su diversi script allo stesso tempo usando diverse schede.

Risultato script:

Quest'area di testo mostra o informazioni di debug o l'output dello script dell'algoritmo, a seconda dell'impostazione attiva in alto. Se lo script produce un errore, viene automaticamente selezionato l'output di debug.

Controlli

Qui puoi trovare i controlli per l'esecuzione degli script. Puoi eseguire lo script attualmente aperto nell'editor degli script premendo il pulsante  **Esegui**. È possibile interrompere lo script mentre è in esecuzione premendo il pulsante  **Interrompi**.

Capitolo 3

Scripting

3.1 Eseguire algoritmi in Rocs

Rocs usa internamente il motore JavaScript QtScript. Questo significa che tutti gli algoritmi che vengono implementati devono usare JavaScript. Di seguito spiegheremo come accedere agli elementi di un grafo e cambiarli usando il motore di scripting. È importante notare che i cambiamenti effettuati dal motore di scripting si riflettono direttamente sulle proprietà dell'editor di elementi del grafo.

3.1.1 Controllare l'esecuzione degli script

Ci sono diversi modi di esecuzione per gli algoritmi:

-  **Esegui**: fa partire lo script e lo esegue fino al termine.
-  **Interrompi**: ferma l'esecuzione di uno script (disponibile solo mentre uno script viene eseguito).

3.1.2 Risultato script

Durante l'esecuzione di un algoritmo, gli output del debug e del programma sono mostrati nell'*output di debug e degli script*. Se il motore di scripting individua un errore nella sintassi dello script lo mostra anche come messaggio di debug. Tutti i messaggi del programma vengono mostrati (in grassetto) insieme all'output di debug.

Puoi controllare il testo mostrato nell'output dello script attraverso le seguenti funzioni:

```
Console.log(string message);            // mostra il messaggio come output  ↩
    dello script
    Console.debug(string message);          // mostra il messaggio come  ↩
        output di debug
    Console.error(string message);          // mostra il messaggio come  ↩
        output di errore
```

3.1.3 API del motore di scripting

Ciascuna delle diverse parti di Rocs fornisce un elemento statico che è accessibile dal motore di scripting. Questi sono:

- **`Documento`** per il grafo
- **`Console`** per la console di registrazione dell'output

. Per l'uso esplicito delle API e per il metodo di riferimento consulta l'aiuto in linea nella barra laterale di Rocs.

Capitolo 4

Importazione ed esportazione

4.1 Condividere i progetti di Rocs

I progetti di Rocs possono essere importati ed esportati come archivi `.tar.gz`, che possono essere utilizzati per condividere progetti. Si possono importare o esportare progetti rispettivamente attraverso gli elementi di menu **Grafo** → **Importa grafo** e **Grafo** → **Esporta grafo come...**.

4.1.1 Importazione ed esportazione di grafi

Rocs attualmente supporta l'importazione e l'esportazione dei seguenti formati di file:

- File DOT, altrimenti noti come GraphViz
- File GML
- File in formato di grafi banale
- Formato di linguaggio a marcatori Keyhole

4.1.1.1 Il formato di grafi banale

Il *formato di grafi banale* (TGF) è un semplice formato di file testuale per descrivere i grafi. Un file TGF consiste di un elenco di definizioni dei nodi che fanno corrispondere i loro identificativi a delle etichette, seguite da un elenco degli archi. In questo formato è possibile avere una sola etichetta per nodo e un valore per arco. Rocs interpreta i grafi importati come grafi non direzionali. I grafi esportati conterranno due archi per connessione se queste sono bidirezionali.

4.1.1.1.1 Specifica del formato

- Il file inizia con una lista di nodi (un nodo per linea) seguito da una linea col solo carattere '#', a cui fa seguito una lista di archi (un arco per linea).
- Un nodo consiste di un intero (identificatore) seguito da uno spazio, seguito da una stringa qualsiasi.
- Un arco è costituito da due interi (gli identificativi) separati da uno spazio, seguiti da uno spazio, seguito da una stringa qualsiasi. Si presume che l'arco direzionale punti dal primo identificativo al secondo.

4.1.1.1.2 Esempio

```
1 nodo iniziale
2 trasmissione
3 termine
#
1 2 blu
2 1 rosso
2 3 verde
```

4.1.1.2 Linguaggio DOT / Formato di grafi Graphviz

Il linguaggio DOT è un linguaggio descrittivo in testo semplice che permette sia una rappresentazione leggibile dei grafi, sia un'elaborazione efficiente da parte di un programma di disposizione dei grafi. DOT è il formato di file predefinito per la serie di programmi di visualizzazione Graphviz, ma è usato anche da altri strumenti. Le estensioni più comuni dei file DOT sono `.gv` e `.dot`.

4.1.1.2.1 Funzionalità non supportate

Rocs può elaborare ogni file grafo che contenga grafi specificati secondo il linguaggio di specifica DOT[1]. Il supporto alle funzionalità del linguaggio è completo, salvo le seguenti eccezioni:

- Sottografi: a causa della mancanza del concetto di sottografo in Rocs, i sottografi sono importati come insiemi di elementi e connessioni. In particolare, le connessioni da o verso sottografi non sono importate.
- Attributi HTML e XML: gli attributi (come etichette) che contengono sintassi HTML o XML vengono letti senza modifica. In particolare, da questi attributi non viene letta nessuna specificazione di carattere o stile.

4.1.1.2.2 Esempio

```
digraph myGraph {
    a -> b -> c;
    b -> d;
}
```

[1] https://graphviz.org/doc/info/lang.html

Capitolo 5

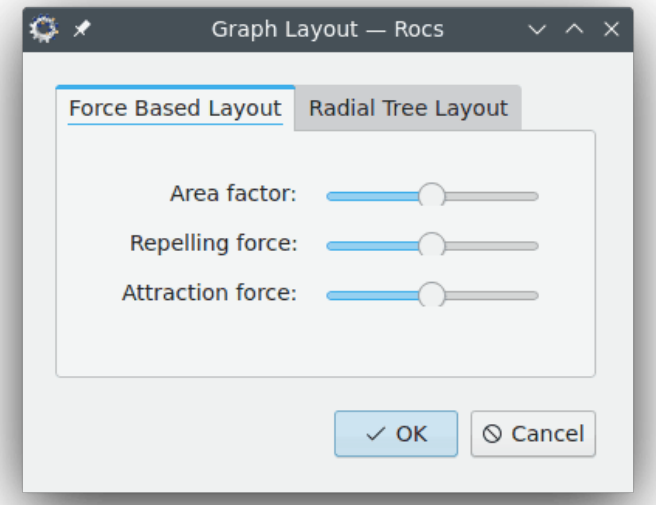
Disposizione dei grafi

5.1 Disporre automaticamente i grafi in Rocs

Rocs può disporre automaticamente i grafi. Lo strumento per la disposizione dei grafi di Rocs si trova nel menu principale in **Grafo** → **Strumenti** → **Disposizione dei grafi**. Sono applicabili due diversi algoritmi di disposizione: Disposizione basata sulla forza e Disposizione ad albero radiale. Per applicarne una, seleziona la scheda corrispondente dello strumento di disposizione dei grafi, scegli i parametri che preferisci ed esegui l'algoritmo premendo il pulsante **OK**. I dettagli specifici di ciascun algoritmo di disposizione sono spiegati nelle sezioni seguenti.

5.1.1 Disposizione basata sulla forza

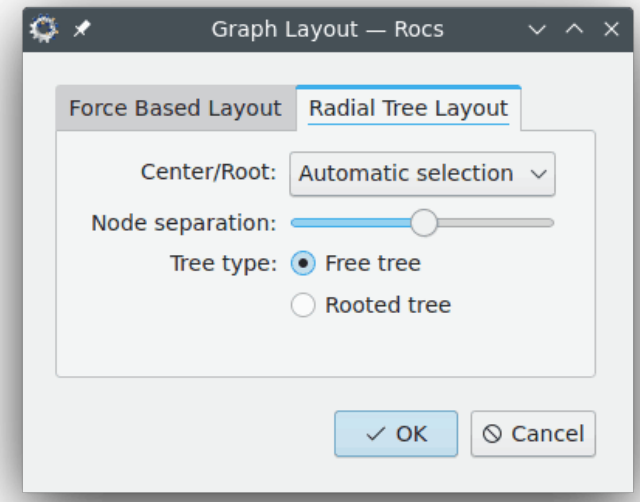
La Disposizione basata sulla forza è applicabile a qualsiasi grafo. Come è facile notare, questo algoritmo simula le forze che agiscono in ciascun nodo. Sono presenti forze di repulsione tra coppie di nodi e forze di attrazione tra coppie di nodi che sono adiacenti. L'ordine di grandezza di queste forze può essere specificata spostando i relativi cursori nell'interfaccia utente.



Un altro parametro controllabile è il «Fattore area». Questo parametro controlla come i nodi si diffondono. Le disposizioni generate con valori elevati del Fattore area tendono ad avere grandi distanze tra i nodi.

5.1.1.1 Disposizione ad albero radiale

La Disposizione ad albero radiale è applicabile solo agli alberi. Qualsiasi tentativo di applicare questo algoritmo ad altri tipi di grafo genererà un messaggio di errore. I parametri per la Disposizione ad albero radiale è selezionabile tramite l'interfaccia utente fornita.



Il parametro a tre tipi seleziona tra una disposizione ad albero libero e una ad albero con radice. Nella disposizione ad albero libero i nodi vengono sistemati liberamente senza alcuna apparente gerarchia. Nella disposizione ad albero con radice, il nodo radice è posizionato in cima all'albero e gli alberi secondari si diffondono sotto di esso, dando l'idea di una gerarchia tra i nodi.

Il parametro Centro/Radice definisce quale nodo verrà utilizzato come radice per la disposizione ad albero con radice o come centro per la disposizione ad albero libero. The center of a free tree layout is the first node to be placed by the algorithm. All other nodes are placed on circles centered at the center node. A center/root can be selected automatically by the layout algorithm.

Il parametro di separazione del nodo controlla la distanza tra i nodi. Con l'aumento del valore del parametro si aumenterà la distanza tra i nodi. Allo stesso modo, con la riduzione del suo valore si ridurrà la distanza tra i nodi.

Capitolo 6

Riconoscimenti e licenza

Rocs

Copyright del programma:

- Copyright 2008 di Ugo Sangiori (ugorox@gmail.com)
- Copyright 2008-2012 di Tomaz Canabrava (tcanabrava@kde.org)
- Copyright 2008-2012 di Wagner Reck (wagner.reck@gmail.com)
- Copyright 2011-2015 Andreas Cord-Landwehr (cordlandwehr AT kde.org)

Copyright della documentazione:

- Copyright della documentazione 2009 di Anne-Marie Mahfouf annma@kde.org
- Copyright della documentazione 2009 di Tomaz Canabrava (tcanabrava@kde.org)
- Copyright della documentazione 2011-2015 Andreas Cord-Landwehr (cordlandwehr AT kde.org)

Traduzione a cura di Valter Mura, Paolo Zamponi, Francesco Nigro e Federico Zenithvaltermura@gmail.com