Manual do Rocs

Tomaz Canabrava
Andreas Cord-Landwehr
Tradução: Marcus Gama
Tradução: André Marcelo Alvarenga

Conteúdo

Resumo

Rocs é a ferramenta da teoria dos grafos do KDE.

Capítulo 1

Introdução

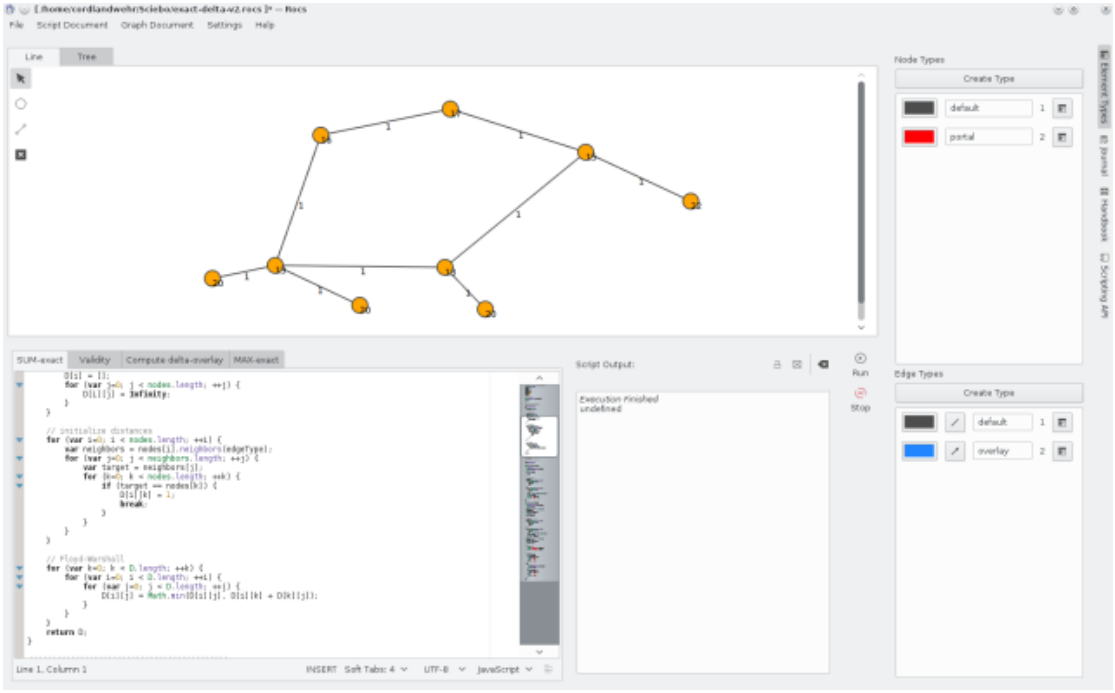
Neste capítulo apresentaremos as funcionalidades e fluxos de trabalho básicos. As partes mais importantes são Seção 1.2 e capítulo 3, que, em conjunto, permitem que novos usuários possam começar a usar o Rocs.

1.1 Objetivos, público-alvo e funcionamento

O Rocs é uma ferramenta da Teoria dos Grafos para todos os interessados no desenho e estudo de algoritmos de grafos. Em particular, estes são

- os professores, que poderão demonstrar algoritmos aos seus alunos,
- os alunos e pesquisadores, que queiram ver como funcionam os seus algoritmos e
- todos que estejam interessados em estruturas de dados e algoritmos.

Para todos eles, o Rocs oferece um editor gráfico fácil de usar para criar grafos, um mecanismo de criação de scripts poderoso para executar algoritmos e diversas ferramentas para ajudar nas simulações, experiências e exportações de grafos. A forma típica de usar o Rocs é criar um grafo, seja à mão (i.e., arrastando nós e arestas para o quadro) ou usando um dos geradores de grafos. Os algoritmos de grafos poderão então ser implementados e executados no grafo criado, assim como todas as alterações feitas pelo algoritmo, serão visíveis imediatamente no editor do grafo.



1.2 O Rocs em resumo

Cada sessão do Rocs é um projeto: Ao abrir o Rocs, um projeto vazio será criado e ao carregar algum projeto existente, ele se tornará o projeto atual. Um projeto é composto de *documentos de grafos*, *scripts/algoritmos* e *registro*.

1.2.1 Documentos de grafo

Um documento de grafo representa o conteúdo de um quadro no editor de grafos. Contém informações sobre os tipos de nós e arestas definidos pelo usuário, suas propriedades e os nós e arestas já criados. Isto é, o Rocs compreende o conjunto de todos os nós e arestas de um documento de grafo para formar um grafo (não necessariamente conectado). Tudo o que pertence ao documento de grafo fica acessível pelo mecanismo de criação de scripts através do objeto global **`Document`**.

1.2.2 Tipos de borda

Em alguns cenários, os grafos consistem em diversos tipos de arestas (p.ex., um grafo não-direcional mais as arestas da árvore calculadas por um algoritmo de pesquisa primeiro-em-largura) que deverá ser tratado e apresentado de forma diferente. Para isso, além de um tipo de aresta padrão, você poderá definir outros tipos de arestas arbitrários. Cada tipo de aresta tem a sua própria representação visual, propriedades dinâmicas e pode ser definido como direcionado ou não-direcionado. A interface de criação de scripts oferece métodos de conveniência para acessar de forma específica apenas as arestas de determinados tipos.

1.2.3 Tipos de nós

De forma análoga aos tipos de arestas, você poderá definir diferentes tipos de nós de um grafo (p.ex., para indicar algumas regras especiais aos nós). Cada tipo de nó tem sua própria representação visual e propriedades dinâmicas.

1.2.4 Propriedades

Cada elemento (nó ou aresta) pode ter propriedades. Essas propriedades devem ser configuradas no tipo de nó ou aresta correspondente. As propriedades são identificadas e acessadas pelos seus nomes e poderão conter qualquer valor. Para criar novas propriedades ou alterar as existentes, use a barra lateral **Tipos de elementos** e o botão **Propriedades** para abrir a janela de propriedades.

Você também pode usar o mecanismo de criação de scripts para acessar as propriedades registradas e alterar os seus valores. No exemplo a seguir, assume-se que a propriedade ″peso″ está registrada para o tipo de aresta padrão.

```
var nos = Document.nodes()
for (var i = 0; i < nos.length; ++i){
    nos[i].peso = i;
}
for (var i = 0; i < nos.length; ++i){
    Console.log("peso do nó " + i + ": " + nos[i].peso);
}
```

1.3 Tutorial

Nesta seção criaremos um projeto de exemplo para explorar algumas das funções mais importantes do Rocs. O objetivo é criar um grafo e um script que ilustre um algoritmo de 2 aproximações para o problema da *cobertura mínima de vértices*. Esse problema consiste em encontrar um subconjunto de nós do grafo C com um tamanho mínimo, de modo que cada aresta do grafo esteja conectada a pelo menos um nó do C. Este problema é conhecido por ser NP-difícil e queremos ilustrar como encontrar uma aproximação com um fator 2 computando uma correspondência no grafo dado.

Nosso objetivo é visualizar o relacionamento da ocorrência e da cobertura mínima de vértices. Para tal, queremos mostrar dois tipos de arestas, um para mostrar as arestas correspondentes e outro para mostrar as ″normais″, assim como dois tipos de nós que usamos para distinguir os nós contidos em C e os não-contidos.

1.3.1 Gerando o grafo

Para criação do grafo, usamos o gerador de grafos padrão oferecido pelo Rocs. Ele pode ser encontrado no menu principal em **Documento do grafo** → **Ferramentas** → **Gerar o grafo**. Aí, selecionamos um **Grafo aleatório** com 30 nós, 90 arestas e com uma raiz 1 (a raiz é o valor de referência inicial para o gerador de grafos aleatórios; a utilização da mesma raiz várias vezes origina grafos iguais).

1.3.2 Criação dos tipos de elementos

Usamos os **Tipos de elementos** e criamos um segundo tipo de nó, assim como um segundo tipo de aresta. Para os novos tipos, abrimos a janela de propriedades, usando os respectivos botões de **Propriedades** e definimos os IDs como `2`. Além disso, mudamos as cores dos elementos desses dois novos tipos (para distingui-los dos tipos padrão). Finalmente, definimos todos os tipos de arestas como sendo bidirecionais e os IDs dos tipos padrão como `1`.

1.3.3 O Algoritmo

Por último, temos de implementar o algoritmo de aproximação. Para isso, iremos usar a seguinte implementação:

```
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(); // define o conjunto de arestas não processadas
var C = new Array();      // arestas correspondentes
while (E.length
> 0) {
    var e = E[0];         // escolhemos primeiro a aresta e={u,v}
    var u = e.from();
    var v = e.to();
    e.type = 2;           // define a aresta como sendo correspondente
    E.shift();            // remove a 'e' (i.e., E[0]) da lista de arestas
    C.push(u);            // adiciona a 'u' ao C
    C.push(v);            // adiciona a 'v' ao C

    // marcar o u,v como nós no C
    u.type = 2;
    v.type = 2;

    // remove do E todas as arestas que incidem em 'u' ou 'v'
    var adjacente = u.edges();
    for (var i=0; i < adjacente.length; i++) {
        var indice = E.indexOf(adjacente[i]); // localiza o índice
        if (indice != -1) {
            E.splice(indice, 1); // remove se já tiver encontrado
        }
    }
    var adjacente = v.edges();
    for (var i=0; i < adjacente.length; i++) {
        var indice = E.indexOf(adjacente[i]); // descobrir o índice
        if (indice != -1) {
            E.splice(indice, 1); // remover se já tiver encontrado
        }
    }
}
Console.log("A cobertura de vértices contém " + C.length + " nós.");
```

1.3.4 Executar o Algoritmo

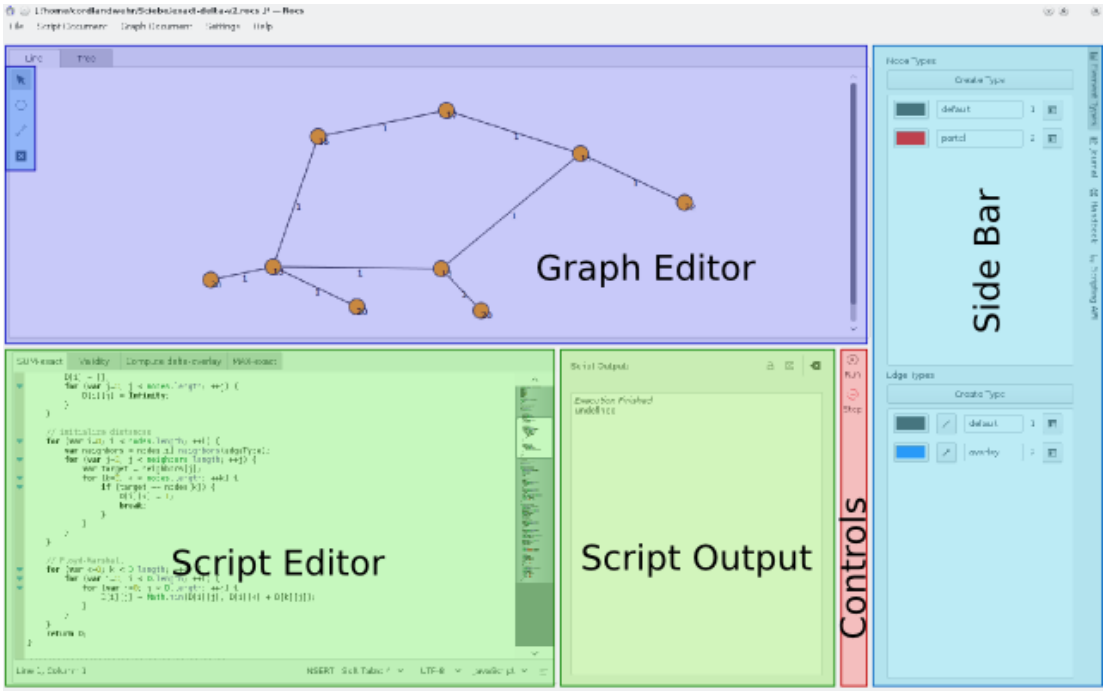
O algoritmo pode ser executado com o botão **Executar** no painel de controle de scripts.

Capítulo 2

A interface do usuário do Rocs

2.1 Elementos principais da interface do usuário

A interface do usuário está dividida em várias partes lógicas, como aparece na imagem abaixo.



Editor de grafos

O editor oferece um quadro onde os nós e arestas podem ser colocados. Ao clicar duas vezes em qualquer um dos elementos, será aberto um menu de propriedades correspondente. Você pode usar as ferramentas da *Barra de Edição do Grafo* para criar e modificar os grafos.

Barra de ferramentas do editor de grafos

A barra oferece as ferramentas **Criar nó** ou **Criar aresta**, são usadas para criação de novos elementos no quadro. Observe que a barra de ferramentas extra, para seleção do tipo de nó ou aresta respectivo, fica visível se uma dessas ferramentas for selecionada. As ferramentas também são usadas para seleção, movimentação e exclusão dos elementos que estiverem disponíveis aqui. Para obter mais detalhes, consulte Seção 2.2.

Barra lateral

À direita, você poderá encontrar a barra lateral que oferece diversas ferramentas para o seu fluxo de trabalho:

- Tipos de elementos: Este elemento fornece acesso direto para os tipos de arestas e nós disponíveis.
- Diário: Cada projeto tem o seu próprio diário que pode ser usado por exemplo para: notas das tarefas, resultados ou observações.
- API de criação de scripts: Para obter acesso direto à documentação do script, você poderá abrir este widget.

Editor de script

Neste editor de texto você pode criar os algoritmos como explicado em detalhes em capítulo 3. Você poderá trabalhar com vários documentos de script ao mesmo tempo usando diversas abas.

Resultado do programa

Esta área de texto poderá apresentar as informações de depuração ou o resultado do script para o seu algoritmo, dependendo da configuração no topo deste elemento. Se o script emitir um erro, o resultado de depuração é automaticamente apresentado.

Controles

Aqui você pode encontrar controles para a execução do script. Você pode executar o script atualmente aberto no editor do script clicando em **Executar**. Enquanto o script estiver em execução, é possível interrompê-lo se clicar no botão **Parar**.

2.2 Barra de ferramentas do editor de grafos

Esta barra de ferramentas consiste nas seguintes ações. Clicar em uma ação significa que o ponteiro do seu mouse aplica esta ação ao quadro do editor do grafo:

- **Selecionar e mover**: Para selecionar os elementos, você poderá clicar no espaço em branco do quadro, manter o mouse pressionado e desenhar um retângulo que contenha alguns elementos de dados e/ou arestas para selecionar estes elementos ou então clicar diretamente em um elemento não selecionado para escolher esse elemento. Se clicar em um elemento ou conjunto de elementos selecionados, você poderá mover estes elementos em conjunto. A movimentação dos elementos selecionados também é possível com as teclas direcionais.
- **Criar nó**: Clique em uma posição arbitrária no quadro do editor do grafo para criar um novo elemento de dados que pertença ao grafo selecionado na estrutura de dados. Mantendo o ponteiro do mouse pressionado no botão, um menu mostra a quais tipos de dados o novo elemento de dados criados pode pertencer (somente se existirem diferentes tipos de dados).
- **Criar aresta**: Clique em um elemento de dado, mantenha o mouse pressionado e desenhe uma linha para outro elemento de dado, apontando a aresta nessa direção. Esta ação só será bem-sucedida se o grafo atual permitir adicionar esta aresta (por exemplo, em um grafo não-direcionado, você não poderá adicionar várias arestas entre dois elementos de dado). Mantendo o ponteiro do mouse pressionado no botão, um menu mostra qual o tipo das novas arestas criadas pode ser selecionado (somente se existirem diferentes tipos de aresta).
- **Excluir**: Clique em um elemento para excluí-lo. Se excluir um nó, todas as arestas adjacentes serão também excluídas.

Capítulo 3

Criação de scripts

3.1 Executando algoritmos no Rocs

O Rocs usa internamente o mecanismo JavaScript do QtScript. Isso significa que todos os algoritmos que implementarem deverão usar JavaScript. Nos seguintes pontos, explicaremos como acessar e modificar os elementos de um documento de grafo a partir do mecanismo de criação de scripts. É importante apontar que as alterações feitas pelo mecanismo de criação de scripts são diretamente refletidas nas propriedades dos elementos do editor de grafos.

3.1.1 Controle da execução dos scripts

Existem diferentes modo de execução para seus algoritmos:

- **Executar**: Executa o script até ele terminar.
- **Parar**: Interrompe a execução do script (só está disponível enquanto um script estiver em execução).

3.1.2 Resultado do programa

Durante a execução de um algoritmo, o resultado do programa e da depuração é exibido no *Resultado do script & depuração*. Se o motor de script detecta um erro de sintaxe em seu script, o erro é também exibido como uma mensagem de depuração. Observe que todas as mensagens do programa são também exibidas na saída de depuração (exibida em texto em negrito).

Você pode controlar o texto que é exibido no resultado do script com as seguintes funções:

```
Console.log(string mensagem);    // exibe a mensagem como resultado do ↩
    script
    Console.debug(string mensagem);  // exibe a mensagem como resultado da ↩
        depuração
    Console.error(string message);          // exibe a mensagem como ↩
        resultado do erro
```

3.1.3 API do mecanismo de criação de scripts

Cada uma das diferentes partes do Rocs oferece um elemento estático que pode ser acessado pelo mecanismo de criação de scripts. São elas:

- **`Document`** para o documento do grafo
- **`Console`** para a saída do registro do console

Para uso explícito da API e para referência dos métodos, consulte a ajuda incorporada na barra lateral do Rocs.

Capítulo 4

Importar e exportar

4.1 Compartilhar projetos do Rocs

Os projetos do Rocs podem ser importados e exportados como arquivos `.tar.gz`. Estes arquivos podem ser usados para compartilhar projetos. A importação e exportação pode ser feita com o **Documento do grafo** → **Importar grafo** e **Documento do grafo** → **Exportar grafo como**, respectivamente.

4.1.1 Importar e exportar documentos de grafo

O Rocs atualmente suporta a importação e exportação para os seguintes formatos de arquivos:

- Arquivos DOT, também conhecidos como arquivos Graphviz
- Arquivos GML
- Arquivos no formato Trivial Graph
- Formato de Linguagem de Formatação Keyhole

4.1.1.1 Formato de Arquivo Trivial Graph

O *Formato Trivial Graph* (TGF, sigla em inglês) é um formato de arquivo simples baseado em texto para descrever gráficos. O arquivo TGF consiste de uma lista de definições de nó, o mapa de IDs dos nós para rótulos, seguido de uma lista de arestas. Neste formato, somente é possível ter um rótulo por nó e um valor por aresta. O Rocs interpreta os gráficos importados como gráficos não direcionados. Os gráficos exportados conterão duas arestas por conexão se as conexões forem bidirecionais.

4.1.1.1.1 Especificação do formato

- O arquivo inicia com uma lista de nós (um nó por linha), seguida de uma linha com somente o caractere "#", seguida pela lista de arestas (uma aresta por linha).
- Um nó consiste de um inteiro (identificador), seguido por um espaço, seguido por um string arbitrário.
- Uma aresta consiste de dois inteiros (identificadores) separados por um espaço, seguida por um espaço, seguida por um string arbitrário. É considerado que a aresta aponta do primeiro para o segundo identificador.

4.1.1.1.2 Exemplo

```
1 nó inicial
2 transmissor
3 fundo
#
1 2 azul
2 1 vermelho
2 3 verde
```

4.1.1.2 Linguagem DOT / Formato do Arquivo de Grafos do Graphviz

DOT é uma linguagem descritiva em texto simples que permite tanto uma representação legível para os usuários dos grafos, assim como um processamento eficiente pelos programas de formatação de grafos. O DOT é o formato de arquivo padrão do pacote de visualização de grafos Graphviz, mas também é largamente usado por outras ferramentas de grafos. As extensões usuais para os nomes de arquivos da linguagem DOT são *.gv* e *.dot*.

4.1.1.2.1 Funcionalidades não suportadas

O Rocs consegue processar todos os arquivos de grafos que tenham um grafo definido de acordo com a especificação da linguagem DOT[1]. O suporte das funcionalidades da linguagem é completo, apesar das seguintes exceções:

- subgrafo: Devido à falta da funcionalidade do conceito de subgrafos no Rocs, os subgrafos só são importados como um conjunto de elementos de dados e ligações. Especialmente, as ligações de e para subgrafos não são importadas.
- Atributos em HTML e XML: Os atributos (como as legendas) que contenham uma sintaxe em HTML ou XML são lidos sem qualquer alteração. Especialmente, não são lidos os ajustes de estilos e tipos de letra destes atributos.

4.1.1.2.2 Exemplo

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

[1] http://www.graphviz.org/content/dot-language

Capítulo 5

Créditos e Licença

Rocs
Direitos autorais do programa:

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

Direitos autorais da documentação:

- Direitos autorais da documentação 2009 Anne-Marie Mahfouf annma@kde.org
- Direitos autorais da documentação 2009 Tomaz Canabrava (tcanabrava AT kde.org)
- Direitos autorais da documentação 2011-2015 Andreas Cord-Landwehr (cordlandwehr AT kde.org)

Tradução de Marcus Gama marcus.gama@gmail.com e André Marcelo Alvarenga alvarenga@kde.org