# O Manual do Rocs

**Tomaz Canabrava**
**Andreas Cord-Landwehr**
**Tradução: José Pires**

# Conteúdo

**Resumo**

O Rocs é uma ferramenta da teoria dos grafos para o KDE.

# Capítulo 1

# Introdução

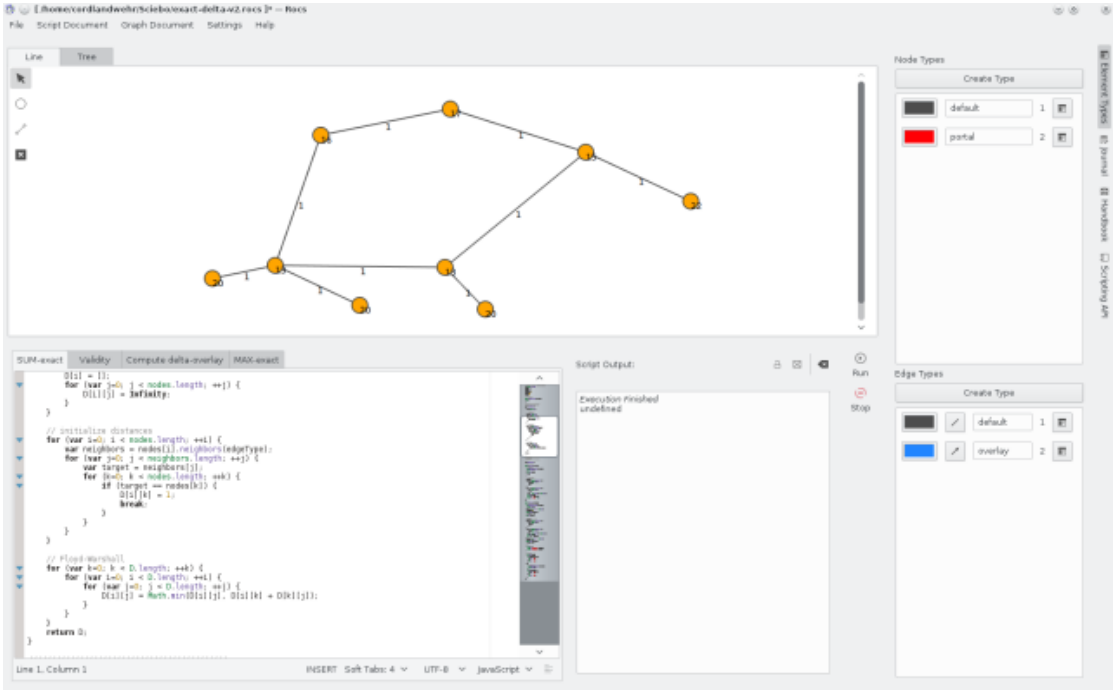
Neste capítulo, iremos dar uma visão geral sobre as funcionalidades e fluxos de trabalho básicos. Para o leitor que quiser começar a usar directamente o Rocs, sugerimos que leia o Seção 1.2 e, enquanto escreve programas, use o capítulo 3 como referência.

## 1.1 Objectivos, Público-Alvo e Funcionamento

O Rocs é uma Ferramenta da Teoria dos Grafos para todos os interessados no desenho e análise de algoritmos de grafos. Isto inclui explicitamente

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

Para todos eles, o Rocs oferece um editor gráfico fácil de usar para criar grafos, um motor de programação poderoso para executar algoritmos e diversos utilitários para simulações, experiências e exportações dos 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 e todas as alterações que o algoritmo efectue serão visíveis imediatamente no editor do grafo.



## 1.2 O Rocs em Resumo

Cada sessão do Rocs é um projecto: ao abrir o Rocs, é criado um projecto vazio e podê-lo-á substituir se carregar ou importar outro projecto. Como tal, um projecto em si consiste nos *documentos de grafos*, nos *programas/algoritmos* e num *registo*.

### 1.2.1 Documentos do 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 utilizador, as 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 ligado). Tudo o que pertence ao documento de um grafo fica acessível ao motor de programação através do objecto global **`Document`**.

### 1.2.2 Tipos de Arestas

Em alguns cenários, os grafos consistem em diversos tipos de arestas (p.ex., um grafo não-direccional mais as arestas da árvore calculadas por um algoritmo de pesquisa primeiro-em-largura) que deverá ser tratado e apresentado de forma diferente. Para tal, para além de um tipo de aresta predefinido, poderá definir outros tipos de arestas arbitrários. Cada tipo de aresta tem a sua representação visual individual, propriedades dinâmicas e poderá ser configurado como direccionado ou não-direccionado. A interface de programação oferece métodos de conveniência para aceder de forma específica apenas às arestas de determinados tipos.

### 1.2.3 Tipos de Nós

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

### 1.2.4 Propriedades

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

Também poderá usar o motor de programação para aceder às propriedades registadas e alterar os seus valores. No exemplo que se segue, assume-se que a propriedade ˝peso˝ está registada para o tipo de aresta predefinido.

```
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 secção, iremos querer criar um projecto de exemplo para explorar algumas das funções mais importantes do Rocs. O objectivo é criar um grafo e um programa que ilustre um algoritmo de 2 aproximações para o problema da *cobertura mínima de vértices*. Este problema consiste em encontrar um sub-conjunto de nós do grafo C com um tamanho mínimo, de modo que cada aresta do grafo esteja ligada a pelo menos um nó do C. Este problema é conhecido por ser NP-difícil e queremos ilustrar como encontrar uma aproximação com um factor 2, encontrando uma correspondência no grafo indicado.

O nosso objectivo é visualizar a correspondência 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 dados que usamos para distinguir os nós contidos em C e os não-contidos.

### 1.3.1 Gerar o Grafo

Para criar o grafo, o Rocs oferece um utilitário que poderá gerar grafos. Este está disponível no menu **Documento do Grafo** → **Ferramentas** → **Gerar o Grafo**. Aí, iremos gerar 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 Criar os Tipos de Elementos

Usamos os **Tipos de Elementos** e criamos um segundo tipo de nó, assim como um segundo tipo de aresta. Para ambos os tipos novos, poderemos abrir a janela de propriedades, usando os respectivos botões de **Propriedades** e configurar os ID's como `2`. Para além disso, iremos mudar as cores dos elementos desses dois novos tipos (para os distinguir dos tipos predefinidos). Finalmente, configuramos todos os tipos de arestas como sendo bidireccionais e os ID's dos tipos predefinidos como `1`.

### 1.3.3 O Algoritmo

Por último, temos de implementar o algoritmo de aproximação. Para tal, 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(); // 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;           // configurar a aresta como sendo correspondente
    E.shift();            // remover a 'e' (i.e., E[0]) da lista de arestas
    C.push(u);            // adicionar a 'u' ao C
    C.push(v);            // adicionar a 'v' ao C

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

    // remover 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]); // descobrir o índice
        if (indice != -1) {
            E.splice(indice, 1); // remover 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

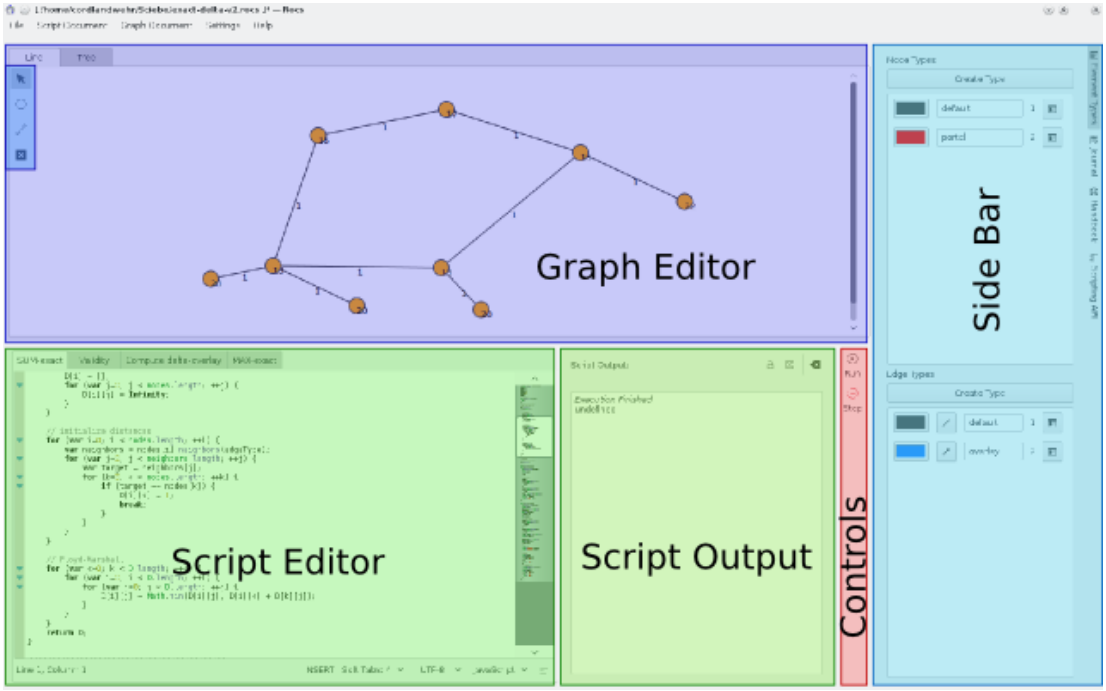
O algoritmo pode ser posto em execução com o botão de **Execução** no painel de controlo do programa.

# Capítulo 2

# A Interface de Utilizador do Rocs

## 2.1 Elementos Gerais da Interface do Utilizador

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




**Editor de Grafos**

O editor oferece um quadro onde poderá criar e modificar as suas estruturas de dados. Carregue com o botão direito no quadro, nos elementos de dados ou nas arestas para abrir os menus de contexto. Poderá usar as ferramentas da *Barra de Edição do Grafo* para modificar os elementos no quadro visual do grafo.

**Barra do Editor de Grafos**

A barra de ferramentas oferece as ferramentas **Criar um Nó** ou **Criar uma Aresta**, para criar novos elementos no quadro (Lembre-se da barra extra para seleccionar o tipo do nó ou aresta respectivo, que fica visível quando seleccionar uma dessas ferramentas. Do mesmo modo, as ferramentas para seleccionar e mover, assim como remover, os elementos estão disponíveis aqui. Para mais detalhes, veja em Seção 2.2.

**Barra Lateral**

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

- Tipos de Elementos: Este elemento dá-lhe acesso directo para os tipos de arestas e nós disponíveis.
- Diário: Cada projecto tem o seu próprio diário que pode ser usado isto é para: notas das tarefas, resultados ou observações.
- API de Programação: Para obter acesso directo ao manual e, deste modo, à documentação do programa, poderá abrir este elemento.

**Editor de Programas**

Neste editor de texto, poderá criar os algoritmos da forma descrita em capítulo 3. Poderá trabalhar com vários documentos de programas ao mesmo tempo, usando diversas páginas.

**Resultado do Programa**

Esta área de texto poderá apresentar a informação de depuração ou o resultado do programa para o seu algoritmo, dependendo da configuração do resultado. Se o programa emitir um erro, será seleccionado automaticamente o resultado da depuração.

**Controlos**

Aqui poderá encontrar os controlos para a execução dos algoritmos. Poderá executar o programa que está aberto de momento no editor do programa se carregar em **Executar**. Enquanto o programa estiver em execução, é possível interrompê-lo se carregar no botão **Parar**.

## 2.2 Barra do Editor de Grafos

Esta barra de ferramentas consiste nas seguintes acções. Se carregar no nome de uma acção, significa que o seu cursor do rato aplica esta acção ao quadro do editor do grafo:

-  **Seleccionar e Mover**: Para seleccionar os elementos, tanto poderá carregar no espaço em branco do quadro, mantenha carregado o rato e desenhe um rectângulo que contenha alguns nós e/ou arestas para seleccionar estes elementos ou poderá então carregar directamente num elemento não seleccionado para escolher esse elemento. Se carregar num elemento ou conjunto de elementos seleccionados, poderá mover estes elementos em conjunto. A movimentação dos elementos seleccionados também é possível com as teclas de cursores.

-  **Criar um Nó**: Carregue numa posição arbitrária no quadro do editor do grafo para criar um novo elemento de dados que pertença ao grafo seleccionado de momento. Mantendo o botão do rato pressionado, irá aparecer um menu de contexto onde poderá escolher o tipo de dados dos novos elementos (somente se existirem outros tipos de dados diferentes).

-  **Criar uma Aresta**: Carregue num elemento de dados, mantenha o rato pressionado e desenhe uma linha para outro nó, apontando a aresta nessa direcção. Esta acção só será bem-sucedida se o grafo actual permitir adicionar esta aresta (isto é, num grafo não-direccionado, não poderá adicionar várias arestas entre dois nós). Mantendo o botão do rato pressionado, irá aparecer um menu de contexto onde poderá escolher o tipo de aresta dos novos elementos (somente se existirem outros tipos de dados diferentes).

-  **Apagar**: Carregue num elemento para o apagar. Se apagar um nó, todas as arestas adjacentes serão também apagadas.

# Capítulo 3

# Programação

## 3.1 Executar os Algoritmos no Rocs

O Rocs usa internamente o motor de JavaScript do QtScript. Isto significa que todos os algoritmos que implementar deverão estar feitos em JavaScript. Nos seguintes pontos, iremos explicar como aceder e modificar os elementos de um documento de grafo a partir do motor de programação. É importante apontar que as alterações feitas pelo motor de programação são reflectidas directamente nas propriedades dos elementos do editor de grafos.

### 3.1.1 Controlar a Execução do Programa

Existem diferentes modos de execução para os seus algoritmos:

-  **Executar**: Executa o programa até este terminar.
-  **Parar**: Interrompe a execução do programa (só disponível se tiver um programa em execução).

### 3.1.2 Resultado do Programa

Durante a execução de um algoritmo, o resultado de depuração e do programa aparece no *Resultado de Depuração ∧ Programa*. Se o motor de programação detectar um erro de sintaxe no seu programa, o erro também aparece como uma mensagem de depuração. Repare que todas as mensagens do programa também são apresentadas no resultado de depuração (aparecendo a negrito).

Pode controlar o texto que é apresentado no resultado do programa, usando as seguintes funções:

```
Console.log(string mensagem);            // mostra a mensagem como  ↩
    resultado do programa
    Console.debug(string mensagem);             // mostra a mensagem como  ↩
        resultado de depuração
    Console.error(string mensagem);             // mostra a mensagem como  ↩
        resultado de erro
```

### 3.1.3 API do Motor de Programação

Cada uma das diferentes partes do Rocs oferece um elemento estático que poderá ser acedido pelo motor de programação. Estas são: o

- **`Document`** para o documento do grafo
- **`Console`** para o resultado do registo da consola

Para o uso explícito da API e para uma referência dos métodos, veja por favor a ajuda incorporada na barra lateral do Rocs.

# Capítulo 4

# Importar e Exportar

## 4.1 Intercâmbio de Projectos Rocs

Poderá importar e exportar projectos do Rocs como ficheiros de pacotes `.tar.gz`. Estes pacotes poderão ser usados para o intercâmbio de projectos. A importação e exportação poderão ser efectuados com as opções **Documento do Grafo** → **Importar um Grafo** e **Documento do Grafo** → **Exportar o Grafo Como**, respectivamente.

### 4.1.1 Importação e Exportação de Documentos de Grafos

O Rocs suporta de momento a importação e exportação dos seguintes formatos de ficheiros:

- ficheiros DOT, também conhecidos por ficheiros do Graphviz
- ficheiros GML
- ficheiros no formato TGF (Trivial Graph Format)
- Formato KML (Keyhole Markup Language)

#### 4.1.1.1 Formato TGF (Trivial Graph Format)

O *Trivial Graph Format* (TGF) é um formato de ficheiros simples em texto para descrever grafos. Um ficheiro TGF consiste numa lista de definições de nós que associam os ID's dos nós às legendas, seguidas de uma lista de arestas. Neste formato só é possível ter uma legenda por nó e um valor por aresta. O Rocs interpreta os grafos importados como não-direccionais. Os grafos exportados irão ter duas arestas por ligação, caso estas sejam bidireccionais.

##### 4.1.1.1.1 Especificação do Formato

- O ficheiro começa com uma lista de nós (um por cada linha), seguido de uma linha com o único carácter ´´#´´, seguido de uma lista de arestas (uma por cada linha).
- Um nó consiste num número inteiro (identificador), seguido de um espaço e depois um texto arbitrário.
- Uma aresta consiste em dois inteiros (identificadores) separados por um espaço, seguidos de um espaço e depois por um texto arbitrário. Assume-se que a aresta direccionada aponta do primeiro identificador para o segundo.

#### 4.1.1.1.2 Exemplo

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

### 4.1.1.2 Formato da Linguagem DOT / Graphviz

A linguagem DOT é uma linguagem de descrição de grafos em texto simples que permite tanto uma boa representação legível dos grafos, assim como um processamento eficiente pelos programas de formatação de grafos. O DOT é o formato de ficheiros por omissão para o pacote de visualização de grafos Graphviz, mas também é amplamente usado por outras ferramentas de grafos. As extensões de ficheiros normais do DOT são a *.gv* e a *.dot*.

#### 4.1.1.2.1 Funcionalidades Não Suportadas

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

- sub-grafos: Devido à falta do conceito de sub-grafos no Rocs, os sub-grafos são importados apenas como um conjunto de elementos de dados e ligações. Especialmente, as ligações de e para sub-grafos não são importadas.
- Atributos em HTML e XML: Os atributos (como as legendas) que tenham uma sintaxe em HTML ou XML são lidos sem qualquer alteração. Especialmente, não serão processados os estilos e ajustes de tamanho do texto a partir desses 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
Copyright do Programa:

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

'Copyright' da Documentação:

- Documentação com 'copyright' 2009 de Anne-Marie Mahfouf annma@kde.org
- Documentação com 'copyright' 2009 de Tomaz Canabrava (tcanabrava at kde dot org)
- Documentação com 'copyright' 2011-2015 de Andreas Cord-Landwehr (cordlandwehr AT googlemail.com)

Tradução de José Nuno Pires zepires@gmail.com