RODOLFO DA COSTA LADEIRA

# UM MÉTODO PARA DETERMINAR A POSIÇÃO DO TERRENO E A ALTURA DE UMA BARRAGEM QUE SEJA CAPAZ DE GERAR UM RESERVATÓRIO COM UM DADO VOLUME

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de Ma*gister Scientiae.*

VIÇOSA
MINAS GERAIS – BRASIL
2013

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
L154m
2013

Ladeira, Rodolfo da Costa, 1983-
Um método para determinar a posição do terreno e a altura de uma barragem que seja capaz de gerar um reservatório com um dado volume / Rodolfo da Costa Ladeira. – Viçosa, MG, 2013.
xiv, 100 f. : il. (algumas color.) ; 29 cm.

Inclui apêndices.
Orientador: Marcus Vinícius Alvim Andrade.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 44-47.

1. Algorítmos - Projetos. 2. Barragens e açudes. 3. Sistemas de informação geográfica. 4. Reservatórios. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 005.1

**RODOLFO DA COSTA LADEIRA**

# UM MÉTODO PARA DETERMINAR A POSIÇÃO DO TERRENO E A ALTURA DE UMA BARRAGEM QUE SEJA CAPAZ DE GERAR UM RESERVATÓRIO COM UM DADO VOLUME

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 27 de junho de 2013.

Luciana Brugiolo Gonçalves

Leacir Nogueira Bastos

Marcus Vinícius Alvim Andrade
(Orientador)

*Dedico esta conquista a uma mulher que em toda sua vida lutou e agora colhe: minha mãe, Maria José.*

*"We never know the worth of water till the well is dry"*
(Thomas Fuller)

# Agradecimentos

Gostaria de agradecer todas as pessoas que diretamente ou indiretamente contribuíram para mais esse sucesso. Esta não é uma tarefa simples de se realizar, pois muitos foram aqueles que ao meu ver contribuíram. Por hora, agradeço aqueles que tiveram relação direta com esta conquista, e, guardo com carinho uma gratidão especial aos não mencionados claramente aqui.

Ao meu Prof. Marcus Vinícius Andrade, meu orientador, que me guiou e me amadureceu ao longo desses anos, permitindo-me realizar este sonho. Ao Prof. Salles Magalhães que contribuiu diretamente com ideias e soluções enriquecendo o trabalho. Ao amigo Maurício Gruppi que também contribui nas soluções deste estudo. Ao conselheiro Prof. Fernando Prusky que compartilhou conosco seus conhecimentos da área.

À todos os professores do Departamento de Informática da UFV, aos demais funcionários do departamento e da universidade que contribuíram para minha formação acadêmica estando sempre atentos e dispostos a me servirem.

Aos meu amigos do laboratório de pesquisa: Thiago Luange, Guilherme Pena, Jaílton Coelho; aos amigos do programa de mestrado e de graduação, aos amigos Asher Coelho e Vitor Fontes, e, ao meu grande amigo Renan Nascimento, que compartilharam dos momentos felizes e tristes nesta jornada.

Aos meus familiares que sempre estiveram dispostos a me ajudar. Ao meu pai, Lair, com sua ajuda e suporte financeiro. Ao meu irmão, Rodrigo, que esteve sempre comigo, superando as dificuldades em se estudar distante de casa.

Em especial, devo eternos agradecimentos à minha mãe e também à minha namorada. Ambas alegraram-se com minhas alegrias e entristeceram-se com minhas tristezas.

A Luciana, minha amada namorada, muito me ouviu, indignou-se com minhas indignações, compartilhou das mais sombrias situações e soube com sabedoria me acalentar.

A Maria José, minha amada mãe, sempre teve um amor infinito, sempre me

ouviu e me guiou, alimentou e fortaleceu o sonho desta conquista mútua dando-me a sabedoria de um homem.

E não há nada para comparar, para poder explicar a gratidão que sinto!

Muito obrigado!

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Símbolos

## Gerais

| | |
|---|---|
| **ANA** | Agência Nacional de Águas |
| **CNRH** | Conselho Nacional de Recursos Hídricos |
| **COPAM** | Conselho Estadual de Política Ambiental |
| **GPS** | Sistema de posicionamento global |
| **MDE** | Modelo digital de elevação |
| **MFD** | Direção de fluxo múltipla |
| **RASTER** | MDE representado por matriz de quadrados regulares |
| **SAR** | Radar de abertura sintética |
| **SFD** | Direção de fluxo única |
| **SIG** | Sistema de informação geográfica |
| **SNISB** | Sistema Nacional de Informações sobre Segurança de Barragens |
| **TIN** | MDE representado por rede de triângulos irregulares |

## Específicos

| | |
|---|---|
| **ALAGA** | Algoritmo para obtenção do reservatório e suas características |
| **C++** | Linguagem de programação |
| **DIRMS** | Método para cálculo da direção da barragem por soma de vetores unitários |
| **DIRMP** | Método para cálculo da direção da barragem por soma de vetores unitários ponderados |
| **DIRMA** | Direção da barragem de forma manual |
| **DIRDP** | Método para cálculo da direção da barragem por uso |

| | |
|---|---|
| | do algoritmo de *Douglas-Peucker* |
| **JRPos** | Algoritmo de posicionamento automático de barragens a ser comparado no estudo |
| **Qt** | Framework multi plataforma com facilidades de criação de janelas na linguagem C++ |
| **MinGW/gcc** | Compilador de linguagens da família C/C++ |
| **qdoc** | Ferramenta do framework `Qt` para gerar documentação de código |
| **RAM** | Memória de accesso randômico (volátil) |
| **Re-Build** | Sistema com interface gráfica para manipulação de reservatórios |
| **RWDamming** | Algorítmo que determina a posição do reservatório mais viável |
| **RWDammingO1** | Algorítmo que determina a posição do reservatório mais viável menos otimizado |
| **RWDammingSO** | Algorítmo que determina a posição do reservatório mais viável sem otimização |
| **RWFlood** | Algorítmo que determina a rede de drenagem |
| **SRTM** | Missão espacial para mapeamento topográfico através do radar demoninado pelo próprio acrônimo |

# Resumo

LADEIRA, Rodolfo da Costa, M.Sc., Universidade Federal de Viçosa, junho de 2013. **Um método para determinar a posição no terreno e a altura de uma barragem que seja capaz de gerar um reservatório com um dado volume.** Orientador: Marcus Vinícius Alvim Andrade.

Esse estudo descreve um novo método para localizar a posição a se construir uma barragem criando um reservatório hídrico. Este problema é pouco explorado na literatura, tendo-se poucos estudos que o solucionam. E ainda, não há evidências de heurísticas para a solução do problema, pois os métodos encontrados utilizam o método de busca exaustiva, avaliando todas as inundações geradas. Além do mais, para a realização do processo de cada inundação, estes métodos precisam definir *a priori* tanto o volume desejado quanto as características da barragem (extensão e altura). Em contrapartida, neste estudo apenas a posição da barragem e o volume desejado são necessários antes que o processo de inundação comece. Neste caso, a extensão e altura da barragem e a área inundada são obtidas pelo resultado da simulação do processo. Este método usa busca exaustiva otimizada aprimorada pela técnica de *Branch-and-Bound* para encontrar o posicionamento mais adequado da barragem. Para se utilizar este método foi desenvolvida uma interface gráfica, com o *framework Qt*, aprimorando a experiência do usuário. Este estudo contribui para a literatura trazendo um método de solução para problema mais eficiente que o encontrado na literatura e com uma nova metodologia.

# Abstract

LADEIRA, Rodolfo da Costa, M.Sc., Universidade Federal de Viçosa, June of 2013. **A method to determine the position in the terrain and the height of a dam which is able to generate a reservoir with a given volume.** Adviser: Marcus Vinícius Alvim Andrade.

This work describes a new method to find the best position to construct a dam creating a water reservoir. This problem is poorly explored in literature, having few studies that solves it. And yet, there is no evidence of heuristics to solve the problem, the found methods use exhaustive search, evaluating all generated floods. Furthermore to perform the process of each flooding, these methods require to define *a priori* the desired volume and the characteristics of the dam (extend and height). In contrast, in this study only the position of the dam and the desired volume are required before the flooding process begins. In this case, the extend and height of the dam and flooded area are obtained through the simulation result of flooding. This method uses optimized exhaustive search improved by *Branch-and-Bound* technique to find the most appropriate positioning of the dam. To use this method we developed a graphical user interface, through the *Qt*, improving the user experience. This study contributes to the literature bringing a more efficiently solution to the problem than related research and having a new methodology.

# Capítulo 1

# Introdução

A água é um recurso natural essencial à vida humana, seja para cultivo, geração de energia, transporte, consumo, entre outros. A Tabela 1.1 [42] expõe a distribuição das águas no planeta, sendo que potável são as água doce de lagos, água misturada no solo, rios e vapor d'água na atmosfera. Sendo assim, de toda água disponível na Terra menos de 0,02% é própria para consumo . A disponibilidade da água (potável) vem se reduzindo cada vez mais devido ao grande aumento da população e, consequentemente, aumento do consumo urbano, desmatamento das regiões de nascentes, entre outros [8, 4], também pelo mau uso dos recursos naturais (inclusive dos recursos hídricos), refletidos pela falta de políticas de conservação. De acordo com o relatório anual das Nações Unidas [20] projeções alarmantes constatam que mais de 45% da população mundial não terão a quantidade mínima necessária de água potável no ano de 2050.

O aumento da população implica, consequentemente, no aumento da necessidade de recursos hídricos, o que tem levado o ser humano a se adaptar ao meio ambiente, seja concentrando-se aos redores de recursos hídricos e/ou modificando o mesmo criando, nas bacias hidrográficas, desvios ou barragens para alagamentos. Por muito tempo, essas adaptações ao meio ambiente foram feitas de forma desordenada, sem muita técnica e, com isso, diversas catástrofes ocorreram, ora por escassez, ora por excesso hídrico.

Gerações futuras dependem que nós usemos os recursos hídricos de forma responsável e, diante disso, o gerenciamento adequado deste recurso tem se tornado cada vez mais importante. Vários elementos podem estar associados a esse gerenciamento: leis ambientais, conscientização da população, técnicas de regularização de vazões e estudos na área. O processo de regularização consiste em adotar medidas para manter a disponibilidade de água mesmo em períodos de estiagem e queda da

Tabela 1.1: Distribuição de água no planeta [42]

| **Local** | **Volume** ($km^3$) | **Percentual do total** (%) |
|---|---|---|
| Oceanos | 1370000 | 97,61 |
| Calotas polares e geleiras | 29000 | 2,08 |
| Água Subterrânea | 4000 | 0,29 |
| Água doce de lagos | 125 | 0,009 |
| Água salgada de lagos | 104 | 0,008 |
| Água misturada no solo | 67 | 0,005 |
| Rios | 1,2 | 0,00009 |
| Vapor d'água na atmosfera | 14 | 0,0009 |

vazão em cursos d'água. Geralmente, esse processo é realizado pelo represamento das águas através da construção de barragens em trechos bem determinados dos cursos d'água naturais. Os reservatórios têm por objetivo acumular parte da água disponível nos períodos chuvosos para compensar as deficiências nos períodos de estiagem exercendo, assim, um efeito regularizador das vazões naturais.

Dessa forma, a regularização das vazões por meio da construção de barragem (formação de reservatório) visa atingir os seguintes objetivos: o atendimento das necessidades do abastecimento urbano ou rural (irrigação); o aproveitamento hidroelétrico (geração de energia); a atenuação de cheias (combate às inundações); o controle de estiagens; o controle de sedimentos; a recreação; e, também, permitir a navegação fluvial. É importante observar que caso a interferência homem/natureza seja feita de forma irracional ao se projetar reservatórios, grandes prejuízos naturais e sociais poderão ocorrer, como: inundações de importantes regiões (ferrovias, rodovias, plantações, regiões de cidades, entre outros), déficit de recursos hídricos em algum ponto da bacia, maior ônus para construção e manutenção da barragem, entre outros.

Assim ao se projetar um reservatório deve-se levar em conta não somente as características da nova barragem e reservatório, mas também a região do terreno que sofrerá alagamento caso se construa a barragem naquele determinado local. Esse processo de avaliar o local mais adequado a se posicionar a barragem pode consumir um tempo considerável no processo de construção. Porém, avaliar o ponto

de posicionamento da barragem permite estabelecer com maior precisão o impacto sobre o alagamento da região, além de permitir, também, determinar os recursos mínimos que serão necessários sua construção.

Com isso, como são vários os fatores que influenciam na solução de posicionamento da barragem perante um trecho de uma rede de drenagem, considera-se este um problema de otimização. Voltados a dinâmica computacional, a literatura carece de estudos mais detalhados que tratam esse problema. Em geral, os estudos existentes [37, 35] solucionam o problema de posicionamento de barragem utilizando busca exaustiva e demandam um longo tempo de processamento. Alguns dos fatores que influenciam e devem ser avaliados na construção de uma barragem são: largura da barragem, topografia do local, área alagada, limitações legais e comerciais, e a capacidade [12] do reservatório a ser construído. Sendo que o último fator é uma questão crucial no processo de regularização de vazão [4].

Sendo assim, o objetivo desse estudo é desenvolver um método de posicionamento de barragens para gerar um reservatório hídrico, que seja mais simples e ao mesmo tempo mais eficiente que os métodos existentes. O método proposto será comparado com o método descrito em [37], o qual será denominado, nesse trabalho, por JRPos. Assim, dado um determinado trecho da rede de drenagem (determinado pelo usuário) que necessite da regularização da vazão, o método proposto deve ser capaz de definir o posicionamento mais adequado dentro desse trecho para a construção da barragem. Além disso, foram desenvolvidos também um método para definição da área alagada e também uma interface gráfica de interação com usuário para auxiliar e complementar o método de posicionamento.

Apesar deste estudo estar ligado ao contexto computacional do problema de posicionamento e criação do reservatório hídrico, vale ressaltar que há estudos na literatura voltados ao contexto da hidrologia que fazem uma avaliação aprimorada do processo de criação do reservatório e todo o seu contexto, porém os mesmos não exploram o processo de posicionamento automático de um reservatório. Como exemplo prático cita-se o estudo denominado “Atlas digital das águas de minas”, encontrado em [40], que fornece um mapeamento completo e atualizado sobre os recursos hídricos superficiais do estado de Minas Gerais, permitindo obter a disponibilidade hídrica em uma região e também simular a construção de um reservatório hídrico.

Este documento está divido nos seguintes capítulos: Revisão Bibliográfica, Materiais e Métodos, Resultados, Conclusões. No capítulo Revisão Bibliográfica (Capítulo 2) é apresentado um conjunto de conceitos técnicos sobre o problema e aquilo que o envolve. O capítulo Materiais e Métodos (Capítulo 3) apresentará

os materiais e métodos empregados nesse trabalho para a solução do problema. No capítulo Resultados (Capítulo 4), são apresentados todos os resultados obtidos pelos métodos empregados. Por fim, no capítulo Conclusões (Capítulo 5), expõe-se as conclusões sobre este estudo e suas contribuições, além de apresentar alguns possíveis trabalhos futuros.

# Capítulo 2

# Revisão Bibliográfica

## 2.1 Representações de terrenos

Objetos do mundo real (estradas, terrenos, recursos hídricos, fluxos de ar e outros) podem ser representados por dados digitais através de SIG's (Sistema de Informação Geográfica) [5].

A elevação de um terreno pode ser representada por um modelo digital de elevação (MDE) que, por definição [5], é uma representação digital da topografia da superfície do solo ou terreno. Os métodos para obtenção de elevação digital são feitos por levantamento geográfico ou sensoriamento remoto. Dentre os quais podemos destacar os radares de abertura sintética (SAR - *Synthetic Aperture Radar* acrônimo inglês) [1]. Os dados utilizados para criar os MDE's normalmente são obtidos por:

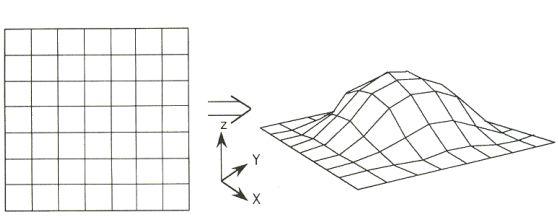
- Cinemática em tempo real (GPS) - sistema de satélites que permite definir a localização, velocidade, direção e tempo do receptor de GPS [10].
- Fotogrametria - tecnologia de sensoriamento remoto no qual propriedades geográficas são obtidas através de imagens fotográficas [39].
- LIDAR - tecnologia de sensoriamento remoto óptico no qual propriedades topográficas são obtidas através da reflexão da luz [9].
- Teodolito - instrumento para medições de ângulos da horizontal ou vertical que possibilita um levantamento terrestre [41].

Geralmente um MDE é representado como uma matriz de quadrados regulares (RASTER) ou como uma rede de triângulos irregulares (TIN) [11]. A Figura 2.1 apresenta, respecti-

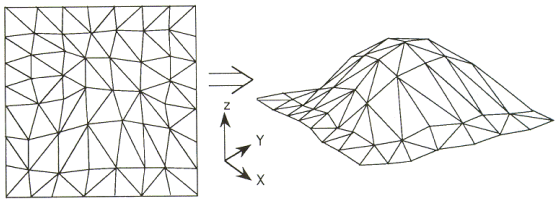
[1] Radar de Abertura Sintética (SAR) - "Satélites que levam a bordo um tipo de sensor imageador (Sensor eletrônico ou óptico para obtenção de imagens de uma superfície a partir de satélite ou aeronave [3].) ativo que opera na região de micro-ondas" [17]

vamente, um terreno real (Figura 2.1a), e as possíveis representações MDE para o mesmo com matriz de quadrados regulares (Figura 2.1b) e com TIN (Figura 2.1c).

(a) Terreno real




(b) Possível representação em grades regulares [21]




(c) Possível representaçao em TIN [21]

Figura 2.1: Representações de um terreno

Uma TIN é uma representação vetorial de um terreno composta por uma distribuição irregular de nós e linhas com coordenadas em três dimensões $(x, y, z)$ que são dispostas em uma rede de triângulos não sobrepostos [11]. Uma das vantagens da TIN é que a distribuição dos pontos pode levar em conta quais pontos são necessários para uma representação mais precisa. Com isso, a superfície é mapeada pelo conjunto de retalhos triangulares e um dado ponto $(x, y)$ qualquer, contido no terreno, terá sua elevação determinada pela interpolação dos pontos do triângulo, cuja projeção planar contenha o ponto em questão.

Uma grade regular retangular é a representação de uma superfície em forma de matriz composta por uma distribuição regular das elevações de um terreno [19]. Logo, cada elemento *(linha,coluna)* da matriz representa um "ponto" do terreno e contem o valor da elevação naquele "ponto". A resolução do terreno é relacionada à resolução do RASTER (grade regular) e define a precisão do mapeamento do terreno. Existem MDE's com resoluções de centímetros à metros de acordo com o método de obtenção dos dados. Um fator importante a ser observado é que o grande aumento no uso dessa forma de representação, tem levado o seu nome a ser associado ao termo MDE. A Tabela 2.1 traz as principais diferenças entre essas duas representações expostas. Neste estudo é adotado o modelo de grade regular retangular.

Tabela 2.1: Comparação entre os modelos TIN e grade regular [11]

| **Grade regular retangular** | **Rede irregular triangular** |
|---|---|
| Apresenta regularidade na distribuição espacial dos vértices das células do modelo | Não apresenta regularidade na distribuição espacial dos vértices das células do modelo |
| Os vértices dos retângulos são estimados a partir das amostras | Os vértices dos triângulos pertencem ao conjunto amostral |
| Apresenta problemas para representar superfícies com variações locais acentuadas | Representa melhor superfícies não homogêneas com variações locais acentuadas |
| Estrutura de dados mais simples | Estrutura de dados mais complexa |
| Relações topológicas entre retângulos são explícitas | É necessário identificar e armazenar as relações topológicas entre os triângulos |
| Mais utilizado em aplicações qualitativas e para análises multiníveis no formato "raster" | Mais utilizado em aplicações quantitativas |

Existe ainda a representação de terrenos através de linhas de tendência, curva de nível ou isolinhas. Tal representação armazena distintas informações do terreno em linhas [41], sendo que cada linha tem um valor constante dessa informação.

Existem diversos métodos que transformam a representação de isolinhas para representação RASTER ou TIN, o que amplia o suporte ao uso dessa representação. Porém, nesse modelo, as informações entre as linhas não são representadas e, com isso, pode-se ter uma perda de precisão. Embora, essa representação tenha sido muito utilizada para criação de mapas cartográficos, com o advento dos SIG's, esse modelo de representação tem sido substituído pelo o modelo RASTER ou TIN.

## 2.2 Reservatório Hídrico

Um reservatório, barragem, açude ou represa, é uma barreira artificial, posicionada em trechos dos cursos d'água para regularização de vazões a fim de manter a disponibilidade de água mesmo em períodos de estiagem ou queda da vazão. Suas características, em

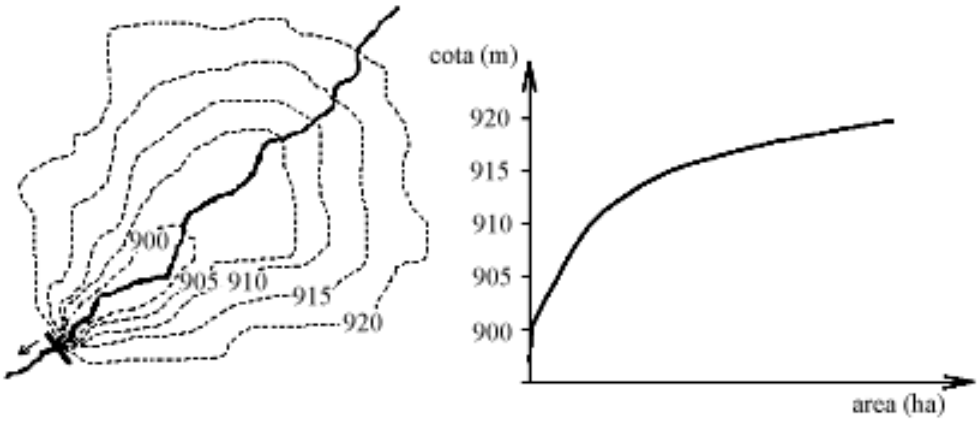



Figura 2.2: Curva cota x área para um reservatório [4]

especial a sua capacidade (volume), dependem da topografia do terreno alagado devido essa contenção [12]. Além do mais, a região alagada depende do nível em que se encontra o reservatório.

Existem duas formas principais para se calcular a capacidade de um reservatório. Através de isolinhas geradas pela relação $cota \times volume$ ou pela soma dos volumes para cada célula alagada do MDE.

### 2.2.1 Curvas $cota \times area$ e $cota \times volume$

Com a relação $cota \times area$ é possível criar um modelo isolinhas e definir uma curva que mapeia a área alagada para cada determinada cota (nível do reservatório). Para a criação da curva $cota \times area$ é necessário o emprego de um mapa topográfico em escala adequada. Para cada cota referida há uma dada curva de nível, planimetra-se[2] a área limitada pela curva de nível. Os pares de valores ($cota$ $(m)$,$area$ ($m^2$, $km^2$ ou ha)) são lançados em um gráfico e uma curva suave é esboçada através dos pontos [12], veja Figura 2.2. Com isso, é possível também obter o volume (capacidade) do reservatório através da curva $cota \times volume$, que é o resultado do cálculo integral da curva $cota \times area$. Essa integração é realizada numericamente, determinando-se os volumes $\Delta$Vol entre duas curvas de nível consecutivas. O volume é obtido de forma aproximada, multiplicando a média das áreas correspondentes às curvas de nível consecutivas pela diferença de cota dessas curvas de nível.

[2]Medir com o planímetro (instrumento para desenho técnico utilizado para medir a área de uma superfície plana arbitrária)

### 2.2.2 Soma dos volumes inundado nas células

Outra forma de se calcular a capacidade de um reservatório é através da soma dos prismas de coluna d'água sobrepostos em cada célula em um modelo MDE [37, 18]. Essa técnica é comumente utilizada quando a representação adotada do terreno é MDE. O cálculo do volume gerado por um reservatório hídrico utilizando um MDE e o procedimento de soma dos prismas é precedido pelos seguintes passos: define-se o posicionamento da barragem, a altura e extensão da mesma, e, daí, determinam-se o conjunto de pontos (células) a serem alagados devido a essa retenção e assim calcula-se a capacidade.

As células alagadas estão diretamente relacionadas ao nível do reservatório. Quanto maior o nível, a tendência é que mais células estejam alagadas. A capacidade (volume) é obtida através do somatório dos volumes armazenados em cada célula alagada, observe a Figura 2.3. Como cada célula é retangular, o volume é dado pelo produto entre a área da base (retângulo) e a altura do alagamento dessa célula (diferença entre o nível do reservatório e a elevação do terreno nessa célula).

### 2.2.3 Classificação e situação de reservatórios no Brasil

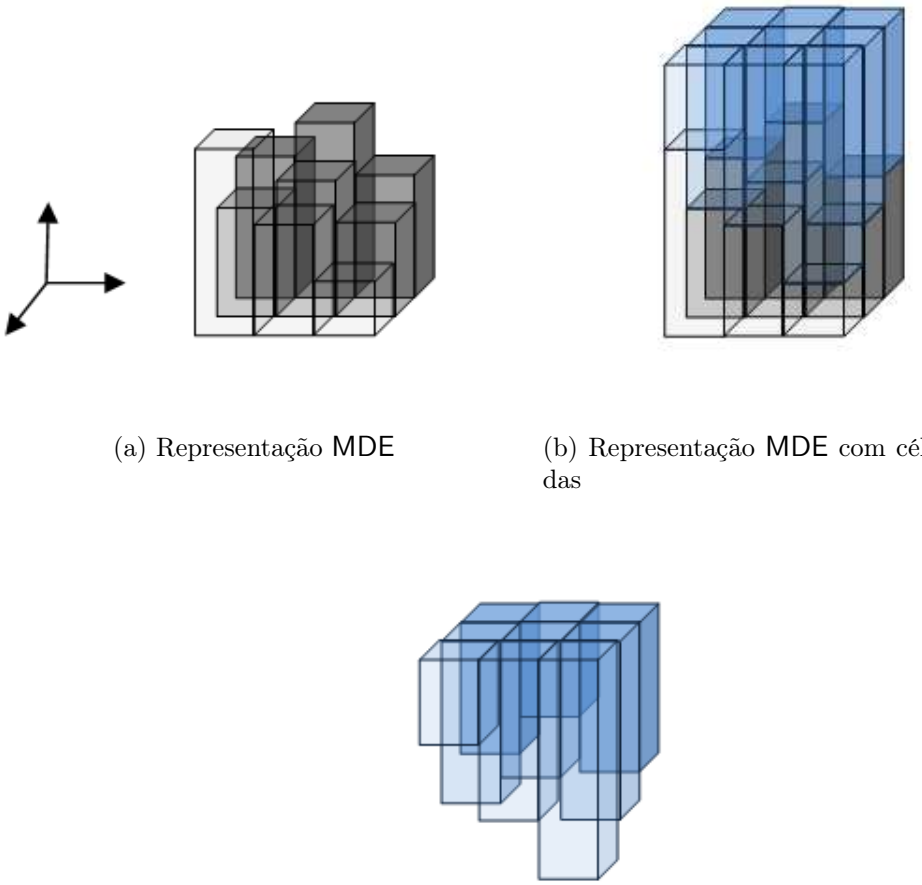
Assim como em outros países, o Brasil tem uma entidade federal destinada à implementação das políticas e gerenciamento dos recursos hídricos, a ANA (Agência Nacional de Águas), que foi criada pela *Lei n° 9.984, de 17 de julho de 2000* [6]. A ANA tem como funções:

- regular o uso da água da União
- coordenar a implementação da Política Nacional de Recursos Hídricos, cuja principal característica é garantir a gestão democrática e descentralizada dos Recursos Hídricos
- fiscalizar a segurança das barragens por ela outorgadas, e ainda, criar e constituir o Sistema Nacional de Informações sobre Segurança de Barragens (SNISB), através da Política Nacional de Segurança de Barragens aprovada pela *Lei n° 12.334, de 20 de setembro de 2010* [7]

A classificação das barragens outorgadas pela ANA é definida pela Seção I da *Lei n° 12.334, de 20 de setembro de 2010*, e é dada como:

**Art. 7°** *"As barragens serão classificadas pelos agentes fiscalizadores, por categoria de risco, por dano potencial associado e pelo seu volume, com base em critérios gerais estabelecidos pelo Conselho Nacional de Recursos Hídricos (CNRH)."*

§ **1°** *"A classificação por categoria de risco em alto, médio ou baixo será feita em função das características técnicas, do estado de conservação do empreendimento e do atendimento ao Plano de Segurança da Barragem."*



(a) Representação MDE

(b) Representação MDE com células alagadas

(c) Coluna d'água sobre as células

Figura 2.3: Representação de área alagada em um terreno MDE

§**2**° *"A classificação por categoria de dano potencial associado à barragem em alto, médio ou baixo será feita em função do potencial de perdas de vidas humanas e dos impactos econômicos, sociais e ambientais decorrentes da ruptura da barragem."*

Tais classificações, apesar de serem válidas por todo o país, não são aplicáveis para casos onde as barragens estão sobre o gerenciamento do estado ou outra jurisdição. Segundo a deliberação normativa COPAM (Conselho Estadual de Política Ambiental) *nº 87, de 17 de junho de 2005* [13], o porte de uma determinada barragem é definido segundo os critérios presentes na Tabela 2.2.

Uma competência da ANA é manter o cadastro informatizado das barragens sob sua jurisdição. Isso irá garantir um melhor gerenciamento e aplicação de políticas sobre

Tabela 2.2: Classificação barragem e reservatório definidos pelo COPAM [13]

| **Porte da Barragem** | **Altura da Barragem H** $(m)$ | **Porte do Reservatório** | **Volume do Reservatório Vr** $(m^3)$ |
|---|---|---|---|
| Pequeno | $H < 15$ | Pequeno | $Vr < 500000$ |
| Médio | $15 \leq H \leq 30$ | Médio | $500000 \leq Vr \leq 5000000$ |
| Grande | $H > 30$ | Grande | $Vr > 5000000$ |

as barragens, além da regularização documental. Por lei, toda barragem para se manter operante deve ser outorgada pela jurisdição responsável pela mesma. Estima-se que mais da metade das barragens do país não contém documentação nenhuma. Por isso, a ANA desenvolveu um sistema de informação para armazenar informações sobre as barragens de domínio da União. Tal sistema é um serviço *on-line* e permite ao usuário gerenciar sua(s) barragem(ns) cadastrada(s). A Figura 2.4 apresenta algumas das informações armazenadas no banco de dados da ANA.

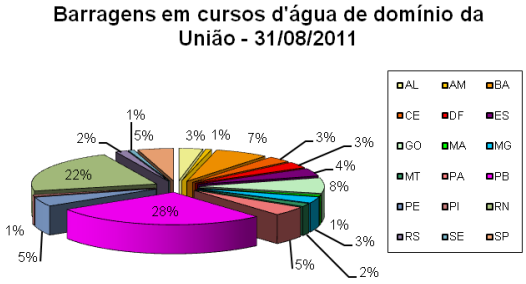
Até o momento, a maioria das barragens cadastradas tem capacidade entre $0.5hm^3$ e $3hm^3$ [3] (Veja a Figura 2.4c), que são consideradas barragens de médio a pequeno porte. Um fator importante que pode ser observado pela Figura 2.4a é que a região do país com maior número de barragens da união é justamente a região que mais necessita de água: o nordeste.

## 2.3 Determinação da rede de drenagem

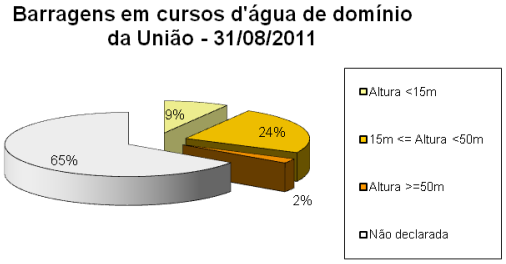
Um elemento importante para a construção de uma barragem e, consequentemente, para a geração de um reservatório é a rede de drenagem de um terreno, isto é, os rios. Um método normalmente utilizado para a obtenção dessa rede de drenagem consiste em se determinar a direção de fluxo para cada célula do terreno e, então, determinar o fluxo acumulado de forma a se identificar quais células fazem parte dos rios. Intuitivamente, a direção de fluxo corresponde ao caminho que a água deve seguir ao longo do terreno e o fluxo acumulado é a quantidade de água que alcança cada célula supondo que o terreno recebe, de uma ou mais áreas de contribuição, um determinado volume de água uniformemente distribuído sobre a sua superfície [27, 12]. Por área de contribuição entende-se a seção do terreno que contribui para a formação de um trecho da rede de drenagem [33, 38].

As principais formas de se modelar a direção de fluxo são: fluxo em direção única (SFD - *Single-flow-direction*) em que o fluxo de uma célula é direcionado para uma única

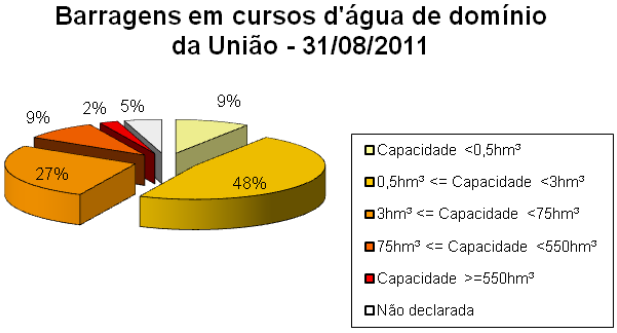
[3] 1 $hm^3$ equivale a 1.000.000$m^3$



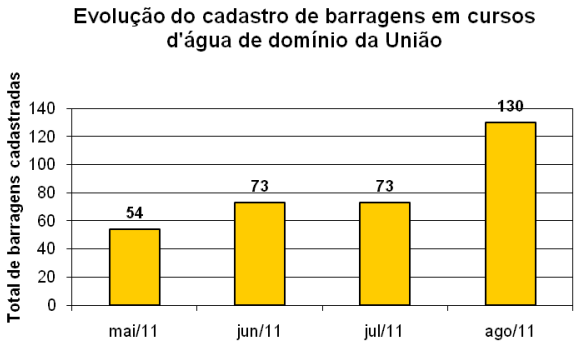

(a) Distribuição por estados do país




(b) Distribuição por altura do reservatório

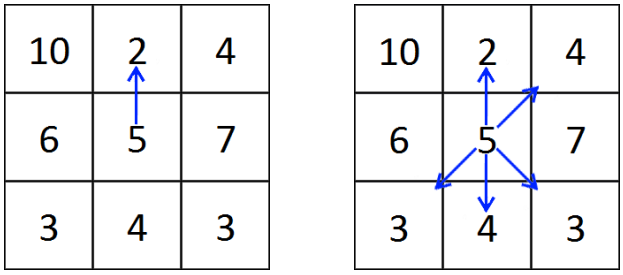



(c) Distribuição por capacidade do reservatório




(d) Crescimento de cadastros de barragens

Figura 2.4: Estatísticas das barragens da união geradas pela ANA [1]




(a) SFD (b) MFD

Figura 2.5: Modelos para direções de fluxo de MDE: 2.5a para direção única de fluxo da célula; 2.5b para direções múltiplas de fluxo da célula.

célula vizinha que possua o menor valor de elevação e que seja menor do que a elevação da célula em questão; e o fluxo em várias direções (MFD - *Multi-flow-directions*) onde o fluxo é dividido proporcionalmente em função da diferença de elevação entre a célula em questão e as suas vizinhas que possuam elevação menor [2]. Os modelos de direção de fluxo podem ser vistos na Figura 2.5.

Do ponto de vista computacional, a escolha dos modelos SFD ou MFD não é crítica, pois a direção de fluxo pode ser computada com mesma complexidade assintótica utilizando ambos modelos. No entanto, do ponto de vista prático, essa escolha é importante, pois o modelo SFD geralmente produz uma rede de drenagem com um menor número de trechos convergentes (afluentes) que são mais longos, enquanto o modelo MFD produz uma rede mais difusa, com um maior número de trechos mais curtos [2]. Nesse trabalho será adotado o modelo SFD.

O problema da direção de fluxo pode ser definido formalmente como a tarefa de atribuir direções de fluxo para todas as células do terreno de tal modo que as três condições seguintes sejam satisfeitas [2]:

1. Cada célula tem pelo menos uma direção de fluxo
2. Não existem caminhos de fluxo cíclicos
3. Cada célula no terreno possui um caminho de fluxo para a borda do terreno

Há diversos métodos para a obtenção da rede de drenagem [33, 22, 36, 38] e, conforme descrito nesses trabalhos, a maior dificuldade nesse processo é a ocorrência de células onde não é possível determinar a direção de fluxo diretamente porque ou a célula é um mínimo local ou pertence a uma região horizontalmente plana. Um mínimo local é uma célula do terreno cuja elevação é menor ou igual à elevação de todas as suas vizinhas e uma região plana corresponde a um conjunto de células adjacentes com uma mesma elevação.

Em casos onde células pertencem a uma região plana e, consequentemente, têm mesma elevação, o processo de definição da direção de fluxo das mesmas é dificultado

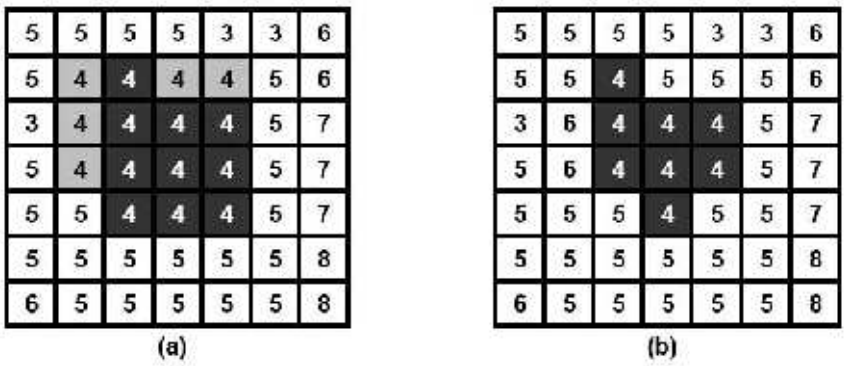



Figura 2.6: Modelo digital de elevação para região platô(a) e fosso(b)

e, portanto, é necessário adotar uma estratégia para contornar tal situação. As células que não são mínimos locais, e que estejam situadas na borda de uma região plana são denominadas pontos de escoamento. As regiões planas podem ser classificadas de duas maneiras: um *platô* (Figura 2.6a), que é uma região plana que possui pelo menos um ponto de escoamento ou um *fosso* (Figura 2.6b), que é uma região plana sem ponto de escoamento [2, 22]. Como a tendência natural da água é escoar para regiões de menor elevação, com menores obstáculos, é comum considerar que o fluxo de um platô seja orientado na direção dos pontos de escoamento. Por outro lado, no caso dos fossos, a estratégia utilizada, em geral, é estabelecer um preenchimento artificial desse fosso até que o mesmo se torne um platô, e daí adota-se a estratégia utilizada para platôs.

Vários métodos de obtenção da rede de drenagem [36, 25, 14] eliminam os fossos realizando um pré-processamento do terreno para preenchê-los até que um ponto da grade com valor de elevação menor do que a elevação máxima do fosso seja encontrado.

Após a obtenção da direção de fluxo, o próximo passo é a determinação do fluxo acumulado em cada célula do terreno, isto é, a quantidade de água que atinge cada célula supondo que cada uma receba inicialmente uma unidade de água e que essa água seguirá as direções obtidas no passo anterior. Diversos métodos para a obtenção do fluxo acumulado como [33, 2, 36] se baseiam no método convencional de seguir as direções de fluxo. Outros [28, 29] modelam esse problema como um sistema de equações lineares, cuja solução fornece o fluxo acumulado em cada célula.

Uma vez obtido o fluxo acumulado, a rede de drenagem pode ser computada selecionando-se todas as células cujo o valor do fluxo acumulado é maior do que certo limite pré-estabelecido, denominado fluxo mínimo. A partir da direção de fluxo e do fluxo acumulado, outros elementos hidrográficos, como as bacias de acumulação, podem ser obtidos.

Esse processo de obtenção da rede de drenagem exige uma quantidade considerável de processamento, principalmente devido à etapa de remoção dos fossos e tratamento dos platôs [34]. Na verdade, na maioria dos métodos baseados nessa estratégia, mais de 50%

do tempo total de processamento é consumido nessa etapa. Para evitar a necessidade da execução dessa etapa de processamento, recentemente foi apresentado por Magalhães em [24] um novo método para obtenção da rede de drenagem, denominado RWFlood, e como este método tem grande impacto nesse estudo, é destinada uma seção própria ao mesmo.

### 2.3.1 Método RWFlood

O método RWFlood [24] baseia-se na simulação de uma inundação para calcular a direção de fluxo. Conceitualmente, o método simula um processo onde o terreno é uma ilha e o nível de água que circunda o terreno é elevado gradativamente simulando um processo de inundação. Note que, naturalmente, o fluxo da água num terreno segue um caminho inverso ao processo de inundação, isto é, as primeiras células a serem inundadas (onde a água entra no terreno) correspondem às células onde a água escoa para fora do terreno (foz dos rios); as próximas células a serem inundadas (vizinhas às primeiras inundadas no processo de inundação) serão as penúltimas antes da foz e assim por diante. Portanto, o processo de inundação permite obter a direção de fluxo, pois essa direção corresponde à direção contrária à inundação.

Resumidamente, esse processo de inundação é simulado inicializando-se o nível da água com o mesmo valor da elevação do(s) ponto(s) mais baixo(s) na borda do terreno, ou seja, esses serão os primeiros pontos a serem inundados e eles são inseridos em uma fila de prioridade, onde o topo contém sempre o ponto de menor elevação. Daí, o ponto $p$ no topo da fila é removido, processado (inundado) e, dentre os seus oito vizinhos, aqueles que ainda não foram visitados (inundados) são inseridos na fila. Porém, se a elevação de um ponto $q$ a ser inserido na fila for menor do que o nível da água (isto é, menor do que a elevação do ponto $p$) então a elevação de $q$ é aumentada, o que corresponde a inundar o ponto $q$ que passa a ser um ponto já visitado[4]. Note que, nesse momento, a direção de fluxo do ponto $q$ pode ser direcionada para o ponto $p$. A Figura 2.7 apresenta um esquema ilustrativo do processo de inundação do método RWFlood. O processo continua enquanto ainda houver células a serem inundadas no terreno.

Após o cálculo da direção de fluxo, o algoritmo RWFlood calcula o fluxo acumulado através de uma estratégia baseada em ordenação topológica. Conceitualmente, a ideia é supor a existência de um grafo onde cada vértice representa uma célula do terreno e há uma aresta ligando um vértice $v$ a um vértice $u$ se, e somente se, a direção de escoamento de $v$ aponta para $u$. Os vértices são inicializados com 1 unidade de água e o processamento se inicia num vértice $v$ cujo grau de entrada é 0. Esse vértice é marcado como visitado e, supondo que $v$ direciona o fluxo para o vértice $u$, então o fluxo do vértice $v$ é adicionado ao fluxo atual do vértice $u$. Além disso, a aresta que conecta o vértice $v$ ao vértice $u$ é

[4]Esse processo de elevação do nível de uma célula corresponde à remoção das depressões, nos demais métodos para obtenção da direção de fluxo o processo de remoção de depressões é realizado a parte.

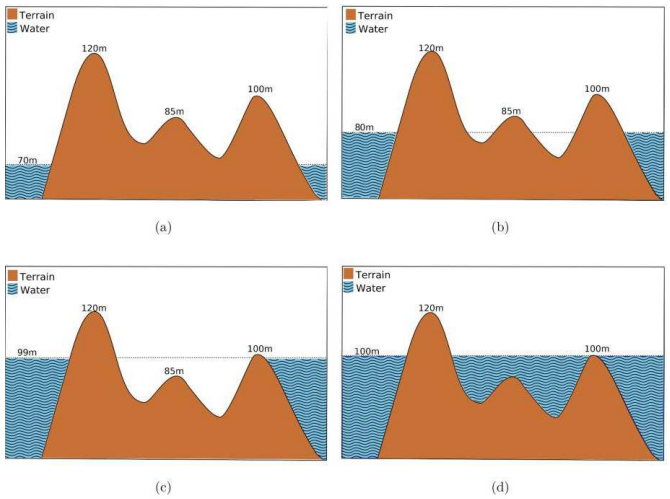



Figura 2.7: Algoritmo RWFlood-Processo de inundação para cálculo da direção de fluxo [24]

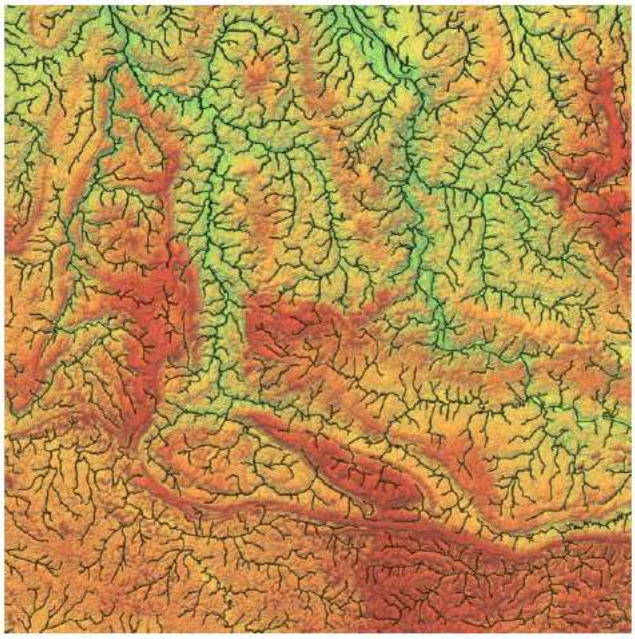


Figura 2.8: Rede de drenagem (destacado na cor preta) para região de Tapajós obtida pelo método RWFlood [24]

removida reduzindo, assim, o grau de entrada do vértice $u$, que será processado (visitado) quando o seu grau de entrada se tornar 0. Observe a Figura 2.8 , que apresenta a rede de drenagem para região contendo o rio Tapajós[5] obtida por esse método.

Como apresentado em [24], o método RWFlood pode ser implementado de forma

[5]Rio brasileiro que nasce no estado de Mato Grosso, que banha parte do estado do Pará e que deságua no rio Amazonas ainda no estado do Pará.

bastante simples e eficiente (com complexidade linear em relação ao tamanho do terreno) chegando a ser 100 vezes mais rápido do que os principais métodos descritos na literatura. Essa eficiência se deve principalmente ao fato de que não é necessário pré-processar o terreno para eliminar as depressões, visto que, elas são naturalmente removidas durante o processo de inundação. Além disso, o RWFlood também é capaz de processar grandes terrenos com mais de $10^9$ células.

# Capítulo 3

# Materiais e Métodos

A finalidade deste estudo é desenvolver um algoritmo que seja capaz de solucionar o problema de posicionamento de barragens a fim de gerar um reservatório hídrico com uma determinada capacidade para suprir a necessidade hídrica de um trecho da rede de drenagem.

Nas seções a seguir serão descritas as etapas desenvolvidas para obter o método que solucione o problema de forma eficiente. Na Seção 3.1 é descrito o método para definição da direção da barragem, tal elemento é importante para o resultado final da construção de um reservatório. Na Seção 3.2 é apresentado um método que calcula o reservatório e suas características. Na Seção 3.3 mostra-se a função utilizada para qualificar um determinado reservatório gerado, determinando-se o custo de construção segundo as variáveis analisadas. E na Seção 3.4 é descrito o método desenvolvido para solucionar o problema de posicionar a barragem. Como complemento do trabalho, foi desenvolvida uma interface gráfica de usuário, que é descrita na Seção 3.5.

## 3.1 Direção da barragem

O formato e a direção da barragem são fatores importantes a serem observados, pois afetam diretamente o resultado da criação de um reservatório hídrico. Nesse trabalho adotam-se somente barragens em formato retilíneo, sendo que o uso de outros formatos são propostos como trabalhos futuros. A direção da barragem deve ser definida em função do ponto de posicionamento da barragem, referente a rede de drenagem.

Devido ao modelo de representação de terrenos adotado neste estudo, temos quatro possíveis direções para uma determinada barragem. A Figura 3.1 apresenta as possíveis direções de uma barragem determinado pelo números 1,2,3,4.

Inicialmente, seria conveniente supor que uma barragem, em forma retilínea, seja construída perpendicularmente à direção de fluxo da rede de drenagem no ponto de posici-

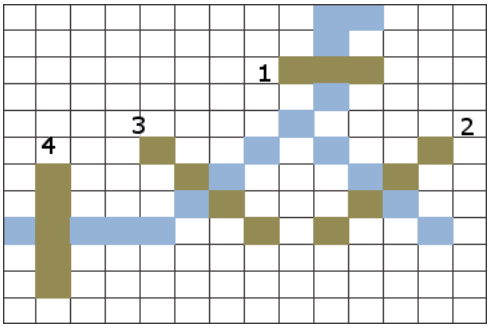



Figura 3.1: MDE com possíveis direções para uma barragem, representadas pelos números 1,2,3 e 4.

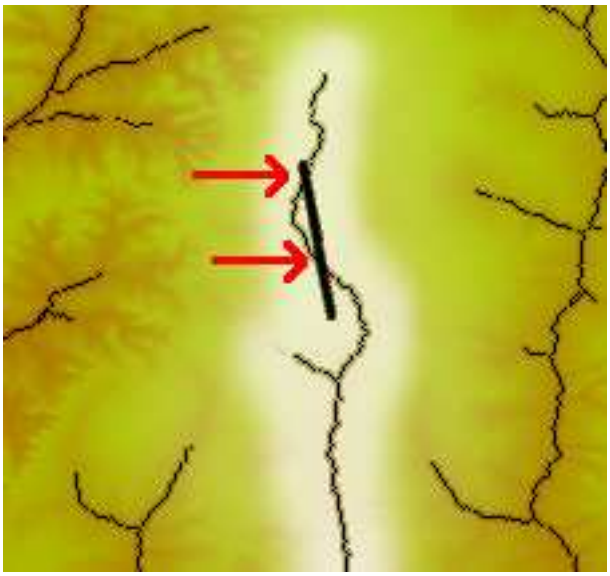


Figura 3.2: Situação indesejável para uma direção de barragem (imagem evidenciada)

onamento da barragem, porém, devido à representação do terreno ser discreta, há situações onde o fluxo, ou trajetória do rio, pode sofrer variações em curto espaço, e isto, poderia prejudicar o papel de contenção da barragem. Entre algumas das situações indesejáveis e que devem ser evitadas, pode-se observar através da Figura 3.2, uma direção de barragem em que mais de um ponto seria represado. Tal situação é indesejável, pois mesmo que esta barragem possa fazer o seu papel de contenção hídrica haveria gasto desnecessário de material para a construção e nem toda barragem estaria contribuindo para contenção, formando dois reservatórios consecutivos, sendo que nos métodos que identificam as células alagadas, apenas um desses reservatórios conseguiria ser tratado, sendo que o outro é um reservatório que não teria demanda hídrica (sua demanda está sendo retida pela outra barragem).

O estudo em questão propõe quatro distintos métodos para cálculo da direção da

barragem: DIRMS, DIRMP, DIRMA e DIRDP. Tais métodos permitem explorar distintas situações da rede de drenagem e tentam evitar algumas das situações indesejáveis como a demonstrada.

Estas técnicas foram elaboradas a fim de permitir ao usuário definir a técnica mais apropriada para a rede de drenagem avaliada. Tais técnicas analisam somente a rede de drenagem à montante do ponto de posicionamento, visto que, no contexto deste estudo, a rede de drenagem à jusante não contribuiria e nem interferiria no processo de retenção hídrica.

Para realizar os cálculos, os métodos DIRMS, DIRMP ou DIRDP utilizam $n$ pontos (vizinhos) consecutivos da rede de drenagem, e como mencionado, estão à montante do ponto de posicionamento, ou os segmentos de reta, que são formados pelos mesmos $n$ pontos tomados dois a dois. A direção da barragem será perpendicular ao vetor obtido pela solução destes métodos. Enquanto que, para o método DIRMA a direção é definida manualmente pelo usuário.

A técnica DIRMS calcula o vetor soma de vetores unitários compostos pelos $n$ vizinhos. Isto é, definem-se vetores do ponto de posicionamento da barragem à todos os $n$ vizinhos, e então, através destes vetores unitários calcula-se o vetor soma, obtendo assim um vetor que contém uma direção média para os vizinhos calculados. Veja Equação 3.1.

$$\overrightarrow{r} = \sum_{i=1}^{n} \frac{\overrightarrow{bv_i}}{\left|\overrightarrow{bv_i}\right|} \tag{3.1}$$

onde $b$ é o ponto de posicionamento da barragem, $v_i$ é o ponto à montante de $b$, para $i$ variando entre [1,$n$], $bv_i$ o vetor conectando esses dois pontos tendo seu módulo representado por $\left|\overrightarrow{bv_i}\right|$. O valor de $n$ é definido pelo usuário. O processo é apresentado de forma mais detalhada na Figura 3.3.

A técnica DIRMP calcula o vetor soma de vetores ponderados compostos pelos $n$ vizinhos. Da mesma maneira que DIRMS, DIRMP define vetores unitários do ponto de posicionamento da barragem à todos os $n$ vizinhos, e então, através destes vetores e dos pesos $P$ associados aos mesmos, calcula-se o vetor soma. Veja Equação 3.2.

$$\overrightarrow{r} = \sum_{i=1}^{n} \frac{P_i\overrightarrow{bv_i}}{\left|\overrightarrow{bv_i}\right|} \tag{3.2}$$

onde $b$ é o ponto de posicionamento da barragem, $v_i$ é o ponto à montante de $b$, para $i$ variando entre [1,$n$] e $bv_i$ o vetor conectando esses dois pontos tendo seu módulo representado por $\left|\overrightarrow{bv_i}\right|$. O valor de $n$ é definido pelo usuário. E o valor do peso $P_i$, associado a cada vetor $bv_i$ é dado por $P_i = n - i + 1$. O processo é apresentado de forma mais detalhada na Figura 3.4.

Por outro lado, o DIRDP é uma técnica que utiliza o conjunto de segmentos de retas

**Require:** $p$ //ponto onde posicionar a barragem
1: $p_i$ //ponto à montante de $p$, $p_0$==$p$
2: $n$ //quantidade de vizinhos a serem processados
3: $i$ //índice do elemento sendo processado
4: $r$ //vetor soma
5: $\|p_ip\|$ //módulo do vetor $p_ip$
6:
7: $i \leftarrow 1$
8: $r.x \leftarrow 0$
9: $r.y \leftarrow 0$
10:
11: **while** $i \leq n$ **do**
12: $\|p_ip\| \leftarrow \sqrt{(p_i.x - p.x)^2 + (p_i.y - p.y)^2}$
13: $r.x \leftarrow r.x + (p_i.x - p.x)/\,|p_ip|$
14: $r.y \leftarrow r.y + (p_i.y - p.y)/\,|p_ip|$
15: $i \leftarrow i + 1$
16: **end while**
17: **return** $r$

Figura 3.3: Algoritmo DIRMS que define direção da barragem

**Require:** $p$ //ponto onde posicionar a barragem
1: $p_i$ //ponto à montante de $p$, $p_0$==$p$
2: $n$ //quantidade de vizinhos a serem processados
3: $i$ //índice do elemento sendo processado
4: $r$ //vetor soma
5: $s$ //soma dos pesos
6: $\|p_ip\|$ //módulo do vetor $p_ip$
7:
8: $i \leftarrow 1$
9: $r.x \leftarrow 0$
10: $r.y \leftarrow 0$
11: $s \leftarrow 0$
12:
13: **while** $i \leq n$ **do**
14: $m_i \leftarrow \sqrt{(p_i.x - p.x)^2 + (p_i.y - p.y)^2}$
15: $r.x \leftarrow r.x + ((p_i.x - p.x)/\,|p_ip|)(n - i + 1)$
16: $r.y \leftarrow r.y + ((p_i.y - p.y)/\,|p_ip|)(n - i + 1)$
17: $i \leftarrow i + 1$
18: **end while**
**return** $r$

Figura 3.4: Algoritmo DIRMP que define direção da barragem

formados pelos $n$ vizinhos à montante do ponto de posicionamento da barragem, tomados dois a dois, formando assim uma isolinha. Esta técnica é composta por um processo de simplificação da isolinha seguido pela técnica DIRMS. O processo de simplificação é realizado através do algoritmo de *Douglas-Peucker* [15, 19], utiliza $n$ pontos e uma dada tolerância para simplificação da isolinha.

Ou seja, define-se inicialmente um segmento de reta $\overline{pq}$ composto pelo primeiro ponto ($p$) e o ultimo ponto ($q$ que seria o $n$-ésimo ponto à montante do ponto de posicionamento da barragem). Caso todos os $n$ pontos estejam a uma distancia menor que a tolerância permitida, o vetor solução será dado por $\overline{pq}$. Caso contrário, se há algum ponto com distância maior que a tolerância, este segmento $\overline{pq}$ será subdividido em dois segmentos de reta, $\overline{pm}$ e $\overline{mq}$, onde $m$ é o ponto que tem a maior distância ao segmento $\overline{pq}$, e então, o algoritmo reinicia para cada novo segmento até que não se tenha pontos com tolerância maior do que a permitida.

Após o processo de simplificação, aplica-se o método DIRMS sobre esta isolinha simplificada, com $n = 1$, e assim, obtém-se o vetor solução.

Finalmente, após o cálculo do vetor $\overrightarrow{r} = (r_x, r_y)$ através dos métodos descritos acima, a direção da barragem será o vetor $\overrightarrow{b} = (b_x, b_y)$ tal que $\overrightarrow{r} \perp \overrightarrow{b}$. E como o modelo de representação adotado é discreto, este vetor $\overrightarrow{b}$ terá sua direção dada para um dos valores apresentados na Figura 3.1.

## 3.2 O processo de definição de um reservatório hídrico

Para se realizar análises e processamentos computacionais sobre a hidrologia de certa região, é necessário que a representação digital do terreno contenha a rede de drenagem. O trabalho em questão faz uso do método RWFlood, descrito na Seção 2.3.1 para cálculo da rede de drenagem. Além disso, utiliza-se uma ideia semelhante àquela adotada pelo RWFlood para determinar a região a ser alagada por uma barragem.

O processo de obtenção do reservatório consiste, basicamente, na determinação da altura e extensão da barragem, obtendo assim a área alagada. Porém, nesse estudo o volume é o elemento essencial no processo de obtenção do reservatório, pois, a finalidade do processo é que o reservatório adquirido retenha uma demanda hídrica capaz de regularizar a vazão numa região. Com isto, dados o ponto onde a barragem deve ser construída e o volume desejado, o objetivo é determinar a altura e extensão da barragem que deve ser construída para que seja obtido o reservatório desejado.

A maioria dos métodos existentes na literatura, como os descritos em [37, 35], associam este problema à determinar a componente conexa que forma a região alagada. Nesse caso, a componente conexa é definida como sendo um conjunto de células que estão co-

nexas a pelo menos um ponto do rio, estão à montante da barragem e possuem elevação inferior à altura da barragem. Daí, para se identificar a componente conexa que forma a região alagada, usa-se, em geral, o algoritmo de busca em largura.

Para obter a altura adequada da barragem, este processo é repetido até que o volume da área alagada alcance o volume desejado. Mais precisamente, a obtenção da altura e extensão da barragem se baseia num processo repetitivo em que, a cada passo, é definido um valor para a altura e a extensão da barragem e se determina o volume da região que essa barragem é capaz de gerar. Caso este não atenda o objetivo, um novo valor para a altura e para extensão é definido e o processo é repetido. Por questões de eficiência, o método proposto em [37] se baseia no método de busca binária para definir a altura e a extensão da barragem.

Para evitar a ineficiência do processo repetitivo para obtenção das características da barragem e área alagada, nesse trabalho é proposto um novo método, denominado ALAGA, que define a barragem com menor altura e extensão que satisfaça a localização e capacidade desejada, sem a necessidade de refazer o processo inúmeras vezes.

O ALAGA se baseia no algoritmo RWFlood, que foi adaptado para alterar o(s) ponto(s) por onde a inundação é desencadeada. Isto é, no RWFlood a inundação ocorre pelas bordas do terreno e no ALAGA, a inundação é desencadeada pelo ponto pertencente à rede de drenagem, adjacente e à montante ao ponto escolhido para posicionamento da barragem. É como se o ponto de início fosse o ponto de entrada da água para inundar a região.

Em outras palavras, o ALAGA determina a área alagada, altura e a extensão da barragem necessária para se gerar um reservatório com capacidade desejada utilizando os dados fornecidos pelo usuário. Mais precisamente, dados o MDE do terreno, a rede de drenagem, o ponto $p$, onde posicionar a barragem e o volume desejado para o reservatório a ser gerado, o algoritmo determina o ponto $p'$, à montante de $p$, onde o processo de alagamento é iniciado. O nível da água é elevado gradativamente inundando $p'$, posteriormente seus vizinhos, os vizinhos dos vizinhos, e assim por diante. Cada célula somente será inundada caso o valor de sua elevação seja inferior ao nível corrente da água e que esteja na iminência de inundação. Por iminência entende-se a célula adjacente a uma célula já inundada. Assim como no RWFlood, este algoritmo usa uma fila de prioridades para armazenar as próximas células a serem inundadas, sendo que o primeiro elemento da fila é a célula de menor elevação não alagada até o momento. À medida que uma célula é inundada, os seus vizinhos que ainda não tiverem sido inseridos na fila são inseridos. O processo termina quando a capacidade desejada for satisfeita.

A altura, a extensão e a área alagada, definidas pelo reservatório, são obtidas no final do processo de alagamento. Quando uma célula é inundada, a área da célula é acrescida à área alagada e, caso necessário, a extensão da barragem é incrementada incluindo novas células (na direção da barragem) que são adjacentes à área alagada, a fim de manter o

processo de retenção. Além disso, o valor para a altura da barragem será o mesmo valor do nível da água. Assim, a extensão e altura da barragem terão o tamanho suficiente para reterem a água. Este processo é descrito mais detalhadamente no algoritmo apresentado na Figura 3.5. Vale lembrar que ao se alagar uma determinada região as células adjacentes ao alagamento e com elevação inferior ao nível da água serão necessariamente alagadas e, assim, o valor da capacidade do reservatório gerado poderá ser superior à capacidade desejada.

## 3.3 A função objetivo

Para avaliar os reservatórios gerados por cada barragem posicionada em um determinado ponto, o método de posicionamento utiliza uma função objetivo que leva em consideração os seguintes fatores:

- Extensão da barragem: é definida pelo ALAGA sendo composta pelas células que são adjacentes às células alagadas e pertencentes à reta que estabelece a direção da barragem.
- Altura da barragem: é definida pelo ALAGA e possui o mesmo valor do nível da água para o reservatório gerado (valor dado em relação ao nível do mar).
- Área de alagamento: área do terreno cujas células são alagadas. Também definida pelo ALAGA.
- Regiões críticas ou áreas de preservação: é importante avaliar a região que irá ser alagada. Para isso, o usuário define as regiões e o valor de impacto de alagamento (peso) dessas regiões, pois pode ser interessante definir regiões mais propícias ao posicionamento da barragem.

  Nesse trabalho, a função objetivo é definida por:

$$f(P) = \alpha A + \theta E_B + \beta H_B + \omega A_B + C \tag{3.3}$$

onde $A$ é a área alagada, $E_B$ é a extensão da barragem, $H_B$ é a altura da barragem, $A_B$ é a área da barragem e $C$ é o valor de impacto das regiões críticas. As constantes $\alpha$, $\gamma$, $\theta$, $\beta$, $\omega$ definem um peso para cada variável. Por padrão, estas constantes assumem valores pré-definidos e o usuário pode alterá-las para adequar a função objetivo, e assim qualificar o modo de impacto de cada variável.

A variável $C$ da Equação 3.3 é obtida através do somatório dos valores de impacto de alagamento (definido pelo usuário) para cada célula que esteja alagada. Quando se diz valores de impacto é porque uma determinada célula alagada pode pertencer a uma ou mais regiões definidas pelo usuário.

**Require:** $k$ //capacidade desejada
1: $p$ //ponto onde posicionar a barragem
2: $p'$ //vizinho à montante a $p$, o processo começa por este ponto
3: $Q$ //fila de prioridades ordenada crescentemente por elevação de células (contém células iminentes à inundação)
4: $h$ //nível da água
5: $elevacao(x)$ //elevação da célula $x$ do MDE
6: $insereNaFila(x)$ //insere o elemento $x$ na fila Q ordenadamente
7: $insereVizinhos(x)$ //insere na fila os vizinhos de $x$ não processados e que estão à montante da barragem
8: $removeDaFila()$ //remove e retorna o primeiro elemento da fila
9: $capacidade$ //volume do reservatório para a inundação corrente
10: $areaAlagada$ //area alagada do reservatório para a inundação corrente
11: $area(x)$ //area da celula $x$
12: $B$ //lista contendo as células pertencentes à barragem. O tamanho da lista define a extensão da barragem
13: $aumentaBarragem(x)$ // verifica se deve aumentar a barragem caso se inunde $x$. Retorna o elemento a ser considerado barragem
14: $Q.insereNaFila(p')$
15: $B.insere(p)$
16: $h \leftarrow elevacao(p')$
17: $areaAlagada \leftarrow 0$
18: $capacidade \leftarrow 0$
19: **while** $capacidade < k$ **do**
20: $q \leftarrow Q.removeDaFila()$
21: $qB \leftarrow aumentaBarragem(q)$
22: **if** $qB \neq NULL$ **then**
23: $B.insere(qB)$
24: **end if**
25: $Q.insereVizinhos(q)$
26: **if** $elevacao(q) > h$ **then**
27: $capacidade \leftarrow capacidade + (elevacao(q) - h) \times areaAlagada$
28: $h \leftarrow elevacao(q)$ //Suba o nível da água
29: **else**
30: $capacidade \leftarrow capacidade + (h - elevacao(q)) \times area(q)$
31: **end if**
32: $areaAlagada \leftarrow areaAlagada + area(q)$
33: **if** $Q.vazia()\&\&capacidade < k$ **then**
34: ERRO
35: **end if**
36: **end while**

Figura 3.5: Algoritmo para inundação e cálculo da represa

Formalmente, seja o conjunto $RA$ contendo as células alagadas do terreno MDE e sejam os conjuntos $RC_i$ (com $i = [0,n]$, onde $n$ é número de regiões definidas pelo usuário) as células do MDE consideradas regiões críticas de alagamento. Seja ainda $p_i$ o impacto, ou peso, de se alagar células na região $i$. Assim, o valor de $C$ é:

$$C = \sum_{i=0}^{n} p_i * n\left(RA \cap RC_i\right) \tag{3.4}$$

onde $n(\mathbf{A})$ é a cardinalidade (número de elementos) do conjunto $\mathbf{A}$

## 3.4 Algoritmo de posicionamento

Neste trabalho, foi desenvolvido um algoritmo que, dados um determinado trecho de uma rede de drenagem e a capacidade desejada para o reservatório a ser construído, além de algumas outras possíveis restrições, determina o ponto mais adequado para a construção de um reservatório. Ele foi denominado RWDamming, utiliza a função objetivo, descrito na Seção 3.3, para qualificar cada reservatório determinando o custo de construção do mesmo e utiliza também o ALAGA para realizar o processo de alagamento da região.

Esse método avalia todos os candidatos, ou seja, todos os pontos da rede de drenagem, para o posicionamento do reservatório dentro do trecho determinado e adota algumas estratégias para reduzir o tempo de execução. A primeira estratégia utilizada é a técnica de *Branch-and-Bound* (ramificar e limitar) [23], que visa limitar o processo de avaliação do candidato caso não seja possível atingir uma determinada restrição ao se construir o reservatório. Ou seja, ao se avaliar um candidato, caso o custo de sua construção, determinado pela função objetivo, ultrapasse o custo do melhor reservatório até o momento, o processo de construção deste reservatório é interrompido, este candidato é descartado e o próximo candidato é processado. Com isto, evita-se que os candidatos que não poderiam ser o ponto de posicionamento mais adequado sejam processados completamente, e assim, sendo avaliados completamente apenas os candidatos que são mais adequados do que o ponto de posicionamento mais adequado até o momento da busca.

Uma matriz compartilhada entre os reservatórios gerados é utilizada para armazenar as informações dos reservatórios processados. Ou seja, esta matriz contém em cada célula a informação (dada em valor numérico) se a mesma é barragem, rede de drenagem, região de alagamento ou terreno para o reservatório. Assim, para se distinguir as células do reservatório processado foi utilizado um rótulo numérico associado a cada reservatório processado. Ou melhor, o valor de cada célula da matriz compartilhada será proporcional a este rótulo do reservatório, tornando-a distinta. Essa otimização previne que a matriz tenha suas células reinicializadas a cada novo processo de construção de reservatório e isto garante que $n \times numLinhas \times numColunas$ ciclos computacionais sejam evitados,

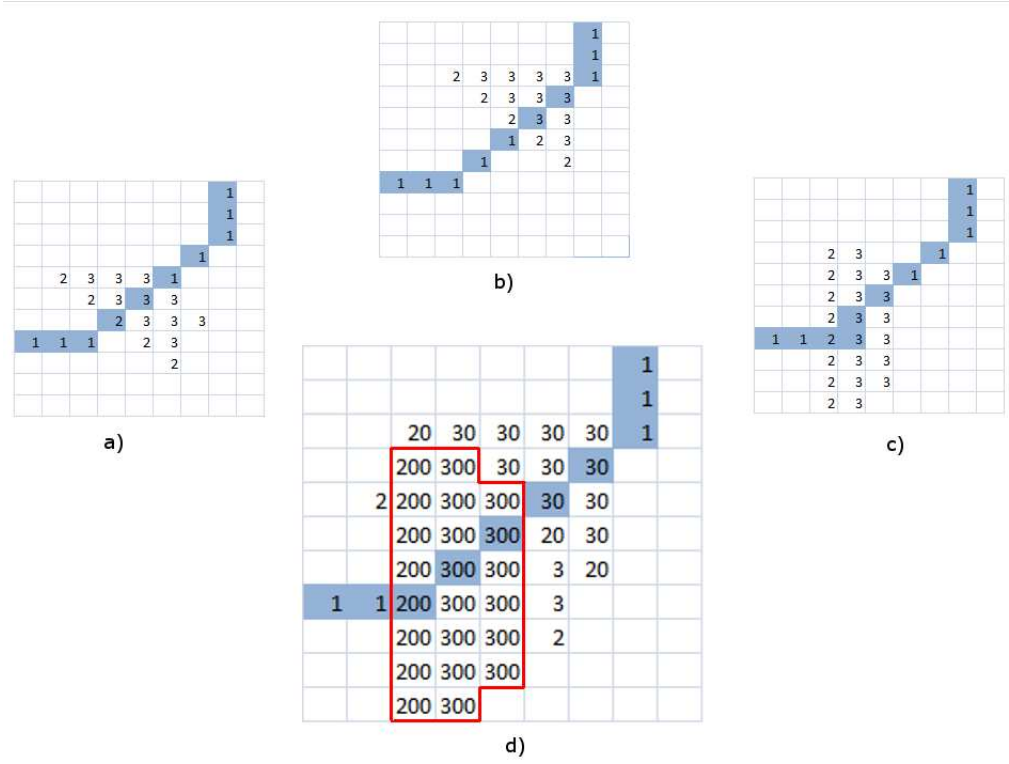



Figura 3.6: Representação de matriz compartilhada (d) e de matrizes não compartilhada (a,b,c).

onde $n$ é o número de candidatos a serem avaliados, *numLinhas* é o número de linhas e *numColunas* é o número de colunas.

As Figuras 3.6a, 3.6b e 3.6c apresentam o comportamento para matrizes que não são compartilhadas no processo de inundação e a Figura 3.6d apresenta um exemplo de matriz compartilhada onde cada inundação tem fator proporcional a base 10, o valor 1 representa o rio sem sofrer inundação, 2 a barragem e 3 a área alagada devido a retenção. A inundação corrente é identificada por uma linha que a contorna. Assim, nesta inundação corrente, caso fosse necessário avaliar mais alguma célula, apenas as não proporcionais ao fator desta inundação seriam avaliadas, evitando assim que seja necessário uma reinicialização prévia de todas as células da matriz.

Foram desenvolvidos outros dois algoritmos de posicionamento: RWDammingO1 e RWDammingSO; semelhantes ao RWDamming, porém, sem algumas das estratégias adotadas. Da mesma forma que o RWDamming, estes algoritmos de posicionamento utilizam o ALAGA e a função objetivo descritos nesse documento. Tais métodos foram desenvolvidos a fim de se analisar o ganho de desempenho obtido pelas estratégias adotadas pelo RWDamming. Assim, os algoritmos de posicionamento resumem-se em:

- RWDamming: Algoritmo de posicionamento com todas as estratégias expostas.

- RWDammingO1: Algoritmo de posicionamento com uso de rótulo para reservatórios apenas.
- RWDammingSO: Algoritmo de posicionamento sem o uso de nenhuma estratégia para a melhoria de eficiência.

## 3.5 Sistema com interface gráfica

No trabalho em questão foi desenvolvida uma versão do sistema de posicionamento automático com interface gráfica. A finalidade desse sistema era permitir a análise visual de diversos reservatórios hídricos gerados para tentar estabelecer um algoritmo de posicionamento automático da barragem que não avaliasse todos os pontos. Ou seja, serviria de suporte para avaliar os algoritmos e expor as soluções através de imagens dos reservatórios gerados. Por fim, o mesmo foi incorporado como resultado final do trabalho, por apresentar resultados satisfatórios.

O sistema foi denominado Re-Build (Construção de Reservatório ou "Reservoir Building") e está disponível em [16]. A Figura 3.7 apresenta a tela principal do sistema e um terreno sendo visualizado. Neste sistema o usuário pode optar também por definir manualmente o local de posicionamento da barragem. Os resultados referentes ao posicionamento da barragem são apresentados em imagens e números, após a inserção de alguns parâmetros. Na parte central do sistema tem-se uma área, denominada "área de visualização", onde pode-se visualizar o terreno e o reservatório gerado, seja pelo posicionamento automático ou manual da barragem. À direita desta área de visualização são apresentados, em um formulário, as características do reservatório: extensão e altura da barragem, área alagada e volume. Também são apresentados a posição da barragem e ponteiro do mouse relativos ao próprio MDE do terreno e o valor do fluxo mínimo para o mapa corrente (visualizado).

Há ainda algumas funcionalidades que contribuem para a decisão e formação do reservatório hídrico desejado pelo usuário e que se encontram à esquerda da área de visualização. O valor do limite para a formação da rede de drenagem[1] pode ser redefinido para alterar a rede de drenagem. Existe uma área neste sistema, à esquerda da área de visualização, denominada "área de projeto", que permite ao usuário gerenciar reservatórios: salvar, excluir e visualizar reservatórios já processados. Essa funcionalidade permite ao usuário analisar os vários reservatórios obtidos e decidir o reservatório mais viável visualmente. Os parâmetros de entrada essenciais para a construção do(s) reservatório(s), valor para capacidade e o método (definido na Seção 3.1) para obtenção da direção da barragem, estão também a esquerda da área de visualização.

[1] A rede de drenagem é composta pelas células cujo o fluxo acumulado é maior do que o limite para a formação da rede de drenagem

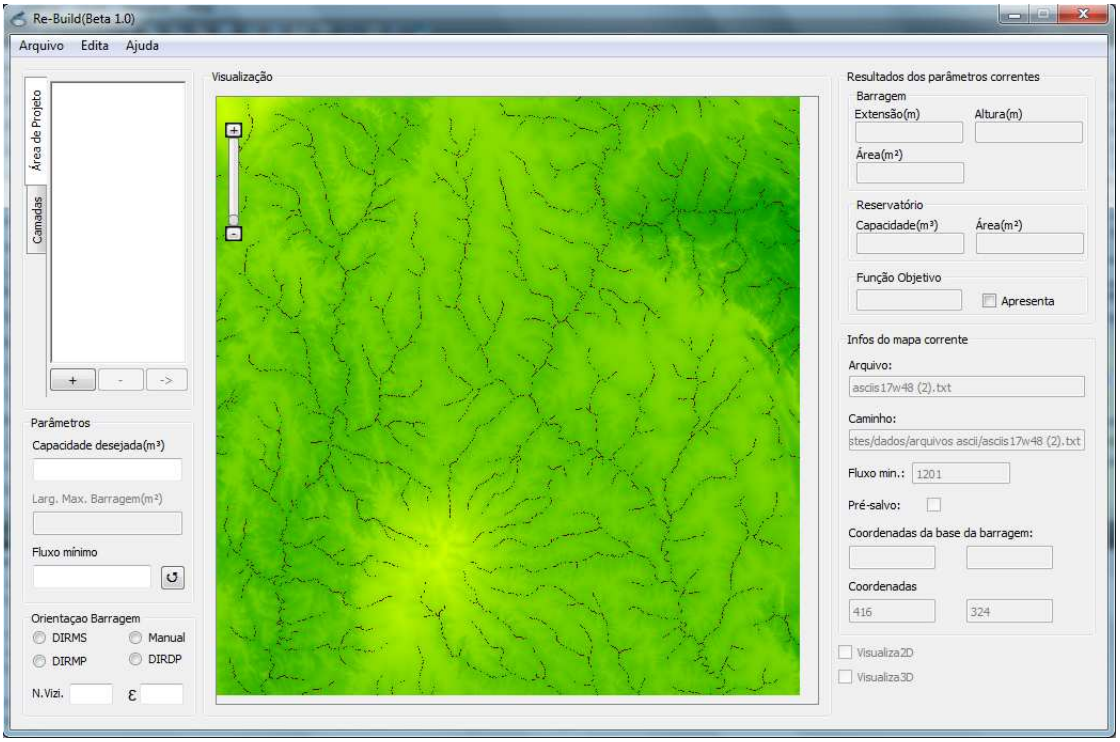



Figura 3.7: Interface gráfica do sistema

Como mencionado, o usuário pode definir um local (com um clique do botão esquerdo do mouse sobre a rede de drenagem) ou trecho (com dois cliques do botão direito do mouse sobre a rede de drenagem) para o posicionamento da rede de drenagem. Caso a opção seja por um trecho, o algoritmo descrito na Seção 3.4 calcula as opções possíveis de posicionamento e define a mais adequada segundo a função objetivo (Equação 3.3) e expõe o resultado ao usuário na tela.

As constantes definidas nesta função podem ter seus valores alterados pelo usuário, através de uma funcionalidade acessada pelo menu "editar". O parâmetro referente às regiões críticas (ou a variável $C$ definida na Equação 3.1), é obtido através da soma dos pesos de cada célula das regiões críticas e que são alagadas. Este peso, definido no momento em que se adiciona a região crítica, é o valor de penalidade para o possível alagamento das células. No sistema, há uma área, denominada "regiões críticas", onde o usuário pode gerenciar (adicionar, excluir, visualizar) as regiões críticas do sistema. Em cada região, o usuário define as células que são críticas, o peso e a ainda a cor, para diferenciar de outras regiões. Caso não seja inserida nenhuma região crítica, o algoritmo de posicionamento assume que $C = 0$. Na Figura 3.8 pode ser observado um terreno com várias regiões críticas.

Existem algumas funcionalidades disponíveis nessa versão que foram implementadas para fins de teste cuja interface está muito simples e merece uma melhoria em uma versão posterior. São elas:

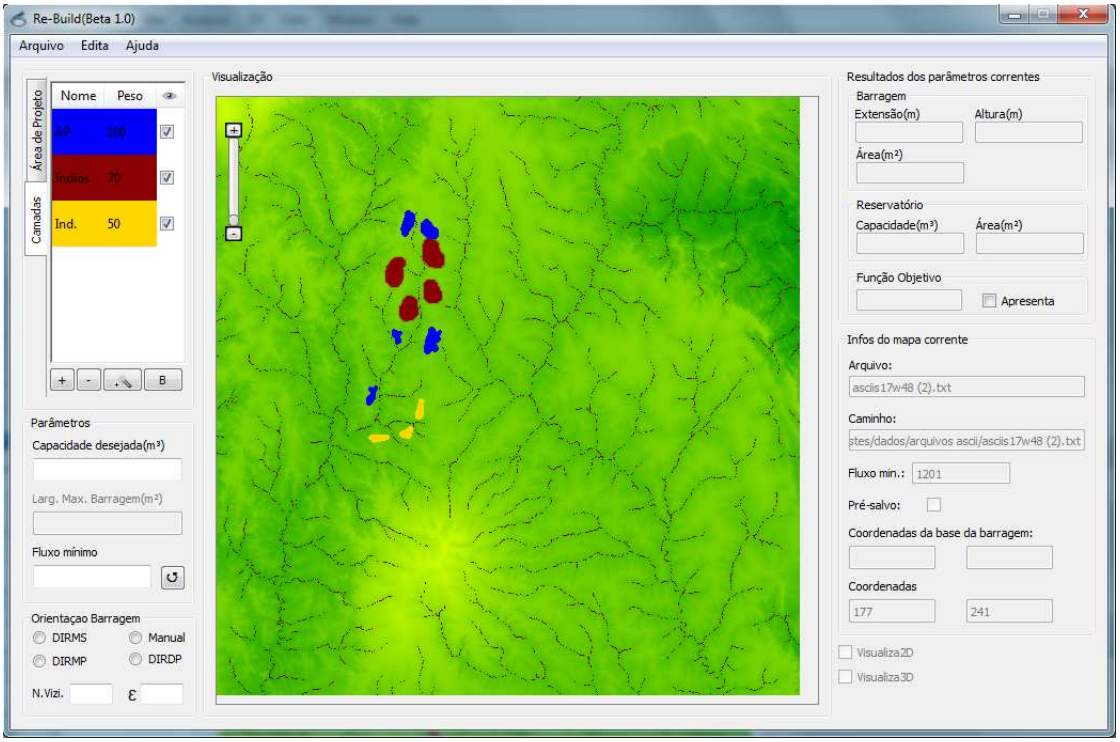



Figura 3.8: Interface gráfica do sistema com uso de regiões críticas

- Visualização 2D
- Visualização 3D
- Valor máximo da extensão da barragem

A funcionalidade visualização 2D, foi desenvolvida para se avaliar as características da barragem através de imagens de perfil da mesma, através de uma janela extra criada pelo sistema Re-Build. Esta funcionalidade permite avaliar visualmente padrões das diversas barragens geradas ao longo da rede de drenagem. A Figura 3.9 apresenta uma imagem do sistema ao se gerar um reservatório hídrico com a visualização 2D ativa. Nesta visualização são mostradas as células que acomodarão a barragem (e suas elevações), cuja(s) célula(s) em preto indica(m) o local onde o rio passa, antes de acontecer o alagamento. No caso da visualização 3D a ideia é apresentar o terreno e o local do reservatório, através de uma janela extra também e permitir diversos ângulos de visualização.

O sistema foi desenvolvido utilizando o framework Qt [31], que é um sistema *open-source* multi plataforma (*Windows, Windows CE, Symbian, OS X, Linux, entre outras*), amplamente utilizado em desenvolvimento de aplicativos com interface gráfica. A linguagem utilizada foi C++. O programa foi compilado utilizando MinGW/gcc 4.4 [26].

A documentação desse sistema é apresentada nos Anexo A e Anexo B onde pode-se encontrar a especificação de requisitos, com informações técnicas para a construção

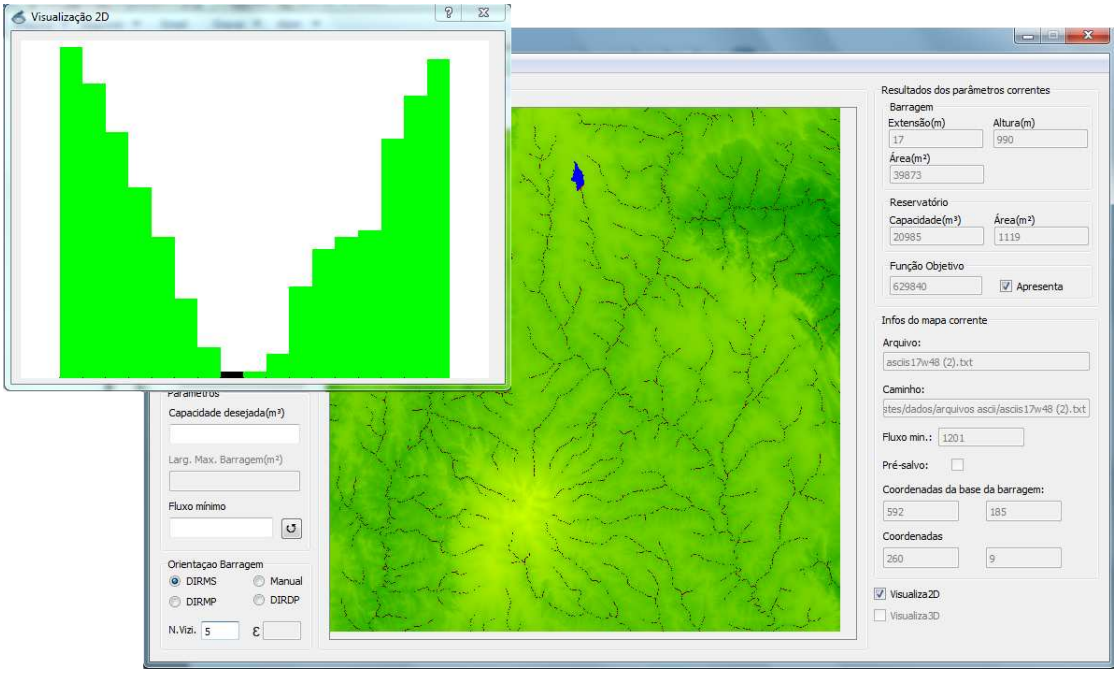



Figura 3.9: Interface gráfica do sistema com visualização do terreno de perfil

do sistema (Anexo A). E para efeitos de manutenção tem-se um documento gerado pela ferramenta *qdoc* [32], do próprio `Qt` (Anexo B). Esta ferramenta é um gerador automático de documentação de códigos com formato padrão de saída HTML.

# Capítulo 4

# Resultados

## 4.1 Análise do RWDamming

Os testes realizados tentam abranger duas situações para validar o algoritmo RWDamming. O objetivo da primeira situação (Análise 1) é avaliar as otimizações propostas contidas no RWDamming. Para isso, foram utilizados outros dois algoritmos (RWDammingO1 e RWDammingSO) como comparação. Na segunda situação (Análise 2), o algoritmo RWDamming é avaliado comparando-o com outro algoritmo proposto na literatura, denominado JRPos [37]. Além do mais, como o método proposto em [35] não contém um método de posicionamento automático de barragem, o mesmo não foi analisado neste estudo.

Para casos onde é determinado um simples trecho da rede de drenagem, os tempos de processamento foram pequenos em ambos os algoritmos, e assim, realizar uma análise torna-se difícil. Com isto, adotou-se que todos os pontos de toda a rede de drenagem seriam avaliados.

Para realização dos testes foram utilizados dados extraídos do *Shuttle Radar Topography Mission* (SRTM) [30] com resolução de 30m.

Na Análise 1 foram utilizados terrenos das regiões 1 e 2 (Figura 4.1). Foram processados terrenos com tamanhos de 100×100, 500×500, 1000×1000, 5000×5000, 10000×10000 células, e para cada dimensão foram gerados reservatórios com capacidades de 100000, 500000, 1000000$m^3$ (0.1, 0.5 ou 1$hm^3$, respectivamente). As capacidades adotadas foram baseadas nas informações dadas na Seção 2.2.3, onde pode ser visto que tais capacidades estão enquadradas no conjunto que exprime maior número de barragens do Brasil.

Na Análise 2, foi escolhida a região de coordenadas 20°48" de latitude sul e 42°53" de longitude oeste, que corresponde à micro bacia do córrego do Paraíso, inserida na bacia hidrográfica do ribeirão São Bartolomeu, no município de Viçosa−MG, conforme pode ser vista na Figura 4.2. Nesta análise, o algoritmo RWDamming, proposto nesse trabalho, foi comparado ao método exposto em [37], denominado nesse estudo por JRPos. Neste

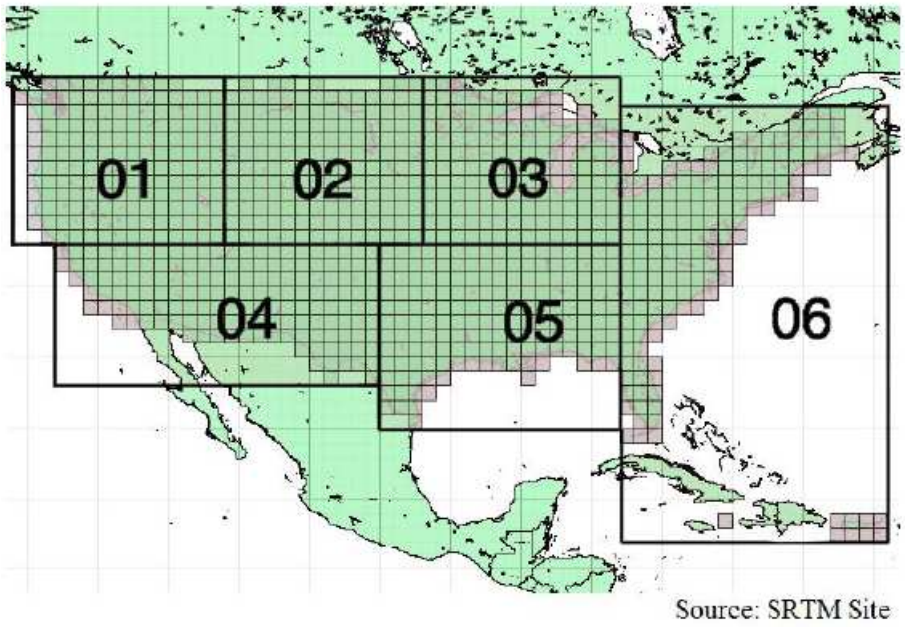



Figura 4.1: Regiões dos EUA com dados de elevações disponibilizados pelo SRTM

caso, foi utilizado um terreno com dimensões 1201×1201 células para as capacidades de reservatório com 100, 500, 1000, 5000, 10000, 100000, 1000000, 5000000, 10000000m$^3$.

Como a função objetivo deste estudo contém variáveis que não são aplicáveis ao método a ser comparado na Análise 2, foi padronizado que apenas as variáveis de extensão da barragem e área alagada seriam utilizadas em ambas análises. Considera-se ambas as análises, pois, na Análise 1 a variável de região crítica e altura da barragem não afetariam a análise de um modo geral, assim apenas a versão com interface gráfica contém a função objetivo com todas as variáveis aplicáveis.

Como o foco principal do estudo era avaliar a velocidade em se obter a barragem mais qualificada, os pesos associados as variáveis presentes na função objetivo para avaliação não seguiram um critério técnico. E assim, adotou-se os valores de 1 e 0.5 para os pesos do comprimento da barragem e da área alagada respectivamente em cada processo de criação da barragem.

Todos os testes foram realizados utilizando um computador com processador Core 2 Duo 2,8GHZ, 1GB de memória RAM e sistema operacional Linux Ubuntu 11.04 32bits. Foram realizados testes de desempenho sobre o algoritmo de posicionamento (não incluindo interface gráfica). Além disso, não foram estipuladas camadas (regiões críticas).

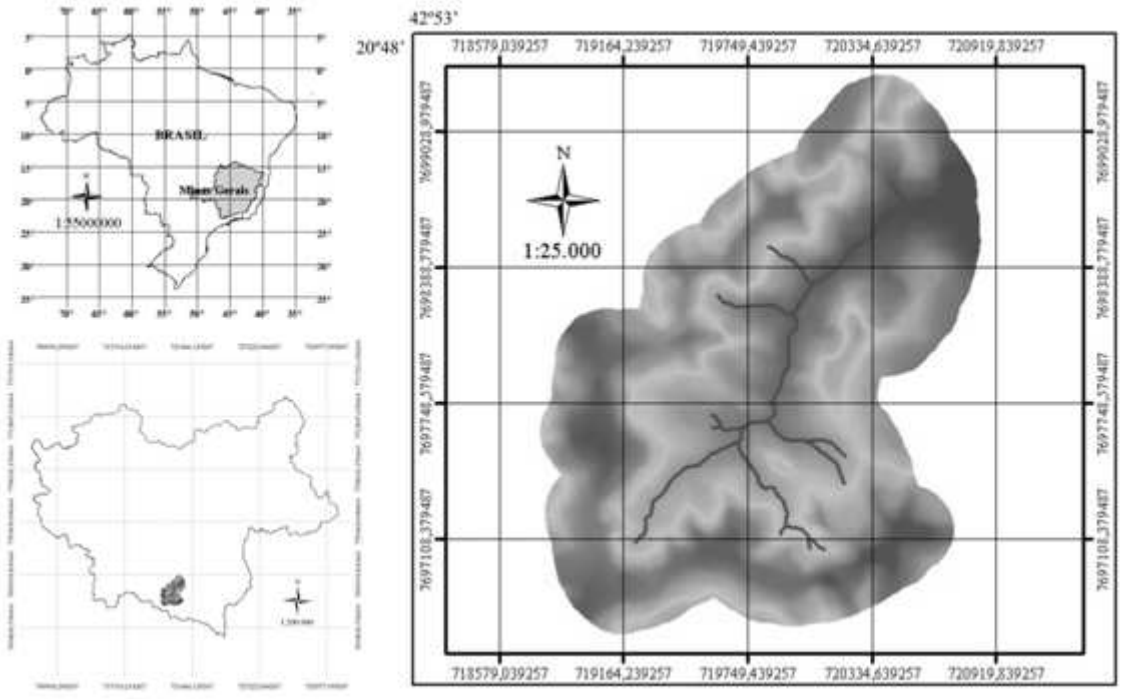



Figura 4.2: Localização da região dos dados aplicados a "Análise 2"

## 4.2 Resultados para a Análise 1

Como o número de pontos a serem processados afeta diretamente a eficiência dos métodos avaliados, a Tabela 4.1 apresenta o número de pontos avaliados para cada região.

As Tabelas 4.2 e 4.3 apresentam os resultados dos tempos gastos pelos métodos para as regiões R2 e R3 respectivamente. Observe que tanto para a região R2 quanto para a R3, quando se está processando terrenos de tamanho 10000×10000 e capacidade superior a 100000m$^3$, apenas o algoritmo RWDamming conseguiu processar todos os candidatos e identificar a barragem mais adequada num tempo relativamente aceitável, os demais tiveram seus tempos superiores a 7 dias (estão demarcados na tabela por "*"). Vale ressaltar que o algoritmo RWDammingO1 para o terreno com tamanho 10000×10000 e capacidade 100000m$^3$ não ultrapassou 7 dias de processamento, mas obteve um tempo próximo a este.

Para os terrenos com 100×100 células não foi possível gerar nenhum reservatório segundo a capacidade desejada. Nestes casos, para todos os reservatórios gerados através do processo de busca, o terreno era totalmente alagado e a capacidade desejada não atingida. E assim, como não é possível prever o comportamento fora dos limites do terreno, estes processamentos são descartados.

Além de analisar o tempo gasto pelo método RWDamming, deve-se analisar a qualidade e as características do reservatório obtido. Para os métodos avaliados é necessário que pelo menos os valores obtidos em suas funções objetivo sejam equivalentes, pois todos minimizam esta função. Veja Tabela 4.4 que apresenta as características, utilizadas ou não

Tabela 4.1: Quantidade de pontos avaliados por região e tamanho do terreno

| **Região** | **Tamanho** ( ♯ células) | **# Pontos Avaliados** |
|---|---|---|
| R2 | $100^2$ | 406 |
| | $500^2$ | 6833 |
| | $1000^2$ | 21730 |
| | $5000^2$ | 245140 |
| | $10000^2$ | 698767 |
| R3 | $100^2$ | 482 |
| | $500^2$ | 6423 |
| | $1000^2$ | 18064 |
| | $5000^2$ | 218473 |
| | $10000^2$ | 635310 |

Tabela 4.2: Resultados para região 2

| **Capacidade** ($m^3$) | **Tamanho** ( ♯ células) | **Tempo de processamento (em segundos)** | | |
|---|---|---|---|---|
| | | **RWDamming** | **RWDammingO1** | **RWDammingSO** |
| 100000 | $500^2$ | 27,63 | 28,96 | 48,88 |
| | $1000^2$ | 97,46 | 114,56 | 372,57 |
| | $5000^2$ | 3881,98 | 40220,31 | 110446,49 |
| | $10000^2$ | 11198,12 | 576367,24 | * |
| 500000 | $500^2$ | 34,41 | 34,57 | 54,01 |
| | $1000^2$ | 149,30 | 149,63 | 407,65 |
| | $5000^2$ | 17843,72 | 60312,73 | 130640,87 |
| | $10000^2$ | 59632,90 | * | * |
| 1000000 | $500^2$ | 36,23 | 36,94 | 55,75 |
| | $1000^2$ | 159,88 | 162,80 | 417,25 |
| | $5000^2$ | 30545,13 | 72086,46 | 147990,70 |
| | $10000^2$ | 113793,31 | * | * |

Tabela 4.3: Resultados para região 3

| **Capacidade** | **Tamanho** | **Tempo de processamento (em segundos)** | | |
|---|---|---|---|---|
| ($m^3$) | ( ♯ células) | RWDamming | RWDammingO1 | RWDammingSO |
| 100000 | $500^2$ | 70,00 | 81,89 | 100,70 |
| | $1000^2$ | 302,03 | 383,63 | 599,74 |
| | $5000^2$ | 5266,47 | 39465,70 | 111243,31 |
| | $10000^2$ | 15556,31 | 572631,80 | * |
| 500000 | $500^2$ | 91,18 | 90,43 | 109,09 |
| | $1000^2$ | 427,61 | 428,31 | 643,65 |
| | $5000^2$ | 21254,27 | 66837,62 | 131171,75 |
| | $10000^2$ | 71560,23 | * | * |
| 1000000 | $500^2$ | 93,26 | 91,14 | 109,19 |
| | $1000^2$ | 435,02 | 435,89 | 650,75 |
| | $5000^2$ | 37122,32 | 81580,25 | 148322,71 |
| | $10000^2$ | 135055,55 | * | * |

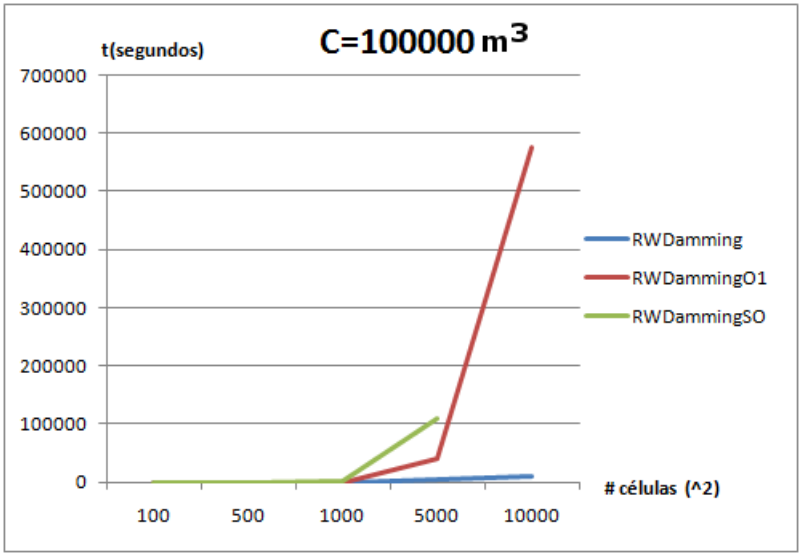
Tabela 4.4: Características das melhores barragens para a região 2

| **Capacidade** | **Tamanho** | RWDamming/RWDammingSO | | | **Func.Obj.** |
|---|---|---|---|---|---|
| ($m^3$) | ( ♯ células) | **A($m^2$)** | **V($m^3$)** | **$E_B$(♯ cel)** | |
| 100000 | $500^2$ | 1374 | 100233 | 73 | 1410 |
| | $1000^2$ | 1118 | 100107 | 61 | 1148 |
| | $5000^2$ | 564 | 100119 | 25 | 576 |
| 500000 | $500^2$ | 9179 | 500673 | 69 | 9213 |
| | $1000^2$ | 4570 | 500066 | 156 | 4648 |
| | $5000^2$ | 1995 | 50091 | 76 | 2033 |
| 1000000 | $500^2$ | 21090 | 1012974 | 101 | 21140 |
| | $1000^2$ | 6982 | 1000231 | 174 | 7609 |
| | $5000^2$ | 3989 | 1000313 | 100 | 4039 |

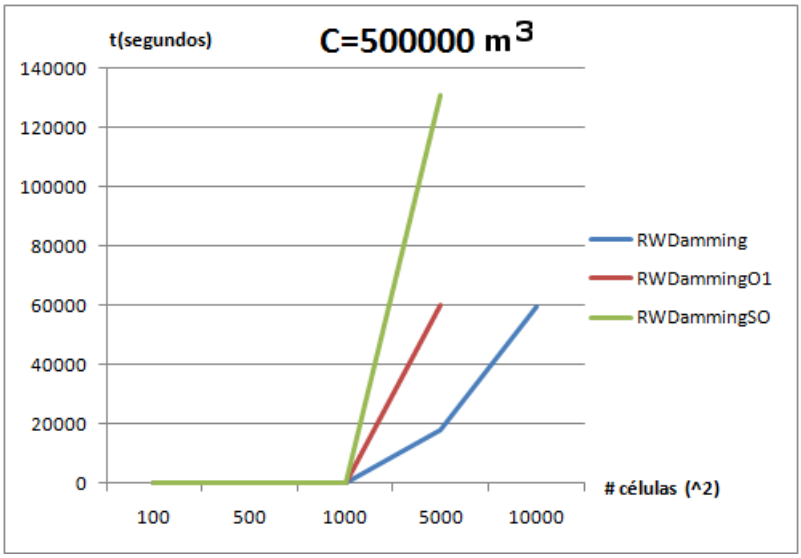
na função objetivo, da barragem mais adequada obtidas pelos algoritmos RWDamming e RWDammingSO, onde a coluna "A" representa a área em $m^2$ da região alagada, "V" é o volume em $m^3$, "$E_B$" é a extensão da barragem dada em número de células e "Func.Obj." é a função objetivo.

A Figura 4.3 apresenta o gráfico contendo a comparação entre os métodos expostos, para os resultados sobre a região R2. Sendo que a Figura 4.3a apresenta o gráfico contendo os resultados para a capacidade $C = 100000m^3$, a Figura 4.3b contém o gráfico para a capacidade $C = 500000m^3$ e a Figura 4.3c contém o gráfico para a capacidade $C = 1000000m^3$. Observe que, como nas tabelas de resultados, alguns resultados não são apresentados nos gráficos, pois têm um valor superior a 7 dias e isto é algo inviável para o propósito do trabalho. Ou seja, enquanto que o método RWDamming realiza o processo para capacidade $C = 1000000m^3$ em um tempo inferior à 2 dias os demais ultrapassam 7 dias. Esta situação de grande diferença nos resultados para os métodos avaliados também é observada para alguns terrenos e reservatório com capacidade $C = 500000m^3$.

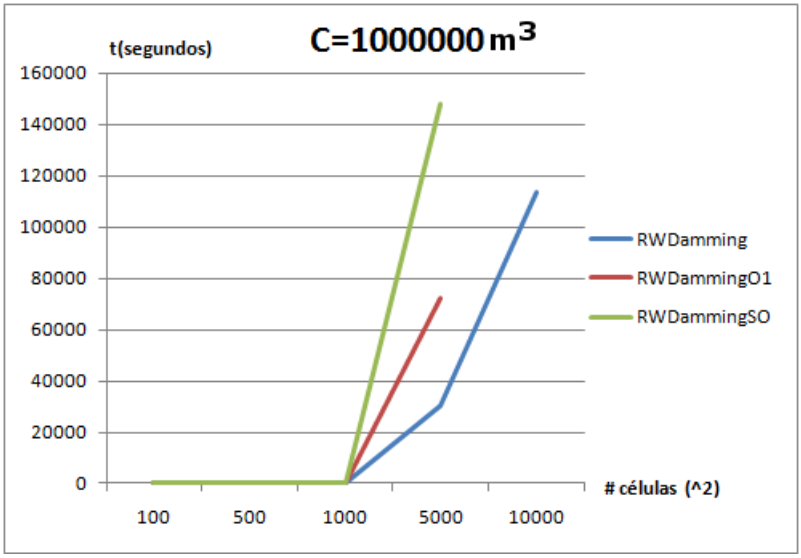
A Figura 4.4 apresenta o gráfico contendo a comparação entre os métodos expostos, para os resultados sobre a região R3, sendo que a Figura 4.4a apresenta o gráfico contendo os resultados para a capacidade $C = 100000m^3$, a Figura 4.4b contém o gráfico para a capacidade $C = 500000m^3$ e a Figura 4.4c contém o gráfico para a capacidade $C = 1000000m^3$. Da mesma maneira que os gráficos apresentados para região R2, para a região R3 alguns valores são descartados.




(a) Comparação para capacidade de $100000m^3$

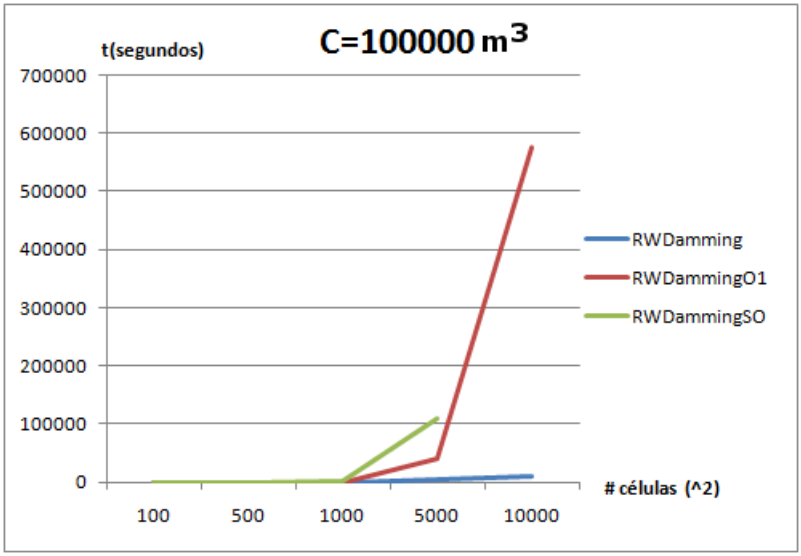



(b) Comparação para capacidade de $500000m^3$

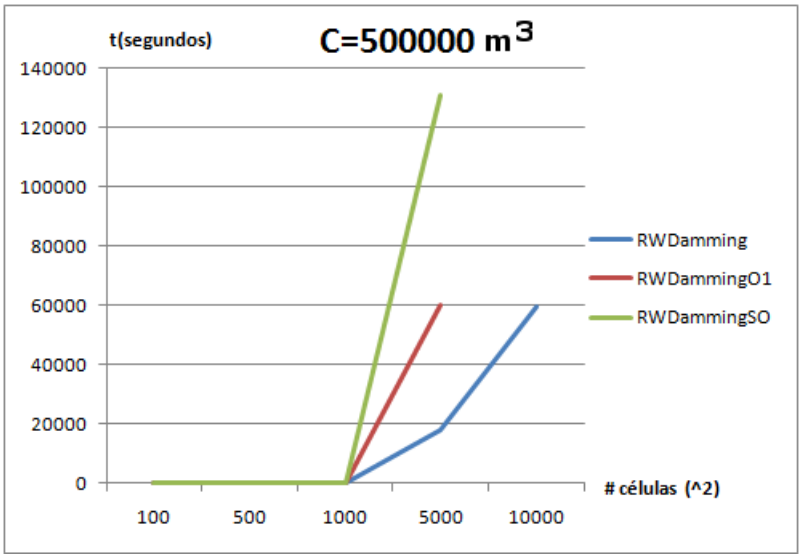



(c) Comparação para capacidade de $1000000m^3$

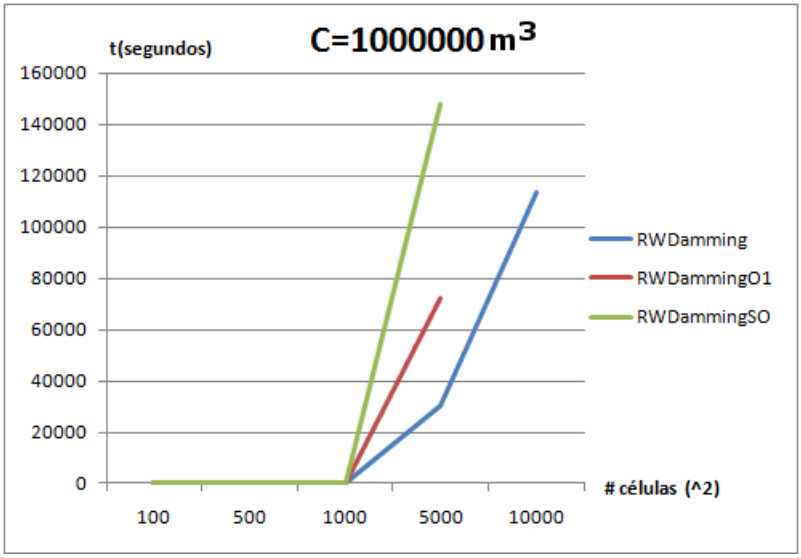
Figura 4.3: Comparações entre os métodos para região R2




(a) Comparação para capacidade de $100000m^3$




(b) Comparação para capacidade de $500000m^3$




(c) Comparação para capacidade de $1000000m^3$

Figura 4.4: Comparações entre os métodos para região R3

## 4.3 Resultados para a Análise 2

Os algoritmos RWDamming e JRPos foram avaliados utilizando a região composta pelo ribeirão de São Bartolomeu, Viçosa-MG, e, conforme mencionado anteriormente, estes métodos consideram redes de drenagem obtidas por diferentes métodos e que, portanto, podem conter um número diferente de pontos. Para o terreno avaliado, a rede utilizada pelo método RWDamming possui 21923 pontos e a do JRPos 21665.

A Tabela 4.5 apresenta os resultados dos tempos gastos para os métodos RWDamming e JRPos sobre a região especificada.

A Figura 4.5 apresenta o gráfico contendo a comparação entre os métodos para a análise 2. Note que o algoritmo RWDamming foi mais eficiente que o algoritmo JRPos, sendo cerca de 400 vezes mais rápido.

Tabela 4.5: Resultados para a Análise 2

| **Capacidade** | **Tempo de processamento (em segundos)** | |
|---|---|---|
| ($m^3$) | **RWDamming** | **JRPos** |
| 100 | 7 | 3394 |
| 500 | 7 | 3396 |
| 1000 | 7 | 3400 |
| 5000 | 7 | 3432 |
| 10000 | 7 | 3382 |
| 100000 | 7 | 3388 |
| 1000000 | 8 | 3388 |
| 5000000 | 9 | 3392 |
| 10000000 | 10 | 3395 |

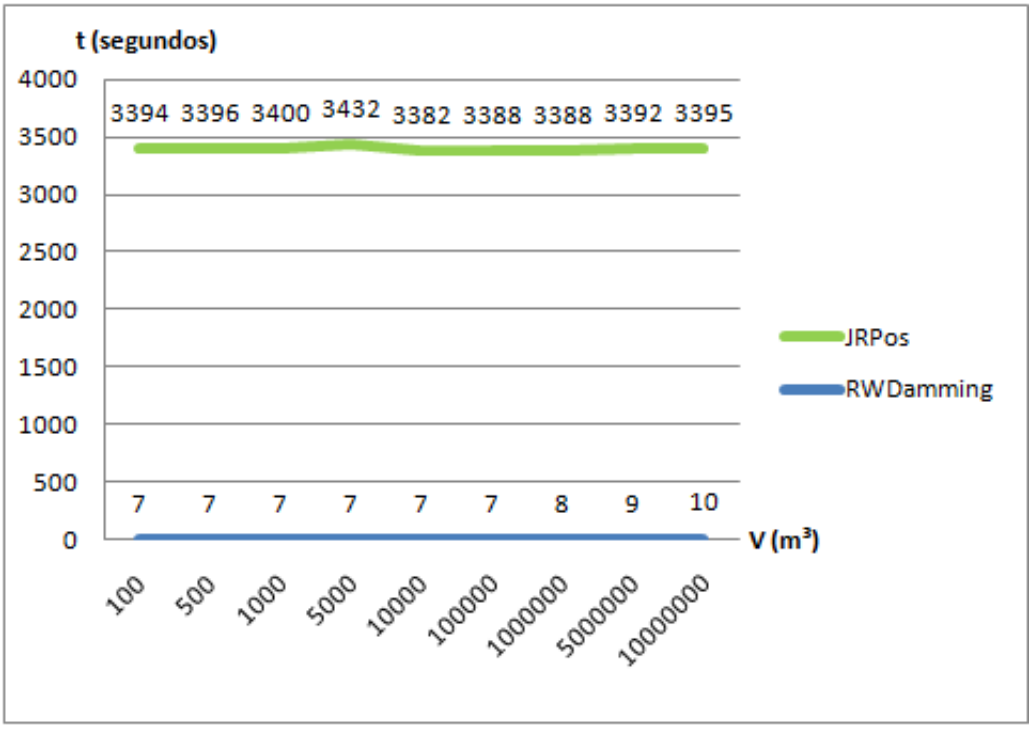



Figura 4.5: Resultados para RWDamming e JRPos.Comparações entre métodos processando a região de Viçosa

# Capítulo 5

# Conclusões

Nesse estudo foi apresentado um algoritmo, RWDamming, para realizar a busca do ponto mais adequado a se posicionar uma barragem em uma rede de drenagem, a fim de gerar um reservatório hídrico para suprir a capacidade desejada. Foi apresentado, também, um algoritmo, baseado no RWFlood, que determina a região alagada, a extensão e altura da barragem necessários para se construir o reservatório. Este algoritmo faz parte do RWDamming e contorna a ineficiência encontrada nos métodos presentes na literatura, pois não realiza um processo repetitivo para obtenção das características da barragem e reservatório, sendo que estas características são obtidas no decorrer do processo.

O algoritmo RWDamming faz uso de uma função objetivo para qualificar os candidatos ao posicionamento. Um ponto positivo desta função objetivo é que há parâmetros que quantificam o impacto para cada característica da barragem ou reservatório, e que, podem ser alterados. Outro ponto positivo é que se pode definir quantas regiões críticas o usuário preferir. Estas regiões são importantes para o processo de busca pois permitem criar um ambiente mais próximo ao real e definir as limitações naturais, legais, sociais ou outras ao se construir as barragens sobre este ambiente. Com isto, no contexto do método, poderia ter regiões indígenas, linhas férreas, área de preservação ambiental, uma região particular criada pelo usuário, entre outras.

Uma importante contribuição desse estudo é a interface gráfica desenvolvida, que é utilizada para tornar mais amigável o contato do usuário com o sistema de busca, permitindo-o também criar distintas situações que melhor refletem sua necessidade.

Então, as principais contribuições que esse estudo trouxe foram: um método que determina o ponto de posicionamento mais adequado a se construir um reservatório em uma rede de drenagem, um algoritmo mais simples e eficiente que os encontrados na literatura para se definir as características da barragem e do reservatório (extensão e altura da barragem, área alagada), uma função objetivo que torna possível identificar regiões cujo custo de alagamento é definido pelo usuário e um sistema com interface gráfica que

aprimora a usabilidade do sistema.

Por fim, pode-se considerar que o estudo em questão alcançou seus objetivos, pois o mesmo soluciona o problema de forma mais eficiente que o método encontrado na literatura (cerca de 400 vezes mais rápido em relação ao tempo de processamento), além de conter uma interface gráfica amigável para auxiliar o processo.

## 5.1 Trabalhos futuros

Como o problema de posicionamento de barragem é pouco explorado, há sugestões sobre trabalhos futuros que podem aprimorar a solução em questão:

- Otimização do sistema com uso de programação paralela. Como o algoritmo de posicionamento pode avaliar distintos candidatos de forma independente, torna-se clara a realização deste método através de busca paralela.
- Analisar mais profundamente o comportamento de reservatórios gerados e desenvolver uma técnica que elimine alguns candidatos antes mesmo de ser realizado o processo de alagamento, e com isto, otimizar o processo.
- Analisar e explorar barragens com formatos não retilíneos e no processo de definição de direção da barragem analisar células a montante e também a jusante.
- Incorporar o sistema em algum sistema de informação geográfica que seja mais reconhecido pela sociedade, permitindo uma avaliação mais precisa de seu comportamento e sua aceitabilidade.

# Referências Bibliográficas


[1] ANA (2005). Página da Agência Nacional de Águas (ANA): Cadastro de Barragens Outorgadas pela ANA. `http://www2.ana.gov.br/Paginas/servicos/cadastros/cnbarragens-outorgadas.aspx#`. Acessada em janeiro de 2012.

[2] Arge, L.; Chase, J. S.; Halpin, P.; Toma, L.; Vitter, J. S.; Urban, D. & Wickremesinghe, R. (2003). Efficient flow computation on massive grid terrain datasets. *Geoinformatica*, 7.

[3] Aulete (2012). Página iDicionário Aulete: Definição imageador. `http://aulete.uol.com.br/site.php?mdl=aulete_coletivo&op=loadVerbete&palavra=imageador`. Acessada em fevereiro de 2012.

[4] Barbosa, A. R. J. (2010). Hidrografia aplicada. Notas de aula, Universidade Federal de Ouro Preto, UFOP.

[5] Bonham-Carter, G. (1994). *Geographic information systems for geoscientists: modelling with GIS*. Computer methods in the geosciences. Pergamon.

[6] Brasil (2000). Lei nº 9.984, de 17 de julho de 2000. Publicado no Diário Oficial da República Federativa do Brasil. `http://www.planalto.gov.br/ccivil_03/Leis/L9984.htm`. Acessada em janeiro de 2012.

[7] Brasil (2010). Lei nº 12.334, de 20 de setembro de 2010. Publicado no Diário Oficial da República Federativa do Brasil. `http://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2010/Lei/L12334.htm`. Acessada em janeiro de 2012.

[8] Bravo, J. M.; Collischonn, W.; Tucci, C. E. M. & Pilar, J. V. (2008). Otimização de regras de operação de reservatórios com incorporação da previsão de vazão. *Revista Brasileira de Recursos Hídricos*, 13:181–196.

[9] Campbell, J. B. (2002). *Introduction to remote sensing / James B. Campbell*. Guilford Press, New York :, 3rd ed. edição.

[10] CiênciaViva (2012). Página Ciência Viva: Latitude e Longitude Instrumentos e Medição. http://www.cienciaviva.pt/latlong/anterior/gps.asp. Acessada em março de 2012.

[11] Câmara, G.; de Medeiros, J. S.; Camargo, E. C. G.; Fucks, S.; Felgueiras, C. A.; Freitas, U. & Barbosa, C. (1998). *Geoprocessamento para Projetos Ambientais*. Cap.4 Modelagem Numérica de Terreno.

[12] Collischonn, W. & Tassi, R. (2010). Introduzindo hidrologia. Technical report, Instituto de Pesquisas Hidráulicas (IPH/UFRGS). http://galileu.iph.ufrgs.br/collischonn/IPH_111/apostila%20Completa%202008.pdf. Acessada em 10 de maio de 2011.

[13] COPAM (2005). Deliberação Normativa COPAM nº 87, de 17 de junho de 2005. Publicado no Diário do Executivo de Minas Gerais. http://www.siam.mg.gov.br/sla/download.pdf?idNorma=8251. Acessada em janeiro de 2012.

[14] Danner, A.; Molhave, T.; Yi, K.; P.; Agarwal, K.; Arge, L. & Mitasova, H. (2007). Terrastream: from elevation data to watershed hierarchies. In *Proc. of ACM GIS*, pp. 117–124.

[15] Douglas, D. H. & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

[16] DPI-UFV (2013). Programa DamBuilding. http://www.dpi.ufv.br/~marcus/. Acessada em abril de 2013.

[17] Dutra, L. V.; Mura, J. C.; Freitas, C. C.; dos Santos, J. R. & Elmiro, M. T. (2003). Processamento de imagens de radar de abertura sintética - princípios e aplicações. *IV Workshop em Tratamento de Imagens*, pp. 4–13.

[18] ESRI (2011). ArcGis Page: ArcGis tutorial. http://www.esri.com/software/arcgis/index.html. Acessada em fevereiro de 2011.

[19] Felgueiras, C. A. & Câmara, G. (2001). Introdução à ciência da geoinformação. http://www.dpi.inpe.br/gilberto/livro/introd/cap7-mnt.pdf. Acessado em outubro de 2011.

[20] Jacobi, P. (2010). Página Geólogo: A água na Terra está se esgotando? É verdade que no futuro próximo teremos uma guerra pela água? http://www.geologo.com.br/aguahisteria.asp. Acessada em dezembro de 2010.

[21] JARS (1999). *Remote Sensing Notes*. National Space Development Agency of Japan (NASDA) / Remote Sensing Technology Center of Japan (RESTEC). Prepared by Asian Center for Research on Remote Sensing (ACRoRS) in Asian Institute of Technology (AIT).

[22] Jenson, S. & Domingue, J. (1988). Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric Engineering and Remote Sensing*, 54(11):1593–1600.

[23] Lawler, E. L. & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719.

[24] Magalhães, S. V. G.; Andrade, M. V. A.; Franklin, W. R. & Pena, G. C. (2012). A new method for computing the drainage network based on raising the level of an ocean surrounding the terrain. *15th AGILE International Conference on Geographic Information Science*, pp. 391–408.

[25] Metz, M.; Mitasova, H. & Harmon, R. S. (2011). Efficient extraction of drainage networks from massive, radar-based elevation models with least cost path search. *Hydrology and Earth System Sciences*, 15(2):667--678.

[26] MinGW (2012). MinGW Page: Welcome to MinGW. `http://www.mingw.org/`. Acessada em fevereiro de 2012.

[27] Moore, I. D.; Grayson, R. B. & Ladson, A. R. (1991). Digital terrain modelling: a review of hydrological, geomorphological and biological aplications. *Hydrological Processes*, 5:3–30.

[28] Muckell, J.; Andrade, M.; Franklin, W. R.; Cutler, B.; Inanc, M.; Xie, Z. & Tracy, D. M. (2007). Drainage network and watershed reconstruction on simplified terrain. In *17th Fall Workshop on Computational Geometry*, IBM TJ Watson Research Center, Hawthorne NY.

[29] Muckell, J.; Andrade, M.; Franklin, W. R.; Cutler, B.; Inanc, M.; Xie, Z. & Tracy, D. M. (2008). Hydrology-aware terrain simplification. In *5th International Conference on Geographic Information Science*, Park City, Utah, USA.

[30] NASA (2012). NASA Shuttle Radar Topography Mission (SRTM) Page. `http://www.jpl.nasa.gov/srtm/`. Acessada em janeiro de 2012.

[31] Nokia (2011). *Qt designer manual*. `http://doc.qt.nokia.com/3.3/designer-manual.html`. Acessada em 23 de julho de 2012.

[32] Nokia (2012). Qt Page: QDoc Reference Documentation. http://doc-snapshot.qt-project.org/qdoc/01-qdoc-manual.html. Acessada em junho de 2012.

[33] O'Callaghan, J. & Mark, D. (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics and Image Processing*, 28:328–344.

[34] Planchon, O. & Darboux, F. (2002). A fast, simple and versatile algorithm to fill the depressions of digital elevation models. *Catena*, 46(2-3):159--176.

[35] Sing, K. D. (2005). Dam and watershed analysis mini project repor. Master's thesis, Department of Computer Science and Engineering Indian Institute of Technolog, Mumbai, Índia.

[36] Soille, P. & Gratin, C. (1994). An efficient algorithm for drainage network extraction on dems. *Journal of Visual Communication and Image Representation*, 5(2):181–189.

[37] Souza, J. R. C.; Andrade, M. V. A. & Nogueira, K. (2010). Heurística para o posicionamento de reservatórios d'água. *XI Brazilian Symposium on GeoInformatics*, 13:181–196.

[38] Tarboton, D. (1997). A new method for the determination of flow directions and contributing areas in grid digital elevation models. *Water Resources Research*, 33:309–319.

[39] Tommaselli, A. M. G. (2011). Fotogametria Básica. http://www4.fct.unesp.br/docentes/carto/tomaseli/Fotogrametria_1/arquivos_pdf_2009/FOTO1_2009_.pdf. Acessada em novembro de 2011.

[40] UFV (2013). Atlas digital das águas de Minas. http://www.atlasdasaguas.ufv.br/apresentacao.html. Acessada em janeiro de 2013.

[41] Universidade de Lisboa, D. d. E. (2012). Teodolito. http://www.educ.fc.ul.pt/icm/icm2003/icm11/napl4.htm. Acessada em março de 2012.

[42] Wetzel, R. G. (1983). *Periphyton of Freshwater Ecosystems*, volume Proceedings of the First International Workshop on Periphyton of Freshwater Ecosystems. Developments in Hydrobiology, Vol. 17. B. V. Junk Publishers, The Hague.

# Apêndice A

# Especificação de requisitos do programa


# Re-Build
# Especificação dos Requisitos de Software

**Versão <1.1>**

# Histórico da Revisão

| Data | Versão | Descrição | Autor |
|---|---|---|---|
| 22/jul/12 | 1.0 | criação | Rodolfo da Costa Ladeira |
| 08/ago/12 | 1.1 | elaboração | Rodolfo da Costa Ladeira |
| | | | |
| | | | |

# Índice Analítico

# Especificação dos Requisitos de Software

## 1. Introdução

### 1.1 Finalidade

Este documento descreve totalmente o comportamento externo do aplicativo ou do subsistema Re-Build. Ele também descreve requisitos não-funcionais, restrições de design e outros fatores necessários para fornecer uma visão completa e abrangente dos requisitos do software.

### 1.2 Escopo

O sistema Re-Build (Reservoir Building) é um sistema de informação geográfica para o posicionamento de barragens gerando um reservatório hídrico. O mesmo visa atingir as situações onde o posicionamento pode ser realizado pelo próprio usuário, definindo o local exato de posicionamento, ou pelo sistema, em que busca em um trecho determinado pelo usuário o reservatório hídrico que tenha menor função objetivo. Esta função objetivo é uma soma simples dos valores obtidos para as características do reservatório proporcionais ao peso que todas as características trazem ao posicionamento. O usuário tem diversas funcionalidades que podem contribuir para a decisão do posicionamento da barragem. Neste documento serão descritas tais funcionalidades.

### 1.3 Definições, Acrônimos e Abreviações

Camadas: regiões críticas ao posicionamento da barragem (ex.: áreas de preservação permanentes)

MDE: modelo digital de elevação

### 1.4 Visão Geral

Este documento é separado de acordo com os casos de uso presentes no sistema. São eles:

- Criar reservatório
- Abrir mapa
- Definir parâmetros do reservatório
- Gerenciar camadas
- Gerenciar reservatórios
- Gerar visualização 2D
- Gerar visualização 3D
- Alterar opções
- Atualizar fluxo mínimo

## 2. Descrição Geral

O sistema seguirá o diagrama de caso de uso proposto na Figura 2-1.

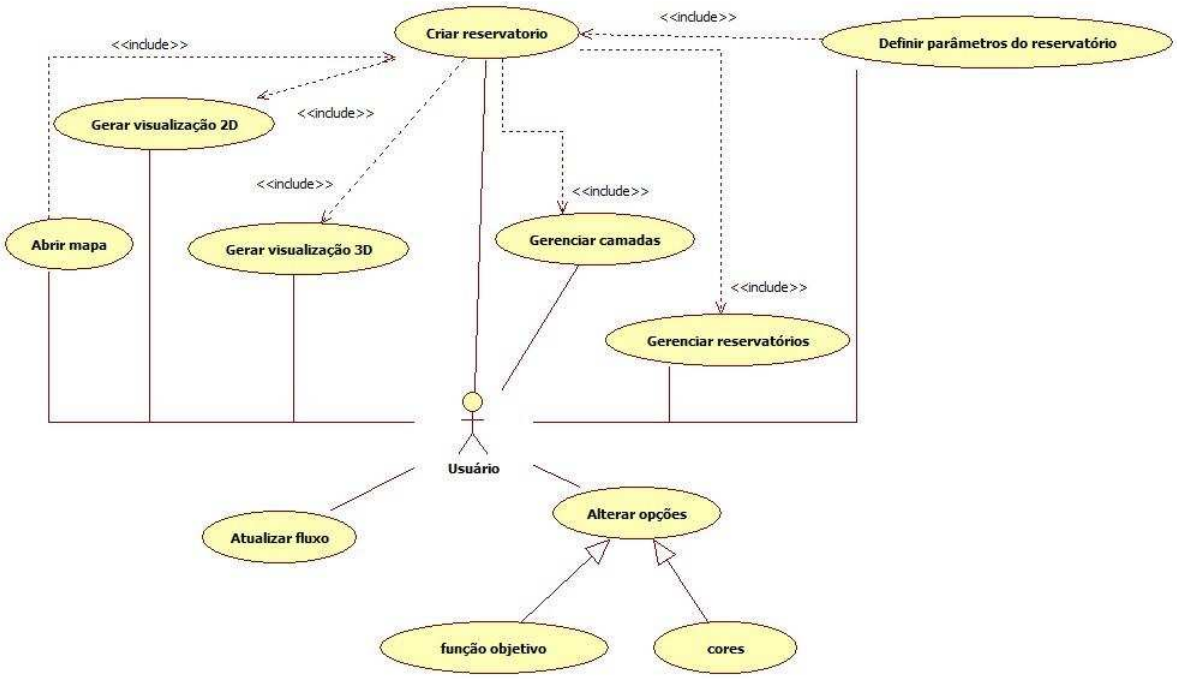



Figura 2-1-Diagrama de caso de uso

## 2.1 Relatório Sintético de Modelo de Casos de Uso

Os casos de uso se resumem em:

- Criar reservatório – O usuário pode definir um local na rede de drenagem do mapa carregado ou definir um trecho em que o sistema irá buscar o melhor reservatório
- Abrir mapa – O usuário pode abrir terrenos de tamanhos diferentes e no formato ASCII/ESRI
- Definir parâmetros do reservatório – O usuário pode definir os parâmetros que interferem no resultado final da criação do reservatório hídrico. São eles: capacidade desejada e orientação da barragem.
- Gerenciar camadas – O usuário pode gerenciar as camadas. Criando novas camadas, inserindo ou excluindo células à camadas e excluindo camadas. Além de expor/omitir camadas no mapa.
- Gerenciar reservatórios – O usuário pode gerenciar reservatórios. O mesmo pode salvar o reservatório corrente, excluir ou visualizar algum outro reservatório salvo anteriormente.
- Gerar visualização 2D – O usuário pode visualizar o reservatório gerado numa visão de perfil com plano paralelo a barragem.
- Gerar visualização 3D – O usuário pode visualizar o reservatório gerado numa visão tridimensional.
- Alterar opções – O usuário pode alterar algumas opções que afetam ou apenas a aparência (cores) do mapa ou a função que determina o custo geral de se construir a determinada barragem (função objetivo)
- Atualizar fluxo mínimo- O usuário pode alterar o fluxo mínimo que determina as células que são consideradas rede de drenagem para o terreno carregado.

### 2.2 Suposições e Dependências

Abaixo se encontra a Tabela 2-1 que relaciona as dependências para cada caso de uso do sistema.

| Caso de uso | Dependências |
|---|---|
| Criar reservatório | Abrir mapa,definir parâmetros de reservatório |
| Gerenciar reservatório | Criar reservatório |
| Gerenciar camadas | Criar reservatório |
| Gerar visualização 2D | Criar reservatório |
| Gerar visualização 3D | Criar reservatório |

**Tabela 2-1-Dependências para casos de uso**

## 3. Requisitos Específicos

Nas próximas subseções será exposto os casos de usos e as ações necessárias (ordenadas) para que os mesmos sejam efetivados.

### 3.1 Relatórios de Caso de Uso

#### *3.1.1 Caso de uso Criar reservatório*

| | Atores | Ação | Dependências |
|---|---|---|---|
| 1<br>Criar reservatório | Usuário | 1.1 Usuário realiza caso de uso Abrir Mapa | 2 |
| | | 1.2 Usuário realiza caso de uso Definir parâmetros | 3 |
| | | 1.3 Usuário alterna sistema para criar reservatório | |
| | | 1.4a Usuário define ponto de posicionamento (com click do botão esquerdo do mouse) | |
| | | 1.4b Usuário define trecho (com click do botão direito do mouse para ponto inicial e final) | |
| | | 1.5 Usuário define se deseja expor ou não o valor da função objetivo | |
| | Sistema | 1.6a O sistema realiza a inundação | |
| | | 1.6b O sistema identifica os pontos candidatos ao posicionamento | |
| | | 1.7b O sistema realiza a inundação para os candidatos e define qual o melhor | |
| | | 1.8 O sistema expões ao usuário os resultados | |

#### *3.1.2 Caso de uso Abrir mapa*

| | Atores | Ação | Dependências |
|---|---|---|---|
| 2<br>Abrir mapa | Usuário | 2.1 Usuário escolhe a opção abrir mapa | |
| | Sistema | 2.2 Sistema expõe uma janela para escolha do arquivo | |
| | Usuário | 2.3 Usuário escolhe o mapa a ser carregado | |

### *3.1.3 Caso de uso Definir parâmetros do reservatório*

| | Atores | Ação | Dependências |
|---|---|---|---|
| 3<br>Definir parâmetros do reservatório | Usuário | 3.1 Usuário define a capacidade desejada | |
| | | 3.2 Usuário escolhe algoritmo para direção de barragem (a – Algoritmo Vizinhos, b – Algoritmo Douglas Peucker) | |
| | | 3.3a Usuário define o número de vizinhos presentes no algoritmo | |
| | | 3.3b Usuário define o número de vizinhos presentes no algoritmo | |
| | | 3.4b Usuário define o valor da tolerância para distância de ponto ao segmento de reta | |

### *3.1.4 Caso de uso Gerenciar camadas*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 4a<br>Incluir camadas | Usuário | 4a.1 Usuário aciona botão para adicionar nova camada | |
| | Sistema | 4a.2 Sistema apresenta janela para atributos de nova camada | |
| | Usuário | 4a.3 Usuário define atributos para esta nova camada(nome, peso e cor) | |
| | Sistema | 4a.4 Sistema armazena esta camada na lista de camadas | |
| | | | |
| 4b<br>Excluir camadas | Usuário | 4b.1 Usuário define na lista de camadas qual camada deseja excluir | 4a |
| | | 4b.2 Usuário aciona botão para excluir a camada selecionada | 4b.1 |
| | Sistema | 4b.3 Sistema elimina a camada da lista de camadas | |
| | | | |
| 4c<br>Adicionar/Eliminar pontos | Usuário | 4c.1 Usuário define se deseja incluir ou excluir pontos da camada | |
| | | 4c.2 Usuário seleciona camada na lista de camadas | 4a |
| | | 4c.3 Usuário alterna sistema para adicionar camadas | |
| | | 4c.4a Usuário aciona botão esquerdo do mouse para seleção unitária de células | 4c.1, 4c.2, 4c.3 |
| | | 4c.5a Usuário aciona botão esquerdo do mouse segura-o e seleciona diversas células por onde movimenta o mouse | 4c.1, 4c.2, 4c.3 |
| | | 4c.4b Usuário aciona botão direito do mouse para seleção de várias células simultâneas | 4c.1, 4c.2, 4c.3 |
| | Sistema | 4c.6 O sistema deve desenhar apenas os pontos de camadas, evitando assim que todo o MDE seja redesenhado | |
| | | | |

| | | | |
|---|---|---|---|
| 4d<br>Expor/omitir camada | Usuário | 4d.1 O usuário seleciona a camada na lista de camadas | 4a |
| | | 4d.2a O usuário marca a caixa de checagem para expor camada | 4d.1 |
| | | 4d.2a O usuário desmarca a caixa de checagem para omitir camada | |
| | Sistema | 4d.3 O sistema redesenha todo o mapa omitindo as camadas que estão marcadas para serem omitidas | |
| 4e Previnir nomes de camadas duplicados | Sistema | 4e.1 Sistema deve evitar que nome duplicados sejam inseridos na lista de camadas | |

### *3.1.5 Caso de uso Gerenciar reservatórios*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 5a<br>Salvar reservatório | Usuário | 5a.1 Usuário aciona botão adicionar reservatório | |
| | | 5a.2 Sistema expõe ao usuário nova janela para nomear o reservatório | |
| | | 5a.3 Usuário define nome ao reservatório | |
| | | 5a.4 Sistema armazena em uma lista de reservatório o reservatório corrente | |
| | | | |
| 5b<br>Excluir reservatório | Usuário | 5b.1 Usuário seleciona reservatório na lista de reservatório | 5a |
| | | 5b.2 Usuário aciona botão de excluir reservatório | |
| | Sistema | 5b.3 Sistema elimina da lista de reservatórios o reservatório selecionado | |
| | | | |
| 5c<br>Visualizar reservatório | Usuário | 5c.1 Usuário seleciona reservatório na lista de reservatório | 5a |
| | | 5c.2 Usuário aciona botão de visualizar reservatório | |
| | Sistema | 5c.3 Sistema apresenta reservatório selecionado ao usuário e suas características | |

### *3.1.6 Caso de uso Gerar visualização 2D*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 6<br>Gerar visualização 2D | Usuário | 6.1 Usuário marca caixa de checagem expor visualização 2D | |
| | | 6.2 Realiza o processo de gerar reservatório | 1 |
| | Sistema | 6.3 Sistema expõe nova janela contendo com plano paralelo a barragem | |

### *3.1.7 Caso de uso Gerar visualização 3D*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 7<br>Gerar visualização 3D | Usuário | 7.1 Usuário marca caixa de checagem expor visualização 3D | |
| | | 7.2 Realiza o processo de gerar reservatório | 1 |
| | Sistema | 7.3 Sistema expõe nova janela contendo visualização tridimensional do reservatório | |

### *3.1.8 Caso de uso Alterar opções*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 8a<br>Alterar opções cores | Usuário | 8a.1 Usuário aciona opção de alterar opções de cores | |
| | Sistema | 8a.2 Sistema expõe ao usuário nova janela contendo opções de cores | |
| | Usuário | 8a.3 Usuário define cores para o MDE e o reservatório (padrão ou personalizada) | |
| | Sistema | 8a.4 Sistema atribui cores ao MDE e o reservatório | |
| | | | |
| 8b<br>Alterar opções função objetivo | Usuário | 8b.1 Usuário aciona opção de alterar opções de função objetivo | |
| | Sistema | 8b.2 Sistema expõe ao usuário nova janela contendo opções de função objetivo | |
| | Usuário | 8b.3 Usuário define valores para peso das características da barragem e reservatório (padrão ou personalizado) | |
| | Sistema | 8b.4 Sistema atribui pesos as características da barragem e reservatório | |

### *3.1.9 Caso de uso Atualizar fluxo mínimo*

| | Atores | Ações | Dependências |
|---|---|---|---|
| 9<br>Atualizar fluxo | Usuário | 9.1 Usuário define novo valor para o fluxo mínimo | 1 |
| | | 9.2 Usuário aciona botão atualizar fluxo | |
| | Sistema | 9.3 Sistema recalcula células presentes na rede de drenagem e expõe ao usuário | |

## 3.2 Requisitos Suplementares

Existe ainda alguns requisitos que não são enquadrados como caso de uso mas devem ser exposto pois podem facilitar o uso do sistema pelo usuário.

| | Atores | Ações | Dependências |
|---|---|---|---|
| 10<br>Efetuar zoom | Usuário | 10.1a Usuário aciona botão de zoom in | |
| | | 10.1b Usuário aciona botão de zoom out | |
| | | 10.1c Usuário reposiciona barra de rolagem de zoom | |
| | Sistema | 10.2 Sistema redefine visualização do mapa, tamanho do pixel, ortho ou view port do opengl | |
| | | | |
| 11<br>Rolar barra de rolagem | Usuário | 11.1 Usuário reposiciona barra de rolagem da visualização | |
| | Sistema | 10.2 Sistema redefine visualização do mapa | |
| | | 10.3 Sistema redefine posicionamento da barra e botões de zoom para estarem sempre visíveis | |

# Apêndice B

# Especificação técnica do programa

### Contents



# Camada Class Reference

Class for creating critical points. More...

```
#include <Camada>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **Camada** () |
| | **Camada** ( QString *n*, double *p*, QColor *c*, int *nLinha*, int *nColun* ) |
| | **Camada** ( Camada const & *c* ) |
| | **~Camada** () |
| void | **alteraPonto** ( Point *p*, bool *add* ) |
| QColor | **getCorCamada** () |
| bool | **getEstaIncluidoNoAlgoritmo** () |
| QString | **getNome** () |
| double | **getPeso** () |
| void | **setCorCamada** ( QColor *c* ) |
| void | **setEstaIncluidoNoAlgoritmo** ( bool *op* ) |
| void | **setNome** ( QString *n* ) |
| void | **setPeso** ( double *p* ) |
| Camada & | **operator=** ( Camada const & *c* ) |
| bool | **operator==** ( Camada const & *c* ) const |

## Public Variables

| | |
|---|---|
| bool | **estaIncluidoNoAlgoritmo** |
| bool | **estaVisivel** |
| QList<Point> | **listaDePontos** |
| int | **nColunas** |

| | |
|---|---|
| int | **nLinhas** |
| bool ** | **pontos** |

# Detailed Description

Class for creating critical points.

When calculating the best position of the dam, the algorithm can use these points to define regions where it is disadvantageous to have a reservoir.

# Member Function Documentation

## Camada::Camada ()

Default constructor.

Set default values to attributes

## Camada::Camada ( QString *n*, double *p*, QColor *c*, int *nLinha*, int *nColun* )

Constructor.

Construct an object (Camada) tha will have:

nome = n

peso = p

cor = c

## Camada::Camada ( Camada const & *c* )

Copy Constructor. Creates an object receiving attributes from c

## Camada::~Camada ()

Destructor

## void Camada::alteraPonto ( Point *p*, bool *add* )

Change status of point p using add. Printed(add==true) or not

## QColor Camada::getCorCamada ()

Return cor

## bool Camada::getEstaIncluidoNoAlgoritmo ()

Return estaIncluidoNoAlgoritmo

### QString Camada::getNome ()

Returns nome

### double Camada::getPeso ()

Returns peso

### void Camada::setCorCamada ( QColor *c* )

Set c to cor

### void Camada::setEstaIncluidoNoAlgoritmo ( bool *op* )

Set op to estaIncluidoNoAlgoritmo

### void Camada::setNome ( QString *n* )

Set n to nome

### void Camada::setPeso ( double *p* )

Set p to peso

### Camada & Camada::operator= ( Camada const & *c* )

This function overloads Camada::operator=(Camada const& c).

The current object camada receive the attributes from c

### bool Camada::operator== ( Camada const & *c* ) const

This function overloads Camada::operator==(Camada const& c) const.

Return if two objects Camada are equal

## Member Variable Documentation

### bool Camada::estaIncluidoNoAlgoritmo

This variable holds the value if this Camada is insert into algorithm dam positioning.

### bool Camada::estaVisivel

This variable holds the value if this Camada is actual showing to user.

## QList<Point> Camada::listaDePontos

This variable holds the list of critical points.

## int Camada::nColunas

This variable holds the number of lines in matrix.

## int Camada::nLinhas

This variable holds the number of colluns in matrix.

## bool ** Camada::pontos

This variable holds the a pointer to pointer, representing the matrix of critical points (I.e. the reservoir being at these points is (peso) disadvantageous ).

[Previous: All Classes] [Next: Fluxo]

- Modules
- Fluxo

[Previous: Camada] [Next: Inundacao]

### Contents



# Fluxo Class Reference

Class for create the drainage network, calculated by the flow directionand flow accumulation More...

```
#include <Fluxo>
```

- List of all members, including inherited members

## Public Types

typedef **ponto**

## Public Functions

**Fluxo** ()
**Fluxo** ( int *nLinhas*, int *nColunas* )
**Fluxo** ( Fluxo const & *fluxoCopia* )
**~Fluxo** ()
void **calculaFluxo** ( short int ** *elevacoes*, int *noData* )

## Public Variables

| | |
|---|---|
| unsigned char ** | **direcao** |
| int ** | **fluxo** |
| int | **fluxoMinimo** |
| int | **nColunas** |
| int | **nLinhas** |
| int | **qtdeCelulasRio** |
| bool ** | **rio** |
| int | **valorPadraoFluxo** |
| queue<ponto> | **vetorDeFilasDeNiveis** [2 * limiteAltitude] |

# Detailed Description

Class for create the drainage network, calculated by the flow directionand flow accumulation

# Member Type Documentation

## typedef Fluxo::ponto

the object to map a cell(point) of MDE easily.

# Member Function Documentation

## Fluxo::Fluxo ()

## Fluxo::Fluxo ( int *nLinhas*, int *nColunas* )

Constructor. Creates an object that have only the variables initialized, no drainage network is calculed yet.

## Fluxo::Fluxo ( Fluxo const & *fluxoCopia* )

Copy Constructor. Creates an object receiving attributes from fluxoCopia

## Fluxo::~Fluxo ()

Destructor

## void Fluxo::calculaFluxo ( short int ** *elevacoes*, int *noData* )

function to calculated the drainage network.

elevecoes is the matrix with cell`s elevation of terrain. noData is the value representing a invalid cell

This function use three another private function: setInundacao to identify the flow direction of each cell. setFluxo to identify the flow accumalate of each cell. setRio to identify cells that are in drainage network.

# Member Variable Documentation

## unsigned char ** Fluxo::direcao

This variable holds the pointer to pointer, representing the matrix of flow direction.

## int ** Fluxo::fluxo

This variable holds the pointer to pointer, representing the matrix of flow accumulation.

## int Fluxo::fluxoMinimo

This variable holds the minimum flow to consider a cell in the drainage network. Cell`s value in rio matrix above fluxoMinimo is considering a cell of network drainage.

### int Fluxo::nColunas

This variable holds the number of columns of MDE.

### int Fluxo::nLinhas

This variable holds the numbers of lines of MDE.

### int Fluxo::qtdeCelulasRio

This variable holds the number of cells that are in drainage network.

### bool ** Fluxo::rio

This variable holds the pointer to pointer, representing the matrix river`s cells (drainage network).

### int Fluxo::valorPadraoFluxo

This variable holds the default value for fluxoMinimo.

### queue<ponto> Fluxo::vetorDeFilasDeNiveis [2 * limiteAltidude]

This variable holds the queue of ponto used to restore the nexts ponto that will be alagado.

[Previous: Camada] [Next: Inundacao]

- Modules
- Inundacao

[Previous: Fluxo] [Next: janelabarragem]

### Contents



# Inundacao Class Reference

Class to create the resorvoir, with the dam and flooded cells More...

```
#include <Inundacao>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **Inundacao** () |
| | **Inundacao** ( int *linhas*, int *colunas* ) |
| | **Inundacao** ( Inundacao const & *inundacaoCopia* ) |
| | **~Inundacao** () |
| void | **acertaTamanhoBarragem** ( short int ** *matrizDeEstados* ) |
| QList<PontoZ> | **algoritmoDouglasPeucker** ( QList<PontoZ> & *lista*, int *epsilon*, QList<PontoZ> & *resultadoLista* ) |
| void | **criaListaParaDouglasPeucker** ( int *numElementos*, QList<PontoZ> & *listaASerSimplificada*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* ) |
| void | **defineVetorNormalABarragem** ( int *op*, int *val*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes*, int *valEp* ) |
| PontoZ | **getPosicaoBarragem** () |
| void | **inicializaBarragem** ( int *x*, int *y*, int *xAcima*, int *yAcima*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* ) |
| void | **inunda** ( int *posX*, int *posY*, unsigned char ** *matrizDeDirecoes*, short int ** *elevacoes*, bool ** *rio*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* ) |
| void | **marcaBarragem** ( short int ** *matrizDeEstados* ) |
| void | **marcaMontante** ( int *xi*, int *yi*, short int ** *matrizDeEstados*, unsigned char ** *matrizDeDirecoes*, short int ** *elevacoes*, bool ** *rio*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* ) |
| bool | **permiteInundacaoEAumentaBarragem** ( PontoZ *pontoAvaliado* ) |

| | |
|---|---|
| void | **sobeNivelDeAgua** ( short int ** *matrizDeEstados*, short int ** *elevacoes* ) |
| void | **vetorNormalPorDouglasPeucker** ( int *numElementos*, int *epsilon*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* ) |
| void | **vetorNormalPorVizinhanca** ( int *viz*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* ) |

# Public Variables

| | |
|---|---|
| int | **areaBarragemTocada** |
| int | **areaLaminaAgua** |
| int | **comprimentoBarragemTocada** |
| short int ** | **matrizEstados** |
| int | **nColunas** |
| int | **nLinhas** |
| int | **nivelAgua** |
| PontoZ | **posicaoBarragem** |
| int | **tamanhoMaximoDaBarragem** |
| PontoZ | **vetorNormalABarragem** |
| int | **volumeAgua** |
| int | **volumeAlvo** |

# Detailed Description

Class to create the resorvoir, with the dam and flooded cells

# Member Function Documentation

## Inundacao::Inundacao ()

## Inundacao::Inundacao ( int *linhas*, int *colunas* )

Constructor. Creates an object that have only the variables initialized, no reservoir is calculed yet.

## Inundacao::Inundacao ( Inundacao const & *inundacaoCopia* )

Copy Constructor. Creates an object receiving attributes from inundacaoCopia

## Inundacao::~Inundacao ()

Destructor

## void Inundacao::acertaTamanhoBarragem ( short int ** *matrizDeEstados* )



Definy the real cells that are dam`s cell. Eliminate cells that arent near a flooded cell.

**QList<PontoZ> Inundacao::algoritmoDouglasPeucker ( QList<PontoZ> & *lista*, int *epsilon*, QList<PontoZ> & *resultadoLista* )**

The Douglas-Peucker algorithm, used to simplify the isoline and obtain a vector to be the normal vector of the dam

**void Inundacao::criaListaParaDouglasPeucker ( int *numElementos*, QList<PontoZ> & *listaASerSimplificada*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* )**

Defiy what points in drainage network will be in the Douglas-Peucker algorithm

**void Inundacao::defineVetorNormalABarragem ( int *op*, int *val*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes*, int *valEp* )**

Defiy what function to call to calculate the dam`s normal vector

**PontoZ Inundacao::getPosicaoBarragem ()**

Returns the dam position.

**void Inundacao::inicializaBarragem ( int *x*, int *y*, int *xAcima*, int *yAcima*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* )**

Initialize the variable needed to create the reservoir

**void Inundacao::inunda ( int *posX*, int *posY*, unsigned char ** *matrizDeDirecoes*, short int ** *elevacoes*, bool ** *rio*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* )**

Function that invoque the sequence of functions to create the reservoir

**void Inundacao::marcaBarragem ( short int ** *matrizDeEstados* )**

Mark the points in MDE that could be a dam cell

**void Inundacao::marcaMontante ( int *xi*, int *yi*, short int ** *matrizDeEstados*, unsigned char ** *matrizDeDirecoes*, short int ** *elevacoes*, bool ** *rio*, int *opVetorNormal*, int *valorVetorNormal*, int *valEp* )**

Initialize the process of construct the dam

**bool Inundacao::permiteInundacaoEAumentaBarragem ( PontoZ *pontoAvaliado* )**

Function used to test if a point pontoAvaliado can be a flooded cell. If this point is in the same direction of the dam, it will grow up. If not the algorithm test if it can be or not a flooded cell.

### void Inundacao::sobeNivelDeAgua ( short int ** *matrizDeEstados*, short int ** *elevacoes* )

Rise the water`s level flooding cells generated new values for variable

### void Inundacao::vetorNormalPorDouglasPeucker ( int *numElementos*, int *epsilon*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* )

Function to prepare and call Douglas-Peucker algorithm

### void Inundacao::vetorNormalPorVizinhanca ( int *viz*, short int ** *elevacoes*, bool ** *rio*, unsigned char ** *matrizDeDirecoes* )

Function to calculate the dam`s normal vector by the simple sum of vector

## Member Variable Documentation

### int Inundacao::areaBarragemTocada

This variable holds the value of the area for the dam.

### int Inundacao::areaLaminaAgua

This variable holds the value of the flooded area in the reservoir.

### int Inundacao::comprimentoBarragemTocada

This variable holds the value of the dam`s extent.

### short int ** Inundacao::matrizEstados

This variable holds a pointer to pointer representing the matrix of states of cell. If cell are dam, flooded, drainage network or nothing.

### int Inundacao::nColunas

This variable holds the number of columns of MDE.

### int Inundacao::nLinhas

This variable holds the number of lines of MDE.

### int Inundacao::nivelAgua

This variable holds the value of current high of the water generated by the flooding.

## PontoZ Inundacao::posicaoBarragem

This variable holds a cell representing by a point to identify the the position downstream the dam position.

## int Inundacao::tamanhoMaximoDaBarragem

This variable holds the maximum value of dam extent. Its not used in the algorithm yet.

## PontoZ Inundacao::vetorNormalABarragem

This variable holds a point to describe the final point of the dam`s normal vector. The start point is the dam position. with this point we can calculate the direction of the dam use simple math calcs.

## int Inundacao::volumeAgua

This variable holds the value of current resorvoir`s volume.

## int Inundacao::volumeAlvo

This variable holds the value volume required by user to construct the reservoir.

[Previous: Fluxo] [Next: janelabarragem]

### Contents



# janelabarragem Class Reference

Class to create a window 3D view of reservoir and terrain More...

```
#include <janelabarragem>
```

- List of all members, including inherited members

## Public Functions

**janelabarragem** ( VisualizacaoMapa * *viMapa*, QWidget * *parent* = 0 )
**~janelabarragem** ()

## Detailed Description

Class to create a window 3D view of reservoir and terrain

## Member Function Documentation

### janelabarragem::janelabarragem ( VisualizacaoMapa * *viMapa*, QWidget * *parent* = 0 )

QDialog(parent), ui(new Ui::janelabarragem)

Constructor the painel with 3D view of reservoir

### janelabarragem::~janelabarragem ()

Destructor

### Contents



# janelaBarragem2D Class Reference

Class to create a window 2D view of the reservoir More...

```
#include <janelaBarragem2D>
```

- List of all members, including inherited members

## Public Functions

**janelaBarragem2D** ( QWidget * *parent* = 0 )
**~janelaBarragem2D** ()
void **recarregandoMapa** ( VisualizacaoMapa * *viMapa* )

## Detailed Description

Class to create a window 2D view of the reservoir

## Member Function Documentation

### janelaBarragem2D::janelaBarragem2D ( QWidget * *parent* = 0 )

### janelaBarragem2D::~janelaBarragem2D ()

### void janelaBarragem2D::recarregandoMapa ( VisualizacaoMapa * *viMapa* )

### Contents



# janelaCamadas Class Reference

Class to create a window for inserting new camadas More...

```
#include <janelaCamadas>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **janelaCamadas** ( QWidget * *parent* = 0 ) |
| | **~janelaCamadas** () |
| QStringList | **getValores** ( bool & ) |

## Detailed Description

Class to create a window for inserting new camadas

## Member Function Documentation

### janelaCamadas::janelaCamadas ( QWidget * *parent* = 0 )

### janelaCamadas::~janelaCamadas ()

### QStringList janelaCamadas::getValores ( bool & )

### Contents



# janelaOpcaoCor Class Reference

Class to create the window color option More...

```
#include <janelaOpcaoCor>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **janelaOpcaoCor** ( QWidget * *parent* = 0 ) |
| | **~janelaOpcaoCor** () |
| QStringList | **getValores** ( bool & *ok* ) |

## Detailed Description

Class to create the window color option

## Member Function Documentation

### janelaOpcaoCor::janelaOpcaoCor ( QWidget * *parent* = 0 )

### janelaOpcaoCor::~janelaOpcaoCor ()

### QStringList janelaOpcaoCor::getValores ( bool & *ok* )

### Contents



# janelaopcaofb Class Reference

Class to create the window objective function option More...

```
#include <janelaopcaofb>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **janelaopcaofb** ( QWidget * *parent* = 0 ) |
| | **~janelaopcaofb** () |
| QStringList | **getValores** ( bool & *ok* ) |

## Detailed Description

Class to create the window objective function option

## Member Function Documentation

### janelaopcaofb::janelaopcaofb ( QWidget * *parent* = 0 )

### janelaopcaofb::~janelaopcaofb ()

### QStringList janelaopcaofb::getValores ( bool & *ok* )

- Modules
- JanelaPrincipal

[Previous: janelaopcaofb] [Next: janelasobre]

### Contents



# JanelaPrincipal Class Reference

Class to create the main window More...

```
#include <JanelaPrincipal>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **JanelaPrincipal** ( QWidget * *parent* = 0 ) |
| | **~JanelaPrincipal** () |
| void | **atualizaCamposSaida** () |
| QPoint | **buscaMelhorPonto** ( VisualizacaoMapa *viMapa*, QList<QPoint> *listaPontos*, double *volumeDesejado* ) |
| void | **criaListaComCamadas** () |
| void | **criaVisualizacaoBarragem** () |
| void | **deletaListaComCamadas** () |
| void | **desAbilitandoBotoes** () |
| void | **habilitandoBotoes** () |
| void | **mousePressionadoParaBarragem** ( QMouseEvent * *ev* ) |
| void | **mousePressionadoParaCamada** ( QMouseEvent * *ev* ) |
| bool | **pontoDoRioMaisProximo** ( int *posX*, int *posY*, int *raio* ) |
| void | **preenchePontosContinuo** ( Point *pIni*, Point *pFim*, int *indCamada* ) |
| void | **preencheSolido** ( int *posClickY*, int *posClickX*, int *indCamada* ) |
| void | **setaTamanhoBarraDeZoom** () |
| void | **setaTelaPadrao** () |

## Public Variables

| | |
|---|---|
| bool | **escolhendoPontosCamada** |
| QList<Point> | **listaDePosicoesMouse** |

| | |
|---|---|
| map<string, VisualizacaoMapa> | **mapeamentoAreaProjeto** |
| bool | **pontosContinuos** |
| int | **posXBarraZoom** |
| int | **posXBotaoZoomMais** |
| int | **posXBotaoZoomMenos** |
| int | **posYBarraZoom** |
| int | **posYBotaoZoomMais** |
| int | **posYBotaoZoomMenos** |
| bool | **selecaoPontoBarragem** |
| bool | **selecaoPontos** |
| QStringList | **stringAreaProjeto** |

# Protected Slots

void **adicionandoPontosACamada** ( QMouseEvent * *evM* )
void **defineCamada** ( int *a*, int *b* )
void **defineInundacao** ( QMouseEvent * *ev* )
bool **eEntruncamentoDeRios** ( int *posx*, int *posy* )
void **exporCamada** ( QTableWidgetItem * *qtwItem* )
void **imprimeBarragem** ( VisualizacaoMapa *viMapa*, int *indBarragem* )
void **moveBarragem** ( QKeyEvent * *kev* )
void **movendoMouse** ( QMouseEvent * *ev* )
void **on_actionAbrir_mapa_triggered** ()
void **on_barraZoom_sliderMoved** ( int *position* )
void **on_botaoAtualizarFluxoMinimo_clicked** ()
void **on_botaoZoomMais_clicked** ()
void **on_botaoZoomMenos_clicked** ()
void **trocaCoordenadas** ( QMouseEvent * *q* )

# Detailed Description

Class to create the main window

# Member Function Documentation

### JanelaPrincipal::JanelaPrincipal ( QWidget * *parent* = 0 )

### JanelaPrincipal::~JanelaPrincipal ()

Destructor



### void JanelaPrincipal::adicionandoPontosACamada ( QMouseEvent * *evM* )

`[protected slot]`

Slot for when the mouse is released. Used for finish selecting points of the Camada.

### void JanelaPrincipal::atualizaCamposSaida ()

Function to update the fields to show user

### QPoint JanelaPrincipal::buscaMelhorPonto ( VisualizacaoMapa *viMapa*, QList<QPoint> *listaPontos*, double *volumeDesejado* )

Function to find the best dam position in a stretch of drainage network

### void JanelaPrincipal::criaListaComCamadas ()

Function to construct the Camada mapping for the current MDE showing in Area de Visualizacao

### void JanelaPrincipal::criaVisualizacaoBarragem ()

Function to decide and invoke the 2D or 3D visualization

### void JanelaPrincipal::defineCamada ( int *a*, int *b* ) `[protected slot]`

Slot for when a cell of Camada map is selected. Is not in use

### void JanelaPrincipal::defineInundacao ( QMouseEvent * *ev* ) `[protected slot]`

Slot for when mouse is pressed. It has a selector to decide what function call (the mousePressionadoParaCamada(QMouseEvent * ev) or the mousePressionadoParaBarragem(QMouseEvent * ev))

### void JanelaPrincipal::deletaListaComCamadas ()

Function to clear the Camada mapping for the current MDE showing in Area de Visualizacao. Doesnt delete the Camada associated with the MDE

### void JanelaPrincipal::desAbilitandoBotoes ()

Function to disable the buttons, fields and others.

### bool JanelaPrincipal::eEntruncamentoDeRios ( int *posx*, int *posy* ) `[protected slot]`

Function to define that a cell in drainage network has more than two neighbors that is in drainage network too

### void JanelaPrincipal::exporCamada ( QTableWidgetItem * *qtwItem* ) `[protected slot]`

Slot for when item of Camada map`s is changed. Used to show or not the Camada to the user

### void JanelaPrincipal::habilitandoBotoes ()

Function to enable the buttons, fields and others.

### void JanelaPrincipal::imprimeBarragem ( VisualizacaoMapa *viMapa*, int *indBarragem* ) [protected slot]

Function to print in a file the MDE with dam characteristics. Used when is necessary to avaliate differents dams

### void JanelaPrincipal::mousePressionadoParaBarragem ( QMouseEvent * *ev* )

Function dedictated for separated the functions that is invoked when the mouse is pressed and the Position Dam is actived

### void JanelaPrincipal::mousePressionadoParaCamada ( QMouseEvent * *ev* )

Function dedictated for separated the functions that is invoked when the mouse is pressed and the Selecting Camada is actived

### void JanelaPrincipal::moveBarragem ( QKeyEvent * *kev* ) [protected slot]

Slot for receive command when key is pressed. Used to repositioning the dam in new position montante/juzante of the current dam position

### void JanelaPrincipal::movendoMouse ( QMouseEvent * *ev* ) [protected slot]

Slot for receive command when mouse is moving. Used to create a line for Camada

### void JanelaPrincipal::on_actionAbrir_mapa_triggered () [protected slot]

Slot for when option actionAbrir_mapa is pressed

### void JanelaPrincipal::on_barraZoom_sliderMoved ( int *position* ) [protected slot]

Slot for when barraZoom is moved

### void JanelaPrincipal::on_botaoAtualizarFluxoMinimo_clicked () [protected slot]

Slot for when button botaoAtualizarFluxoMinimo is pressed. Use to update the drainage network based on this value

### void JanelaPrincipal::on_botaoZoomMais_clicked () [protected slot]

Slot for when button botaoZoomMais is pressed. Zoom in the map

### void JanelaPrincipal::on_botaoZoomMenos_clicked () [protected slot]

Slot for when button botaoZoomMenos is pressed. Zoom out the map

### bool JanelaPrincipal::pontoDoRioMaisProximo ( int *posX*, int *posY*, int *raio* )

Function to attract the click of mouse for a drainage network cell

### void JanelaPrincipal::preenchePontosContinuo ( Point *pIni*, Point *pFim*, int *indCamada* )

Function to help the system for creating a continuos line when user is choosing the cells of Camada

### void JanelaPrincipal::preencheSolido ( int *posClickY*, int *posClickX*, int *indCamada* )

Function to fill the solid drawed by user before

### void JanelaPrincipal::setaTamanhoBarraDeZoom ()

Function to set the size of zoom bar

### void JanelaPrincipal::setaTelaPadrao ()

Function to set the default window with clear fields, disable buttons and others

### void JanelaPrincipal::trocaCoordenadas ( QMouseEvent * *q* ) [protected slot]

Slot for when mouse is moved. Used to set the position of the mouse in the main window

# Member Variable Documentation

### bool JanelaPrincipal::escolhendoPontosCamada

This variable holds the flag to control if the user is selecting Camada's points in map or not.

### QList<Point> JanelaPrincipal::listaDePosicoesMouse

This variable holds the list with mouse positions for helping to create a continuos line. With the last and the before last create a cell's continuos line.

### map<string, VisualizacaoMapa> JanelaPrincipal::mapeamentoAreaProjeto

This variable holds a map to mapping stored VisualizacaoMapa in Area de Projeto by its name map to its VisualizacaoMapa.

## bool JanelaPrincipal::pontosContinuos

This variable holds the flag to control if will have continuos line or not.

## int JanelaPrincipal::posXBarraZoom

This variable holds the position of bar BarraZoom in axis-x. It's help when scroll bar rolls to make always visible.

## int JanelaPrincipal::posXBotaoZoomMais

This variable holds the position of button BotaoZoomMais in axis-x. It's help when scroll bar rolls to make always visible.

## int JanelaPrincipal::posXBotaoZoomMenos

This variable holds the position of button BotaoZoomMenos in axis-x. It's help when scroll bar rolls to make always visible.

## int JanelaPrincipal::posYBarraZoom

This variable holds the position of bar BarraZoom in axis-y. It's help when scroll bar rolls to make always visible.

## int JanelaPrincipal::posYBotaoZoomMais

This variable holds the position of button BotaoZoomMais in axis-y. It's help when scroll bar rolls to make always visible.

## int JanelaPrincipal::posYBotaoZoomMenos

This variable holds the position of button BotaoZoomMenos in axis-y. It's help when scroll bar rolls to make always visible.

## bool JanelaPrincipal::selecaoPontoBarragem

This variable holds the flag to control if user will add point to Camada or choose the dam position when mouse is pressed.

## bool JanelaPrincipal::selecaoPontos

This variable holds the flag to control if user will add or sub cells from the Camada.

## QStringList JanelaPrincipal::stringAreaProjeto

This variable holds a list with the names in mapeamentoAreaProjeto to help find an element in it.

### Contents



# janelasobre Class Reference

Class to create the about window More...

```
#include <janelasobre>
```

- List of all members, including inherited members

## Public Functions

**janelasobre** ( QWidget * *parent* = 0 )
**~janelasobre** ()

## Detailed Description

Class to create the about window

## Member Function Documentation

### janelasobre::janelasobre ( QWidget * *parent* = 0 )

QDialog(parent), ui(new Ui::janelasobre)

Constructor

### janelasobre::~janelasobre ()

Destructor

### Contents



# MapaMDE Class Reference

Class for create the MDE of terrain More...

```
#include <MapaMDE>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **MapaMDE** ( const char * *caminhoArquivo* ) |
| | **MapaMDE** ( MapaMDE const & *mapaCopia* ) |
| | **MapaMDE** () |
| | **~MapaMDE** () |
| char * | **getCaminhoArquivo** () |
| short int | **getDadoInvalido** () |
| int | **getNColunas** () |
| int | **getNLinhas** () |
| char * | **getNomeArquivo** () |
| short int | **getResolucao** () |

## Public Variables

| | |
|---|---|
| short int ** | **matrizDeElevacoes** |
| int | **maxElev** |
| int | **minElev** |

## Detailed Description

Class for create the MDE of terrain

## Member Function Documentation

### MapaMDE::MapaMDE ( const char * *caminhoArquivo* )

Constructor. Creates a object (MDE) for especified caminhoArquivo path

### MapaMDE::MapaMDE ( MapaMDE const & *mapaCopia* )

Copy Constructor. Creates an object receiving attributes from mapaCopia

### MapaMDE::MapaMDE ()

Constructor. Default Constructor. Do nothing

### MapaMDE::~MapaMDE ()

Destructor.

### char * MapaMDE::getCaminhoArquivo ()

Returns the full path of file contain the MDE

### short int MapaMDE::getDadoInvalido ()

Returns the value of invalid data

### int MapaMDE::getNColunas ()

Returns the number of columns

### int MapaMDE::getNLinhas ()

Returns the number of lines

### char * MapaMDE::getNomeArquivo ()

Returns the name of the file MDE

### short int MapaMDE::getResolucao ()

Returns the resolution of the cell

## Member Variable Documentation

### short int ** MapaMDE::matrizDeElevacoes

This variable holds a pointer to pointer, representing a matrix of status of the cell. If it is dam, drainage network, flooded or nothing.



### int MapaMDE::maxElev

This variable holds the maximum cell's elevation.

## int MapaMDE::minElev

This variable holds the minimun cell's elevation.

[Previous: janelasobre] [Next: painelvisualizacaobarragem2dopengl]

### Contents



# painelvisualizacaobarragem2dopengl Class Reference

Class to create a opengl painel with dam 2D visualization More...

```
#include <painelvisualizacaobarragem2dopengl>
```

- List of all members, including inherited members

## Public Functions

**painelvisualizacaobarragem2dopengl** ( QWidget * *parent* = 0 )
void **carregandoInformacoes** ( VisualizacaoMapa * *viMapa* )
void **defineBarragem** ()

## Protected Functions

void **defineOrtho** ()
void **initializeGL** ()
void **paintGL** ()
void **resizeGL** ( int *w*, int *h* )

## Detailed Description

Class to create a opengl painel with dam 2D visualization

## Member Function Documentation

### painelvisualizacaobarragem2dopengl::painelvisualizacaobarragem2dopengl ( QWidget * *parent* = 0 )

### void painelvisualizacaobarragem2dopengl::carregandoInformacoes (

VisualizacaoMapa * *viMapa* )

**void painelvisualizacaobarragem2dopengl::defineBarragem ()**

**void painelvisualizacaobarragem2dopengl::defineOrtho ()** `[protected]`

**void painelvisualizacaobarragem2dopengl::initializeGL ()** `[protected]`

**void painelvisualizacaobarragem2dopengl::paintGL ()** `[protected]`

**void painelvisualizacaobarragem2dopengl::resizeGL ( int *w*, int *h* )** `[protected]`

[Previous: MapaMDE] [Next: PainelVisualizacaoBarragemOpenGL]

### Contents



# PainelVisualizacaoBarragemOpenGL Class Reference

Class to create a opengl painel with dam 3D visualization More...

```
#include <PainelVisualizacaoBarragemOpenGL>
```

- List of all members, including inherited members

## Public Functions

**PainelVisualizacaoBarragemOpenGL** ( QWidget * *parent* = 0 )
void **carregandoInformacoes** ( VisualizacaoMapa * )
void **desenhaTerreno** ()

## Public Slots

void **keyPressEvent** ( QKeyEvent * )

## Protected Functions

void **defineOrtho** ()
void **initializeGL** ()
void **paintGL** ()
void **resizeGL** ( int *w*, int *h* )

## Detailed Description

Class to create a opengl painel with dam 3D visualization

## Member Function Documentation

**PainelVisualizacaoBarragemOpenGL::PainelVisualizacaoBarragemOpenGL ( QWidget * *parent* = 0 )**

**void PainelVisualizacaoBarragemOpenGL::carregandoInformacoes ( VisualizacaoMapa * )**

**void PainelVisualizacaoBarragemOpenGL::defineOrtho () [protected]**

**void PainelVisualizacaoBarragemOpenGL::desenhaTerreno ()**

**void PainelVisualizacaoBarragemOpenGL::initializeGL () [protected]**

**void PainelVisualizacaoBarragemOpenGL::keyPressEvent ( QKeyEvent * ) [slot]**

**void PainelVisualizacaoBarragemOpenGL::paintGL () [protected]**

**void PainelVisualizacaoBarragemOpenGL::resizeGL ( int *w*, int *h* ) [protected]**

[Previous: painelvisualizacaobarragem2dopengl] [Next: PainelVisualizacaoOpenGL]

- Modules
- PainelVisualizacaoOpenGL

[Previous: PainelVisualizacaoBarragemOpenGL] [Next: VisualizacaoMapa]

### Contents



# PainelVisualizacaoOpenGL Class Reference

```
#include <PainelVisualizacaoOpenGL>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **PainelVisualizacaoOpenGL** ( QWidget * *parent* = 0 ) |
| void | **carregandoMapa** ( const char * *arquivo* ) |
| void | **computaFluxo** () |
| void | **defineOrtho** () |
| QColor | **getCorGradient** ( double *gradientProporcao* ) |
| void | **mudaCores** ( QStringList *listaCores* ) |
| void | **pintaCamadas** () |
| void | **pintaEixo** () |
| void | **pintaTudo** () |
| void | **setCoresPadrao** () |
| void | **zoomMapa** ( int *zoom* ) |

## Signals

| | |
|---|---|
| void | **keyPressEvent** ( QKeyEvent * ) |
| void | **mouseMoveEvent** ( QMouseEvent * ) |
| void | **mousePressEvent** ( QMouseEvent * *event* ) |
| void | **mouseReleaseEvent** ( QMouseEvent * ) |

## Public Variables

| | |
|---|---|
| QColor | **corAlagamento** |

QColor **corBarragem**
QColor **corMaiorElevacao**
QColor **corMenorElevacao**
QColor **corPadraoAlagamento**
QColor **corPadraoBarragem**
QColor **corPadraoMaiorElevacao**
QColor **corPadraoMenorElevacao**
QColor **corPadraoRedeDeDrenagem**
QColor **corRedeDeDrenagem**
bool **flagPinta**
bool **mapaEstaCarregado**
int **posClickX**
int **posClickY**

# Protected Functions

void **initializeGL** ()
void **paintGL** ()
void **resizeGL** ( int *width*, int *height* )

# Detailed Description

Class to create a opengl painel to show user the MDE

# Member Function Documentation

## PainelVisualizacaoOpenGL::PainelVisualizacaoOpenGL ( QWidget * *parent* = 0 )

Constructor

## void PainelVisualizacaoOpenGL::carregandoMapa ( const char * *arquivo* )

Function to create and load a new MDE to show user.

## void PainelVisualizacaoOpenGL::computaFluxo ()

Function to call the function for calculate the flow

## void PainelVisualizacaoOpenGL::defineOrtho ()

Function to definy the ortho of opengl painel



## QColor PainelVisualizacaoOpenGL::getCorGradient ( double *gradientProporcao*

)

Function to discovery a color from a gradient made by the maximum and minimum cell`s elevation

### void PainelVisualizacaoOpenGL::initializeGL () [protected]

This function overloads PainelVisualizacaoOpenGL::initializeGL().

Opengl functions needed. It is the init function of opengl painel

### void PainelVisualizacaoOpenGL::keyPressEvent ( QKeyEvent * ) [signal]

Signal for keyPressEvent. The slot JanelaPrincipal::moveBarragem(QKeyEvent *) catch this signal

### void PainelVisualizacaoOpenGL::mouseMoveEvent ( QMouseEvent * ) [signal]

Signal for mouseMoveEvent. The slot JanelaPrincipal::trocaCoordenadas(QMouseEvent *) and JanelaPrincipal::movendoMouse(QMouseEvent *) catch this signal

### void PainelVisualizacaoOpenGL::mousePressEvent ( QMouseEvent * *event* ) [signal]

Signal for mousePressEvent. The slot JanelaPrincipal::defineInundacao(QMouseEvent *) catch this signal

### void PainelVisualizacaoOpenGL::mouseReleaseEvent ( QMouseEvent * ) [signal]

Signal for mouseReleaseEvent. The slot JanelaPrincipal::adicionandoPontosACamada(QKeyEvent *) catch this signal

### void PainelVisualizacaoOpenGL::mudaCores ( QStringList *listaCores* )

Function to modify the color`s variable. The new colors is stored by its name in the list listaCores.

### void PainelVisualizacaoOpenGL::paintGL () [protected]

Opengl functions needed. For opengl painting. Decide here if it will paint all or a part of scenario.

### void PainelVisualizacaoOpenGL::pintaCamadas ()

Function to draw only the Camadas in listaDeCamadas

### void PainelVisualizacaoOpenGL::pintaEixo ()

Function to draw a guidance axis



### void PainelVisualizacaoOpenGL::pintaTudo ()

Function to draw everything

### void PainelVisualizacaoOpenGL::resizeGL ( int *width*, int *height* ) [protected]

Opengl functions needed. For when painel is resized or changed

### void PainelVisualizacaoOpenGL::setCoresPadrao ()

Function to set the default colors to the color`s variable

### void PainelVisualizacaoOpenGL::zoomMapa ( int *zoom* )

Function to update the zoom in the opengl painel

## Member Variable Documentation

### QColor PainelVisualizacaoOpenGL::corAlagamento

This variable holds the color value for the flooded area.

### QColor PainelVisualizacaoOpenGL::corBarragem

This variable holds the color value for the dam.

### QColor PainelVisualizacaoOpenGL::corMaiorElevacao

This variable holds the color value for the higher cell`s elevation.

### QColor PainelVisualizacaoOpenGL::corMenorElevacao

This variable holds the color value for the higher cell`s elevation.

### QColor PainelVisualizacaoOpenGL::corPadraoAlagamento

This variable holds the default color value for the flooded area.

### QColor PainelVisualizacaoOpenGL::corPadraoBarragem

This variable holds the color value for the dam.

### QColor PainelVisualizacaoOpenGL::corPadraoMaiorElevacao

This variable holds the default color value for the higher cell`s elevation.

### QColor PainelVisualizacaoOpenGL::corPadraoMenorElevacao

This variable holds the default color value for the higher cell`s elevation.

## QColor PainelVisualizacaoOpenGL::corPadraoRedeDeDrenagem

This variable holds the defaultcolor value for the drainage network.

## QColor PainelVisualizacaoOpenGL::corRedeDeDrenagem

This variable holds the color value for the drainage network.

## bool PainelVisualizacaoOpenGL::flagPinta

This variable holds the flag to control the opengl painel need to be all painted or only Camadas painted.

## bool PainelVisualizacaoOpenGL::mapaEstaCarregado

This variable holds the flag to control if the MDE is open.

## int PainelVisualizacaoOpenGL::posClickX

This variable holds the value in axis-x for click position.

## int PainelVisualizacaoOpenGL::posClickY

This variable holds the value in axis-y for click position.

[Previous: PainelVisualizacaoBarragemOpenGL] [Next: VisualizacaoMapa]

### Contents



# VisualizacaoMapa Class Reference

Class to represent a visualization of all contents in the process of creating a reservoir More...

```
#include <VisualizacaoMapa>
```

- List of all members, including inherited members

## Public Functions

| | |
|---|---|
| | **VisualizacaoMapa** () |
| | **VisualizacaoMapa** ( const char * *caminhoMapa*, int *largura*, int *altura* ) |
| | **VisualizacaoMapa** ( VisualizacaoMapa const & *viMapa* ) |
| | **~VisualizacaoMapa** () |
| void | **atualizaCamadasSomadas** () |
| void | **atualizaPesoCamadas** () |
| void | **atualizaPesos** ( int *aa*, int *ab*, int *c*, int *eb*, int *hb* ) |
| void | **encontraPontosJusante** ( int *x*, int *y*, bool & *encontrado*, bool ** *matrizVisitados*, int & *pontosBuscados* ) |
| int | **getFuncaoObjetivo** () |
| int | **getIndCamada** ( QString *nome* ) |
| MapaMDE * | **getMapa** () |
| double | **getProporcaoZoom** () |
| int | **getZoom** () |
| void | **insereCamada** ( Camada *c* ) |
| void | **setInundacao** ( int *posX*, int *posY*, int *op*, int *val*, int *valEp* ) |
| void | **setTamanhoPonto** ( int *larguraTela*, int *alturaTela* ) |
| void | **setZoom** ( int *zoom* ) |
| QList<Camada> & | **operator=** ( QList<Camada> const & *listaCamada* ) |
| VisualizacaoMapa & | **operator=** ( VisualizacaoMapa const & *viMapa* ) |
| bool | **operator==** ( QList<Camada> const & ) |

# Public Variables

| | |
|---|---|
| short int ** | **camadasSomadas** |
| double | **constanteDeProporcaoZoom** |
| bool | **estaPreSalvo** |
| Fluxo * | **fluxo** |
| Inundacao * | **inundacao** |
| QList<Camada> * | **listaDeCamadas** |
| MapaMDE * | **mapa** |
| int | **marcaFinalX** |
| int | **marcaFinalY** |
| int | **marcaInicioX** |
| int | **marcaInicioY** |
| bool | **marcouFim** |
| bool | **marcouInicio** |
| int | **maxPointSize** |
| int | **maxPontosABuscar** |
| int | **pesoABarragem** |
| int | **pesoAreaAlagada** |
| int | **pesoCamadas** |
| int | **pesoEBarragem** |
| int | **pesoHBarragem** |
| int | **pesoVolume** |
| double | **proporcaoZoom** |
| int | **tamanhoDaBarraDeZoom** |
| double ** | **vetorDeZooms** |

# Detailed Description

Class to represent a visualization of all contents in the process of creating a reservoir

# Member Function Documentation

## VisualizacaoMapa::VisualizacaoMapa ()

Default Constructor. Creates a object with its pointers set to null

## VisualizacaoMapa::VisualizacaoMapa ( const char * *caminhoMapa*, int *largura*, int *altura* )



Constructor. Creates a object VisualizacaoMapa for the path caminhoMapa

### VisualizacaoMapa::VisualizacaoMapa ( VisualizacaoMapa const & *viMapa* )

Copy Constructor. Creates an object receiving attributes from viMapa

### VisualizacaoMapa::~VisualizacaoMapa ()

Destructor.

### void VisualizacaoMapa::atualizaCamadasSomadas ()

Function to sum all Camadas in listaDeCamadas, cell by cell. The answer is seted in th matrix camadasSomadas

### void VisualizacaoMapa::atualizaPesoCamadas ()

Function to update the value of the Camada`s weight. Calculate price of all flooded cell.

### void VisualizacaoMapa::atualizaPesos ( int *aa*, int *ab*, int *c*, int *eb*, int *hb* )

Function to update the value of the weights

### void VisualizacaoMapa::encontraPontosJusante ( int *x*, int *y*, bool & *encontrado*, bool ** *matrizVisitados*, int & *pontosBuscados* )

Function to look for points(and creates the list of candidates to position the dam) until the end point of stretch drainage network is founded.

### int VisualizacaoMapa::getFuncaoObjetivo ()

Function to return the value of objective function with the weights.

### int VisualizacaoMapa::getIndCamada ( QString *nome* )

Function to return the index of Camada named as nome in listaDeCamadas

### MapaMDE * VisualizacaoMapa::getMapa ()

Returns mapa

### double VisualizacaoMapa::getProporcaoZoom ()

Function to return the proportion of the zoom

### int VisualizacaoMapa::getZoom ()

Returns zoom



### void VisualizacaoMapa::insereCamada ( Camada *c* )

Function to insert Camada c in listaDeCamadas

### void VisualizacaoMapa::setInundacao ( int *posX*, int *posY*, int *op*, int *val*, int *valEp* )

Function to create a reservoir.

### void VisualizacaoMapa::setTamanhoPonto ( int *larguraTela*, int *alturaTela* )

Function to calculate and set the point size

### void VisualizacaoMapa::setZoom ( int *zoom* )

Function to set the zoom value

### QList<Camada> & VisualizacaoMapa::operator= ( QList<Camada> const & *listaCamada* )

This function overloads VisualizacaoMapa::operator=(QList<Camada> const& listaCamada).

Operator to set listaCamada to variable listaDeCamadas

### VisualizacaoMapa & VisualizacaoMapa::operator= ( VisualizacaoMapa const & *viMapa* )

This function overloads VisualizacaoMapa::operator=(VisualizacaoMapa const& viMapa).

Overloaded operator to assign the object viMapa to the current object(this)

### bool VisualizacaoMapa::operator== ( QList<Camada> const & )

## Member Variable Documentation

### short int ** VisualizacaoMapa::camadasSomadas

This variable holds a pointer to pointer, representing a matrix of sum of Camadas to help in position algorithm.

### double VisualizacaoMapa::constanteDeProporcaoZoom

This variable holds the value for a constant variable used in proporcaoZoom.

### bool VisualizacaoMapa::estaPreSalvo

This variable holds the flag to control if this visualization is saved in Area de Projeto.



### Fluxo * VisualizacaoMapa::fluxo

This variable holds a pointer to Fluxo object.

### Inundacao * VisualizacaoMapa::inundacao

This variable holds a pointer to Inundacao object.

### QList<Camada> * VisualizacaoMapa::listaDeCamadas

This variable holds a list of Camada to hold the Camadas create by user.

### MapaMDE * VisualizacaoMapa::mapa

This variable holds a pointer to MapaMDE object.

### int VisualizacaoMapa::marcaFinalX

This variable holds the end position in axis-x of the drainage network's stretch.

### int VisualizacaoMapa::marcaFinalY

This variable holds the end position in axis-y of the drainage network's stretch.

### int VisualizacaoMapa::marcaInicioX

This variable holds the start position in axis-x of the drainage network's stretch.

### int VisualizacaoMapa::marcaInicioY

This variable holds the start position in axis-y of the drainage network's stretch.

### bool VisualizacaoMapa::marcouFim

This variable holds the flag to hold if user has marked the end position of drainage network's stretch.

### bool VisualizacaoMapa::marcouInicio

This variable holds the flag to hold if user has marked the start position of drainage network's stretch.

### int VisualizacaoMapa::maxPointSize

This variable holds the maximum point's size.

### int VisualizacaoMapa::maxPontosABuscar

This variable holds the maximun iteration for recursive functions.

### int VisualizacaoMapa::pesoABarragem

This variable holds the value of the weight for dam's area. Used in position algorithm.

## int VisualizacaoMapa::pesoAreaAlagada

This variable holds the value of the weight for flooded area. Used in position algorithm.

## int VisualizacaoMapa::pesoCamadas

This variable holds the value of the sum of cell's in camadasSomadas that are in the resorvoir. Used in position algorithm.

## int VisualizacaoMapa::pesoEBarragem

This variable holds the value of the weight for dam's extent. Used in position algorithm.

## int VisualizacaoMapa::pesoHBarragem

This variable holds the value of the weight for dam's high. Used in position algorithm.

## int VisualizacaoMapa::pesoVolume

This variable holds the value of the weight for reservoir's volume. Used in position algorithm.

## double VisualizacaoMapa::proporcaoZoom

This variable holds an internal variable for the proportion of the zoom. It's used to definy the point size.

## int VisualizacaoMapa::tamanhoDaBarraDeZoom

This variable holds the size of zoom bar.

## double ** VisualizacaoMapa::vetorDeZooms

This variable holds a pointer to pointer, representig a matrix with the value of zooms possible in this MDE.

[Previous: PainelVisualizacaoOpenGL] [Next: All Classes]