Métodos Heurísticos e Exatos para o Problema de Inundação em Grafos

Bruno José da Silva Barros[1] **, Rian G. S. Pinheiro**[1] **, Uéverton dos Santos Souza**[2]

[1] Universidade Federal Rural de Pernambuco
Garanhuns – PE – Brasil
[2]Departamento de Informática – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
Rio de Janeiro – RJ – Brasil

{brrsbruno,rian.gabriel,usouza}@gmail.com

RESUMO

INUNDAÇÃO é um jogo combinatório jogado em um grafo colorido G, cujo objetivo é fazer com que o grafo se torne monocromático usando o número mínimo de movimentos. Um movimento consiste de atribuir uma nova cor c à casa pivô e também a toda casa conectada a p. Neste trabalho, foram propostos uma formulação matemática, quatro heurísticas construtivas e instâncias para o problema. No conhecimento dos autores, essa é a primeira formulação para o problema. Observou-se que o método exato foi capaz resolver todas as instâncias testadas em um baixo tempo computacionais, enquanto as heurísticas encontraram boas soluções em um tempo computacional muito inferior ao exato.
PALAVRAS CHAVE. Inundação. Heurísticas. Formulação Matemática.
Área principal OC - Otimização Combinatória.

ABSTRACT

Flood-it is a combinatorial game played on a colored graph G whose aim is to make the graph monochromatic usingthe minimum number of flooding moves. A move consists of assigning a color c to the pivot p and also to every vertex connected to p. In this work, we proposed a mathematical formulation, four constructive heuristics and new instances to the problem. Up to the author' knowledge, this is the first formulation to the problem. Computational experiments performed on instances show that our exact method was able to resolve all instances, and heuristics produces solutions close to the optimal solution with a low computational time.
KEYWORDS. Flood it Problem. Heuristics. Mathematical Formulation.
Main area OC - Combinatorial Optimization.

1 Introdução

O jogo INUNDAÇÃO é um jogo combinatório jogado por apenas um jogador, originalmente, em um tabuleiro de dimensão $n \times m$, onde cada casa deste tabuleiro possui uma coloração de entrada dentre um conjunto fixo de cores. Conforme descrito em (Souza , 2012), neste jogo, duas casas são consideradas vizinhas se elas pertencem a mesma linha (resp. coluna) e em consecutivas colunas (resp. linhas). Uma sequência C de casas é considerada um *caminho* quando todo par consecutivo de casas em C é formada de casas vizinhas. Um *caminho monocromático* é um caminho onde todas as casas possuem a mesma cor. Duas casas a e b são *conectadas* quando existe um caminho monocromático entre elas.

No jogo INUNDAÇÃO, a cada movimento o jogador deve atribuir uma nova cor c a casa mais à esquerda do topo do tabuleiro, denotamos por p ("pivô") a casa do tabuleiro que encontra-se nesta posição. Ao atribuir uma nova coloração c a casa p, todas as casas do tabuleiro conectadas a p também terão a sua coloração alterada para c. Note que após uma movimentação do jogo, o número de casas conectadas a p eventualmente será incrementado. O objetivo do jogo é fazer o tabuleiro monocromático ("inundar o tabuleiro") com o número mínimo de movimentos possíveis. A Figura 1 ilustra uma sequência de movimentos para inundar uma grade 3×3.

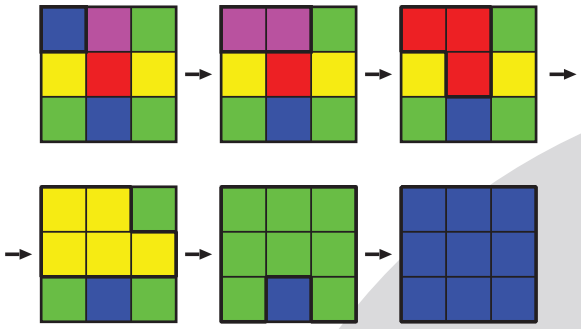


Figura 1: (Souza , 2012) Sequência ótima de movimentos para inundar uma grade 3×3.

O jogo INUNDAÇÃO é generalizado quando damos ao jogador a liberdade de escolher a cada movimento qual casa do tabuleiro será o pivô da jogada, esta versão do jogo é conhecida como INUNDAÇÃO LIVRE. Além disso, estes jogos podem facilmente ser generalizados para ser jogados sobre um grafo qualquer com uma coloração inicial ω.

Alguns modelos de propagação de doenças descrito em (Aschwanden , 2004) operam de forma similar a jogos de inundação. Em (Arthur , 2010), Arthur, Clifford, Jalsenius, Montanaro e Sach mostraram que INUNDAÇÃO e INUNDAÇÃO LIVRE são NP-difíceis em grades $n \times n$ coloridas com mais que três cores. Em (Meeks , 2012), Meeks e Scott mostraram que INUNDAÇÃO LIVRE é solucionável em tempo polinomial em grades $1 \times n$, tanto INUNDAÇÃO quanto INUNDAÇÃO LIVRE permanecem NP-difícil para grades $3 \times n$ coloridas com pelo menos quatro cores. (A complexidade de INUNDAÇÃO (LIVRE) em grades $3 \times n$ coloridas com três cores permanece como um problema em aberto.) Clifford, Jalsenius, Montanaro e Sach apresentaram em (Clifford , 2011) um algoritmo de tempo polinomial para INUNDAÇÃO em grades $2 \times n$. Em (Meeks , 2013), Meeks e Scott mostraram que INUNDAÇÃO LIVRE permanece NP-difícil em grades $2 \times n$, eles também mostraram que INUNDAÇÃO LIVRE é solucionável em tempo polinomial em grafos dois coloridos. Em (Lagoutte , 2014), Lagoutte mostrou que INUNDAÇÃO é polinomialmente solucionável em ciclos, e INUNDAÇÃO e INUNDAÇÃO LIVRE são NP-difíceis em árvores.

Em (Souza , 2013; Fellows , 2015), Fellows, Souza, Protti, e Dantas da Silva mostraram que INUNDAÇÃO em árvores é análogo a um importante subcaso do problema *Supersequência*

Comum mais Curta. Em (Souza , 2014), Souza, Protti e Dantas da Silva descreveram algoritmos de tempo polinomial para o jogo em C_n^2 ou P_n^2 (a segunda potência de um ciclo ou um caminho com n vértices) e $2 \times n$ grades circulares, e Fellows, Souza, Protti, e Dantas da Silva (Fellows , 2015) desenvolveram uma investigação multivariada sobre a complexidade de INUNDAÇÃO em árvores, analisando as consequências de parametrizar o problema de várias maneiras.

1.1 Definições e Notações

Ao longo do texto, é utilizado o termo *vértice* em vez de *casa* quando conveniente. Casas vizinhas naturalmente correspondem a vértices vizinhos de um grafo G representando o tabuleiro. Um subgrafo H de G é *adjacente* a um vértice $v \in V(G)$ se v tem um vizinho em $V(H)$. O *pivô* de um movimento é o vértice escolhido para ter sua cor alterada por m, onde m é a m-ésima jogada feita pelo jogador. Um subgrafo H é dito ser *inundado*, quando H torna-se monocromático. Uma inundação(-livre) é uma sequência de movimentos no jogo INUNDAÇÃO(LIVRE) que inunda G (o tabuleiro de entrada). Uma inundação(-livre) ótima é uma inundação com o número mínimo de movimentos. Uma cor c é *jogada em um movimento* m se c é a cor atribuída ao pivô de m. Um vértice v é *inundado por um movimento* m se a cor de v é jogada em m e v torna-se conectado a novos vértices depois da aplicação de m. Um movimento m é *economizado* se m inunda pelo menos dois vértices. Um movimento m é dito ser *jogado em um subconjunto de vértices* S (resp. *um subgrafo* H) se pelo menos um vértice de S (resp. H) é inundado por m. Um subgrafo monocromático H' de um subgrafo H é abreviado como *um mcs de* H. No jogo INUNDAÇÃO denotamos por $P(G)$ o mcs máximo de G contendo o pivô. Finalmente, dizemos que um movimento m *inunda um vértice* v *através de um vértice* w se v e w são vizinhos e m altera a cor de w para inundar v. Uma *grade circular* é uma grade $n \times m$ com a propriedade adicional de que a primeira e ultima casa de uma mesma linha são consideradas vizinhas.

Um componente monocromático é definido como um subgrafo maximal $G' = (V', E')$ de G na qual todos os vértices possuem a mesma cor, e para cada par de vértice v, w, existe um caminho monocromático de v até w. A Figura 2 ilustra um componente monocromático azul claro em uma grade 5×5. A distância de um subgrafo (componente monocromático) a um vértice v é definida como

$$d(v, G') = \min_{w \in V(G')} d(v, w).$$

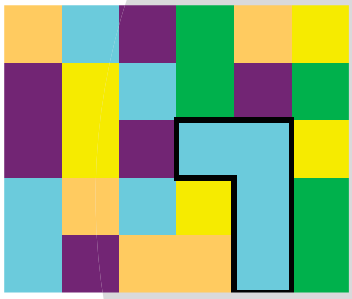


Figura 2: Componente monocromático azul claro em uma grade 5×5.

A seguir será apresentada a definição formal do problema INUNDAÇÃO.

INUNDAÇÃO
Instância: Um grafo $G = (V, E)$ em que cada vértice v possui uma cor c_v e um vértice pivô p.
Objetivo: Determinar a sequência de movimentos para inundar G que possua o comprimento mínimo.

Neste trabalho, foi proposto uma nova formulação matemática que é apresentada na Seção 2. Na Seção 3 são propostas quatro heurísticas construtivas. Já a Seção 4 apresenta novas instâncias para o problema além dos resultados computacionais. Por fim, a Seção 5 discute as conclusões e trabalhos futuros.

2 Formulação Matemática

Nesta seção será apresentada uma formulação matemática para o Problema INUNDAÇÃO. Os autores desconhecem formulações matemática para o problema. Nesta formulação um grafo $G = (B, E)$ em que cada vértice é um vértice da grade original ou um componente monocromático da grade original. A formulação é definida como:

$$\min \quad z \tag{1}$$

$$\text{s.a:} \quad y_{p,0} = 1 \tag{2}$$

$$\sum_{t=0}^{|B|} y_{b,t} = 1 \qquad \forall b \in B \tag{3}$$

$$\sum_{c=0}^{|C|} x_{c,t} = 1 \qquad \forall t \in [0, |B|] \tag{4}$$

$$y_{b,t} \leq x_{c(b),t} \qquad \forall (b,t) \in B \times [0, |B|] \tag{5}$$

$$y_{b,t} \leq \sum_{u \in N(b)} \sum_{i=0}^{t-1} y_{u,i} \qquad \forall (b,t) \in B \times [0, |B|] \tag{6}$$

$$z \geq t\, y_{b,t} \qquad \forall (b,t) \in B \times [0, |B|] \tag{7}$$

$$y_{b,t} \in \mathbb{B} \qquad \forall (b,t) \in B \times [0, |B|] \tag{8}$$

$$x_{c,t} \in \mathbb{B} \qquad \forall (c,t) \in C \times [0, |B|] \tag{9}$$

Nesta formulação z representa o número de movimentos, B o conjunto de componentes monocromáticos e C o conjunto de cores. A variável binária $y_{b,t}$ vale 1 se e somente se o componente monocromático $b \in B$ for inundado no movimento $t \in \{0, |B|\}$. Já a variável binária $x_{c,t}$ vale 1 se e somente se o componente monocromático pivô p mudou para a cor $c \in C$ no movimento t.

Função Objetivo (1), minimiza o número de movimentos. A restrição (2) indica que o componente monocromático pivô já está inundado. O conjunto de restrições (3) indica que cada componente monocromático deve ser inundado uma única vez. Em (4), cada movimento deve possuir uma única modificação de cor. Nas restrições (5), cada componente monocromático b só pode ser inundado no movimento t se no movimento t for atribuída a cor $c(b)$ (cor inicial do componente monocromático b). As restrições (6) indicam que cada componente monocromático b só pode ser inundado no movimento t se pelo menos um componente monocromático vizinho já estiver sido inundado. Em (7), o número de movimentos z é definido como o movimento do último componente monocromático inundado. Por fim, (8) e (9) definem o domínio das variáveis.

3 Algoritmos Heurísticos

Nesta seção, serão propostas quatro heurísticas construtivas para a criação de uma solução para o problema de INUNDAÇÃO. A primeira é uma heurística gulosa e as outras três são heurísticas

baseadas no vértice mais distante. Todos os algoritmos descritos possuem como entrada um grafo $G = (B, E)$ em que cada vértice representa um componente monocromático na grade original e um conjunto de cores C.

3.1 Heurística Gulosa

O algoritmo proposto se propõe a inundar todo o grafo de forma simples. Primeiramente, o subgrafo $I \subseteq G$ representa o subgrafo inundado a cada iteração do algoritmo. Inicialmente esse subgrafo contém apenas o pivô, pois este está sempre inundado. Agora considere o conjunto $N(I) = \{v | vw \in E \text{ e } w \in I\}$, ou seja, os vértices que já tem pelo menos um vizinho inundado mas que ainda não foram inundados. Calcula-se agora qual das cores que podem ser jogadas irá inundar mais vértices, seja essa a cor c, então inunda-se o conjunto $N_c(I) \subseteq N(I)$ formado pelo vértice em $N(I)$ de cor c. Por fim, repete-se esse processo até que não sobrem mais vértices a serem inundados. O pseudocódigo da heurística gulosa pode ser visto no Algoritmo 1.

Algoritmo 1 Heurística Gulosa

```
1:  procedure GULOSO(grafo G, cores C)
2:      I ← {p}                                  ▷ Conjunto de vértices inundados
3:      S ← {}                                   ▷ Sequência de inundação
4:      while |I| ≠ |V(G)| do
5:          c ← cor mais frequente em N(I)
6:          S ← S + c                            ▷ Adiciona a cor c na sequência
7:          I ← I ∪ N_c(I)                       ▷ Inunda os vizinhos de I que possuem a cor c
8:      end while
9:      return S
10: end procedure
```

3.2 Heurística Flooding-I

O Algoritmo 2 tenta decidir qual melhor caminho para começar a inundar todo o grafo $G(B, E)$ com uma relação de cores C. No algoritmo proposto, primeiramente é calculada a distância entre o componente monocromático pivô os demais componentes monocromáticos. Com isso, o algoritmo escolhe o vértice mais distante do pivô. Seja P o caminho entre o pivô e o vértice mais distante, o algoritmo aplica mudanças de cor (movimentos) de acordo com as cores encontradas em P. Essa estratégia corresponde a função `inundaCaminho` na linha 5 do algoritmo. Note que um vértice v que não faz parte de P, pode ou não ter sido inundado ao fim da inundação P.

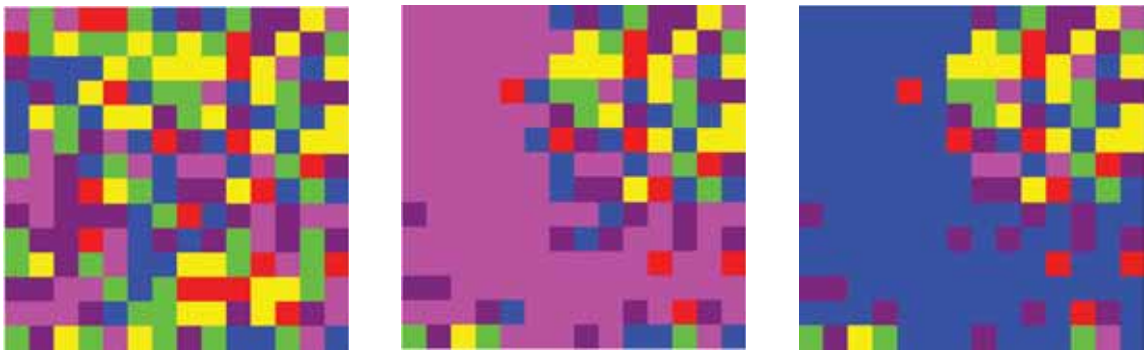
Algoritmo 2 Heurística Flooding-I

```
1: procedure FLOODING(grafo G, cores C)
2:     I ← {p}                                  ▷ Conjunto de vértices inundados
3:     while |I| ≠ |V(G)| do
4:         v_d ← maisDistante(I)   ▷ Seleciona o vértice v_d mais distante do conjunto de vértice I
5:         inundaCaminho(v_d, I, S)   ▷ Inunda o caminho de um vértice de I até o vértice mais
   distante v_d
6:     end while
7:     return S
8: end procedure
```

Quando todo caminho P for inundado, ainda podem ter restado vértices a serem inundados, então criam-se um novo grafo com a seguinte característica: o subgrafo inundado I, será convertido

em um único vértice no novo grafo, e este vértice será o pivô. Com o novo grafo criado, repete-se o procedimento até que todos os vértices sejam inundados.

A Figura 3 ilustra os passos do Algoritmo 2. Dado a tabuleiro inicial (Figura 3a), as Figuras 3b e 3c mostram a inundação do vértice mais distante.



(a) Início do Tabuleiro (b) Tabuleiro após 10 movimentos (c) Tabuleiro após 11 movimentos

Figura 3: Forma de inundar do Algoritmo 2 num tabuleiro 14×14 com 6 cores

3.3 Heurística Flooding-II

O Algoritmo 3 é uma mistura dos dois algoritmos anteriores, pois ele combina os dois algoritmos para tentar montar um novo algoritmo que gere soluções melhores que os anteriores. No inicio, o algoritmo realiza os mesmos passos que o Algoritmo 2 até que $\frac{3}{4}$ dos vértices tenham sido inundados. Quando essa condição for verdadeira, então o algoritmo utiliza a estratégia do guloso do Algoritmo 1.

Algoritmo 3 Heurística Flooding-II

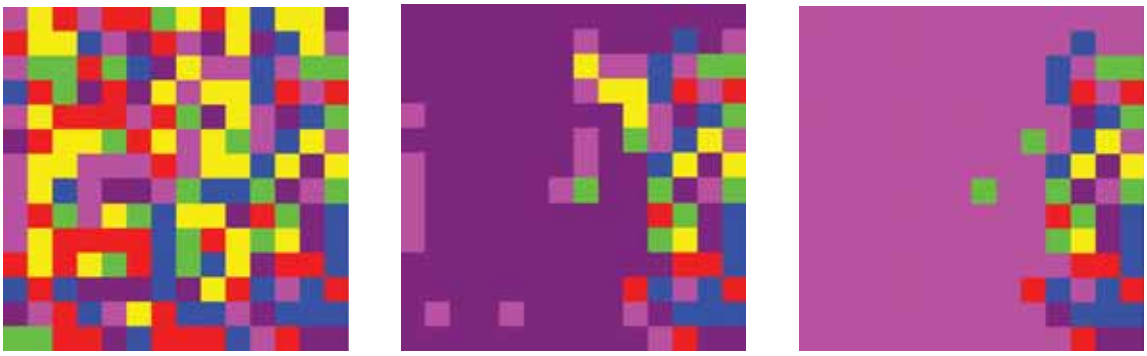
1: **procedure** FLOODING2(grafo G, cores C)
2: $I \leftarrow \{p\}$ ▷ Conjunto de vértices inundados
3: **while** $|V(G)| - |I| \geq \frac{1}{4}|V(G)|$ **do**
4: $v_d \leftarrow$ maisDistante(I) ▷ Seleciona o vértice v_d mais distante do conjunto de vértice I
5: inundaCaminho(v_d, I, S) ▷ Inunda o caminho de um vértice de I até o vértice mais distante v_d
6: **end while**
7: $G' \leftarrow$ novoGrafo(G, I, S, C)
8: $S \leftarrow S +$ GULOSO(G', C)
9: **return** S
10: **end procedure**

O valor $\frac{1}{4}|V(G)|$ foi obtido de maneira experimental num tabuleiro 14×14 com 6 cores. Esse algoritmo raramente apresenta uma solução com número de movimento maior que o Algoritmo 2.

3.4 Heurística Flooding-III

O Algoritmo 4 não difere muito do anterior, é apenas uma versão melhorada que visa tratar o seguinte caso. Quando se inunda todo o caminho P, cria-se uma forma de inundar desesperadamente um dos vértices que está mais distante do pivô. Isso cria um caminho monocromático entre eles, mas, no entanto, não leva em consideração os outros vértices do grafo.

A Figura 4 ilustra a execução do Algoritmo 4. Dado o tabuleiro inicial (Figura 4a), o objetivo era inundar o vértice mais distante (no caso o vértice inferior a direita). Ao inundar esse caminho, os últimos movimentos acabam inundando poucas casa, pois o objetivo é inundar uma casa particular. No entanto, inundar uma casa em questão não é uma boa jogada se ainda faltam muitas casas a serem inundadas e esse movimento inundar apenas ela. Então, para resolver esse problema o Algoritmo 4 não realiza os 4 últimos movimento do caminho P, e isso faz com que o conjunto I se aproxime de forma organizada aos demais vértices como mostram as Figuras 4b e 4c.



(a) Início do Tabuleiro (b) Tabuleiro após 15 movimentos (c) Tabuleiro após 17 movimentos

Figura 4: Forma de inundar do Algoritmo 4 num tabuleiro 14×14 com 6 cores

Sendo assim, este é o único detalhe que diferencia do algoritmo anterior no pseudocódigo é realizado através da função `inundaCaminhoParcial` na linha 5, que é justamente não fazer a inundação dos últimos 4 vértices do caminho P. Este algoritmo geralmente tem uma solução com número de movimentos igual ou menor aos anteriores.

Algoritmo 4 Heurística Flooding-III

1: **procedure** FLOODING3(grafo G, cores C)
2: $I \leftarrow \{p\}$ ▷ Conjunto de vértices inundados
3: **while** $|V(G)| - |I| \geq \frac{1}{4}|V(G)|$ **do**
4: $v_d \leftarrow$ maisDistante(I) ▷ Seleciona o vértice v_d mais distante do conjunto de vértice I
5: inundaCaminhoParcial(v_d, I, S) ▷ Inunda parcialmente o caminho de um vértice de I até o vértice mais distante v_d
6: **end while**
7: $G' \leftarrow$ novoGrafo(G, I, S, C)
8: $S \leftarrow S +$ GULOSO(G', C)
9: **return** S
10: **end procedure**

4 Experimentos Computacionais

Esta seção mostra as ferramentas utilizadas, a forma que as instâncias foram criadas além dos resultados alcançados com o método exato e as heurísticas descritas neste artigo.

4.1 Ferramentas

O método exato baseada na formulação matemática apresentada na Seção 2 foi desenvolvido utilizando a ferramenta CPLEX 12.5, que é um dos pacotes de *software* de otimização linear mista mais utilizados na literatura. Todas os métodos, exatos e heurísticos, foram desenvolvidos em C++ (gcc-4.9.0) e executados numa máquina Intel Core i3-2120 com 3,30 GHz e 8 Gb de memória RAM no sistema operacional Debian GNU/Linux 3.2.0-4 com 64 bits.

4.2 Geração de Instâncias

Como o problema ainda é pouco explorado, os autores não encontraram instâncias na literatura. Dessa forma, propõe-se a criação de instâncias aleatórias utilizando grafos grades com n vértices e um conjunto de c cores. Para escolher a cor de um vértice i, foi utilizada uma variável aleatória discreta uniforme. Dessa forma, foram criadas 20 instâncias com grades 14×14 e 6 cores de baseando-se no site `flooit` [1].

4.3 Resultados

Na Tabela 1, tem-se o número da instância seguido do número de movimentos obtidos pelos algoritmo propostos. Observa-se que o algoritmo exato obteve a solução ótima em tempo relativamente baixo, entretanto se levarmos em consideração que neste trabalho utilizamos apenas instâncias pequenas, este tempo não é tão baixo. Com relação às heurísticas a Flooding-III obteve os melhores resultados seguida da heurística Flooding-II. O método guloso obteve os piores resultados, no entanto ressalta-se que ela é utilizada nas duas últimas heurísticas.

A Tabela 2 apresenta os tempos computacionais dos métodos propostos, observa-se que o método exato conseguiu resolver as instâncias propostas em um tempo razoável (tempo máximo de 1 hora e 47 minutos). A heurística gulosa obteve os menores tempos de execução, mesmo assim, todas as heurísticas obtiveram baixos tempos computacionais.

Instância	Exato	Guloso	Flooding-I	Flooding-II	Flooding-III
1	20	23	25	26	**22**
2	22	28	28	27	**25**
3	19	29	**22**	**22**	24
4	21	25	**23**	24	**23**
5	21	**25**	29	28	29
6	20	26	25	**24**	**24**
7	19	25	22	**22**	25
8	20	31	27	**26**	**26**
9	21	31	**27**	**27**	28
10	22	31	30	**29**	30
11	21	29	31	30	**27**
12	22	29	28	27	**25**
13	21	29	26	26	**25**
14	22	31	25	25	**25**
15	21	29	**25**	26	26
16	20	27	22	22	**21**
17	21	31	**26**	27	**26**
18	21	31	29	**27**	31
19	21	34	**25**	**25**	28
20	21	30	**24**	**24**	**24**
Melhores Resultados	—	1	7	9	12

Tabela 1: Número de movimentos dos algoritmos propostos

5 Conclusão

Neste trabalho, foi abordado o Problema INUNDAÇÃO. Este problema possibilita a modelagem de diversos problemas reais e aplicações. Foram propostos uma formulação matemática e

[1] http://floodit.appspot.com/

Instância	Exato	Guloso	Flooding-I	Flooding-II	Flooding-III
1	1495,00	0,019	0,020	0,007	0,020
2	3286,36	0,008	0,033	0,007	0,010
3	199,22	0,007	0,006	0,006	0,011
4	6431,21	0,008	0,007	0,006	0,007
5	659,06	0,008	0,010	0,007	0,011
6	1116,90	0,008	0,008	0,007	0,014
7	151,27	0,006	0,007	0,006	0,012
8	355,98	0,008	0,023	0,009	0,041
9	1835,48	0,022	0,012	0,010	0,016
10	584,38	0,010	0,010	0,008	0,011
11	417,63	0,008	0,014	0,012	0,011
12	5854,25	0,009	0,023	0,021	0,015
13	5204,77	0,009	0,010	0,009	0,019
14	1197,41	0,010	0,008	0,007	0,009
15	5079,35	0,009	0,009	0,007	0,012
16	396,01	0,006	0,007	0,006	0,013
17	270,39	0,006	0,008	0,006	0,010
18	3453,87	0,010	0,009	0,007	0,023
19	1586,66	0,011	0,008	0,007	0,014
20	826,67	0,008	0,007	0,006	0,009
Total	40401,8	**0,190**	0,232	0,161	0,288

Tabela 2: Tempo de execução dos algoritmos propostos (em segundos)

quatro heurísticas construtivas. No conhecimento dos autores, essa é a primeira formulação para o problema. Além disso, foram criadas as primeiras instâncias para o problema.

Observou-se que o método exato foi capaz de encontrar a solução ótima em um tempo que variou de 3,5 a 107 minutos para todas as instâncias testadas, enquanto as heurísticas encontraram boas soluções em um tempo computacional muito inferior. Para trabalhos futuros, propõe-se o estudo de buscas locais e meta-heurísticas para este problema, assim como trabalhar com instâncias de tamanhos variados e com números diferentes de cores, além da derivação novas desigualdades válidas para incrementar na formulação proposta.

Referências

Arthur, D., Clifford, R., Jalsenius, M., Montanaro, A., and Sach, B. (2010). The complexity of flood filling games. *FUN, volume 6099 of Lecture Notes in Computer Science, Springer, ISBN 978-3-642-13121-9*, pages 307–318.

Aschwanden, C. (2004). Spatial simulation model for infectious viral disease with focus on sars and the common flu. *37th Annual Hawaii International Conference on System Sciences, IEEE Computer Society*.

Clifford, R., Jalsenius, M., Montanaro, A., and Sach, B. (2011). The complexity of flood filling games. *arXiv.1001.4420v3 [cs.DS]*.

Fellows, M. R., Souza, U. S., Protti, F., and da Silva, M. D. (2015). Tractability and hardness of flood-filling games on trees. *Theoretical Computer Science 576*, pages 102–116.

Lagoutte, A., Noual, M., and Thierry, E. (2014). Flooding games on graphs. *Discrete Applied Mathematics*, pages 532–538.

Meeks, K. and Scott, A. (2012). The complexity of flood-filling games on graphs. *Discrete Applied Mathematics 160*, pages 959–969.

Meeks, K. and Scott, A. (2013). The complexity of free-flood-it on 2xn boards. *Theoretical Computer Science 500*, pages 25–43.

Souza, U. S., Protti, F., and da Silva, M. D. (2012). Inundação em grafos. *XVI Congreso Latino Iberoamericano de Investigación Operativa & XLIV Simpósio Brasileiro de Pesquisa Operacional*.

Souza, U. S., Protti, F., and da Silva, M. D. (2013). Parameterized complexity of flood-filling games on trees. *Lecture Notes in Computer Science 7936*, pages 531–542.

Souza, U. S., Protti, F., and da Silva, M. D. (2014). An algorithmic analysis of flood-it and free-flood-it on graph powers. *Discrete Mathematics and Theoretical Computer Science 16:3*, pages 279–290.