UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ

INSTITUTO DE MATEMÁTICA – IM
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – DCC

Implementação do jogo Campo Minado utilizando Busca Local

Disciplina: Inteligência Artificial
Professor: Carla Delgado

Júlio César Machado Bueno	106033507
Carina Dias Sucupira	108056420
Guilherme Bomfim	108056519
Marcus Vinícuis do Couto	105063402

Sumário

Introdução.....3
Proposta de trabalho.....3
Solução Adotada.....3
Descrição detalhada da solução 4
Implementação realizada.....4
Telas da implementação.....7
Análise dos resultados e conclusões.....9

Introdução

Proposta do trabalho

O trabalho proposto foi a implementação do jogo conhecido como Campo Minado utilizando técnicas de Inteligência artificial. A proposta era gerar, utilizando algoritmo de Busca Local e parâmetros informados pelo usuário, uma representação do estado inicial do jogo. Não foi requerido que a implementação permitisse ao usuário jogá-lo.

Solução Adotada

Dados requisitos do problema, como a inserção de número de linhas, colunas, quantidade de bombas e nível de dificuldade, foi proposto uma interface gráfica onde usuário forneceria os dados iniciais do problema. A partir disto adotamos a seguinte abordagem:

Inicialmente, com base dos dados fornecidos pelo usuário, o jogo gera uma configuração que possui um distribuição aleatória de bombas e calcula os valores numéricos próximos a ela.

Classificamos então os jogos de acordo com a probabilidade do usuário realizar, com o campo disponível, o maior número de jogadas aleatórias. Este tipo de jogada significa que o usuário não tem nenhuma forma, a princípio, de prever com certeza o posicionamento das minas.

Desta forma, consideramos um jogo fácil, computando o tamanho do jogo e o número de bombas, um jogo com o maior número de espaços em branco possível. Considerando que espaços em branco são os espaços que não contem nem valores numéricos nem bombas. De modo contrário o jogo difícil é aquele que possui o menor número de espaços em branco. Tendo como base o nível fácil e difícil relativos a configuração do mapa gerada pelo sistema, tomamos como o nível intermediário o intervalo entre 1/3 à 2/3 da dificuldade máxima calculada. Por exemplo:

Em um jogo com a seguinte configuração; 10 linhas, 10 colunas e 10 bombas. O número máximo de espaços em branco (fácil) possui 80 espaços em branco. Portanto a avaliação probabilística é de 80%. No contrário um jogo que possui o mínimo de espaços em branco (difícil) possui 16 espaços em branco. Isto equivale a 16%. Representamos estes valores com números decimais e adequamos os mesmos a valores crescente entre 0 e 1 a fim de gerar um escala de dificuldade relativa. Então calculamos que o valor 0,2 representa o jogo mais fácil e 0,84 representa o jogo mais difícil de acordo com os dados do usuário. Assim o intervalo de avaliações de jogos é de 0,64. Dividimos este valor pela quantidade de níveis de dificuldade e atribuímos ao menor nível obtendo a escala desejada.

Para este caso apresentado obtemos o seguintes valores:

Nível	Intervalo
Fácil	0,2000 .. 0,4133
Intermediário	0,4133 .. 0,6266
Difícil	0,6266 .. 0,8400

A implementação com bases na avaliação obtida do jogo inicial, modifica a disposição de minas de forma a alterar a avaliação obtida até que a mesma possua um valor da avaliação dentro do intervalo referente ao nível escolhido pelo usuário.

Descrição detalhada da solução

Implementação realizada

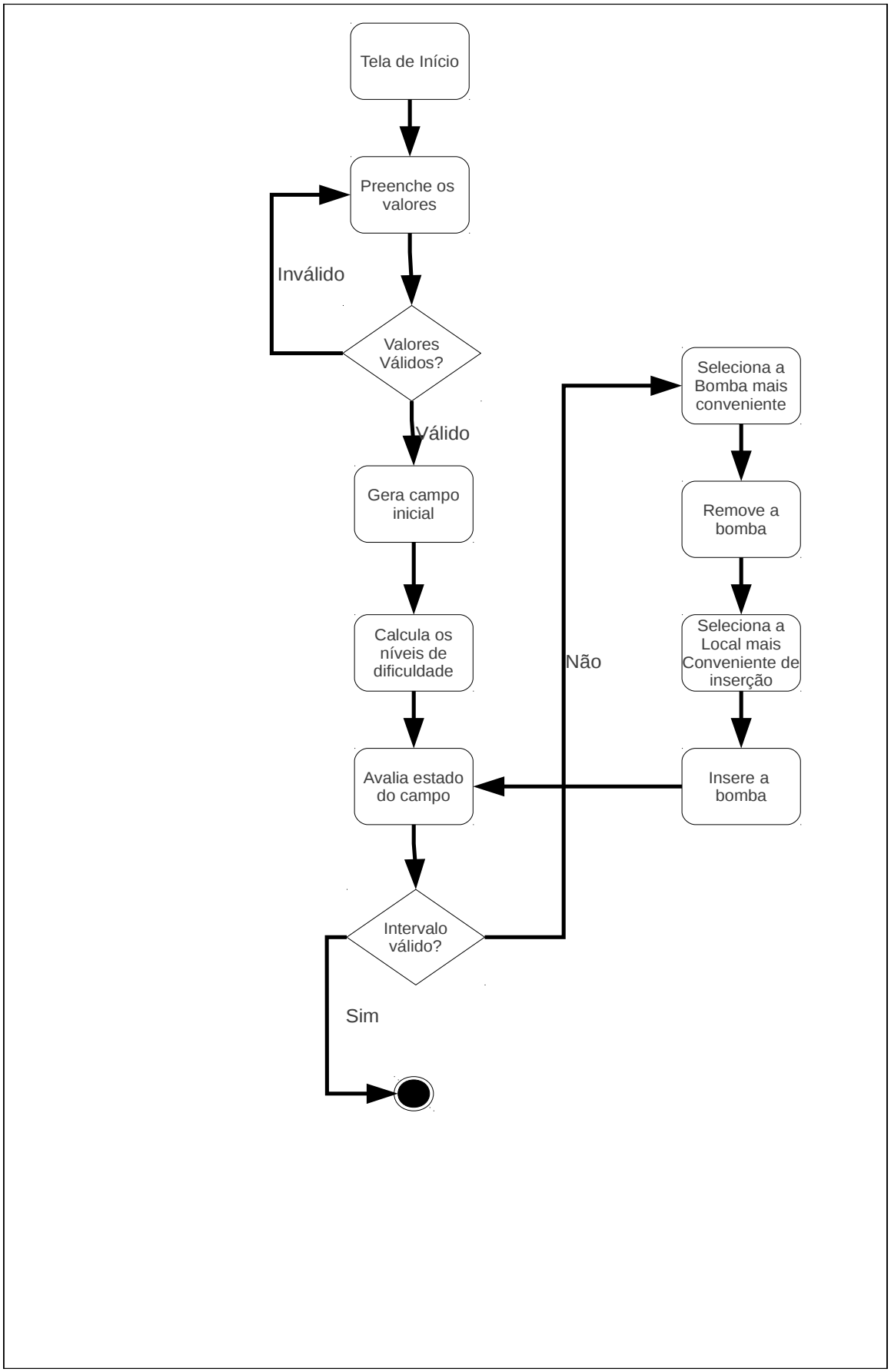
A linguagem escolhida para a implementação foi Java SE. Utilizando-se o padrão de desenvolvimento MVC construímos as seguintes classes:

- Main – Inicializa as demais classes, gera a interface gráfica inicial e executa os métodos da classe Field.
- Field – Contem todos os métodos de inicialização dos dados do campo e manipulação das minas e de seu posicionamento.
- MatrixPosition – Cria uma matriz de posições.
- MatrixUtil – Dado uma posição na matriz de bombas ela possibilita a navegação entre as posições adjacentes e posição inicial da matriz.
- NearPositionEnum – Define as direções próximas a serem navegadas na matriz.
- AlertWindow – Exibe uma mensagem de erro caso o usuário insira valores inválidos para a geração do campo.
- Button – Define o botão que representa graficamente cada uma das posições do campo minado além de definir seu conteúdo.
- PopUpWindow – Inicializa a janela que exibe as configurações necessárias para a inserção dos dados do usuário.
- ReportWindow – Janela secundária que exibe um relatório com os valores apresentados pelas iterações do sistema.
- Window – Janela principal onde é exibido o campo final.

A seguir descrevemos basicamente a classe Main onde os principais métodos são executados. Ele sumariza o algoritmo do implementação.

```
public class Main {

	static Window view;

	public static int mapper[][];
	public static int blankMapper[][];

	public static void main(String[] args) {
		/* Inicializa a GUI */
		view = new Window();
	}

	public static void generateField() {

		/* Inicializa o campo */
		mapper = new int[Field.lines][Field.columns];
		blankMapper = new int[Field.lines][Field.columns];
		Field.bombPosition = new ArrayList<MatrixPosition>();

		/* Configura o campo em branco */
		Field.setUpMap(mapper);
		Field.setUpBlankMap(blankMapper);

		/* Gera as bombas aleatórias */
		Field.generateBombs(mapper);

		/* Avalia o fácil e difícil */
		Field.evaluateGame();

		/* Avalia o mapa atua */
		Field.evaluateMap(mapper);

		Field.populateBlankMap(mapper, blankMapper);

		/* Realiza a busca local */
		Field.localSearch(mapper, blankMapper);

		/* Exibe a janela com o jogo final */
		Field.printGame(mapper);
	}

}
```

O fluxograma a seguir descreve funcionamento da implementação.



Telas da implementação

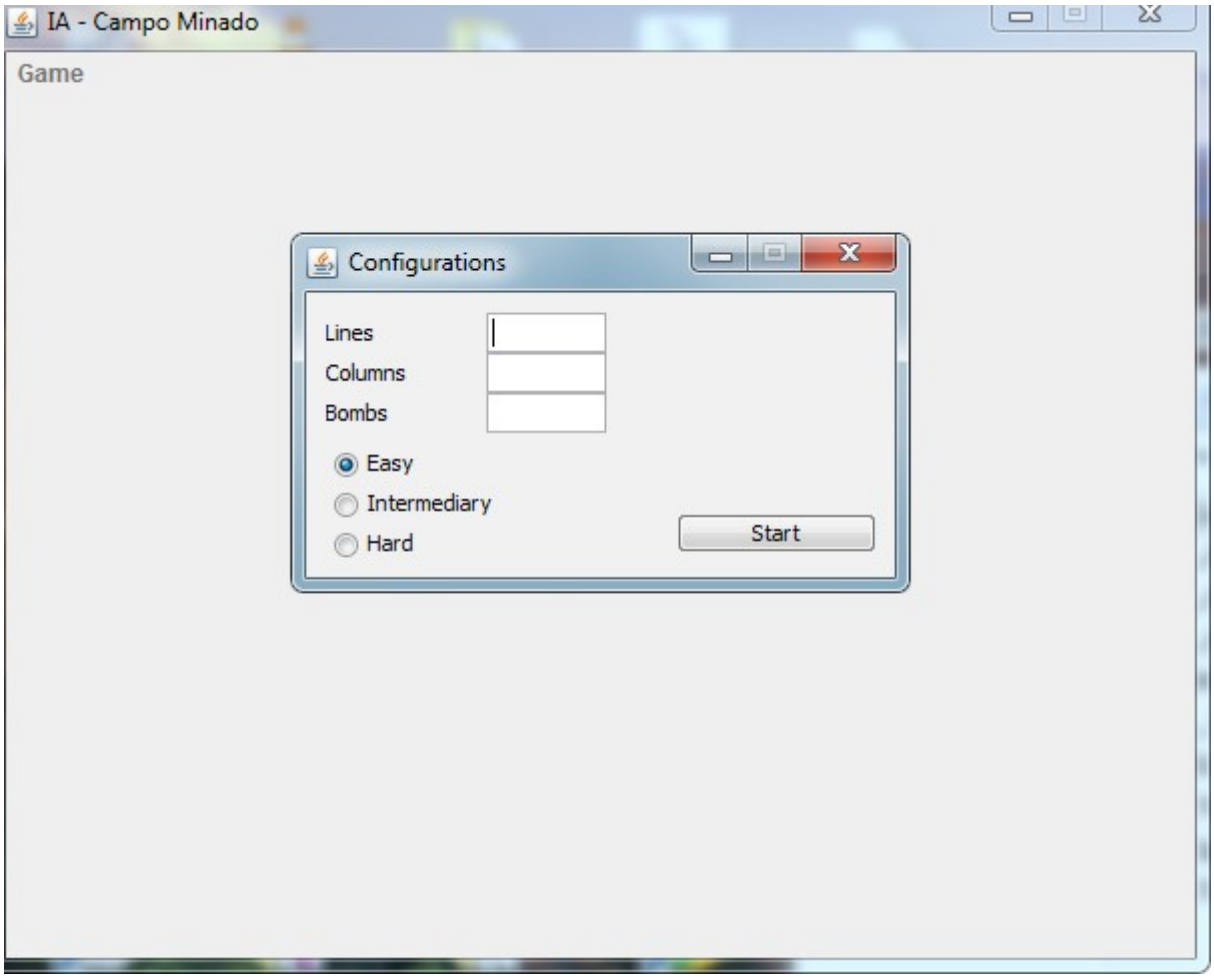


Figura 1 Tela inicial do sistema

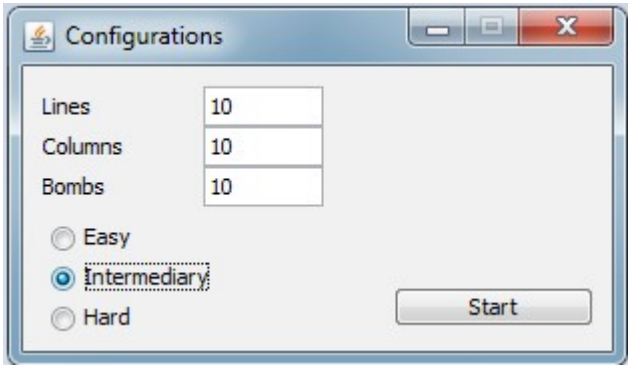


Figura 2 Tela configuração do campo

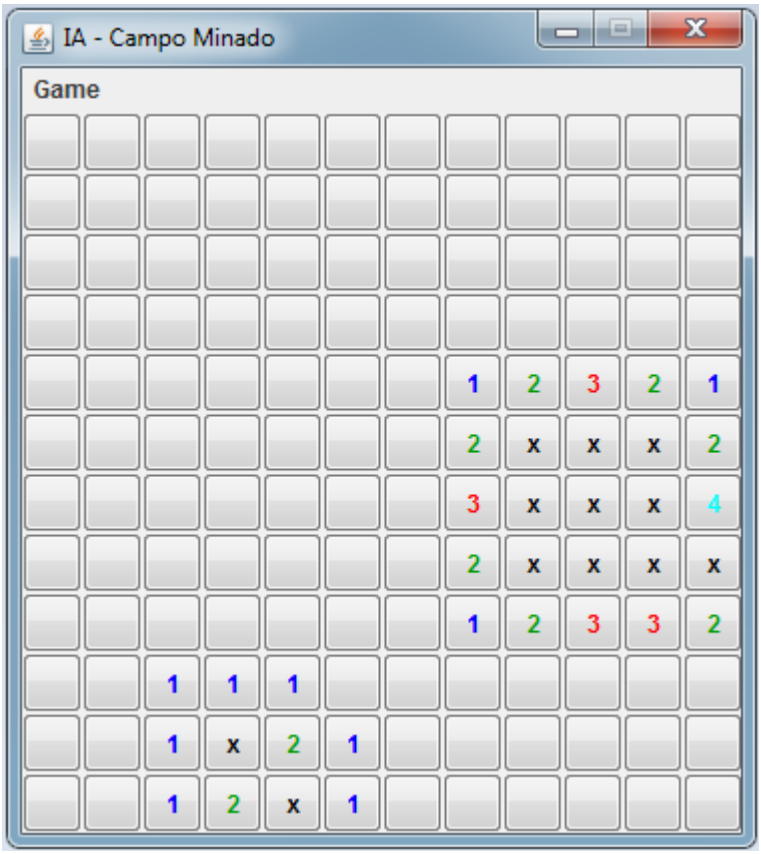


Figura 3 Representação do campo minado

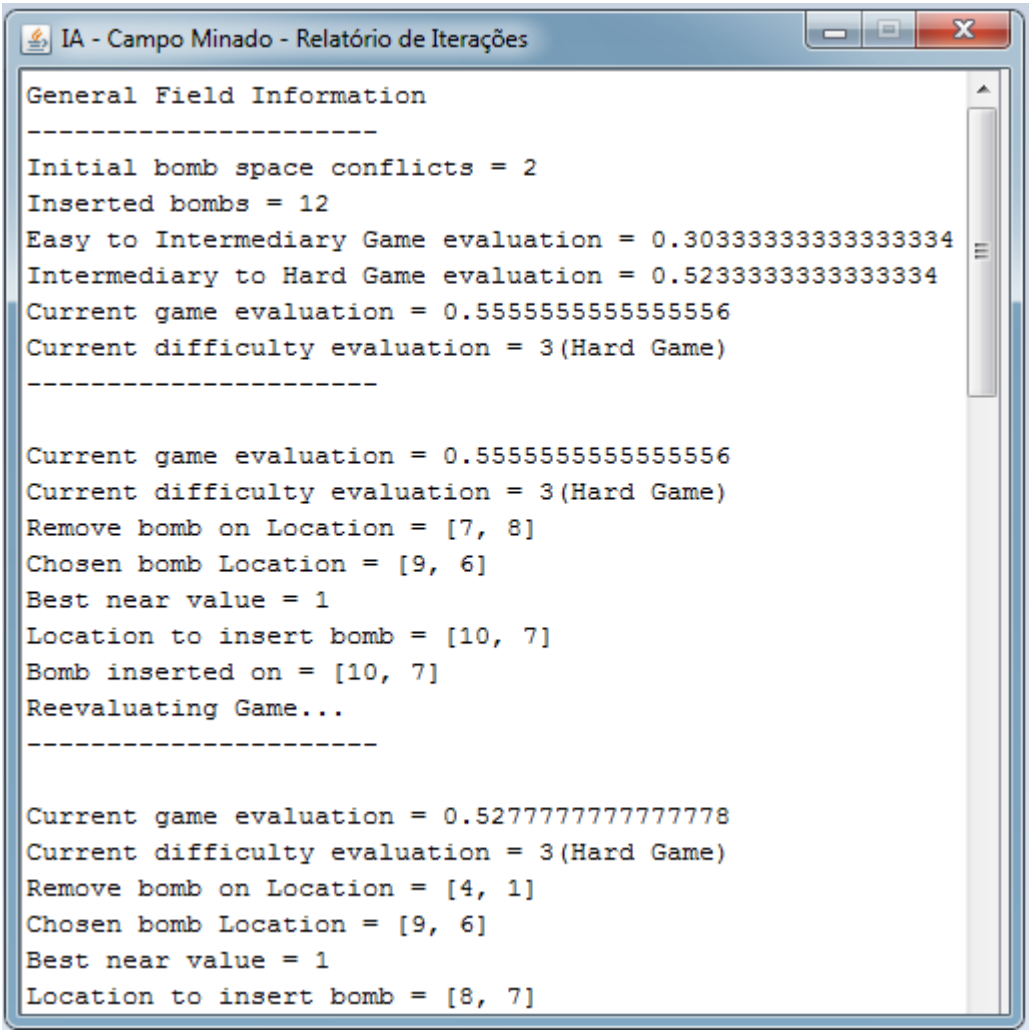


Figura 4 Relatório de iterações

Análise dos resultados e conclusões

Foi observado que é possível que a geração aleatória inicial do jogo pode ser capaz de gerar uma distribuição de minas que satisfaça ao intervalo calculado de acordo com o nível de dificuldade selecionado. Outro fator que determina a dificuldade do jogo é a proporção de minas por área. Esta proporção pode intrinsecamente definir a dificuldade global do jogo.

Concluímos com base nos resultados que de fato a busca local atinge o objetivo de gerar um campo com a distribuição de minas de acordo com o desejado em um espaço de tempo e recursos computacionais aceitáveis.