Instituto Federal de Educação,
Ciência e Tecnologia do Maranhão
Campus Monte Castelo - São Luís

Diretoria de Ensino Superior
Depto. Acad. de Informática (DAI)
Bach. em Sistemas de Informação

Prof. Josenildo Silva
jcsilva@ifma.edu.br

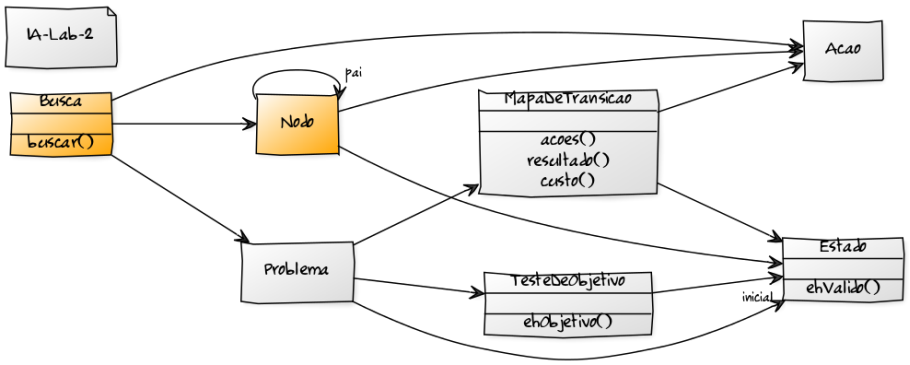
1º Semestre de 2015
Inteligência Artificial (SI 214)

Notas de Aula (prática) 1 – I.A.

Busca Cega

Biblioteca e Exemplo de Implementação de Missionários e Canibais

1 Biblioteca IA-labs-lib



1.1 Interfaces

```
public interface Estado {
    public boolean estadoValido();
    public boolean igual(Estado e);
}

public interface Acao {
}

public interface MapaDeTransicao {
    public Set<Acao> acoes(Estado e);
    public Estado resultado(Acao a, Estado e);
    public double custo(Acao a, Estado de, Estado para);
}

public interface TesteDeObjetivo {
    public boolean ehObjetivo(Estado e);
}

public interface Problema {
    public Estado resultado(Acao a, Estado e);
    public Set<Acao> acoes(Estado e);
    public boolean testaObjetivo(Estado e);
    public double custo(Acao a, Estado ei, Estado ej);
    public Estado estadoInicial();
```

```
}

public interface Busca {
    public List<Acao> buscar(Problema p);
}
```

1.2 Estruturas

```
import interfaces.Acao;
import interfaces.Estado;

public class Nodo {
    private final Nodo pai; // o antecessor deste nodo
    private final Estado estado; // o estado deste nodo
    private final Acao acao; // acao usada para gerar o estado deste nodo
    private final double custo; // custo do pai ate este nodo com a acao
    private final int profund; // profundidade do nodo na estrutura se for arvore

    public Nodo(Nodo pai, Estado estado, Acao acao, double custo, int profund) {
        this.pai = pai;
        this.estado = estado;
        this.acao = acao;
        this.custo = custo;
        this.profund = profund;
    }

    // getters e setters ...

}

public class FIFOQueue<E> extends LinkedList<E> implements Queue<E> {}

public class LIFOQueue<E> extends LinkedList<E> implements Queue<E> {}
```

1.3 Busca

```
public class BuscaEmLargura implements Busca {

    private final FIFOQueue<Nodo> fronteira; // <<<---- FILA
    private final List<Estado> estadosVisitados;

    public BuscaEmLargura() {
        this.fronteira = new FIFOQueue<>();
        this.estadosVisitados = new ArrayList();
    }

    @Override
    public List<Acao> buscar(Problema p) {

        Set<Acao> acoes = null;
        Nodo raiz = new Nodo(null, p.estadoInicial(), null, 0.0, 0);
        fronteira.add(raiz); // <<<--- nao importa se pilha ou fila
        while (!fronteira.isEmpty()) {
            Nodo aExpandir = fronteira.remove();
            this.getEstadosVisitados().add(aExpandir.getEstado());
            if (p.testaObjetivo(aExpandir.getEstado())) {
                  return this.solucao(aExpandir);
            }
            acoes = p.acoes(aExpandir.getEstado());
            if (acoes == null) {
                return null;
```

```
            }
            for (Acao a : acoes) {
                Estado sucessor = p.resultado(a, aExpandir.getEstado());
                boolean sucessorRepetido = false;
                for (Estado e : getEstadosVisitados()) {
                    if (e.igual(sucessor)) {
                        sucessorRepetido = true;
                        break;
                    };
                }
                if (!sucessorRepetido) {
                    getEstadosVisitados().add(sucessor);
                    Nodo nodoFilho = new Nodo(aExpandir, sucessor, a, 1, 1);
                    fronteira.add(nodoFilho);
                    if (p.testaObjetivo(sucessor)) {
                    }
                }
            }
        }
        return null;
    }

    private List<Acao> solucao(Nodo n) {
        List<Acao> s = null;
        if (n != null) {
            s = new ArrayList();
        }
        while (n != null) {
            if (n.getAcao() != null) {
                s.add(0, n.getAcao());
            }
            n = n.getPai();
        }
        return s;
    }

    public List<Estado> getEstadosVisitados() {
        return estadosVisitados;
    }
}

public class BuscaEmProfundidade implements Busca {

    private final LIFOQueue<Nodo> fronteira; // <<<---- PILHA
    private final List<Estado> estadosVisitados;

    public BuscaEmProfundidade() {
        this.fronteira = new LIFOQueue<>();
        this.estadosVisitados = new ArrayList();
    }

    // resto do codigo igual ao busca em largura...
}
```

2 Parte Dependente do Domínio

2.1 Implementando Missionários e Canibais

```
public class EstadoMissCanib implements Estado{
    private int md=0; // quantidade de missionarios na margem direita
    private int me=0; // quantidade de missionarios na margem esquerda
```

```
    private int cd=0; // quantidade de missionarios na margem direita
    private int ce=0; // quantidade de missionarios na margem esquerda
    private char b='E';  // localizacao do barco (E esquerda, D direita)

    public EstadoMissCanib(int me, int ce, char b, int md, int cd){
        this.me = me;
        this.md = md;
        this.ce = ce;
        this.cd = cd;
        this.b = b;
    }

    @Override
    public boolean estadoValido() {
        // a soma total de missionarios ou canibais eh 3 e o barco esta em uma
        // localizacao correta
        boolean positivos = (this.md>=0 && this.me>=0 && this.ce>=0 && this.cd>=0);
        boolean total3 = ((this.md+this.me==3)&& (this.cd+this.ce==3)
                );
        boolean barcoCorreto = ((this.b=='D')||(this.b=='E'));
        // missionarios sao mais numerosos que canibais em ambos os lados
        boolean semCanibalismo =
                ((this.md==0) || (this.md>=this.cd )) &&
                ((this.me==0) || (this.me >=this.ce));

        return positivos && total3 && barcoCorreto && semCanibalismo;
    }

    @Override
    public boolean igual(Estado e) {

        return (
                (((EstadoMissCanib)e).getMd()==this.md) &&
                (((EstadoMissCanib)e).getMe()==this.me) &&
                (((EstadoMissCanib)e).getCe()==this.ce) &&
                (((EstadoMissCanib)e).getCd()==this.cd) &&
                (((EstadoMissCanib)e).getB()==this.b));
    }

    // getters e setters ...
    @Override
    public String toString(){ ... }
}

public class AcaoMissCanib implements Acao{
    private String id = "NoOp";

    public AcaoMissCanib(String id){ this.id=id; }
    public String getId() { return id; }
    @Override
    public String toString(){ return id;  }
}

public class MapaDeTransicaoMissCanib implements MapaDeTransicao {
    public static Acao M = new AcaoMissCanib("Mover␣1␣missionario");
    public static Acao MM = new AcaoMissCanib("Mover␣2␣missionarios");
    public static Acao C = new AcaoMissCanib("Mover␣1␣canibal");
    public static Acao CC = new AcaoMissCanib("Mover␣2␣canibais");
    public static Acao MC = new AcaoMissCanib("Mover␣1␣missionario␣e␣1␣canibal");

    @Override
```

```
public Set<Acao> acoes(Estado e) {
    Set<Acao> acoes = new HashSet();
    if (resultado(M, e).estadoValido()) {
        acoes.add(M);
    }
    if (resultado(MM, e).estadoValido()) {
        acoes.add(MM);
    }
    if (resultado(C, e).estadoValido()) {
        acoes.add(C);
    }
    if (resultado(CC, e).estadoValido()) {
        acoes.add(CC);
    }
    if (resultado(MC, e).estadoValido()) {
        acoes.add(MC);
    }
    return acoes;
}

public Estado resultado(Acao a, Estado e) {
    char barco = ((EstadoMissCanib) e).getB();
    int md = ((EstadoMissCanib) e).getMd(),
        me = ((EstadoMissCanib) e).getMe(),
        cd = ((EstadoMissCanib) e).getCd(),
        ce = ((EstadoMissCanib) e).getCe();
    if (barco == 'E') {
        barco = 'D';
        // esquerda para direita
        if (((AcaoMissCanib) a) == M) {
            me--;
            md++;
        } else if (((AcaoMissCanib) a) == MM) {
            me -= 2;
            md += 2;
        } else if (((AcaoMissCanib) a) == C) {
            ce--;
            cd++;
        } else if (((AcaoMissCanib) a) == CC) {
            ce -= 2;
            cd += 2;
        } else if (((AcaoMissCanib) a) == MC) {
            me--;
            md++;
            ce--;
            cd++;
        }

    } else {
        barco = 'E';
        // direita para esquerda
        // repete tudo, invertendo os sinais
        // ...
    }
    Estado novoEstado = new EstadoMissCanib(me, ce, barco, md, cd);
    return novoEstado;
}

@Override
public double custo(Acao a, Estado de, Estado para) {
    return 1.0;
```

```
    }
}

public class TesteDeObjetivoMissCanib implements TesteDeObjetivo {

    private EstadoMissCanib objetivo;
    public TesteDeObjetivoMissCanib() {
        this.objetivo = new EstadoMissCanib (0,0,'D',3,3);
    }

    public TesteDeObjetivoMissCanib(EstadoMissCanib inicial) {
        char bi = inicial.getB();
        char barco = '0';
        int me = 0, ce = 0, md = 0, cd = 0;

        // Caso o estado informado seja invalido, usa o estado default
        this.objetivo = new EstadoMissCanib(0,0,'D',3,3);

        if (inicial.estadoValido()) {
            // inverte os valores de esquerda para direita
            md = inicial.getMe();
            cd = inicial.getCe();
            me = inicial.getMd();
            ce = inicial.getCd();
            if ('E' == bi) {
                barco = 'D';
            } else if ('D' == bi) {
                barco = 'E';
            }
            this.objetivo = new EstadoMissCanib(me, ce, barco, md, cd);
        }
    }
    @Override
    public boolean ehObjetivo(Estado e) {
        return ((EstadoMissCanib) e).igual(this.objetivo);
    }

}

public class ProblemaMissCanib implements Problema {
    private final MapaDeTransicaoMissCanib mapa;
    private final EstadoMissCanib inicio;
    private final TesteDeObjetivoMissCanib testObjetivo;

    public ProblemaMissCanib() {
        this.mapa = new MapaDeTransicaoMissCanib();
        this.inicio = new EstadoMissCanib(3, 3, 'E', 0, 0);
        this.testObjetivo = new TesteDeObjetivoMissCanib(this.inicio);
    }

    public ProblemaMissCanib(EstadoMissCanib ini) {
        ...
        if (ini.estadoValido()){
         this.inicio=ini;
        }else{
            this.inicio=new EstadoMissCanib(3,3,'E',0,0);
        }
    }

    public ProblemaMissCanib(EstadoMissCanib ini, boolean objetivoDefault) {
        ...
```

```
        }
        if (objetivoDefault){//ir ate o objetivo defalt
            this.testObjetivo = new TesteDeObjetivoMissCanib();
        }else{// ir ate o inverso do estado inicial
            this.testObjetivo = new TesteDeObjetivoMissCanib(ini);
        }
    }
    @Override
    public Estado resultado(Acao a, Estado e) {
        return this.mapa.resultado(a, e);
    }
    @Override
    public Set<Acao> acoes(Estado e) {
        return this.mapa.acoes(e);
    }
    @Override
    public boolean testaObjetivo(Estado e) {
        return this.testObjetivo.ehObjetivo(e);
    }
    @Override
    public double custo(Acao a, Estado ei, Estado ej) {
        return this.mapa.custo(a, ej, ej);
    }
    @Override
    public Estado estadoInicial() {
        return this.inicio;
    }
}
```

2.2 Teste

```
// Em uma classe de teste ...
public void testBuscar() {
        Problema p = new ProblemaMissCanib();
        BuscaEmLargura busca = new BuscaEmLargura() ;
        List<Acao> result = new ArrayList();
        List<Acao> result = busca.buscar(p);

        for (int i=0;i<result.size();i++){
            System.out.println("[Passo␣"+i+"]"+result.get(i).toString());
        }

        // tambem pode-se testar com outros estados iniciais
        p = new ProblemaMissCanib(new EstadoMissCanib(3,2,'E',0,1),true);
}
```