Sistema de Partilha de Bicicletas - SuperBike

André Almeida Gonçalves A75625, Rogério Gomes Lopes Moreira A74634,
Tiago Filipe Oliveira Sá A71835,

Agentes Inteligentes, Perfil de Sistemas Inteligentes, Universidade do Minho

Resumo O presente trabalho tem como objetivo o desenvolvimento de um Sistema de Partilha de Bicicletas que permita ao utilizador alugar bicicletas para realizar viagens curtas. Pretende-se assim desenvolver um sistema multi-agente utilizando o ambiente de desenvolvimento JADE, complementando com JADEX e JESS.

1 Introdução

O projeto proposto no âmbito da unidade curricular de Agentes Inteligentes tem como objetivo desenvolver um sistema no ambiente multiagente JADE, complementando com a utilização de JADEX e JESS. O projeto consiste no desenvolvimento de um Sistema de Partilha de Bicicletas, que permita aos utilizadores alugar bicicletas numa dada estação, realizar uma viagem e entregar a bicicleta numa outra estação. No entanto, o grande problema em jogo é a capacidade das estações. Como as estações de recolha/entrega têm uma capacidade limitada torna-se importante fazer a gestão das bicicletas na cidade, não deixando uma estação ficar sem bicicletas e outra com excesso, impedindo a sua devolução. A isto se chama o Problema de Reequilíbrio de Partilha de Bicicletas (PRPB). Uma das formas de resolução deste problema é o incentivo a utilizadores, para que estes entreguem as bicicletas em estações menos lotadas. A devolução ou não das bicicletas por parte dos utilizadores é calculada através de diversos fatores que influenciam a escolha.

2 Descrição do sistema

O projeto estruturado segue a seguinte estrutura. O agente JADE "Mother" cria agentes utilizadores com informações, relevantes para o ambiente onde estão inseridos, definidas pelo grupo. Estas informações variáveis são o estado de espírito, a idade, o sexo, as doenças e a condição física. Para além disso, os utilizadores têm ainda uma posição inicial e uma posição de destino.
Este utilizador procura as estações disponíveis na sua proximidade e escolhendo assim a estação de onde irá partir com base nos fatores definidos na sua criação e no meio exterior, isto é, assim que as informações estiverem reunidas o utilizador comunica com o agente "Decider" que com base no perfil do utilizador e nos casos anteriores irá decidir qual a melhor estação de destino. Por exemplo se for uma pessoa idosa, talvez irá ignorar os descontos oferecidos por estações ligeiramente mais longe devido à dificuldade em se movimentar, enquanto um jovem estudante no topo da sua condição física, não se importará de se deslocar mais algum tempo e assim aproveitar o desconto oferecido por uma estação mais distante, de forma a regular o número de bicicletas em todas as estações.
Desta forma pretendemos retratar da forma mais fidedigna um sistema real onde os utilizadores podem ou não aceitar as ofertas que o sistema lhes propõem, baseando-se em diversos fatores externos ao sistema e não apenas nos descontos oferecidos. Achamos também boa opção ter uma base de casos anteriores para que o sistema tenha cada vez mais eficácia nas decisões que define para o utilizador, e assim, abranger a parte da inteligência do sistema, tornando-se este cada vez mais "inteligente"no tratamento do problema em questão. Para além disso, como dito anteriormente, tivemos em conta também o meio exterior e não só as características do utilizador, para isto criamos um utilizador Meteorologia que a cada 5000ms muda a meteorologia de foram aleatória. Um utilizador ao fazer a escolha da estação a entregar tem também a meteorologia atual.

3 Tipos de Agentes

O sistema terá alguns tipos de agentes diferentes, permitindo uma abordagem de reequilíbrio:

Agente Estação - representam as estações do Sistema de Partilha de Bicicletas e esperam pelos pedidos dos Utilizadores. Sempre que um utilizador entra ou sai da área de influencia de uma estação, a estação atualiza o seu conhecimento e associa os Utilizadores que estão dentro da sua área de proximidade. Para isto é necessário que cada agente estação tenha um número de bicicletas disponíveis que pode variar de estação para estação, uma área de influência, uma posição (x,y) e um custo base. O custo base é o custo que o utilizador terá por alugar uma bicicleta em determinada estação. De notar também que estações com maior capacidade, terão uma maior área de influência.

Agente Utilizador - Com base na posição inicial e de destino do utilizador, o agente determina as estações de SPB, baseando-se na APE de cada estação. Quando a distância da viagem percorrida ultrapassar da totalidade do trajeto, são enviadas solicitações de entrega da bicicleta, de acordo com as estações próximas. O utilizador poderá aceitar o rejeitar o pedido, de acordo com os incentivos definidos e que variam consoante fatores como a meteorologia, a idade do utilizador, o sexo, a condição física, as doenças e o tempo de percurso. Os utilizadores registam-se no Directory Facilitator ao serem inicializados, para que as estações possam comunicar com eles. O Agente Utilizador tem assim que ter disponíveis os seguintes dados: estado de espírito, idade, sexo, doenças, condição física, posição inicial, posição destino e o tempo do percurso. O agente aceita a oferta de uma estação mediante os parâmetros, como por exemplo: se tiver doenças é provável que não aceite ofertas de estações a mais de 5km do destino, se a condição física for má não aceita ofertas de estações a mais de 1km do destino.

Agente Interface - agente com o qual o utilizador interage e que tem uma lista das estações de entrega disponíveis. Será neste agente que a interface gráfica será feita, utilizando JFreeChart.

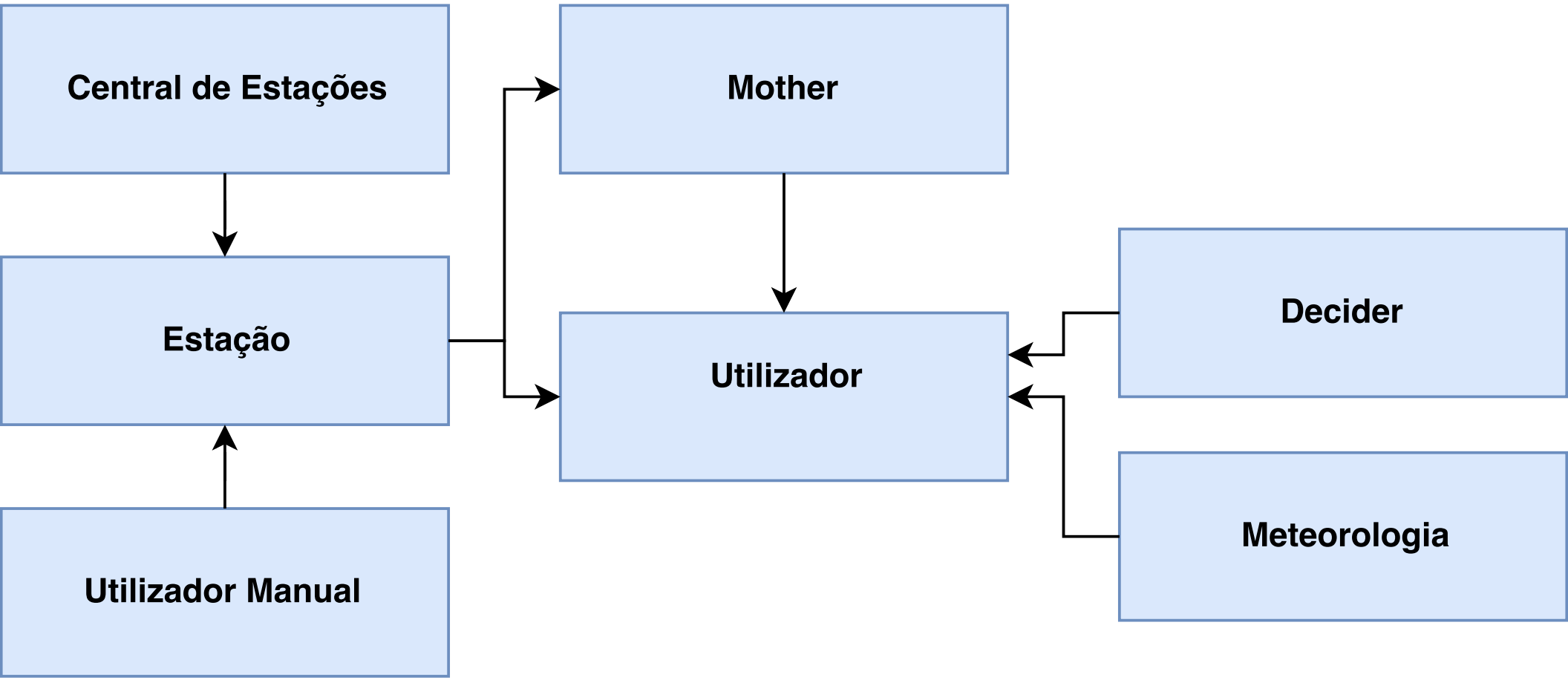
Agente Decider - agente que guarda os dados dos agentes passados semelhante a CBR [1], ou seja, se um utilizador numa determinada altura, com determinados parâmetros decidiu de uma maneira então é mais provável que outro utilizador com os mesmos parâmetros decida igualmente.

Agente Manual - agente que permite a interação manual com o sistema.

Agente Meteorologia - agente responsável pela meteorologia. Como a meteorologia é igual para todos os utilizadores (dentro de uma cidade a variação da meteorologia varia muito pouco), o grupo chegou à conclusão que seria vantajoso ter um agente deste tipo para calcular a meteorologia e que esta fosse igual para todos os utilizadores a usufruir do sistema em determinada altura.

Mother - agente que cria os utilizadores com os parâmetros de forma "random", para que a diversidade de utilizadores no sistema seja grande e assim se possa ver a afetação de diferentes parâmetros nas decisões dos utilizadores.

[1] CBR - Case-Based Reasoning


Central de Estações
Mother
Estação
Utilizador
Decider
Utilizador Manual
Meteorologia

4 Decisões Arquiteturais

Durante o decorrer do desenvolvimento do projeto foi necessário tomar diferentes decisões arquiteturais, tendo em conta não só o progresso do trabalho como também a fluidez do sistema.

No planeamento da arquitetura levada a cabo pelo grupo foi definido o uso das diferentes plataformas.

JADE
- Agente Mother
- Agente Estação
- Agente Manual
- Agente Utilizador
- Meteorologia
- Agente Decider

JADEX
- Agente Utilizador

JESS
- Agente Decider

Achamos também útil definir já os custos consoante as áreas que os utilizadores percorrem mediante o seu percurso previamente calculado. Para isto definimos três patamares de preço:

Estação com baixa afluência ($\leq 25\%$) - 50% do custo base
Estação com média afluência (26% - 74%) - 100% do custo base
Estação com alta afluência ($\geq 75\%$) - 150% do custo base

Os utilizadores têm definido uma gama de parâmetros que, ao serem criados, assumem. Estes parâmetros são os seguintes:

Estado de Espírito
- 1 - Depressivo
- 2 - Triste
- 3 - Neutro
- 4 - Contente
- 5 - Eufórico

Idade : Entre 12 anos e 80 anos.

Sexo
- 1 - Masculino
- 2 - Feminino

Doenças

- 1 - Saudável
- 2 - Grande doença
- 3 - Pequena doença

Condição Física

- 1 - Pessoa Passiva
- 2 - Pessoa Pouco Ativa
- 3 - Pessoa Ativa
- 4 - Atleta
- 5 - Atleta Profissional

Assim e baseando-se nestes dados, o algoritmo que toma a decisão se um determinado utilizador deve ou não aceitar uma oferta de uma estação é baseado numa árvore em que os níveis da árvore são os parâmetros mais relevantes para o cálculo do próximo caso. Existe um ficheiro com dados de cálculos anteriores que é usado para calcular uma percentagem de utilizadores que aceitaram ou rejeitaram determinada oferta consoante os parâmetros, esse valor é usado como "peso"e multiplicado pelos valores gerados pela "Mother"aquando da criança. Depois disso, se a estação que estiver mais perto for a mais barata vai diretamente para essa, se a mais barata for diferente da que está mais perto, faz o cálculo consoante os pesos, se esse resultado for menor que 10, vai para a mais perto, se for maior ou igual vai para a mais barata.

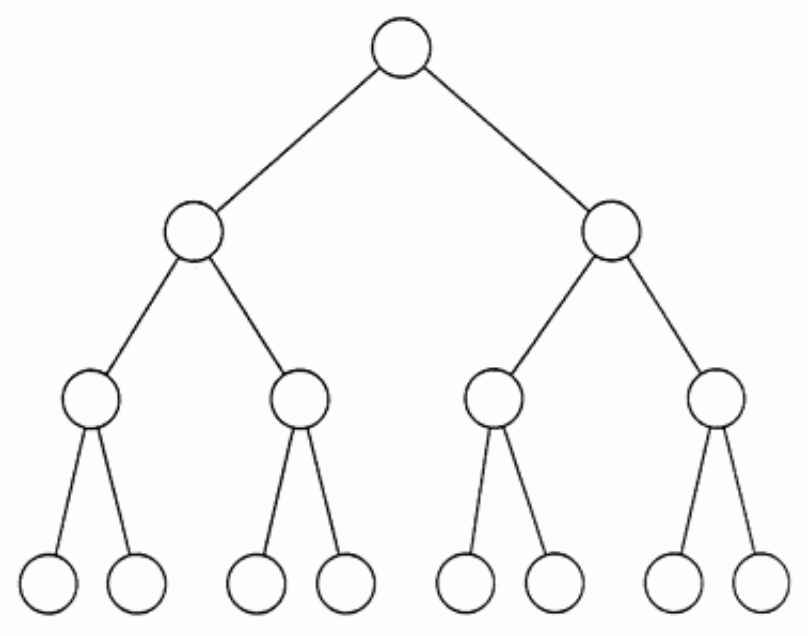


Figura 1. Árvore de Decisão

5 JESS e JADE

O grupo começou por optar por definir o Decider em JESS mas uma das principais dificuldades que surgiu foi a comunicação entre o JESS e o JADE para a tomada de decisões. Posto isto, concluímos que a melhor opção a seguir para o projeto era desenvolver em JADE.

Código JADE:

```
package decider;
import java.util.ArrayList;

import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jess.JessException;
import jess.Rete;
import stacion.Stacion;

@SuppressWarnings("serial")
public class JessInfo extends CyclicBehaviour {
        private Rete engine;
        Node nodos;
        private ArrayList<Stacion> stacions = new ArrayList<Stacion>();
        private String weather;

    public JessInfo(Agent a,String filename){
        engine = new Rete();
        Writer w = new Writer();
        nodos = w.populate();

        try {
                engine.batch(filename);
            engine.reset();
        } catch (JessException ex) {
            ex.printStackTrace();
        }
    }


    public void updateStacions(String go,Float x, Float y) {
        ACLMessage inf = new ACLMessage( ACLMessage.INFORM );
        inf.setContent("Pos: " + x + " " + y + " " + go );
        inf.addReceiver(new AID("StacionHead", AID.ISLOCALNAME));
        myAgent.send(inf);
```

```
    try {
            ACLMessage msg =myAgent.receive();
            String[] splited = msg.getContent().split("\n");
            int i;
            for(i=0;i!=splited.length;i++) {
                    stacions.add(new Stacion(splited[i]));
            }
    }catch (Exception e) {}
}

public void getWeather() {

            ACLMessage inf = new ACLMessage(ACLMessage.INFORM);
                    inf.addReceiver(new AID("Meteo", AID.ISLOCALNAME));
                    inf.setContent("M");
                    myAgent.send(inf);

                    ACLMessage msg =myAgent.receive();
            if(msg!=null) {
                    if(msg.getContent().charAt(0)=='W')
                            weather = msg.getContent();
            }
}


    @Override
    public void action() {
   ACLMessage msg = myAgent.receive();
   if(msg != null){
       if(msg.getContent().length() > 0) {
               String[] res = msg.getContent().split("\\s+");
               updateStacions("Go",Float.parseFloat(res[7]),Float.parseFloat
               getWeather();
               try {
                       ACLMsg2Jess(msg);
                       engine.run();
                       System.out.println(engine.fetch(res[0]));
                } catch (JessException ex) {
                    ex.printStackTrace();
                }
       }
   }
}
```

```
    /*
     * Asserts a Fact that represents the msg in Jess
    */
    public boolean addFact(String fact){
         try {
             engine.executeCommand(fact);
         } catch (JessException ex) {
             return false;
         }
        return true;
    }

         /*
      * Convert a ACLMessage to a JESS Fact
      */
      public boolean ACLMsg2Jess(ACLMessage msg){
          StringBuilder sb = new StringBuilder();
          sb.append("(assert (ACLMessage ");
          sb.append("(communicative-act ").append(msg.getPerformative()).append("

          if(msg.getSender() != null){
                 sb.append(" (sender ").append(msg.getSender().getName()).append(
          }

          if(msg.getContent()!=null){
                 sb.append("  (content ").append(msg.getContent()).append(")");
          }

          sb.append("))");
          String jmsg = sb.toString();
          return addFact(jmsg);
      }
}
```

Código JESS:

```
(deftemplate ACLMessage
        (slot communicative-act)
        (slot sender)
        (multislot receiver)
        (slot reply-with)
        (slot in-reply-to)
        (slot envelope)
        (slot conversation-id)
        (slot protocol)
        (slot language)
        (slot ontology)
```

```
        (multislot content)
        (slot encoding)
        (multislot reply-to)
        (slot reply-by)
)

(deftemplate Case
        (slot spirit)
                (slot age)
                (slot healt)
                (slot condicion)
                (multislot position)
                (multislot destiny)
                (slot weather)
)

(deftemplate Station
        (slot name)
                (multislot position)
                (slot price)
)

(defrule receivedUser
    (ACLMessage (sender ?se)
                (content ?u ?s ?a ?sex ?h ?c ?x ?y ?xd ?yd))
    =>
    (store "word" ?s)
        (printout t "message from " ?se "with temp " ?s crlf)
)
```

6 Padrões de Comunicação

Os agentes comunica entre si para estabelecer as ligações do sistema. Os padrões de comunicação definidos para cada agente pelo grupo são os seguintes:

Estação :

In : Recebe o comando "listar";

Out : Envia todas as estações disponíveis no sistema depois de lidas de um ficheiro onde foram previamente definidas. O formato em que são definidas é "Nome Total de Bicicletas Bicicletas Disponíveis Área de Influência Custo Base ¡Posição X, Posição Y¿"

Utilizador :

Out : Pede ao "Decider" que baseado nos dados enviados no formato "Dados-Dados-Dados..." , num tempo previamente definido.

Mother : Cria utilizadores com parâmetros aleatórios de 1 em 1 segundo.

Meteorologia :

In : Recebe o pedido de informação da meteorologia

Out : Envia a meteorologia atual que poderá ser Chuva, Quente, Frio, Nevoeiro ou Trovoada.

Estação :

In : Recebe "+" para aumentar o número de bicicletas e consequentemente mexer na área e no custo base e recebe "-" para diminuir o número de bicicletas e também mexer na área e no custo base.

Decider :

In : Recebe "Dados-Dados..." e toma uma decisão baseada nisso.

Out : Decisão para o ficheiro.

7 Gestão de conflitos

O sistema desenhado tem a possibilidade de ocorrerem alguns conflitos. De seguida listamos alguns previamente identificados e as suas possíveis resoluções. É natural que no decorrer do desenvolvimento do projeto surjam mais.

- Quando uma estação tem uma taxa de ocupação alta, há uma necessidade de reequilibrar o sistema, por forma a tentar garantir que as bicicletas sejam entregues noutras estações.
  Resolução: Diminuir a área abrangente da estação e aumentar o preço para 150% do custo base nos casos em que a lotação é superior a 75%.
  Vantagens: Permite atenuar o PRPB[2].
  Desvantagens: Incorre numa possível perda de lucro da estação

- Quando uma estação tem uma taxa de ocupação baixa, há uma necessidade de reequilibrar o sistema, para tentar garantir que as bicicletas sejam entregues nesta estação, para que os utilizadores têm bicicletas disponíveis para alugar nessa estação.
  Resolução: Aumentar a área abrangente da estação, diminuir o preço para 50% do custo base, nos casos em que a lotação é inferior a 25%.
  Vantagens: Permite atenuar o PRPB.
  Desvantagens: Incorre numa possível perda de lucro da estação.

- Quando um agente utilizador, faz o percurso entre o seu ponto inicial e o seu destino após ultrapassar do trajeto, este pode ser solicitado por 2 ou mais estações para entregar a bicicleta.
  Resoução : Usado um algoritmo que tem em conta o custo e a distância da estação, as características e a meteorologia atual. Se depois de efetuado o cálculo, o resultado for maior do que um determinado número então o utilizador aceita a proposta da estação.
  Vantagens: Implementação que não opta pelos dois extremos (mais perto ou mais barata), permitindo transmitir uma interação mais realista entre cada utilizador e as estações, tentando criar decisões únicas para cada agente utilizador e não uma solução geral.
  Desvantagens: Aumenta a complexidade do sistema, já que é necessário a interação com outros agentes.

- No nosso sistema, implementamos um agente que criará novos utilizadores, atribuindo-lhes diversos atributos. É necessário por isso, determinar a melhor forma de criar estes agentes com atributos o mais diverso possível.
  Resolução: Atribuir valores aleatórios a cada atributo.
  Vantagens: Esta abordagem é de fácil implementação. Seria atribuída uma gama de valores para cada atributo, e o valor de cada um seria escolhido de forma aleatória, a partir desse intervalo.

[2] PRPB - Problema do Reequilíbrio de Partilha de Bicicletas

Desvantagens: Não existe qualquer relação entre os atributos, e pode-se vir a verificar incoerências, num contexto real, uma vez que alguns destes atributos terão dependências intrínsecas a outros. Por exemplo, a idade e a condição física.

8 Conclusão

O projeto desenvolvido pelo grupo é uma extensão ao que é proposto no enunciado, cumprindo todos os requisitos pedidos. Contudo, surgiram algumas dificuldades durante o desenvolvimento do projeto. No sistema a nossa intenção sempre foi estender aquilo que era pretendido e tornar o sistema verdadeiramente "inteligente". Não só inserindo outras variáveis que num sistema real são talvez mais importantes que apenas as compensações dadas por cada estação, como é descrito no enunciado do trabalho prático, como por exemplo, a idade, a condição física ou a meteorologia, mas também criando um sistema de raciocínio baseado em casos anteriores para os agentes não se limitarem apenas a uma escolha quase aleatória mas terem antecedentes que os ajudam a tomar a melhor decisão em determinado contexto, simulando assim o problema real.