UNIVERSIDADE DE SÃO PAULO

PCS 5703 – SISTEMAS MULTIAGENTES

Exercício Prático 3 - Estratégia

Grupo 1

Akira Aricê de Moura Galvão - 4963541

Luiza Pellin Biasoto - 6846703

Mariane Salomão dos Reis - 8004749

Estratégia

Nossa estratégia consiste em criar um índice que define a ordenação de lances dos agentes no leilão ativo. Após isso, seguindo os critérios restritivos de cada agente para cada leilão, cada agente realiza seu lance, conforme a ordem especificada.

O índice que define a ordenação de lances é chamado de Iniciativa. Esse índice representa o quão importante é para esse agente dar o lance primeiro que os outros e expressamos essa importância em um valor de 0 a 1, sendo que quanto mais próximo de zero, mais importante dar o lance primeiro e quanto mais próximo de 1, menos importante. Para fazer o cálculo desse índice, utilizamos as seguintes variáveis:

- **Soma dos Valores Mínimos (SVMin)**: essa variável consiste na soma de todos os lances mínimos que esse agente pode dar. Por exemplo, se um agente pode participar de 3 leilões e seu lance mínimo para cada um dos 3 leilões é 5, então essa variável recebe 15.
- **Soma dos Valores Máximos (SVMax):** essa variável consiste na soma dos valores de início dos leilões que este agente ainda pode participar. Por exemplo, no caso dos 3 leilões do agente citado acima, a soma dos valores de início solicitados pelo Giacomo é 50.
- **Soma Valores Máximos Encerrados (SVMaxE):** essa variável consiste na soma dos valores de início dos leilões que o agente pôde participar e já encerraram.
- **Soma Valores Ganhos (SVG):** soma dos lances do agente que foram vencedores
- **Leilões Totais (LT):** Total de leilões (sempre 10).
- **Leilões Agente (LA):** Total de leilões que o agente pode participar.

➔ **Força de Negociabilidade (FNI):**

A junção das variáveis **SVMin** com **SVMax** dá origem a um indicador de ***força de negociabilidade inversa (FNI)***, sendo este definido por:

$$FNI = SVMin/SVMax$$

Quanto maior o **FNI**, menor o poder do agente para negociar uma condição mais competitiva nos leilões (limitações de lances mínimos, por exemplo). Pode-se associar este poder ao fato de pequenas empresas (neste caso, pequenos empreiteiros) não possuírem estrutura para competir com outras grandes empresas,

que possuem produção em escala e conseguem garantir um produto por um menor preço. Dessa forma, uma estratégia para essas pequenas empresas é, logo no começo do leilão, arriscar mais e já dar um lance bem mais abaixo do máximo ou próximo do limite que aceitam receber pelo produto, i.e., quão mais alto o FNI, mais risco o agente vai tomar e mais rápido vai enviar um lance.

➔ **Fator de Aproveitamento (FA):**

A junção das variáveis **SVMaxE** com **SVG** dá origem a um indicador de ***fator de aproveitamento (FA)***, sendo este definido por:

$$FA = SVG/SVMaxE$$

Quanto maior o **FA**, maior o aproveitamento do agente nos leilões já encerrados. Este indicador ajuda a **reduzir o risco**, reduzindo a velocidade de envio de lances, do agente que conseguiu ganhar em leilões anteriores.

➔ **Fator de Participação (FP):**

A junção das variáveis **LA** com **LT** dá origem a um indicador de ***fator de participação (FP)***, sendo este definido por:

$$FP = LA/LT$$

Quanto maior o **FP**, maior a chance de um agente ganhar em algum leilão, dada a sua maior participação nestes. Assim, quanto maior o FP de um agente, menos risco este tomará e enviará o seu lance mais tarde.

➔ **Equação Final:**

$$Iniciativa = (1 - FNI) * FP/(1 - FA)$$

Código do Servidor Local (Cálculo da Iniciativa)

Na Classe Agente.java:

```
public void AtualizarIniciativa(List<Leilao> listaLeiloes)
{
    BigDecimal iniciativaAtualizada;
    int somaValoresMaximos = 0;
    int somaValoresMaximosEncerrados = 0;
    int somaValoresGanhos = 0;
    int somaValoresMinimos = 0;
    int leiloesTotais = listaLeiloes.size();
    int leiloesAgente = this.QtdLeiloesParticipantes();

    for (String tipoLeilao : this.auctionTypes) {
        List<Leilao> leiloesNaoFinalizados = listaLeiloes.stream().filter(t -> (t.auctionType.name()
                .contains(tipoLeilao))&&(!t.encerrado)).collect(Collectors.toList());
        somaValoresMaximos = somaValoresMaximos + leiloesNaoFinalizados.stream().map(x -> x.maximumValue)
                .reduce( identity: 0, Integer::sum);

        somaValoresMaximosEncerrados = somaValoresMaximosEncerrados + listaLeiloes.stream()
                .filter(t -> (t.auctionType.name().contains(tipoLeilao))&&(t.encerrado)).map(x -> x.maximumValue)
                .reduce( identity: 0, Integer::sum);

        if (leiloesNaoFinalizados.size()>0) {
            for (Leilao leilao : leiloesNaoFinalizados) {
                somaValoresMinimos = somaValoresMinimos + this.ObterValorMinimo(tipoLeilao);
            }
        }
    }
```

```
    somaValoresGanhos = this.SomarLancesGanhadores();

    if (somaValoresMaximos == 0) somaValoresMaximos = 1;
    if (somaValoresMaximosEncerrados == 0) somaValoresMaximosEncerrados = 1;

    BigDecimal bdsomaValoresMinimos = new BigDecimal(somaValoresMinimos).setScale(3);
    BigDecimal bdsomaValoresMaximos = new BigDecimal(somaValoresMaximos).setScale(3);
    BigDecimal bdsomaValoresMaximosEncerrados = new BigDecimal(somaValoresMaximosEncerrados).setScale(3);
    BigDecimal bdsomaValoresGanhos = new BigDecimal(somaValoresGanhos).setScale(3);
    BigDecimal bdleiloesTotais = new BigDecimal(leiloesTotais).setScale(3);
    BigDecimal bdleiloesAgente = new BigDecimal(leiloesAgente).setScale(3);


    iniciativaAtualizada = new BigDecimal(String.valueOf(
            (new BigDecimal( val: 1.000).subtract(bdsomaValoresMinimos.divide(bdsomaValoresMaximos, RoundingMode.HALF_UP)))
            .multiply(bdleiloesAgente.divide(bdleiloesTotais, RoundingMode.HALF_UP))
            .divide(new BigDecimal( val: 1.000).subtract(bdsomaValoresGanhos
                    .divide(bdsomaValoresMaximosEncerrados, RoundingMode.HALF_UP)),RoundingMode.HALF_UP)))
            .setScale( newScale: 3, RoundingMode.HALF_EVEN);

    this.AtualizarIniciativa(iniciativaAtualizada);
}
```

Mecanismo

Para cada leilão ativo, temos um controle interno de quais agentes podem participar desse leilão e atualizamos o valor do índice de Iniciativa dos agentes para o leilão em questão. Após isso, ordenamos os agentes conforme o valor do índice, sendo que o agente com o menor valor índice dará os lances primeiro.

Ao dar o lance, o agente busca dar um lance menor do que o menor lance atual, caso o menor lance já seja 5, ele busca o empate. Lembrando que aqui, há uma verificação para garantir qual é o menor lance que o agente pode dar, baseado nas restrições do exercício, caso o mínimo que um agente possa dar, em um leilão específico seja 10 e o menor lance atual já é 5, o agente dá o lance de 10, que é o mínimo que ele pode dar e perde o leilão. Uma exceção desse comportamento ocorre quando o agente já ganhou o número de leilões que ele deveria ter ganho, nesse caso ele passa a dar sempre o valor máximo de lance que ele consegue. Na imagem abaixo, temos o código que garante esse comportamento.

```
public Lance DoBid(Leilao leilaoAtual) throws Exception {
    String auctionType = leilaoAtual.auctionType.name();
    Logging.Log(Util.EnumLog.INFO, texto: "Lance do agente: " + this.name);

    int bidValue = 0;
    int max_possible_bid = leilaoAtual.maximumValue;
    int actual_bestBid = leilaoAtual.bestBid;

    if (this.RestricoesEspecificasValidas())
    {
        if (actual_bestBid == 0 && max_possible_bid - 10 > 0)
        {
            bidValue = Util.ObterMaior( a: max_possible_bid - 10, this.estrategia.limitesMinimos.get(auctionType));
        }
        else if (actual_bestBid == 0)
        {
            bidValue = Util.ObterMaior( a: max_possible_bid - 5, this.estrategia.limitesMinimos.get(auctionType));
        }
        else if (actual_bestBid > 5)
        {
            bidValue = Util.ObterMaior( a: actual_bestBid - 5, this.estrategia.limitesMinimos.get(auctionType));
        }
```

```
        else
        {
            bidValue = Util.ObterMaior( a: 5, this.estrategia.limitesMinimos.get(auctionType));
        }
    }
    else bidValue = max_possible_bid;

    if (this.name == "Empresa5" && this.winningBids.stream().map(x->x.bidValue)
            .reduce(0,Integer::sum) + bidValue > 80)
    {
        bidValue = max_possible_bid;
    }

    Lance lance = new Lance(this.id,auctionType,bidValue);
    String info_bid = InterfaceHecate.POSTConnection( command: "/placeBid", idAgente: "", JSONSerializer.SerializarLance(lance));

    this.lances.add(lance);
    return lance;
}
```