Atividade II - Implementação Sequencial

Simulação Monte Carlo da UEFA Champions League 2025

Marcos Vinicius Santos da Silva Reis

Professor: Omar Andres Carmona Cortes

13 de novembro de 2025

1 Introdução

Este relatório apresenta a implementação sequencial de um simulador probabilístico Monte Carlo para a UEFA Champions League 2025. O projeto tem como objetivo desenvolver um sistema capaz de executar milhares de simulações completas do torneio, utilizando técnicas de programação paralela com OpenMP e MPI para demonstrar ganhos reais de performance.

A versão sequencial apresentada neste documento serve como *baseline* para as futuras comparações de desempenho com as versões paralelizadas.

2 Problema e Justificativa

Simular uma única edição da Champions League é computacionalmente simples. No entanto, para estimar probabilidades estatisticamente significativas — como chances de um time ser campeão, avançar às fases eliminatórias ou distribuição de resultados — é necessário realizar **milhares ou milhões de simulações** completas do torneio.

Esse volume de processamento torna-se rapidamente inviável em execução sequencial. Por outro lado, cada simulação é **independente**, caracterizando um problema do tipo *"embarrassingly parallel"*, ideal para aplicação de OpenMP e MPI.

Com 10 milhões de simulações executadas na versão serial, o tempo de processamento foi de aproximadamente **119,42 segundos**. Esse tempo justifica plenamente a necessidade de paralelização para executar volumes ainda maiores de simulações.

3 Modelo Probabilístico

3.1 Atributos dos Times

Cada time é representado por quatro atributos normalizados entre 0 e 100:

- **ATK (Ataque):** Capacidade ofensiva, derivada de gols marcados e expected goals (xG)
- **DEF (Defesa):** Capacidade defensiva, derivada de gols sofridos e clean sheets
- **FORM (Forma):** Performance recente nos últimos jogos

- **EXP (Experiência):** Tradição e histórico em competições europeias

A **força geral** de cada time é calculada através de uma combinação ponderada:

$$\text{FORÇA} = 0{,}35 \times \text{ATK} + 0{,}35 \times \text{DEF} + 0{,}20 \times \text{FORM} + 0{,}10 \times \text{EXP} \tag{1}$$

3.2 Geração de Placares

Para simular o resultado de cada partida, utilizamos a distribuição de Poisson para gerar o número de gols marcados por cada time. O parâmetro λ (média de gols esperada) é calculado com base nas forças dos times:

$$\lambda_A = \left(\frac{\text{ATK}_A + (100 - \text{DEF}_B)}{100} \right) \times \frac{\text{FORM}_A}{100} \times 1{,}5 \tag{2}$$

A implementação da distribuição de Poisson segue o método clássico de Knuth:

```c
int poisson(double lambda) {
    if (lambda <= 0.0) return 0;

    double L = exp(-lambda);
    double p = 1.0;
    int k = 0;

    do {
        k++;
        p *= rand_uniform();
    } while (p > L);

    return k - 1;
}
```

Listing 1: Implementação da distribuição de Poisson

4 Estrutura do Código

4.1 Organização Modular

O projeto foi desenvolvido com uma arquitetura modular, separando responsabilidades:

- `teams.h/c`: Gerenciamento dos times, atributos e cálculo de forças
- `random_utils.h/c`: Funções de aleatoriedade e distribuição de Poisson
- `sim.h/c`: Motor de simulação (sorteio, fase de grupos, mata-mata)
- `stats.h/c`: Acumulação e impressão de estatísticas
- `main.c`: Orquestração do loop Monte Carlo

4.2 Estrutura de Dados

A representação dos times utiliza uma struct compacta:

```c
typedef struct {
    char nome[50];
    double atk;    // ataque (0-100)
    double def;    // defesa (0-100)
    double form;   // forma atual (0-100)
    double exp;    // experiencia (0-100)
    double forca; // forca calculada
    int id;        // identificador unico
} Time;
```

Listing 2: Estrutura de dados de um time

4.3 Pipeline de Simulação

Cada simulação completa da Champions League segue este pipeline:

1. **Montagem dos potes:** Divisão dos 18 times em 3 potes de 6 times cada
2. **Sorteio dos grupos:** Distribuição aleatória em 3 grupos de 6 times
3. **Fase de grupos:** Todos jogam contra todos (ida e volta), totalizando 90 partidas
4. **Classificação:** 2 primeiros de cada grupo + 2 melhores terceiros (8 times)
5. **Mata-mata:** Quartas de final → Semifinais → Final
6. **Registro:** Acumulação das estatísticas

```c
void simular_campeonato(Time times[], Estatisticas *estat_local)
   {
    Time *pote1[TIMES_POR_POTE];
    Time *pote2[TIMES_POR_POTE];
    Time *pote3[TIMES_POR_POTE];

    Grupo grupos[NUM_GRUPOS];
    Time *classificados[8];
    Time *campeao = NULL;

    montar_potes(times, pote1, pote2, pote3);
    sortear_grupos(pote1, pote2, pote3, grupos);
    simular_fase_grupos(grupos, classificados);
    simular_mata_mata(classificados, &campeao);

    registrar_campeao(estat_local, campeao);
    registrar_classificados(estat_local, classificados, 8);
}
```

Listing 3: Função principal de simulação

5 Resultados da Implementação Serial

5.1 Configuração do Experimento

A execução foi realizada com os seguintes parâmetros:

- **Número de simulações:** 10.000.000
- **Ambiente:** Container Docker com Ubuntu 22.04, GCC
- **Compilação:** `gcc -O3 -lm`

5.2 Performance

Tabela 1: Métricas de desempenho da versão serial

Métrica	**Valor**
Tempo de execução (wall-clock)	119,42 segundos
Simulações por segundo	83.736
Tempo médio por simulação	11,94 μs

5.3 Resultados Probabilísticos

Os resultados obtidos após 10 milhões de simulações mostram probabilidades coerentes com a força dos times. A Tabela 2 apresenta as estatísticas completas.

Tabela 2: Probabilidades de cada time por fase

Time	**Grupos**	**Quartas**	**Semis**	**Final**	**Título**
Manchester City	68,11%	68,11%	0,00%	22,12%	22,12%
Real Madrid	62,60%	62,60%	0,00%	15,61%	15,61%
Bayern Munich	59,48%	59,48%	0,00%	12,16%	12,16%
Liverpool	54,48%	54,48%	0,00%	8,87%	8,87%
PSG	54,79%	54,79%	0,00%	7,15%	7,15%
Barcelona	49,72%	49,72%	0,00%	6,54%	6,54%
Arsenal	49,00%	49,00%	0,00%	4,36%	4,36%
Inter Milan	48,33%	48,33%	0,00%	5,18%	5,18%
Atletico Madrid	44,08%	44,08%	0,00%	3,88%	3,88%
Borussia Dortmund	39,35%	39,35%	0,00%	2,16%	2,16%
Juventus	38,98%	38,98%	0,00%	2,48%	2,48%
AC Milan	37,89%	37,89%	0,00%	2,16%	2,16%
Bayer Leverkusen	37,39%	37,39%	0,00%	1,90%	1,90%
RB Leipzig	34,14%	34,14%	0,00%	1,28%	1,28%
Napoli	33,28%	33,28%	0,00%	1,31%	1,31%
Porto	30,38%	30,38%	0,00%	1,16%	1,16%
Benfica	29,44%	29,44%	0,00%	0,88%	0,88%
Newcastle	28,56%	28,56%	0,00%	0,79%	0,79%

6 Observações e Trabalhos Futuros

6.1 Limitações da Versão Atual

Observa-se que as colunas "Semis" apresentam valores zerados devido a um bug na contabilização das estatísticas intermediárias. Este será corrigido na próxima iteração.

6.2 Próximos Passos (Atividade III)

Para a próxima entrega, serão realizadas as seguintes melhorias:

- **Refinamento do modelo probabilístico:** Ajuste dos pesos e inclusão de fatores como vantagem de jogar em casa
- **Atualização dos dados dos times:** Incorporação de estatísticas mais recentes e precisas
- **Paralelização com OpenMP:** Implementação com threads para explorar múltiplos cores
- **Paralelização com MPI:** Implementação distribuída para múltiplos processos
- **Análise de speedup e eficiência:** Comparação quantitativa entre as três versões

7 Conclusão

A implementação sequencial do simulador Monte Carlo da Champions League demonstrou ser funcional e eficiente, processando 83.736 simulações por segundo. O tempo de execução de aproximadamente 2 minutos para 10 milhões de simulações evidencia claramente a necessidade de paralelização para volumes maiores.

O código modular e bem estruturado facilita a futura implementação das versões paralelas com OpenMP e MPI. Os resultados probabilísticos obtidos são coerentes e refletem adequadamente as forças relativas dos times, validando o modelo probabilístico implementado.