UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

JHOAN FERNANDES DE OLIVEIRA

GERAÇÃO EQUITATIVA DE TABELAS EM CAMPEONATOS DE PONTOS CORRIDOS: UM MODELO DE BALANCEAMENTO DAS DISTÂNCIAS DE VIAGEM

PAU DOS FERROS
2025

JHOAN FERNANDES DE OLIVEIRA

GERAÇÃO EQUITATIVA DE TABELAS EM CAMPEONATOS DE PONTOS CORRIDOS: UM MODELO DE BALANCEAMENTO DAS DISTÂNCIAS DE VIAGEM

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Orientador: Prof. Dr. Kennedy Reurison Lopes

PAU DOS FERROS

2025

O48g Oliveira, Jhoan Fernandes de.
Geração equitativa de tabelas em campeonatos de pontos corridos: um modelo de balanceamento das distâncias de viagem / Jhoan Fernandes de Oliveira. – 2025.
74 f. : il.

Orientador: Kennedy Reurison Lopes.
Monografia (graduação) – Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2025.

1. Otimização combinatória. 2. Torneios Round Robin. 3. Balanceamento de distâncias. 4. Meta-heurísticas. 5. Geração de calendários esportivos. I. Lopes, Kennedy Reurison, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

JHOAN FERNANDES DE OLIVEIRA

GERAÇÃO EQUITATIVA DE TABELAS EM CAMPEONATOS DE PONTOS CORRIDOS: UM MODELO DE BALANCEAMENTO DAS DISTÂNCIAS DE VIAGEM

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Defendida em: 05 / 12 / 2025

BANCA EXAMINADORA

Kennedy Reurison Lopes, Prof. Dr. (UFERSA)
Presidente

George Felipe Fernandes Vieira, Prof. (UFERSA)
Membro Examinador

Bruno Borges da Silva, Prof. (UFERSA)
Membro Examinador

AGRADECIMENTOS

Primeiro, quero agradecer a Deus, por me dar vida, saúde, força e sabedoria para chegar até aqui. Sem Ele, nada disso seria possível.

Minha maior gratidão vai para a minha família, principalmente para meu pai e minha mãe. Vocês sempre acreditaram em mim, me apoiaram em tudo e me ensinaram, com muito amor, o valor do esforço e da dedicação. Tudo que conquistei é por causa de vocês.

Quero também agradecer ao meu orientador, Kennedy, por toda paciência e apoio durante esse trabalho.

Aos professores incríveis que tive na graduação, como o próprio Kennedy, Mabel, Rosana, Walber e tantos outros, meu muito obrigado. Cada um de vocês deixou sua marca na minha formação e me inspirou a ir sempre mais longe.

E, claro, a todos que, de alguma forma, estiveram comigo nessa caminhada: meu muito obrigado pelo incentivo, pela força e pelas palavras que fizeram diferença.

Por fim, agradeço aos meus amigos da graduação — Tomaz, Heitor, Gabriel, Vladmir, Bruno, Augusto e todos que fizeram parte dessa jornada.

RESUMO

Campeonatos de pontos corridos no Brasil, devido à sua ampla extensão territorial, geram desigualdades logísticas significativas, que impactam a equidade competitiva entre os clubes. Este trabalho propõe um modelo computacional voltado à geração de tabelas mais equilibradas, com foco na minimização das disparidades de deslocamento. O modelo combina o Método Circular com o unranking de permutação para explorar o vasto espaço de soluções possíveis e aplica uma heurística gulosa para definir os mandos de campo. Implementado em C com paralelização OpenMP, o sistema busca soluções quase-ótimas de forma eficiente. A avaliação foi realizada com base em dados das temporadas de 2022, 2023 e 2024 do Campeonato Brasileiro, comparando o desempenho logístico do modelo com os calendários oficiais da Confederação Brasileira de Futebol (CBF). Os resultados indicam uma melhora substancial na equidade das viagens entre os clubes, demonstrando o potencial da abordagem para reduzir desigualdades logísticas em competições esportivas nacionais.

Palavras-chave: otimização combinatória; torneios Round Robin; balanceamento de distâncias; meta-heurísticas; geração de calendários esportivos.

ABSTRACT

Round-robin tournaments in Brazil, due to their vast territorial extension, generate significant logistical inequalities that impact competitive equity among clubs. This work proposes a computational model aimed at generating more balanced schedules, focusing on minimizing travel disparities. The model combines the Circle Method with permutation unranking to explore the vast space of possible solutions and applies a greedy heuristic to determine home-field assignments. Implemented in C with OpenMP parallelization, the system efficiently searches for near-optimal solutions. The evaluation was conducted using data from the 2022, 2023, and 2024 seasons of the Brazilian Championship, comparing the model's logistical performance with the official CBF schedules. The results indicate a substantial improvement in travel equity among clubs, demonstrating the approach's potential to reduce logistical inequalities in national sports competitions.

Keywords: combinatorial optimization; Round Robin tournaments; distance balancing; metaheuristics; sports schedule generation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

2RR	*double round-robin*
CBF	Confederação Brasileira de Futebol
CT	Centro de Treinamento
ILS	*Iterated Local Search*
mTTP	*Mirrored Traveling Tournament Problem*
NBA	*National Basketball Association*
NFL	*National Football League*
PSO	Otimização por Enxame de Partículas
TARS	*Teams and Rounds Swap*
TCC	Trabalho de Conclusão de Curso
TTP	*Traveling Tournament Problem*
TTPPV	*Traveling Tournament Problem with Predefined Venues*
VNL	*Volleyball Nations League*

SUMÁRIO

1 INTRODUÇÃO

Campeonatos esportivos disputados no formato de pontos corridos, em dois turnos, são comuns em diferentes modalidades ao redor do mundo. Nesse sistema, cada equipe se enfrenta duas vezes, uma em casa e outra fora, o que, em teoria, deveria garantir equilíbrio na distribuição dos confrontos. No entanto, quando esse modelo é aplicado em países ou regiões de grande extensão territorial, como o Brasil, esse equilíbrio pode ser comprometido por fatores logísticos, em especial as longas distâncias percorridas entre rodadas.

No esporte de alto rendimento, viajar não se resume a uma questão operacional ou financeira: trata-se também de um elemento que influencia diretamente a fisiologia e o desempenho dos atletas. Estudos em medicina esportiva indicam que deslocamentos aéreos, sobretudo os de longa duração, podem gerar efeitos adversos no organismo (Leatherwood; Dragoo, 2013). Alterações no ritmo circadiano (o "relógio biológico"), ocorrência de *jet lag*, privação de sono e fadiga acumulada ao longo da temporada estão entre as consequências mais frequentes, todas associadas à queda de performance (Leatherwood; Dragoo, 2013; Charest *et al.*, 2021).

No contexto brasileiro, esse desequilíbrio é evidente e quantitativo. A vasta extensão territorial do país e a distribuição geográfica dos clubes geram disparidades logísticas extremas que afetam a justiça da competição. Análises do Campeonato Brasileiro de Futebol - Série A revelam que, nas temporadas de 2017 e 2018, as equipes campeãs (Corinthians e Palmeiras, respectivamente) foram precisamente aquelas que percorreram as menores distâncias (Araujo, 2024). Em contrapartida, clubes de outras regiões, como o Sport Club do Recife em 2017, chegaram a viajar 3,4 vezes mais que o campeão daquele ano (Araujo, 2024). Esse cenário demonstra que, sem um planejamento logístico focado na equidade, a geografia e as distâncias acabam por influenciando o princípio da isonomia.

Diante desse problema, o presente trabalho propõe o desenvolvimento de um modelo computacional capaz de gerar tabelas mais equitativas para campeonatos de pontos corridos. O objetivo é distribuir de forma mais balanceada as distâncias de viagem entre os participantes, sem alterar o formato tradicional da competição. Assim, busca-se contribuir tanto para a justiça esportiva quanto para o avanço científico na área de otimização aplicada a calendários esportivos.

Para alcançar o objetivo proposto, este trabalho está estruturado nos seguintes capítulos: o Capítulo 2 apresenta a Fundamentação Teórica, abordando os formatos de torneio, o impacto logístico no desempenho e os algoritmos de base. O Capítulo 3 apresenta os Trabalhos Relacionados, analisando outras soluções existentes na literatura. O Capítulo 4 detalha a Metodologia,

incluindo a coleta de dados e as métricas de validação. O Capítulo 5 descreve o Desenvolvimento do Modelo Computacional e sua arquitetura. O Capítulo 6 apresenta e analisa os Resultados obtidos em comparação com os calendários oficiais. Finalmente, o Capítulo 7 apresenta as Conclusões e Trabalhos Futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta a fundamentação teórica que sustenta o desenvolvimento do modelo computacional proposto. São discutidos os principais conceitos associados à estrutura de torneios do tipo *double round robin*, os efeitos logísticos das viagens sobre o desempenho esportivo das equipes e as evidências empíricas que demonstram a disparidade geográfica no Campeonato Brasileiro. Além disso, são introduzidas as técnicas e algoritmos utilizados na solução, incluindo o cálculo de distâncias pela fórmula de Haversine, o método circular para geração de rodadas, o algoritmo de *unranking* via *factoradics* para permutações determinísticas, heurísticas locais para definição de mandante e estratégias de amostragem quasi-aleatória. que constituem o suporte teórico, metodológico e computacional necessário para a formulação e validação do modelo apresentado nos capítulos subsequentes.

2.1 Formato de torneios *double round robin*

O modelo de torneio *double round-robin* (turno e returno), conhecido pela sigla *double round-robin* (2RR), é uma estrutura amplamente adotada na organização de ligas esportivas baseadas em pontos corridos. A ideia central consiste em garantir que cada equipe enfrente todas as demais exatamente duas vezes, sendo uma partida em casa e outra fora. Esse formato assegura simetria de confrontos e comparabilidade de desempenho entre as equipes, o que o torna a base estrutural de campeonatos de longa duração, como o Campeonato Brasileiro de Futebol.

A modelagem de um cronograma 2RR consiste em atribuir cada partida a um período de tempo específico, denominado *time slot* ou rodada, garantindo que nenhuma equipe dispute mais de um jogo por rodada (Bulck; Goossens, 2023). Na literatura especializada, destacam-se dois paradigmas principais de modelagem para torneios 2RR: o modelo compacto e o modelo faseado.

2.1.1 Modelo Compacto

O modelo compacto prioriza a eficiência e a menor duração total do calendário (Bulck; Goossens, 2023). Seu objetivo é garantir que o campeonato seja o mais curto possível em número de rodadas, sem impor restrições quanto à ordem dos jogos. Em um campeonato 2RR com n equipes, esse modelo resulta em $2n-2$ rodadas para cada time, assegurando que todas as partidas sejam disputadas com o mínimo de datas possíveis. Entretanto, ele não controla a sequência

temporal dos confrontos, de modo que a partida de volta entre duas equipes pode ocorrer logo após o jogo de ida, o que reduz a representatividade competitiva do torneio.

2.1.2 Modelo Faseado

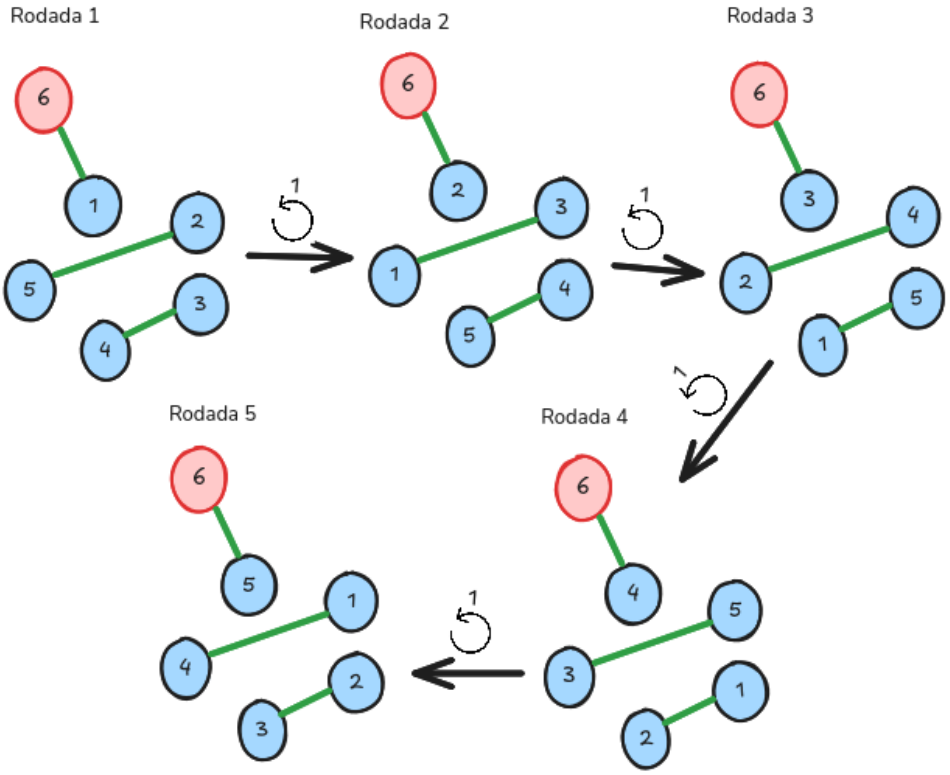
PPara superar as limitações do modelo compacto, adota-se o *modelo faseado*, que mantém a eficiência em termos de número de rodadas, mas introduz uma estrutura interna mais organizada (Bulck; Goossens, 2023). Nesse formato, o campeonato é dividido em duas metades equivalentes: turno e returno. As primeiras $n-1$ rodadas correspondem ao turno, enquanto as $n-1$ rodadas finais compõem o returno, garantindo que a partida de volta entre duas equipes ocorra apenas na segunda metade da competição. Essa estrutura promove maior equilíbrio competitivo e serve de base para a aplicação de heurísticas de mando e restrições de sequência, como o limite máximo de partidas consecutivas em casa ou fora, abordagem adotada neste trabalho.

A Figura 1 ilustra o mecanismo de rotação utilizado para gerar as $n-1$ rodadas de um turno em competições do tipo *round-robin* com n equipes. Nesse esquema, a equipe 6 permanece fixa, enquanto as demais rotacionam a cada rodada, permitindo o agendamento completo do primeiro turno.

Esse procedimento corresponde ao algoritmo de agendamento conhecido como método do círculo ou método de rotação, aplicado aqui ao caso de $n = 6$ equipes. A definição teórica e a implementação computacional desse método são apresentadas na Seção 2.4.2. O funcionamento dos confrontos, representados pelas linhas verdes na Figura 1, baseia-se em dois componentes fundamentais. O primeiro componente corresponde à equipe 6, que atua como pivô fixo. Por se tratar do elemento estático da construção, essa equipe permanece na mesma posição ao longo das cinco rodadas, enfrentando, em cada uma delas, a equipe localizada imediatamente acima, o que resulta em uma sequência ordenada e previsível de partidas. O segundo componente refere-se ao conjunto das demais equipes, numeradas de 1 a 5, que realizam uma rotação de uma posição a cada rodada, no sentido horário indicado pelas setas na Figura 1. Essa movimentação assegura que, ao final das rotações, todas as equipes se enfrentem exatamente uma vez, completando o turno sem sobreposição de partidas em uma mesma rodada.

O método do círculo, portanto, oferece uma construção determinística e eficiente para o agendamento completo de um turno em competições *round-robin*, garantindo o enfrentamento entre todas as equipes e fornecendo a estrutura necessária para a aplicação das restrições e heurísticas consideradas neste estudo.

Figura 1 – Agendamento do primeiro turno para 6 equipes em formato round-robin



Fonte: Autoria própria (2025).

2.2 O impacto da viagem no desempenho esportivo

Viagens de longa distância representam um desafio importante para atletas de alto rendimento, pois diversos fatores podem afetar o desempenho, isoladamente ou combinados. Entre os principais impactos relatados na literatura estão o *jet lag*, a fadiga de viagem, a privação de sono e as condições específicas do voo (Leatherwood; Dragoo, 2013).

2.2.1 Ritmo Circadiano e *Jet Lag*

O principal mecanismo afetado é o ritmo circadiano, nosso "relógio biológico" interno, responsável por regular o ciclo sono-vigília, a temperatura corporal e a produção hormonal. Viagens rápidas que cruzam três ou mais fusos horários causam dessincronização entre o relógio biológico do atleta e o novo ambiente, fenômeno conhecido como *jet lag* (Leatherwood; Dragoo, 2013).

Os sintomas incluem fadiga, distúrbios do sono, dificuldade de concentração, dores de cabeça e desconforto gastrointestinal, todos com potencial para reduzir o desempenho. A direção da viagem também influencia a adaptação: viagens para oeste geralmente são mais fáceis de

ajustar do que para leste, devido ao ritmo circadiano humano ser ligeiramente superior a 24 horas (Leatherwood; Dragoo, 2013).

2.2.2 Fadiga de Viagem e Fatores Associados

Além do *jet lag*, a fadiga acumulada por deslocamentos repetitivos ao longo da temporada pode comprometer a recuperação e a performance, manifestando-se em cansaço, alterações de humor e maior suscetibilidade a doenças (Charest *et al.*, 2021). Entre os fatores que intensificam esse quadro destacam-se a privação de sono, já que a dificuldade de adaptação ao novo fuso horário prejudica o estado de alerta e a cognição; as condições de voo em altitude, pois a pressurização das cabines comerciais equivale a altitudes entre 1.520 e 2.440 metros, provocando hipóxia leve; e ainda a nutrição, uma vez que a dessincronização circadiana e a limitação de acesso a uma dieta adequada podem comprometer o processo de recuperação (Leatherwood; Dragoo, 2013).

2.2.3 Impactos Práticos nos Resultados de Jogos

O efeito dessas variáveis pode ser observado diretamente nos resultados das competições, refletido na conhecida vantagem do time da casa (*home court advantage*). Dados da *National Basketball Association* (NBA), principal liga profissional de basquete dos Estados Unidos, entre 2013 e 2020 indicam que equipes jogando a segunda partida de uma sequência em casa (*Away-Home*) venceram 54,4% das vezes, enquanto aquelas atuando fora venceram apenas 36,8% (*Home-Away*) ou 39,2% (*Away-Away*) (Charest *et al.*, 2021). Esses números evidenciam que a ordem de ocorrência dos jogos exerce influência relevante.

Além disso, tanto a distância percorrida quanto a direção da viagem interferem no desempenho: cada 500 km adicionais de deslocamento reduzem a chance de vitória em aproximadamente 4% (Charest *et al.*, 2021), e viagens no sentido leste tendem a ser mais prejudiciais que no sentido oeste, devido à interação entre horário e ritmo circadiano (Leatherwood; Dragoo, 2013).

2.3 Análise empírica da disparidade logística no campeonato brasileiro

Como discutido na seção anterior, viagens de longa distância e a fadiga associada podem afetar significativamente o desempenho dos atletas. Para além dos impactos fisiológicos, estudos

de observação também têm buscado mensurar a dimensão do problema logístico em campeonatos de futebol, evidenciando como a distribuição das partidas ao longo do calendário pode criar desigualdades competitivas.

A análise das tabelas oficiais de jogos revela diferenças importantes nos deslocamentos entre os clubes, mostrando que a disposição dos confrontos pode impactar de forma desigual o desempenho das equipes (Araujo, 2024). Equipes da região Sudeste, especialmente do eixo Rio de Janeiro-São Paulo, registram deslocamentos consideravelmente inferiores à média da competição. Em contrapartida, clubes da região Nordeste percorrem distâncias muito superiores aos demais concorrentes (Araujo, 2024).

Um dos pontos mais relevantes desse estudo é a correlação entre menor deslocamento e maior desempenho. Nas edições de 2017 e 2018, as equipes campeãs foram justamente aquelas que percorreram a menor distância total entre todos os participantes (Araujo, 2024). A desigualdade logística fica evidente quando se observa que, em 2017, o Sport Club do Recife viajou 3,4 vezes mais que o campeão Corinthians, e, em 2018, o Ceará Sporting Club percorreu 3,3 vezes mais quilômetros que o campeão Palmeiras (Araujo, 2024).

Esses dados reforçam a necessidade de ferramentas computacionais capazes de organizar campeonatos de forma mais equilibrada, minimizando as diferenças de deslocamento entre os participantes e promovendo maior justiça competitiva.

2.4 Algoritmos e técnicas computacionais utilizadas

2.4.1 Fórmula de Haversine

Funcionamento do Método

A fórmula de Haversine é utilizada para calcular a menor distância entre dois pontos na superfície de uma esfera (Winarno; Hadikurniawati; Rosso, 2017). Embora a forma real da Terra seja um esferoide oblato (ligeiramente achatado nos polos), a fórmula de Haversine considera o planeta como uma esfera perfeita (Winarno; Hadikurniawati; Rosso, 2017).

O método utiliza como entrada os valores de latitude (lat) e longitude (long) dos dois pontos. Um passo preliminar essencial, antes da aplicação das equações, é converter essas coordenadas, geralmente expressas em graus decimais, para radianos, pois as funções trigonométricas envolvidas exigem essa unidade (Winarno; Hadikurniawati; Rosso, 2017).

As Fórmulas

O cálculo da distância d é realizado em etapas, conforme a metodologia apresentada por (Winarno; Hadikurniawati; Rosso, 2017). Dados dois pontos $P_1 = (\text{lat}_1, \text{long}_1)$ e $P_2 = (\text{lat}_2, \text{long}_2)$, com todas as coordenadas previamente convertidas para radianos, inicia-se determinando as variações de latitude e longitude.

$$\Delta\text{lat} = \text{lat}_2 - \text{lat}_1, \tag{1}$$

$$\Delta\text{long} = \text{long}_2 - \text{long}_1. \tag{2}$$

Em seguida, calcula-se a variável intermediária a, que corresponde à metade do quadrado do comprimento da corda entre os pontos, conforme a Equação 3.

$$a = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat}_1)\cos(\text{lat}_2)\sin^2\left(\frac{\Delta\text{long}}{2}\right). \tag{3}$$

A distância angular c em radianos é então obtida pela Equação 4, que utiliza a função atan2 para apresentar essa relação.

$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right). \tag{4}$$

Por fim, a distância real d resulta da multiplicação de c pelo raio médio da Terra R, conforme a Equação 5.

$$d = R \cdot c. \tag{5}$$

O valor de R é considerado constante. Um valor amplamente adotado é $R = 6371$ quilômetros. O resultado final d corresponde à menor distância entre os dois pontos sobre a superfície esférica.

A implementação computacional desta fórmula, utilizada no modelo final para o cálculo da matriz de distâncias, é detalhada no Apêndice A.

2.4.2 Geração de Pares Round-Robin (Método Circular)

O Método Circular, também denominado método do polígono ou procedimento canônico, é uma técnica clássica para a construção de tabelas de torneios do tipo *single round-robin* (SRR). Proposto originalmente por Kirkman em 1847, o método permanece amplamente utilizado em competições esportivas contemporâneas, incluindo ligas de futebol europeias (Lambrechts *et al.*, 2017).

O algoritmo é idealizado um número par de equipes, denotado por n. De maneira intuitiva, conforme ilustrado na Figura 1, uma das equipes, tipicamente a equipe n, é mantida fixa, enquanto as demais $n-1$ são distribuídas circularmente. Na rodada inicial, a equipe fixa enfrenta a equipe posicionada diametralmente oposta no círculo. Os demais confrontos formam-se por pares de equipes cuja posição também é oposta no arranjo circular (Lambrechts *et al.*, 2017).

A geração das rodadas subsequentes ocorre por meio da rotação das equipes ao redor do círculo, mantendo a equipe fixa em sua posição original. A formulação matemática do método estabelece que, na rodada $r \in \{1,\ldots,n-1\}$, a equipe n enfrenta a equipe r. Para as demais equipes i e j, com $i,j \in N \setminus \{r,n\}$, o confronto ocorre quando $i+j \equiv 2r \pmod{n-1}$. Esse procedimento garante que cada equipe enfrente todas as demais exatamente uma vez, resultando em um turno completo de competição (Lambrechts *et al.*, 2017).

A implementação em linguagem C utilizada para gerar os calendários-base deste trabalho encontra-se no Apêndice B.

2.4.3 Algoritmo de Unranking de Permutação

A geração de uma permutação específica a partir de um índice numérico, processo conhecido como *unranking*, constitui um problema combinatório clássico. No presente estudo, tal mecanismo desempenha papel fundamental, pois permite que uma única `seed` numérica determine de maneira determinística uma ordem de equipes, assegurando reprodutibilidade dos experimentos e possibilitando a exploração estruturada do espaço de soluções.

Uma abordagem eficiente para realizar o unranking utiliza o sistema de numeração fatorial, também denominado *factoradics* (Genitrini; Pépin, 2021). Esse sistema estabelece uma correspondência bijetiva entre um inteiro u, interpretado como o rank ou índice, e uma permutação de n elementos. Para qualquer $u \in [0, n!-1]$, existe uma representação fatorádica única, expressa pela decomposição:

$$u = f_{n-1}(n-1)! + f_{n-2}(n-2)! + \cdots + f_l\, l! + \cdots + f_0\, 0!. \tag{6}$$

Nessa representação, n corresponde ao número total de elementos. O índice l indica qual termo fatorial está sendo utilizado e varia de 0, associado a $0!$, até $n-1$, associado a $(n-1)!$. Cada coeficiente f_l deve satisfazer a condição $0 \leq f_l \leq l$, assim como um dígito decimal não pode exceder o valor 9. Dessa forma, um "dígito" fatorádico f_l também não pode ultrapassar seu índice l, o que garante que exista apenas uma sequência possível de coeficientes $(f_{n-1},\ldots,f_0)$

para qualquer valor de u. Essa restrição assegura a unicidade da decomposição fatorádica (Genitrini; Pépin, 2021).

O algoritmo de unranking utiliza diretamente os coeficientes definidos pela Equação 6. O procedimento inicia com a criação de uma lista contendo todos os n elementos a serem permutados, tipicamente representados como $[0, 1, \ldots, n-1]$. Para cada posição i da permutação final, determina-se o coeficiente fatorádico f_{n-i-1} correspondente ao valor corrente de u. Esse coeficiente é então usado como índice para selecionar um elemento da lista de itens disponíveis.

Após a seleção, o elemento é alocado na posição i da permutação e removido da lista de disponíveis. Em seguida, o valor de u é atualizado para refletir o restante da decomposição fatorádica. O processo repete-se de maneira sequencial até que todos os elementos tenham sido escolhidos e a lista esteja vazia. Ao final da execução, a sequência obtida corresponde exatamente à u-ésima permutação na ordem lexicográfica(Genitrini; Pépin, 2021).

A implementação em linguagem C utilizada neste trabalho, baseada no sistema fatorádico, é apresentada no Apêndice D.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão da literatura científica focada em problemas de otimização e geração de calendários esportivos, com ênfase especial em abordagens que consideram o balanceamento de distâncias de viagem. A busca por trabalhos correlatos foi realizada em bases de dados como *Periódicos CAPES* e *Google Scholar*, utilizando palavras-chave como *"sports scheduling"*, *"round-robin tournaments"*, *"distance balancing"* e *"travel optimization"*.

O objetivo desta análise não é apenas listar contribuições existentes, mas sim identificar as metodologias empregadas, as limitações reconhecidas e as lacunas que o presente trabalho busca endereçar. As seções a seguir detalham as propostas mais relevantes encontradas, analisando suas contribuições e deficiências no contexto do desequilíbrio logístico. Ao final, um quadro comparativo sintetiza as características de cada abordagem em relação à solução desenvolvida neste Trabalho de Conclusão de Curso (TCC).

3.1 Busca local para otimização de distâncias

O trabalho de Januario e Urrutia (2016) foca no desenvolvimento de heurísticas de busca local para problemas de agendamento esportivo *round-robin*. A principal contribuição dos autores é uma nova "estrutura de vizinhança", chamada *Teams and Rounds Swap* (TARS) (*Teams and Rounds Swap*), projetada para explorar o espaço de soluções de forma mais eficaz do que os métodos de troca tradicionais (Januario; Urrutia, 2016).

Os autores utilizam dois problemas como estudo de caso: o *Traveling Tournament Problem with Predefined Venues* (*Traveling Tournament Problem with Predefined Venues* (TTPPV)) e a minimização de *carry-over effects*. Para a análise deste TCC, o foco recai sobre o TTPPV, pois este visa explicitamente minimizar a distância total viajada pelas equipes, um objetivo de eficiência, que difere do foco em equidade do presente trabalho.

Metodologicamente, os autores aplicam uma heurística de *Iterated Local Search* (*Iterated Local Search* (ILS)). Esta abordagem parte de uma solução inicial e aplica perturbações (movimentos de busca local) para refinar iterativamente uma solução "vizinha". É crucial notar que a validação do TTPPV (o problema de distância) não foi realizada em um cenário real como o Campeonato Brasileiro, mas sim em instâncias de *benchmark* abstratas, que não utilizam distâncias geodésicas reais (km).

3.2 Equidade via teoria dos jogos

O trabalho de Osicka e Guajardo (2023) fornece uma fundamentação teórica robusta para a premissa deste TCC. Os autores criticam o *Traveling Tournament Problem Traveling Tournament Problem* (TTP) clássico por focar apenas na minimização da distância total, o que "ignora a distribuição da viagem entre as equipes" e pode gerar um "desequilíbrio" (Osicka; Guajardo, 2023). O artigo propõe explicitamente uma metodologia para encontrar tabelas com base em "critérios de justiça" (*fairness*).

A abordagem dos autores, baseada na Teoria dos Jogos Cooperativos, difere metodologicamente deste TCC. Osicka e Guajardo (2023) utilizam essa teoria para primeiro calcular uma "distribuição de distância ideal" (justa), usando conceitos como o "Método Igualitário". Em seguida, buscam um calendário real que mais se aproxime desse ideal teórico, introduzindo métricas de justiça e discutindo o "conflito" (*trade-off*) entre justiça e eficiência (Osicka; Guajardo, 2023).

Enquanto este TCC adota uma abordagem computacional *bottom-up* (amostragem em larga escala para *encontrar* a menor amplitude), este artigo utiliza uma abordagem teórica *top-down* (*definir* o que é justo e depois tentar modelar um calendário). A validação dessa abordagem é demonstrada em uma instância clássica de 6 equipes da liga norte-americana(Osicka; Guajardo, 2023).

3.3 Agendamento do campeonato brasileiro

O trabalho de Ribeiro e Urrutia (2011) é de grande relevância, pois descreve a metodologia usada em parceria com a Confederação Brasileira de Futebol (CBF) para criar as tabelas do Campeonato Brasileiro. O objetivo principal do modelo dos autores foi a maximização da receita de bilheteria e das audiências de TV. Para isso, o modelo busca alocar os "jogos clássicos" em datas de maior audiência, como fins de semana.

Embora o artigo mencione "justiça" (*fairness*) e "equilíbrio" (*balance*), esses conceitos são aplicados a regras operacionais da CBF, como a sequência de jogos em casa e fora ou restrições de jogos no mesmo estado. A abordagem deles difere fundamentalmente deste TCC, pois a equidade é tratada em termos de audiência e padrões de jogos, enquanto as distâncias de viagem, o foco central deste trabalho, não são o principal objetivo da otimização.

3.4 Equidade de viagem na liga das nações de voleibol

O trabalho de Lambers, Rothuizen e Spieksma (2023) fornece uma forte fundamentação para este TCC, pois foca explicitamente na injustiça competitiva (*unfairness*) causada pela logística de viagem. O artigo analisa a Liga das Nações de Voleibol (*Volleyball Nations League* (VNL)), um torneio global onde é comum haver uma "grande discrepância entre as cargas de viagem das equipes adversárias", o que é considerado uma "desvantagem para a equipe que viajou mais". O objetivo principal dos autores é idêntico ao deste trabalho: "minimizar uma medida que captura o desequilíbrio (*imbalance*) nos tempos de viagem entre equipes adversárias". O artigo também faz a distinção crucial de que o *Traveling Tournament Problem* (TTP) clássico foca na distância *total*, enquanto o problema deles (e o deste TCC) foca no *desequilíbrio* da distância.

Metodologicamente, o trabalho de Lambers et al. (2023) difere deste TCC. Devido ao formato do torneio (baseado no *Social Golfer Problem*), eles propõem uma abordagem de decomposição em duas fases: o *Venue Assignment Problem* (Problema de Alocação de Sedes) e o *Nation Assignment Problem* (Problema de Alocação das Nações). Os autores provam que a alocação das sedes é o que determina a injustiça total e resolvem este problema usando métodos de Programação Inteira, alcançando reduções drásticas na injustiça em comparação com os calendários reais.

3.5 O TTP clássico e eficiência total

O trabalho de Goerigk *et al.* (2014) serve como um exemplo claro da abordagem clássica do *Traveling Tournament Problem* (TTP). O objetivo central do modelo é a eficiência, definida como a minimização da distância total (a soma) viajada por todas as equipes da liga (Goerigk *et al.*, 2014). Esta abordagem difere do foco deste TCC, que não visa a distância total, mas sim a equidade (a amplitude) na distribuição dessa distância entre as equipes.

Metodologicamente, Goerigk *et al.* (2014) propõem uma abordagem híbrida e inovadora para resolver o TTP. Eles introduzem um novo método de geração de calendário-base (Fase 1) fundamentado no conceito de *P3-packing* (empacotamento de caminhos de três vértices) da teoria dos grafos. Este calendário inicial é, então, otimizado através de busca local e Programação Inteira (Fases 2 e 3) para encontrar a menor soma total de distâncias, validando a eficácia do método em instâncias de *benchmark* como a *National Football League* (NFL) que é a liga

profissional de futebol americano dos Estados Unidos.(Goerigk *et al.*, 2014).

3.6 Meta-heurística PSO para o TTP espelhado

O trabalho de Tole, Milani e Mwakondo (2021) aborda o (*Mirrored Traveling Tournament Problem* (mTTP)) (Tole; Milani; Mwakondo, 2021), o mesmo formato *double round-robin* espelhado deste TCC (Tole; Milani; Mwakondo, 2021). No entanto, o artigo foca em um objetivo de eficiência total, visando especificamente "minimizar o número de viagens" (Tole; Milani; Mwakondo, 2021) realizadas por todas as equipes, em vez de minimizar a distância total em quilômetros. Esta abordagem difere do foco deste TCC, que é a equidade da distância (amplitude) e não a soma total de viagens.

A principal contribuição metodológica do artigo é a aplicação da meta-heurística de Otimização por Enxame de Partículas (Otimização por Enxame de Partículas (PSO))(Tole; Milani; Mwakondo, 2021) para encontrar uma atribuição de mando de campo (*home-away assignment*) quase-ótima (Tole; Milani; Mwakondo, 2021). Embora o trabalho mencione a importância de "questões de justiça (*fairness*)" (Tole; Milani; Mwakondo, 2021) e uma regra heurística para que as equipes tenham distâncias de viagem "semelhantes" (Tole; Milani; Mwakondo, 2021), a equidade não é a função-objetivo principal, mas sim uma consideração secundária. O foco na eficiência total (número de viagens) e o uso de PSO fornecem um contraponto metodológico claro a este TCC.

3.7 Análise comparativa

Quadro 1 – Análise e comparação dos trabalhos relacionados

Trabalho	Foco na Equidade (Amplitude)	Formato Double Round-Robin	Usa Método Circular (Base)	Otimiza Mando de Campo (decidir quem será o mandante ou sede)	Explora Permutações Iniciais (para o metodo circular)	Amostragem inicial de tabelas em larga escala	Execução Paralela	Validado em circunstâncias reais	Contexto principal: Campeonato Brasileiro
Januario e Urrutia (2016)			X						
Osicka e Guajardo (2023)	X	X						X	
Ribeiro e Urrutia (2011)		X		X				X	X
Lambers, Rothuizen e Spieksma (2023)	X			X				X	
Goerigk *et al.* (2014)		X		X	X			X	
Tole, Milani e Mwakondo (2021)		X		X					
Este Trabalho	X	X	X	X	X	X	X	X	X

Fonte: Autoria própria (2025).

Como observado no Quadro 1, a literatura de agendamento esportivo divide-se em duas vertentes principais: trabalhos focados na **eficiência** (minimização da distância ou número total de viagens), como o TTP clássico (Goerigk *et al.*, 2014; Tole; Milani; Mwakondo, 2021), e um número emergente de estudos focados na **equidade** (balanceamento ou justiça da distribuição), como Osicka e Guajardo (2023) e Lambers, Rothuizen e Spieksma (2023). Este TCC se alinha a esta segunda vertente, abordando uma lacuna deixada por trabalhos anteriores no contexto do Campeonato Brasileiro, que historicamente priorizaram métricas de audiência em vez de equidade logística (Ribeiro; Urrutia, 2011).

Metodologicamente, este trabalho propõe uma abordagem computacional distinta. Enquanto os trabalhos relacionados sobre equidade empregam Teoria dos Jogos (Osicka; Guajardo, 2023) ou Programação Inteira (Lambers; Rothuizen; Spieksma, 2023), este TCC introduz uma nova arquitetura híbrida. A solução aqui desenvolvida combina a geração de um calendário-base determinístico (o Método Circular) com a **exploração de permutações iniciais** e uma **amostragem em larga escala** do espaço de soluções, viabilizada por **execução paralela**. Esta combinação de técnicas, focada explicitamente na minimização da amplitude de viagem, representa uma contribuição original para a geração de tabelas logisticamente mais justas.

4 METODOLOGIA

Este capítulo apresenta a metodologia adotada para o desenvolvimento e validação do modelo computacional proposto. São descritos o plano de pesquisa, o tipo de estudo, o processo de coleta e preparo dos dados, o desenho do experimento, as ferramentas utilizadas e as métricas de análise que permitem avaliar o desempenho do modelo em comparação com os calendários oficiais da CBF.

4.1 Classificação da pesquisa

A pesquisa desenvolvida combina características de investigação aplicada e experimental, com o objetivo de propor soluções práticas para um problema real de logística esportiva. Inicialmente, foi realizada uma revisão de estudos e métodos existentes sobre otimização de calendários esportivos, de modo a embasar o desenvolvimento do modelo computacional.

4.1.1 Abordagem da Pesquisa

A metodologia adotada é de caráter aplicado, pois busca criar um modelo computacional capaz de gerar tabelas do campeonato que apresentem menor desigualdade logística, levando em conta as distâncias percorridas pelas equipes. A pesquisa também possui natureza quantitativa, uma vez que se apoia em métricas numéricas para avaliar e comparar o desempenho do modelo em relação aos dados oficiais da CBF.

4.1.2 Tipo de Pesquisa

O estudo pode ser classificado como exploratório e experimental. É exploratório porque busca compreender o estado da arte em algoritmos de otimização aplicados a calendários esportivos e identificar possíveis melhorias. É também experimental, pois envolve a implementação prática do modelo e a avaliação dos seus resultados na redução da desigualdade logística, comparando-os com as tabelas oficiais da CBF.

4.2 Coleta e preparo dos dados

4.2.1 Universo e Amostra

O universo da pesquisa é formado pelas edições do Campeonato Brasileiro de Futebol - Série A. A amostra selecionada inclui as temporadas de 2022, 2023 e 2024, permitindo observar o comportamento do modelo em diferentes contextos geográficos e esportivos. Essa escolha possibilita avaliar a consistência dos resultados e compará-los com as tabelas oficiais da CBF.

4.2.2 Obtenção dos Dados Geográficos

As coordenadas geográficas dos clubes foram coletadas seguindo uma ordem de prioridade: inicialmente buscou-se o endereço oficial do Centro de Treinamento (Centro de Treinamento (CT)); quando não disponível, utilizou-se o da sede administrativa; e, por fim, o do estádio, caso as opções anteriores não fossem encontradas. As informações foram obtidas principalmente nos sites oficiais dos clubes e com cordenadas cedidas pelo Google Maps.

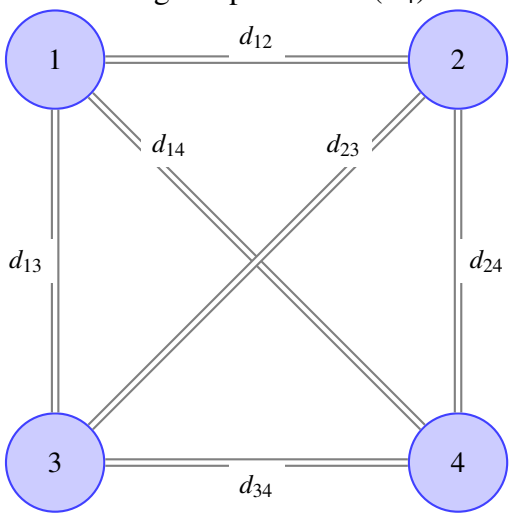
4.2.3 Modelagem dos Deslocamentos como Grafo

Para modelar computacionalmente o problema dos deslocamentos, a rede de viagens entre os clubes foi representada como um grafo ponderado completo e não-direcionado, $G = (V, E)$. Esta é uma abstração padrão, onde $G = (V, E)$ representa o grafo (Cormen *et al.*, 2024) e a noção de "ponderado" é usada para associar um custo, ou distância, a cada aresta (Cormen *et al.*, 2024).

Nesta abstração fundamental, V é o conjunto de vértices e E é o conjunto de arestas. Cada vértice $v \in V$ representa a localização geográfica de um clube, conforme os dados coletados na Seção 4.2.2, e para $N = 20$ clubes tem-se $|V| = 20$. Uma aresta $e \in E$ conecta cada par de vértices (u, v), representando a rota de viagem direta entre dois clubes. O peso $w(e)$ associado a cada aresta corresponde à distância geodésica (em quilômetros) entre os dois vértices conectados, calculada pela fórmula de Haversine, conforme fundamentado na Seção 2.4.1.

O modelo é um grafo completo, denotado por K_N, pois assume-se que existe uma rota direta entre qualquer par de clubes. Essa modelagem de grafo completo é classificada como um grafo denso (Cormen *et al.*, 2024), justificando a estrutura de dados escolhida a seguir. A Figura 2 ilustra o conceito de conectividade total usando um exemplo simplificado de um grafo completo com $N = 4$ vértices (K_4).

Figura 2 – Ilustração conceitual de um grafo ponderado (K_4)



Fonte: Autoria própria (2025).

A representação teórica deste grafo é uma matriz de adjacência $N \times N$, uma das representações computacionais padrão para grafos e preferível para grafos densos (Cormen *et al.*, 2024). Como o grafo é não-direcionado (a distância $i \to j$ é a mesma que $j \to i$), essa matriz é simétrica ($M_{ij} = M_{ji}$), propriedade confirmada por (Cormen *et al.*, 2024, p. 387), que afirmam que "a matriz de adjacências A de um grafo não dirigido é sua própria transposta: $A = A_T$".

Para otimizar o armazenamento e o processamento, o modelo computacional não utiliza a matriz $N \times N$ completa como entrada. Em vez disso, ele é alimentado com uma estrutura de dados que armazena apenas os pares únicos de distâncias, correspondentes à triangular inferior ou superior da matriz. Trata-se de uma otimização padrão para matrizes simétricas, que "reduz quase à metade a memória necessária para armazenar o grafo" (Cormen *et al.*, 2024, p. 387). Essa estrutura, referida neste trabalho como Tabela de Distâncias, contém os $N(N-1)/2 = 190$ valores únicos de deslocamento para $N = 20$ clubes. As matrizes de adjacência simétricas completas, das quais essas tabelas foram derivadas, encontram-se nos apêndices (H, I e J) para fins de ilustração.

4.2.4 Dados de Referência

Os calendários oficiais da CBF foram utilizados como grupo de controle. Eles não foram usados como entrada do modelo, mas serviram como referência de comparação. Como a CBF não divulga seus pontos de referência geográficos nem sua metodologia de cálculo, aplicou-se o mesmo procedimento adotado neste trabalho para estimar as distâncias percorridas em seus calendários. Assim, a comparação feita no Capítulo 6 considera ambos os agendamentos com a

mesma base metodológica.

4.3 Desenho do experimento computacional

4.3.1 Variáveis de Entrada

O modelo recebe três entradas principais: o número de iterações, que define o esforço computacional; a tabela de distâncias, contendo os pares únicos entre os clubes; e o mapeamento dos times, que associa cada equipe ao identificador utilizado na tabela.

4.3.2 Função Objetivo

A função objetivo busca minimizar a amplitude das distâncias percorridas pelos clubes, isto é, a diferença entre a maior e a menor distância total. Também são analisados o desvio padrão, a média e a soma total das distâncias, permitindo avaliar o nível de equilíbrio logístico.

4.3.3 Processo Experimental

O experimento é executado iterativamente. Em cada iteração, o modelo gera um novo calendário e o avalia segundo a função objetivo. O melhor resultado encontrado é armazenado, e ao final retorna-se o calendário mais equilibrado entre todos os testados dentro do limite de iterações definido.

4.3.4 Justificativa das Ferramentas e Técnicas

As escolhas de implementação adotadas neste trabalho respondem diretamente à escala do problema. A geração de um calendário esportivo constitui um problema de otimização combinatória de elevada complexidade, no qual a dificuldade não está nos dados de entrada, como a tabela de distâncias, mas sim no espaço de busca de todas as soluções possíveis.

Quantificação do Espaço de Busca

O modelo proposto explora o espaço de soluções por meio da permutação da ordem dos 20 times antes da aplicação do Método Circular (fundamentado na Seção 2.4.2 e detalhado no Apêndice B). O número total de permutações de n elementos é dado por $n!$. Para $n = 20$, tem-se:

$$20! = 2.432.902.008.176.640.000$$

o que corresponde a aproximadamente 2,43 quintilhões de tabelas-base distintas. A verificação exaustiva de todas essas possibilidades é impraticável. Além disso, o cálculo desconsidera a complexidade adicional associada à definição dos mandos de campo, que adicionaria outro fator exponencial caso não fosse tratado por uma heurística específica.

Diante dessa explosão combinatória, buscar a solução ótima global é inviável. O presente trabalho foi, portanto, estruturado para obter uma solução quase-ótima[1] por meio da amostragem de uma pequena fração desse espaço. Essa necessidade orientou diretamente as decisões de implementação.

A linguagem C foi escolhida devido ao seu desempenho e à capacidade de executar operações de baixo nível, permitindo a geração e avaliação de milhões de calendários candidatos por segundo. A paralelização com OpenMP ampliou essa capacidade ao distribuir o trabalho entre todos os núcleos de processamento disponíveis. Essa combinação viabilizou, conforme relatado no Capítulo 6, a análise de 1×10^9 soluções para cada temporada em um tempo inferior a 600 segundos em um computador pessoal.

A definição dos mandos de campo, caso tratada de forma exata, acrescentaria um novo nível de complexidade. Assim, foi adotada uma heurística gulosa, detalhada no Apêndice C, que resolve esse subproblema tomando decisões locais simples e eficientes. Essa escolha permite concentrar o esforço computacional na exploração do espaço de permutações, que é o principal fator responsável pela variação das distâncias totais.

O processo de amostragem também foi cuidadosamente projetado. Embora o experimento avalie 1×10^9 soluções, essa quantidade ainda representa apenas:

$$\frac{1 \times 10^9}{2{,}43 \times 10^{18}} \approx 4{,}1 \times 10^{-10} \quad \text{ou} \quad 0{,}000000041\%.$$

Dado que apenas uma fração extremamente pequena do espaço total pode ser explorada, faz-se necessário um método de amostragem sistemático e eficiente. O algoritmo de unranking, fundamentado na Seção 2.4.3 e apresentado no Apêndice D, possibilita gerar deterministicamente qualquer permutação a partir do seu índice. Isso permite selecionar e testar conjuntos amplos de soluções espalhadas pelo espaço de busca, aumentando a probabilidade de identificar configurações de alta qualidade.

Em síntese, a combinação entre C, OpenMP, a heurística de mandos de campo e o unranking constitui uma solução pragmática e de alto desempenho, adequada à enorme escala

[1] O uso de meta-heurísticas é comum para encontrar soluções aproximadas em espaços de busca muito grandes ou com múltiplos ótimos locais, nos quais métodos exatos tornam-se impraticáveis. (Arcas *et al.*, 2024)

combinatória que caracteriza o problema investigado.

4.4 Métricas de análise e validação

4.4.1 Métricas Quantitativas

O desempenho do modelo é avaliado por métricas como amplitude, desvio padrão, média e distância total individual percorrida pelos clubes. Essas medidas permitem observar tanto a dispersão quanto o equilíbrio das viagens.

4.4.2 Análise Comparativa

Os resultados do modelo são apresentados em tabelas e gráficos comparativos, permitindo observar lado a lado o Calendário Gerado e o Calendário Oficial da CBF.

O modelo, desenvolvido em C, gera como saída as distâncias percorridas por cada clube. Esses dados são processados em Python para a criação dos gráficos, utilizando a biblioteca `matplotlib`. Foi escolhido um gráfico polar, onde cada ponto representa um clube e sua distância total é mostrada em milhares de quilômetros.

No gráfico, os clubes que viajam acima da média são destacados em vermelho, enquanto os que viajam menos aparecem em verde. Uma linha azul conecta os pontos e uma linha tracejada indica a média geral, permitindo visualizar facilmente o nível de equilíbrio logístico entre os clubes.

4.4.3 Limitações e Delimitação do Escopo

O modelo desenvolvido neste trabalho tem como foco principal a otimização da equidade nas distâncias percorridas.

Sabe-se que a elaboração do calendário oficial da CBF envolve diversas restrições como esportivas, logísticas, comerciais e de segurança, muitas das quais não são publicamente documentadas. Por isso, este trabalho optou por delimitar o escopo, considerando apenas as restrições básicas de um torneio *round-robin*, como o controle de jogos consecutivos em casa e fora.

Assim, o modelo representa um cenário idealizado de otimização logística. A comparação com os calendários oficiais da CBF, apresentada no Capítulo 6, deve ser interpretada sob essa

perspectiva: o modelo proposto busca o equilíbrio nas distâncias, enquanto o calendário real atende a múltiplas restrições práticas e operacionais.

É importante ressaltar que a equidade logística, foco deste trabalho, é apenas um dos componentes da complexa noção de "justiça" em um torneio. A proposta de equalizar distâncias não é a única maneira de tornar o campeonato mais justo, mas busca mitigar uma das disparidades mais evidentes e mensuráveis no contexto continental do Brasil.

Diversos outros fatores que contribuem para o desempenho dos times e o desgaste dos atletas não foram analisados. Tais variáveis incluem, mas não se limitam a: desgaste gerado por viagens através de diferentes fusos horários, o tempo de recuperação (em horas) entre as partidas, restrições de malha aérea que forçam conexões, ou mesmo fatores econômicos que limitam as opções de fretamento das equipes. Tais variáveis, embora fora do escopo deste estudo, representam oportunidades valiosas para pesquisas futuras que busquem um modelo de otimização ainda mais holístico.

5 DESENVOLVIMENTO DO MODELO COMPUTACIONAL

O presente capítulo detalha o processo de implementação e a arquitetura do modelo computacional proposto, cujo objetivo central é a geração de tabelas para campeonatos de pontos corridos com foco no balanceamento das distâncias de viagem. Fundamentado nos algoritmos e conceitos teóricos discutidos no Capítulo 2 e aderindo ao desenho experimental definido no Capítulo 3, este capítulo foca na tradução prática de tais conceitos em um modelo computacional.

Será apresentada a evolução do desenvolvimento, demonstrando a jornada investigativa que se iniciou em abordagens exploratórias, como a geração puramente pseudo-aleatória, e culminou na arquitetura híbrida final. O objetivo é justificar as decisões de design e detalhar o funcionamento dos componentes que permitem ao modelo construir, otimizar e explorar o espaço de soluções, buscando atender à função objetivo de minimizar a amplitude logística entre as equipes.

O desenvolvimento do modelo computacional não foi um processo linear. A arquitetura final, detalhada na Seção 5.3, resulta de um processo iterativo de pesquisa e prototipação. As abordagens iniciais, embora mais simples em conceito, revelaram-se computacionalmente inviáveis ou incapazes de satisfazer a função objetivo de minimização da amplitude.

5.1 abordagem inicial: geração pseudo-aleatória por amostragem

A primeira versão do modelo computacional foi desenvolvida com o objetivo de gerar calendários completos através de um método de amostragem pseudo-aleatória. A lógica consistia em gerar um grande volume de soluções candidatas e, posteriormente, avaliá-las, retendo apenas o calendário que apresentasse a melhor métrica de equidade logística (a menor amplitude entre as distâncias percorridas pelas equipes).

O núcleo dessa implementação tentava montar uma rodada válida, na qual nenhum time joga duas vezes, selecionando partidas pseudo-aleatoriamente de um conjunto de confrontos disponíveis. Um mecanismo de embaralhamento era usado extensivamente para randomizar a ordem das partidas antes de cada tentativa de alocação, enquanto uma rotina de validação assegurava a integridade da rodada, verificando se os times já haviam sido escalados.

Esta abordagem apresentou limitações críticas que a tornaram inviável. A principal delas foi a ausência de garantia estrutural: diferente de algoritmos determinísticos como o Método Circular, o modelo não oferecia nenhuma garantia de que uma rodada completa pudesse

ser formada, dependendo inteiramente da sorte para encontrar uma combinação válida de $n/2$ partidas.

Associado a isso, o mecanismo de recuperação de falhas era ineficiente. Quando a geração de uma rodada falhava, pois o algoritmo não conseguia encontrar uma partida válida para completar a rodada, ele não tentava reverter apenas o último passo. Em vez disso, reiniciava a criação da rodada inteira, descartando todo o progresso parcial daquela rodada.

Por fim, o descarte completo por falha tornava o custo computacional proibitivo. O sistema possuía um limite de tentativas. Se uma única rodada não pudesse ser formada após N tentativas, todo o calendário em construção era descartado. A probabilidade de "adivinhar" a sequência correta de partidas para 19 rodadas (no caso de $n = 20$) sem falhar era baixa.

Concluiu-se que esta abordagem era computacionalmente ineficiente, pois a vasta maioria dos ciclos de processamento era gasta na geração de calendários inválidos e em embaralhamentos desnecessários. Essa limitação fundamental motivou a pesquisa e a adoção de métodos algorítmicos mais estruturados.

5.2 Abordagem intermediária: heurística de créditos

Após a constatação da inviabilidade do modelo de amostragem pseudo-aleatória, uma segunda abordagem foi desenvolvida, desta vez baseada em uma heurística[2] gulosa. "Um algoritmo guloso sempre faz a escolha que parece ser a melhor no momento em questão." (Cormen *et al.*, 2024, p. 295). A lógica central era pré-alocar "créditos" para cada time, calculado a partir da distância média do campeonato. O algoritmo, então, tentaria construir o calendário rodada por rodada, "gastando" esses créditos de forma equilibrada.

A cada passo da construção da rodada, o modelo selecionava um time prioritário e avaliava seus confrontos pendentes. Para cada adversário potencial, o algoritmo calculava o custo da viagem e o comparava com o saldo de créditos do time. A decisão era "gulosa", pois o modelo selecionava a partida que minimizava a diferença absoluta entre o custo da viagem e o saldo de créditos, ou seja, a viagem que melhor se "encaixava" no orçamento restante. Após a seleção, o custo daquela viagem era debitado dos créditos de ambos os times envolvidos.

Para mitigar o risco de o algoritmo atingir um estado em que o calendário não pudesse mais ser completado, uma rotina de simulação recursiva foi implementada. Antes de confirmar

[2] Uma heurística, no contexto da ciência da computação, é um método que, embora não garanta encontrar a solução ótima, busca encontrar uma solução "boa" ou "suficientemente próxima" em um tempo computacional viável.

uma partida, o modelo tentava simular a geração do campeonato inteiro a partir daquela decisão. Embora projetado para garantir a validade, esse mecanismo de verificação resultou em uma complexidade computacional relativamente alta.

A abordagem falhou em sua métrica principal. Os calendários gerados apresentavam uma amplitude logística pior do que a do modelo pseudo-aleatório. Este é um resultado clássico de heurísticas gulosas, que são "míopes": ao tomar a decisão localmente ótima, como gastar os créditos de forma ideal no momento, o algoritmo se colocava em posições insustentáveis no futuro. A teoria de algoritmos confirma que "Essa estratégia heurística nem sempre produz uma solução ótima" (Cormen *et al.*, 2024, p. 300). O desequilíbrio, ao invés de ser mitigado, era amplificado. Esta falha dupla justificou o abandono desta linha de desenvolvimento e a necessidade de uma arquitetura fundamentalmente diferente.

5.3 Arquitetura do modelo computacional final

As falhas das abordagens de amostragem pseudo-aleatória e da heurística gulosa, documentadas nas seções anteriores, demonstraram a necessidade de uma arquitetura fundamentalmente diferente. O modelo final, portanto, foi projetado como um sistema híbrido, estruturalmente sólido e mais viável computacionalmente.

Sua arquitetura resolve os problemas anteriores ao separar as responsabilidades em componentes distintos: a geração determinística de um calendário-base, a permutação do espaço de soluções para exploração, a otimização heurística do mando de campo e a exploração paralela em larga escala.

5.3.1 Fundação Determinística: O Método Circular

A principal falha dos modelos iniciais era a ausência de garantia estrutural. O modelo final resolve este problema ao adotar o Método Circular para a geração dos confrontos de cada rodada. Conforme detalhado na Fundamentação Teórica (Seção 2.4.2), este algoritmo constrói deterministicamente os $n/2$ pares de cada uma das $n-1$ rodadas do turno. A implementação em C deste algoritmo, utilizada no modelo final, é apresentada no Código-fonte B.

Ao implementar este método, a validade estrutural do calendário (garantindo que todas as equipes se enfrentem exatamente uma vez por turno) deixa de ser um resultado probabilístico e passa a ser uma garantia por construção. Isso elimina a necessidade de mecanismos de

recuperação de falhas ou simulações recursivas complexas, resolvendo o problema central das abordagens anteriores.

5.3.2 Exploração do Espaço de Soluções: Unranking de Permutação

O Método Circular, por si só, gera apenas um calendário-base para uma ordem fixa de equipes. Para que o modelo possa buscar uma solução otimizada, é necessário explorar o vasto espaço de soluções de $n!$ permutações de equipes.

O modelo implementa isso através do algoritmo de *unranking* de permutação via *factoradics*, também apresentado na Fundamentação Teórica (Seção 2.4.3). Conforme demonstrado no Código-fonte D, um único número inteiro, a "semente" (*seed*), é fornecido ao algoritmo. Esta semente gera deterministicamente uma permutação única da lista de equipes.

Esta lista permutada de equipes é então usada como entrada para o Método Circular (Seção 5.3.1), resultando em um calendário-base provavelmente distinto. Ao iterar por diferentes sementes, o modelo é capaz de amostrar e avaliar calendários estruturalmente distintos.

5.3.3 Otimização do Mando de Campo: Heurística de Equilíbrio

Uma vez que os pares de uma rodada são definidos (Seção 5.3.1) e as equipes permutadas (Seção 5.3.2), o modelo precisa tomar a decisão logisticamente mais importante: quem será o mandante.

Esta é a principal otimização do modelo e substitui a heurística gulosa de "créditos" (Seção 5.2), que se mostrou "míope" e ineficaz. A nova abordagem é uma heurística local de "equilíbrio", que avalia cada partida individualmente, visando manter todos os times o mais próximo possível da média de distância percorrida. Esta abordagem está alinhada com a metodologia definida no Capítulo 4, que previa uma heurística para definir os jogos em casa e fora e o controle de restrições básicas.

Para cada um dos $n/2$ jogos da rodada, o algoritmo segue um processo de decisão:

1. Verificação de Restrições: O modelo primeiro avalia restrições básicas do torneio. Conforme o código final, uma restrição de exemplo é o limite de partidas consecutivas em casa (ex: 3). Se uma equipe violar a restrição, ela é forçada a ser visitante.
2. Análise Heurística (Cenários): Se nenhuma restrição for violada, a heurística é ativada. O modelo utiliza os dados de distância percorrida por cada time, que são atualizados cumulativamente ao final de cada rodada. A partir desses valores, o modelo calcula a

média atual de distância do campeonato.

3. Simulação de Custos: O modelo simula os dois cenários possíveis (Equipe A como mandante; Equipe B como mandante) e calcula a nova distância total acumulada para ambas as equipes em cada cenário.
4. Decisão de Equilíbrio: O algoritmo então seleciona o cenário que minimiza a maior diferença de uma equipe em relação à média. Esta lógica busca o maior equilíbrio logístico.
5. Resolução de Empates: No caso de um empate, onde ambos os cenários de mando resultam na mesma penalidade máxima, a decisão é tomada por um mecanismo de *hash* determinístico, baseado na semente, para garantir a reprodutibilidade.

A lógica de decisão detalhada nos passos acima constitui a principal otimização do modelo e é aplicada a cada um dos $n/2$ jogos de cada rodada. O código-fonte completo desta implementação, utilizado no modelo computacional final, encontra-se no Apêndice C.

Após a definição de todos os mandos de uma rodada, os totais de distância de cada equipe são atualizados, garantindo que a heurística da rodada seguinte opere sobre dados cumulativos e precisos. A implementação desta função de atualização de dados, que também gerencia os estados de "último jogo" e "partidas seguidas em casa", encontra-se no Apêndice E.

5.3.4 Amostragem Paralelizada em Larga Escala

A combinação dos algoritmos de *unranking* (Seção 5.3.2) e do Método Circular (Seção 5.3.1) permite a geração determinística de um calendário-base para qualquer semente numérica. O espaço de soluções explorável é, portanto, o número de permutações de equipes ($n!$), que para $n = 20$ é um número da ordem de 2.4×10^{18}.

Uma busca exaustiva neste espaço é computacionalmente intratável. Conforme definido na Metodologia (Capítulo 4), a estratégia adotada é uma amostragem em larga escala. O modelo é projetado para gerar e avaliar um número massivo, mas finito, de sementes, buscando a semente que resulta na menor amplitude logística.

Para tornar esta busca viável em tempo hábil, o processo de amostragem foi paralelizado com a API OpenMP. Esta técnica, alinhada à metodologia, possibilita avaliar múltiplas soluções candidatas simultaneamente. A implementação segue uma lógica de "map-reduce" simples:

1. O volume total de sementes a ser testado é dividido entre todas as *threads*[3] de processamento disponíveis no sistema.
2. Cada *thread* executa, em paralelo, o processo completo de geração e avaliação em seu subconjunto de sementes (permutação, geração do calendário-base e aplicação da heurística de equilíbrio).
3. Cada *thread* armazena em memória o seu "melhor calendário local" (a semente que gerou a menor amplitude encontrada por ela).
4. Ao final da execução paralela, os resultados de todas as *threads* são comparados para se obter o "melhor calendário global", que é a solução final retornada pelo modelo.

Esta arquitetura híbrida final, que combina a solidez estrutural dos algoritmos determinísticos com a otimização da heurística de equilíbrio e a velocidade da paralelização, permite que o modelo explore eficientemente um vasto espaço de soluções.

5.4 Síntese do capítulo

Este capítulo apresentou a jornada de desenvolvimento do modelo computacional, partindo de abordagens iniciais de amostragem pseudo-aleatória e heurísticas gulosas, que se mostraram inviáveis ou ineficazes. As limitações identificadas justificaram a arquitetura híbrida final.

O modelo desenvolvido combina a garantia estrutural do Método Circular, a capacidade de exploração do *unranking* de permutação e uma heurística de equilíbrio local para otimizar o mando de campo. A paralelização com OpenMP torna a amostragem em larga escala computacionalmente viável. O modelo está, portanto, validado em sua construção e pronto para ser executado, com seus resultados analisados no capítulo subsequente.

[3] Uma *thread* (linha de execução) é a menor unidade de processamento que pode ser executada por um sistema operacional. No contexto desta paralelização, cada *thread* funciona como um 'trabalhador' independente, executando uma parte da tarefa de amostragem simultaneamente em um núcleo de CPU.

6 RESULTADOS

Este capítulo apresenta os resultados práticos obtidos pela execução do modelo computacional proposto, cuja arquitetura foi detalhada no Capítulo 5. O objetivo é quantificar a eficácia do modelo na geração de tabelas logisticamente equitativas, comparando-as diretamente com os calendários oficiais da CBF, que servem como grupo de controle, conforme estabelecido na Seção 4.2.4.

A análise é conduzida de acordo com as métricas de validação definidas na Seção 4.4. O foco principal é a amplitude métrica central da função objetivo apresentada na Seção 4.3.2, mas também são consideradas o desvio padrão e a média das distâncias totais percorridas, de modo a permitir uma avaliação abrangente do equilíbrio logístico.

Conforme o desenho experimental descrito na Seção 4.3, o modelo foi executado para gerar calendários otimizados para as temporadas de 2022, 2023 e 2024, utilizando a amostra de clubes definida na Seção 4.2.1. Os resultados são apresentados em tabelas e nos gráficos polares comparativos elaborados segundo a metodologia exposta na Seção 4.4.2.

É fundamental reiterar a delimitação do escopo estabelecida na Seção 4.4.3: esta comparação é realizada sob a perspectiva da otimização logística. O modelo proposto foca exclusivamente na minimização da disparidade de viagens, enquanto o calendário real da CBF atende a múltiplas restrições operacionais e comerciais que não fazem parte deste trabalho.

6.1 Análise da temporada 2022

Iniciando a análise, a temporada de 2022 é avaliada. A matriz de distâncias geodésicas para esta temporada, calculada conforme a metodologia, encontra-se no Apêndice H.

Para a análise do grupo de controle, a tabela oficial de jogos do Campeonato Brasileiro Série A 2022 foi extraída do site oficial da CBF[4]. Conforme a ressalva metodológica estabelecida na Seção 4.2.4, todos os cálculos de distância para o calendário oficial utilizaram a mesma matriz de distâncias do modelo proposto, garantindo uma comparação justa dos agendamentos.

[4] Disponível em: <https://www.cbf.com.br/futebol-brasileiro/tabelas/campeonato-brasileiro/serie-a/2022>. Acesso em: 03/11/2025.

6.1.1 Comparativo de Métricas Logísticas (2022)

O modelo computacional foi executado por 1×10^9 (um bilhão) de iterações para encontrar a solução de menor amplitude. A Tabela 1 apresenta o comparativo das métricas logísticas centrais.

Tabela 1 – Comparativo das métricas logísticas (Temporada 2022)

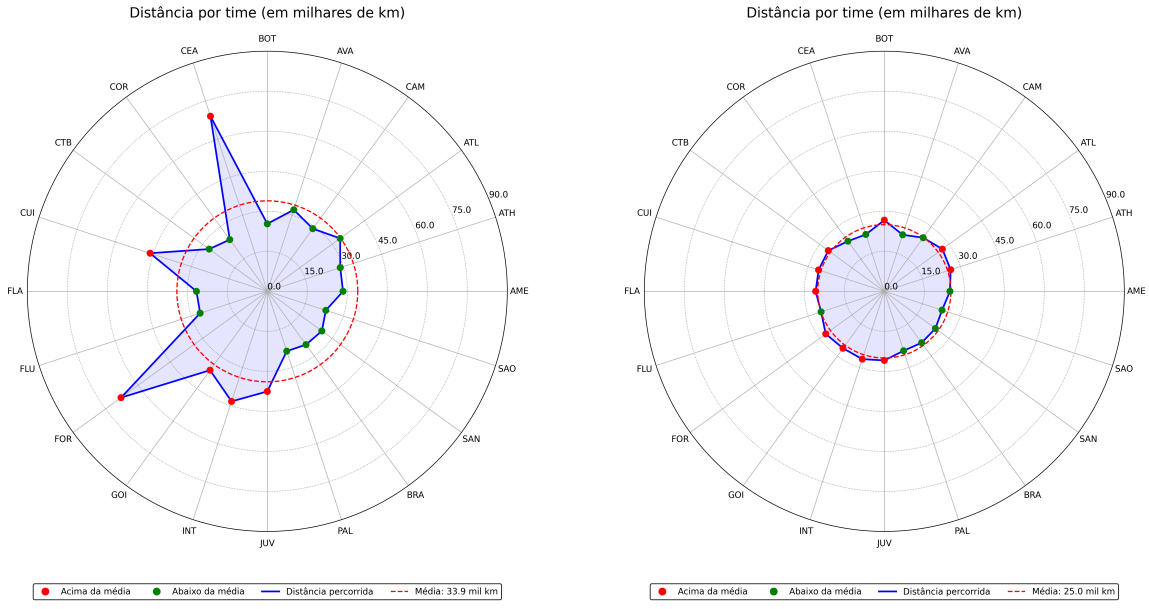
Métrica	Calendário Oficial (CBF)	Calendário Proposto (Modelo)
Distância Total (km)	678.508	499.617
Distância Média (km)	33.925	24.981
Desvio Padrão (km)	13.116	1.554
Amplitude (km)	**45.940**	**4.862**

Fonte: Autoria própria (2025).

A Tabela 1 demonstra o impacto da otimização. O modelo proposto não apenas reduziu a distância total percorrida em 178.891 km (uma redução de 26,4%), mas, mais importante, atingiu a função objetivo (Seção 4.3.2) ao reduzir a amplitude em 89,4%. O desvio padrão também caiu de 13.116 km para 1.554 km, indicando que as distâncias percorridas pelos times são muito mais próximas umas das outras.

A Figura 3 ilustra visualmente essa disparidade, conforme a metodologia de gráficos polares descrita na Seção 4.4.2.

Figura 3 – Gráfico polar comparativo das distâncias percorridas (2022)



(a) Calendário Oficial (CBF)

(b) Calendário Proposto (Modelo)

Fonte: Autoria própria (2025).

Tabela 2 – Distâncias totais percorridas por clube (Temporada 2022)

Clube (Sigla)	Distância CBF (km)	Distância Modelo (km)
América Mineiro (AME)	28.396	24.617
Athletico Paranaense (ATH)	28.762	26.164
Atlético Goianiense (ATL)	33.757	26.909
Atlético Mineiro (CAM)	29.030	24.838
Avaí (AVA)	32.136	22.290
Botafogo (BOT)	25.301	26.678
Ceará (CEA)	69.019	22.425
Corinthians (COR)	23.948	23.225
Coritiba (CTB)	26.948	25.972
Cuiabá (CUI)	46.223	25.925
Flamengo (FLA)	26.579	25.754
Fluminense (FLU)	26.488	24.937
Fortaleza (FOR)	67.805	27.152
Goiás (GOI)	36.543	26.376
Internacional (INT)	43.387	26.731
Juventude (JUV)	37.529	25.904
Palmeiras (PAL)	23.592	23.431
Red Bull Bragantino (BRA)	24.695	23.820
Santos (SAN)	25.291	23.691
São Paulo (SAO)	23.079	22.778
Max (km)	**69.019**	**27.152**
Min (km)	**23.079**	**22.290**

Fonte: Autoria própria (2025).

A Tabela 2 detalha o impacto desta otimização em cada um dos 20 clubes. A eficácia do modelo em corrigir disparidades extremas é evidente. No calendário da CBF, o Ceará (CEA) foi o time que mais viajou, percorrendo 69.019 km. No calendário proposto, o mesmo time (CEA) viajaria apenas 22.425 km, uma redução de 67,5%.

A amplitude final de apenas 4.862 km no modelo proposto (entre o máximo de 27.152 km do Fortaleza e o mínimo de 22.290 km do Avaí) comprova o sucesso da abordagem de equilíbrio.

6.2 Análise da temporada 2023

Seguindo a análise, a temporada de 2023 é avaliada. A matriz de distâncias geodésicas para esta temporada, calculada conforme a metodologia, encontra-se no Apêndice I.

Para a análise do grupo de controle, a tabela oficial de jogos do Campeonato Brasileiro Série A 2023 foi extraída do site oficial da CBF[5]. Conforme a ressalva metodológica estabelecida

[5] Disponível em: <https://www.cbf.com.br/futebol-brasileiro/tabelas/campeonato-brasileiro/serie-a/2023>. Acesso em: 02/11/2025.

na Seção 4.2.4, todos os cálculos de distância para o calendário oficial utilizaram a mesma matriz de distâncias do modelo proposto, garantindo uma comparação justa dos agendamentos.

6.2.1 Comparativo de Métricas Logísticas (2023)

O modelo computacional foi executado por 1×10^9 (um bilhão) de iterações para encontrar a solução de menor amplitude. A Tabela 3 apresenta o comparativo das métricas logísticas centrais.

Tabela 3 – Comparativo das métricas logísticas (Temporada 2023)

Métrica	Calendário Oficial (CBF)	Calendário Proposto (Modelo)
Distância Total (km)	610.224	405.013
Distância Média (km)	30.511	20.251
Desvio Padrão (km)	12.824	1.417
Amplitude (km)	**49.116**	**5.910**

Fonte: Autoria própria (2025).

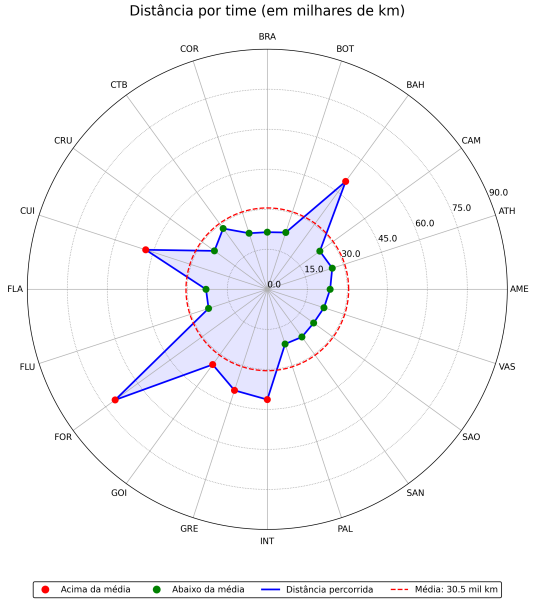
A Tabela 3 demonstra o impacto da otimização. O modelo proposto não apenas reduziu a distância total percorrida em mais de 205.000 km (uma redução de 33,6%), mas, mais importante, atingiu a função objetivo (Seção 4.3.2) ao reduzir a amplitude em 88,0%. O desvio padrão também caiu de 12.824 km para 1.417 km, indicando que as distâncias percorridas pelos times são muito mais próximas umas das outras.

A Figura 4 ilustra visualmente essa disparidade, conforme a metodologia de gráficos polares descrita na Seção 4.4.2.

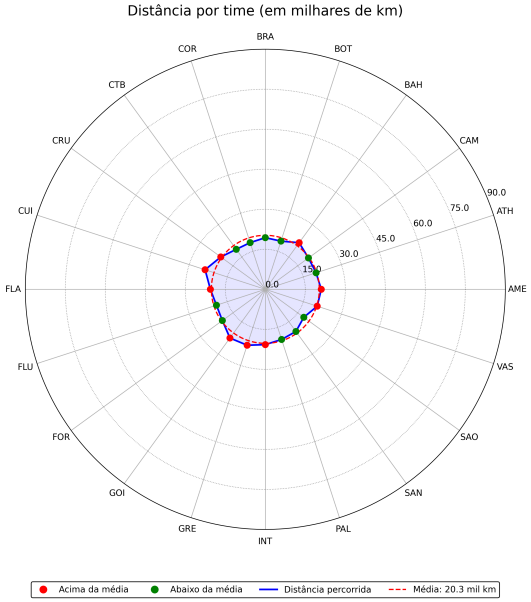
A Tabela 4 detalha o impacto desta otimização em cada um dos 20 clubes. A eficácia do modelo em corrigir disparidades extremas é evidente. No calendário da CBF, o Fortaleza (FOR) foi o time que mais viajou, percorrendo 70.537 km. No calendário proposto, o mesmo time (FOR) viajaria apenas 19.914 km, uma redução de 71,8%.

A amplitude final de apenas 5.910 km no modelo proposto (entre o máximo de 23.752 km do Cuiabá e o mínimo de 17.842 km do São Paulo) comprova o sucesso da abordagem de equilíbrio.

Figura 4 – Gráfico polar comparativo das distâncias percorridas (2023)



(a) Calendário Oficial (CBF)



(b) Calendário Proposto (Modelo)

Fonte: Autoria própria (2025).

Tabela 4 – Distâncias totais percorridas por clube (Temporada 2023)

Clube (Sigla)	Distância CBF (km)	Distância Modelo (km)
América-MG (AME)	23.557	20.987
Athletico-PR (ATH)	25.604	20.024
Atlético-MG (CAM)	24.348	19.995
Bahia (BAH)	49.988	21.595
Botafogo (BOT)	22.400	18.997
Bragantino (BRA)	21.421	19.357
Corinthians (COR)	22.109	18.410
Coritiba (CTB)	28.177	18.536
Cruzeiro (CRU)	24.526	20.635
Cuiabá (CUI)	47.946	23.752
Flamengo (FLA)	22.992	20.596
Fluminense (FLU)	23.165	19.272
Fortaleza (FOR)	70.537	19.914
Goiás (GOI)	34.858	22.568
Grêmio (GRE)	39.824	22.039
Internacional (INT)	41.328	20.733
Palmeiras (PAL)	21.563	19.768
Santos (SAN)	22.073	19.532
São Paulo (SAO)	21.473	17.842
Vasco da Gama (VAS)	22.335	20.461
Max (km)	**70.537**	**23.752**
Min (km)	**21.421**	**17.842**

Fonte: Autoria própria (2025).

6.3 Análise da temporada 2024

A temporada de 2024 foi utilizada como principal caso de estudo para a validação do modelo. A matriz de distâncias geodésicas para esta temporada, calculada conforme a metodologia, encontra-se no Apêndice J.

Para a análise do grupo de controle, a tabela oficial de jogos do Campeonato Brasileiro Série A 2024 foi extraída do site oficial da CBF[6]. Conforme a ressalva metodológica estabelecida na Seção 4.2.4, todos os cálculos de distância para o calendário oficial utilizaram a mesma matriz de distâncias do modelo proposto, garantindo uma comparação justa dos agendamentos.

6.3.1 Comparativo de Métricas Logísticas (2024)

O modelo computacional foi executado por 1×10^9 (um bilhão) de iterações para encontrar a solução de menor amplitude. A Tabela 5 apresenta o comparativo das métricas logísticas centrais.

Tabela 5 – Comparativo das métricas logísticas (Temporada 2024)

Métrica	Calendário Oficial (CBF)	Calendário Proposto (Modelo)
Distância Total (km)	708.643	510.593
Distância Média (km)	35.432	25.530
Desvio Padrão (km)	12.417	1.971
Amplitude (km)	**50.438**	**6.085**

Fonte: Autoria própria (2025).

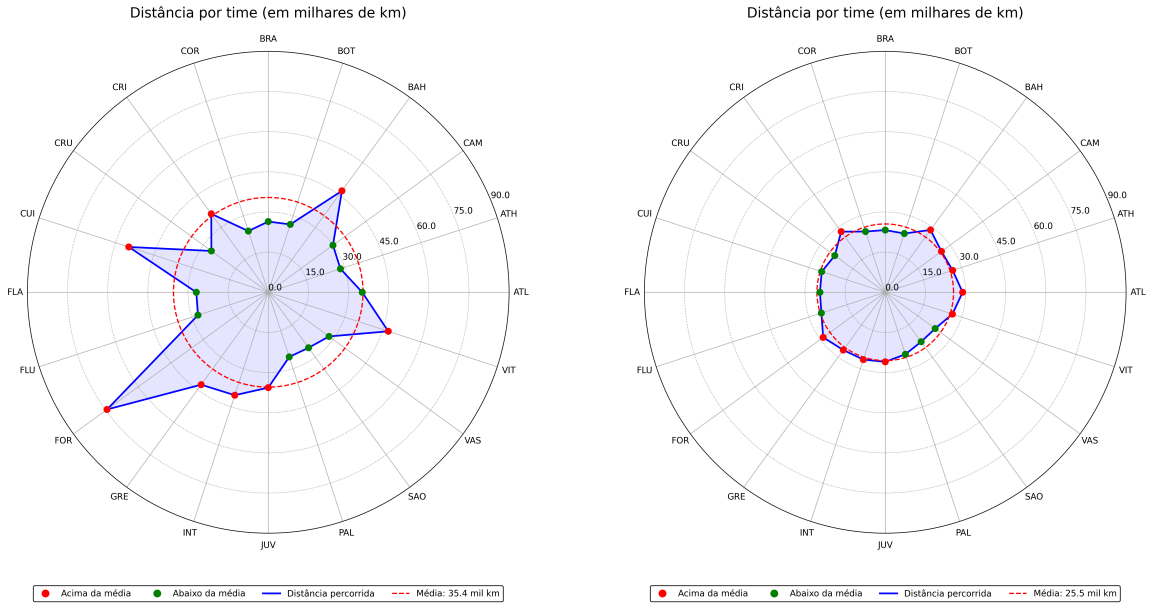
A Tabela 5 demonstra o impacto da otimização. O modelo proposto não apenas reduziu a distância total percorrida em quase 200.000 km (uma redução de 28%), mas, mais importante, atingiu a função objetivo (Seção 4.3.2) ao reduzir a amplitude em 87,9%. O desvio padrão também caiu de 12.417 km para 1.971 km, indicando que as distâncias percorridas pelos times são muito mais próximas umas das outras.

A Tabela 5 demonstra que o modelo reduziu significativamente a distância total e a amplitude das viagens, tornando as distâncias percorridas pelos times muito mais equilibradas.

A Figura 5 ilustra visualmente essa disparidade, e a Tabela 6 detalha o impacto em cada clube, evidenciando, por exemplo, que o Fortaleza (FOR) reduziu suas viagens de 74.486 km para 28.672 km.

[6] Disponível em: <https://www.cbf.com.br/futebol-brasileiro/tabelas/campeonato-brasileiro/serie-a/2024>. Acesso em: 01 nov.2025.

Figura 5 – Gráfico polar comparativo das distâncias percorridas (2024)



(a) Calendário Oficial (CBF)

(b) Calendário Proposto (Modelo)

Fonte: Autoria própria (2025).

Tabela 6 – Distâncias totais percorridas por clube (Temporada 2024)

Clube (Sigla)	Distância CBF (km)	Distância Modelo (km)
Atlético-GO (ATL)	35.132	28.923
Athletico-PR (ATH)	28.376	26.502
Atlético-MG (CAM)	29.890	25.969
Bahia (BAH)	46.844	28.782
Botafogo (BOT)	26.662	23.084
Bragantino (BRA)	26.417	23.242
Corinthians (COR)	24.048	23.859
Criciúma (CRI)	36.220	28.011
Cruzeiro (CRU)	26.271	23.318
Cuiabá (CUI)	54.796	24.973
Flamengo (FLA)	26.886	24.418
Fluminense (FLU)	27.618	25.101
Fortaleza (FOR)	74.486	28.672
Grêmio (GRE)	42.670	26.667
Internacional (INT)	40.435	26.445
Juventude (JUV)	35.626	25.987
Palmeiras (PAL)	25.396	24.356
São Paulo (SAO)	25.627	22.838
Vasco da Gama (VAS)	28.079	23.057
Vitória (VIT)	47.164	26.389
Max (km)	**74.486**	**28.923**
Min (km)	**24.048**	**22.838**

Fonte: Autoria própria (2025).

7 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho propôs-se a desenvolver e validar um modelo computacional de alta performance capaz de gerar tabelas para campeonatos de pontos corridos com um foco explícito na equidade logística. O objetivo central era mitigar as disparidades extremas de distâncias de viagem, um problema notório em países de dimensão continental como o Brasil, onde a logística pode influenciar a isonomia da competição.

Para enfrentar a explosão combinatória do problema, que para 20 equipes aproxima-se de $2,43 \times 10^{18}$ permutações possíveis, foi desenvolvida uma arquitetura híbrida. Esta arquitetura combinou a garantia estrutural do Método Circular com a exploração sistemática do espaço de soluções via unranking de permutação e uma heurística de equilíbrio local para otimização do mando de campo. A implementação em linguagem C com paralelização via OpenMP permitiu a amostragem e avaliação de um bilhão (1×10^9) de calendários candidatos por temporada, demonstrando a viabilidade da abordagem.

Os resultados obtidos demonstram que o modelo atingiu seu objetivo principal de forma conclusiva. Ao comparar os calendários gerados com as tabelas oficiais da CBF para as temporadas 2022, 2023 e 2024, o modelo proposto alcançou reduções drásticas na principal métrica de desigualdade, a **amplitude** (diferença entre a equipe que mais viaja e a que menos viaja). As reduções de amplitude foram de **89,4%** para a temporada 2022, **88,0%** para 2023 e **87,9%** para 2024.

Além disso, o modelo gerou calendários mais homogêneos, como evidenciado pela queda acentuada no desvio padrão (e.g., de 13.116 km para 1.554 km em 2022), e mais eficientes, reduzindo a distância total percorrida por todas as equipes em mais de 178.000 km em 2022, 205.000 km em 2023 e 198.000 km em 2024. Conclui-se, portanto, que a metodologia de amostragem paralela de larga escala, guiada por uma heurística de equilíbrio, é uma estratégia eficaz para a geração de tabelas esportivas logisticamente mais justas.

Reconhece-se, contudo, as limitações deste estudo, conforme delimitado na metodologia. O modelo foi otimizado exclusivamente para a equidade de distâncias. O calendário oficial, por outro lado, está sujeito a um conjunto complexo de restrições não documentadas, como demandas comerciais, de segurança pública e de logística de transmissão. Fatores que impactam o desgaste do atleta, como viagens através de fusos horários, tempo de recuperação em horas entre as partidas e a malha aérea (forçando conexões), não fizeram parte do escopo deste modelo.

Essas limitações, no entanto, abrem caminhos claros para trabalhos futuros. Sugere-se a

evolução do modelo para um sistema de otimização multiobjetivo, que poderia buscar balancear não apenas a distância, mas também:

- **Incorporar restrições da malha aérea real**, substituindo a distância geodésica (Haversine) por rotas de voo práticas, possivelmente penalizando itinerários que exigem conexões.
- **Adicionar parâmetros de entrada para restrições complexas**, transformando o modelo em uma ferramenta mais robusta para os organizadores de torneios. Isso permitiria, por exemplo, garantir que "clássicos" regionais ocorram em uma rodada comum, evitar que times da mesma cidade joguem em casa na mesma rodada, ou atender a datas reservadas para eventos de outras competições.
- **Otimizar a busca pelo calendário-base:** Em vez da amostragem de larga escala do espaço de permutações ($N!$), aplicar meta-heurísticas para buscar ativamente a permutação de equipes que gere a solução de menor amplitude.
- **Explorar outras meta-heurísticas** para a otimização dos mandos de campo, comparando-as com a heurística de equilíbrio desenvolvida neste trabalho, a fim de encontrar soluções "quase-ótimas" de forma ainda mais eficiente.

Este TCC demonstrou ser computacionalmente viável gerar calendários esportivos significativamente mais equitativos. A aplicação de tais modelos poderia contribuir para um campeonato mais justo, mitigando a desvantagem competitiva imposta pela geografia continental do país.

REFERÊNCIAS

ARAUJO, M. N. d. **Análise de dados estatísticos sobre o Campeonato Brasileiro de Futebol**. Monografia (Graduação) — Universidade Federal Rural do Semi-Árido, Pau dos Ferros, 2024.

ARCAS, G. I. *et al.* Edge offloading in smart grid. **Smart Cities**, v. 7, n. 1, p. 680–711, 2024.

BULCK, D. V.; GOOSSENS, D. The international timetabling competition on sports timetabling (itc2021). **European Journal of Operational Research**, v. 308, n. 3, p. 1249–1267, 2023.

CHAREST, J. *et al.* Impacts of travel distance and travel direction on back-to-back games in the national basketball association. **Journal of Clinical Sleep Medicine**, v. 17, n. 11, p. 2269–2274, 2021.

CORMEN, T. H. *et al.* **Algoritmos: teoria e prática**. 4. ed. Rio de Janeiro: GEN LTC, 2024.

GENITRINI, A.; PéPIN, M. Lexicographic unranking of combinations revisited. **Algorithms**, v. 14, n. 3, 2021.

GOERIGK, M. *et al.* Solving the traveling tournament problem by packing three-vertex paths. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2014. v. 28, n. 1.

JANUARIO, T.; URRUTIA, S. A new neighborhood structure for round robin scheduling problems. **Computers & Operations Research**, v. 70, p. 127–139, 2016.

LAMBERS, R.; ROTHUIZEN, L.; SPIEKSMA, F. C. R. How to schedule the volleyball nations league. **Journal of Sports Analytics**, v. 9, n. 2, p. 157–169, 2023.

LAMBRECHTS, E. *et al.* Round-robin tournaments generated by the circle method have maximum carry-over. **Mathematical Programming**, v. 172, 2017.

LEATHERWOOD, W. E.; DRAGOO, J. L. Effect of airline travel on performance: a review of the literature. **British Journal of Sports Medicine**, v. 47, n. 9, p. 561–567, 2013.

OSICKA, O.; GUAJARDO, M. Fair travel distances in tournament schedules: a cooperative game theory approach. **Sports Economics Review**, v. 2, p. 100011, 2023.

RIBEIRO, C. C.; URRUTIA, S. Scheduling the brazilian soccer tournament: solution approach and practice. **Interfaces**, v. 42, n. 3, p. 260–272, 2011.

TOLE, K.; MILANI, M.; MWAKONDO, F. Particle swarm algorithm for improved handling of the mirrored traveling tournament problem. **Tehnički Vjesnik**, v. 28, n. 5, 2021.

WINARNO, E.; HADIKURNIAWATI, W.; ROSSO, R. N. Location based service for presence system using haversine method. In: **Proceedings of the International Conference on Innovative and Creative Information Technology**. [S.l.: s.n.], 2017. p. 1–4.

APÊNDICE A – IMPLEMENTAÇÃO DA FÓRMULA DE HAVERSINE

A função calcular_distancia, apresentada no Bloco de Código a seguir, foi desenvolvida na linguagem Python e utiliza a biblioteca math para realizar as operações trigonométricas.

Essa função corresponde a uma implementação computacional das equações matemáticas apresentadas por (Winarno; Hadikurniawati; Rosso, 2017). e constitui o núcleo do cálculo de distâncias empregado neste trabalho, conforme referenciado na Seção 2.4.

```
1 def calcular_distancia(lat1, lon1, lat2, lon2):
2     raio_terra = 6371.0
3     lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2
    , lon2])
4     dlat = lat2 - lat1
5     dlon = lon2 - lon1
6     a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) *
    math.sin(dlon/2)**2
7     c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
8     return round(raio_terra * c)
```

APÊNDICE B – IMPLEMENTAÇÃO DO MÉTODO CIRCULAR (ROUND-ROBIN)

O núcleo do código responsável por gerar os pares de confrontos em cada rodada, conforme referenciado na Seção 2.4.2, é apresentado no Bloco de Código a seguir.

```
int idx1, idx2;

if (i == 0)
{
    idx1 = rodada % (qnt_times - 1);
    idx2 = qnt_times - 1;
}
else
{
    idx1 = (rodada + i) % (qnt_times - 1);
    idx2 = (rodada + qnt_times - 1 - i) % (qnt_times - 1);
}
```

Fonte: Autoria própria (2025).

APÊNDICE C – IMPLEMENTAÇÃO DA HEURÍSTICA DE EQUILÍBRIO

O bloco de código a seguir apresenta o trecho de implementação em C da lógica de decisão de mando de campo, conforme discutido na Seção 5.3.3. Este código é executado pela função gera_rodada (apresentada integralmente no Apêndice G) para cada par de equipes em cada rodada.

```
1
2  if (partidas_time1 > 3 && partidas_time2 <= 3)
3  {
4      mandante = time2;
5      visitante = time1;
6  }
7  else if (partidas_time2 > 3 && partidas_time1 <= 3)
8  {
9      mandante = time1;
10     visitante = time2;
11 }
12 else
13 {
14     // 3. Simulacao de Custos
15     int ultimo_jogo_time1 = calendar_data->
       ultimo_jogo_primeiro_turno[time1];
16     int ultimo_jogo_time2 = calendar_data->
       ultimo_jogo_primeiro_turno[time2];
17
18     int ultimo_jogo_time1_segundo_turno = calendar_data->
       ultimo_jogo_segundo_turno[time1];
19     int ultimo_jogo_time2_segundo_turno = calendar_data->
       ultimo_jogo_segundo_turno[time2];
20
21     int valor_caso_time1_1 = calendar_data->distancias[time1] +
       calendar_data->matriz_distancia[ultimo_jogo_time1][time1];
22     int valor_caso_time1_2 = calendar_data->distancias[time2] +
       calendar_data->matriz_distancia[ultimo_jogo_time2][time1];
```

```
23
24      int valor_caso_time2_1 = calendar_data->distancias[time1] +
        calendar_data->matriz_distancia[ultimo_jogo_time1][time2];
25      int valor_caso_time2_2 = calendar_data->distancias[time2] +
        calendar_data->matriz_distancia[ultimo_jogo_time2][time2];
26
27      if (ultimo_jogo_time2_segundo_turno != -1 &&
        ultimo_jogo_time2_segundo_turno != -1)
28      {
29          valor_caso_time1_1 += calendar_data->matriz_distancia[
        ultimo_jogo_time1_segundo_turno][time1];
30          valor_caso_time1_2 += calendar_data->matriz_distancia[
        ultimo_jogo_time2_segundo_turno][time1];
31          valor_caso_time2_1 += calendar_data->matriz_distancia[
        ultimo_jogo_time1_segundo_turno][time2];
32          valor_caso_time2_2 += calendar_data->matriz_distancia[
        ultimo_jogo_time2_segundo_turno][time2];
33      }
34
35      // 4. Decisao de Equilibrio
36      int aux1, aux2;
37      aux1 = abs(valor_caso_time1_1 - media_distancias);
38      aux2 = abs(valor_caso_time1_2 - media_distancias);
39      int maior_diff_case_time1 = aux1 >= aux2 ? aux1 : aux2;
40
41      aux1 = abs(valor_caso_time2_1 - media_distancias);
42      aux2 = abs(valor_caso_time2_2 - media_distancias);
43      int maior_diff_case_time2 = aux1 >= aux2 ? aux1 : aux2;
44
45      if (maior_diff_case_time1 < maior_diff_case_time2)
46      {
47          mandante = time1;
48          visitante = time2;
49      }
```

```
50      else if (maior_diff_case_time2 < maior_diff_case_time1)
51      {
52          mandante = time2;
53          visitante = time1;
54      }
55      else
56      {
57          // 5. Resolucao de Empates
58          Prng_ctx rng;
59          uint32_t combined_seed = calendar_data->seed_calendario;
60          combined_seed ^= rodada * 0x9E3779B9;
61          combined_seed ^= (time1 * 0x85EBCA6B) ^ (time2 * 0
    xC2B2AE35);
62          prng_init(&rng, combined_seed);
63          int hash = prng_next(&rng) & 1;
64
65          mandante = hash ? time2 : time1;
66          visitante = hash ? time1 : time2;
67      }
68  }
```

Fonte: Autoria própria (2025).

APÊNDICE D – IMPLEMENTAÇÃO DO *UNRANKING* DE PERMUTAÇÃO

O Bloco de Código a seguir apresenta a função em C que implementa o algoritmo de *unranking* discutido na Seção 2.4.3. Esta função permite gerar permutações determinísticas a partir de uma *seed*.

```c
void seed_para_permutacao(int *saida, int n, unsigned long long
    seed)
{
     int disponiveis[n];
     for (int i = 0; i < n; i++)
     {
          disponiveis[i] = i;
     }
     for (int i = 0; i < n; i++)
     {
          int items_restantes = n - i;
          unsigned long long f = fatoriais[items_restantes - 1];
          int index = seed / f;
          seed %= f;

          saida[i] = disponiveis[index];

          if (index < items_restantes - 1)
          {
               memmove(&disponiveis[index], &disponiveis[index +
    1],(items_restantes - index - 1) * sizeof(int));
          }
     }
}
```

Fonte: Autoria própria (2025).

APÊNDICE E – IMPLEMENTAÇÃO DA ATUALIZAÇÃO CUMULATIVA

Conforme discutido na Seção 5.3.3, para que a heurística de equilíbrio funcione, ela precisa ter acesso ao estado atual do campeonato. A função atualiza_dados, apresentada no Bloco de Código a seguir, é chamada ao final de cada rodada pela função gera_rodada (Apêndice G).

Esta função é responsável por atualizar cumulativamente as distâncias percorridas, registrar a localização do último jogo de cada time e gerenciar o contador de partidas seguidas em casa.

```
1  void atualiza_dados(int rodada, int (*jogos_da_rodada)[2],
       Calendar_data *dados_calendario)
2  {
3      int quantidade_jogos = dados_calendario->qnt_times / 2;
4
5      for (int jogo = 0; jogo < quantidade_jogos; jogo++)
6      {
7          int time_mandante = jogos_da_rodada[jogo][0];
8          int time_visitante = jogos_da_rodada[jogo][1];
9
10         // Busca o local do ultimo jogo de cada time
11         int ult_jogo_mandante_t1 = dados_calendario->
       ultimo_jogo_primeiro_turno[time_mandante];
12         int ult_jogo_visitante_t1 = dados_calendario->
       ultimo_jogo_primeiro_turno[time_visitante];
13         int ult_jogo_mandante_t2 = dados_calendario->
       ultimo_jogo_segundo_turno[time_mandante];
14         int ult_jogo_visitante_t2 = dados_calendario->
       ultimo_jogo_segundo_turno[time_visitante];
15
16         // Calcula a distancia da viagem desta rodada (Turno 1)
17         int dist_mandante = dados_calendario->matriz_distancia[
       ult_jogo_mandante_t1][time_mandante];
18         int dist_visitante = dados_calendario->matriz_distancia[
       ult_jogo_visitante_t1][time_mandante];
19
```

```
20          // Adiciona a distancia da viagem (Turno 2), se
    aplicavel
21          if (ult_jogo_mandante_t2 != -1)
22          {
23              dist_mandante += dados_calendario->matriz_distancia[
    ult_jogo_mandante_t2][time_visitante];
24          }
25          if (ult_jogo_visitante_t2 != -1)
26          {
27              dist_visitante += dados_calendario->matriz_distancia
    [ult_jogo_visitante_t2][time_visitante];
28          }
29
30          // Atualiza o total cumulativo de distancias
31          dados_calendario->distancias[time_mandante] +=
    dist_mandante;
32          dados_calendario->distancias[time_visitante] +=
    dist_visitante;
33
34          // Atualiza o estado para a proxima rodada (Turno 1)
35          dados_calendario->ultimo_jogo_primeiro_turno[
    time_mandante] = time_mandante;
36          dados_calendario->ultimo_jogo_primeiro_turno[
    time_visitante] = time_mandante;
37
38          // Atualiza o estado para a proxima rodada (Turno 2)
39          dados_calendario->ultimo_jogo_segundo_turno[
    time_mandante] = time_visitante;
40          dados_calendario->ultimo_jogo_segundo_turno[
    time_visitante] = time_visitante;
41
42          // Atualiza restricoes
43          dados_calendario->partidas_seguidas_em_casa[
    time_mandante]++;
```

```
44          dados_calendario->partidas_seguidas_em_casa[
    time_visitante] = 0;
45
46          // Logica de transicao entre turnos
47          if (rodada == (dados_calendario->qnt_times - 2))
48          {
49              ult_jogo_mandante_t1 = dados_calendario->
    ultimo_jogo_primeiro_turno[time_mandante];
50              ult_jogo_visitante_t1 = dados_calendario->
    ultimo_jogo_primeiro_turno[time_visitante];
51
52              int casa_t2_r1_mandante = -1;
53              int casa_t2_r1_visitante = -1;
54
55              for (int i = 0; i < quantidade_jogos; i++)
56              {
57                  if (dados_calendario->campeonato[rodada + 1][i].
    id_mandante == time_mandante ||
58                      dados_calendario->campeonato[rodada + 1][i].
    id_visitante == time_mandante)
59                  {
60                      casa_t2_r1_mandante = dados_calendario->
    campeonato[rodada + 1][i].id_mandante;
61                  }
62                  if (dados_calendario->campeonato[rodada + 1][i].
    id_mandante == time_visitante ||
63                      dados_calendario->campeonato[rodada + 1][i].
    id_visitante == time_visitante)
64                  {
65                      casa_t2_r1_visitante = dados_calendario->
    campeonato[rodada + 1][i].id_mandante;
66                  }
67              }
68
```

```
69              if (casa_t2_r1_mandante != -1 &&
    casa_t2_r1_visitante != -1)
70              {
71                  dados_calendario->distancias[time_mandante] +=
72                      dados_calendario->matriz_distancia[
    ult_jogo_mandante_t1][casa_t2_r1_mandante];
73                  dados_calendario->distancias[time_visitante] +=
74                      dados_calendario->matriz_distancia[
    ult_jogo_visitante_t1][casa_t2_r1_visitante];
75              }
76          }
77      }
78  }
```

Fonte: Autoria própria (2025).

APÊNDICE F – FUNÇÃO PRINCIPAL DE GERAÇÃO PARALELA

Conforme discutido na Seção 5.3.4, a amostragem em larga escala é viabilizada pela paralelização com OpenMP. A função gerar_calendario_definitivo, apresentada no Bloco de Código seguir, é a função principal do modelo computacional.

Esta função é responsável por inicializar a região paralela, dividir o espaço de sementes entre as *threads*, gerenciar a busca local de cada *thread* e, ao final, coletar os resultados para determinar o melhor calendário global. Ela chama a função gera_calendario_aux (Apêndice G) para cada semente.

```
1  Calendario *gerar_calendario_definitivo(int qnt_times,
       Calendario *calendario,
2                                          int **matriz_distancia,
3                                          uint64_t
       num_calendarios_gerar)
4  {
5      if (qnt_times <= 0 || !calendario || !matriz_distancia ||
       num_calendarios_gerar == 0)
6          return NULL;
7
8      const uint64_t max_calendars = FATORIAL_20;
9      int n_threads = omp_get_max_threads();
10
11     uint64_t melhor_seed_thread[n_threads];
12     int melhor_amplitude_thread[n_threads];
13     volatile int flag_erro = 0;
14
15     for (int t = 0; t < n_threads; t++)
16         melhor_amplitude_thread[t] = INT32_MAX;
17
18 #pragma omp parallel
19     {
20         int tid = omp_get_thread_num();
21         Calendar_data *calendar_data = alocar_calendario_dados(
       qnt_times, 0);
```

```
22
23          if (!calendar_data) {
24  #pragma omp atomic write
25              flag_erro = 1;
26          }
27
28          if (flag_erro == 0)
29          {
30              calendar_data->matriz_distancia = matriz_distancia;
31
32              // Divide o trabalho entre as threads
33              uint64_t calendars_per_thread = (
    num_calendarios_gerar + n_threads - 1) / n_threads;
34              uint64_t start_seed = tid * calendars_per_thread;
35              uint64_t end_seed = start_seed +
    calendars_per_thread;
36              if (end_seed > max_calendars)
37                  end_seed = max_calendars;
38
39              int melhor_amplitude_local = INT32_MAX;
40              uint64_t melhor_seed_local = 0;
41
42              // Loop de busca de cada thread
43              for (uint64_t seed = start_seed; seed < end_seed &&
    flag_erro == 0; seed++)
44              {
45                  seed_para_permutacao(calendar_data->times,
    qnt_times, seed);
46                  calendar_data->seed_calendario = seed;
47
48                  if (gera_calendario_aux(calendar_data,
    melhor_amplitude_local))
49                  {
50                      if (calendar_data->amplitude <
```

```
    melhor_amplitude_local)
51                  {
52                      melhor_amplitude_local = calendar_data->
    amplitude;
53                      melhor_seed_local = seed;
54                  }
55              }
56              resetar_calendario_dados(calendar_data);
57          }
58
59          melhor_seed_thread[tid] = melhor_seed_local;
60          melhor_amplitude_thread[tid] =
    melhor_amplitude_local;
61          liberar_calendario_dados(calendar_data);
62      }
63   }
64
65   if (flag_erro) return NULL;
66
67   // Reducao: Determina o melhor global
68   uint64_t melhor_seed_global = 0;
69   int melhor_amplitude_global = INT32_MAX;
70   for (int t = 0; t < n_threads; t++)
71   {
72       if (melhor_amplitude_thread[t] < melhor_amplitude_global
    )
73       {
74           melhor_amplitude_global = melhor_amplitude_thread[t
    ];
75           melhor_seed_global = melhor_seed_thread[t];
76       }
77   }
78
79   // Gera o calendario final com a melhor seed
```

```
80      if (melhor_amplitude_global < INT32_MAX)
81      {
82          Calendar_data *calendar_data_final =
    alocar_calendario_dados(qnt_times, melhor_seed_global);
83          if (calendar_data_final)
84          {
85              calendar_data_final->matriz_distancia =
    matriz_distancia;
86              seed_para_permutacao(calendar_data_final->times,
    qnt_times, melhor_seed_global);
87              if (gera_calendario_aux(calendar_data_final,
    INT32_MAX))
88              {
89                  copia_calendario(calendario, calendar_data_final
    );
90              }
91              liberar_calendario_dados(calendar_data_final);
92          }
93      }
94
95      return calendario;
96  }
```

Fonte: Autoria própria (2025).

APÊNDICE G – FUNÇÕES AUXILIARES DE GERAÇÃO DE CALENDÁRIO

Este apêndice detalha as funções auxiliares que são chamadas pela rotina principal (Apêndice F). A função gera_calendario_aux itera pelas $n-1$ rodadas. Por sua vez, a função gera_rodada aplica o Método Circular e invoca a lógica da heurística (detalhada no Apêndice C) e a função de atualização de dados (detalhada no Apêndice E).

```
1  int gera_calendario_aux(Calendar_data *calendar_data, int
       melhor_amplitude_atual)
2  {
3      int max_rounds = calendar_data->qnt_times - 1;
4      int i;
5      for (i = 0; i < max_rounds; i++)
6      {
7          calendar_data->rodada_atual = i;
8          gera_rodada(calendar_data);
9      }
10
11     // Calcula a amplitude final do calendario gerado
12     calendar_data->amplitude = calcula_amplitude(calendar_data->
       distancias,
13                                                  calendar_data->
       qnt_times);
14
15     // Otimizacao: descarta calendarios piores que o melhor
       atual
16     if (calendar_data->amplitude > melhor_amplitude_atual)
17         return 0;
18
19     return 1;
20 }
```

Fonte: Autoria própria (2025).

```
1  int gera_rodada(Calendar_data *calendar_data)
2  int gera_rodada(Calendar_data *calendar_data)
```

```
3  {
4      int qnt_times = calendar_data->qnt_times;
5      int rodada = calendar_data->rodada_atual;
6      int rodada_jogos[(qnt_times / 2)][2];
7
8      // Calcula a media atual
9      int soma_distancias = 0;
10     for (int i = 0; i < qnt_times; i++)
11     {
12         soma_distancias += calendar_data->distancias[i];
13     }
14     int media_distancias = soma_distancias / qnt_times;
15
16     for (int i = 0; i < qnt_times / 2; i++)
17     {
18         int idx1, idx2;
19
20         if (i == 0)
21         {
22             idx1 = rodada % (qnt_times - 1);
23             idx2 = qnt_times - 1; // fixo
24         }
25         else
26         {
27             idx1 = (rodada + i) % (qnt_times - 1);
28             idx2 = (rodada + qnt_times - 1 - i) % (qnt_times -
    1);
29         }
30
31         int time1 = calendar_data->times[idx1];
32         int time2 = calendar_data->times[idx2];
33
34         int partidas_time1 = calendar_data->
    partidas_seguidas_em_casa[time1];
```

```
35          int partidas_time2 = calendar_data->
    partidas_seguidas_em_casa[time2];
36
37          int mandante, visitante;
38
39          // Regra: quem passou do limite, nao pode ser mandante
40          if (partidas_time1 > 3 && partidas_time2 <= 3)
41          {
42              mandante = time2;
43              visitante = time1;
44          }
45          else if (partidas_time2 > 3 && partidas_time1 <= 3)
46          {
47              mandante = time1;
48              visitante = time2;
49          }
50          else
51          {
52
53              int ultimo_jogo_time1 = calendar_data->
    ultimo_jogo_primeiro_turno[time1];
54              int ultimo_jogo_time2 = calendar_data->
    ultimo_jogo_primeiro_turno[time2];
55
56              int ultimo_jogo_time1_segundo_turno = calendar_data
    ->ultimo_jogo_segundo_turno[time1];
57              int ultimo_jogo_time2_segundo_turno = calendar_data
    ->ultimo_jogo_segundo_turno[time2];
58
59              int valor_caso_time1_1 = calendar_data->distancias[
    time1] + calendar_data->matriz_distancia[ultimo_jogo_time1][
    time1];
60              int valor_caso_time1_2 = calendar_data->distancias[
    time2] + calendar_data->matriz_distancia[ultimo_jogo_time2][
```

```
time1];
61          int valor_caso_time2_1 = calendar_data->distancias[
time1] + calendar_data->matriz_distancia[ultimo_jogo_time1][
time2];
62          int valor_caso_time2_2 = calendar_data->distancias[
time2] + calendar_data->matriz_distancia[ultimo_jogo_time2][
time2];
63
64          if (ultimo_jogo_time2_segundo_turno != -1 &&
ultimo_jogo_time2_segundo_turno != -1)
65          {
66              valor_caso_time1_1 += calendar_data->
matriz_distancia[ultimo_jogo_time1_segundo_turno][time1];
67              valor_caso_time1_2 += calendar_data->
matriz_distancia[ultimo_jogo_time2_segundo_turno][time1];
68              valor_caso_time2_1 += calendar_data->
matriz_distancia[ultimo_jogo_time1_segundo_turno][time2];
69              valor_caso_time2_2 += calendar_data->
matriz_distancia[ultimo_jogo_time2_segundo_turno][time2];
70          }
71
72          int aux1, aux2;
73          aux1 = abs(valor_caso_time1_1 - media_distancias);
74          aux2 = abs(valor_caso_time1_2 - media_distancias);
75          int maior_diff_case_time1 = aux1 >= aux2 ? aux1 :
aux2;
76
77          aux1 = abs(valor_caso_time2_1 - media_distancias);
78          aux2 = abs(valor_caso_time2_2 - media_distancias);
79          int maior_diff_case_time2 = aux1 >= aux2 ? aux1 :
aux2;
80
81          if (maior_diff_case_time1 < maior_diff_case_time2)
82          {
```

```
83                  mandante = time1;
84                  visitante = time2;
85              }
86              else if (maior_diff_case_time2 <
    maior_diff_case_time1)
87              {
88                  mandante = time2;
89                  visitante = time1;
90              }
91              else
92              {
93                  // Empate: usa aleatoriedade
94                  Prng_ctx rng;
95                  uint32_t combined_seed = calendar_data->
    seed_calendario;
96                  combined_seed ^= rodada * 0x9E3779B9;
97                  combined_seed ^= (time1 * 0x85EBCA6B) ^ (time2 *
     0xC2B2AE35);
98                  prng_init(&rng, combined_seed);
99                  int hash = prng_next(&rng) & 1;
100
101                 mandante = hash ? time2 : time1;
102                 visitante = hash ? time1 : time2;
103             }
104         }
105
106         rodada_jogos[i][0] = mandante;
107         rodada_jogos[i][1] = visitante;
108
109         // Atualiza a matriz de partidas
110         calendar_data->campeonato[rodada][i].id_mandante =
    mandante;
111         calendar_data->campeonato[rodada][i].id_visitante =
    visitante;
```

```
112         calendar_data->campeonato[rodada + (qnt_times - 1)][i].
    id_mandante = visitante;
113         calendar_data->campeonato[rodada + (qnt_times - 1)][i].
    id_visitante = mandante;
114     }
115
116     atualiza_dados(rodada, rodada_jogos, calendar_data);
117     return 1;
118 }
```

Fonte: Autoria própria (2025).

APÊNDICE H – MATRIZ DE DISTÂNCIAS (2022)

Conforme mencionado na Seção 6.1, a matriz de distâncias geodésicas (em quilômetros) utilizada como entrada para o modelo e para a análise do calendário oficial da CBF para a temporada 2022 é detalhada na Tabela abaixo.

Matriz de distâncias geodésicas (em km) – Temporada 2022

	AME	ATH	ATL	CAM	AVA	BOT	CEA	COR	CTB	CUI	FLA	FLU	FOR	GOI	INT	JUV	PAL	BRA	SAN	SAO
AME	0	823	661	11	982	353	1871	479	811	1351	357	352	1875	657	1348	1258	481	445	514	492
ATH	823	0	979	822	257	653	2657	355	13	1291	673	660	2660	970	548	449	353	382	341	339
ATL	661	979	0	671	1228	933	1840	814	973	727	947	936	1840	10	1506	1401	814	774	868	811
CAM	11	822	671	0	978	344	1876	476	810	1361	347	343	1880	668	1345	1255	478	442	510	489
AVA	982	257	1228	978	0	730	2847	507	260	1539	746	737	2851	1219	375	304	505	546	468	497
BOT	353	653	933	344	730	0	2186	319	640	1548	22	8	2191	927	1105	1030	322	316	315	337
CEA	1871	2657	1840	1876	2847	2186	0	2341	2646	2318	2180	2182	12	1849	3201	3105	2342	2301	2381	2350
COR	479	355	814	476	507	319	2341	0	342	1319	341	327	2344	806	869	779	3	43	53	18
CTB	811	13	973	810	260	640	2646	342	0	1292	660	648	2649	964	558	459	340	370	328	327
CUI	1351	1291	727	1361	1539	1548	2318	1319	1292	0	1566	1553	2315	725	1671	1575	1317	1291	1365	1307
FLA	357	673	947	347	746	22	2180	341	660	1566	0	15	2185	942	1120	1046	343	338	336	359
FLU	352	660	936	343	737	8	2182	327	648	1553	15	0	2187	930	1112	1037	330	324	323	345
FOR	1875	2660	1840	1880	2851	2191	12	2344	2649	2315	2185	2187	0	1849	3204	3108	2345	2305	2384	2354
GOI	657	970	10	668	1219	927	1849	806	964	725	942	930	1849	0	1496	1391	805	766	859	803
INT	1348	548	1506	1345	375	1105	3201	869	558	1671	1120	1112	3204	1496	0	105	867	905	836	857
JUV	1258	449	1401	1255	304	1030	3105	779	459	1575	1046	1037	3108	1391	105	0	777	814	749	766
PAL	481	353	814	478	505	322	2342	3	340	1317	343	330	2345	805	867	777	0	43	54	16
BRA	445	382	774	442	546	316	2301	43	370	1291	338	324	2305	766	905	814	43	0	95	49
SAN	514	341	868	510	468	315	2381	53	328	1365	336	323	2384	859	836	749	54	95	0	59
SAO	492	339	811	489	497	337	2350	18	327	1307	359	345	2354	803	857	766	16	49	59	0

Fonte: Autoria própria (2025).

APÊNDICE I – MATRIZ DE DISTÂNCIAS (2023)

Conforme mencionado na Seção 6.2, a matriz de distâncias geodésicas (em quilômetros) utilizada como entrada para o modelo e para a análise do calendário oficial da CBF para a temporada 2023 é detalhada na Tabela abaixo.

Matriz de distâncias geodésicas (em km) – Temporada 2023

	AME	ATH	CAM	BAH	BOT	BRA	COR	CTB	CRU	CUI	FLA	FLU	FOR	GOI	GRE	INT	PAL	SAN	SAO	VAS
AME	0	823	11	1010	353	445	479	811	3	1351	357	352	1875	657	1350	1348	481	514	492	351
ATH	823	0	822	1833	653	382	355	13	826	1291	673	660	2660	970	551	548	353	341	339	660
CAM	11	822	0	1012	344	442	476	810	12	1361	347	343	1880	668	1347	1345	478	510	489	342
BAH	1010	1833	1012	0	1273	1454	1487	1821	1007	1942	1263	1268	979	1271	2356	2353	1489	1517	1500	1267
BOT	353	653	344	1273	0	316	319	640	355	1548	22	8	2191	927	1107	1105	322	315	337	8
BRA	445	382	442	1454	316	0	43	370	447	1291	338	324	2305	766	907	905	43	95	49	324
COR	479	355	476	1487	319	43	0	342	482	1319	341	327	2344	806	871	869	3	53	18	327
CTB	811	13	810	1821	640	370	342	0	814	1292	660	648	2649	964	560	558	340	328	327	648
CRU	3	826	12	1007	355	447	482	814	0	1352	359	354	1872	657	1353	1351	484	517	494	354
CUI	1351	1291	1361	1942	1548	1291	1319	1292	1352	0	1566	1553	2315	725	1673	1671	1317	1365	1307	1552
FLA	357	673	347	1263	22	338	341	660	359	1566	0	15	2185	942	1123	1120	343	336	359	15
FLU	352	660	343	1268	8	324	327	648	354	1553	15	0	2187	930	1114	1112	330	323	345	0
FOR	1875	2660	1880	979	2191	2305	2344	2649	1872	2315	2185	2187	0	1849	3207	3204	2345	2384	2354	2187
GOI	657	970	668	1271	927	766	806	964	657	725	942	930	1849	0	1498	1496	805	859	803	930
GRE	1350	551	1347	2356	1107	907	871	560	1353	1673	1123	1114	3207	1498	0	2	869	838	859	1114
INT	1348	548	1345	2353	1105	905	869	558	1351	1671	1120	1112	3204	1496	2	0	867	836	857	1112
PAL	481	353	478	1489	322	43	3	340	484	1317	343	330	2345	805	869	867	0	54	16	330
SAN	514	341	510	1517	315	95	53	328	517	1365	336	323	2384	859	838	836	54	0	59	323
SAO	492	339	489	1500	337	49	18	327	494	1307	359	345	2354	803	859	857	16	59	0	345
VAS	351	660	342	1267	8	324	327	648	354	1552	15	0	2187	930	1114	1112	330	323	345	0

Fonte: Autoria própria (2025).

APÊNDICE J – MATRIZ DE DISTÂNCIAS (2024)

Conforme mencionado na Seção 6.3, a matriz de distâncias geodésicas (em quilômetros) utilizada como entrada para o modelo e para a análise do calendário oficial da CBF para a temporada 2024 é detalhada na Tabela abaixo.

Matriz de distâncias geodésicas (em km) – Temporada 2024

	ATL	ATH	CAM	BAH	BOT	BRA	COR	CRI	CRU	CUI	FLA	FLU	FOR	GRE	INT	JUV	PAL	SAO	VAS	VIT
ATL	0	979	671	1265	933	774	814	1341	660	727	947	936	1840	1508	1506	1401	814	811	936	1238
ATH	979	0	822	1833	653	382	355	362	826	1291	673	660	2660	551	548	449	353	339	660	1798
CAM	671	822	0	1012	344	442	476	1118	12	1361	347	343	1880	1347	1345	1255	478	489	342	976
BAH	1265	1833	1012	0	1273	1454	1487	2123	1007	1942	1263	1268	979	2356	2353	2266	1489	1500	1267	36
BOT	933	653	344	1273	0	316	319	868	355	1548	22	8	2191	1107	1105	1030	322	337	8	1238
BRA	774	382	442	1454	316	0	43	684	447	1291	338	324	2305	907	905	814	43	49	324	1419
COR	814	355	476	1487	319	43	0	645	482	1319	341	327	2344	871	869	779	3	18	327	1452
CRI	1341	362	1118	2123	868	684	645	0	1125	1601	883	875	2989	240	238	183	644	635	875	2087
CRU	660	826	12	1007	355	447	482	1125	0	1352	359	354	1872	1353	1351	1261	484	494	354	972
CUI	727	1291	1361	1942	1548	1291	1319	1601	1352	0	1566	1553	2315	1673	1671	1575	1317	1307	1552	1918
FLA	947	673	347	1263	22	338	341	883	359	1566	0	15	2185	1123	1120	1046	343	359	15	1228
FLU	936	660	343	1268	8	324	327	875	354	1553	15	0	2187	1114	1112	1037	330	345	0	1232
FOR	1840	2660	1880	979	2191	2305	2344	2989	1872	2315	2185	2187	0	3207	3204	3108	2345	2354	2187	1010
GRE	1508	551	1347	2356	1107	907	871	240	1353	1673	1123	1114	3207	0	2	107	869	859	1114	2320
INT	1506	548	1345	2353	1105	905	869	238	1351	1671	1120	1112	3204	2	0	105	867	857	1112	2318
JUV	1401	449	1255	2266	1030	814	779	183	1261	1575	1046	1037	3108	107	105	0	777	766	1037	2231
PAL	814	353	478	1489	322	43	3	644	484	1317	343	330	2345	869	867	777	0	16	330	1454
SAO	811	339	489	1500	337	49	18	635	494	1307	359	345	2354	859	857	766	16	0	345	1465
VAS	936	660	342	1267	8	324	327	875	354	1552	15	0	2187	1114	1112	1037	330	345	0	1232
VIT	1238	1798	976	36	1238	1419	1452	2087	972	1918	1228	1232	1010	2320	2318	2231	1454	1465	1232	0

Fonte: Autoria própria (2025).