Universidade Federal Fluminense

ALLISON DA COSTA BATISTA GUEDES

Heurísticas para o Problema de Minimização de Efeitos Ponderados de Carry-Over na Construção de Tabelas de Torneios Round-Robin

NITERÓI

2009

Universidade Federal Fluminense

ALLISON DA COSTA BATISTA GUEDES

Heurísticas para o Problema de Minimização de Efeitos Ponderados de Carry-Over na Construção de Tabelas de Torneios Round-Robin

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientador:
Prof. Celso da Cruz Carneiro Ribeiro, Dr. Hab.

NITERÓI

2009

Heurísticas para o Problema de Minimização de Efeitos Ponderados de Carry-Over na Construção de Tabelas de Torneios Round-Robin

Allison da Costa Batista Guedes

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Aprovada por:

Prof. Celso da Cruz Carneiro Ribeiro, Dr. Hab. / IC-UFF
(Orientador)

Prof. Luiz Satoru Ochi, D.Sc. / IC-UFF

Prof. Simone de Lima Martins, D.Sc. / IC-UFF

Prof. Sebastián Alberto Urrutia, D.Sc. / DCC-UFMG

Niterói, 16 de Março de 2009.

"Um atleta não pode chegar à competição muito motivado se nunca foi posto à prova."

Sêneca

A Expedita, Alexandre e Álvaro, as três pessoas mais importantes de minha vida.

Agradecimentos

Em primeiro lugar, agradeço à minha amada mãe, cujos esforços sempre proporcionaram os meus. Também agradeço em especial aos meus dois irmãos, Alexandre e Álvaro, pelo carinho e apoio dispensados ao longo de toda a minha vida. Agradeço também à minha família, pela confiança e incentivos prestados.

Agradecer aos amigos é uma tarefa difícil, por serem muitos e nos mais variados âmbitos. Contudo, vou procurar me ater aos que estiveram mais presentes durante esses meus dois anos no RJ.

Agradeço a Ígor Petherson, por ter me recepcionado no aeroporto Galeão e me alocado na casa onde morava. Agradeço aos amigos com quem dividí residência em São Gonçalo: Robson Leandro, um dos caras mais "safos" e expertos que já conheci e Bruno Aguiar, a serenidade e tranquilidade em pessoa. Agradeço pelo acolhimento, pela convivência agradável e pela minha inclusão em seu ciclo de amizades, o que me proporcionou uma enorme aquisição de experiência de vida. Ainda em São Gonçalo, não posso deixar de citar Jerivan e Dona Olga. Agradeço também aos colegas e professores do Aliança Francesa de São Gonçalo.

Agradeço aos amigos mais recentes, Tarcísio Novis e Klauko Mota, do atual apartamento em Niterói. Esse último em especial, pelo grande companheirismo e amizade rapidamente fomentados. Ainda em Niterói, não posso deixar de citar o amigo Luciano Mesquita e o meu conterrâneo João Paulo.

Agradeço a todos os amigos e colegas que eu encontrei na UFF. Correndo o risco de cometer injustiças, cito aqueles com quem tive maior contato: Henrique Bueno, a primeira e mais simpática pessoa que conheci no IC; Carlos, a quem eu visitei várias vezes na tentativa de resolver algum problema e que sempre me foi bastante atencioso e gentil; Renatha Capua, Stênio Sã, Renato Moraes e Rafael Augusto, os colegas que têm ou tinham os mesmos "pais" que eu, dentro do IC; Warley, o TOca; Jaques, a figura de presença mais constante no IC; e todos os outros colegas com quem eu tenha tido

ao menos cinco minutos de conversa. Também gostaria de agradecer aos professores com quem tive uma maior convivência, seja em disciplinas, estudos dirigidos ou projetos: Cristina Boeres, Luiz Satoru, Alexandre Plastino, Loana Nogueira, Carlos Martinhon e Simone Martins. Também agradeço a Maria e Ângela, pela prontidão e simpatia com que sempre me atenderam.

Dentre os amigos que fiz na UFF, não poderia deixar de agradecer em separado àqueles que são hoje como irmãos para mim. Cito-os na ordem em que os conheci. Em primeiro lugar, o grande amigo Daniel Stevens, parceiro desde os primeiros momentos na UFF. Em seguida, Yuri Ferreira, o cara mais "bom-moço" do IC. Por último, mas não menos importante, o amigo Marcelo Pinheiro, que embora tenha passado pouco tempo no IC, acabou plantando a semente de uma grande amizade. Espero ver o quarteto reunido em breve.

Deixo um agradecimento especial ao professor e amigo Dario José Aloise, com quem sempre estive em contato nesses dois últimos anos e cuja amizade me é muito cara.

Agradeço em especial também ao professor e amigo Celso Ribeiro, pela orientação e ensinamentos dados, sem mencionar a preocupação que sempre demonstrou ter comigo. Não posso deixar de citar também o quanto sua paciência sempre me surpreendeu e o quanto sua gentileza sempre me motivou.

No mais, gostaria de agradecer aos amigos de Natal, em especial àqueles do laboratório PROMETH: João Saturnino, Ana Cristina, Wagner de Oliveira, Fábio Ferreira e Werner Soares. Além desses, também agradeço aos amigos que deram um jeito de arrumar um tempinho pra mim, nas oportunidades extremamente corridas em que pude voltar a Natal: Breno Costa, Catiane Dantas, Klívio Loreno, Allan Rocha, Francislí Galdino, Leônidas Pereira, André Gustavo, Vinícius Rodrigues, Wendell Klayton, Aline Gomes, Gustavo Mousinho, Thiago do Nascimento e Juciano Araújo. Por fim, agradeço aos amigos que nunca se distanciaram, mesmo que o único canal de comunicação tenha sido a internet: Lidjia Bruna, João Claudino (Joãozinho) e Dorian Fredson.

Os nomes citados aqui tornaram possível, em maior ou menor escala, a conclusão de mais essa etapa de minha vida.

Peço desculpas por algum nome que eu eventualmente tenha esquecido.

Resumo

O valor de efeitos de *carry-over* é um dentre vários critérios que podem ser utilizados para aferir a qualidade de um escalonamento de um torneio *round-robin*. É introduzida e discutida uma nova variante, ponderada, do problema de minimização de efeitos de *carry-over*. O problema é formulado por meio de programação inteira e uma heurística baseada na hibridização da metaheurística *Iterated Local Search* com uma estratégia multipartida é apresentada para sua solução aproximada. Instâncias de *benchmark* são propostas e resultados numéricos são apresentados.

Palavras-chave: *Carry-over*, *Round-robin*, Escalonamento em esportes.

Abstract

The carry-over effects value is one of the various measures one can consider to assess the quality of a round robin tournament schedule. We introduce and discuss a new, weighted variant of the minimum carry-over effects value problem. The problem is formulated by integer programming and a heuristic based on the hybridization of the Iterated Local Search metaheuristic with a multistart strategy is proposed for its approximate solution. Benchmark instances are proposed and numerical results are presented.

Keywords: Carry-over, Round-robin, Sports scheduling.

Siglas e Abreviações

RRT	:	Torneio *Round-Robin*, página 1
SRRT	:	Torneio *Round-Robin* Simples, página 2
DRRT	:	Torneio *Round-Robin* Duplo, página 2
COEV	:	Valor de efeitos de *Carry-Over*, página 2
ILS	:	*Iterated Local Search*, página 2
COEM	:	Matriz de efeitos de *Carry-Over*, página 16
GF	:	Corpo de *Galois*, página 18
VND	:	*Variable Neighborhood Descent*, página 27
TE	:	Troca de Equipes, página 34
TR	:	Troca de Rodadas, página 35
TPE	:	Troca Parcial de Equipes, página 35
TPR	:	Troca Parcial de Rodadas, página 36
RJ	:	Rotação de Jogos, página 37

Sumário

Lista de Tabelas

Capítulo 1

Introdução

As competições esportivas são uma importante atividade econômica e envolvem grande investimentos. Ligas profissionais atraem milhões de fãs e investimentos significativos em atletas, direitos de transmissão e propaganda. Também estão envolvidos muitos outros agentes, tais como organizadores, mídia e segurança. Dessa forma, existem vários problemas complexos a serem resolvidos, tais como a maximização de lucros, a otimização da logística e a organização das tabelas de competições. Problemas similares ocorrem na organização de competições amadoras, onde o montante de capital investido é menor mas o número de pessoas diretamente envolvidas costuma ser bem maior. Por exemplo, enquanto o campeonato brasileiro de futebol é disputado por 20 equipes, existem ligas amadoras locais que contam com a participação de um número muito maior de equipes – às vezes centenas –, exigindo um grande esforço de coordenação e logística.

O tema de *otimização em esportes* tem atraído a atenção de um número cada vez maior de pesquisadores, em áreas multi-disciplinares tais como pesquisa operacional, escalonamento de tarefas, programação de restrições, teoria dos grafos, otimização combinatória e matemática aplicada. Importância especial é dada a problemas de programação de torneios *round-robin*, devido à relevância prática desses problemas e à sua interessante estrutura matemática. A dificuldade dos problemas nesse campo leva ao uso de várias abordagens, incluindo programação inteira [6, 10, 21, 39, 42], programação por restrições [25, 27, 49], métodos híbridos [4, 15] e heurísticas [1, 18, 24, 45]. Alguns *surveys* tratando do assunto em questão são [16, 43]. O trabalho de Kendall et al. [30] é o *survey* mais recente sobre aplicações e métodos para a solução de problemas de otimização em esportes.

Um torneio *round-robin* (RRT – do inglês *Round-Robin Tournament*) é uma competição em que cada equipe joga contra todas as demais exatamente uma vez em cada

fase. Esse tipo de torneio ocorre durante um período de tempo previamente definido e é organizado em *rodadas*, nas quais são disputados no máximo um jogo por equipe. Cada uma dessas rodadas corresponde a um ou mais dias de competição. Se um torneio *round-robin* é composto de apenas uma fase, é dito que se trata de um torneio *round-robin* simples (SRRT – do inglês *Single Round-Robin Tournament*). Quando as equipes se enfrentam duas vezes, tem-se um torneio *round-robin* duplo (DRRT – Double Round-Robin Tournament). O campeonato brasileiro de futebol é um exemplo de competição que segue o modelo *round-robin* duplo. Nesse campeonato, cada par de equipes se encontra exatamente uma vez em cada uma das duas fases disputadas.

Na organização de torneios *round-robin*, a programação de jogos a ser seguida é de grande importância. Existem vários aspectos relevantes que devem ser considerados na determinação da melhor tabela possível. Em algumas situações, pode ser necessário minimizar a distância total viajada. Esse é o caso do Problema do Torneio Viajante [17] e sua variante, o Problema do Torneio Viajante Espelhado [45] – tipo de torneio comum em várias competições na América Latina. Outros problemas consistem em minimizar ou maximizar o número de quebras, isso é, o número de pares de jogos consecutivos disputados em casa (no próprio estádio, quadra, etc.) ou jogos consecutivos fora de casa (nas instalações do adversário). Alguns exemplos de trabalhos que tratam desse tipo de problema são [44, 48, 50]. Ribeiro e Urrutia [46] consideraram o problema do escalonamento do campeonato brasileiro de futebol. A função-objetivo considerada consistia em maximizar o número de jogos passíveis de serem televisionados por canais abertos. Esses e outros aspectos considerados estão associados a vários critérios de qualidade utilizados para avaliar tabelas de competição.

Nessa dissertação, considera-se o critério de avaliação de tabelas conhecido como *valor de efeitos de carry-over* (COEV – do inglês *Carry-Over Effects Value*). Esse critério permite avaliar a qualidade da distribuição dos jogos a serem disputados. Trata-se o problema de minimização do valor de efeitos de *carry-over*, que consiste em encontrar uma tabela onde o *COEV* é mínimo. Uma nova versão desse é proposta, considerando-se a utilização de pesos para cada efeito de *carry-over* computado. Também é apresentada uma formulação matemática por programação inteira para o problema. Serão apresentados métodos construtivos para a geração de tabelas iniciais, assim como os algoritmos utilizados para criar as estruturas matemáticas em que elas são baseadas. Também serão apresentadas as estruturas de vizinhança que foram utilizadas, bem como uma heurística do tipo *Iterated Local Search* (ILS) para a resolução do problema em apreço. São propostas instâncias de teste para a versão ponderada, obtidas por um gerador também proposto na tese.

Várias instâncias para a versão ponderada são propostas, assim como os métodos usados para sua criação. Por fim, serão mostrados os resultados computacionais obtidos com o procedimento implementado, realizando-se uma comparação com os métodos existentes para a versão original do problema.

O restante desta dissertação está organizado como segue:

- Capítulo 2: são apresentados os conceitos gerais relativos aos torneios *round-robin*. Também são mostradas as técnicas utilizadas para a geração das estruturas algébricas (1-fatorações) que serviram de base para a criação das soluções iniciais.

- Capítulo 3: são apresentadas as definições relativas ao problema de minimização de efeitos de *carry-over*. Também é feita uma revisão bibliográfica, onde são apresentados os métodos de solução (e seus resultados) para a versão original (não-ponderada) do problema. Em seguida, será proposta uma nova variante do problema em questão, onde é considerada a ponderação dos elementos da função-objetivo sendo tratada. Também será apresentada uma formulação por programação inteira para o problema em apreço.

- Capítulo 4: é apresentada a estratégia de solução empregada para resolver o problema em questão. São vistos os métodos construtivos utilizados para a criação de soluções, bem como todas as estruturas de vizinhança consideradas. Neste capítulo, o procedimento híbrido ILS com uma estratégia multi-partida é mostrado e o seu funcionamento é detalhado.

- Capítulo 5: inicialmente, é apresentado o gerador de instâncias ponderadas criado. Em seguida, são relatados os experimentos computacionais realizados e é feita uma análise dos resultados obtidos.

- Capítulo 6: discutem-se algumas conclusões e propostas de trabalhos futuros.

Capítulo 2

Torneios Round-Robin

Ligas e campeonatos profissionais movimentam uma grande quantidade de recursos. Organizadores desse tipo de competição estão frequentemente procurando otimizar os seus lucros e minimizar os seus custos. Um dos principais fatores que influencia um evento esportivo é a distribuição dos seus jogos ao longo do período de tempo definido. Dessa forma, uma das principais preocupações na organização de competições é a definição da tabela de jogos a ser seguida. Esse escalonamento costuma ser uma tarefa difícil, devido à existência de vários objetivos que devem ser otimizados. Como exemplo dos critérios envolvidos, pode-se citar aspectos logísticos, fatores econômicos e critérios de justiça. Além dos múltiplos critérios, também é provável que existam vários agentes envolvidos no processo de decisão. Entre tais decisores, pode-se mencionar federações esportivas, administradores de clubes e redes de televisão. Com isso, a definição da tabela de jogos a ser seguida é sempre uma tarefa bastante complicada e de alto custo computacional. Para vários dos problemas conhecidos hoje em dia, instâncias com apenas 20 ou mais equipes já são consideradas de grande porte e de difícil resolução. Isso faz com que exista a necessidade de se empregar métodos aproximados na busca de boas programações para competições esportivas.

Neste capítulo, serão apresentadas as definições relativas aos torneios *round-robin* que são utilizadas na dissertação. Também serão vistos métodos que podem ser empregados para a geração de escalonamentos de jogos em competições desse tipo.

2.1 Definições e nomenclatura

Muitas competições são organizadas como torneios *round-robin* (RRT). Nesse tipo de torneio, cada equipe (ou atleta) joga contra todas as demais exatamente uma vez em cada fase da competição. Um torneio *round-robin* contendo apenas uma fase é chamado de

torneio round-robin simples (SRRT). Se as equipes se enfrentarem duas vezes, dá-se o nome de *torneio round-robin duplo* (DRRT). Como exemplo de um torneio desse último tipo, pode-se citar o campeonato brasileiro de futebol, onde cada equipe enfrenta todas as outras exatamente uma vez em cada uma das duas fases.

Ao se escalonar um torneio, os jogos devem ser alocados a um determinado número de rodadas – em muitos casos, as rodadas correspondem aos dias de competição. Essa alocação é feita de forma que todas as equipes joguem no máximo uma vez em cada rodada. Quando o número de equipes n é par, são necessários pelo menos $n - 1$ rodadas para a realização da competição. Quando n é ímpar, o número mínimo de rodadas é igual a n. Se uma competição for disputada em um número mínimo de rodadas, então diz-se que se trata de um *RRT compacto*. Caso contrário, tem-se um *RRT relaxado*. A Figura 2.1 ilustra um torneio *round-robin* do primeiro tipo e a Figura 2.2 mostra um competição da segunda espécie. Em cada quadro mostrado, está representado uma rodada. Cada uma delas pode ser interpretada como um dia ou horário de competição em que os jogos indicados ocorrem simultaneamente. É interessante notar que ambas as figuras representam o mesmo conjunto de jogos. Contudo, a segunda figura é relativa a um torneio espalhado em mais do que $n - 1$ *rounds*.

Rodada 1	Rodada 2	Rodada 3	Rodada 4	Rodada 5
t1–t6	t1–t3	t1–t5	t1–t2	t1–t4
t2–t5	t2–t6	t2–t4	t3–t5	t2–t3
t3–t4	t4–t5	t3–t6	t4–t6	t5–t6

Figura 2.1: Exemplo de RRT compacto.

Rodada 1	Rodada 2	Rodada 3	Rodada 4	Rodada 5
t1–t6	t1–t3	t1–t5	t1–t2	t2–t3
t3–t4	t2–t6	t2–t4	t3–t5	t5–t6
	t4–t5	t3–t6		

Rodada 6	Rodada 7
t2–t5	t1–t4
t4–t6	

Figura 2.2: Exemplo de RRT relaxado.

2.1.1 Tabelas de competições

A alocação de jogos a rodadas pode ser representada por uma matriz. Essa matriz corresponde ao escalonamento de jogos a ser seguido em todo o campeonato. Na Figura 2.3,

é ilustrada a tabela de competição de um possível torneio disputado por oito equipes. Nessa matriz, cada linha representa uma rodada a ser jogada e cada coluna representa uma equipe participante da competição. A entrada da linha i e da coluna j corresponde à equipe enfrentada pela equipe j na rodada i. Por exemplo, a tabela mostrada indica que a equipe C enfrentaria a equipe A na segunda rodada.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	C	B	E	D	G	F	A
2	C	D	A	B	G	H	E	F
3	D	E	F	A	B	C	H	G
4	E	F	H	G	A	B	D	C
5	F	G	E	H	C	A	B	D
6	G	H	D	C	F	E	A	B
7	B	A	G	F	H	D	C	E

Figura 2.3: Exemplo de uma tabela de competição para um RRT com oito equipes.

A programação de uma tabela de um torneio *round-robin* consiste em definir, para cada rodada, o oponente de cada equipe participante da competição. Em outros problemas, outros tipos de decisões também são consideradas. Por exemplo, pode-se querer definir o local em que cada jogo será disputado. Nesse caso, uma tabela de competição é definida não só pelo escalonamento dos jogos, mas também pela especificação dos locais em que eles ocorrem. Assim, pode ser necessário definir se o jogo entre A e B será jogado nas instalações de A ou nas de B. Mais informações a respeito desse tipo de problema podem ser encontradas, por exemplo, em [16, 35, 42, 43, 44, 46, 51]. Para o caso do problema em apreço, esse tipo de informação não é relevante e não haverá preocupação com os locais onde os jogos são disputados.

O problema de definir tabelas de competição para torneios *round-robin* ocorre no contexto de diversos esportes, tais como futebol [29, 40, 42, 45, 46], basquete [26, 39], *baseball* [48] e outros, como pode ser encontrado em [30].

2.1.2 Formalização

Um escalonamento para um torneio *round-robin* compacto consiste de um conjunto de rodadas r_k ($k = 1, \ldots, n-1$), cada uma contendo $n/2$ jogos e onde nenhum par de rodadas possui jogos em comum.

Uma formulação matemática de programação inteira proposta por Trick para a caracterização de torneios *round robin* compactos pode ser encontrada em [16]. Esta formulação

define as variáveis binárias $x_{ijk} = 1$, se o jogo entre as equipes i e j ocorre na rodada r_k; $x_{ijk} = 0$, caso contrário. As restrições são:

$$\sum_{j:j>i} x_{ijk} = 1, \quad i = 1, \ldots, n \text{ e } k = 1, \ldots, n-1 \tag{2.1}$$

$$\sum_{k} x_{ijk} = 1, \quad i, j = 1, \ldots, n \quad \text{com } i < j \tag{2.2}$$

$$x_{ijk} \in \{0, 1\}, \quad i, j = 1, \ldots, n, \quad k = 1, \ldots, n-1 \text{ e } i < j. \tag{2.3}$$

As restrições (2.1) garantem que cada equipe joga exatamente uma vez em cada rodada. As restrições (2.2) garantem que cada equipe enfrenta todos os demais oponentes exatamente uma vez. As restrições (2.3) são relativas aos tipos binários das variáveis utilizadas.

2.1.3 Fatores e fatorações

Tabelas de competição para torneios *round-robin* podem ser obtidas de estruturas algébricas conhecidas como 1-fatorações. Essas estruturas possuem uma equivalência biunívoca com escalonamentos de competições do tipo estudado. Esse tipo de objeto matemático é amplamente utilizado para representar soluções para problemas que consistem em definir tabelas de competições. Exemplos de trabalhos que consideram fatores e fatorações são [11, 36, 41, 52]. A seguir, são vistas as definições relativas a esse tipo de estrutura.

Seja um grafo $G = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas de G. Um *fator* de um grafo $G = (V, E)$ é um subgrafo $G' = (V, E')$, com $E' \subseteq E$. Na Figura 2.4, vê-se o grafo completo K_6 e um de seus possíveis fatores.

Diz-se que um fator G' é um *1-fator* se todos os seus nós têm grau igual a um. A Figura 2.5 ilustra um exemplo de um 1-fator do grafo completo K_6.

Uma *fatoração* F de G é um conjunto de fatores disjuntos (em relação às arestas) de G, tal que a união dos seus conjuntos de arestas seja igual a E. A Figura 2.6 mostra uma ilustração de uma estrutura desse tipo.

Uma fatoração formada exclusivamente por 1-fatores é chamada de *1-fatoração*. Em uma *1-fatoração ordenada* de G, os seus 1-fatores são tomados em uma determinada

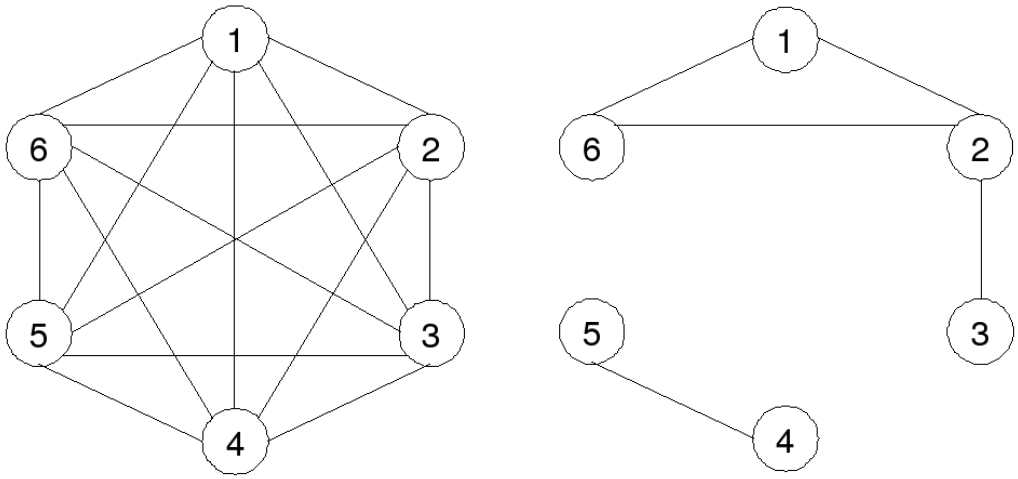

Figura 2.4: (a) Grafo completo K_6 e (b) um de seus possíveis fatores.

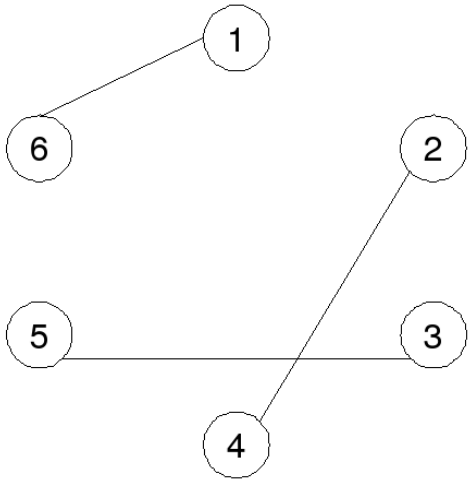

Figura 2.5: Exemplo de 1-fator do grafo completo K_6.

ordem. A Figura 2.7 ilustra uma 1-fatoração do grafo completo K_6.

2.2 Torneios *round-robin* e 1-fatorações

Existe uma correspondência direta entre torneios *round-robin* com n participantes e 1-fatorações ordenadas do grafo completo K_n. Cada vértice do grafo está associado a uma equipe do RRT considerado e cada aresta está relacionada com um jogo a ser disputado. Desse modo, cada rodada equivale a um 1-fator. Enfim, uma tabela completa de competição pode ser vista como uma 1-fatoração ordenada de um grafo completo.

2.3 Métodos de geração de 1-fatorações

Na seção anterior, foi visto que 1-fatorações de grafos completos são equivalentes a tabelas de competições do tipo *round-robin*.

Nesta seção, serão mostrados alguns métodos conhecidos para se obter 1-fatorações e, como consequência, obter tabelas de RRTs. Os dois primeiros algoritmos apresentados dão origem a objetos conhecidos como *1-fatorações canônicas* e *1-fatorações binárias*. Eles foram amplamente utilizados e são empregados no método proposto nesta dissertação. O terceiro procedimento mostrado faz uso de um outro tipo de estrutura algébrica – *starters* – para gerar 1-fatorações. O quarto método discutido é o de coloração.

2.3.1 1-fatorações canônicas

O método do polígono foi proposto em [31]. Sendo frequentemente citado na literatura – ver [14, 16, 36, 43, 52], por exemplo –, ele é o algoritmo mais conhecido e amplamente utilizado para a geração de 1-fatorações. Esse procedimento gera objetos conhecidos como 1-fatorações canônicas.

Seja $V = \{1, \ldots, n\}$ o conjunto de vértices do grafo completo K_n. Sejam F a 1-fatoração a ser gerada e F_k cada um de seus 1-fatores, para $k = 1, \ldots, n-1$. Utilizando-se este método, o conjunto de arestas do 1-fator F_k é dado por $\{(k, n)\} \cup \{(a(k, \ell), b(k, \ell)) : \ell = 1, \ldots, n/2 - 1\}$, para $k = 1, \ldots, n - 1$, com

$$a(k, \ell) = \begin{cases} k + \ell, & \text{se } (k + \ell) < n, \\ k + \ell - n + 1, & \text{se } (k + \ell) \geq n, \end{cases} \tag{2.4}$$

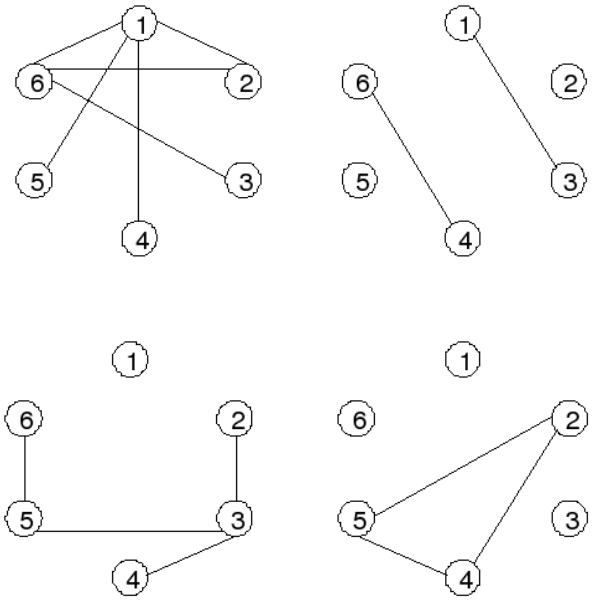

Figura 2.6: Exemplo de fatoração do grafo completo K_6.

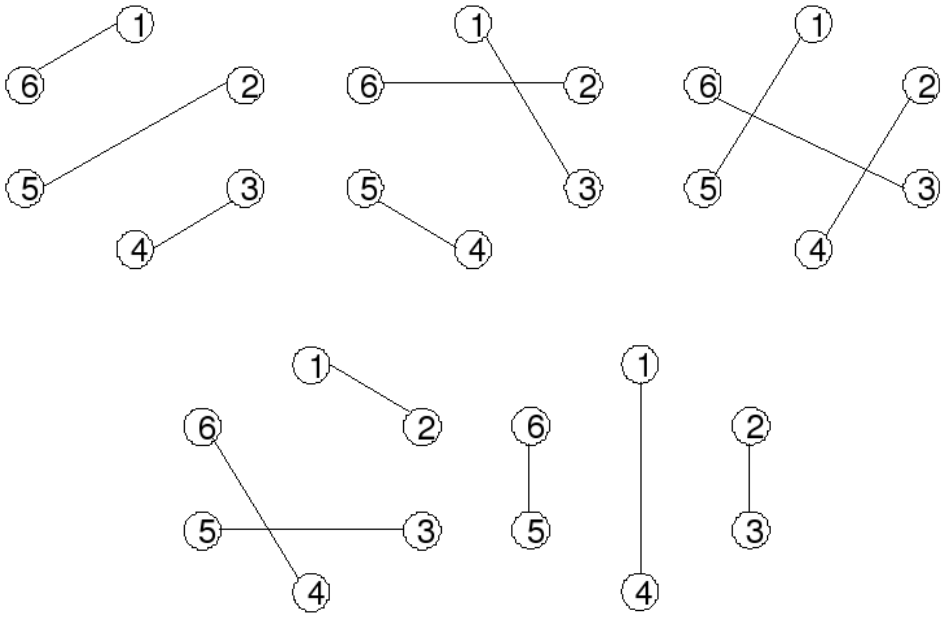

Figura 2.7: Exemplo de 1-fatoração do grafo completo K_6.

e

$$b(k,\ell) = \begin{cases} k-\ell, & \text{se } (k-\ell) > 0, \\ k-\ell+n-1, & \text{se } (k-\ell) \leq 0. \end{cases} \tag{2.5}$$

O procedimento descrito pode ser visualizado da seguinte forma. Fixando-se o vértice n em um ponto de um plano, os demais nós são dispostos ao seu redor na forma de um polígono regular, com um dos nós posicionado logo acima do vértice n. Depois, faz-se a ligação de n (nó central) com o vértice acima dele. Os demais jogos são definidos pelo emparelhamento dos vértices dispostos na mesma altura. Para gerar um outro 1-fator, basta rotacionar o polígono considerado em $360/(n-1)$ graus e repetir o procedimento. A Figura 2.8 ilustra o resultado obtido pelo método do polígono para K_6.

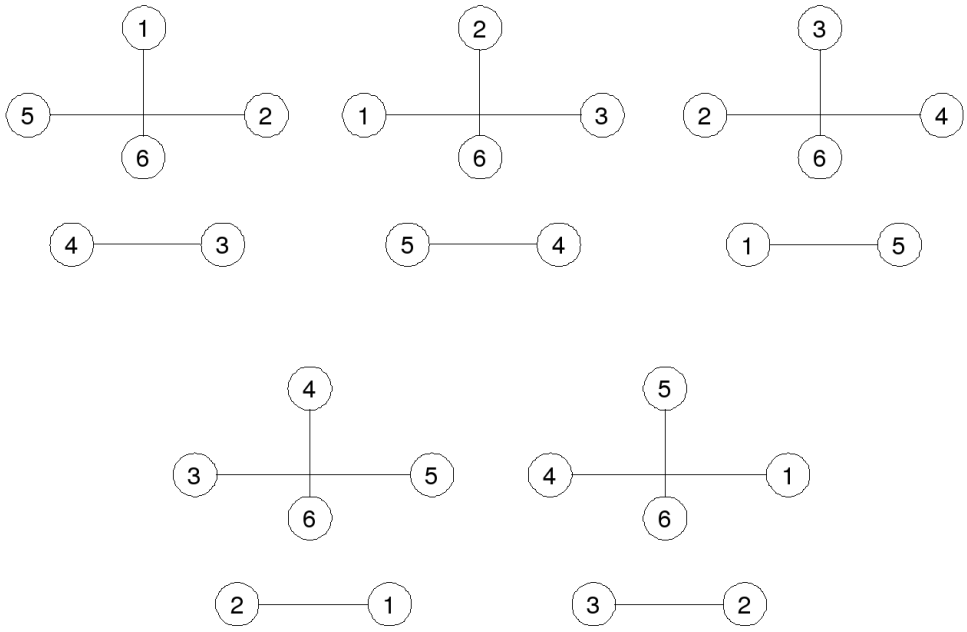

Figura 2.8: Método do polígono aplicado sobre o grafo completo K_6.

2.3.2 1-fatorações binárias

De Werra [11] introduziu um método capaz de gerar 1-fatorações essencialmente diferentes daquelas obtidas pelo o método do polígono, quando n é múltiplo de quatro. Os objetos criados por esse procedimento são chamados de 1-fatorações binárias. Outros trabalhos que mostram o algoritmo em questão são [12, 36, 52].

Inicialmente, separam-se os vértices em dois conjuntos $V_1 = \{1, \ldots, n/2\}$ e $V_2 = \{n/2+1, \ldots, n\}$. As primeiras $n/2$ rodadas são formadas por jogos com uma equipe em V_1 e a outra em V_2. As arestas dos 1-fatores correspondentes são definidas como

$\{(\ell, c(k,\ell)) : \ell = 1, \ldots, n/2\}$, para $k = 1, \ldots, n/2$, com

$$c(k,\ell) = [(k + \ell - 2) \bmod (n/2)] + n/2 + 1 \tag{2.6}$$

A Figura 2.9 ilustra o funcionamento da primeira fase do presente algoritmo. São mostrados os $n/2$ primeiros 1-fatores gerados para o grafo completo K_8.

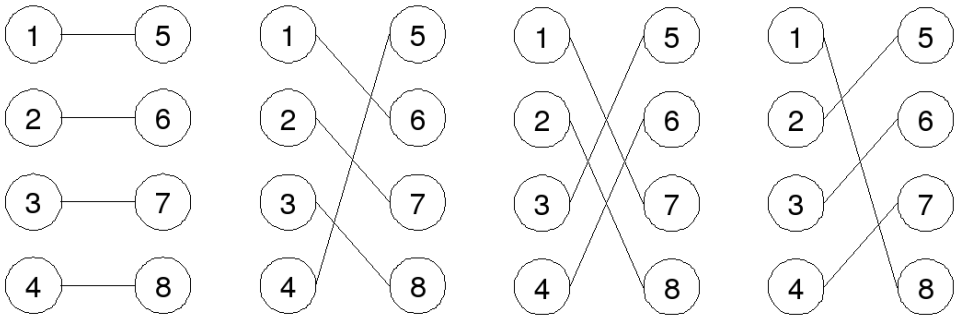

Figura 2.9: Primeira fase do método de de Werra aplicado sobre o grafo completo K_8.

Cada uma das $n/2-1$ rodadas restantes são formadas através da junção de um 1-fator da 1-fatoração do grafo completo associado a V_1 com um 1-fator da 1-fatoração do grafo completo associado a V_2.

2.3.3 1-fatorações por *starters*

Anderson [2] propôs a utilização de estruturas algébricas conhecidas como *starters* [13] na geração de 1-fatorações do grafo completo K_n. Seja H um grupo abeliano aditivo baseado no conjunto $Q = \{0, 1, 2, \ldots, n-2\}$. Então, um *starter* em H de ordem $n-1$ é definido por um conjunto de pares $\{x_1, y_1\}, \ldots, \{x_{n-1}, y_{n-1}\}$ se:

- $x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1}$ são os elementos não-nulos de Q; e
- $\pm(x_1 - y_1) \bmod (n-1), \ldots, \pm(x_{n-1} - y_{n-1}) \bmod (n-1)$ são também os elementos não-nulos de Q.

Com um *starter* disponível, gera-se o primeiro 1-fator da 1-fatoração F a ser criada. Os pares $(0, n-1)$ e $(x_1, y_1), \ldots, (x_{n/2-1}, y_{n/2-1})$ indicam as arestas desse primeiro 1-fator – ou os jogos da primeira rodada. A seguir, são gerados $n-2$ novos conjuntos de pares. Para cada nova rodada r_k $(k = 1, \ldots, n-2)$, os jogos são definidos por $(k, n-1)$ mais os pares mostrados em (2.7).

$$\{(x_i + k) \bmod (n-1), (y_i + k) \bmod (n-1)\}, \text{ para } i = 1, \ldots, n/2 - 1 \tag{2.7}$$

Tome-se por exemplo o *starter* $\{(1,7), (2,5), (3,10), (4,6)$ e $(8,9)\}$, para $n = 12$. A Figura 2.10 apresenta as rodadas geradas a partir desse *starter*.

Rodada 1	$(0,11)$	$(1,7)$	$(2,5)$	$(3,10)$	$(4,6)$	$(8,9)$
Rodada 2	$(1,11)$	$(2,8)$	$(3,6)$	$(4,0)$	$(5,7)$	$(9,10)$
Rodada 3	$(2,11)$	$(3,9)$	$(4,7)$	$(5,1)$	$(6,8)$	$(10,0)$
Rodada 4	$(3,11)$	$(4,10)$	$(5,8)$	$(6,2)$	$(7,9)$	$(0,1)$
Rodada 5	$(4,11)$	$(5,0)$	$(6,9)$	$(7,3)$	$(8,10)$	$(1,2)$
Rodada 6	$(5,11)$	$(6,1)$	$(7,10)$	$(8,4)$	$(9,0)$	$(2,3)$
Rodada 7	$(6,11)$	$(7,2)$	$(8,0)$	$(9,5)$	$(10,1)$	$(3,4)$
Rodada 8	$(7,11)$	$(8,3)$	$(9,1)$	$(10,6)$	$(0,2)$	$(4,5)$
Rodada 9	$(8,11)$	$(9,4)$	$(10,2)$	$(0,7)$	$(1,3)$	$(5,6)$
Rodada 10	$(9,11)$	$(10,5)$	$(0,3)$	$(1,8)$	$(2,4)$	$(6,7)$
Rodada 11	$(10,11)$	$(0,6)$	$(1,4)$	$(2,9)$	$(3,5)$	$(7,8)$

Figura 2.10: Exemplo de escalonamento gerado a partir de um *starter*.

2.3.4 1-fatorações por coloração

Outro método existente para a criação de 1-fatorações é o de coloração de arestas. Uma k-coloração de arestas do grafo completo K_n é equivalente a uma 1-fatoração desse mesmo grafo [8]. Cada uma das k cores obtidas indica um 1-fator.

O método inicia gerando um grafo auxiliar G', de forma a transformar a busca por uma coloração de arestas em um problema de coloração de vértices. O grafo auxiliar é gerado da forma apresentada a seguir. Para cada aresta (i, j) de K_n, cria-se um vértice correspondente V_{ij}. Então, faz-se a ligação dos vértices auxiliares, de modo a conectar aqueles que representam arestas que possuiam extremidades em comum no grafo original. No passo seguinte, o algoritmo encontra colorações (de vértices) viáveis para o grafo auxiliar. A coloração dos vértices de G define as cores das arestas de K_n.

No capítulo seguinte, é definido o problema sendo tratado nessa dissertação. São dadas as definições básicas e é feita uma revisão da literatura. Também é proposta uma variante do problema em apreço.

Capítulo 3

O Problema de Minimização de Efeitos de Carry-Over

No presente capítulo, será estudado o *Problema de Minimização de Efeitos de Carry-Over*. Esse problema consiste em encontrar tabelas de competição com o menor *valor de efeitos de carry-over* possível. Esse é um critério de justiça que leva a uma distribuição equilibrada da sequência de jogos ao longo da competição. Serão dadas algumas definições preliminares e ilustradas as estruturas matemáticas utilizadas. Em seguida, define-se o problema em questão e apresentam-se as estratégias de solução encontradas na literatura. Serão vistas duas conjeturas formuladas em trabalhos anteriores, assim como a refutação de uma delas. Ao fim do capítulo, é proposto o *Problema de Minimização de Efeitos Ponderados de Carry-Over*, onde são associados pesos a cada efeito de *carry-over* computado. Também é proposta uma formulação matemática por programação inteira para essa variante do problema, que é uma extensão da versão original.

3.1 Motivação

Um importante aspecto da estratégia de uma equipe (ou de um atleta), particularmente em competições longas, consiste em balancear seus esforços ao longo da competição disputada. Se uma equipe faz uma partida contra um oponente fraco, é provável que ela esteja em melhores condições para atuar na próxima rodada do que se ela tivesse enfrentado um adversário difícil. Equipes que jogam contra oponentes fortes provavelmente estarão mais cansadas para o seu próximo jogo. Portanto, espera-se que uma equipe (ou um atleta) faça menos esforço ao jogar contra um oponente que acabou de enfrentar uma equipe forte do que ela faria se jogasse contra um adversário que tivesse vindo de uma disputa fácil.

A situação anterior é especialmente verdadeira no caso de esportes que exigem uma grande quantidade de esforço físico (tais como luta livre ou greco-romana, *rugby* e artes marciais). Nesse tipo de esporte, não é incomum que uma equipe (ou um atleta) tenha várias disputas seguidas em um curto intervalo de tempo. Isso faz com que uma sequência de oponentes muito cansados seja extremamente atraente para qualquer competidor. Da mesma forma, qualquer equipe desejaria evitar uma sequência de oponentes bem repousados. Algumas tabelas de competição podem conter várias dessas sequências de jogos mais fáceis ou mais difíceis associados a uma ou mais equipes. Essa situação não caracteriza um escalonamento justo e é altamente indesejada em qualquer competição. Para ilustrar esse efeito, suponha-se uma competição de Karatê-Dô ou Judô. Não há divisão por pesos na categoria *absoluto* de tais esportes: um atleta franzino pode vir a enfrentar um oponente de grande massa muscular. Um competidor que tenha acabado de lutar contra um oponente muito forte provavelmente estará muito cansado (ou mesmo machucado) na sua próxima luta. Isso poderia deteriorar a sua performance, dando ao seu próximo adversário uma grande vantagem que este não deveria ter.

O presente trabalho aborda o problema de se encontrar tabelas de competição que minimizem as vantagens do tipo descrito. Cada equipe participante deve ser privilegiada o menor número de vezes possível pelo desgaste de outros adversários.

3.2 Definições

Numa competição esportiva, suponha-se que a equipe A joga contra a equipe C logo após ter jogado contra a equipe B. Se a equipe B for muito mais forte que os demais competidores, então a equipe C provavelmente levará alguma vantagem sobre A no jogo entre eles. Isso é devido ao grande esforço que a equipe A precisou fazer no seu jogo anterior. Essa situação em que uma das equipes pode ser beneficiada deve ser evitada ou, ao menos, minimizada.

É dito que uma equipe C recebe um *efeito de carry-over* devido à equipe B se, e somente se, existe uma equipe A que joga com C logo após ter jogado contra B. Nessa dissertação, são considerados torneios *round-robin* simples compactos disputados por n (par) equipes, no qual cada equipe joga contra todas as outras exatamente uma vez. Nas tabelas de competições consideradas, as rodadas são tomadas ciclicamente, de forma que a última rodada é vista como adjacente à primeira. A Figura 3.1 (a) mostra uma matriz que representa a tabela de um torneio hipotético. Cada elemento (i, j) dessa tabela informa

a equipe que joga contra a equipe j na rodada i. É possível contar o número de efeitos de *carry-over* que uma determinada equipe fornece a cada uma das demais e, com isso, montar a *matriz de efeitos de carry-over* (COEM - do inglês *Carry-Over Effects Matrix*). Cada elemento (i, j) dessa última matriz indica o número de efeitos de *carry-over* que a equipe i dá à equipe j. Na Figura 3.1 (b), pode-se ver a matriz de efeitos de *carry-over* subjacente à tabela mostrada na Figura 3.1 (a).

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	C	B	E	D	G	F	A
2	C	D	A	B	G	H	E	F
3	D	E	F	A	B	C	H	G
4	E	F	H	G	A	B	D	C
5	F	G	E	H	C	A	B	D
6	G	H	D	C	F	E	A	B
7	B	A	G	F	H	D	C	E

(a)

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	0	0	3	0	1	2	1	0
B	5	0	0	0	1	0	0	1
C	0	1	0	3	0	3	0	0
D	0	2	0	0	2	0	3	0
E	1	1	0	2	0	2	0	1
F	0	0	0	0	2	0	3	2
G	0	3	1	0	0	0	0	3
H	1	0	3	2	1	0	0	0

(b)

Figura 3.1: (a) Tabela de um RRT compacto com oito equipes e (b) a matriz de efeitos de *carry-over* subjacente.

Denotando-se a matriz de efeitos de *carry-over* por $C = (c_{ij})$, com $i, j = 1, \ldots, n$, pode-se verificar que $c_{ii} = 0$ e $\sum_{j=1}^{n} c_{ij} = \sum_{i=1}^{n} c_{ij} = n - 1$, para toda linha e coluna $i, j = 1, \ldots, n$. A qualidade de uma tabela de competição em relação aos efeitos de *carry-over* é avaliada pelo *valor de efeitos de carry-over* (COEV - do inglês *Carry-Over Effects Value*), definido como:

$$COEV = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^2 \ . \tag{3.1}$$

Em uma tabela de competição ideal (balanceada) segundo esse critério, obtém-se o limitante inferior $n(n-1)$ para esse valor, ou seja, todos os elementos que não estão na diagonal principal da matrix C são iguais a 1. Com isso, minimiza-se a variância dos elementos de C e obtém-se a tabela de competição mais justa possível. Na Figura 3.2, são mostradas uma tabela de competição satisfazendo esse critério (a) e sua matriz de efeitos de *carry-over* (b).

Uma tabela de boa qualidade, considerando-se os efeitos de *carry-over*, é aquela em que esses últimos estão distribuídos de forma bem equilibrada sobre as células da matriz de efeitos de *carry-over* e o valor de efeitos de *carry-over* é minimizado. Assim, define-se o *Problema de Minimização de Efeitos de Carry-Over*, originalmente proposto por

Russel [47], como o problema de encontrar tabelas de competição cujo valor de COEV seja mínimo.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	B	A	D	C	F	E	H	F
2	H	G	F	E	D	C	B	A
3	E	F	G	H	A	B	C	D
4	G	H	E	F	C	D	A	B
5	D	C	B	A	H	G	F	E
6	C	D	A	B	G	H	E	F
7	F	E	H	G	B	A	D	C

(a)

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	0	1	1	1	1	1	1	1
B	1	0	1	1	1	1	1	1
C	1	1	0	1	1	1	1	1
D	1	1	1	0	1	1	1	1
E	1	1	1	1	0	1	1	1
F	1	1	1	1	1	0	1	1
G	1	1	1	1	1	1	0	1
H	1	1	1	1	1	1	1	0

(b)

Figura 3.2: (a) RRT balanceado e (b) sua matriz COEM subjacente.

3.3 Abordagens da literatura

Nessa seção, serão abordados os métodos descritos na literatura que foram anteriormente empregados para a resolução do Problema de Minimização de Efeitos de *Carry-Over*.

3.3.1 Método de Russel

Ao definir o problema de minimização de efeitos de *carry-over*, Russel [47] também propôs um algoritmo para a construção de tabelas ótimas quando o número de equipes n é uma potência de 2. Mais especificamente, o método apresentado permite a obtenção de tabelas cujos COEVs são iguais ao limitante inferior $n(n-1)$. Esse algoritmo faz uso de estruturas matemáticas conhecidas como *corpos de Galois*. Antes de definir o procedimento em questão, serão dadas algumas definições básicas em relação a esse tipo de estrutura algébrica (para mais detalhes, ver [5]).

Um *corpo* $(B, +, *)$ é uma estrutura matemática que consiste em um conjunto-base $B = \{\alpha_0, \ldots, \alpha_{|B|}\}$ associado a duas operações binárias, $+$ e $*$. Essas operações são fechadas em relação ao conjunto-base, ou seja, $\alpha + \alpha' \in B$ e $\alpha * \alpha' \in B$ para quaisquer $\alpha, \alpha' \in B$. O conjunto-base possui um *elemento neutro* para cada operação – 0 e 1, respectivamente – tais que $\alpha + 0 = \alpha$ e $\alpha * 1 = \alpha$, para todo $\alpha \in B$. Para cada elemento α de B, existe um *elemento inverso aditivo* $-\alpha$ e um *elemento inverso multiplicativo* α^{-1} tal que $\alpha + (-\alpha) = 0$ e $\alpha * \alpha^{-1} = 1$. Além disso, as operações $+$ e $*$ são associativas e comutativas.

Um *corpo de Galois* de ordem p^m, denotado $GF(p^m)$, é um corpo cujo conjunto-base possui p^m elementos. Cada um desses elementos é comumente representado por um polinômio não-redutível (não-fatorável) de ordem estritamente inferior a m e cujos coeficientes se encontram no conjunto $\{0, 1, \ldots, m-1\}$. Utilizando-se essa representação, os elementos de B são tomados módulo algum polinômio $f(x)$ de ordem m, com os coeficientes sendo calculados módulo p.

Todo corpo de Galois possui pelo menos uma raiz primitiva α que gera os elementos não-nulos do conjunto G ciclicamente: $\{0, \alpha^0 = 1, \alpha, \alpha^2, \ldots, \alpha^{p^m-2}, \alpha^{p^m-1} = \alpha^0 = 1, \alpha^{p^m} = \alpha, ...\}$. Na Tabela 3.1, tem-se um exemplo de um corpo de Galois de ordem 2^3 com $f(x) = x^3 + x + 1$.

Representação exponencial	**Representação polinomial**
0	0
α^0	1
α^1	x
α^2	x^2
α^3	$x + 1$
α^4	$x^2 + x$
α^5	$x^3 + x^2 \equiv x^2 + x + 1 \pmod{f(x)}$
α^6	$x^2 + 1$
α^7	1

Tabela 3.1: Exemplo de um corpo de Galois de ordem 2^3.

No algoritmo proposto por Russel [47], são considerados os corpos de Galois que possuem a forma $\{t_0 = 0, t_1, \ldots, t_{n-1}\} = GF(2^m)$, para algum número inteiro positivo m. Cada elemento de B está associado a uma equipe participante da competição a ser escalonada.

Seja $\{t_1^*, \ldots, t_{n-1}^*\}$ uma sequência dos $(n-1)$ elementos não-nulos de $GF(2^m)$. Este método considera sequências tais que as $(n-1)$ somas parciais S_i sejam distintas:

$$S_i = \sum_{j=1}^{i} t_j^*, \quad \text{para } i = 1, \ldots, n-1 \tag{3.2}$$

Se α é uma raiz primitiva de $GF(2^m)$, pode-se obter uma sequência do tipo descrito fazendo-se $t_i^* = \alpha^i$, para $i = 1, \ldots, n-1$. A sequência S possui $(n-1)$ elementos. Seja t_N o elemento que não figura nessa sequência. Fazendo-se $S_0 = S_{n-1} = 0$, uma tabela balanceada em relação aos efeitos de *carry-over* – para n potência de 2 – é obtida fazendo-se a equipe t_i, $i = 1, \ldots, n-1$, enfrentar a equipe $(t_i + t_N + S_{k-1})$ na k-ésima rodada

$(k = 1, \ldots, n-1)$.

Para exemplificar o funcionamento desse procedimento, considere-se uma competição com oito equipes. A Tabela 3.2 mostra um corpo de Galois $GF(8)$ de ordem 2^3. Com isso, seria obtida a sequência de somas parciais mostrada na Tabela 3.3. Pode-se ver que o elemento t_3 não aparece na sequência mostrada. Assim, faz-se $t_N = t_3$ e calculam-se os adversários da equipe t_i, para cada rodada k, de acordo com a fórmula dada. Na Tabela 3.4, tem-se o cálculo da primeira rodada obtida.

Elemento	**Representação polinomial**
t_0	0
t_1	x
t_2	x^2
t_3	x^2+1
t_4	x^2+x+1
t_5	$x+1$
t_6	x^2+x
t_7	1

Tabela 3.2: Exemplo de elementos de um corpo de Galois.

i	$\boldsymbol{S_i}$ **(rep. polinomial)**	$\boldsymbol{S_i}$ **(elemento)**
0	0	t_0
1	x	t_1
2	x^2+x	t_6
3	$x+1$	t_5
4	x^2	t_2
5	x^2+x+1	t_4
6	1	t_7
7	0	t_0

Tabela 3.3: Exemplo de sequência de somas parciais.

Equipe	**Cálculo**					**Adversário**
t_0	$(t_0+t_3+S_0)$	$=$	$0+x^2+1+0$	$\equiv$	x^2+1	t_3
t_1	$(t_1+t_3+S_0)$	$=$	$x+x^2+1+0$	$\equiv$	x^2+x+1	t_4
t_2	$(t_2+t_3+S_0)$	$=$	x^2+x^2+1+0	$\equiv$	1	t_7
t_5	$(t_5+t_3+S_0)$	$=$	$x+1+x^2+1+0$	$\equiv$	x^2+x	t_6

Tabela 3.4: Primeira rodada obtida com o método de Russel.

Na Figura 3.3, tem-se uma tabela de competição para $n = 8$ obtida com o método apresentado, continuando-se o procedimento que gerou a rodada mostrada na Tabela 3.4.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
1	t_3	t_4	t_7	t_0	t_1	t_6	t_5	t_2
2	t_4	t_3	t_5	t_1	t_0	t_2	t_7	t_6
3	t_5	t_7	t_4	t_6	t_2	t_0	t_3	t_1
4	t_6	t_2	t_1	t_5	t_7	t_3	t_0	t_4
5	t_7	t_5	t_3	t_2	t_6	t_1	t_4	t_0
6	t_1	t_0	t_6	t_4	t_3	t_7	t_2	t_5
7	t_2	t_6	t_0	t_7	t_5	t_4	t_1	t_3

Figura 3.3: Exemplo de uma tabela de competição gerada pelo método de Russel.

Ao propor o método descrito, Russel conjeturou que podem não existir tabelas balanceadas quando o número de equipes n não é uma potência de 2. Será visto mais adiante que essa conjetura é falsa.

Ainda em [47], Russel também propôs um método de construção de tabelas para quando o número de equipes é da forma $n = p^m + 1$, onde p é primo e m é inteiro. É interessante notar que todo n positivo e menor ou igual a 20 ou tem esse formato ou é uma potência de 2. Esse segundo algoritmo faz uso das idéias empregadas no primeiro método.

Considere-se o corpo de Galois $GF(p^m)$ de ordem p^m. Seja o seu conjunto-base representado por $\{t_0 = 0, t_1, \ldots, t_{p^m-1}\}$. Como no método anterior, sejam $\{t_1^*, \ldots, t_{p^m-1}^*\}$ os elementos não-nulos de $GF(p^m)$ organizados de forma que as somas parciais S_i sejam todas distintas. Sejam $S_0 = S_{p^m-1} = 0$ e t_N o elemento que não está presente na sequência de somas parciais.

Constrói-se uma matriz auxiliar de dimensão $(p^m-1)\times p^m$ cujos elementos $(i+1, j+1)$ correspondem a $S_i + t_j$, para $i = 0, 1, \ldots, p-2$ e $j = 0, 1, \ldots, p^m - 1$. Acrescenta-se uma p^m-ésima linha, adicionando-se t_N a cada elemento da primeira linha. Em seguida, é feito o emparelhamento de $(p^m - 1)$ colunas (duas a duas), de forma que as diferenças entre os primeiros elementos de cada par de colunas sejam distintas e tal que nenhuma diferença seja o inverso aditivo de alguma outra diferença calculada. A coluna restante é então emparelhada com um vetor-coluna de dimensão $p^m \times 1$ cujas entradas são todas iguais a p^m. Para cada par de colunas, faz-se com que seus i-ésimos elementos se enfrentem na rodada i, com $i = 1, \ldots, p^m = n - 1$.

Para exemplificar o método descrito nos parágrafos acima, considere-se uma competição com $n = 6$ equipes. Dessa forma, tem-se $n = 6 = 5^1 + 1 = p^m + 1$, o que resulta em $p = 5$ e $m = 1$. Seja o corpo de Galois cujo conjunto-base é $\{0, 1, 2, 3, 4, 5\}$ e sejam $t_1^* = 2$, $t_2^* = 4$, $t_3^* = 3$ e $t_4^* = 1$. As somas parciais correspondentes (módulo $p^m = 5$) são:

$S_1 = 2$, $S_2 = 1$, $S_3 = 4$ e $S_4 = 0 = S_0$. Pode-se verificar que $t_N = 3$. A Tabela 3.5 mostra a matriz calculada em seguida. Emparelhando-se a coluna 1 com a 2 e a coluna 3 com a 5, obtém-se a tabela de competição mostrada na Figura 3.4.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 0 & 1 & 5 \\ 1 & 2 & 3 & 4 & 0 & 5 \\ 4 & 0 & 1 & 2 & 3 & 5 \\ 3 & 4 & 0 & 1 & 2 & 5 \end{pmatrix}$$

Tabela 3.5: Matriz auxiliar calculada pelo método de Russel.

	$\boldsymbol{t_0}$	$\boldsymbol{t_1}$	$\boldsymbol{t_2}$	$\boldsymbol{t_3}$	$\boldsymbol{t_4}$	$\boldsymbol{t_5}$
1	t_1	t_0	t_4	t_5	t_2	t_3
2	t_5	t_4	t_2	t_2	t_1	t_0
3	t_3	t_2	t_1	t_0	t_5	t_4
4	t_4	t_3	t_5	t_1	t_0	t_2
5	t_2	t_5	t_0	t_4	t_3	t_1

Figura 3.4: Tabela gerada pelo método de Russel, para $n = 6$.

Os resultados (valores de COEV) obtidos pelos métodos de Russel são apresentados na Tabela 3.6. Nessa tabela, também são apresentados os valores do limitante inferior $n(n-1)$ para cada valor de n menor ou igual a 20.

$\boldsymbol{n}$	**Russel**	$\boldsymbol{n(n-1)}$
4	12	12
6	60	30
8	56	56
10	138	90
12	196	132
14	260	182
16	240	240
18	428	306
20	520	380

Tabela 3.6: Resultados obtidos pelos métodos de Russel e valores do limitante inferior $n(n-1)$.

3.3.2 Método de Anderson

Na Seção 2.3.3, foi visto o método de geração de 1-fatorações proposto por Anderson [2]. Nesse algoritmo, são utilizadas estruturas algébricas conhecidas como *starters* [13]. Essas estruturas são utilizadas para gerar a primeira rodada de uma tabela de competição. Em seguida, a rodada gerada é utilizada para obter as demais.

O método de Anderson para o Problema de Minimização de Efeitos de *Carry-Over* consiste basicamente em encontrar um *starter* que gere a tabela com o menor valor de efeitos de *carry-over* possível. Anderson [2] não menciona o modo como foram achados todos os *starters* que ele utilizou, mas cita que alguns deles foram encontrados computacionalmente, sugerindo algum tipo de estratégia enumerativa. Além disso, não foram relatados os tempos computacionais dispendidos na busca por esses *starters*.

Na Tabela 3.7, são mostrados os resultados obtidos por Anderson [2], para $8 \leq n \leq 24$. Os valores apresentados constituem os melhores resultados conhecidos até a presente data. É interessante notar que esse método obteve resultados bem melhores do que aqueles apresentados por Russel [47]. Além disso, o valores obtidos para $n = 20$ e $n = 22$ permitem refutar a conjetura feita por esse último pois os resultados apresentados igualam os valores dos limitantes inferiores $n(n-1)$. Assim, percebe-se que existem tabelas balanceadas em relação aos efeitos de *carry-over*, mesmo quando n não é uma potência de 2.

n	**Anderson**	$n(n-1)$
8	56	56
10	108	90
12	176	132
14	234	182
16	240	240
18	340	306
20	380	380
22	462	462
24	644	552

Tabela 3.7: Resultados obtidos pelo método de Anderson e valores do limitante inferior $n(n-1)$.

3.3.3 Método de Miyashiro e Matsui

Miyashiro e Matsui [37] propuseram um novo precedimento para a resolução do problema de minimização de efeitos de *carry-over*. Foram considerados duas estratégias diferentes. Em uma delas, considerou-se a utilização do método do polígono, descrito na Seção 2.3.1, para gerar uma 1-fatoração F inicial. Os 1-fatores de uma 1-fatoração podem ser tomados em qualquer ordem possível. Assim, Miyashiro e Matsui passaram a considerar tabelas de competições geradas a partir de permutações dos 1-fatores de F. As permutações foram geradas da forma descrita a seguir.

Seja $\{F_1, F_2, \ldots, F_{n-1}\}$ o conjunto de 1-fatores de F, com n sendo o número de equipes. Seja $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ uma permutação dos números inteiros de 1 a n. Seja $F(\pi)$ a

1-fatoração formada pelos 1-fatores de F tomados na ordem especificada pela permutação π: $(F_{\pi_1}, F_{\pi_2}, \ldots, F_{\pi_{n-1}})$. A estratégia empregada por Miyashiro e Matsui consiste em gerar diversas permutações π aleatoriamente e analisar a qualidade das tabelas $F(\pi)$ geradas. Foi relatado que os melhores resultados obtidos por essa estratégia foram conseguidos após dois dias de geração aleatória de permutações.

A segunda estratégia utilizada foi baseada em programação por restrições. Miyashiro e Matsui acrescentaram restrições à formulação apresentada em [50], possibilitando a melhoria de seus resultados para $n = 10$ e $n = 12$. Na Tabela 3.8, pode-se verificar os resultados apresentados em [37]. Estão sublinhados os resultados cujos valores correspondem aos limitantes inferiores $n(n-1)$.

n	Miyashiro e Matsui	$n(n-1)$
4	<u>12</u>	<u>12</u>
6	<u>60</u>	<u>60</u>
8	<u>56</u>	<u>56</u>
10	108	90
12	176	132
14	254	182
16	<u>240</u>	<u>240</u>
18	400	306
20	488	380

Tabela 3.8: Resultados obtidos pelos métodos de Miyashiro e Matsui e valores do limitante inferior $n(n-1)$.

Um aspecto interessante dos resultados apresentados em [37] merece uma atenção especial. Os autores desse trabalho afirmaram que os resultados obtidos pelo seu método seriam os melhores conhecidos até então. Contudo, foi visto na Seção 3.3.2 que o método de Anderson [2], publicado vários anos antes, ainda apresenta os melhores valores de COEV conhecidos na literatura, o que se verifica até os dias de hoje.

3.3.4 Outros métodos

Alguns outros trabalhos, além dos mencionados anteriormente, também abordam o tema de minimização de efeitos de *carry-over*. Em alguns deles, são apresentados resultados teóricos que fogem ao escopo do presente trabalho [3, 28]. Em outros, são tratados tipos de torneios diferentes do da espécie considerada nessa dissertação [3]. Além desses, existem dois trabalhos em particular que são dignos de nota. Eles são discutidos brevemente a seguir.

Trick [50] apresentou uma metodologia para a abordagem de problemas de escalonamento de torneios *round-robin*. Essa metodologia se baseia na decomposição de um problema em mais de uma fase. Mais especificamente, Trick expõe a idéia de se criar tabelas em três etapas distintas. Primeiramente, sugere que se faça o escalonamento dos jogos a serem disputados, sem se preocupar com quais equipes realmente participarão de cada disputa. Em outras palavras, são consideradas equipes abstratas, definindo-se as rodadas do torneio sem a preocupação de associar equipes reais a equipes abstratas. Na segunda fase, as equipes reais são associadas a equipes hipotéticas. Na terceira e última etapa, definem-se os locais onde os jogos ocorrerão.

Trick utiliza a disciplina de programação por restrições, lançando mão de resolvedores comerciais para obter os escalonamentos dos jogos a serem disputados. Embora o seu método não tenha sido criado especificamente para o problema considerado nessa dissertação, são reportados alguns valores de COEVs obtidos com o seu algoritmo. Para competições com $n = 8$ equipes, o seu procedimento foi capaz de obter rapidamente o resultado ótimo. Para $n = 16$, o valor ótimo de COEV foi obtido com um tempo computacional da ordem de um dia de processamento. Para o caso $n = 6$, é relatado um valor de COEV igual a 60 e é confirmada sua otimalidade. Para torneios com $n = 10$ equipes, Trick reportou um valor final de COEV igual a 122, obtido após um dia de execução.

Outro trabalho que aborda marginalmente o problema em apreço é [27]. Esse trabalho também emprega a técnica de programação por restrições, fugindo do escopo da dissertação. Nele, Henz et al. ratificam a otimalidade do valor de COEV igual a 60 para torneios com seis equipes. Também são relatados valores de COEV iguais a 128 e 188 para $n = 10$ e $n = 12$, respectivamente.

Assim como no trabalho de Miyashiro e Matsui [37], existe uma inconsistência nos resultados apresentados em [27]. Nesse último, Henz et al. mencionam o valor 196 como sendo o melhor resultado conhecido para competições com $n = 12$ equipes. Mais uma vez é ignorado o trabalho de Anderson [2], publicado anos antes, já sendo relatado um valor de COEV igual a 176 para o caso de torneios disputados por $n = 12$ equipes.

3.4 Problema de minimização de efeitos ponderados de *carry-over*

Nesta seção, é proposta uma variação do problema descrito. Nela, considera-se a associação de pesos aos efeitos de *carry-over* levados em conta. Apresenta-se também uma

formulação matemática por programação inteira 0-1 para o problema. Essa formulação também contempla o caso original, não-ponderado, tratando-o como um caso particular da versão ponderada.

O problema original (não-ponderado) de minimização de efeitos de *carry-over* não leva em consideração qualquer informação a respeito das forças relativas das equipes participantes de um torneio: todos os efeitos de *carry-over* têm o mesmo peso unitário. Contudo, em uma competição esportiva real, é bastante provável que os organizadores tenham uma estimativa inicial de quão bem cada equipe deve vir a atuar. Pode-se considerar, por exemplo, a performance das equipes na edição anterior do torneio, a tradição de cada equipe, a qualidade dos seus jogadores e o apoio dos torcedores em seus jogos dentro de casa. Noronha et al. [40] apresentam um exemplo concreto de uma competição (o campeonato chileno de futebol) em que as equipes são classificadas *a priori* de acordo com alguns desses critérios, na tentativa de se construir uma tabela de competição justa.

Dessa forma, pode-se chegar à conclusão de que minimizar o valor de efeitos de *carry-over* não garante um escalonamento justo. De fato, uma abordagem mais ampla consiste em associar um peso w_{ij} a cada par ordenado (i, j) de equipes, baseando-se nas suas forças ou fraquezas relativas, e minimizar a soma ponderada dos efeitos de *carry-over*.

Para cada par de equipes $i, j = 1, \ldots, n$ (com $i \neq j$) e para cada rodada $k = 1, \ldots, n$ (com as rodadas representadas ciclicamente, de forma que a rodada $n-1$ seja seguida pela primeira rodada), define-se a varável binária $y_{kij} = 1$ se, e somente se, a equipe i joga contra a equipe j na rodada k. Caso contrário, tem-se $y_{kij} = 0$. Assim, a formulação de programação inteira do Problema de Minimização de Efeitos Ponderados de *Carry-Over* é apresentada abaixo:

$$\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} \cdot \left(\sum_{\ell=1}^{n} \sum_{k=1}^{n-1} y_{k\ell i} \cdot y_{(k+1)\ell j} \right)^2 \tag{3.3}$$

Sujeito a:

$$y_{kij} = y_{kji}, \qquad i, j, k = 1, \ldots, n \tag{3.4}$$

$$\sum_{i=1}^{n} y_{kij} = 1, \qquad j, k = 1, \ldots, n \tag{3.5}$$

$$\sum_{k=1}^{n-1} y_{kij} = 1, \qquad i, j = 1, \ldots, n : i \neq j \tag{3.6}$$

$$y_{kii} = 0, \qquad i, k = 1, \ldots, n \tag{3.7}$$

$$y_{nij} = y_{1ij}, \qquad i, j = 1, \ldots, n \tag{3.8}$$

$$y_{kij} \in \{0, 1\}, \qquad i, j, k = 1, \ldots, n. \tag{3.9}$$

Na equação (3.3), tem-se a função-objetivo, a qual minimiza a soma ponderada dos efeitos de *carry-over*. As restrições (3.4) garantem que as variáveis $y_{kij} = y_{kji}$ são as mesmas. Alternativamente, pode-se dizer que esse último conjunto de restrições garante que os jogos disputados entre as equipes i e j são os mesmos que aqueles entre as equipe j e i – no tipo de torneio sendo considerado, cada par de equipes só se enfrenta uma única vez. As restrições (3.5) asseguram que cada equipe jogue exatamente uma vez em cada rodada de um SRRT compacto. Os conjuntos de retrições (3.6) e (3.7) garantem que cada equipe joga contra cada outra equipe exatamente uma vez. As restrições (3.8) fazem com que a n-ésima rodada seja equivalente à primeira, de forma que as rodadas sejam tomadas ciclicamente. Nas restrições (3.9), têm-se a imposição de valores binários às variáveis utilizadas.

Observa-se que a formulação proposta utiliza uma função-objetivo quadrática. Também se verifica que são utilizadas $O(n^3)$ variáveis binárias do tipo 0-1.

No próximo capítulo, apresenta-se o algoritmo implementado para a resolução do problema proposto. O seu funcionamento é detalhado e são discutidas as técnicas utilizadas.

Capítulo 4

Estratégia de solução

Neste capítulo, será apresentada a estratégia algorítmica empregada para resolver o problema em apreço. Trata-se de um procedimento híbrido baseado na metaheurística *Iterated Local Search* (ILS). Utiliza-se também a estratégia de busca local conhecida como *Variable Neighborhood Descent* (VND). Essas duas técnicas serão brevemente descritas, assim como os conceitos básicos relativos a busca local. Serão vistos os métodos construtivos utilizados para a criação de soluções iniciais, bem como o algoritmo de busca local e todas as estruturas de vizinhança consideradas. O procedimento implementado é definido e o seu funcionamento é detalhado.

4.1 *Iterated Local Search*

A metaheurística conhecida como *Iterated Local Search* (ILS) [33, 34] – busca local iterada – foi utilizada como modelo para a criação do principal elemento algorítmico da estratégia de solução implementada. A essência dessa metaheurística consiste em combinar um procedimento de busca local com um método para gerar perturbações das soluções. Uma perturbação ocasiona uma pequena mudança na solução em que é aplicada, preferencialmente de maneira que a alteração efetuada não possa ser facilmente desfeita pelo algoritmo de busca local. Assim, uma iteração da metaheurística considerada ocorre da forma explicada a seguir.

Seja S a solução atual do processo de busca. Uma iteração consiste em se obter uma nova solução a partir de S, realizando-se uma perturbação seguida de uma busca local. Caso a solução assim gerada satisfaça um dado critério de aceitação, a solução atual é substituída e a busca prossegue a partir da nova solução S. Uma descrição em alto-nível do modelo de ILS é dada no Algoritmo 1.

Algoritmo 1 Modelo de *Iterated Local Search*

1: Gera-se uma solução inicial S_0;
2: Obtém-se S através de uma busca local partindo de S_0;
3: $S^* \leftarrow S$;
4: **Repita**
5: Obtém-se S' com a aplicação de uma perturbação a S;
6: Obtém-se S'' com a aplicação de uma busca local a S';
7: Substitui-se S por S'', se o critério de aceitação for satisfeito;
8: Atualiza-se a melhor solução encontrada S^*;
9: **Até** Critério de parada satisfeito;
10: **Retorne** S^*;

Na linha 1, é gerada uma solução inicial S_0. Em seguida, nas linhas 2 e 3, realiza-se uma busca local sobre S_0 e inicializa-se a melhor solução encontrada S^*. Na linhas 4 a 9, são realizadas as iterações do algoritmo. Na linha 5, perturba-se a solução corrente. Em seguida, na linha 6, executa-se busca local sobre a solução perturbada. As linhas 7 e 8 cuidam da atualização da solução corrente e da melhor solução encontrada, respectivamente.

4.2 *Variable Neighborhood Descent*

Variable Neighborhood Descent (VND) [23, 38] – descida com vizinhança variável – é uma estratégia de busca local que se baseia na utilização de várias estruturas de vizinhança diferentes.

A idéia básica de um algoritmo VND consiste em realizar movimentos aprimorantes baseados em mais de uma estrutura de vizinhança, denominadas $N_1, N_2, \ldots, N_t$. Quando um ótimo local é atingido com a vizinhança atual, a busca prossegue com a utilização de uma outra vizinhança. Isso é feito para que se possa escapar do ótimo local encontrado. As vizinhanças são consideradas alternadamente, até que a solução corrente seja um ótimo local com respeito a todas elas.

O Algoritmo 2 mostra o pseudo-código de um algoritmo do tipo VND. Nas linhas 1 e 2, são inicializados a solução corrente e o valor de k. Esse último é um parâmetro utilizado para indicar a estrutura de vizinhança a ser utilizada. O laço que vai da linha 3 até a linha 10 é executado até que todas as vizinhanças disponíveis sejam exploradas sem que haja melhora na solução corrente. Na linha 4, executa-se sobre a solução S o procedimento de busca local utilizando a estrutura de vizinhança N_k. Se a solução S' encontrada for melhor do que a corrente, atualiza-se essa última e reinicializa-se k. Caso

Algoritmo 2 Modelo de *Variable Neighborhood Descent*

Entrada: S_0 – solução inicial;
$N_1, N_2, \ldots, N_t$ – vizinhanças a serem utilizadas;
Saída: S – ótimo local encontrado;

```
1:  S ← S_0;
2:  k ← 1;
3:  Enquanto k ≤ t faça
4:      Obtém-se S' com a aplicação de uma busca local a S, utilizando a vizinhança N_k;
5:      Se S' é melhor do que S então
6:          S ← S';
7:          k ← 1;
8:      senão
9:          k ← k + 1;
10:     Fim Se
11: Fim Enquanto
12: Retorne S;
```

contrário, incrementa-se o valor de k, de forma que a próxima estrutura de vizinhança venha a ser considerada. Na linha 12, o algoritmo retorna o ótimo local correspondente à melhor solução encontrada.

4.3 Algoritmo proposto

A partir da presente seção, o algoritmo implementado para a resolução do Problema de Minimização de Efeitos Ponderados de *Carry-Over* é apresentado. Serão vistas as formas pelas quais são obtidas as 1-fatorações iniciais, assim como os métodos construtivos que as utilizam para gerar soluções. Serão apresentadas todas as estruturas de vizinhança implementadas, bem como o método usado para realizar perturbações. Também são apresentadas três variações pelas quais esse procedimento pode ser realizado. Em seguida, discutem-se os parâmetros que foram ajustados para a execução dos testes computacionais. Por fim, são apresentadas as estruturas de dados utilizadas e é feita uma análise da complexidade computacional do algoritmo implementado.

Os algoritmos apresentados na Seção 3.3 não exploram o uso de diferentes estratégias para a geração de tabelas iniciais, o que limita a qualidade das soluções por eles encontradas. Além disso, alguns desses procedimentos requerem grandes tempos computacionais, o que os tornam inviáveis para a resolução eficiente do problema em apreço. O algoritmo de Anderson [2], por exemplo, pressupõe que um *starter* inicial seja conhecido de antemão. Isso torna necessária a existência de um passo enumerativo para encontrar um *starter* adequado, o que sugere tempos computacionais de grande magnitude. Os algoritmos que

utilizam programação por restrições também apresentam o mesmo tipo de problema: os tempos de processamento são bastante elevados até mesmo para valores de n razoavelmente pequenos ($14 \leq n \leq 20$). No caso específico do método de Miyashiro e Matsui [37], o algoritmo é baseado em um procedimento puramente aleatório. Isso deixa margem para o desenvolvimento de métodos criados especificamente para o problema, levando em conta as suas propriedades. O algoritmo de Russel [47] é o único método dentre os apresentados que não requer altos tempos de processamento. Contudo, esse procedimento não é capaz de achar boas soluções quando são considerados valores de n que não são potências de 2.

O presente trabalho propõe uma abordagem heurística para a minimização de efeitos de *carry-over* ponderados. O algoritmo proposto é baseado na hibridação da metaheurística *Iterated Local Search* (ILS) – discutida na Seção 4.1 – com uma estratégia de inicialização multi-partida. O procedimento possui duas fases principais: uma fase multi-partida e uma fase ILS. Uma sequência de execução do algoritmo é composta por essas duas fases, nessa ordem. Um execução completa compreende várias sequências independentes, retornando-se a melhor solução encontrada.

Antes de iniciar uma sequência de execução do algoritmo proposto, gera-se uma 1-fatoração inicial F. Os 1-fatores F_j de F são então usados para compor as soluções que servirão de ponto de partida para o processo de busca. Na Seção 4.3.1, descreve-se o procedimento pelo qual essa 1-fatoração inicial é obtida.

Durante a fase multi-partida, são geradas 100 soluções iniciais. Cada uma delas é obtida através da aplicação de um método construtivo, seguida da realização de uma busca local. Os métodos construtivos implementados utilizam os 1-fatores da 1-fatoração inicial F para formar soluções viáveis. Na Seção 4.3.2, discutem-se os meios pelos quais são obtidas soluções completas a partir dos 1-fatores de F.

A melhor solução encontrada na fase multi-partida é utilizada como ponto inicial da segunda fase (ILS). Nessa fase, utilizam-se os procedimentos de perturbação e busca local na tentativa de melhorar as soluções encontradas na primeira etapa. A Seção 4.3.5 detalha os modos pelos quais uma solução pode ser perturbada. A estrutura de vizinhança utilizada para tal é detalhada na Seção 4.3.3, juntamente com as demais vizinhanças empregadas.

O Algoritmo 3 apresenta, em linhas gerais, o pseudo-código da heurística híbrida proposta, que realiza dez execuções independentes das duas fases descritas. Na linha 2, é gerada uma 1-fatoração que servirá de base para a criação de soluções viáveis. Nas linhas 3 a 7, está representada a fase multi-partida. São geradas 100 soluções distintas.

Algoritmo 3 Heurística híbrida

1: **Para** iteração = 1 até 10 **faça**
2: Constrói-se uma 1-fatoração inicial F;
3: **Repita**
4: Obtém-se S a partir de um método construtivo, utilizando F;
5: Obtém-se S' com a aplicação de uma busca local a S;
6: Atualiza-se a melhor solução encontrada S^*;
7: **Até** 100 soluções iniciais geradas;
8: $S \leftarrow S^*$;
9: **Repita**
10: Obtém-se S' com a aplicação de uma perturbação a S;
11: Obtém-se S'' com a aplicação de uma busca local a S';
12: Substitui-se S por S'', se o critério de aceitação for satisfeito;
13: Atualiza-se a melhor solução encontrada S^*;
14: **Até** critério de parada satisfeito;
15: **Fim Para**
16: **Retorne** S^*;

Na linha 4, uma solução S é construída através de um método construtivo, a ser descrito na Seção 4.3.1. Na linha 5, faz-se um refinamento de S através do procedimento de busca local. Esse procedimento é discutido na Seção 4.3.4. Na linha 6, atualiza-se a melhor solução encontrada até então, armazenando-a na variável S^*. Na linha 8, faz-se S igual a S^*, garantindo-se que a próxima fase seja iniciada com a melhor solução encontrada na fase multi-partida.

Nas linhas 9 a 14, tem-se a fase ILS do algoritmo considerado. Inicialmente, na linha 10, é feita uma perturbação na solução corrente S. Na linha 11, a solução perturbada é refinada por meio de busca local. Em seguida, a solução corrente S é substituída por S'' se o valor de COEV da segunda é menor ou igual do que $(1 + b)$ multiplicado pelo COEV da primeira, onde b é um parâmetro do algoritmo. O valor de b dobra a cada $2n$ iterações sem que o critério de aceitação seja satisfeito e a solução corrente seja alterada. Quando ocorre uma atualização, b é redefinido com o seu valor inicial, seguindo a mesma estratégia utilizada em [45]. Por fim, a melhor solução encontrada é atualizada, na linha 13. A fase ILS termina quando é detectado, na linha 14, que um número máximo de movimentos deteriorantes foram aceitos desde a última atualização da melhor solução encontrada. Esse e os demais parâmetros que influenciam o processo de busca são discutidos no Capítulo 5.

4.3.1 1-fatorações iniciais

O algoritmo proposto inicia com a obtenção de uma 1-fatoração inicial F. Para gerar esse objeto, utiliza-se indistintamente ou o método do polígono (Seção 2.3.1) ou o método da fatoração binária (Seção 2.3.2), quando n é múltiplo de quatro. Para os demais casos, utiliza-se somente o primeiro método.

No Capítulo 5, será visto que a utilização de mais de um método para gerar 1-fatorações iniciais melhorou a performance do algoritmo proposto. Foi possível encontrar soluções melhores do que aquelas obtidas apenas com o método do polígono, que é o procedimento mais utilizado na literatura.

4.3.2 Método construtivo

Na linha 4 do Algoritmo 3, um método construtivo é empregado para gerar a solução S. Para esse fim, são utilizados os 1-fatores (rodadas) presentes em F.

Os 1-fatores de uma 1-fatoração podem ser considerados em qualquer ordem. Analogamente, as rodadas de uma tabela de competição podem ser permutadas livremente sem que seja violada nenhuma de suas propriedades. Invertendo-se a ordem de duas rodadas quaisquer, ainda se verifica que cada equipe joga exatamente uma vez contra todos os demais e que cada equipe joga exatamente uma vez em cada rodada. Dessa forma, novas tabelas de competição podem ser obtidas ao se considerar as rodadas de uma tabela pré-existente em qualquer ordem possível. O método construtivo proposto considera esse fato. Seu pseudo-código é apresentado no Algoritmo 4.

Algoritmo 4 Método construtivo

Entrada: F – 1-fatoração usada como base;
Saída: S – solução completa;
1: $S \leftarrow \varnothing$;
2: Escolhem-se dois 1-fatores aleatórios de F;
3: Colocam-se os dois 1-fatores selecionados em S;
4: **Enquanto** $F \setminus S \neq \varnothing$ **faça**
5: Seja F' um 1-fator de $F \setminus S$;
6: Coloca-se F' em S;
7: **Fim Enquanto**
8: **Retorne** S;

Inicialmente, a solução S ainda não possui rodada alguma. A linha 1 faz a inicialização de S. Na linha 2, dois 1-fatores da 1-fatoração F são escolhidos aleatoriamente. Em seguida, na linha 3, esses dois 1-fatores são inseridos na solução S. O laço das linhas 4

a 7 é executado até que a solução S esteja completa, contendo todos os 1-fatores de F. Na linha 5, escolhe-se um 1-fator F_k ainda não utilizado. Na linha seguinte (6), o 1-fator selecionado é inserido em S. O procedimento retorna a solução construída.

As regras utilizadas para escolher um 1-fator de F e inserí-lo em S seguem aquelas usadas em dois diferentes métodos para o Problema do Caixeiro Viajante [22, 32]. Os procedimentos considerados são o do *Vizinho Mais Próximo* e o da *Inserção Arbitrária*. As regras utilizadas em cada um dos dois métodos são explicadas a seguir.

- **Vizinho mais próximo**: São verificadas todos os 1-fatores de F que ainda não foram utilizados. Calcula-se o incremento ocasionado no COEV de S, ao se inserir cada um deles como o último 1-fator dessa solução. Escolhe-se o 1-fator que apresenta o menor incremento, posicionando-o como último 1-fator de S.
- **Inserção arbitrária**: Escolhe-se um 1-fator qualquer de F. Calcula-se o incremento que seria ocasionado no COEV de S, se ele fosse inserido entre um par de 1-fatores adjacentes nessa solução. A inserção é feita na posição que produz o menor incremento possível.

Quando o Algoritmo 4 é chamado, uma das duas regras acima é selecionada aleatoriamente, cada uma com probabilidade de 50%. O algoritmo construtivo utiliza a regra escolhida até o final de sua execução. Ou seja, apenas uma regra é usada para cada solução gerada.

4.3.3 Estruturas de vizinhança

A *vizinhança* $N(S)$ de uma solução S é definida como o conjunto de soluções que podem ser obtidos a partir de modificações feitas nessa última. Uma *estrutura de vizinhança* N define as alterações que devem ser realizadas em S para que se possa obter os seus vizinhos. Ou seja, uma estrutura de vizinhança define quais são os vizinhos de fato de uma determinada solução. Diferentes estruturas levam a diferentes conjuntos de vizinhos.

Em um procedimento de busca local, os vizinhos da solução corrente S são verificados com o intuito de se achar uma solução S' de melhor qualidade. Caso nenhuma solução superior seja encontrada, diz-se que S é um *ótimo local*. Caso contrário, faz-se a solução corrente igual a S' e verificam-se os vizinhos dessa nova solução. O processo de se deslocar de uma solução para outra é chamado de *movimento*. Assim, uma busca local consiste

na realização de movimentos que levam de uma solução a outra, até que se alcance um ótimo local.

O processo de busca local pode ser de dois tipos: *mais aprimorante* ou *primeiro aprimorante*. Considerando-se o primeiro tipo, são avaliados todos os movimentos possíveis a partir de S e é realizado aquele que leva à melhor solução vizinha. Para o segundo caso, realiza-se um movimento que leve à primeira solução encontrada que seja melhor do que a corrente. No algoritmo proposto, foram utilizados movimentos do tipo mais aprimorante.

Algoritmo 5 Modelo de busca local

Entrada: S_0 – solução inicial;
Saída: S – ótimo local encontrado;
1: $S \leftarrow S_0$;
2: **Enquanto** existir $S' \in N(S)$ melhor do que S **faça**
3: Obtém alguma solução S' em $N(S)$ que seja melhor do que S;
4: $S \leftarrow S'$;
5: **Fim Enquanto**
6: **Retorne** S;

O Algoritmo 5 mostra, em linhas gerais, o funcionamento de uma busca local. Na linha 1, inicializa-se a solução corrente S com a solução S_0 dada. Na linha 2, verifica-se a existência de alguma solução vizinha a S que seja melhor do que ela. Caso haja alguma, atualiza-se a solução corrente nas linhas 3 e 4. Na linha 6, é retornada a melhor solução encontrada, que se trata de um ótimo local em relação à estrutura de vizinhança N utilizada.

Nas sub-seções a seguir, são vistas as estruturas de vizinhança implementadas. As quatro primeiras são consideradas no procedimento de busca local, enquanto a última é utilizada para a realização de perturbações. As definições das vizinhanças consideradas são dadas em [9].

4.3.3.1 Troca de Equipes (TE)

Uma solução na vizinhança *Troca de Equipes* (TE) é obtida com a troca de adversários entre duas equipes t_1 e t_2, em todas as rodadas do torneio do qual participam. Se a equipe t_3 enfrenta t_1 em uma determinada rodada, t_3 passaria a enfretar t_2 nessa mesma rodada. Um movimento nessa vizinhança é equivalente a trocar a posição de duas colunas da tabela de competição. Na Figura 4.1, é ilustrado um possível movimento nessa estrutura de vizinhança, apresentando-se uma tabela de competição hipotética e a solução vizinha obtida pela troca dos adversários da equipe C com os da equipe G.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	C	B	E	D	G	F	A
2	C	D	A	B	G	H	E	F
3	D	E	F	A	B	C	H	G
4	E	F	H	G	A	B	D	C
5	F	G	E	H	C	A	B	D
6	G	H	D	C	F	E	A	B
7	B	A	G	F	H	D	C	E

(a)

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	G	F	E	D	C	B	A
2	G	D	E	B	C	H	A	F
3	D	E	H	A	B	G	F	C
4	E	F	D	C	A	B	H	G
5	F	C	B	H	G	A	E	D
6	C	H	A	G	F	E	D	B
7	B	A	G	F	H	D	C	E

(b)

Figura 4.1: Exemplo de um movimento na vizinhança Troca de Equipes: (a) tabela original e (b) tabela com os jogos das equipes C e G trocados.

4.3.3.2 Troca de Rodadas (TR)

Uma solução na vizinhança *Troca de Rodadas* (TR) é obtida com a troca de posições entre duas rodadas r_1 e r_2. Os confrontos que ocorriam em r_1 passariam a ocorrer em r_2 e vice-versa. Um movimento nessa vizinhança é equivalente a trocar a posição de duas linhas da tabela de competição. Um possível movimento utilizando essa estrutura de vizinhança é ilustrado na Figura 4.2, apresentando-se uma solução hipotética e a solução obtida com a permutação das rodadas 2 e 5.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	C	B	E	D	G	F	A
2	C	D	A	B	G	H	E	F
3	D	E	F	A	B	C	H	G
4	E	F	H	G	A	B	D	C
5	F	G	E	H	C	A	B	D
6	G	H	D	C	F	E	A	B
7	B	A	G	F	H	D	C	E

(a)

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	H	C	B	E	D	G	F	A
2	F	G	E	H	C	A	B	D
3	D	E	F	A	B	C	H	G
4	E	F	H	G	A	B	D	C
5	C	D	A	B	G	H	E	F
6	G	H	D	C	F	E	A	B
7	B	A	G	F	H	D	C	E

(b)

Figura 4.2: Exemplo de um movimento na vizinhança Troca de Rodadas: (a) tabela original e (b) tabela com as rodadas 2 e 5 permutadas.

4.3.3.3 Troca Parcial de Equipes (TPE)

A estrutura de vizinhança conhecida como *Troca Parcial de Equipes* (TPE) é uma versão mais geral da vizinhança TE. Em vez de se fazer a troca de todos os adversários das equipes t_1 e t_2, considera-se a permutação de apenas alguns de seus oponentes.

Sejam alguma rodada r e duas equipes t_1 e t_2 quaisquer. Seja R o subconjunto de rodadas de cardinalidade mínima incluindo r e nas quais os oponentes de t_1 e t_2 são os

mesmos. Um movimento da vizinhança TPE consiste em trocar os oponentes de t_1 e t_2 em todas as rodadas contidas em R.

Na Figura 4.3, é ilustrado um possível movimento nessa estrutura de vizinhança, apresentando-se uma tabela hipotética e o resultado do movimento que leva em consideração a rodada r e as equipes t_1 e t_2. As rodadas contidas em R estão marcadas com um asterisco. Os oponentes que serão permutados, nas colunas das equipes t_1 e t_2, estão destacados em cinza escuro. Quando a alteração é efetuada, também é preciso modificar as entradas das colunas relativas às equipes que enfrentam t_1 e t_2, em cada rodada de R. As células marcadas em cinza claro indicam as posições em que essas modificações devem ocorrer – essas entradas são iguais a t_1 e t_2.

	t_1	t_2						
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	D	F	E	A	C	B	H	G
$r =$ **2**	H	E	D	C	B	G	F	A
3	F	D	G	B	H	A	C	E
4	B	A	F	H	G	C	E	D
* **5**	C	H	A	G	F	E	D	B
* **6**	E	G	H	F	A	D	B	C
* **7**	G	C	B	E	D	H	A	F

(a)

	t_1	t_2						
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	D	F	E	A	C	B	H	G
$r =$ **2**	E	H	D	C	A	G	F	B
3	F	D	G	B	H	A	C	E
4	B	A	F	H	G	C	E	D
* **5**	H	C	B	G	F	E	D	A
* **6**	G	E	H	F	B	D	A	C
* **7**	C	G	A	E	D	H	B	F

(b)

Figura 4.3: Exemplo de um movimento na vizinhança Troca Parcial de Times, com $r = 2$, $t_1 = A$ e $t_2 = B$.

4.3.3.4 Troca Parcial de Rodadas (TPR)

Assim como a vizinhança TPE é uma versão mais elaborada de TE, tem-se que a estrutura de vizinhança conhecida como *Troca Parcial de Rodadas* (TPR) é uma generalização de TR. Em vez de considerar a troca de posição entre duas rodadas r_1 e r_2, faz-se

a permutação de apenas alguns jogos entre as duas rodadas escolhidas.

Sejam alguma equipe t e duas rodadas r_1 e r_2 quaisquer. Seja ainda T o subconjunto de equipes de cardinalidade mínima incluindo t e para os quais os oponentes nas rodadas r_1 e r_2 são os mesmos. Um movimento na vizinhança TPR consiste em permutar os jogos das equipes em T nas rodadas r_1 e r_2.

Na Figura 4.4, ilustra-se um possível movimento na estrutuda de vizinhança sendo discutida, mostrando-se uma tabela hipotética para um torneio *round-robin* com $n = 10$ equipes e a tabela resultante do movimento que leva em consideração a equipe t e as rodadas r_1 e r_2. As equipes contidas em T estão destacadas com um asterisco. Os elementos que serão deslocados, nas colunas relativas às equipes em T, estão marcados em cinza escuro. Quando a modificação é efetivada, também é necessário atualizar as entradas da tabela para as demais equipes envolvidas. Essas últimas células estão representadas em cinza claro.

4.3.3.5 Rotação de Jogos (RJ)

A última estrutura de vizinhança utilizada é chamada de *Rotação de Jogos* (RJ) [45]. Essa vizinhança consiste em forçar a ocorrência de um determinado jogo em uma rodada específica, reparando-se a viabilidade da solução modificada logo em seguida. Para que a solução alterada seja novamente factível, é executado um processo que utiliza movimentos de *cadeias de ejeção* [19, 20]. Uma cadeia de ejeção consiste em uma sequência de movimentos compostos, onde modificações em um ou mais elementos da solução fazem com que outros elementos sejam "ejetados" do seu estado atual. Quando um elemento tem o seu valor modificado, pode-se tornar necessária a realização de um segundo movimento, que pode vir a encadear um terceiro e assim sucessivamente.

Um movimento na vizinhança RJ começa com a inserção de um jogo (t_1, t_2) em uma rodada r. Suponha-se que os jogos (t_1, t_3) e (t_2, t_4) pertençam a essa mesma rodada. Assim, as equipes t_1 e t_2 passariam a jogar duas vezes em r. Com isso, os jogos (t_1, t_3) e (t_2, t_4) são removidos dessa rodada, fazendo com que as equipes t_3 e t_4 não enfrentem nenhum oponente em r. O jogo (t_3, t_4) é inserido nessa rodada, fazendo com que ele ocorra duas vezes na competição considerada. Da mesma forma, os jogos (t_1, t_3) e (t_2, t_4) passaram a não mais existir no torneio. O procedimento continua recursivamente, reinserindo o jogo (t_1, t_3) na rodada que originalmente continha o jogo (t_3, t_4). Esse processo continua até que seja reinserido o jogo (t_2, t_4) e removido o jogo (t_1, t_2), que foi duplicado no primeiro passo da cadeia de ejeção.

	t		*	*		*	*	*		
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
$r_1 = 1$	C	I	A	F	J	D	H	G	B	E
2	J	H	D	C	G	I	E	B	F	A
3	G	F	I	J	H	B	A	E	C	D
$r_2 = 4$	H	E	F	G	B	C	D	A	J	I
5	E	C	B	H	A	J	I	D	G	F
6	B	A	J	E	D	G	F	I	H	C
7	D	J	G	A	I	H	C	F	E	B
8	I	D	H	B	F	E	J	C	A	G
9	F	G	E	I	C	A	B	J	D	H

(a)

	t		*	*		*	*	*		
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
$r_1 = 1$	H	I	F	G	J	C	D	A	B	E
2	J	H	D	C	G	I	E	B	F	A
3	G	F	I	J	H	B	A	E	C	D
$r_2 = 4$	C	E	A	F	B	D	H	G	J	I
5	E	C	B	H	A	J	I	D	G	F
6	B	A	J	E	D	G	F	I	H	C
7	D	J	G	A	I	H	C	F	E	B
8	I	D	H	B	F	E	J	C	A	G
9	F	G	E	I	C	A	B	J	D	H

(b)

Figura 4.4: Exemplo de um movimento na vizinhança Troca Parcial de Rodadas, com $t = A$, $r_1 = 1$ e $r_2 = 4$.

4.3.4 Busca local

A busca local implementada utilizou as quatro primeiras estruturas de vizinhança definidas na Seção 4.3.3. Seguindo a estratégia VND descrita na Seção 4.2, o procedimento considera a realização de movimentos do tipo mais aprimorante em cada uma das vizinhanças empregadas, na ordem mostrada a seguir:

- Troca de Equipes (TE)
- Troca de Rodadas (TR)
- Troca Parcial de Equipes (TPE)
- Troca Parcial de Rodadas (TPR)

4.3.5 Perturbações

Cada iteração da fase ILS inicia com uma perturbação da solução corrente S. Uma perturbação consiste de uma sequência de movimentos aleatórios dentro da vizinhança Rotação de Jogos (RJ). Para cada movimento, escolhe-se aleatoriamente duas equipes t_1 e t_2 e força-se a ocorrência do jogo disputado por eles em uma rodada r, também escolhida de forma aleatória.

No presente trabalho, também foram consideradas três formas diferentes de se escolher as equipes t_1 e t_2 e a rodada r iniciais de uma perturbação. Para isso, foi introduzindo um *bias* – tendência – com o intuito de fazer com que os movimentos em RJ não sejam totalmente aleatórios. Embora não seja possível saber a priori os efeitos de uma cadeia de ejeção completa no valor de COEV de uma solução, pode-se calcular facilmente o incremento (ou decremento) causado pelo primeiro passo da cadeia. Assim, os *bias* considerados são os seguintes:

1. As equipes t_1 e t_2 são escolhidas aleatoriamente, mas o jogo entre elas é forçado na rodada r que ocasionar o menor incremento inicial no COEV da solução;
2. A equipe t_2 é escolhida aleatoriamente, mas a equipe t_1 e a rodada r são escolhidos de forma a minimizar o incremento do primeiro movimento da cadeia de ejeção; e
3. As equipes t_1 e t_2 e a rodada r são escolhidos de forma que o incremento do primeiro movimento da cadeia de ejeção seja o menor possível.

No Capítulo 5, são apresentados os experimentos computacionais realizados. São mostrados os testes executados para guiar a escolha das estratégias e parâmetros da heurística proposta.

Capítulo 5

Experimentos computacionais

Neste capítulo, apresentam-se as definições de quatro classes de instâncias ponderadas, as quais dão origem a um conjunto de instâncias que representam competições disputadas por diversos números de equipes. As instâncias produzidas são utilizadas para a realização dos experimentos com o algoritmo proposto. Essas instâncias estão disponíveis para serem usadas como referência em trabalhos futuros, no endereço eletrônico `http://www.ic.uff.br/~celso/grupo/Weighted_carry-over_instances.zip`.

Serão vistos os testes executados na fase de ajustes de parâmetros. Também serão apresentados os experimentos computacionais realizados com a versão final do algoritmo proposto e será feita uma análise dos resultados obtidos.

Os testes computacionais realizados para o presente trabalho foram executados em uma máquina com processador AMD Athlon 64 X2 de 2.3 GHz e 1 GB de memória RAM. O código foi implementado na linguagem C++ e compilado com o compilador C/C++ do projeto GNU (GCC) versão 4.2.4 sob o sistema operacional Ubuntu Linux 8.04 com *kernel* 2.6.24.

5.1 Instâncias ponderadas

Para efeito da realização de testes com o algoritmo proposto, foram geradas diversas instâncias ponderadas para o problema em apreço. Na primeira linha de cada arquivo de dados, tem-se o número de equipes n que participam da competição representada. Em seguida, é dada uma matriz numérica W de dimensão $n \times n$. Cada entrada w_{ij} dessa matriz representa o peso atribuído ao efeito de *carry-over* dado pela equipe i à equipe j.

Nas seções seguintes, serão vistas quatro classes distintas de instâncias ponderadas, cada uma delas dando origem a um determinado conjunto de arquivos. Para as três

primeiras classes foram geradas instâncias com o valor de n variando de 4 a 20. As instâncias da última classe apresentada são baseadas em várias edições do campeonato brasileiro de futebol. Foram geradas instâncias para 20, 22 e 24 equipes, de acordo com a edição considerada desse torneio.

5.1.1 Instâncias aleatórias

Nas instâncias desta classe, o peso atribuído a cada efeito de *carry-over* dado por uma equipe a alguma outra é gerado de forma randômica. Considerando-se uma instância de uma competição com n equipes, a entrada w_{ij} da matriz de pesos é um número gerado aleatoriamente no intervalo $[1, 2n]$.

Para cada valor de n no conjunto $\{4, 6, 8, 10, 12, 14, 16, 18, 20\}$, foram geradas três instâncias distintas, identificadas pelas letras A, B e C. As instâncias dessa classe são nomeadas da seguinte forma: *inst<n>random<letra>*. Por exemplo, uma das instâncias com 18 equipes é denominada *inst18randomB*.

5.1.2 Instâncias lineares

Para cada instância desta classe, considera-se que existe uma classificação *a priori* das equipes participantes. Atribui-se uma *força* a cada um deles e então são calculados os pesos dos efeitos de *carry-over* cedidos. Sendo as equipes indexadas por $\{1, 2, \ldots, n\}$, atribui-se à equipe t_i uma força igual a i. Dessa forma, faz-se $w_{ij} = |\text{força}(t_i) - \text{força}(t_j)|$.

As instâncias dessa classe são nomeadas da seguinte forma: *inst<n>linear*. Por exemplo, a instância com 14 equipes é denominada *inst14linear*.

5.1.3 Instâncias lineares com perturbação

Para gerar esta classe de instâncias, são utilizadas como base as instâncias lineares descritas na seção anterior. Cada entrada da matriz de pesos de uma instância linear é somada a um valor gerado aleatoriamente no intervalo $[-n/2, n/2]$ e toma-se o módulo do valor resultante.

Assim como no caso das instâncias randômicas, foram geradas três instâncias distintas para cada valor par de n entre 4 e 20. As diferentes instâncias geradas para um mesmo valor de n são novamente identificadas pelas letras A, B e C. As instâncias dessa classe são nomeadas da seguinte forma: *inst<n>linearperturbacao<letra>*. Por exemplo, uma

das instâncias com 12 equipes é denominada *inst12linearperturbacaoA*.

5.1.4 Instâncias inspiradas em uma competição real

A presente classe de instâncias é composta por seis elementos. Baseando-se no campeonato brasileiro de futebol, cada instância é derivada de uma edição anual dessa competição. A classificação final das equipes é considerada e o número de pontos que cada equipe obteve é tomado como sua força. Assim, os pesos são calculados da mesma forma já vista para as instâncias lineares: $w_{ij} = |\text{força}(t_i) - \text{força}(t_j)|$.

As edições do campeonato brasileiro de futebol consideradas foram as disputadas entre os anos de 2003 e 2008, inclusive. Na edição do ano de 2008, o primeiro colocado (t_1) obteve 75 pontos e o segundo colocado (t_2) 72. Assim, na matriz de pesos relativa à instância do ano de 2008, tem-se $w_{12} = w_{21} = 75 - 72 = 3$.

As instâncias dessa classe são nomeadas da seguinte forma: *inst<n>brasileirao<ano>*. Por exemplo, a instância relativa ao campeonato brasileiro do ano de 2007, disputada por 20 equipes, é denominada *inst20brasileirao2007*.

5.2 Ajuste de parâmetros

Na presente seção, serão vistos os experimentos que foram realizados para guiar a definição dos parâmetros do algoritmo proposto. Em primeiro lugar, são vistos os testes realizados para definir se é interessante o uso de *bias* no procedimento de perturbação e, em caso positivo, qual deles escolher. Em seguida, são vistos o experimentos realizados para escolher o modo de geração das 1-fatorações iniciais. São testadas duas formas de inicialização: uma delas empregando somente o método do polígono; e outra que considera o uso desse último método e o algoritmo de de Werra, que gera 1-fatorações binárias quando n é múltiplo de quatro. Por último, os valores dos três principais parâmetros numéricos do algoritmo são ajustados.

5.2.1 *Bias* na perturbação

Na Seção 4.3.5, foi visto o funcionamento do algoritmo de perturbação. Foi visto também que existem quatro formas de empregar essa função: sem utilizar *bias* ou usando um de seus três tipos. Para verificar qual opção seria mais vantajosa, foram feitos testes para cada forma possível. Mantendo-se todos os demais parâmetros fixos, as quatro

variações foram testadas sobre instâncias não-ponderadas do problema em apreço. A matriz de pesos de uma instância desse tipo é composta somente por pesos unitários. Foram consideradas instâncias com n par variando de 4 a 20.

Na Tabela 5.1, têm-se os resultados obtidos para a versão sem *bias* do algoritmo. Os valores apresentados foram obtidos com cinco execuções independentes para cada instância. São mostrados os COEVs médios e o melhor COEV obtido para cada instância.

n	COEV médio	Melhor COEV
4	12.0	12.0
6	60.0	60.0
8	56.0	56.0
10	108.0	108.0
12	165.6	160.0
14	254.0	254.0
16	307.6	272.0
18	400.4	400.0
20	487.6	486.0
Média	205.7	200.89

Tabela 5.1: Resultados obtidos sem a utilização de *bias*.

Nas Tabelas 5.2, 5.3 e 5.4, são mostrados os resultados conseguidos com a utilização dos três tipo de bias considerados.

n	COEV médio	Melhor COEV
4	12.0	12.0
6	60.0	60.0
8	56.0	56.0
10	108.0	108.0
12	162.0	160.0
14	254.0	254.0
16	300.4	240.0
18	400.4	400.0
20	487.6	486.0
Média	204.5	197.3

Tabela 5.2: Resultados obtidos com a utilização do primeiro tipo de *bias*.

Na Tabela 5.5, são listados os valores médios e os melhores valores obtidos para cada uma das quatro estratégias consideradas. É possível verificar que os melhores resultados foram obtidos com a utilização do primeiro tipo de *bias*. Portanto, essa estratégia foi a escolhida para a realização das perturbações.

n	**COEV médio**	**Melhor COEV**
4	12.0	12.0
6	60.0	60.0
8	56.0	56.0
10	110.4	108.0
12	167.6	164.0
14	254.0	254.0
16	300.4	240.0
18	400.4	400.0
20	487.6	486.0
Média	205.4	197.8

Tabela 5.3: Resultados obtidos com a utilização do segundo tipo de *bias*.

n	**COEV médio**	**Melhor COEV**
4	12.0	12.0
6	60.0	60.0
8	56.0	56.0
10	114.0	114.0
12	172.8	172.0
14	254.0	254.0
16	316.4	314.0
18	400.4	400.0
20	487.6	486.0
Média	208.1	207.6

Tabela 5.4: Resultados obtidos com a utilização do terceiro tipo de *bias*.

Estratégia	**COEV médio**	**Melhor COEV**
Sem *bias*	205.7	200.9
Primeiro tipo	204.5	197.3
Segundo tipo	205.4	197.8
Terceiro tipo	208.1	207.6

Tabela 5.5: Síntese dos resultados obtidos com a utilização dos vários tipos de *bias*.

5.2.2 Parâmetros numéricos e 1-fatorações iniciais

A estratégia utilizada para gerar as 1-fatorações iniciais influencia sensivelmente a qualidade das soluções encontradas. Foram testadas duas alternativas diferentes para a obtenção da 1-fatoração inicial quando n é multiplo de quatro. Na primeira, utiliza-se somente o método do polígono. Na segunda, o método do polígono e o método de fatoração binária (algoritmo de de Werra [11]), descritos na Seção 4.3.1 são empregados indistintamente para criar 1-fatorações – um por vez.

Foi observado em resultados preliminares que os principais parâmetros que influenciam o comportamento da busca na fase ILS são o número máximo de movimentos deteriorantes aceitos antes do término dessa etapa, o número de movimentos na vizinhança RJ aplicados em cada perturbação e o valor inicial do parâmetro b. Também foi observado que escolhas razoáveis para esses parâmetros são:

$$\begin{aligned} \text{número máximo de movimentos deteriorantes:} &\quad \{10, 100, 200\} \\ \text{número de movimentos na vizinhança RJ aplicados na perturbação:} &\quad \{1, 5, 10\} \\ \text{valor inicial de } b\text{:} &\quad \{0.01, 0.05, 0.10\} \end{aligned} \tag{5.1}$$

Para verificar o comportamento do algoritmo proposto com relação aos valores dos parâmetros mencionados, foram executados testes com três instâncias não-ponderadas onde n era múltiplo de quatro. Para cada uma dessas instâncias, testaram-se todas as 27 combinações possíveis dos parâmetros descritos acima com cada uma das duas alternativas de geração das 1-fatorações iniciais. Para cada configuração, foram tomadas as médias dos resultados obtidos em três execuções independentes.

Inicialmente, será discutida a melhor estratégia de geração das 1-fatorações iniciais. A Tabela 5.6 mostra os resultados obtidos por cada uma das duas alternativas para instâncias de tamanhos 12, 16 e 20: valor médio de COEV, melhor valor de COEV, tempo computacional médio (em segundos), e maior tempo computacional (em segundos). Os valores apresentados foram obtidos sobre três execuções para cada combinação de parâmetros. A parte superior dessa tabela mostra os resultados para a primeira alternativa, baseada exclusivamente no método do polígono. A parte inferior relata os resultados para a segunda alternativa, a qual emprega ambos os métodos considerados (método do polígono e método de de Werra). Embora a primeira alternativa (parte superior) apresente tempos computacionais menores, a segunda (parte inferior) mostra-se capaz de encontrar soluções melhores (ou comparáveis). Consequentemente, a segunda alternativa foi

selecionada para a geração das 1-fatorações iniciais.

Estratégia	n	COEV (méd.)	COEV (melhor)	Tempo (méd.)	Tempo (máx.)
fatorações	12	192.000	192.000	3.519	4.000
Canônicas	16	306.914	304.444	178.765	183.296
	20	489.852	488.444	20.790	21.296
Ambas as	12	167.630	164.370	64.790	69.111
fatorações	16	267.630	258.519	244.827	263.926
	20	492.000	490.370	54.481	62.481

Tabela 5.6: Valores de COEV e tempos computacionais em segundos para duas estratégias diferentes de geração das 1-fatorações iniciais.

Para escolher a configuração de parâmetros mais apropriada, executou-se o algoritmo proposto com todas as 27 combinações possíveis de valores. Foram coletados os resultados máximos e mínimos obtidos nas três execuções, para cada instância testada. Para cada uma dessas instâncias, os resultados encontrados com cada combinação foram normalizados para o intervalo $[0, 1]$. Dessa forma, a melhor combinação para uma determinada instância teria seu resultado normalizado em 0, enquanto o resultado da pior combinação seria normalizado em 1. Em seguida, foram computados os resultados normalizados médios sobre todas as instâncias consideradas, de modo a obter-se um valor que indique a performance geral de cada combinação testada.

Na Figura 5.1, tem-se um gráfico que mostra os COEVs normalizados médios e os tempos normalizados médios para cada configuração de parâmetros. Da mesma forma, o gráfico da Figura 5.2 mostra os melhores COEVS normalizados e os maiores tempos normalizados para cada combinação de parâmetros. A criação e a análise dos gráficos são inspiradas no trabalho feito em [7].

Cada ponto desses gráficos representa uma combinação única de valores dos parâmetros. As cinco melhores combinações, mais próximas do ponto ideal $(0, 0)$, são as mesmas nos dois gráficos. Os pontos associados são identificados e rotulados com os valores dos parâmetros correspondentes, nesta ordem: número máximo de movimentos deteriorantes aceitos antes do término da fase ILS; o número de movimentos na vizinhança RJ executados no procedimento de perturbação; e o valor inicial de b.

Como quatro das cinco melhores combinações aceitam um máximo de 200 movimentos deteriorantes antes do término da fase ILS, esse valor foi escolhido para o primeiro parâmetro. O número de movimentos feitos em uma perturbação é fixado em um, haja vista que três dos cinco melhores pontos correspondem a esse valor. A melhor escolha para o valor inicial do parâmetro b é menos evidente a partir desses resultados. Com o

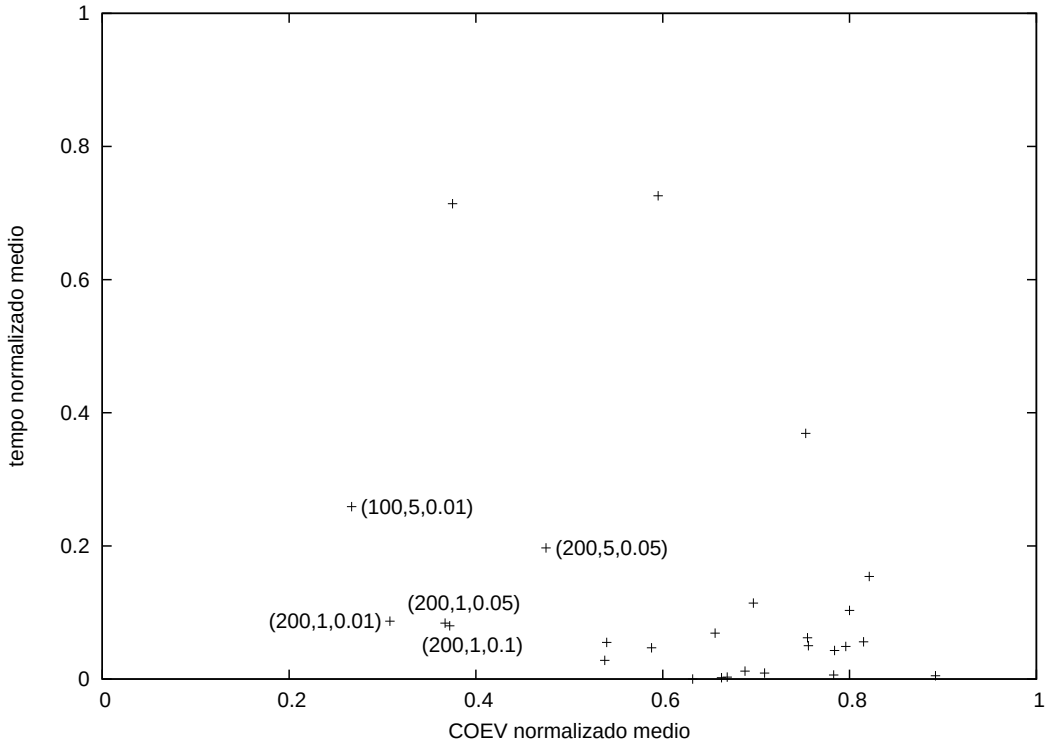

Figura 5.1: Resultados normalizados médios obidos com diferentes combinações de parâmetros.

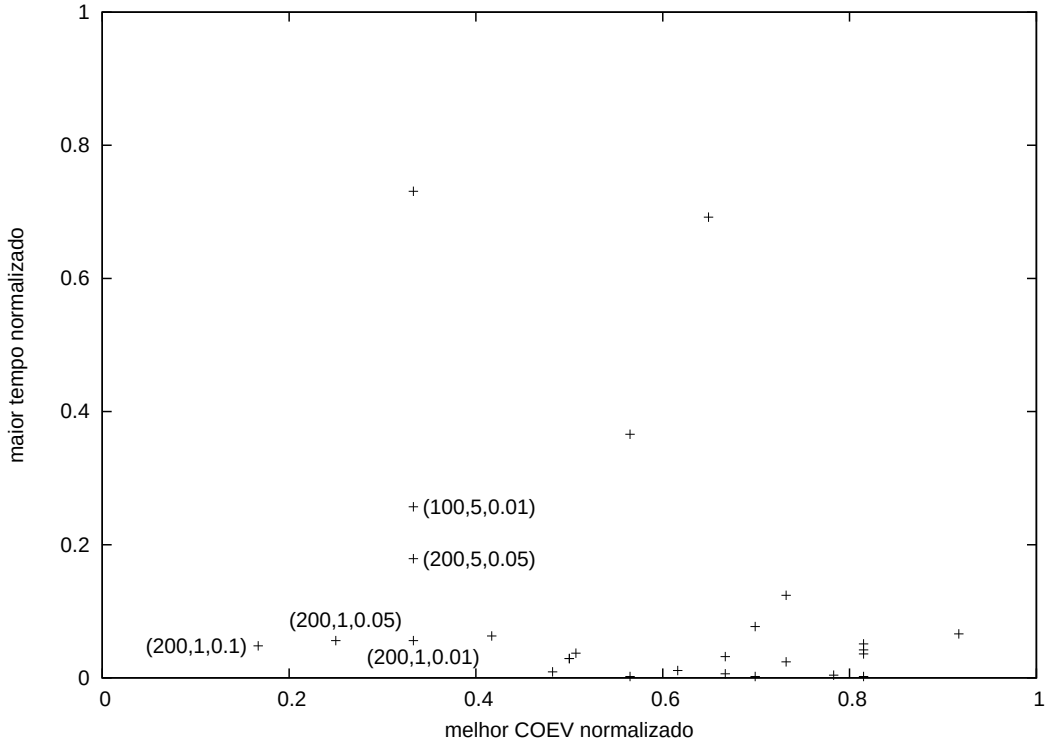

Figura 5.2: Resultados normalizados extremos obtidos com diferentes combinações de parâmetros.

fim de privilegiar a robustez do algoritmo, selecionou-se $b = 0.01$. Isso é feito porque o ponto $(200, 1, 0.01)$ apresenta-se como a melhor opção no gráficos das médias.

5.3 Resultados

A seguir, são apresentadas as tabelas com os resultados encontrados, para cada classe de instâncias. Para a obtenção dos resultados finais, a heurística híbrida proposta foi executada cinco vezes para cada instância considerada, utilizando os parâmetros definidos nas seções anteriores. Os seguintes resultados são relatados nas tabelas a seguir: COEV médio, melhor COEV, tempo médio (em segundos) e pior tempo (em segundos), obtidos nas cinco execuções do algoritmo.

Os resultados para as instâncias aleatórias são apresentados na Tabela 5.7. Essas instâncias apresentaram os menores tempos computacionais entre todas as classes. Foi observado que a fase ILS do algoritmo proposto foi sempre capaz de melhorar as soluções encontradas na fase multi-partida, com exceção das instâncias com os três menores valores de n.

Em seguida, são considerados os resultados para as instâncias lineares, mostrados na Tabela 5.8. A fase ILS foi capaz de melhorar as soluções encontradas na fase multi-partida, para toda instância com $n \geq 10$ equipes. Os tempos computacionais são bem maiores do que os observados para a classe anterior. Isso parece ser particularmente verdade para a instância com 20 equipes.

Os resultados para as instâncias lineares perturbadas são mostradas na Tabela 5.9. Assim como nas duas classes anteriores, a fase ILS conseguiu melhorar todas as melhores soluções obtidas na fase multi-partida, com exceção dos três menores valores de n. Os tempos computacionais observados são maiores do que aqueles observados para as instâncias aleatórias e lineares.

Na Tabela 5.10, são mostrados os resultados obtidos para as instâncias inspiradas em edições do campeonato brasileiro de futebol. Essa classe apresenta os maiores tempos computacionais para as instâncias com 20 equipes. Isso parece ser devido ao fato de que os pesos dessa classe são mais diversificados do que em qualquer outra.

Finalmente, na Tabela 5.11, são apresentados os resultados para as instâncias não-ponderadas, obtidos sobre cinco execuções distintas. Essas instâncias são equivalentes a instâncias ponderadas com pesos unitários. Resolver essas instâncias é o mesmo que

Instância	**COEV médio**	**Melhor COEV**	**Tempo médio (s)**	**Maior tempo (s)**
inst4randomA	63.0	63.0	0.2	1.0
inst4randomB	53.0	53.0	0.2	1.0
inst4randomC	45.0	45.0	0.0	0.0
inst6randomA	233.0	233.0	0.8	1.0
inst6randomB	274.0	274.0	0.8	1.0
inst6randomC	235.0	235.0	1.2	2.0
inst8randomA	505.0	505.0	7.2	8.0
inst8randomB	495.0	495.0	18.6	20.0
inst8randomC	470.0	470.0	13.8	14.0
inst10randomA	912.6	895.0	139.4	150.0
inst10randomB	793.0	781.0	151.0	173.0
inst10randomC	744.6	737.0	149.0	172.0
inst12randomA	1549.2	1521.0	453.2	560.0
inst12randomB	1520.2	1489.0	402.4	451.0
inst12randomC	1594.4	1572.0	455.8	546.0
inst14randomA	2620.0	2608.0	39.8	44.0
inst14randomB	2903.8	2870.0	38.0	45.0
inst14randomC	2777.6	2760.0	36.8	42.0
inst16randomA	3812.2	3756.0	2605.8	2878.0
inst16randomB	3846.6	3797.0	2848.2	3294.0
inst16randomC	3773.6	3755.0	3138.0	3483.0
inst18randomA	5574.4	5515.0	1381.2	2442.0
inst18randomB	5816.2	5745.0	792.4	1271.0
inst18randomC	5598.0	5548.0	1547.2	3157.0
inst20randomA	7764.4	7760.0	136.8	145.0
inst20randomB	7928.0	7888.0	147.8	174.0
inst20randomC	7700.2	7636.0	136.8	144.0
Médias	2577.8	2555.8	542.3	711.8

Tabela 5.7: Resultados para as instâncias aleatórias.

Instância	**COEV médio**	**Melhor COEV**	**Tempo médio (s)**	**Maior tempo (s)**
inst4linear	20.0	20.0	0.2	1.0
inst6linear	114.0	114.0	0.6	1.0
inst8linear	168.0	168.0	8.4	9.0
inst10linear	318.0	318.0	64.6	70.0
inst12linear	504.0	496.0	271.4	309.0
inst14linear	960.0	958.0	18.6	20.0
inst16linear	1088.0	1076.0	2007.2	2313.0
inst18linear	1698.0	1660.0	4102.0	4729.0
inst20linear	2257.6	2212.0	5213.0	5624.0
Médias	792.0	780.2	1298.4	1452.9

Tabela 5.8: Resultados para as instâncias lineares.

Instância	COEV médio	Melhor COEV	Tempo médio (s)	Maior tempo (s)
inst4linearperturbacaoA	16.0	16.0	0.2	1.0
inst4linearperturbacaoB	17.0	17.0	0.2	1.0
inst4linearperturbacaoC	22.0	22.0	0.0	0.0
inst6linearperturbacaoA	68.0	68.0	1.4	2.0
inst6linearperturbacaoB	73.0	73.0	0.6	1.0
inst6linearperturbacaoC	60.0	60.0	0.8	1.0
inst8linearperturbacaoA	137.0	137.0	31.0	32.0
inst8linearperturbacaoB	141.0	141.0	22.4	23.0
inst8linearperturbacaoC	162.0	162.0	26.8	28.0
inst10linearperturbacaoA	329.0	326.0	136.0	162.0
inst10linearperturbacaoB	277.0	274.0	134.0	163.0
inst10linearperturbacaoC	291.8	284.0	128.6	152.0
inst12linearperturbacaoA	601.2	587.0	484.2	638.0
inst12linearperturbacaoB	528.6	525.0	494.6	580.0
inst12linearperturbacaoC	486.6	478.0	546.6	648.0
inst14linearperturbacaoA	940.0	920.0	42.8	45.0
inst14linearperturbacaoB	947.4	932.0	39.8	43.0
inst14linearperturbacaoC	993.2	990.0	36.2	40.0
inst16linearperturbacaoA	1403.0	1376.0	3432.8	3905.0
inst16linearperturbacaoB	1360.2	1348.0	3343.8	3668.0
inst16linearperturbacaoC	1118.0	1098.0	3961.0	4683.0
inst18linearperturbacaoA	2063.4	2005.0	5162.4	6983.0
inst18linearperturbacaoB	1965.4	1921.0	4038.0	5343.0
inst18linearperturbacaoC	1712.0	1585.0	4833.2	5805.0
inst20linearperturbacaoA	3092.2	3065.0	482.0	1304.0
inst20linearperturbacaoB	2866.8	2800.0	2923.4	4719.0
inst20linearperturbacaoC	2837.2	2791.0	1479.6	3509.0
Médias	907.74	888.9	1177.1	1573.3

Tabela 5.9: Resultados para as instâncias lineares com perturbação.

Instância	COEV médio	Melhor COEV	Tempo médio (s)	Maior tempo (s)
inst24brasileirao2003	7730.4	7542.0	13897.8	15755.0
inst24brasileirao2004	7179.6	7088.0	12992.8	13468.0
inst22brasileirao2005	5228.8	5158.0	10599.0	13959.0
inst20brasileirao2006	5310.0	5236.0	5705.4	6358.0
inst20brasileirao2007	4876.0	4834.0	2715.8	3655.0
inst20brasileirao2008	4045.6	3944.0	6805.6	8308.0
Médias	5728.4	5633.7	8786.1	10250.5

Tabela 5.10: Resultados para as instâncias inspiradas no campeonato brasileiro de futebol.

resolver a versão original, não-ponderada, do problema em apreço. Na Tabela 5.12, são mostrados os resultados obtidos pelo algoritmo proposto e os melhores resultados encontrados na literatura [2], para $4 \leq n \leq 20$. O melhor resultado conhecido para $n = 12$ foi atualizado, passando de 176 para 160, representando uma melhora de quase 10% no valor da melhor solução conhecida até então..

n	COEV médio	Melhor COEV	Tempo médio (s)	Maior tempo (s)
4	12.0	12.0	0.2	1.0
6	60.0	60.0	0.4	1.0
8	56.0	56.0	5.4	6.0
10	111.6	108.0	12.0	13.0
12	164.8	160.0	24.6	26.0
14	254.0	254.0	8.8	9.0
16	259.2	240.0	129.8	140.0
18	401.6	400.0	28.8	31.0
20	491.2	486.0	65.2	72.0
Médias	201.2	197.3	30.6	33.2

Tabela 5.11: Resultados para as instâncias não-ponderadas.

n	Melhor COEV conhecido	Algoritmo proposto
4	12	12
6	60	60
8	56	56
10	108	108
12	176	160
14	234	254
16	240	240
18	340	400
20	380	486

Tabela 5.12: Resultados obtidos com a heurística híbrida proposta e melhores valores conhecidos na literatura, para as instâncias não-ponderadas.

Capítulo 6

Conclusões e trabalhos futuros

O presente trabalho abordou o Problema de Minimização de Efeitos Ponderados de *Carry-Over* em torneios *round-robin*. Esse problema foi concebido como uma variação do problema de minimização de efeitos de *carry-over*, considerando-se a associação de pesos aos elementos da função objetivo desse último.

Foram apresentadas algumas aplicações possíveis do valor de efeitos de *carry-over* como um critério de justiça na criação de boas tabelas para torneios *round-robin*. Também foram discutidas razões pelas quais a versão ponderada desse valor seria mais apropriada para modelar situações que ocorrem na prática.

Várias formas de gerar tabelas de torneios *round-robin* foram apresentadas, utilizando-se objetos matemáticos conhecidos como 1-fatorações. Também foram propostas duas formas de utilizar seus 1-fatores para formar soluções viáveis para o problema em apreço. Cinco estruturas de vizinhança foram implementadas e adaptadas para o problema estudado, sendo quatro delas utilizadas no procedimento de busca local e a última empregada na realização de perturbações.

Nessa dissertação, também foi proposta uma formulação matemática por programação inteira 0-1 para o Problema de Minimização de Efeitos Ponderados de *Carry-Over*.

Para a resolução do problema estudado, foi proposta uma abordagem baseada na metaheurística *Iterated Local Search*. Com ela, foi possível obter uma nova melhor solução para a instância do problema original com $n = 12$. Também foi possível verificar a influência da utilização de mais de uma estratégia para gerar 1-fatorações no processo de busca. Além disso, foram propostas três novas maneiras de se utilizar a vizinhança *Rotação de Jogos* na aplicação de perturbações, considerando-se a influência de cada uma delas nos resultados do algoritmo proposto.

Para a execução dos experimentos computacionais, foram propostas quatro classes de

instâncias ponderadas. Arquivos de instâncias de *benchmark* foram gerados e disponibilizados na internet, podendo vir a servir como referência em outras pesquisas.

Para a realização de trabalhos futuros, uma primeira idéia seria a consideração de novas variantes do problema abordado. Inicialmente, pode-se considerar uma função-objetivo do tipo *min-max* para o problema, onde se desejaria minimizar a quantidade máxima de efeitos de *carry-over* que cada equipe dá às demais. Em outras palavras, seria calculado um valor de COEV individual para cada equipe e o valor máximo entre todos eles deveria ser minimizado.

Um outro caminho que parece promissor é a utilização de novas formas de gerar 1-fatorações. Além disso, a consideração de novos métodos construtivos e estruturas de vizinhança também seria uma interessante linha de estudo.

Outra oportunidade de melhoria no algoritmo proposto seria através do uso de algum tipo de memória, fazendo com que uma sequência de execução pudesse aproveitar informações das sequências anteriores.

Por fim, uma outra modificação interessante seria utilizar uma estratégia do tipo VNS, onde as perturbações fossem geradas em vizinhanças progressivamente maiores.

Referências

[1] Anagnostopoulos, A., Michel, L., Hentenryck, P. V., Vergados, Y. A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling 9* (2006), 177–193.

[2] Anderson, I. *Combinatorial Designs and their Applications*. CRC Research Notes in Mathematics. Chapman & Hall, 1999, cap. Balancing carry-over effects in tournaments, p. 1–16.

[3] Anderson, I., Bailey, R. A. Completeness properties of conjugates of latin squares based on groups, and an application to bipartite tournaments. *Bulletin of the Institute of Combinatorics and its Applications 21* (1997), 95–99.

[4] Benoist, T., Laburthe, F., Rottembourg, B. Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. Em *Proceedings of the Third International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (Ashford, 2001), p. 15–26.

[5] Bewersdorff, J. *Galois Theory for Beginners: A Historical Perspective*. American Mathematical Society, 2006.

[6] Briskorn, D. Scheduling sport leagues using branch-and-price. Em *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling* (Brno, 2006), E. Burke e H. Rudová, Eds., p. 367–369.

[7] Buriol, L., França, P. M., Moscato, P. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics 10* (2004), 483–506.

[8] Burke, E., de Werra, D., Kingston, J. *Handbook of Graph Theory*. CRC Press, 2004, cap. Applications to timetabling, p. 445–474.

[9] Costa, F. N., Urrutia, S., Ribeiro, C. C. An ILS heuristic for the traveling tournament problem with fixed venues. Em *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling* (Montréal, 2008), E. K. Burke e M. Gendreau, Eds.

[10] Croce, F. D., Oliveri, D. Scheduling the Italian football league: An ILP-based approach. *Computers and Operations Research 33* (2006), 1963–1974.

[11] de Werra, D. Geography, games and graphs. *Discrete Applied Mathematics 2* (1980), 327–337.

[12] de Werra, D. *Studies on Graphs and Discrete Programming*. North-Holland, 1981, cap. Scheduling in sports, p. 381–395.

[13] Dinitz, J. H. *The CRC handbook of Combinatorial Designs*. CRC Press, Boca Raton, 1996, cap. Starters, p. 467–473.

[14] Dinitz, J. H., Lamken, E. R., Wallis, W. D. *The CRC Handbook of Combinatorial Designs*. CRC Press, Boca Raton, 1996, cap. Scheduling a Tournament, p. 578–591.

[15] Easton, K., Nemhauser, G., Trick, M. Solving the traveling tournament problem: A combined integer programming and constraint programming approach. Em *Practice and Theory of Automated Timetabling IV* (2003), G. Goos, J. Hartmanis, e J. van Leeuwen, Eds., vol. 2740 de *Lecture Notes in Computer Science*, Springer, p. 100–109.

[16] Easton, K., Nemhauser, G., Trick, M. Sports scheduling. Em *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, J. Leung, Ed. CRC Press, 2004, p. 52.1–52.19.

[17] Easton, K., Nemhauser, G., Trick, M. A. The travelling tournament problem: Description and benchmarks. Em *Principles and Practice of Constraint Programming*, T. Walsh, Ed., vol. 2239 de *Lecture Notes in Computer Science*. Springer, 2001, p. 580–585.

[18] Gaspero, L., Schaerf, A. A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics 13* (2007), 189–207.

[19] Glover, F. New ejection chain and alternating path methods for traveling salesman problems. Em *Computer Science and Operations Research:*, R. Sharda, O. Balci, e S. Zenios, Eds. Elsevier, 1992, p. 449–509.

[20] Glover, F. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics 65* (1996), 223–253.

[21] Goossens, D., Spieksma, F. Scheduling the Belgian soccer league. Em *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling* (Brno, 2006), E. Burke e H. Rudová, Eds., p. 420–422.

[22] Gutin, G., Punnen, P., Eds. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[23] Hansen, P., Mladenović, N. Variable neighborhood search. Em *Handbook of Metaheuristics*, F. Glover e G. Kochenberger, Eds. Kluwer Academic Publishers, 2002.

[24] Hentenryck, P., Vergados, Y. Minimizing breaks in sport scheduling with local search. Em *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling* (Monterey, 2005), S. Biundo, K. Myers, e K. Rajan, Eds., p. 22–29.

[25] Henz, M. Constraint-based round robin tournament planning. Em *International Conference on Logic Programming* (New Mexico, 1999), D. D. Schreye, Ed., p. 545–557.

[26] Henz, M. Scheduling a major college basketball conference – revisited. *Operations Research 49* (2001), 163–168.

[27] Henz, M., Müller, T., Thiel, S. Global constraints for round robin tournament scheduling. *European Journal of Operational Research 153* (2004), 92–101.

[28] Keedwell, A. D. Construction, properties and application of finite neofields. *Comment. Math. Univ. Carolinae 41* (2000), 283–297.

[29] Kendall, G. Scheduling english footbal fixtures over holiday periods. *Journal of the Operational Research Society 59* (2008), 743–755.

[30] Kendall, G., Knust, S., Ribeiro, C. C., Urrutia, S. Scheduling in sports: An annotated bibliography, 2008.

[31] Kirkman, T. On a problem in combinations. *Cambridge Dublin Math Journal 2* (1847), 191–204.

[32] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B., Eds. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[33] Lourenço, H. R., Martin, O. C., Stutzle, T. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003, cap. Iterated Local Search, p. 321–353.

[34] Martin, O. C., Otto, S. W., Felten, E. W. Large-step Markov chains for the traveling salesman problem. *Complex Systems 5* (1991), 299–326.

[35] Melo, R. A. Modelos de programação inteira para o problema do torneio com viagens com estádios fixos. Dissertação de Mestrado, Universidade Federal Fluminense, Brasil, Setembro 2007.

[36] Mendelsohn, E., Rosa, A. One-factorizations of the complete graph: A survey. *Journal of Graph Theory 9* (1985), 43–65.

[37] Miyashiro, R., Matsui, T. Minimizing the carry-over effects value in a round robin tournament. Em *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling* (Brno, 2006), p. 460–463.

[38] Mladenovic, N., Hansen, P. Variable neighborhood search. *Computers and Operations Research 34* (1997), 1097–1100.

[39] Nemhauser, G., Trick, M. Scheduling a major college basketball conference. *Operations Research 46* (1998), 1–8.

[40] Noronha, T. F., Ribeiro, C. C., Duran, G., Souyris, S., Weintraub, A. A branch-and-cut algorithm for scheduling the highly-constrained chilean soccer tournament. Em *Lecture Notes in Computer Science*, vol. 3867. Springer, Berlin, 2007, p. 174–186.

[41] Plummer, M. *Handbook of Graph Theory*. CRC Press, 2004, cap. Factors and Factorizations, p. 445–474.

[42] Rasmussen, R. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research 185* (2008), 795–810.

[43] Rasmussen, R., Trick, M. Round robin scheduling - A survey. *European Journal of Operational Research 188* (2008), 617–636.

[44] Régin, J.-C. Minimization of the number of breaks in sports scheduling problems using constraint programming. Em *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Constraint Programming and Large Scale Discrete Optimization*, E. Freuder e R. Wallace, Eds., vol. 57. American Mathematical Society, 2000, p. 115–130.

[45] Ribeiro, C., Urrutia, S. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research 179* (2007), 775–787.

[46] Ribeiro, C. C., Urrutia, S. Scheduling the Brazilian soccer tournament with fairness and broadcast objectives. Em *Practice and Theory of Automated Timetabling VI*, vol. 3867 de *Lecture Notes in Computer Science*. Springer, Berlin, 2007, p. 147–157.

[47] Russel, K. G. Balancing carry-over effects in round robin tournaments. *Biometrika 67* (1980), 127–131.

[48] Russel, R. A., Leung, J. Devising a cost effective schedule for a baseball league. *Operations Research 42* (1994), 614–625.

[49] Russell, R., Urban, T. A constraint programming approach to the multiple-venue, sport-scheduling problem. *Computers and Operations Research 33* (2006), 1895–1906.

[50] Trick, M. A. A schedule-then-break approach to sports timetabling. Em *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III* (Berlin, 2000), E. Burke e W. Erben, Eds., vol. 2079 de *Lecture Notes in Computer Science*, Springer, p. 242–253.

[51] Urrutia, S., Ribeiro, C. C. Maximizing breaks and bounding solutions to the mirrored traveling tournament problem. *Discrete Applied Mathematics 154* (2006), 1932–1938.

[52] Wallis, W. One-factorizations of the complete graph. Em *Contemporary Design Theory: A Collection of Surveys*, J. Dinitz e D. Stinson, Eds. Wiley, 1992, p. 593–639.