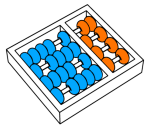
Universidade Estadual de Campinas
Instituto de Computação



Lucas de Oliveira

O problema dos árbitros viajantes: complexidade, modelagem e algoritmos

CAMPINAS
2016

Lucas de Oliveira

O problema dos árbitros viajantes: complexidade, modelagem e algoritmos

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Cid Carvalho de Souza
Coorientador: Prof. Dr. Tallys Hoover Yunes

Este exemplar corresponde à versão final da Tese defendida por Lucas de Oliveira e orientada pelo Prof. Dr. Cid Carvalho de Souza.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): CNPq, 142278/2013-0; CAPES, 01-P-01965-2012

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

OL4p

Oliveira, Lucas de, 1987-
O problema dos árbitros viajantes : complexidade, modelagem e algoritmos / Lucas de Oliveira. – Campinas, SP : [s.n.], 2016.

Orientador: Cid Carvalho de Souza.
Coorientador: Tallys Hoover Yunes.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Esportes - Planejamento. 2. Complexidade computacional. 3. Programação inteira. 4. Heurística. 5. Otimização combinatória. I. Souza, Cid Carvalho de,1963-. II. Yunes, Tallys Hoover. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: The traveling umpire problem : complexity, modeling and algorithms
Palavras-chave em inglês:
Sports - Planning
Computational complexity
Integer programming
Heuristic
Combinatorial optimization
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Cid Carvalho de Souza [Orientador]
Celso da Cruz Carneiro Ribeiro
Sebastián Alberto Urrutia
Fábio Luiz Usberti
Pedro Jussieu de Rezende
Data de defesa: 07-12-2016
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Lucas de Oliveira

O problema dos árbitros viajantes: complexidade, modelagem e algoritmos

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza (Orientador)
  Instituto de Computação - UNICAMP
- Prof. Dr. Celso da Cruz Carneiro Ribeiro
  Instituto de Computação - UFF
- Prof. Dr. Sebastián Alberto Urrutia
  Departamento de Ciência da Computação - UFMG
- Prof. Dr. Fábio Luiz Usberti
  Instituto de Computação - UNICAMP
- Prof. Dr. Pedro Jussieu de Rezende
  Instituto de Computação - UNICAMP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 07 de dezembro de 2016

Dedicatória

Dedico esta tese à minha esposa, Aline, aos meus pais, Maria e Wilson, aos meus irmãos, Lucimar, Renê, Tarciso e Vanderlúcia, e aos meus cunhados, Áriston, José Aloísio e Mônica, por sempre estarem presentes em todos os momentos da minha vida, e por terem me dado a chance de realizar esse sonho. Amo todos vocês!

Agradecimentos

Primeiramente, agradeço à minha família por todo apoio e amor incondicional. Por serem tão solidários e afetuosos, por sempre acreditarem em mim e estarem presentes em cada momento dessa conquista.

À minha esposa Aline, por todo amor, carinho e atenção. O seu incentivo, dedicação e compreensão foram fundamentais para concluir este trabalho. Você é a fonte de toda minha alegria e inspiração.

Ao meu orientador Cid, por ser paciente e por me mostrar o caminho para que eu me tornasse cada vez melhor tanto nos estudos quanto como pessoa.

Ao meu coorientador Tallys, pelos valiosos conselhos e ensinamentos que contribuíram muito para a minha formação e serão levados por toda a minha vida.

Ao meu orientador de graduação Claudio, por ter guiado meus primeiros passos na vida acadêmica e me mostrado o caminho para chegar até essa realização.

Aos meus amigos Ednaldo e Rilson, que me proporcionam ótimos momentos que sempre terei como lembranças.

Aos meus sobrinhos Eduardo, Luíza, Pedro, Victor e Vitória, que vivem pegando no meu pé e criando situações engraçadas que trazem mais alegria para minha vida mesmo nas horas mais difíceis.

Aos professores e funcionários do Instituto da Computação da UNICAMP.

Aos membros da banca examinadora Celso, Fábio, Pedro e Sebastián, que ajudaram no aperfeiçoamento desta tese.

Ao CNPq e Capes, pelo suporte financeiro.

Enfim, agradeço de coração a todos que participaram desta conquista.

Muito obrigado!

Resumo

Estudamos nesta tese um problema de otimização da área de esportes denominado problema dos árbitros viajantes (TUP, do inglês *traveling umpire problem*). O TUP é um problema abstrato baseado no problema real que a Liga Profissional de Beisebol dos Estados Unidos enfrenta todos os anos ao preparar o calendário de jogos com seus respectivos árbitros. O TUP recebe como entrada um torneio *round robin* duplo e tem como objetivo atribuir árbitros às partidas deste torneio minimizando a distância total viajada por eles durante toda a competição. São impostas neste problema restrições que impedem que qualquer árbitro apite mais de um jogo em um mesmo local, ou de um mesmo time, durante determinados intervalos de rodadas consecutivas, e exige que cada árbitro apite um jogo de cada time em sua casa ao menos uma vez. Desde a sua proposição, o TUP tem se revelado um problema de elevado grau de dificuldade. Nesta tese, demonstramos que o TUP é um problema $\mathcal{NP}$-completo, fechando esta questão em relação à sua complexidade que ficou em aberto durante sete anos. Também introduzimos duas novas formulações matemáticas e uma heurística *relax-and-fix* para este problema. As análises de resultados computacionais comprovam que as formulações matemáticas e a heurística *relax-and-fix* produzem limitantes inferiores e superiores bastante competitivos para o TUP, melhorando diversos resultados da literatura.

Abstract

We study in this thesis an optimization problem in sports area known as traveling umpire problem (TUP). The TUP is an abstract problem based on a real problem faced by the Major League Baseball every year when preparing the game schedule with their respective umpires. The TUP takes as input a double round robin tournament and the goal is to assign umpires to the games of this tournament while minimizing the total distance traveled by them throughout the entire competition. This problem imposes constraints forbidding any umpire to referee more than one game in a same venue, or of a same team, over a certain interval of consecutive rounds, and requiring that each umpire referees at least one game of each team in its home venue. Since its proposition, TUP has revealed itself to be a very difficult problem. In this thesis, we demonstrate that TUP is an $\mathcal{NP}$-complete problem, closing this question relative to its complexity that remained unsolved for seven years. Also, we introduce two new mathematical formulations and a relax-and-fix heuristic for this problem. The analyses of computational results attest that the mathematical formulations and the relax-and-fix heuristic yield very competitive lower and upper bounds for the TUP, improving many results known in the literature.

Lista de Abreviações e Siglas

ASP: *Answer set programming* (programação de conjunto de resposta).
BB: *Branch-and-bound* proposto em [56].
BB-LID: *Branch-and-bound* com limitantes inferiores baseados em decomposição proposto em [43].
BC: *Branch-and-cut* apresentado na seção 5.3.3.
BP: *Branch-and-price* proposto em [42].
BPC: *Branch-and-price-and-cut* proposto em [56].
FFR: Formulação baseada em fluxo em rede proposta em [56].
FPC: Formulação baseada em partição de conjuntos proposta em [42] e [56].
FPCCA: Formulação baseada em partição de conjuntos com cortes adicionais proposta em [56].
IDS: *Iterative deepening search* (busca em profundidade iterativa).
ILS: *Iterated local search* (busca local iterativa).
MD3: Método de decomposição proposto em [51] limitado em 3 horas de execução.
MD+: Método de decomposição proposto em [51] limitado em mais de 3 horas de execução.
MLB: *Major League Baseball* (Liga Profissional de Beisebol).
NFL: *National Football League* (Liga Nacional de Futebol Americano).
RF: Heurística *relax-and-fix* apresentada na seção 4.2.
TTP: *Traveling tournament problem* (problema do torneio viajante).
TUP: *Traveling umpire problem* (problema dos árbitros viajantes).

Sumário

Capítulo 1

Introdução

Nos últimos anos houve um crescimento do número de estudos abordando problemas relacionados com esportes [25]. Existem diversos problemas nesta área, que surgem nas mais diferentes modalidades esportivas, tais como basquete [4, 55], beisebol [15, 16, 49], críquete [1, 54], futebol [2, 36], hóquei [18, 19], tênis [17, 27], etc. Vários deles são destinados ao planejamento do calendário de disputas entre os times ou atletas, visando atender às diversas restrições particulares de cada competição e otimizar diferentes critérios. Além destes, também é muito comum encontrar problemas que lidam com a organização dos árbitros que irão apitar as partidas de uma competição. Neste tipo de problema, normalmente deseja-se atender a restrições que objetivam promover uma influência justa dos árbitros nos resultados da competição. Estudos mais aprofundados sobre os problemas relacionados com a área de esportes podem ser encontrados nas revisões bibliográficas apresentadas em [20, 25, 37].

Nesta tese, nos concentramos em estudar o problema dos árbitros viajantes (TUP, sigla de sua denominação em inglês, *traveling umpire problem*). O TUP surgiu de uma pesquisa que estuda o problema de atribuição de árbitros às partidas da Liga Profissional de Beisebol dos Estados Unidos. Este problema é uma versão simplificada do problema real que ocorre nesta competição. Ele visa minimizar as distâncias percorridas pelos árbitros durante todo o evento, respeitando restrições que impõem que um juiz apite ao menos uma vez um jogo de cada time em sua casa, e impede que ele apite mais de uma partida em um mesmo local, ou de um mesmo time, durante determinados intervalos de rodadas consecutivas.

Antes do início das pesquisas realizadas neste doutorado, havia na literatura apenas três trabalhos abordando o TUP: [47] (extensão de [46]), [48] e [49]. Com base neles, constatamos que: *(i)* era difícil obter boas soluções para o TUP, visto que todos os métodos desenvolvidos até então, em alguns casos, retornavam soluções muito piores que as melhores conhecidas ou não encontravam soluções para instâncias comprovadamente factíveis, *(ii)* a formulação matemática de programação linear inteira proposta para o problema produzia resultados razoavelmente satisfatórios apenas para instâncias com no máximo 16 times e, além disso, não havia limitantes inferiores publicados para as instâncias com mais times, e *(iii)* apesar das fortes suspeitas deste problema ser $\mathcal{NP}$-completo, não se conhecia nenhum resultado teórico sólido corroborando esse sentido.

Neste doutoramento desenvolvemos três diferentes linhas de pesquisa que tiveram como

objetivo promover avanços a fim de resolver os problemas discriminados no parágrafo anterior. Estes esforços culminaram com a publicação de três artigos em periódicos internacionais: [33], [34] e [35]. Primeiramente, propusemos e avaliamos em [33] uma formulação matemática de programação linear inteira baseada em fluxo em rede mais forte e compacta para o TUP. Com ela foi possível obter resultados melhores tanto para as instâncias com até 16 times quanto para as demais instâncias com até 32 times. Como consequência disso, publicamos os primeiros limitantes inferiores para instâncias com mais de 16 times. Também através desta formulação, desenvolvemos um método heurístico *relax-and-fix* que encontrou melhores soluções para 24 das 25 instâncias avaliadas na literatura e foi capaz de obter soluções para todas as instâncias em que se conhecia ao menos uma solução factível. Em seguida, realizamos um estudo teórico sobre a complexidade do TUP e demonstramos em [34] que este problema é, de fato, $\mathcal{NP}$-completo para determinados tipos de instâncias. Por fim, apresentamos em [35] uma formulação matemática de programação linear inteira baseada em sub-rotas para o TUP. Ao resolver esta formulação foram obtidos resultados competitivos com aqueles mais recentes da literatura para as instâncias com até 18 times e melhores limitantes inferiores para todas as instâncias grandes com 20 ou mais times.

Ao longo do texto desta tese, assumimos que o leitor possui conhecimento básico prévio sobre programação linear, programação linear inteira, algoritmo *branch-and-bound*, algoritmo *branch-and-cut*, método de planos de corte, método de geração de colunas, fluxo em rede e heurística *relax-and-fix*. Os capítulos 1–3, 7–9, 11 e 12 do livro *Integer Programming* de Wolsey [53] introduzem de forma extremamente didática todos estes assuntos. Além deste livro, ótimos conteúdos sobre estes temas também podem ser encontrados em [3], [6], [8], [31] e [40].

O restante do texto está estruturado da seguinte forma. No próximo capítulo são apresentadas uma descrição formal do TUP e uma revisão dos trabalhos publicados na literatura. No capítulo 3 é demonstrado que o TUP é um problema $\mathcal{NP}$-completo (conteúdo publicado em [34]). O capítulo 4 introduz uma formulação matemática de programação linear inteira baseada em fluxo em rede e uma heurística *relax-and-fix* (conteúdo publicado em [33]). O capítulo 5 apresenta uma formulação matemática de programação linear inteira baseada em sub-rotas (conteúdo publicado em [35]). Por fim, no capítulo 6 discutimos as conclusões dos trabalhos realizados neste doutorado.

Capítulo 2

Problema dos árbitros viajantes

O problema dos árbitros viajantes (TUP, do inglês *traveling umpire problem*) é uma versão abstrata do problema real que visa atribuir equipes de árbitros às partidas da Liga Profissional de Beisebol (MLB, do inglês *Major League Baseball*) dos Estados Unidos. Neste problema real são impostas várias restrições para atender às exigências específicas da liga, tais como descansos entre partidas e viagens longas, férias, intervalos ao apitar jogos de um mesmo time, arbitragem de jogos de cada time em casa e fora de casa ao menos uma vez, etc. A descrição completa deste problema é apresentada em [49]. O TUP foi proposto com o intuito de incorporar apenas as principais características do problema real que tornam difícil a sua resolução. Desta forma, ao estudar o TUP focamos nossos esforços para tratar os aspectos mais importantes do problema real deixando de lado os detalhes menos relevantes.

O TUP recebe como entrada um torneio *round robin* duplo com $2n$ times dividido em $4n - 2$ rodadas. Neste tipo de torneio, cada time enfrenta todos os demais exatamente duas vezes (uma vez em sua sede e outra na sede do adversário) e disputa exatamente uma partida por rodada. Também são fornecidos como entrada as distâncias entre as sedes dos times e dois inteiros $0 \leq d_1 < n$ e $0 \leq d_2 < \lfloor \frac{n}{2} \rfloor$. Uma solução para este problema é uma atribuição de n árbitros às partidas do torneio que respeita as seguintes restrições:

(2.1) Cada partida deve ser apitada por exatamente um árbitro;

(2.2) Cada árbitro é atribuído a exatamente uma partida em cada rodada do torneio;

(2.3) Cada árbitro apita pelo menos uma partida de cada time em sua casa;

(2.4) Cada árbitro apita no máximo um jogo em uma mesma sede durante $q_1 = n - d_1$ rodadas consecutivas;

(2.5) Cada árbitro apita no máximo um jogo de um mesmo time durante $q_2 = \lfloor \frac{n}{2} \rfloor - d_2$ rodadas consecutivas.

O objetivo do TUP é encontrar uma solução que respeita estas restrições e minimiza a distância total viajada pelos árbitros durante todo o torneio. Para o cálculo das distâncias percorridas pelos árbitros, considera-se que eles viajam sempre do local de uma partida

diretamente para o local da próxima partida a ser apitada e desconsideram-se as distâncias para chegar no local da primeira partida e para deixar o local da última partida do torneio.

Ilustramos na figura 2.1 uma solução para uma instância do TUP com um torneio de 8 times e $d_1 = d_2 = 0$. Neste exemplo, cada jogo do torneio é representado por um par ordenado com os índices dos times que disputam a partida, sendo que o primeiro time no par ordenado é aquele que sedia o jogo. Cada linha na tabela apresentada contém os jogos atribuídos a um árbitro. Desta forma, as restrições (2.1) e (2.2) são satisfeitas. Todos os árbitros visitam cada sede ao menos uma vez o que satisfaz a restrição (2.3) (por exemplo, as oito sedes são visitadas pelo árbitro 1 nas rodadas 1, 2, 3, 4, 5, 6, 7 e 13). Como temos $d_1 = d_2 = 0$, nesta solução nenhum árbitro visita uma mesma sede mais de uma vez durante quaisquer $q_1 = 4 - 0 = 4$ rodadas consecutivas e apita um jogo de um mesmo time durante quaisquer $q_2 = 2 - 0 = 2$ rodadas consecutivas, respeitando as restrições (2.4) e (2.5) (por exemplo, o árbitro 1 visita as sedes 1, 3, 7 e 2 nas quatro primeiras rodadas e apita os jogos dos times 1, 5, 3 e 6 nas duas primeiras rodadas, sem repetir sedes ou times nestas rodadas consecutivas).

	Rodadas													
Árbitro	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	(1,5)	(3,6)	(7,1)	(2,3)	(4,1)	(8,2)	(6,7)	(2,8)	(4,5)	(3,8)	(6,2)	(1,8)	(5,3)	(7,6)
2	(4,8)	(1,2)	(6,4)	(8,1)	(5,3)	(1,6)	(4,2)	(6,1)	(7,8)	(5,2)	(3,7)	(6,5)	(2,7)	(1,3)
3	(6,2)	(7,8)	(2,5)	(4,7)	(6,8)	(7,5)	(1,3)	(5,7)	(6,3)	(1,7)	(8,4)	(3,2)	(4,1)	(5,8)
4	(7,3)	(5,4)	(8,3)	(6,5)	(2,7)	(4,3)	(8,5)	(3,4)	(2,1)	(4,6)	(1,5)	(7,4)	(8,6)	(2,4)

Figura 2.1: Solução de um instância do TUP com um torneio de 8 times e $d_1 = d_2 = 0$.

O TUP foi primeiramente introduzido por Trick e Yildiz em [46] e posteriormente descrito com mais detalhes em [49] juntamente com o problema real que o originou. Desde então, diversos trabalhos foram publicados na literatura abordando este problema e deixaram evidente a sua grande dificuldade de resolução. Até mesmo encontrar soluções factíveis para o TUP revela-se uma tarefa difícil e demanda métodos mais robustos. Portanto, isso nos motivou a estudá-lo e tê-lo como tema desta tese de doutorado.

A seguir apresentamos uma revisão dos trabalhos sobre o TUP encontrados na literatura. Optamos por incluir também no texto desta revisão os artigos que publicamos, que compõem o conteúdo dos próximos capítulos desta tese, a fim de fornecer uma visão completa do estado da arte no momento da escrita deste texto.

Um conjunto de instâncias *benchmark* do TUP foi introduzido em [46, 49]. Este *benchmark* possui instâncias com torneios de 4 até 32 times, que são de domínio público e estavam disponíveis para serem baixadas em [44], mas recentemente passaram a ser mantidas pelo novo *benchmark* automatizado [41]. Estas instâncias do TUP foram baseadas nas instâncias *benchmark* do problema do torneio viajante [14] (TTP, do inglês *traveling tournament problem*), e suas soluções, disponibilizadas em [45]. O TTP consiste em um problema que surge antes do TUP, que é o planejamento dos jogos de um torneio. Neste problema é dado como entrada um conjunto de times e as distâncias entre suas sedes, e o objetivo é estabelecer um torneio *round robin* duplo, onde nenhum par de times se enfrenta em rodadas consecutivas e nenhum time disputa mais do que três jogos seguidos em casa ou fora de casa, minimizando a distância total viajada pelos times. As instâncias

do TUP foram criadas a partir do TTP da seguinte forma: aquelas com até 14 times têm as mesmas matrizes de distâncias entre as sedes das instâncias NL do TTP, ao passo que aquelas com 16 ou mais times possuem as mesmas matrizes de distâncias entre as sedes das instâncias da Liga Nacional de Futebol Americano (NFL, do inglês *National Football League*) do TTP. Os torneios das instâncias do TUP são soluções obtidas para estas instâncias do TTP. O nome da cada instância do TUP possui o número de times em seu torneio, seguido facultativamente por uma letra. A presença da letra indica que a instância é uma variação da instância original (sem letra), mantendo o mesmo torneio mas permutando os nomes dos times para obter uma matriz de distâncias entre as sedes diferente. Todos os trabalhos na literatura do TUP até o momento utilizam este conjunto de instâncias em seus experimentos computacionais para comparar os resultados obtidos pelos métodos propostos.

Os autores em [47] (extensão do estudo apresentado em [46]) propuseram um modelo de programação linear inteira e um modelo de programação por restrições para o TUP. Métodos exatos foram aplicados para resolver estes modelos. As execuções destes métodos foram limitadas a intervalos de tempo considerados aceitáveis pelos autores. Apenas instâncias com no máximo 10 times foram resolvidas na otimalidade e uma instância com 12 times foi provada ser infactível. Em algumas das instâncias maiores, os métodos exatos tiveram dificuldades em encontrar soluções factíveis ou produziram soluções ruins dentro dos limites de tempo estabelecidos. Por este motivo, os autores também desenvolveram uma heurística gulosa baseada em emparelhamentos de custo mínimo a fim de encontrar boas soluções. Durante o processo de construção de uma solução, é bastante comum que a heurística fique presa em uma situação onde todas as opções de escolha levam a soluções infactíveis. Neste momento, uma etapa de busca local guiada por cortes de Bender é aplicada para modificar as escolhas feitas anteriormente de tal forma que a heurística passe a ter escolhas factíveis, permitindo que a execução do método possa continuar e possivelmente consiga terminar com uma solução. Respeitando os limites de tempo impostos, esta heurística encontrou soluções melhores que aquelas obtidas pelos métodos exatos para várias instâncias com 14 e 16 times e obteve soluções factíveis para uma instância com 30 times onde os métodos exatos não foram capazes.

Um modelo de programação linear inteira é proposto para o problema real da Liga Profissional de Beisebol em [49]. Entretanto, este modelo não pôde ser otimizado por métodos exatos devido à grande quantidade de variáveis e restrições necessárias para descrever todas as regras do problema. Isso motivou a proposição do TUP como uma versão abstrata deste problema real para capturar apenas os seus aspectos principais. Os autores implementaram uma heurística *simulated annealing* para obter boas soluções tanto para o TUP quanto para o problema real. Embora esta heurística não tenha conseguido obter soluções melhores que aquelas encontradas pela heurística gulosa apresentada em [47] para o TUP, as soluções obtidas por ela para o problema real são melhores que as oficiais utilizadas na liga em anos anteriores.

Os resultados experimentais apresentados em [47, 49] deixaram evidente que encontrar boas soluções para as instâncias do TUP com mais de 12 times não é uma tarefa simples. Isto motivou o surgimento de pesquisas propondo métodos heurísticos mais robustos. Os autores em [48] desenvolveram um algoritmo genético composto por um sofisticado

operador de cruzamento. Este operador emprega uma rotina de emparelhamentos de custo mínimo para recombinar duas soluções efetuando uma otimização local. Esta heurística encontrou melhores soluções para diversas instâncias com 14, 16 e 30 times.

O trabalho apresentado por nós em [33] introduziu uma formulação matemática de programação linear inteira baseada em fluxo em rede mais forte e compacta para o TUP, inspirada naquela que foi proposta em [47]. A nova formulação reduz a quantidade de variáveis e restrições da formulação anterior e inclui restrições adicionais que fortalecem os limitantes inferiores. Ao resolver esta formulação, foram obtidos melhores limitantes inferiores para todas as instâncias *benchmark* do TUP. Até mesmo para as instâncias com 16 ou mais times, as quais não se conheciam limitantes inferiores até então, pelo fato desta formulação ser mais compacta, tornou-se viável resolver a sua relaxação e gerar pela primeira vez limitantes inferiores para todas essas instâncias. Através desta formulação também foi desenvolvida uma heurística *relax-and-fix* que encontrou melhores soluções para 24 das 25 instâncias avaliadas.

Em [51] foram desenvolvidas uma busca em profundidade iterativa (IDS, do inglês *iterative deepening search*) e uma busca local iterativa (ILS, do inglês *iterated local search*). Estes métodos foram considerados complementares no sentido em que a IDS melhorou diversas soluções nas instâncias de médio porte (com 14 e 16 times) ao passo que a ILS encontrou melhores soluções para as instâncias grandes com 26 ou mais times. Os autores também propuseram um esquema de decomposição para obter limitantes inferiores mais fortes. Este esquema consiste em subdividir o torneio em partes menores e para cada parte é resolvida uma versão relaxada do modelo apresentado por nós em [33]. Através deste método foram obtidos limitantes inferiores melhores para todas as instâncias consideradas nos experimentos.

Um modelo de programação linear inteira baseado em partição de conjuntos foi proposto para o TUP em [42]. Neste modelo, cada variável representa uma viagem completa de um árbitro ao longo das $4n - 2$ rodadas do torneio. Um algoritmo *branch-and-price* foi desenvolvido para otimizar este modelo uma vez que ele possui uma quantidade exponencial de variáveis. A rotina de *pricing* deste método consiste na resolução de problemas de caminhos mínimos com restrições adicionais que tornam os problemas complexos. Por isso, um algoritmo *branch-and-bound* foi utilizado para resolver estes problemas da rotina de *pricing*. Para as instâncias com 14 e 16 times, o *branch-and-price* obteve vários limitantes inferiores melhores ao adotar uma estratégia que escolhe a variável com valor mais fracionário para ramificar durante a enumeração, e encontrou algumas soluções melhores ao utilizar uma estratégia que escolhe a variável com valor mais próximo de um número inteiro. Os autores também conseguiram provar a infactibilidade de quatro instâncias com 16 times através de uma adaptação para o TUP do *branch-and-bound* originalmente proposto para resolver os problemas na rotina de *pricing*.

Dois modelos de programação linear inteira foram formulados em [56]. Um modelo é baseado em fluxo em rede e o outro é baseado em partição de conjuntos. Este último é similar àquele visto em [42], todavia neste os autores incluíram cortes adicionais para fortalecer ainda mais os limitantes inferiores. O modelo baseado em fluxo em rede foi resolvido com um algoritmo *branch-and-bound* que otimiza uma relaxação Lagrangiana do modelo em cada nó da árvore de enumeração. O *branch-and-bound* conseguiu melho-

rar vários dos limitantes inferiores nas instâncias com mais de 18 times. Para resolver o modelo baseado em partições de conjuntos, os autores desenvolveram um algoritmo *branch-and-price-and-cut*. A rotina de *pricing* deste método consiste em resolver problemas de caminhos mínimos semelhantes aos solucionados no *pricing* visto em [42], exceto pelos cortes adicionais neste modelo que também precisam ser tratados. Um algoritmo *label-setting* foi elaborado para resolver os problemas no *pricing* e um algoritmo enumerativo para efetuar a separação dos cortes. O *branch-and-price-and-cut* melhorou vários dos limitantes inferiores das instâncias com até 18 times e resolveu duas instâncias com 14 times na otimalidade gastando por volta de 11 horas em uma e 34 horas na outra. Este foi o primeiro método a conseguir solucionar instâncias com mais de 10 times na otimalidade.

Os modelos baseados em partição de conjuntos propostos em [42] e [56] fornecem limitantes inferiores muito fortes mas, em contrapartida, consomem grandes esforços computacionais para serem resolvidos. Isto ocorre porque os problemas que precisam ser otimizados para efetuar o *pricing* destes modelos são complexos. Por este motivo, a aplicação deles fica limitada na prática às instâncias com no máximo 18 times. Tendo isto em mente, em [35] propusemos um modelo de programação linear inteira baseado em sub-rotas em que é possível determinar o tamanho das viagens (quantidade de sedes percorridas ao longo de rodadas consecutivas) que as variáveis representam. Ou seja, diferentemente dos modelos baseados em partição de conjuntos vistos em [42] e [56] onde este tamanho é fixado em $4n - 2$, neste é possível ajustar o tamanho em qualquer valor entre 2 e $4n - 2$. Desta forma, ao utilizar tamanhos moderados é possível enumerar *a priori* todas as variáveis do modelo e evitar o custo computacional das inúmeras execuções de uma rotina de *pricing*. Um algoritmo *branch-and-cut* foi desenvolvido para otimizar o modelo e obteve limitantes inferiores melhores para todas as instâncias com mais de 18 times. Além disso, este método resolveu na otimalidade as mesmas duas instâncias com 14 times solucionadas em [56], contudo gastando em torno de 9 minutos em uma e 30 minutos na outra, e também mais outras duas instâncias com 14 times do *benchmark*.

Em [43] foi proposto um *branch-and-bound* combinado com uma rotina de geração de limitantes inferiores que aplica a técnica de decomposição apresentada em [51]. Esta rotina calcula os limitantes inferiores de forma independente e em paralelo ao *branch-and-bound*. Ela emprega uma estratégia *bottom-up*, onde o problema é inicialmente decomposto em subproblemas menores que são resolvidos e os resultados são reutilizados para solucionar subproblemas maiores. Desta forma, o tamanho dos subproblemas vai sendo gradativamente aumentado e, consequentemente, os limitantes inferiores providos são iterativamente melhorados. É importante ressaltar também que ao retirar o peso do cálculo dos limitantes inferiores do *branch-and-bound*, este se torna capaz de efetuar a enumeração dos nós extremamente rápida. Os experimentos computacionais apresentados revelaram que este *branch-and-bound* conseguiu resultados impressionantes para as instâncias de médio porte. Foram resolvidas na otimalidade todas as instâncias com 14 times gastando poucos minutos e 11 instâncias com 16 times gastando poucas horas. Este foi o primeiro método que conseguiu resolver na otimalidade instâncias com 16 times. Além disso, alguns limitantes inferiores e superiores também foram melhorados para as instâncias com 16 times. Entretanto, embora os excelentes resultados obtidos para as

instâncias com 14 e 16 times, este método não atingiu bons resultados para as instâncias com 18 ou mais times, sendo bem inferiores àqueles alcançados em [35].

A primeira prova de complexidade para o TUP foi apresentada por nós em [34]. Neste estudo, efetuamos uma redução de uma variação do problema do ciclo hamiltoniano à versão de decisão do TUP, demonstrando que o último é $\mathcal{NP}$-completo para instâncias onde $d_1 \leq \frac{n}{2}$ e $d_2 = \left\lfloor \frac{n}{2} \right\rfloor - 1$.

Uma aproximação combinada foi desenvolvida para o TUP em [5]. Nesta aproximação, primeiramente é considerado o TTP (citado anteriormente na descrição das instâncias do TUP na página 14). Os autores aplicam um algoritmo aproximativo proposto por eles para o TTP a fim de obter um torneio, e então mostram como efetuar uma atribuição dos árbitros aos jogos deste torneio garantindo um fator de aproximação constante.

Por fim, recentemente foi proposto em [50] um algoritmo de programação de conjunto de resposta (ASP, do inglês *answer set programming*) para o TUP. O ASP mostrou-se competitivo quando comparado com os métodos de programação por restrições e programação linear inteira e com a heurística gulosa guiada por cortes de Bender avaliados em [47]. Contudo, este algoritmo não superou nenhum dos melhores resultados da literatura.

Capítulo 3

Complexidade do problema dos árbitros viajantes

Desde a introdução do TUP, diversas pesquisas surgiram propondo métodos aproximativos, exatos e heurísticos para este problema [5, 33, 35, 43, 42, 47, 48, 49, 50, 51, 56]. Os resultados experimentais apresentados nestes trabalhos deixaram claro que o TUP é um problema extremamente difícil de ser resolvido. Atualmente, a maior instância solucionada na otimalidade possui 16 times ao passo que a competição esportiva que deu origem a este problema possui 30 times. Desta forma, apesar dos avanços feitos, nota-se que ainda não estamos perto de resolver de forma exata instâncias com o tamanho do problema real.

Apesar das evidências empíricas, muito tempo passou até conhecermos uma prova teórica de complexidade do TUP. A primeira prova, e única até o presente momento, foi publicada em [34] e surgiu de um estudo aprofundado neste doutorado. Efetuamos a prova através de uma redução em tempo polinomial de uma variação do problema de decidir se um dado grafo é hamiltoniano à versão de decisão do TUP. Vamos apresentar esta prova neste capítulo. Para tanto, primeiramente será introduzida na próxima seção a notação adotada ao longo do texto e também provaremos alguns resultados auxiliares utilizados posteriormente. Em seguida efetuamos a demonstração da prova de complexidade do TUP e, por fim, discutimos as conclusões deste estudo teórico.

3.1 Notação e resultados preliminares

Usaremos as letras i e j para representar os times e suas respectivas sedes (nos referindo à sede do time i (ou j) diretamente como sede i (ou j)), u para representar os árbitros e s para representar as rodadas do torneio.

A fim de simplificar as expressões matemáticas envolvidas na prova de complexidade, excepcionalmente neste capítulo adotamos índices (para os times, rodadas, etc) que iniciam em zero. Nos demais capítulos os índices sempre começarão em um.

Seja T um torneio com $2n$ times e m rodadas. Cada jogo deste torneio será representado por um par ordenado contendo os índices dos times da partida, sendo que o primeiro índice no par ordenado deve ser do time cuja sede é o local onde é realizado o jogo. En-

tão, T é definido com sendo uma sequência de conjuntos de pares ordenados escrevendo $T = S_0, S_1, \ldots, S_{m-1}$, onde S_s é o conjunto dos jogos (pares ordenados) disputados na $(s+1)$-ésima[1] rodada. Seja $C = \{(i_0, j_0), (i_1, j_1), \ldots, (i_{v-1}, j_{v-1})\}$ um conjunto com v pares ordenados. Vamos denotar por $\overline{C}$ o conjunto obtido de C invertendo todos os pares ordenados. Ou seja, $\overline{C} = \{(j_0, i_0), (j_1, i_1), \ldots, (j_{v-1}, i_{v-1})\}$. Através desta notação, a inversão das sedes dos jogos de T pode ser denotada por $\overline{T} = \overline{S}_0, \overline{S}_1, \ldots, \overline{S}_{m-1}$. Desta forma, para cada par de times i e j, se i joga em casa contra j na rodada s de T, então j joga em casa contra i na rodada s de $\overline{T}$.

Um torneio *round robin* simples (duplo) é um torneio em que cada time enfrenta cada um dos demais times exatamente uma vez (duas vezes: uma vez em casa e outra fora). As equações (3.1)–(3.3) definem uma forma de construir um torneio *round robin* simples $U_{a,b}$ com uma quantidade par de times $a \geq 2$, $a-1$ rodadas, e times com índices variando de b até $b+a-1$.

$$U_{a,b} = U_{a,b}[0, a-2], \tag{3.1}$$

$$U_{a,b}[s_1, s_2] = Q_{a,b}[s_1], Q_{a,b}[s_1+1], ..., Q_{a,b}[s_2], \quad \forall\ 0 \leq s_1 \leq s_2 \leq a-2, \tag{3.2}$$

$$\begin{aligned} Q_{a,b}[s] = \{&(b + (s \bmod (a-1)), b+a-1), \\ &(b + ((s+1) \bmod (a-1)), b + ((s+a-2) \bmod (a-1))), \\ &(b + ((s+2) \bmod (a-1)), b + ((s+a-3) \bmod (a-1))), \\ &\vdots \\ &(b + ((s+a/2-1) \bmod (a-1)), b + ((s+a-a/2) \bmod (a-1))\}, \\ &\forall\ 0 \leq s \leq a-2. \end{aligned} \tag{3.3}$$

A figura 3.1 mostra os torneios $U_{a,b}$ com $a = 8$ e $b = 0, 8, 16, 24$. Apresentamos estes quatro torneios em específico porque eles serão utilizados em um exemplo posterior (visto nas figuras 3.5 e 3.6) para compor um torneio maior. Desta forma, o leitor que queira conferir o exemplo mencionado poderá confrontá-lo com os torneios nesta figura.

A definição algébrica de $U_{a,b}$ resulta em um método equivalente ao método do círculo ou polígono [13, 28], conhecido também como método de Kirkman que, até onde sabemos, foi primeiramente introduzido em [26]. Neste método, a construção do torneio $U_{a,b}$ pode ser feita de uma forma geométrica aplicando os seguintes passos. Disponha os índices $b+0$, $b+1$, ..., $b+a-2$ igualmente espaçados ao longo do perímetro de um círculo (ou nos vértices de um $(a-1)$-ágono regular) em ordem crescente no sentido horário, e coloque $b+a-1$ no centro do círculo. Para obter os jogos da rodada $0 \leq s \leq a-2$ do torneio, trace um segmento de reta ligando $b+s$ a $b+a-1$ e conecte os demais pares de índices através de segmentos de reta perpendiculares ao segmento de reta entre $b+s$ e $b+a-1$ (veja a rodada $s=0$ na figura 3.2). Os jogos da rodada s correspondem aos pares de times cujos índices estão ligados pelos segmentos de reta. A figura 3.2 ilustra a construção geométrica das rodadas do torneio $U_{8,0}$. Note nessa figura que, após a primeira rodada, as demais são obtidas rotacionando os segmentos de reta no interior do círculo no sentido horário.

[1] A rigor, é necessário somar um aos índices que iniciam em zero empregados na notação ordinal. Isto deve ser feito para que o primeiro elemento (1-ésimo) corresponda àquele com índice zero.

$U_{8,0}$

Rodadas						
0	1	2	3	4	5	6
(0, 7)	(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)
(1, 6)	(2, 0)	(3, 1)	(4, 2)	(5, 3)	(6, 4)	(0, 5)
(2, 5)	(3, 6)	(4, 0)	(5, 1)	(6, 2)	(0, 3)	(1, 4)
(3, 4)	(4, 5)	(5, 6)	(6, 0)	(0, 1)	(1, 2)	(2, 3)

$U_{8,8}$

Rodadas						
0	1	2	3	4	5	6
(8, 15)	(9, 15)	(10, 15)	(11, 15)	(12, 15)	(13, 15)	(14, 15)
(9, 14)	(10, 8)	(11, 9)	(12, 10)	(13, 11)	(14, 12)	(8, 13)
(10, 13)	(11, 14)	(12, 8)	(13, 9)	(14, 10)	(8, 11)	(9, 12)
(11, 12)	(12, 13)	(13, 14)	(14, 8)	(8, 9)	(9, 10)	(10, 11)

$U_{8,16}$

Rodadas						
0	1	2	3	4	5	6
(16, 23)	(17, 23)	(18, 23)	(19, 23)	(20, 23)	(21, 23)	(22, 23)
(17, 22)	(18, 16)	(19, 17)	(20, 18)	(21, 19)	(22, 20)	(16, 21)
(18, 21)	(19, 22)	(20, 16)	(21, 17)	(22, 18)	(16, 19)	(17, 20)
(19, 20)	(20, 21)	(21, 22)	(22, 16)	(16, 17)	(17, 18)	(18, 19)

$U_{8,24}$

Rodadas						
0	1	2	3	4	5	6
(24, 31)	(25, 31)	(26, 31)	(27, 31)	(28, 31)	(29, 31)	(30, 31)
(25, 30)	(26, 24)	(27, 25)	(28, 26)	(29, 27)	(30, 28)	(24, 29)
(26, 29)	(27, 30)	(28, 24)	(29, 25)	(30, 26)	(24, 27)	(25, 28)
(27, 28)	(28, 29)	(29, 30)	(30, 24)	(24, 25)	(25, 26)	(26, 27)

Figura 3.1: Torneios $U_{8,0}$, $U_{8,8}$, $U_{8,16}$, e $U_{8,24}$.

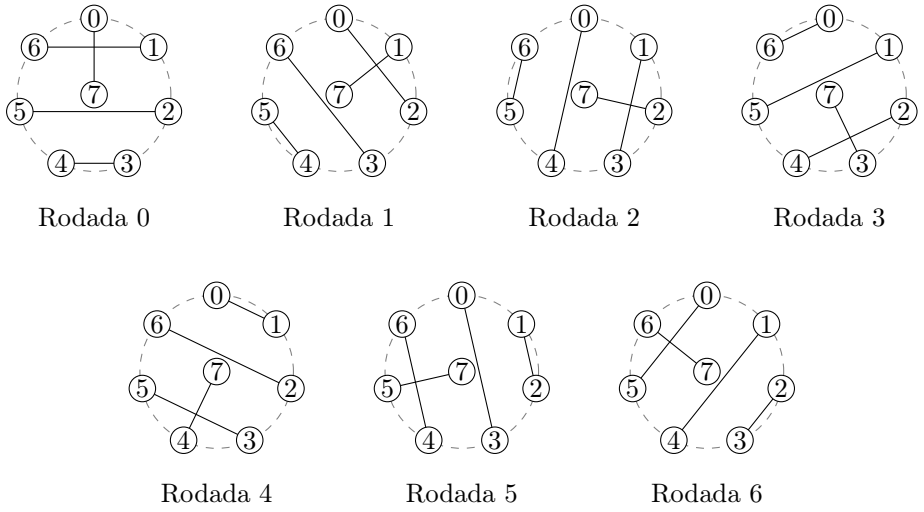


Figura 3.2: Construção geométrica do torneio $U_{8,0}$ pelo método de Kirkman.

A fim de prover ao leitor uma prova de complexidade autocontida, o lema a seguir demonstra que $U_{a,b}$ sempre resulta corretamente em um torneio *round robin* simples. Uma

prova equivalente também pode ser vista em [57].

Lema 3.1. *Dado um número par de times $a \geq 2$ representados por índices consecutivos iniciando em $b \geq 0$, $U_{a,b}$ é um torneio round robin simples para estes times.*

Demonstração. Com base na definição $U_{a,b}$, sabemos que os primeiros elementos dos pares ordenados da rodada $0 \leq s \leq a-2$ são os times $b + ((s+0) \bmod (a-1))$, ..., $b + ((s + (a/2)-1) \bmod (a-1))$, e os segundos elementos são os times $b+((s+(a/2)) \bmod (a-1))$, ..., $b+((s+a-2) \bmod (a-1))$ e $b+a-1$. Para qualquer valor de s, estas duas sequências juntas compõem os times 0, 1, ..., $b+a-1$ sem repetições. Portanto, cada time disputa exatamente um jogo por rodada. Além disso, note que o time $b+a-1$ joga contra os times $b + ((0 \bmod (a-1))$, $b + (1 \bmod (a-1))$, ..., $b + ((a-2) \bmod (a-1)) = b$, $b+1$, ..., $b+a-2$. Logo, o time $b+a-1$ joga exatamente uma vez contra todos os demais. Agora suponha por contradição que os times $i \neq b+a-1$ e $j \neq b+a-1$ se enfrentem duas vezes, uma vez na rodada s_1 e outra vez na rodada s_2, com $0 \leq s_1 \neq s_2 \leq a-2$. Sem perda de generalidade, considere que o jogo disputado na rodada s_1 é (i,j), ou seja, na casa do time i. A partir da definição de $U_{a,b}$, temos

$$i = b + ((s_1 + r_1) \bmod (a-1)), \tag{3.4}$$
$$j = b + ((s_1 + a - r_1 - 1) \bmod (a-1)), \tag{3.5}$$

onde r_1 é um inteiro no intervalo $[1, (a/2)-1]$. Primeiramente, vamos supor que o jogo disputado na rodada s_2 é (i,j) e então temos

$$i = b + ((s_2 + r_2) \bmod (a-1)), \tag{3.6}$$
$$j = b + ((s_2 + a - r_2 - 1) \bmod (a-1)), \tag{3.7}$$

onde r_2 é também um inteiro no intervalo $[1, (a/2)-1]$. Subtraindo j de i, aplicando módulo $a-1$ e substituindo i e j por (3.4) e (3.5), e por (3.6) e (3.7) obtemos (3.8) e (3.9), respectivamente.

$$(((s_1 + r_1) \bmod (a-1)) - ((s_1 + a - r_1 - 1) \bmod (a-1))) \bmod (a-1), \tag{3.8}$$
$$(((s_2 + r_2) \bmod (a-1)) - ((s_2 + a - r_2 - 1) \bmod (a-1))) \bmod (a-1). \tag{3.9}$$

Ao aplicar a definição da operação de subtração $((A \bmod C) - (B \bmod C)) \bmod C = (A-B) \bmod C$ da aritmética modular (veja [7, p. 940]) a (3.8) e (3.9), e igualar os resultados, uma vez que eles devem ser iguais, vamos obter

$$((2r_1 - a + 1) \bmod (a-1)) = ((2r_2 - a + 1) \bmod (a-1)). \tag{3.10}$$

Esta igualdade implica que $r_1 = r_2$ pois $1 \leq r_1, r_2 \leq (a/2)-1$. Como (3.4) é igual a (3.6), (3.5) é igual a (3.7) e temos $r_1 = r_2$, concluímos que $s_1 = s_2$ desde que $0 \leq s_1, s_2 \leq a-2$, contradizendo nossa suposição $s_1 \neq s_2$.

Agora considere que o jogo disputado na rodada s_2 é (j, i). Neste caso, temos

$$i = b + ((s_2 + a - r_2 - 1) \bmod (a-1)), \tag{3.11}$$
$$j = b + ((s_2 + r_2) \bmod (a-1)). \tag{3.12}$$

Subtraindo j de i, aplicando módulo $a-1$ e substituindo i e j por (3.11) e (3.12) resulta em

$$(((s_2 + a - r_2 - 1) \bmod (a-1)) - ((s_2 + r_2) \bmod (a-1))) \bmod (a-1). \tag{3.13}$$

De forma análoga, vamos obter (3.14) ao aplicar a definição da operação de subtração mencionada antes a (3.13) e igualar o resultado a (3.8).

$$((2r_1 - a + 1) \bmod (a-1)) = ((a - 2r_2 - 1) \bmod (a-1)). \tag{3.14}$$

Note que é impossível escolher valores para $1 \leq r_1, r_2 \leq (a/2) - 1$ que satisfaçam esta igualdade. Portanto, novamente chegamos em uma contradição já que não pode existir o jogo (j,i) em $U_{a,b}$ ao supor que existe o jogo (i,j) neste torneio. Desta forma, cada time não joga duas vezes contra outro time nas rodadas 0,1,...,$a-2$. Isso implica que todos os times jogam exatamente uma vez contra todos os demais. □

Além de U, vamos definir um outro tipo de torneio denotado por P. Posteriormente, vamos combinar torneios U e P para criar torneios *round robin* duplos que serão a base da nossa prova principal.

As equações (3.15)–(3.17) definem um torneio $P_{a,b}$ com $a > 0$ rodadas e $2a$ times com índices consecutivos iniciando em $b \geq 0$. Neste torneio, os primeiros a times $(b, \ldots, b + a - 1)$ jogam contra cada um dos demais a times $(b + a, \ldots, b + 2a - 1)$ exatamente uma vez. Além disso, os times $b, \ldots, b + a - 1$ não se enfrentam entre si, assim como os times $b + a, \ldots, b + 2a - 1$ também não se enfrentam entre si.

$$P_{a,b} = P_{a,b}[0, a-1], \tag{3.15}$$
$$P_{a,b}[s_1, s_2] = X_{a,b}[s_1], X_{a,b}[s_1 + 1], ..., X_{a,b}[s_2], \quad \forall\ 0 \leq s_1 \leq s_2 \leq a-1, \tag{3.16}$$
$$\begin{aligned} X_{a,b}[s] = \{&(b + 0, b + a + (s \bmod a)), \\ &(b + 1, b + a + ((s+1) \bmod a)), \\ &\vdots \\ &(b + a - 1, b + a + ((s + a - 1) \bmod a))\}, \\ &\forall\ 0 \leq s \leq a - 1. \end{aligned} \tag{3.17}$$

Note que em um torneio $P_{a,b}$ cada time disputa exatamente um jogo em cada rodada. As figuras 3.3 e 3.4 apresentam três exemplos de torneios $P_{a,b}$, dois com 8 times e um com 16 times (estes torneios também servirão de auxílio ao leitor que queira conferir o exemplo visto posteriormente nas figuras 3.5 e 3.6). A notação $P_{a,b}[s_1, s_2]$ definida em (3.16) é usada para representar a parte do torneio $P_{a,b}$ que vai da rodada s_1 até a rodada s_2.

A seguir vamos apresentar três operações para sequências genéricas de elementos,

$P_{8,0}$

Rodadas							
0	1	2	3	4	5	6	7
(0, 8)	(0, 9)	(0, 10)	(0, 11)	(0, 12)	(0, 13)	(0, 14)	(0, 15)
(1, 9)	(1, 10)	(1, 11)	(1, 12)	(1, 13)	(1, 14)	(1, 15)	(1, 8)
(2, 10)	(2, 11)	(2, 12)	(2, 13)	(2, 14)	(2, 15)	(2, 8)	(2, 9)
(3, 11)	(3, 12)	(3, 13)	(3, 14)	(3, 15)	(3, 8)	(3, 9)	(3, 10)
(4, 12)	(4, 13)	(4, 14)	(4, 15)	(4, 8)	(4, 9)	(4, 10)	(4, 11)
(5, 13)	(5, 14)	(5, 15)	(5, 8)	(5, 9)	(5, 10)	(5, 11)	(5, 12)
(6, 14)	(6, 15)	(6, 8)	(6, 9)	(6, 10)	(6, 11)	(6, 12)	(6, 13)
(7, 15)	(7, 8)	(7, 9)	(7, 10)	(7, 11)	(7, 12)	(7, 13)	(7, 14)

$P_{8,16}$

Rodadas							
0	1	2	3	4	5	6	7
(16, 24)	(16, 25)	(16, 26)	(16, 27)	(16, 28)	(16, 29)	(16, 30)	(16, 31)
(17, 25)	(17, 26)	(17, 27)	(17, 28)	(17, 29)	(17, 30)	(17, 31)	(17, 24)
(18, 26)	(18, 27)	(18, 28)	(18, 29)	(18, 30)	(18, 31)	(18, 24)	(18, 25)
(19, 27)	(19, 28)	(19, 29)	(19, 30)	(19, 31)	(19, 24)	(19, 25)	(19, 26)
(20, 28)	(20, 29)	(20, 30)	(20, 31)	(20, 24)	(20, 25)	(20, 26)	(20, 27)
(21, 29)	(21, 30)	(21, 31)	(21, 24)	(21, 25)	(21, 26)	(21, 27)	(21, 28)
(22, 30)	(22, 31)	(22, 24)	(22, 25)	(22, 26)	(22, 27)	(22, 28)	(22, 29)
(23, 31)	(23, 24)	(23, 25)	(23, 26)	(23, 27)	(23, 28)	(23, 29)	(23, 30)

Figura 3.3: Torneios $P_{8,0}$ e $P_{8,16}$.

que usaremos para combinar os torneios U e P. Sejam $A = A_1, A_2, ..., A_g$ e $B = B_1, B_2, ..., B_h$ duas sequências com g e h elementos, respectivamente. A operação de *concatenação* $A \oplus B$ produz $A_1, A_2, \ldots, A_g, B_1, B_2, \ldots, B_h$. A operação de *intercalação* $A \oslash B$ produz $A_1, B_1, A_2, B_2, \ldots, A_g, B_h$ quando $g = h$, e produz $A_1, B_1, A_2, B_2, \ldots, A_h, B_h, A_{h+1}, A_{h+2}, \ldots, A_g$ quando $g > h$. O caso $g < h$ funciona de forma análoga ao caso $g > h$. Por fim, quando A e B são sequências de conjuntos, a operação *união par a par* $A \odot B$ resulta em $(A_1 \cup B_1), (A_2 \cup B_2), \ldots, (A_g \cup B_h)$ para $g = h$.

Agora estamos prontos para explicar como os torneios U e P serão utilizados para construir um torneio *round robin* duplo. Vamos combinar os torneios $U_{k,0}$, $\overline{U}_{k,0}$, $U_{k,k}$, $\overline{U}_{k,k}$, $U_{k,2k}$, $\overline{U}_{k,2k}$, $U_{k,3k}$, $\overline{U}_{k,3k}$, $P_{k,0}$, $\overline{P}_{k,0}$, $P_{k,2k}$, $\overline{P}_{k,2k}$, $P_{2k,0}$ e $\overline{P}_{2k,0}$ para obter o torneio *round robin* duplo T com $4k$ times e $8k-2$ rodadas. As figuras 3.1, 3.3 e 3.4 ilustram os torneios $U_{k,0}$, $U_{k,k}$, $U_{k,2k}$, $U_{k,3k}$, $P_{k,0}$, $P_{k,2k}$ e $P_{2k,0}$ para $k = 8$. Omitimos as ilustrações de $\overline{U}_{k,0}$, $\overline{U}_{k,k}$, $\overline{U}_{k,2k}$, $\overline{U}_{k,3k}$, $\overline{P}_{k,0}$, $\overline{P}_{k,2k}$ e $\overline{P}_{2k,0}$ pois estes são iguais aos apresentados nestas figuras diferindo apenas nas sedes dos jogos que são invertidas. O torneio T é definido por (3.18)–(3.21). As figuras 3.5 e 3.6 apresentam todos os jogos de T para $k = 8$, indicando qual termo em (3.18)–(3.21) gerou cada parte do torneio.

$$T_1 = \overline{P}_{2k,0}[0, k-1] \oslash (\overline{U}_{k,0} \odot \overline{U}_{k,k} \odot \overline{U}_{k,2k} \odot \overline{U}_{k,3k}), \tag{3.18}$$

$$T_2 = \overline{P}_{2k,0}[k, 2k-1] \oslash (U_{k,0} \odot U_{k,k} \odot U_{k,2k} \odot U_{k,3k}), \tag{3.19}$$

$$T_3 = (\overline{P}_{k,0} \odot \overline{P}_{k,2k}) \oslash (P_{k,0} \odot P_{k,2k}), \tag{3.20}$$

$$T = T_1 \oplus P_{2k,0}[0, k-1] \oplus T_2 \oplus P_{2k,0}[k, 2k-1] \oplus T_3. \tag{3.21}$$

$P_{16,0}$																
Rodadas																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(0, 16)	(0, 17)	(0, 18)	(0, 19)	(0, 20)	(0, 21)	(0, 22)	(0, 23)	(0, 24)	(0, 25)	(0, 26)	(0, 27)	(0, 28)	(0, 29)	(0, 30)	(0, 31)
(1, 17)	(1, 18)	(1, 19)	(1, 20)	(1, 21)	(1, 22)	(1, 23)	(1, 24)	(1, 25)	(1, 26)	(1, 27)	(1, 28)	(1, 29)	(1, 30)	(1, 31)	(1, 16)
(2, 18)	(2, 19)	(2, 20)	(2, 21)	(2, 22)	(2, 23)	(2, 24)	(2, 25)	(2, 26)	(2, 27)	(2, 28)	(2, 29)	(2, 30)	(2, 31)	(2, 16)	(2, 17)
(3, 19)	(3, 20)	(3, 21)	(3, 22)	(3, 23)	(3, 24)	(3, 25)	(3, 26)	(3, 27)	(3, 28)	(3, 29)	(3, 30)	(3, 31)	(3, 16)	(3, 17)	(3, 18)
(4, 20)	(4, 21)	(4, 22)	(4, 23)	(4, 24)	(4, 25)	(4, 26)	(4, 27)	(4, 28)	(4, 29)	(4, 30)	(4, 31)	(4, 16)	(4, 17)	(4, 18)	(4, 19)
(5, 21)	(5, 22)	(5, 23)	(5, 24)	(5, 25)	(5, 26)	(5, 27)	(5, 28)	(5, 29)	(5, 30)	(5, 31)	(5, 16)	(5, 17)	(5, 18)	(5, 19)	(5, 20)
(6, 22)	(6, 23)	(6, 24)	(6, 25)	(6, 26)	(6, 27)	(6, 28)	(6, 29)	(6, 30)	(6, 31)	(6, 16)	(6, 17)	(6, 18)	(6, 19)	(6, 20)	(6, 21)
(7, 23)	(7, 24)	(7, 25)	(7, 26)	(7, 27)	(7, 28)	(7, 29)	(7, 30)	(7, 31)	(7, 16)	(7, 17)	(7, 18)	(7, 19)	(7, 20)	(7, 21)	(7, 22)
(8, 24)	(8, 25)	(8, 26)	(8, 27)	(8, 28)	(8, 29)	(8, 30)	(8, 31)	(8, 16)	(8, 17)	(8, 18)	(8, 19)	(8, 20)	(8, 21)	(8, 22)	(8, 23)
(9, 25)	(9, 26)	(9, 27)	(9, 28)	(9, 29)	(9, 30)	(9, 31)	(9, 16)	(9, 17)	(9, 18)	(9, 19)	(9, 20)	(9, 21)	(9, 22)	(9, 23)	(9, 24)
(10, 26)	(10, 27)	(10, 28)	(10, 29)	(10, 30)	(10, 31)	(10, 16)	(10, 17)	(10, 18)	(10, 19)	(10, 20)	(10, 21)	(10, 22)	(10, 23)	(10, 24)	(10, 25)
(11, 27)	(11, 28)	(11, 29)	(11, 30)	(11, 31)	(11, 16)	(11, 17)	(11, 18)	(11, 19)	(11, 20)	(11, 21)	(11, 22)	(11, 23)	(11, 24)	(11, 25)	(11, 26)
(12, 28)	(12, 29)	(12, 30)	(12, 31)	(12, 16)	(12, 17)	(12, 18)	(12, 19)	(12, 20)	(12, 21)	(12, 22)	(12, 23)	(12, 24)	(12, 25)	(12, 26)	(12, 27)
(13, 29)	(13, 30)	(13, 31)	(13, 16)	(13, 17)	(13, 18)	(13, 19)	(13, 20)	(13, 21)	(13, 22)	(13, 23)	(13, 24)	(13, 25)	(13, 26)	(13, 27)	(13, 28)
(14, 30)	(14, 31)	(14, 16)	(14, 17)	(14, 18)	(14, 19)	(14, 20)	(14, 21)	(14, 22)	(14, 23)	(14, 24)	(14, 25)	(14, 26)	(14, 27)	(14, 28)	(14, 29)
(15, 31)	(15, 16)	(15, 17)	(15, 18)	(15, 19)	(15, 20)	(15, 21)	(15, 22)	(15, 23)	(15, 24)	(15, 25)	(15, 26)	(15, 27)	(15, 28)	(15, 29)	(15, 30)

Figura 3.4: Torneio $P_{16,0}$.

Rodadas														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\overline{P}_{16,0}[0,7] \oslash (\overline{U}_{8,0} \odot \overline{U}_{8,8} \odot \overline{U}_{8,16} \odot \overline{U}_{8,24})$														
(16,0)	(7,0)	(17,0)	(7,1)	(18,0)	(7,2)	(19,0)	(7,3)	(20,0)	(7,4)	(21,0)	(7,5)	(22,0)	(7,6)	(23,0)
(17,1)	(6,1)	(18,1)	(0,2)	(19,1)	(1,3)	(20,1)	(2,4)	(21,1)	(3,5)	(22,1)	(4,6)	(23,1)	(5,0)	(24,1)
(18,2)	(5,2)	(19,2)	(6,3)	(20,2)	(0,4)	(21,2)	(1,5)	(22,2)	(2,6)	(23,2)	(3,0)	(24,2)	(4,1)	(25,2)
(19,3)	(4,3)	(20,3)	(5,4)	(21,3)	(6,5)	(22,3)	(0,6)	(23,3)	(1,0)	(24,3)	(2,1)	(25,3)	(3,2)	(26,3)
(20,4)	(15,8)	(21,4)	(15,9)	(22,4)	(15,10)	(23,4)	(15,11)	(24,4)	(15,12)	(25,4)	(15,13)	(26,4)	(15,14)	(27,4)
(21,5)	(14,9)	(22,5)	(8,10)	(23,5)	(9,11)	(24,5)	(10,12)	(25,5)	(11,13)	(26,5)	(12,14)	(27,5)	(13,8)	(28,5)
(22,6)	(13,10)	(23,6)	(14,11)	(24,6)	(8,12)	(25,6)	(9,13)	(26,6)	(10,14)	(27,6)	(11,8)	(28,6)	(12,9)	(29,6)
(23,7)	(12,11)	(24,7)	(13,12)	(25,7)	(14,13)	(26,7)	(8,14)	(27,7)	(9,8)	(28,7)	(10,9)	(29,7)	(11,10)	(30,7)
(24,8)	(23,16)	(25,8)	(23,17)	(26,8)	(23,18)	(27,8)	(23,19)	(28,8)	(23,20)	(29,8)	(23,21)	(30,8)	(23,22)	(31,8)
(25,9)	(22,17)	(26,9)	(16,18)	(27,9)	(17,19)	(28,9)	(18,20)	(29,9)	(19,21)	(30,9)	(20,22)	(31,9)	(21,16)	(16,9)
(26,10)	(21,18)	(27,10)	(22,19)	(28,10)	(16,20)	(29,10)	(17,21)	(30,10)	(18,22)	(31,10)	(19,16)	(16,10)	(20,17)	(17,10)
(27,11)	(20,19)	(28,11)	(21,20)	(29,11)	(22,21)	(30,11)	(16,22)	(31,11)	(17,16)	(16,11)	(18,17)	(17,11)	(19,18)	(18,11)
(28,12)	(31,24)	(29,12)	(31,25)	(30,12)	(31,26)	(31,12)	(31,27)	(16,12)	(31,28)	(17,12)	(31,29)	(18,12)	(31,30)	(19,12)
(29,13)	(30,25)	(30,13)	(24,26)	(31,13)	(25,27)	(16,13)	(26,28)	(17,13)	(27,29)	(18,13)	(28,30)	(19,13)	(29,24)	(20,13)
(30,14)	(29,26)	(31,14)	(30,27)	(16,14)	(24,28)	(17,14)	(25,29)	(18,14)	(26,30)	(19,14)	(27,24)	(20,14)	(28,25)	(21,14)
(31,15)	(28,27)	(16,15)	(29,28)	(17,15)	(30,29)	(18,15)	(24,30)	(19,15)	(25,24)	(20,15)	(26,25)	(21,15)	(27,26)	(22,15)

Rodadas							
15	16	17	18	19	20	21	22
$P_{16,0}[0,7]$							
(0,16)	(0,17)	(0,18)	(0,19)	(0,20)	(0,21)	(0,22)	(0,23)
(1,17)	(1,18)	(1,19)	(1,20)	(1,21)	(1,22)	(1,23)	(1,24)
(2,18)	(2,19)	(2,20)	(2,21)	(2,22)	(2,23)	(2,24)	(2,25)
(3,19)	(3,20)	(3,21)	(3,22)	(3,23)	(3,24)	(3,25)	(3,26)
(4,20)	(4,21)	(4,22)	(4,23)	(4,24)	(4,25)	(4,26)	(4,27)
(5,21)	(5,22)	(5,23)	(5,24)	(5,25)	(5,26)	(5,27)	(5,28)
(6,22)	(6,23)	(6,24)	(6,25)	(6,26)	(6,27)	(6,28)	(6,29)
(7,23)	(7,24)	(7,25)	(7,26)	(7,27)	(7,28)	(7,29)	(7,30)
(8,24)	(8,25)	(8,26)	(8,27)	(8,28)	(8,29)	(8,30)	(8,31)
(9,25)	(9,26)	(9,27)	(9,28)	(9,29)	(9,30)	(9,31)	(9,16)
(10,26)	(10,27)	(10,28)	(10,29)	(10,30)	(10,31)	(10,16)	(10,17)
(11,27)	(11,28)	(11,29)	(11,30)	(11,31)	(11,16)	(11,17)	(11,18)
(12,28)	(12,29)	(12,30)	(12,31)	(12,16)	(12,17)	(12,18)	(12,19)
(13,29)	(13,30)	(13,31)	(13,16)	(13,17)	(13,18)	(13,19)	(13,20)
(14,30)	(14,31)	(14,16)	(14,17)	(14,18)	(14,19)	(14,20)	(14,21)
(15,31)	(15,16)	(15,17)	(15,18)	(15,19)	(15,20)	(15,21)	(15,22)

Figura 3.5: Rodada 0 à 22 do torneio T com $k = 8$.

Teorema 3.1. *T é um torneio round robin duplo.*

Demonstração. Por definição e pelo lema 3.1, sabemos que $U_{k,0}$ e $U_{k,k}$ são torneios *round robin* simples com times diferentes. $U_{k,0} \odot U_{k,k}$ é um torneio com $k-1$ rodadas nas quais cada par de times no intervalo $[0, k-1]$ se enfrenta exatamente uma vez, assim como cada par de times no intervalo $[k, 2k-1]$. Para este torneio se tornar um torneio *round robin* é necessário que os times em $[0, k-1]$ enfrentem os times em $[k, 2k-1]$ exatamente uma vez durante k rodadas adicionais. Note que isso é exatamente o que ocorre nas k rodadas de $P_{k,0}$. Portanto, $(U_{k,0} \odot U_{k,k}) \oplus P_{k,0}$ é um torneio *round robin* simples. O torneio $(U_{k,2k} \odot U_{k,3k}) \oplus P_{k,2k}$ corresponde ao torneio $(U_{k,0} \odot U_{k,k}) \oplus P_{k,0}$ com todos os índices acrescidos em $2k$. Isto significa que $(U_{k,2k} \odot U_{k,3k}) \oplus P_{k,2k}$ também é um torneio *round robin* simples, e todos os seus times são diferentes daqueles em $(U_{k,0} \odot U_{k,k}) \oplus P_{k,0}$. Combinamos estes dois torneios da mesma forma que $U_{k,0}$ e $U_{k,k}$ foram combinados para obter um torneio *round robin* simples maior. Desta forma, o resultado de $(((U_{k,0} \odot U_{k,k}) \oplus P_{k,0}) \odot ((U_{k,2k} \odot U_{k,3k}) \oplus P_{k,2k})) \oplus P_{2k,0}$ também é um torneio *round robin* simples. Agora perceba que $(((U_{k,0} \odot U_{k,k}) \oplus P_{k,0}) \odot ((U_{k,2k} \odot U_{k,3k}) \oplus P_{k,2k})) \oplus P_{2k,0} = (U_{k,0} \odot U_{k,k} \odot U_{k,2k} \odot U_{k,3k}) \oplus (P_{k,0} \odot P_{k,2k}) \oplus P_{2k,0}$. Ao concatenar este torneio com uma cópia de si mesmo mas com as sedes invertidas, vamos obter um torneio *round robin*

Rodadas														
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
$\overline{P}_{16,0}[8,15] \oslash (U_{8,0} \odot U_{8,8} \odot U_{8,16} \odot U_{8,24})$														
(24,0)	(0,7)	(25,0)	(1,7)	(26,0)	(2,7)	(27,0)	(3,7)	(28,0)	(4,7)	(29,0)	(5,7)	(30,0)	(6,7)	(31,0)
(25,1)	(1,6)	(26,1)	(2,0)	(27,1)	(3,1)	(28,1)	(4,2)	(29,1)	(5,3)	(30,1)	(6,4)	(31,1)	(0,5)	(16,1)
(26,2)	(2,5)	(27,2)	(3,6)	(28,2)	(4,0)	(29,2)	(5,1)	(30,2)	(6,2)	(31,2)	(0,3)	(16,2)	(1,4)	(17,2)
(27,3)	(3,4)	(28,3)	(4,5)	(29,3)	(5,6)	(30,3)	(6,0)	(31,3)	(0,1)	(16,3)	(1,2)	(17,3)	(2,3)	(18,3)
(28,4)	(8,15)	(29,4)	(9,15)	(30,4)	(10,15)	(31,4)	(11,15)	(16,4)	(12,15)	(17,4)	(13,15)	(18,4)	(14,15)	(19,4)
(29,5)	(9,14)	(30,5)	(10,8)	(31,5)	(11,9)	(16,5)	(12,10)	(17,5)	(13,11)	(18,5)	(14,12)	(19,5)	(8,13)	(20,5)
(30,6)	(10,13)	(31,6)	(11,14)	(16,6)	(12,8)	(17,6)	(13,9)	(18,6)	(14,10)	(19,6)	(8,11)	(20,6)	(9,12)	(21,6)
(31,7)	(11,12)	(16,7)	(12,13)	(17,7)	(13,14)	(18,7)	(14,8)	(19,7)	(8,9)	(20,7)	(9,10)	(21,7)	(10,11)	(22,7)
(16,8)	(16,23)	(17,8)	(17,23)	(18,8)	(18,23)	(19,8)	(19,23)	(20,8)	(20,23)	(21,8)	(21,23)	(22,8)	(22,23)	(23,8)
(17,9)	(17,22)	(18,9)	(18,16)	(19,9)	(19,17)	(20,9)	(20,18)	(21,9)	(21,19)	(22,9)	(22,20)	(23,9)	(16,21)	(24,9)
(18,10)	(18,21)	(19,10)	(19,22)	(20,10)	(20,16)	(21,10)	(21,17)	(22,10)	(22,18)	(23,10)	(16,19)	(24,10)	(17,20)	(25,10)
(19,11)	(19,20)	(20,11)	(20,21)	(21,11)	(21,22)	(22,11)	(22,16)	(23,11)	(16,17)	(24,11)	(17,18)	(25,11)	(18,19)	(26,11)
(20,12)	(24,31)	(21,12)	(25,31)	(22,12)	(26,31)	(23,12)	(27,31)	(24,12)	(28,31)	(25,12)	(29,31)	(26,12)	(30,31)	(27,12)
(21,13)	(25,30)	(22,13)	(26,24)	(23,13)	(27,25)	(24,13)	(28,26)	(25,13)	(29,27)	(26,13)	(30,28)	(27,13)	(24,29)	(28,13)
(22,14)	(26,29)	(23,14)	(27,30)	(24,14)	(28,24)	(25,14)	(29,25)	(26,14)	(30,26)	(27,14)	(24,27)	(28,14)	(25,28)	(29,14)
(23,15)	(27,28)	(24,15)	(28,29)	(25,15)	(29,30)	(26,15)	(30,24)	(27,15)	(24,25)	(28,15)	(25,26)	(29,15)	(26,27)	(30,15)

Rodadas							
38	39	40	41	42	43	44	45
$P_{16,0}[8,15]$							
(0,24)	(0,25)	(0,26)	(0,27)	(0,28)	(0,29)	(0,30)	(0,31)
(1,25)	(1,26)	(1,27)	(1,28)	(1,29)	(1,30)	(1,31)	(1,16)
(2,26)	(2,27)	(2,28)	(2,29)	(2,30)	(2,31)	(2,16)	(2,17)
(3,27)	(3,28)	(3,29)	(3,30)	(3,31)	(3,16)	(3,17)	(3,18)
(4,28)	(4,29)	(4,30)	(4,31)	(4,16)	(4,17)	(4,18)	(4,19)
(5,29)	(5,30)	(5,31)	(5,16)	(5,17)	(5,18)	(5,19)	(5,20)
(6,30)	(6,31)	(6,16)	(6,17)	(6,18)	(6,19)	(6,20)	(6,21)
(7,31)	(7,16)	(7,17)	(7,18)	(7,19)	(7,20)	(7,21)	(7,22)
(8,16)	(8,17)	(8,18)	(8,19)	(8,20)	(8,21)	(8,22)	(8,23)
(9,17)	(9,18)	(9,19)	(9,20)	(9,21)	(9,22)	(9,23)	(9,24)
(10,18)	(10,19)	(10,20)	(10,21)	(10,22)	(10,23)	(10,24)	(10,25)
(11,19)	(11,20)	(11,21)	(11,22)	(11,23)	(11,24)	(11,25)	(11,26)
(12,20)	(12,21)	(12,22)	(12,23)	(12,24)	(12,25)	(12,26)	(12,27)
(13,21)	(13,22)	(13,23)	(13,24)	(13,25)	(13,26)	(13,27)	(13,28)
(14,22)	(14,23)	(14,24)	(14,25)	(14,26)	(14,27)	(14,28)	(14,29)
(15,23)	(15,24)	(15,25)	(15,26)	(15,27)	(15,28)	(15,29)	(15,30)

Rodadas															
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
$(\overline{P}_{8,0} \odot \overline{P}_{8,16}) \oslash (P_{8,0} \odot P_{8,16})$															
(8,0)	(0,8)	(9,0)	(0,9)	(10,0)	(0,10)	(11,0)	(0,11)	(12,0)	(0,12)	(13,0)	(0,13)	(14,0)	(0,14)	(15,0)	(0,15)
(9,1)	(1,9)	(10,1)	(1,10)	(11,1)	(1,11)	(12,1)	(1,12)	(13,1)	(1,13)	(14,1)	(1,14)	(15,1)	(1,15)	(8,1)	(1,8)
(10,2)	(2,10)	(11,2)	(2,11)	(12,2)	(2,12)	(13,2)	(2,13)	(14,2)	(2,14)	(15,2)	(2,15)	(8,2)	(2,8)	(9,2)	(2,9)
(11,3)	(3,11)	(12,3)	(3,12)	(13,3)	(3,13)	(14,3)	(3,14)	(15,3)	(3,15)	(8,3)	(3,8)	(9,3)	(3,9)	(10,3)	(3,10)
(12,4)	(4,12)	(13,4)	(4,13)	(14,4)	(4,14)	(15,4)	(4,15)	(8,4)	(4,8)	(9,4)	(4,9)	(10,4)	(4,10)	(11,4)	(4,11)
(13,5)	(5,13)	(14,5)	(5,14)	(15,5)	(5,15)	(8,5)	(5,8)	(9,5)	(5,9)	(10,5)	(5,10)	(11,5)	(5,11)	(12,5)	(5,12)
(14,6)	(6,14)	(15,6)	(6,15)	(8,6)	(6,8)	(9,6)	(6,9)	(10,6)	(6,10)	(11,6)	(6,11)	(12,6)	(6,12)	(13,6)	(6,13)
(15,7)	(7,15)	(8,7)	(7,8)	(9,7)	(7,9)	(10,7)	(7,10)	(11,7)	(7,11)	(12,7)	(7,12)	(13,7)	(7,13)	(14,7)	(7,14)
(24,16)	(16,24)	(25,16)	(16,25)	(26,16)	(16,26)	(27,16)	(16,27)	(28,16)	(16,28)	(29,16)	(16,29)	(30,16)	(16,30)	(31,16)	(16,31)
(25,17)	(17,25)	(26,17)	(17,26)	(27,17)	(17,27)	(28,17)	(17,28)	(29,17)	(17,29)	(30,17)	(17,30)	(31,17)	(17,31)	(24,17)	(17,24)
(26,18)	(18,26)	(27,18)	(18,27)	(28,18)	(18,28)	(29,18)	(18,29)	(30,18)	(18,30)	(31,18)	(18,31)	(24,18)	(18,24)	(25,18)	(18,25)
(27,19)	(19,27)	(28,19)	(19,28)	(29,19)	(19,29)	(30,19)	(19,30)	(31,19)	(19,31)	(24,19)	(19,24)	(25,19)	(19,25)	(26,19)	(19,26)
(28,20)	(20,28)	(29,20)	(20,29)	(30,20)	(20,30)	(31,20)	(20,31)	(24,20)	(20,24)	(25,20)	(20,25)	(26,20)	(20,26)	(27,20)	(20,27)
(29,21)	(21,29)	(30,21)	(21,30)	(31,21)	(21,31)	(24,21)	(21,24)	(25,21)	(21,25)	(26,21)	(21,26)	(27,21)	(21,27)	(28,21)	(21,28)
(30,22)	(22,30)	(31,22)	(22,31)	(24,22)	(22,24)	(25,22)	(22,25)	(26,22)	(22,26)	(27,22)	(22,27)	(28,22)	(22,28)	(29,22)	(22,29)
(31,23)	(23,31)	(24,23)	(23,24)	(25,23)	(23,25)	(26,23)	(23,26)	(27,23)	(23,27)	(28,23)	(23,28)	(29,23)	(23,29)	(30,23)	(23,30)

Figura 3.6: Rodada 23 à 61 do torneio T com $k = 8$.

duplo onde cada time joga contra todos os demais duas vezes, uma vez em casa e outra fora. Formalmente, o resultado é $(U_{k,0} \odot U_{k,k} \odot U_{k,2k} \odot U_{k,3k}) \oplus (P_{k,0} \odot P_{k,2k}) \oplus P_{2k,0} \oplus (\overline{U}_{k,0} \odot \overline{U}_{k,k} \odot \overline{U}_{k,2k} \odot \overline{U}_{k,3k}) \oplus (\overline{P}_{k,0} \odot \overline{P}_{k,2k}) \oplus \overline{P}_{2k,0}$. Por fim, note que podemos obter T rearranjando a ordem das rodadas deste último torneio, o que completa a prova. □

Este teorema assegura que (3.18)–(3.21) definem uma forma correta de construir um

torneio *round robin* duplo. Existem na literatura diversos outros métodos para elaboração de torneios *round robin* (simples ou duplo). As revisões bibliográficas apresentadas em [25] e [37] elencam várias referências que descrevem formas de criar tais torneios com diferentes propriedades. Os métodos mais comumente empregados na literatura são: o método de Kirkman/círculo/polígono [10, 13, 22, 26, 28, 30], visto anteriormente, e um outro baseado na teoria dos grafos que consiste em associar os times aos vértices de um grafo completo e obter uma 1-fatoração (decomposição do grafo em emparelhamentos perfeitos, veja a seção 3.3 em [52]) deste grafo [9, 11, 12, 29, 38, 39]. Em particular, na seção 4 em [9] é descrito um tipo de torneio com dois grupos de times intercalando rodadas com partidas apenas entre times do mesmo grupo e rodadas com partidas apenas entre times de grupos diferentes. Este tipo de torneio é semelhante ao obtido pela operação $P_{k,0} \oslash (U_{k,0} \odot U_{k,k})$ que é um rearranjo das rodadas de $(U_{k,0} \odot U_{k,k}) \oplus P_{k,0}$ visto na prova do teorema acima.

Na próxima seção vamos usar o torneio T para reduzir em tempo polinomial uma instância de um outro problema $\mathcal{NP}$-completo a uma instância do TUP.

3.2 Redução em tempo polinomial

Para concluirmos que o TUP é $\mathcal{NP}$-completo, vamos mostrar que o problema de determinar se existe ou não um ciclo hamiltoniano em um grafo com uma quantidade par de vértices e ao menos um vértice universal (adjacente a todos os outros vértices) é $\mathcal{NP}$-completo, e então reduzir este problema ao TUP em tempo polinomial.

Lema 3.2. *Decidir se um grafo com uma quantidade par de vértices e ao menos um vértice universal possui um ciclo hamiltoniano é um problema $\mathcal{NP}$-completo.*

Demonstração. Decidir se um grafo geral possui um caminho hamiltoniano é um problema $\mathcal{NP}$-completo (veja [21]). Seja $G = (V, E)$ um grafo geral e, portanto, uma instância do problema do caminho hamiltoniano. Vamos construir a partir de G, em tempo polinomial, um grafo G' que é uma instância do problema estabelecido no enunciado deste lema. Primeiramente, considere $G' = G$ e os dois seguintes casos (exemplificados na figura 3.7):

Caso 1: G possui uma quantidade ímpar de vértices. Adicione um vértice universal p em G', isto é, p é um novo vértices adjacente a todos os demais em G'. Isto pode ser feito em tempo $O(|V|)$.

Caso 2: G possui uma quantidade par de vértices. Adicione quatro novos vértices em G': p, q, r e s. Faça p e q se tornarem vértices universais (também adjacentes entre si, e a r e s), e adicione uma aresta entre r e s em G'. Isto também toma tempo $O(|V|)$.

É evidente que o problema do enunciado está em $\mathcal{NP}$ já que este é um caso particular do problema do ciclo hamiltoniano. Desta forma, nos resta verificar que existe um caminho hamiltoniano em G se, e somente se, existe um ciclo hamiltoniano em G'.

($\Rightarrow$) Se G possui um caminho hamiltoniano, então este caminho também está em G'. No caso 1, ao conectar as duas extremidades deste caminho em p criamos um ciclo hamiltoniano em G'. No caso 2, vamos obter um ciclo hamiltoniano em G' da seguinte forma: conecte uma das extremidades do caminho hamiltoniano em p, conecte p em s, s em r, r em q, e finalmente conecte q na outra extremidade do caminho hamiltoniano.

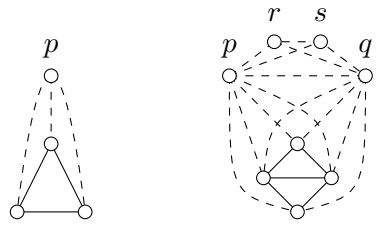


Figura 3.7: Exemplos dos dois casos (número par e ímpar de vértices) da redução do problema do caminho hamiltoniano ao problema enunciado no lema 3.2. As arestas tracejadas e os vértices p, q, r e s são adicionados no grafo original para obter a instância reduzida.

($\Leftarrow$) Suponha que G' possui um ciclo hamiltoniano. No caso 1, simplesmente remova p do ciclo para resultar em um caminho hamiltoniano em G. No caso 2, a única forma de alcançar os vértices r e s é através de p e q. Isto significa que qualquer ciclo hamiltoniano em G' passa por p (ou q), segue imediatamente para r e s (em qualquer ordem), e então vai para q (ou p). Portanto, ao remover p, q, r e s do ciclo hamiltoniano em G' terminamos com um caminho hamiltoniano em G. □

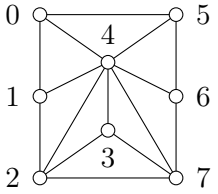
Vamos mostrar agora como converter, em tempo polinomial, uma instância do problema de decisão descrito no enunciado do lema 3.2 em uma instância do TUP com $d_1 \leq n/2$ e $d_2 = \lfloor n/2 \rfloor - 1$.

Definição 3.1. *Dado um grafo G com uma quantidade par de vértices $k \geq 4$ e ao menos um vértice universal, definimos $I(G)$ como sendo a instância do TUP criada da seguinte forma. O torneio T desta instância é o torneio com $4k$ times definido por (3.18)–(3.21), no qual vamos ter uma correspondência um-para-um arbitrária entre os times $0, \ldots, k-1$ e os vértices de G. O número de árbitros é $n = 2k$, $0 \leq d_1 \leq n/2$ e $d_2 = \lfloor n/2 \rfloor - 1$. Por fim, para cada par de times i e j, $d_{ij} = 0$ se $0 \leq i, j \leq k-1$ e os vértices que correspondem aos times i e j são adjacentes em G. Caso contrário, $d_{ij} = 1$.*

A figura 3.8 ilustra um grafo G com 8 vértices, sendo um deles universal, (instância da variação do problema do ciclo hamiltoniano vista no lema 3.2) e a matriz de distâncias entre as sedes d_{ij} da instância $I(G)$ do TUP apresentada na definição acima. O torneio T de $I(G)$ com 32 times neste exemplo é aquele apresentado nas figuras 3.5 e 3.6. As sedes $0, \ldots, 7$ correspondem aos vértices do grafo G. A distância entre duas sedes é 0 quando ambas correspondem a vértices adjacentes no grafo, e 1 caso contrário.

Teorema 3.2. *Seja G um grafo com uma quantidade par de vértices $k \geq 4$ e ao menos um vértice universal. G possui um ciclo hamiltoniano se, e somente se, a instância $I(G)$ do TUP determinada na definição 3.1 possuir uma solução ótima com distância total igual a $n(4n-3) - 2k(k-1)$, onde $n = 2k$ é o número de árbitros.*

Antes de provar o teorema 3.2 será necessário introduzir duas sequências numéricas auxiliares e demonstrar algumas de suas propriedades.



Sede	0	1	2	3	4	5	6	7	8	⋯	31
0	1	0	1	1	0	0	1	1	1	⋯	1
1	0	1	0	1	0	1	1	1	1	⋯	1
2	1	0	1	0	0	1	1	0	1	⋯	1
3	1	1	0	1	0	1	1	0	1	⋯	1
4	0	0	0	0	1	0	0	0	1	⋯	1
5	0	1	1	1	0	1	0	1	1	⋯	1
6	1	1	1	1	0	0	1	0	1	⋯	1
7	1	1	0	0	0	1	0	1	1	⋯	1
8	1	1	1	1	1	1	1	1	1	⋯	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
31	1	1	1	1	1	1	1	1	1	⋯	1

Figura 3.8: Exemplo de um grafo G com 8 vértices (à esquerda) e a matriz de distâncias entre as sedes da instância $I(G)$ do TUP (à direita).

Dado um inteiro positivo a e três inteiros não-negativos b, c e d, definimos $Z_{a,b,c,d}$ como sendo a sequência estabelecida por

$$Z_{a,b,c,d} = d + (b \bmod a), d + ((b+1) \bmod a), ..., d + ((b+c-1) \bmod a). \tag{3.22}$$

Note que $Z_{a,b,c,d}$ possui tamanho c e seus números pertencem ao intervalo $[d, d+a-1]$, sendo que o primeiro número é determinado pelo valor de b. A sequência consiste de inteiros consecutivos exceto quando o maior valor no intervalo é atingido, pois neste momento o próximo número é o menor número do intervalo. Por isso, se $c \leq a$, todos os números em $Z_{a,b,c,d}$ são diferentes. As sequências definidas por (3.22) também obedecem à seguinte propriedade.

Lema 3.3. *Dadas duas sequências $Z_{a,b,c,d}$ e $Z_{a+1,b,c+1,d}$, se $c \leq a$ e $c+b \leq 2a$, o i-ésimo número em $Z_{a,b,c,d}$ ocorre exatamente uma vez em $Z_{a+1,b,c+1,d}$, sendo o i-ésimo ou o $(i+1)$-ésimo número desta última sequência.*

Demonstração. Considere as sequências $Z_{a,0,2a,d} = d, d+1, \ldots, d+a-1, d, d+1, \ldots, d+a-1$ e $Z_{a+1,0,2a+1,d} = d, d+1, \ldots, d+a-1, d+a, d, d+1, \ldots, d+a-1$. O i-ésimo número em $Z_{a,0,2a,d}$ é igual ao i-ésimo ou $(i+1)$-ésimo número em $Z_{a+1,0,2a+1,d}$. Impondo $c+b \leq 2a$ implica que as sequências $Z_{a,b,c,d}$ e $Z_{a+1,b,c+1,d}$ são, respectivamente, subsequências de $Z_{a,0,2a,d}$ e $Z_{a+1,0,2a+1,d}$ que iniciam no $(b+1)$-ésimo número destas últimas sequências. Então, o i-ésimo número em $Z_{a,b,c,d}$ é igual ao i-ésimo ou $(i+1)$-ésimo número em $Z_{a+1,b,c+1,d}$. Além disso, pelo fato de termos $c+1 \leq a+1$, todos os números em $Z_{a+1,b,c+1,d}$ são diferentes, implicando que o i-ésimo número em $Z_{a,b,c,d}$ ocorre exatamente uma vez em $Z_{a+1,b,c+1,d}$. □

Nossa segunda sequência auxiliar $Y_{a,b,c,d,e}$ é definida por (3.23)–(3.25), onde $a \geq 4$ é

um inteiro positivo par, b, c, d e e são inteiros não negativos, e $0 \leq b \leq a/2 - 1$.

$$Y_{a,b,c,d,e} = (c,d) \oplus Z_{a-1,a+b,a-2b-3,e}, \quad \text{se} \quad b = 0, \tag{3.23}$$

$$Y_{a,b,c,d,e} = Z_{a-1,a-b-1,2b,e} \oplus (c,d) \oplus Z_{a-1,a+b,a-2b-3,e}, \quad \text{se} \quad 1 \leq b \leq a/2 - 2, \tag{3.24}$$

$$Y_{a,b,c,d,e} = Z_{a-1,a-b-1,2b,e} \oplus (c), \quad \text{se} \quad b = a/2 - 1. \tag{3.25}$$

Os dois seguintes lemas estabelecem propriedades úteis das sequências $Y_{a,b,c,d,e}$.

Lema 3.4. *Seja $a \geq 4$ um inteiro par, b, c, d e e inteiros não-negativos, $c \neq d$ e $0 \leq b \leq a/2 - 1$. Se c e d não pertencem ao intervalo $[e, e + a - 2]$, então os números em $Y_{a,b,c,d,e}$ são todos diferentes.*

Demonstração. É claro que os valores permitidos para c e d não aparecem em $Z_{a-1,a-b-1,2b,e}$ e nem em $Z_{a-1,a+b,a-2b-3,e}$. Note que $Z_{a-1,a-b-1,a-2,e}$, cujos números são todos diferentes entre si pois esta sequência possui $a - 2$ elementos, é equivalente a $Z_{a-1,a-b-1,2b,e} \oplus Z_{a-1,a+b-1,1,e} \oplus Z_{a-1,a+b,a-2b-3,e}$. Logo, $Z_{a-1,a-b-1,2b,e}$ e $Z_{a-1,a+b,a-2b-3,e}$ juntas não apresentam repetições de números. □

Lema 3.5. *Seja $a \geq 4$ um inteiro par, e b, c, d e e inteiros não-negativos. Dado um inteiro não-negativo $i \leq a - 2$, o conjunto composto pelos $(i+1)$-ésimos números das sequências $Y_{a,b,c,d,e}$ obtidas para cada valor de $b \in \{0, \ldots, a/2 - 1\}$ contém todos os números no intervalo $[e + ((i + a/2) \bmod (a-1)), e + ((i + a - 2) \bmod (a-1))]$. Este conjunto também contém c quando i é par, ou d quando i é ímpar. Além disso, se c e d não pertencerem ao intervalo $[e, e + a - 2]$, os $(i+1)$-ésimos números em todas estas sequências $Y_{a,b,c,d,e}$ são diferentes entre si.*

Demonstração. Por definição, o $(i+1)$-ésimo número em $Y_{a,\lfloor i/2 \rfloor,c,d,e}$ é c quando i é par, ou d quando i é ímpar. O $(i+1)$-ésimo número em cada uma das $a/2 - 1$ demais sequências $Y_{a,b,c,d,e}$ (para $b \neq \lfloor i/2 \rfloor$) pode ser determinado da seguinte forma. Se $i \leq 1$, o $(i+1)$-ésimo número em $Y_{a,b,c,d,e}$ para $1 \leq b \leq a/2 - 1$ é dado por $e + (((a - b - 1) + i) \bmod (a-1))$ pois ele é o $(i+1)$-ésimo número na subsequência $Z_{a-1,a-b-1,2b,e}$ em (3.24) ou (3.25). Se $i = a - 2$, o $(i+1)$-ésimo número em $Y_{a,b,c,d,e}$ para $0 \leq b \leq a/2 - 2$ é dado por $e + (((a + b) + i - (2b + 2)) \bmod (a - 1))$ pois ele é o $(i - (2b + 2) + 1)$-ésimo número na subsequência $Z_{a-1,a+b,a-2b-3,e}$ em (3.23) ou (3.24). Se $2 \leq i \leq a - 3$, ambos casos anteriores podem acontecer pois o $(i+1)$-ésimo número em $Y_{a,b,c,d,e}$, dependendo do valor de b, pode estar na subsequência $Z_{a-1,a+b,a-2b-3,e}$ em (3.23) ou (3.24), ou na subsequência $Z_{a-1,a-b-1,2b,e}$ em (3.24) ou (3.25). O $(i+1)$-ésimo número em $Y_{a,b,c,d,e}$ é dado por $e + (((a + b) + i - (2b + 2)) \bmod (a - 1))$ para $0 \leq b \leq \lfloor i/2 \rfloor - 1$, e dado por $e + (((a - b - 1) + i) \bmod (a - 1))$ para $\lfloor i/2 \rfloor + 1 \leq b \leq a/2 - 1$. Ao variar o valor de $b \neq \lfloor i/2 \rfloor$, em ordem decrescente, para um valor fixo de i como mencionado acima, verifica-se que os $(i+1)$-ésimos números obtidos são exatamente todos aqueles no intervalo $[e + ((i + a/2) \bmod (a-1)), e + ((i + a - 2) \bmod (a-1))]$. Por fim, note que este intervalo contém $a/2 - 1$ números distintos e, pela definição do lema, não inclui c e d. Portanto, os $(i+1)$-ésimos números das $a/2$ sequências $Y_{a,b,c,d,e}$ são todos diferentes. □

Agora estamos prontos para provar o teorema 3.2.

Demonstração do teorema 3.2. Pelo fato do torneio T possuir $4n-2$ rodadas, toda solução para $I(G)$ tem exatamente o total de $n(4n-3)$ viagens (cada um dos n árbitros realiza $4n-3$ viagens). A partir da definição de d_{ij}, sabemos que as distâncias das viagens são iguais a 0 ou 1. Para simplificar, vamos denominar por *viagem nula* e *viagem unitária* as viagens de distância 0 e 1, respectivamente. Da forma com que os jogos e as rodadas de T foram organizados, as viagens nulas só podem ocorrer entre rodadas consecutivas nos intervalos $[2k-1, 3k-2]$ e $[5k-2, 6k-3]$, pois estas são as únicas rodadas com jogos consecutivos nas sedes dos times $0, \ldots, k-1$. Note que é possível ter no máximo k viagens nulas entre cada par de rodadas consecutivas nestes dois intervalos. Por isso, podem haver no máximo $2k(k-1)$ viagens nulas durante todo o torneio. Logo, qualquer solução com distância total $n(4n-3) - 2k(k-1)$ é ótima.

($\Leftarrow$) Por hipótese, existe uma solução S para $I(G)$ com distância total igual a $n(4n-3) - 2k(k-1)$. Isso implica que k árbitros em S realizam viagens nulas entre cada par de rodadas consecutivas nos intervalos $[2k-1, 3k-2]$ e $[5k-2, 6k-3]$. Seja u um árbitro que, na rodada $2k-1$, está na sede de um time associado a um vértice universal de G. Se u não realizar uma viagem nula da rodada $2k-1$ para a rodada $2k$, então não podem existir k viagens nulas entre estas duas rodadas pois a sede de u está entre as únicas k sedes que originam viagens nulas. Como consequência, na rodada $2k$, u novamente estará em uma sede no intervalo $[0, k-1]$. O argumento anterior pode ser reaplicado para a viagem de u da rodada $2k$ para a rodada $2k+1$, e ser estendido para todo o caminho percorrido por u até a rodada $3k-2$. Isto é, todas as viagens de u são nulas (partindo e chegando em sedes no intervalo $[0, k-1]$) da rodada $2k-1$ até a rodada $3k-2$. Visto que $d_1 \leq n/2$, temos $n - d_1 \geq n/2 = k$. Já que S é uma solução factível para o TUP, a restrição (2.4) implica que as k sedes visitadas por u da rodada $2k-1$ até a rodada $3k-2$ são todas diferentes. Portanto, a rota percorrida por u corresponde a um caminho hamiltoniano em G. Além disso, como u está no início desta rota em uma sede que corresponde a um vértice universal, as extremidades deste caminho hamiltoniano podem ser conectadas para formar um ciclo hamiltoniano em G.

($\Rightarrow$) Por hipótese, G possui um ciclo hamiltoniano $C = v_0, v_1, \ldots, v_{k-1}, v_0$, onde cada v_i ($i = 0, \ldots, k-1$) é o índice do time que corresponde ao vértice no ciclo. A partir de C, vamos criar uma solução S para a instância $I(G)$ do TUP com distância total $n(4n-3) - 2k(k-1)$. (Relembre que a $(i+1)$-ésima sede é visitada por um árbitro na rodada i de T porque os índices das rodadas começam em zero.) Para cada árbitro

$u \in \{0, \ldots, 2k-1\}$, a sequência de sedes visitadas por u em S é determinada por

$$(Z_{k,u,k,3k} \oslash Y_{k,u,k-1,3k-1,0}) \oplus Z_{k,u,k,k} \oplus (Z_{k,u,k,3k} \oslash Z_{k-1,u,k-1,2k}) \oplus W_u \oplus (Z_{k,u+(k/2),k,k} \oslash Z_{k,u,k,2k}), \quad \forall\ 0 \leq u \leq k/2-1, \tag{3.26}$$

$$(Z_{k,u',k,2k} \oslash Y_{k,u',2k-1,4k-1,k}) \oplus W_{u'} \oplus (Z_{k,u',k,2k} \oslash Z_{k-1,u',k-1,3k}) \oplus Z_{k,u',k,k} \oplus (Z_{k,u',k,3k} \oslash Z_{k,u,k,0}), \quad \forall\ k/2 \leq u \leq k-1, \tag{3.27}$$

$$(Z_{k,u',k,3k} \oslash Y_{k,u-k,3k-1,k-1,2k}) \oplus Z_{k,u',k,k} \oplus (Z_{k,2k-u-1,k,2k} \oslash Z_{k-1,u-k,k-1,0}) \oplus Z_{k,u',k,k} \oplus (Z_{k,u',k,3k} \oslash Z_{k,u-k,k,0}), \quad \forall\ k \leq u \leq 3k/2-1, \tag{3.28}$$

$$(Z_{k,u-k,k,2k} \oslash Y_{k,u'',4k-1,2k-1,3k}) \oplus W_{u-k} \oplus (Z_{k,k-u''-1,k,3k} \oslash Z_{k-1,u'',k-1,k}) \oplus W_{u-k} \oplus (Z_{k,u'',k,k} \oslash Z_{k,u-k,k,2k}), \quad \forall\ 3k/2 \leq u \leq 2k-1, \tag{3.29}$$

onde $u' = u - k/2$, $u'' = u - 3k/2$ e $W_a = v_{(a \bmod k)}, v_{((a+1) \bmod k)}, \ldots, v_{((a+k-1) \bmod k)}$ para $0 \leq a \leq k-1$. A figura 3.9 ilustra S para $k = 8$ e $C = 0, 1, 2, 3, 4, 5, 6, 7, 0$, com cada subsequência de (3.26)–(3.29) especificada acima das sedes visitadas pelos respectivos árbitros.

Vamos mostrar agora que S é uma solução factível para a instância $I(G)$ com distância total igual a $n(4n-3) - 2k(k-1)$. Antes, note que não será necessário nos preocupar com a restrição (2.5) porque $d_2 = \lfloor n/2 \rfloor - 1$ implica que ela é trivialmente satisfeita.

As restrições (2.1) e (2.2) são satisfeitas por S se, para cada rodada de T, a sede de cada jogo é atribuída a um árbitro diferente por (3.26)–(3.29). Para verificar isto é necessário confrontar as sedes definidas para o jogos em T por (3.18)–(3.21) com as sedes em S atribuídas aos árbitros por (3.26)–(3.29). Criamos a tabela 3.1 para facilitar esta tarefa. Nela apresentamos as rodadas de T convenientemente separadas em grupos que correspondem aos subtorneios em (3.18)–(3.21) que compõem T. A tabela também apresenta os árbitros atribuídos às sedes dos subtorneios, bem como as subsequências em (3.26)–(3.29) que definem estas atribuições. Desta forma, a tabela 3.1 servirá de auxílio para facilitar o entendimento da demonstração apresentada a seguir que, baseada nas definições de U, $\overline{U}$, P, $\overline{P}$, W, Y e Z, garante que as atribuições dos árbitros em S estão de acordo com as sedes dos jogos em T.

As sedes na $(i+1)$-ésima rodada de $\overline{U}_{k,b}$ são $b + ((i + k/2) \bmod (k-1)), \ldots, b + ((i+k-2) \bmod (k-1))$ e $b + k - 1$, para $0 \leq i \leq k-2$. O lema 3.5 assegura que, para $0 \leq b \leq k/2-1$, as $(i+1)$-ésimas sedes em $Y_{k,b,c,d,e}$ são $e + ((i + k/2) \bmod (k-1)), \ldots, e + ((i+k-2) \bmod (k-1))$ e c, se i é par, ou d caso contrário. Com base nestas duas observações podemos ver que, durante as rodadas $1, 3, \ldots, 2k-3$, as sedes em $\overline{U}_{k,0}$ e $\overline{U}_{k,2k}$ são corretamente atribuídas por $Y_{k,u,k-1,3k-1,0}$ em (3.26) e por $Y_{k,u-k,3k-1,k-1,2k}$ em (3.28). De forma análoga, as sedes em $\overline{U}_{k,k}$ e $\overline{U}_{k,3k}$ são também devidamente atribuídas por $Y_{k,u',2k-1,4k-1,k}$ em (3.27) e por $Y_{k,u'',4k-1,2k-1,3k}$ em (3.29).

Igualmente, as sedes na $(i+1)$-ésima rodada de $U_{k,b}$ são $b + (i \bmod (k-1)), \ldots, b + ((i + k/2 - 1) \bmod (k-1))$, para $0 \leq i \leq k-2$, ao passo que as $(i+1)$-ésimas sedes em

	Rodadas																						
Árbitro	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	$(Z_{8,u,8,24} \oslash Y_{8,u,7,23,0})$															$Z_{8,u,8,8}$							
0	24	7	25	23	26	1	27	2	28	3	29	4	30	5	31	8	9	10	11	12	13	14	15
1	25	6	26	0	27	7	28	23	29	2	30	3	31	4	24	9	10	11	12	13	14	15	8
2	26	5	27	6	28	0	29	1	30	7	31	23	24	3	25	10	11	12	13	14	15	8	9
3	27	4	28	5	29	6	30	0	31	1	24	2	25	7	26	11	12	13	14	15	8	9	10
	$(Z_{8,u',8,16} \oslash Y_{8,u',15,31,8})$															$W_{u'}$							
4	16	15	17	31	18	9	19	10	20	11	21	12	22	13	23	0	1	2	3	4	5	6	7
5	17	14	18	8	19	15	20	31	21	10	22	11	23	12	16	1	2	3	4	5	6	7	0
6	18	13	19	14	20	8	21	9	22	15	23	31	16	11	17	2	3	4	5	6	7	0	1
7	19	12	20	13	21	14	22	8	23	9	16	10	17	15	18	3	4	5	6	7	0	1	2
	$(Z_{8,u',8,24} \oslash Y_{8,u-8,23,7,16})$															$Z_{8,u',8,8}$							
8	28	23	29	7	30	17	31	18	24	19	25	20	26	21	27	12	13	14	15	8	9	10	11
9	29	22	30	16	31	23	24	7	25	18	26	19	27	20	28	13	14	15	8	9	10	11	12
10	30	21	31	22	24	16	25	17	26	23	27	7	28	19	29	14	15	8	9	10	11	12	13
11	31	20	24	21	25	22	26	16	27	17	28	18	29	23	30	15	8	9	10	11	12	13	14
	$(Z_{8,u-8,8,16} \oslash Y_{8,u'',31,15,24})$															W_{u-8}							
12	20	31	21	15	22	25	23	26	16	27	17	28	18	29	19	4	5	6	7	0	1	2	3
13	21	30	22	24	23	31	16	15	17	26	18	27	19	28	20	5	6	7	0	1	2	3	4
14	22	29	23	30	16	24	17	25	18	31	19	15	20	27	21	6	7	0	1	2	3	4	5
15	23	28	16	29	17	30	18	24	19	25	20	26	21	31	22	7	0	1	2	3	4	5	6

	Rodadas														
Árbitro	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
	$(Z_{8,u,8,24} \oslash Z_{7,u,7,16})$														
0	24	16	25	17	26	18	27	19	28	20	29	21	30	22	31
1	25	17	26	18	27	19	28	20	29	21	30	22	31	16	24
2	26	18	27	19	28	20	29	21	30	22	31	16	24	17	25
3	27	19	28	20	29	21	30	22	31	16	24	17	25	18	26
	$(Z_{8,u',8,16} \oslash Z_{7,u',7,24})$														
4	16	24	17	25	18	26	19	27	20	28	21	29	22	30	23
5	17	25	18	26	19	27	20	28	21	29	22	30	23	24	16
6	18	26	19	27	20	28	21	29	22	30	23	24	16	25	17
7	19	27	20	28	21	29	22	30	23	24	16	25	17	26	18
	$(Z_{8,15-u,8,16} \oslash Z_{7,u-8,7,0})$														
8	23	0	16	1	17	2	18	3	19	4	20	5	21	6	22
9	22	1	23	2	16	3	17	4	18	5	19	6	20	0	21
10	21	2	22	3	23	4	16	5	17	6	18	0	19	1	20
11	20	3	21	4	22	5	23	6	16	0	17	1	18	2	19
	$(Z_{8,7-u'',8,24} \oslash Z_{7,u'',7,8})$														
12	31	8	24	9	25	10	26	11	27	12	28	13	29	14	30
13	30	9	31	10	24	11	25	12	26	13	27	14	28	8	29
14	29	10	30	11	31	12	24	13	25	14	26	8	27	9	28
15	28	11	29	12	30	13	31	14	24	8	25	9	26	10	27

	Rodadas																						
Árbitro	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
	W_u								$(Z_{8,u+4,8,8} \oslash Z_{8,u,8,16})$															
0	0	1	2	3	4	5	6	7	12	16	13	17	14	18	15	19	8	20	9	21	10	22	11	23
1	1	2	3	4	5	6	7	0	13	17	14	18	15	19	8	20	9	21	10	22	11	23	12	16
2	2	3	4	5	6	7	0	1	14	18	15	19	8	20	9	21	10	22	11	23	12	16	13	17
3	3	4	5	6	7	0	1	2	15	19	8	20	9	21	10	22	11	23	12	16	13	17	14	18
	$Z_{8,u',8,8}$								$(Z_{8,u',8,24} \oslash Z_{8,u,8,0})$															
4	8	9	10	11	12	13	14	15	24	4	25	5	26	6	27	7	28	0	29	1	30	2	31	3
5	9	10	11	12	13	14	15	8	25	5	26	6	27	7	28	0	29	1	30	2	31	3	24	4
6	10	11	12	13	14	15	8	9	26	6	27	7	28	0	29	1	30	2	31	3	24	4	25	5
7	11	12	13	14	15	8	9	10	27	7	28	0	29	1	30	2	31	3	24	4	25	5	26	6
	$Z_{8,u',8,8}$								$(Z_{8,u',8,24} \oslash Z_{8,u-8,8,0})$															
8	12	13	14	15	8	9	10	11	28	0	29	1	30	2	31	3	24	4	25	5	26	6	27	7
9	13	14	15	8	9	10	11	12	29	1	30	2	31	3	24	4	25	5	26	6	27	7	28	0
10	14	15	8	9	10	11	12	13	30	2	31	3	24	4	25	5	26	6	27	7	28	0	29	1
11	15	8	9	10	11	12	13	14	31	3	24	4	25	5	26	6	27	7	28	0	29	1	30	2
	W_{u-8}								$(Z_{8,u'',8,8} \oslash Z_{8,u-8,8,16})$															
12	4	5	6	7	0	1	2	3	8	20	9	21	10	22	11	23	12	16	13	17	14	18	15	19
13	5	6	7	0	1	2	3	4	9	21	10	22	11	23	12	16	13	17	14	18	15	19	8	20
14	6	7	0	1	2	3	4	5	10	22	11	23	12	16	13	17	14	18	15	19	8	20	9	21
15	7	0	1	2	3	4	5	6	11	23	12	16	13	17	14	18	15	19	8	20	9	21	10	22

Figura 3.9: Solução S para $k = 8$ e $C = 0, 1, 2, 3, 4, 5, 6, 7, 0$. Distância total viajada igual a 864.

Rodadas	Subtorneio em T	Árbitros	Subsequências em S
$0, 2, \ldots, 2k-2$	$\overline{P}_{2k,0}[0, k-1]$	$0 \ldots k/2-1$	$Z_{k,u,k,3k}$
		$k/2 \ldots k-1$	$Z_{k,u',k,2k}$
		$k \ldots 3k/2-1$	$Z_{k,u',k,3k}$
		$3k/2 \ldots 2k-1$	$Z_{k,u-k,k,2k}$
$1, 3, \ldots, 2k-3$	$\overline{U}_{k,0}$	$0 \ldots k/2-1$	$Y_{k,u,k-1,3k-1,0}$
	$\overline{U}_{k,k}$	$k/2 \ldots k-1$	$Y_{k,u',2k-1,4k-1,k}$
	$\overline{U}_{k,2k}$	$k \ldots 3k/2-1$	$Y_{k,u-k,3k-1,k-1,2k}$
	$\overline{U}_{k,3k}$	$3k/2 \ldots 2k-1$	$Y_{k,u'',4k-1,2k-1,3k}$
$2k-1, 2k, \ldots, 3k-2$	$P_{2k,0}[0, k-1]$	$0 \ldots k/2-1$	$Z_{k,u,k,k}$
		$k/2 \ldots k-1$	$W_{u'}$
		$k \ldots 3k/2-1$	$Z_{k,u',k,k}$
		$3k/2 \ldots 2k-1$	W_{u-k}
$3k-1, 3k+1, \ldots, 5k-3$	$\overline{P}_{2k,0}[k, 2k-1]$	$0 \ldots k/2-1$	$Z_{k,u,k,3k}$
		$k/2 \ldots k-1$	$Z_{k,u',k,2k}$
		$k \ldots 3k/2-1$	$Z_{k,2k-u-1,k,2k}$
		$3k/2 \ldots 2k-1$	$Z_{k,k-u''-1,k,3k}$
$3k, 3k+2, \ldots, 5k-4$	$U_{k,0}$	$k \ldots 3k/2-1$	$Z_{k-1,u-k,k-1,0}$
	$U_{k,k}$	$3k/2 \ldots 2k-1$	$Z_{k-1,u'',k-1,k}$
	$U_{k,2k}$	$0 \ldots k/2-1$	$Z_{k-1,u,k-1,2k}$
	$U_{k,3k}$	$k/2 \ldots k-1$	$Z_{k-1,u',k-1,3k}$
$5k-2, 5k-1, \ldots, 6k-3$	$P_{2k,0}[k, 2k-1]$	$0 \ldots k/2-1$	W_u
		$k/2 \ldots k-1$	$Z_{k,u',k,k}$
		$k \ldots 3k/2-1$	$Z_{k,u',k,k}$
		$3k/2 \ldots 2k-1$	W_{u-k}
$6k-2, 6k, \ldots, 8k-4$	$\overline{P}_{k,0}$	$0 \ldots k/2-1$	$Z_{k,u+k/2,k,k}$
		$3k/2 \ldots 2k-1$	$Z_{k,u'',k,k}$
	$\overline{P}_{k,2k}$	$k/2 \ldots k-1$	$Z_{k,u',k,3k}$
		$k \ldots 3k/2-1$	$Z_{k,u',k,3k}$
$6k-1, 6k+1, \ldots, 8k-3$	$P_{k,0}$	$k/2 \ldots k-1$	$Z_{k,u,k,0}$
		$k \ldots 3k/2-1$	$Z_{k,u-k,k,0}$
	$P_{k,2k}$	$0 \ldots k/2-1$	$Z_{k,u,k,2k}$
		$3k/2 \ldots 2k-1$	$Z_{k,u-k,k,2k}$

Tabela 3.1: Atribuição das sedes em T aos árbitros em S.

$Z_{k-1,b,k-1,d}$ são $d+(i \bmod (k-1)), \ldots, d+((i+k/2-1) \bmod (k-1))$, para $0 \leq b \leq k/2-1$. Portanto, durante as rodadas $3k, 3k+2, \ldots, 5k-4$, as sedes em $U_{k,0}$, $U_{k,k}$, $U_{k,2k}$ e $U_{k,3k}$ são corretamente atribuídas por $Z_{k-1,u-k,k-1,0}$, $Z_{k-1,u'',k-1,k}$, $Z_{k-1,u,k-1,2k}$ e $Z_{k-1,u',k-1,3k}$, respectivamente. Esta últimas sequências mencionadas são empregadas em (3.28), (3.29), (3.26) e (3.27), respectivamente.

Para completar esta parte da prova, nos resta verificar as atribuições das sedes nas rodadas definidas por P e $\overline{P}$. As sedes em qualquer rodada em $P_{a,b}$ e $\overline{P}_{a,b}$ são, respectivamente, $b, \ldots, b+a-1$ e $b+a, \ldots, b+2a-1$. As $(i+1)$-ésimas sedes em W_a, para $0 \leq a, i \leq k-1$, são $0, \ldots, k-1$, ao passo que as $(i+1)$-ésimas sedes em $Z_{k,b,k,d}$, para $0 \leq b \leq k-1$, são $d+(i \bmod k), \ldots, d+((i+k-1) \bmod k)$, o que pode ser simplificado para $d, \ldots, d+k-1$. Estas observações nos permitem concluir que a atribuição das sedes nas rodadas citadas acima são corretamente feitas pelas sequências W e Z. Por exemplo, durante as rodadas $0, 2, \ldots, 2k-2$, cujos jogos são definidos por $\overline{P}_{2k,0}[0, k-1]$, as sedes

$2k, \ldots, 3k-1$ são atribuídas por $Z_{k,u',k,2k}$ em (3.27) e por $Z_{k,u-k,k,2k}$ em (3.29), ao passo que as sedes $3k, \ldots, 4k-1$ são atribuídas por $Z_{k,u,k,3k}$ em (3.26) e por $Z_{k,u',k,3k}$ em (3.28). As atribuições nas rodadas remanescentes podem ser verificadas de forma similar, o que completa o argumento que S satisfaz as restrições (2.1) e (2.2).

As sequências $Z_{k,u,k,3k}$, $Z_{k,u,k,k}$, W_u e $Z_{k,u,k,2k}$ em (3.26), $Z_{k,u',k,2k}$, $W_{u'}$, $Z_{k,u',k,k}$ e $Z_{k,u',k,3k}$ em (3.27), $Z_{k,u',k,3k}$, $Z_{k,u',k,k}$, $Z_{k,2k-u-1,k,2k}$ e $Z_{k,u-k,k,0}$ em (3.28), $Z_{k,u-k,k,2k}$, W_{u-k}, $Z_{k,k-u''-1,k,3k}$ e $Z_{k,u'',k,k}$ em (3.29) garantem que cada árbitro visita a sede de cada time em T ao menos uma vez. Portanto, S também satisfaz a restrição (2.3).

A restrição (2.4) é satisfeita se todas as sedes visitadas por um dado árbitro durante quaisquer $n - d_1$ rodadas consecutivas são diferentes. Vamos mostrar que S satisfaz (2.4) para $d_1 = 0$ pois isto implica que esta restrição será satisfeita para qualquer $0 \leq d_1 \leq n/2$. Em termos de k, é equivalente dizer que nenhum árbitro visita mais de uma vez uma mesma sede durante $2k$ rodadas consecutivas.

As sequências W, Y (veja o lema 3.4) e Z em (3.26)–(3.29) consistem individualmente de sedes diferentes e, portanto, respeitam (2.4). Se duas sequências em (3.26)–(3.29) são separadas por $2k-1$ ou mais rodadas, nenhuma sede que aparece em ambas sequências pode estar num intervalo de $2k$ rodadas consecutivas. Desta forma, é evidente que elas satisfazem (2.4). Quando duas sequências estão separadas por menos que $2k-1$ rodadas mas consistem de sedes que são todas diferentes entre si, elas também satisfazem (2.4). Por isso, a seguir vamos nos concentrar nas sequências em (3.26)–(3.29) que são separadas por menos que $2k-1$ rodadas e têm sedes em comum.

Durante as rodadas $0, 2, \ldots, 2k-2$, os árbitros $0, \ldots, k/2-1$ viajam passando pelas sedes de $Z_{k,u,k,3k}$ em (3.26), e os árbitros $k/2, \ldots, k-1$ viajam percorrendo as sedes de $Z_{k,u',k,2k}$ em (3.27). Cada um destes árbitros revisitam a mesma sequência de sedes durante as rodadas $3k-1, 3k+1, \ldots, 5k-3$. Isto significa que a sede visitada durante as rodadas $0, 2, \ldots, 2k-2$ será novamente visitada pelo mesmo árbitro após $3k-1$ rodadas. De forma análoga, as sedes visitadas por todos os árbitros nas rodadas $3k, 3k+2, \ldots, 5k-4$ (sequências $Z_{k-1,u,k-1,2k}$, $Z_{k-1,u',k-1,3k}$, $Z_{k-1,u-k,k-1,0}$ e $Z_{k-1,u'',k-1,k}$ em (3.26)–(3.29)) são revisitadas pelos árbitros $0, \ldots, k/2-1$ e $k, \ldots, 3k/2-1$ nas rodadas $6k-1, 6k+1, \ldots, 8k-3$ (sequências $Z_{k,u,k,2k}$ e $Z_{k,u-k,k,0}$ em (3.26) e (3.28)), e pelos árbitros $k/2, \ldots, k-1$ e $3k/2, \ldots, 2k-1$ nas rodadas $6k-2, 6k, \ldots, 8k-4$ (sequências $Z_{k,u',k,3k}$ e $Z_{k,u'',k,k}$ em (3.27) e (3.29)). Pelo lema 3.3, a i-ésima sede nas sequências visitadas por um árbitro nas rodadas $3k, 3k+2, \ldots, 5k-4$ ocorre exatamente uma vez na sequência revisitada pelo mesmo árbitro nas rodadas $6k-2, 6k, \ldots, 8k-4$ ou $6k-1, 6k+1, \ldots, 8k-3$, sendo a i-ésima ou $(i+1)$-ésima sede destas duas últimas sequências. Portanto, as sedes visitadas nas rodadas $3k, 3k+2, \ldots, 5k-4$ são novamente visitadas pelo mesmo árbitro depois de no mínimo $3k-2$ rodadas.

Vamos focar agora nas sequências $Y_{k,u-k,3k-1,k-1,2k}$ e $Z_{k,2k-u-1,k,2k}$ em (3.28), as quais determinam as viagens dos árbitros $k, \ldots, 3k/2-1$ nas rodadas $1, 3, \ldots, 2k-3$ e $3k-$

$1, 3k+1, \ldots, 5k-3$, respectivamente. Aplicando (3.23)–(3.25) em $Y_{k,u-k,3k-1,k-1,2k}$ temos

$$Y_{k,u-k,3k-1,k-1,2k} = (3k-1, k-1) \oplus Z_{k-1,u,3k-2u-3,2k}, \quad \text{se} \quad u = k, \tag{3.30}$$

$$Y_{k,u-k,3k-1,k-1,2k} = Z_{k-1,2k-u-1,2u-2k,2k} \oplus (3k-1, k-1) \oplus Z_{k-1,u,3k-2u-3,2k}, \quad \text{se} \quad k+1 \leq u \leq 3k/2-2, \tag{3.31}$$

$$Y_{k,u-k,3k-1,k-1,2k} = Z_{k-1,2k-u-1,2u-2k,2k} \oplus (3k-1), \quad \text{se} \quad u = 3k/2-1. \tag{3.32}$$

Primeiramente, note que a sede $k-1$ em $Y_{k,u-k,3k-1,k-1,2k}$ não aparece em $Z_{k,2k-u-1,k,2k}$. A sede $3k-1$ ocorre na $(2(u-k)+1)$-ésima posição na sequência $Y_{k,u-k,3k-1,k-1,2k}$ atribuída ao árbitro u. Desta forma, u visita esta sede na rodada $4(u-k)+1$. Com base na definição de $Z_{k,2k-u-1,k,2k}$, considere a equação $2k+((2k-u-1+i) \bmod k) = 3k-1$ para $0 \leq i \leq k-1$. Uma vez que $0 \leq 2k-u-1 \leq k-1$ quando $k \leq u \leq 3k/2-1$, esta equação simplifica para $(2k-u-1+i) = k-1$. Logo, sabemos que a sede $3k-1$ ocupa a $((u-k)+1)$-ésima posição na sequência $Z_{k,2k-u-1,k,2k}$ atribuída ao árbitro u, o que implica que u visita $3k-1$ na rodada $3k-1+2(u-k) = 2u+k-1$. Portanto, a sede $3k-1$ é novamente visitada pelo mesmo árbitro u depois de $2u+k-1-(4(u-k)+1) = 5k-2u-2 \geq 2k$ rodadas para $4 \leq k \leq u \leq 3k/2-1$.

Agora, note que $Z_{k,2k-u-1,k,2k} = Z_{k,2k-u-1,2u-2k+1,2k} \oplus Z_{k,u,3k-2u-2,2k} \oplus Z_{k,3k-u-2,1,2k}$. Os árbitros $k \leq u \leq 3k/2-2$ visitam as sedes de $Z_{k-1,u,3k-2u-3,2k}$ em (3.30) e (3.31) durante as rodadas $4(u-k)+5, 4(u-k)+7, \ldots, 2k-3$ e revisitam estas sedes em $Z_{k,u,3k-2u-2,2k}$ (subsequência de $Z_{k,2k-u-1,k,2k}$) durante as rodadas $3k-1+2(2u-2k+1) = 4u-k+1, 4u-k+3, \ldots, 5k-5$. Os árbitros $k+1 \leq u \leq 3k/2-1$ visitam as sedes de $Z_{k-1,2k-u-1,2u-2k,2k}$ em (3.31) e (3.32) durante as rodadas $1, 3, \ldots, 4(u-k)-1$, e revisitam estas sedes em $Z_{k,2k-u-1,2u-2k+1,2k}$ (subsequência de $Z_{k,2k-u-1,k,2k}$) durante as rodadas $3k-1, 3k+1, \ldots, 4u-k-1$. Aplicando o lema 3.3 nos dois casos acima e considerando as sequências $Z_{k-1,u,3k-2u-3,2k}$ e $Z_{k,u,3k-2u-2,2k}$, bem como $Z_{k-1,2k-u-1,2u-2k,2k}$ e $Z_{k,2k-u-1,2u-2k+1,2k}$, concluímos que estes árbitros retornam novamente nestas sedes nas rodadas $1, 3, \ldots, 2k-3$, depois de pelo menos $3k-4$ rodadas.

O raciocínio apresentado no parágrafo anterior pode ser aplicado também para as sequências $Y_{k,u'',4k-1,2k-1,3k}$ e $Z_{k,k-u''-1,k,3k}$ em (3.29), porque elas são equivalentes às sequências $Y_{k,u-k,3k-1,k-1,2k}$ e $Z_{k,2k-u-1,k,2k}$, respectivamente, com os índices incrementados em k. Portanto, concluímos que todas as sequências em (3.26)–(3.29) que podem ter sedes em comum e são separadas por menos de $2k-1$ rodadas satisfazem (2.4), o que implica que S é factível.

Todas as sedes nas sequências W possuem índices no intervalo $[0, k-1]$, e aparecem na mesma ordem seguida pelo ciclo hamiltoniano C. Portanto, qualquer par de sedes consecutivas em uma sequência W corresponde a dois vértices adjacentes em G. Como consequência, os árbitros que percorrem estas sequências efetuam exatamente $k-1$ viagens nulas. Em S, os árbitros $k/2, \ldots, k-1$ e $3k/2, \ldots, 2k-1$ percorrem as sequências $W_{u'}$ e W_{u-k}, respectivamente, nas rodadas $2k-1, \ldots, 3k-2$, ao passo que os árbitros $0, \ldots, k/2-1$ e $3k/2, \ldots, 2k-1$ percorrem as sequências W_u e W_{u-k}, respectivamente, nas rodadas

$5k-2, \ldots, 6k-3$. Logo, estes árbitros efetuam exatamente $2k(k-1)$ viagens nulas. Pelo fato de todas as demais viagens serem necessariamente viagens unitárias, a distância total percorrida pelos $2k$ árbitros em S é igual a $n(4n-3) - 2k(k-1)$. □

Encontrar uma solução factível com uma estrutura bem definida se torna uma tarefa muito difícil quando a restrição (2.5) é imposta com $d_2 < \lfloor n/2 \rfloor - 1$. Por isso, na prova de $\mathcal{NP}$-completude (teorema 3.2, direção ($\Rightarrow$)), nós desativamos a restrição (2.5) ajustando $d_2 = \lfloor n/2 \rfloor - 1$ para que a solução definida por (3.26)–(3.29) seja factível. Isto fica mais evidente ao observar as figuras 3.5, 3.6 e 3.9. As sequências de sedes que correspondem ao ciclo hamiltoniano C definidas por W em (3.26), (3.27) e (3.29) nem sempre resultam em uma solução que satisfaz (2.5) se $d_2 < \lfloor n/2 \rfloor - 1$.

Corolário 3.1. *A versão de decisão do TUP com $d_1 \leq n/2$ e $d_2 = \lfloor n/2 \rfloor - 1$ é um problema $\mathcal{NP}$-completo.*

Demonstração. A versão de decisão do TUP está claramente em $\mathcal{NP}$. A instância $I(G)$ determinada na definição 3.1 pode ser criada em tempo $O(k^2)$, e os teoremas 3.1 e 3.2 mostram que a redução em tempo polinomial de um problema $\mathcal{NP}$-completo para a versão de decisão do TUP está correta. □

Corolário 3.2. *A versão de decisão do TUP desconsiderando a restrição (2.3) com $d_1 \leq n/2$ e $d_2 = \lfloor n/2 \rfloor - 1$ também é um problema $\mathcal{NP}$-completo.*

Demonstração. Para concluir isso basta observar que esta prova consiste na mesma que vimos neste capítulo, facultando demonstrar na condição necessária (direção ($\Rightarrow$)) do teorema 3.2 que os árbitros em S visitam a sede de cada time ao menos uma vez. □

3.3 Conclusões

Antes deste trabalho, a complexidade do TUP estava em aberto. Nós fornecemos uma prova formal demonstrando que a versão de decisão do TUP é um problema computacional difícil. De certo modo, este resultado não nos surpreende pois, desde que o TUP foi introduzido em [46], vários artigos fizeram menção à crescente dificuldade de resolução do problema à medida que se aumentava o tamanho das instâncias de entrada. Esperamos que este trabalho motive outros pesquisadores a promover futuramente avanços teóricos para este e outros problemas desta área, bem como encoraje o desenvolvimento de novas abordagens computacionais mais sofisticadas capazes de lidar melhor com estes problemas.

Deixamos como sugestão de trabalho futuro, a investigação de uma extensão desta prova para demonstrar que o TUP permanece $\mathcal{NP}$-completo mesmo quando $d_1 = d_2 = 0$. Acreditamos que isso seja verdade, contudo esperamos que tal demonstração exija um grau muito maior de dificuldade para ser efetuada. Também suspeitamos que o problema de decidir quando uma instância do TUP é factível também é $\mathcal{NP}$-completo, como sugerem as experiências práticas. Esta é outra linha de pesquisa que deve ser considerada.

Capítulo 4

Formulação matemática baseada em fluxo em rede

Neste capítulo vamos estudar a primeira formulação matemática de programação linear inteira baseada em fluxo em rede proposta para o TUP na literatura. Com base nela, propusemos outra formulação que resulta em um modelo matemático mais forte e mais compacto (em termos de variáveis e restrições). Empregando este novo modelo, também desenvolvemos uma heurística *relax-and-fix*. Por fim, conduzimos um estudo experimental para avaliar a nova formulação e a heurística *relax-and-fix* propostas. O conteúdo apresentado neste capítulo foi publicado em [33].

4.1 Formulações matemáticas

Primeiramente vamos apresentar a formulação matemática proposta para o TUP que se encontra na literatura, e em seguida descrevemos a nossa formulação.

4.1.1 Uma primeira formulação

A primeira formulação matemática de programação linear inteira para o TUP, baseada em fluxo em rede, foi introduzida no estudo apresentado em [46], posteriormente estendido em [47], e também utilizada em [48] e [49]. Esta formulação recebe os seguintes dados de entrada:

- Conjunto de árbitros $U = \{1, \ldots, n\}$;
- Conjunto de times $T = \{1, \ldots, 2n\}$;
- Conjunto de rodadas $S = \{1, \ldots, 4n-2\}$;
- $\text{OPP}[s,i] = \begin{cases} j, & \text{se } i \text{ joga contra } j \text{ na sede } i \text{ durante a rodada } s, \\ -j, & \text{se } i \text{ joga contra } j \text{ na sede } j \text{ durante a rodada } s; \end{cases}$
- d_{ij} = distância em milhas entre as sedes i e j;
- $CV_s = \{s, \ldots, s+n-d_1-1\}$ para qualquer rodada $s \in \{1, \ldots, 4n-2-(n-d_1-1)\}$;

- $CT_s = \{s, \ldots, s + \lfloor \frac{n}{2} \rfloor - d_2 - 1\}$ para qualquer rodada $s \in \{1, \ldots, 4n - 2 - (\lfloor \frac{n}{2} \rfloor - d_2 - 1)\}$.

As variáveis de decisão são:

- $x_{isu} = \begin{cases} 1, & \text{se o jogo na sede } i \text{ durante a rodada } s \text{ é atribuído ao árbitro } u, \\ 0, & \text{caso contrário;} \end{cases}$
- $z_{ijsu} = \begin{cases} 1, & \text{se o árbitro } u \text{ está na sede } i \text{ durante a rodada } s, \text{ e viaja para a sede } j \\ & \text{na rodada } s+1, \\ 0, & \text{caso contrário.} \end{cases}$

Agora estamos prontos para apresentar a formulação.

$$\text{Minimizar} \sum_{i \in T} \sum_{j \in T} \sum_{u \in U} \sum_{s \in S: s < |S|} d_{ij} z_{ijsu} \tag{4.1}$$

Sujeito a:

$$\sum_{u \in U} x_{isu} = 1, \quad \forall\, i \in T, s \in S : \text{OPP}[s,i] > 0, \tag{4.2}$$

$$\sum_{i \in T: \text{OPP}[s,i] > 0} x_{isu} = 1, \quad \forall\, s \in S, u \in U, \tag{4.3}$$

$$\sum_{s \in S: \text{OPP}[s,i] > 0} x_{isu} \geq 1, \quad \forall\, i \in T, u \in U, \tag{4.4}$$

$$\sum_{c \in CV_s: \text{OPP}[c,i] > 0} x_{icu} \leq 1, \quad \begin{array}{l} \forall\, i \in T, u \in U, s \in S : \\ s \leq |S| - (n - d_1 - 1), \end{array} \tag{4.5}$$

$$\sum_{c \in CT_s} \left(x_{icu} + \sum_{j \in T: \text{OPP}[c,j] = i} x_{jcu} \right) \leq 1, \quad \begin{array}{l} \forall\, i \in T, u \in U, s \in S : \\ s \leq |S| - (\lfloor \frac{n}{2} \rfloor - d_2 - 1), \end{array} \tag{4.6}$$

$$x_{isu} + x_{j(s+1)u} - z_{ijsu} \leq 1, \quad \forall\, i, j \in T, u \in U, s \in S : s < |S|, \tag{4.7}$$

$$x_{isu} \in \{0, 1\}, \quad \forall\, i \in T, u \in U, s \in S, \tag{4.8}$$

$$z_{ijsu} \in \{0, 1\}, \quad \forall\, i, j \in T, u \in U, s \in S : s < |S|. \tag{4.9}$$

A função objetivo (4.1) minimiza a distância total viajada pelos árbitros. As restrições (4.2) e (4.3) determinam que cada jogo deve ser atribuído a um árbitro e que cada árbitro deve apitar um jogo, respectivamente. As restrições (2.3), (2.4) e (2.5) são modeladas por (4.4), (4.5) e (4.6), respectivamente. A consistência entre as atribuições dos árbitros aos jogos e suas viagens (variáveis x e z) é imposta por (4.7). Por fim, as restrições (4.8) e (4.9) definem o domínio das variáveis.

Ainda em [47], a formulação acima foi fortalecida com a inclusão das seguintes restrições:

$$x_{isu} = 0, \quad \forall\, i \in T, u \in U, s \in S : \mathrm{OPP}[s,i] < 0, \tag{4.10}$$

$$z_{ijsu} - x_{isu} \leq 0, \quad \forall\, i,j \in T, u \in U, s \in S : s < |S|, \tag{4.11}$$

$$z_{ijsu} - x_{j(s+1)u} \leq 0, \quad \forall\, i,j \in T, u \in U, s \in S : s < |S|, \tag{4.12}$$

$$\sum_{j \in T} z_{jisu} - \sum_{j \in T} z_{ij(s+1)u} = 0, \quad \forall\, i \in T, u \in U, s \in S : s < |S| - 1, \tag{4.13}$$

$$\sum_{i \in T} \sum_{j \in T} z_{ijsu} = 1, \quad \forall\, u \in U, s \in S : s < |S|. \tag{4.14}$$

A restrição (4.10) proíbe a atribuição de árbitros às sedes onde não são realizados jogos em uma dada rodada. As restrições (4.11) e (4.12) permitem que um árbitro viaje de uma sede i para um sede j, entre as rodadas s e $s+1$, somente se ele estiver atribuído aos jogos destas sedes nestas rodadas. A restrição de conservação de fluxo (4.13) estabelece que se o árbitro u está (não está) na sede i na rodada $s+1$, então a sua viagem entre as rodadas $s+1$ e $s+2$ deve (não deve) partir da sede i. A restrição (4.14) impõe que todo árbitro deve se mover de um jogo para outro entre rodadas consecutivas.

A formulação (4.1)–(4.14) sofre de um problema de simetria onde, dada uma solução, é possível obter $n!$ soluções equivalentes entre si permutando os árbitros. Para solucionar isso, os autores em [58] fixaram previamente os jogos que são apitados por cada árbitro em uma dada rodada k, que pode ser livremente escolhida no intervalo $[1, 4n-2]$. Desta forma, vamos acrescentar a restrição (4.15) na formulação, onde K é um conjunto de n tuplas que atribui cada árbitro à sede de um jogo diferente na rodada k:

$$x_{iku} = 1, \quad \forall\, (i,u) \in K. \tag{4.15}$$

Vamos nos referir à rodada k como *rodada de quebra de simetria*, e à formulação (4.1)–(4.15) como $\mathcal{F}_1$. A relaxação linear de $\mathcal{F}_1$, onde as restrições de integralidade (4.8)–(4.9) são substituídas por $0 \leq x_{isu} \leq 1$ e $0 \leq z_{ijsu} \leq 1$, será denotada por $\mathcal{F}_1^R$.

4.1.2 Uma formulação mais forte

Com base em $\mathcal{F}_1$, vamos propor uma formulação mais forte para o TUP que inclui as seguintes identidades válidas:

$$x_{i1u} = \sum_{j \in T} z_{ij1u}, \quad \forall\, i \in T, u \in U, \tag{4.16}$$

$$x_{isu} = \sum_{j \in T} z_{ji(s-1)u}, \quad \forall\, i \in T, u \in U, s \in S : s > 1. \tag{4.17}$$

A restrição (4.16) garante que o árbitro u seja atribuído à sede i na primeira rodada se, e somente se, u viaja partindo de i nesta rodada. De forma análoga, (4.17) determina que o árbitro u seja atribuído à sede i na rodada $s > 1$ se, e somente se, u viaja para i entre as rodadas $s-1$ e s.

Como (4.16) e (4.17) são igualdades, a variável x pode ser eliminada da formulação substituindo a sua ocorrência pelo respectivo somatório de z contido nestas expressões. Denotaremos por $\mathcal{F}_2$ a formulação obtida a partir de $\mathcal{F}_1$, usando (4.16) e (4.17) para eliminar as variáveis x e removendo as restrições (4.3), (4.7), (4.11), (4.12) e (4.14), e por $\mathcal{F}_2^R$ a relaxação linear de $\mathcal{F}_2$.

Proposição 4.1. *$\mathcal{F}_2$ é uma formulação matemática de programação linear inteira válida para o TUP, e sua relaxação linear sempre fornece limitantes inferiores maiores ou iguais àqueles providos pela relaxação linear de $\mathcal{F}_1$.*

Demonstração. É suficiente mostrar que (4.3), (4.7), (4.11), (4.12) e (4.14) são satisfeitas, com $0 \leq x_{isu} \leq 1$ e $0 \leq z_{ijsu} \leq 1$, pela conjunção de (4.16) e (4.17) com as restrições remanescentes de $\mathcal{F}_1$. Para simplificar, e sem perder generalidade, vamos mostrar isso antes de eliminar a variável x do modelo.

Primeiramente, ao combinar (4.13), (4.16) e (4.17) vamos obter a seguinte identidade:

$$x_{isu} = \sum_{j \in T} z_{ijsu}, \quad \forall\, i \in T, u \in U, s \in S : s < |S|. \tag{4.18}$$

Para ver que (4.18) implica em (4.11), note que $\sum_{j \in T} z_{ijsu} \geq z_{ihsu}$ para qualquer $h \in T$. A prova que (4.17) implica em (4.12) é análoga.

Vamos mostrar agora que (4.3) e (4.14) são satisfeitas. Aplicando (4.13), podemos escrever

$$\sum_{i \in T}\sum_{j \in T} z_{ij1u} = \cdots = \sum_{i \in T}\sum_{j \in T} z_{ijku} = \cdots = \sum_{i \in T}\sum_{j \in T} z_{ij(|S|-1)u}, \ \ \forall\, u \in U. \tag{4.19}$$

Ao substituir os somatórios internos em (4.19) pelas variáveis x apropriadas de acordo com (4.18), vamos obter igualdades equivalentes a (4.20), exceto pela última:

$$\sum_{i \in T} x_{i1u} = \cdots = \sum_{i \in T} x_{iku} = \cdots = \sum_{i \in T} x_{i(|S|-1)u} = \sum_{i \in T} x_{i(|S|)u}, \ \ \forall\, u \in U. \tag{4.20}$$

Para obter a última igualdade em (4.20) substituímos o somatório interno mais à direita em (4.19) pela variável x apropriada de acordo com (4.17).

Como temos $\sum_{i \in T} x_{iku} = 1$ em (4.15), então todas as expressões em (4.19) e (4.20) são iguais a 1. Logo, (4.3) e (4.14) são satisfeitas visto que os seus somatórios são iguais àqueles em (4.20) e (4.19), respectivamente.

Por fim, vamos verificar a satisfação da restrição (4.7). Para qualquer $u \in U, s \in S, s < |S|$, vimos que combinar (4.15), (4.19) e (4.20) resulta em

$$\sum_{i \in T}\sum_{j \in T} z_{ijsu} = 1.$$

Portanto, para um dado par de sedes i, j no somatório duplo acima, temos

$$\sum_{h \in T : h \neq j} z_{ihsu} + \sum_{h \in T : h \neq i} z_{hjsu} + z_{ijsu} \leq 1,$$

que equivale a

$$\sum_{h \in T} z_{ihsu} + \sum_{h \in T} z_{hjsu} - z_{ijsu} \leq 1. \tag{4.21}$$

Por causa de (4.18), podemos substituir x_{isu} por $\sum_{h \in T} z_{ihsu}$ em (4.21). De forma análoga, (4.17) nos permite substituir $x_{j(s+1)u}$ por $\sum_{h \in T} z_{hjsu}$ em (4.21), resultando em (4.7). □

Agora estamos prontos para mostrar que $\mathcal{F}_2$ é mais forte que $\mathcal{F}_1$.

Proposição 4.2. *O limitante inferior provido por $\mathcal{F}_2^R$ pode ser estritamente maior que aquele provido por $\mathcal{F}_1^R$.*

Demonstração. Vamos mostrar um exemplo onde o limitante inferior de $\mathcal{F}_2^R$ é estritamente maior que o de $\mathcal{F}_1^R$. Considere a instância do TUP com quatro times apresentada na figura 4.1, cujo torneio é representado por uma tabela de jogos (pares ordenados de times) por rodada. Relembre que o primeiro time de cada par ordenado sedia a partida.

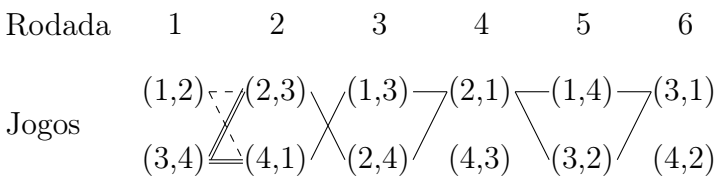


Figura 4.1: Uma instância do TUP com linhas tracejadas, simples e duplas que representam as viagens dos árbitros em uma solução fracionária ótima para $\mathcal{F}_1^R$.

Considere que as sedes dos times 1, 2 e 3 sejam muito próximas umas das outras (digamos, 1 milha distante), e a sede do time 4 fique mais longe dos demais times (digamos, 10 milhas distante). Neste cenário, uma solução ótima para $\mathcal{F}_1^R$ com $d_1 = d_2 = 0$, $k = 1$ e $K = \{(1,1),(3,2)\}$ corresponde a atribuir valor 1 às variáveis x_{111} e x_{312}, valor 0 às variáveis x_{311} e x_{112}, valor 0,5 para todas as demais variáveis x_{isu} com $i \in T, s \in S, u \in U$ e $OPP[i,s] > 0$, valor 0,5 às variáveis z associadas às viagens do árbitro 1 determinadas pelas linhas sólidas simples e as tracejadas na figura 4.1, e valor 0,5 às variáveis z associadas às viagens do árbitro 2 determinadas pelas linhas sólidas simples e duplas na figura 4.1. Todas as demais variáveis são iguais a zero. O valor desta solução fracionária ótima é 25. Este é um limitante inferior muito fraco uma vez que o valor ótimo do problema inteiro para esta instância é 55.

Observe agora que (4.17) não é satisfeita pela solução anterior, pois as variáveis x_{441}, x_{442}, x_{461} e x_{462} possuem valor 0,5 e todas as variáveis z associadas às viagens que chegam ou partem da sede do time 4 nas rodadas 4 e 6 possuem valor 0. Logo, esta solução fracionária se torna infactível quando (4.17) é incluída. Ao resolver $\mathcal{F}_1^R$ adicionando (4.16) e (4.17), a nova solução fracionária ótima encontrada possui valor 52, que é um valor muito mais próximo do ótimo do problema inteiro. □

É possível fortalecer $\mathcal{F}_2^R$ ainda mais fixando as seguintes variáveis:

$$z_{iisu} = 0, \quad \forall\, i \in T, u \in U, s \in S : s < |S|, \text{ se } d_1 < n-1, \tag{4.22}$$

$$z_{ijsu} = 0, \quad \begin{array}{l} \forall\, i \neq j \in T, u \in U, s \in S : s < |S|, \text{ se } d_2 < \lfloor \frac{n}{2} \rfloor - 1 \text{ e} \\ (\ \mathrm{OPP}[s,i] = \mathrm{OPP}[s+1,j] \text{ ou } \mathrm{OPP}[s,i] = j \text{ ou} \\ \mathrm{OPP}[s+1,j] = i\ ). \end{array} \tag{4.23}$$

Note que (4.22) e (4.23) são válidas porque elas proíbem a atribuição de valores positivos às variáveis z que violariam as restrições (2.4) e (2.5) (nenhum árbitro pode permanecer em uma sede ou seguir um time em rodadas consecutivas) quando d_1 e d_2 são estritamente menores que os seus valores máximos permitidos. Denotaremos por $\mathcal{F}_{2+}$ e $\mathcal{F}_{2+}^R$, respectivamente, as formulações obtidas a partir de $\mathcal{F}_2$ e $\mathcal{F}_2^R$ após a inclusão de (4.22) e (4.23). No exemplo da figura 4.1, a solução ótima apresentada para $\mathcal{F}_1^R$ (veja a prova da proposição 4.2) viola (4.22) várias vezes. Por exemplo, nela temos $z_{2221} = z_{2222} = z_{2231} = z_{2232} = z_{3351} = z_{3352} = 0{,}5$. Resolvendo $\mathcal{F}_{2+}^R$ para o exemplo desta figura resultamos em uma solução inteira (ótima) com valor 55. Relembre que os limitantes inferiores produzidos por $\mathcal{F}_1^R$ e $\mathcal{F}_2^R$ para este exemplo foram 25 e 52, respectivamente.

4.2 Heurística *relax-and-fix*

Uma heurística *relax-and-fix* é um método que resolve iterativamente uma relaxação de um modelo de programação linear inteira e fixa progressivamente as variáveis até que uma solução factível seja encontrada [53, seção 12.5]. Desenvolvemos uma heurística *relax-and-fix* baseada em $\mathcal{F}_{2+}$. Ela recebe como entrada uma instância do TUP e um inteiro $1 \leq b \leq 4n-3$. O parâmetro b define o tamanho da janela de rodadas consecutivas cujas variáveis terão domínio binário no modelo relaxado resolvido em cada iteração do algoritmo. Desta forma, o método começa com a formulação $\mathcal{F}_{2+}$, sendo a rodada de quebra de simetria $k = 1$, e relaxa esta formulação para que somente as variáveis associadas às b primeiras rodadas sejam binárias e as demais sejam contínuas. Este modelo é resolvido e, se for infactível, o algoritmo termina sem encontrar uma solução. Caso contrário, as variáveis da primeira rodada são fixadas com os valores da melhor solução encontrada até o método atingir algum critério de parada. Na próxima iteração, as variáveis da rodada $b+1$ também são ajustadas como binárias e o modelo resultante é resolvido. Novamente, o método encerra sem solução em caso de infactibilidade ou, caso contrário, fixa as variáveis da segunda rodada com os valores da melhor solução encontrada. Estes passos são repetidos até que todas as variáveis sejam fixadas ou o modelo se torne infactível. O algoritmo 1 apresenta detalhes mais específicos desta heurística, que serão discutidos a seguir.

Algoritmo 1 Pseudocódigo da heurística *relax-and-fix*.

```
1: procedimento RELAX-AND-FIX(instância do TUP I, inteiro b)
2:     M ← modelo F_{2+} com k = 1 para a instância I;
3:     t ← 1;
4:     enquanto t ≤ 4n − 3 faça
5:         Para todo i, j ∈ T, u ∈ U, s ∈ S, t ≤ s < t + b, faça z_{ijsu} ∈ {0, 1} em M;
6:         Para todo i, j ∈ T, u ∈ U, s ∈ S, s ≥ t + b, faça 0 ≤ z_{ijsu} ≤ 1 em M;
7:         Resolva M;
8:         se uma solução z̄_{ijsu} foi encontrada dentro do limite permitido de nós explorados então
9:             Fixe em M as variáveis z_{ijsu} = z̄_{ijsu} para todo i, j ∈ T, u ∈ U, s = t;
10:            t ← t + 1;
11:        senão
12:            devolve aviso solução não encontrada;
13:        fim se
14:    fim enquanto
15:    devolve a solução encontrada;
16: fim procedimento
```

Durante a iteração t, a heurística resolve uma relaxação de $\mathcal{F}_{2+}$ cujo domínio das variáveis z das rodadas $s \leq t-1$, $t \leq s < t+b$ e $s \geq t+b$ é fixo, binário e contínuo, respectivamente. Se este modelo for factível, as variáveis da rodada t são fixadas com os valores da melhor solução encontrada após explorar uma quantidade limitada de nós (veja a seção 4.3.3 para mais detalhes). Como as soluções intermediárias encontradas na linha 8 do algoritmo não são necessariamente ótimas para os últimos b modelos resolvidos, a heurística continua a executar até atingir a iteração $t = 4n-3$, quando todas as variáveis no modelo são fixadas.

A fim de manter os tempos de execução sob controle, usamos duas estratégias diferentes para resolver as relaxações de $\mathcal{F}_{2+}$ em cada iteração da heurística *relax-and-fix*. Para as instâncias com no máximo 18 times, resolvemos os modelos com todas as variáveis e restrições, da forma como foi descrito acima. No entanto, para as instâncias com mais de 18 times, em cada iteração t da heurística resolvemos a relaxação de $\mathcal{F}_{2+}$ que inclui somente as variáveis e restrições associadas às rodadas menores ou iguais a $\min(\lfloor \frac{2}{5}(4n-2) \rfloor + t - 1, 4n-2)$. (Observe que, na iteração t, todas as variáveis associadas às $t-1$ primeiras rodadas estão fixadas.) Isso é equivalente a redefinir o conjunto S em cada iteração para ser $S = \{1, \ldots, \min(\lfloor \frac{2}{5}(4n-2) \rfloor + t - 1, 4n-2)\}$. O valor $\lfloor \frac{2}{5}(4n-2) \rfloor$ foi escolhido experimentalmente com o objetivo de acelerar a heurística, mas ainda permitindo que o modelo olhe rodadas suficientes à frente da rodada t e seja capaz de encontrar boas soluções. Para desconsiderar de forma apropriada rodadas futuras durante a resolução das relaxações de $\mathcal{F}_{2+}$, é necessário lidar com a restrição (4.4) de uma forma diferente. Nas iterações iniciais, o número de rodadas consideradas nas relaxações de $\mathcal{F}_{2+}$ é insuficiente para satisfazer esta restrição. Entretanto, se desconsiderarmos (4.4) completamente durante muitas iterações as variáveis fixadas podem tornar impossível a satisfação de (4.4) mais tarde. Portanto, vamos introduzir (4.4) gradativamente da seguinte forma. Desconsideramos completamente esta restrição durante as iterações $t \leq n$. Nas iterações $n+1, n+2, \ldots, 2n-1$ impomos (4.4) somente para os árbitros cujos índices são menores ou iguais a 1, 2, ..., $n-1$, respectivamente, e nas iterações $t > 2n-1$ impomos (4.4) para todos os árbitros.

Note que nossa heurística começa com uma atribuição de árbitros na primeira rodada ($k = 1$) e, a cada iteração t, determina as viagens dos árbitros entre as rodadas t e $t + 1$. Isto é semelhante ao que é realizado na heurística gulosa guiada por cortes de Bender proposta em [47]. A diferença vem do fato que nossa heurística *relax-and-fix* estabelece as atribuições dos árbitros em cada rodada levando em consideração, de uma forma relaxada, as rodadas subsequentes do torneio. Além disso, ela lida diretamente com as distâncias das viagens ao invés de usar uma função de custo modificada para efetuar as atribuições em cada rodada. Por fim, como nosso método não garante encontrar uma solução a cada iteração, poderíamos usar uma estratégia de *backtracking* semelhante àquela adotada na heurística gulosa proposta em [47]. Contudo, na prática, quase sempre são encontradas soluções para as instâncias que sabemos que são factíveis. Portanto, a fim de manter nossa implementação simples, optamos por não efetuar *backtracking*.

4.3 Resultados computacionais

Nesta seção, vamos avaliar experimentalmente a nova formulação e a heurística *relax-and-fix* propostas. Consideramos em nossos experimentos as instâncias do *benchmark* do TUP [44] (uma descrição desse *benchmark* se encontra na página 14). Os experimentos foram conduzidos em uma máquina executando o sistema operacional Linux Ubuntu 12.04.1, equipada com um processador i7-2600 3,40GHz e 8GB de RAM. As implementações foram feitas em C++ e utilizam a biblioteca em C (*Callable Library*) do pacote ILOG CPLEX versão 12.5.0.1 para resolver os modelos de programação linear e programação linear inteira.

É importante ressaltar que durante o desenvolvimento do estudo apresentado neste capítulo, a literatura do TUP possuía apenas os seguintes trabalhos: [46], [47], [48], [49] e [58]. Os melhores limitantes inferiores e superiores reportados no site do *benchmark* do TUP [44] nesta época eram compostos por resultados destas referências e também por resultados preliminares do trabalho apresentado em [51]. Desta forma, as análises realizadas nesta seção comparam os resultados obtidos pelos métodos propostos com os melhores resultados disponíveis até então na literatura.

4.3.1 Resultados da formulação nas instâncias estritas

Antes de resolvermos as formulações inteiras, vamos analisar a qualidade dos limitantes inferiores de suas respectivas relaxações lineares. Primeiramente, vamos focar nas instâncias estritas ($d_1 = d_2 = 0$), e fixar a rodada de quebra de simetria em 1 ($k = 1$). A tabela 4.1 apresenta os limitantes inferiores obtidos ao resolver as relaxações lineares da formulação original ($\mathcal{F}_1^R$) e das formulações mais fortes ($\mathcal{F}_2^R$ e $\mathcal{F}_{2+}^R$) que propusemos. Incluímos também nesta tabela o tempo gasto na otimização das formulações e as distâncias ótimas do problema inteiro para as instâncias em que conhecemos este valor. Note que, exceto pela menor instância (com 4 times), $\mathcal{F}_2^R$ e $\mathcal{F}_{2+}^R$ sempre fornecem limitantes inferiores melhores que $\mathcal{F}_1^R$. Em particular, $\mathcal{F}_{2+}^R$ obtém limitantes inferiores consideravelmente melhores (até 340% maiores), e que são razoavelmente mais próximos dos valores ótimos conhecidos, o que comprova a importância das restrições (4.22) e (4.23). Além disso, $\mathcal{F}_2^R$

		Limitante inferior			Tempo (segundos)		
Instância	Ótimo	$\mathcal{F}_1^R$	$\mathcal{F}_2^R$	$\mathcal{F}_{2+}^R$	$\mathcal{F}_1^R$	$\mathcal{F}_2^R$	$\mathcal{F}_{2+}^R$
4	5176	5176,0	5176,0	5176,0	0,0	0,0	0,0
6	14077	5874,3	7971,0	14077,0	0,0	0,0	0,0
6A	15457	7351,0	9228,0	13672,3	0,0	0,0	0,0
6B	16716	6541,0	9766,0	15786,3	0,0	0,0	0,0
6C	14396	5097,7	6717,3	14396,0	0,0	0,0	0,0
8	34311	11836,7	14787,3	33723,2	0,1	0,0	0,0
8A	31490	11659,4	13792,4	30193,9	0,0	0,0	0,0
8B	32731	11057,2	15408,7	31724,4	0,0	0,0	0,0
8C	29879	10717,6	12532,7	27718,3	0,0	0,0	0,0
10	48942	16781,1	20026,1	48040,1	0,2	0,1	0,1
10A	46551	13261,6	14991,3	44909,8	0,2	0,1	0,1
10B	45609	13832,1	17358,5	44238,7	0,2	0,1	0,1
10C	43149	14922,3	16594,7	39618,2	0,2	0,1	0,1
12		26993,5	39722,3	82753,4	0,6	0,3	0,3
14		43664,2	68706,4	140180,0	1,7	0,8	0,9
14A		42484,1	67680,4	132063,0	1,7	0,9	0,8
14B		41587,2	63116,8	129671,0	1,8	0,8	0,8
14C		45634,8	68155,2	125719,0	2,0	0,9	0,8
16		46133,6	65830,9	131264,0	6,1	2,3	2,0
16A		51454,1	76574,5	145901,0	5,5	2,3	2,0
16B		48635,1	86315,4	143592,0	6,1	2,6	2,0
16C		36402,5	81699,4	144402,0	4,8	2,1	1,9
18		49031,3	88031,7	152548,0	14,2	4,2	4,0
20		47183,0	92467,9	183820,0	40,4	9,3	8,0
22		55629,1	109778,0	209092,0	73,4	21,2	15,2
24		56630,0	119211,0	226090,0	136,9	35,2	28,9
26		59919,4	129532,0	266366,0	302,3	69,3	47,1
28		72861,6	161630,0	302138,0	463,9	100,3	85,9
30		83710,2	166316,0	338267,0	1169,8	170,7	143,6
32		97058,9	186105,0	371968,0	1786,5	322,0	227,8

Tabela 4.1: Limitantes inferiores e tempos de execução das relaxações lineares das diferentes formulações matemáticas para as instâncias estritas ($d_1 = d_2 = 0$) com $k = 1$.

e $\mathcal{F}_{2+}^R$ são resolvidas entre 2 e 8 vezes mais rápido que $\mathcal{F}_1^R$ porque elas são mais compactas (possuem menos variáveis e restrições), sendo que $\mathcal{F}_{2+}^R$ possui uma ligeira vantagem sobre $\mathcal{F}_2^R$ neste aspecto.

Nossos experimentos revelaram que a escolha da rodada de quebra de simetria k pode afetar consideravelmente o valor do limitante inferior provido pelas relaxações lineares. Contudo, é difícil determinar *a priori* qual valor de k leva aos melhores resultados para as formulações inteiras. Portanto, resolvemos as formulações inteiras com três diferentes valores de k: 1 (primeira rodada), $2n - 1$ (rodada do meio), e o valor k^* que produz o melhor limitante inferior pela relaxação linear da respectiva formulação. As formulações ajustadas com cada um destes valores escolhidos para k serão denotadas como antes mas incluindo os sufixos "-F" (do inglês *first*) quando $k = 1$, "-M" (do inglês *middle*) quando $k = 2n - 1$ e "-BB" (do inglês *best bound*) quando $k = k^*$, respectivamente.

Vistos os limitantes das relaxações lineares, agora vamos resolver as formulações inteiras para as instâncias estritas. Limitamos o tempo de execução em 3 horas e permitimos que o método de otimização execute apenas uma *thread*. A tabela 4.2 apresenta as distâncias ótimas encontradas e os tempos de execução para as instâncias estritas com até

		Tempo (segundos)					
Instância	Ótimo	$\mathcal{F}_1$-F	$\mathcal{F}_1$-M	$\mathcal{F}_1$-BB	$\mathcal{F}_{2+}$-F	$\mathcal{F}_{2+}$-M	$\mathcal{F}_{2+}$-BB
4	5176	0,00	0,00	0,00	0,00	0,00	0,00
6	14077	0,02	0,02	0,02	0,02	0,01	0,02
6A	15457	0,05	0,02	0,03	0,02	0,01	0,01
6B	16716	0,03	0,03	0,04	0,02	0,02	0,02
6C	14396	0,02	0,02	0,03	0,01	0,01	0,02
8	34311	0,53	0,06	0,15	0,38	0,03	0,03
8A	31490	0,30	0,06	0,05	0,24	0,03	0,02
8B	32731	0,30	0,06	0,06	0,12	0,02	0,02
8C	29879	0,65	0,06	0,04	0,53	0,03	0,02
10	48942	38,10	1,68	1,64	14,56	1,17	1,87
10A	46551	52,05	4,77	5,48	21,25	1,00	4,27
10B	45609	25,52	2,32	3,59	5,67	0,73	0,73
10C	43149	335,33	9,19	13,54	34,49	2,78	1,92
12	Infactível	3172,34	101,10	857,48	2556,69	71,47	627,42

Tabela 4.2: Resultados ótimos obtidos pelas formulações inteiras nas instâncias estritas com até 12 times ($d_1 = d_2 = 0$).

12 times. Em geral, $\mathcal{F}_{2+}$ consegue um desempenho melhor que $\mathcal{F}_1$. Para uma mesma formulação, ajustar a rodada de quebra de simetria como sendo o meio do torneio frequentemente acelera a otimização, ao passo que escolher a primeira rodada é uma péssima ideia. Por isso, vamos desconsiderar $\mathcal{F}_1$-F e $\mathcal{F}_{2+}$-F nos experimentos subsequentes.

A tabela 4.3 sumariza os resultados para as instâncias estritas com pelo menos 14 times, as quais não se conheciam soluções ótimas até então. Calcular k^* para $\mathcal{F}_1$-BB nas instâncias com até 26 times, e para $\mathcal{F}_{2+}$-BB na instância com 32 times leva mais de 3 horas. Desta forma, nestes casos reportamos os melhores limitantes inferiores que encontramos em 3 horas, marcados com um "*", e não resolvemos a formulação inteira. A tabela 4.3 exibe resultados inesperados para $\mathcal{F}_1$. Embora os limitantes inferiores de $\mathcal{F}_1^R$ vistos na tabela 4.1 sejam ruins, aparentemente as rotinas de *presolve* e fortalecimento do CPLEX são capazes de melhorar esta formulação, permitindo que ela consiga limitantes inferiores competitivos em 3 horas. Por exemplo, os melhores limitantes inferiores para as instâncias com 16 e 18 times são fornecidos por $\mathcal{F}_1$-BB. Por outro lado, esta formulação obtém limitantes inferiores muito fracos para as instâncias com mais de 24 times dentro do limite de tempo. As formulações $\mathcal{F}_{2+}$ fornecem os melhores limitantes inferiores nas instâncias com 14 times, e naquelas com mais de 18 times, sendo que nestas últimas os resultados são consideravelmente melhores que aqueles providos por $\mathcal{F}_1$. No geral, $\mathcal{F}_{2+}$-BB obtém 8 dos 16 melhores limitantes inferiores, e chega bem próximo dos demais melhores resultados, exceto pela instância com 30 times. Os melhores limitantes inferiores da tabela 4.3 são melhores que todos aqueles reportados no site do *benchmark* do TUP naquela época. Cabe ressaltar também que, até aquele momento, ainda não haviam sido computados limitantes inferiores para as instâncias com mais de 16 times.

Em se tratando da qualidade das soluções factíveis, nenhuma formulação encontrou soluções para as instâncias estritas com mais de 14 times [1]. A formulação $\mathcal{F}_{2+}$-M encontra três das quatro melhores soluções para as instâncias com 14 times, sendo todas estas

[1] Atualmente, foi provado que as instâncias estritas com 16 times são infactíveis [42, 43], a instância estrita com 18 times é factível [41], e as demais ainda estão em aberto.

	Melhor limitante inferior em 3 horas				Melhor solução em 3 horas			
Instância	$\mathcal{F}_1$-M	$\mathcal{F}_1$-BB	$\mathcal{F}_{2+}$-M	$\mathcal{F}_{2+}$-BB	$\mathcal{F}_1$-M	$\mathcal{F}_1$-BB	$\mathcal{F}_{2+}$-M	$\mathcal{F}_{2+}$-BB
14	149636	149963	150045	**150871**	179343	177139	175524	**174715**
14A	142394	142759	**143517**	141308	166712	172367	**165968**	171961
14B	141795	141608	141645	**142614**	172828	176709	**168659**	170804
14C	141196	141148	**141268**	138601	168037	168787	**164512**	167898
16	148059	**151748**	147906	150523				
16A	161087	**166626**	160831	166101				
16B	158322	**162251**	159324	161882				
16C	162640	**165431**	162505	164235				
18	171264	**177055**	173815	175321				
20	198706	199460	201769	**204278**				
22	222713	228074	224841	**231809**				
24	246620	248817	248977	**253506**				
26	283971	73900*	286239	**286847**				
28	312655	87273*	317629	**319044**				
30	343013	94713*	**352258**	344831				
32	371143	97059*	377531	**382508***				

Tabela 4.3: Resultados para as formulações inteiras nas instâncias estritas com pelo menos 14 times ($d_1 = d_2 = 0$). Tempo de execução limitado em 3 horas. Os melhores valores aparecem em negrito.

melhores que as soluções encontradas pelos métodos exatos (dentro dos limites de tempo) reportadas em [48].

4.3.2 Resultados da formulação nas instâncias relaxadas

A tabela 4.4 sumariza os resultados das formulações propostas nas instâncias relaxadas ($d_1 + d_2 > 0$) com pelo menos 14 times. Uma vez que calcular k^* para $\mathcal{F}_1$-BB na instância com 30 times toma muito tempo, reportamos o melhor limitante inferior que encontramos em 3 horas marcado com um "*". Exceto pelas duas primeiras instâncias com 14 times, os limitantes inferiores fornecidos pelas formulações $\mathcal{F}_{2+}$ são sempre melhores que aqueles providos pelas formulações $\mathcal{F}_1$. A formulação $\mathcal{F}_{2+}$-BB é responsável por 14 dos 21 melhores limitantes inferiores encontrados. A formulação $\mathcal{F}_{2+}$-M é melhor nas instâncias com 14 times, encontrando 4 dos 8 melhores limitantes inferiores, e também obtém o melhor limitante inferior na instância com 30 times. Em relação às soluções, $\mathcal{F}_{2+}$-M encontra 8 dos 21 melhores resultados.

Os melhores limitantes inferiores apresentados na tabela 4.4 são melhores que todos aqueles reportados no *benchmark* do TUP até então. Além disso, 18 das 20 melhores soluções apresentadas nesta tabela são melhores que as melhores soluções encontradas pelos métodos exatos (dentro dos limites de tempo) reportadas em [48]. Mais importante que isso, 11 destas soluções (valores sublinhados) são melhores que a melhor solução obtida por qualquer método publicado na literatura daquela época. Na próxima seção, vamos mostrar que a heurística proposta consegue obter resultados ainda melhores.

			Melhor limitante inferior em 3 horas				Melhor solução em 3 horas			
Instância	$n-d_1$	$\lfloor \frac{n}{2} \rfloor - d_2$	$\mathcal{F}_1$-M	$\mathcal{F}_1$-BB	$\mathcal{F}_{2+}$-M	$\mathcal{F}_{2+}$-BB	$\mathcal{F}_1$-M	$\mathcal{F}_1$-BB	$\mathcal{F}_{2+}$-M	$\mathcal{F}_{2+}$-BB
14	6	3	149422	**150041**	149883	149904	167931	167800	**164169**	165975
14	5	3	149446	**150270**	149541	150194	160069	160591	161905	**159511**
14A	6	3	143138	143207	**143931**	142456	163872	163622	**159054**	161201
14A	5	3	142230	142321	**143504**	142600	158604	158878	**154840**	159703
14B	6	3	141839	142196	142608	**143378**	165052	164691	**158050**	164089
14B	5	3	142117	142237	**143706**	143147	157907	157174	**154739**	157724
14C	6	3	140967	141213	**141791**	140393	163934	159388	**153841**	157034
14C	5	3	141717	141299	141558	**141801**	154944	156541	**152046**	153178
16	8	2	142005	143420	142019	**143840**	185505	**183167**	206299	186108
16	7	3	143780	144265	144325	**145987**	207226	**188486**	190281	195551
16	7	2	138027	137443	139888	**141440**	177925	167697	170288	**163193**
16A	8	2	155542	157013	155743	**157972**	216598	199684	194050	**190233**
16A	7	3	156544	159146	157393	**160314**	210267	**205345**	209604	215136
16A	7	2	152063	154563	154023	**155342**	186439	180372	**176793**	182317
16B	8	2	152675	157356	153734	**158035**	**206044**	212300	240845	208859
16B	7	3	154667	157955	155988	**158244**	**215692**	237571		217261
16B	7	2	151406	155173	152218	**155403**	203491	192062	191213	**184991**
16C	8	2	157239	158164	158051	**160596**	205626	**192741**	213953	205095
16C	7	3	159916	160438	159970	**161838**	211880	**206505**	222039	223262
16C	7	2	156421	156810	157357	**158527**	186989	192311	182307	**182011**
30	5	5	336470	75891*	**367877**	361175				

Tabela 4.4: Resultados para as formulações inteiras nas instâncias relaxadas com pelo menos 14 times ($d_1 + d_2 > 0$). Tempo de execução limitado em 3 horas. Os melhores valores aparecem em negrito. Os valores sublinhados superam os melhores valores da literatura.

4.3.3 Resultados da heurística *relax-and-fix*

Usamos o CPLEX para resolver as formulações na heurística *relax-and-fix*, ajustado com os parâmetros apresentados na tabela 4.5 (um valor ausente nesta tabela indica que o valor padrão do parâmetro foi mantido). Permitimos que o *branch-and-cut* execute somente uma *thread*. Uma vez que resolver estas formulações na otimalidade torna a heurística muito lenta, limitamos a enumeração em no máximo 100 nós. Desativamos a repetição da rotina de *presolve* por causa de um erro no CPLEX 12.5.0.1, reportado pelos desenvolvedores da IBM, que é ativado pela nossa heurística. Para as instâncias com no máximo 18 times, aplicamos as heurísticas periódica e RINS com maior frequência, e ativamos a heurística *feasibility pump*, limitando que os subproblemas resolvidos nestas rotinas enumerem no máximo 10 nós. Optamos também por efetuar *probing* muito agressivo nas variáveis e utilizar a estratégia *probing dive* na enumeração, o que ajuda a encontrar soluções inteiras mais cedo. Para as instâncias com mais de 18 times, como precisamos de velocidade, ajustamos o CPLEX para parar assim que a primeira solução inteira for encontrada.

Aplicamos a heurística *relax-and-fix* variando $b = 1, 2, \ldots, 10$ para todas as instâncias incluídas nos experimentos anteriores. A heurística obteve uma solução ótima para todas as instâncias estritas com até 10 times, à exceção de uma. (A melhor solução obtida pela heurística para a instância 10C possui distância 43193, muito próxima da distância ótima que é 43149.) O tempo total gasto para obter estas soluções para todos os valores de b foi

Parâmetro	≤ 18 times	> 18 times
Número de *threads* paralelas (CPX_PARAM_THREADS)	1	1
Limite de nós do modelo principal (CPX_PARAM_NODELIM)	100	100
Repetir presolve (CPX_PARAM_REPEATPRESOLVE)	Desativado	Desativado
Frequência da heurística periódica (CPX_PARAM_HEURFREQ)	2	
Frequência da heurística RINS (CPX_PARAM_RINSHEUR)	20	
Aplicar heurística *feasibility pump* (CPX_PARAM_FPHEUR)	Ativada	
Limite de nós dos subproblemas (CPX_PARAM_SUBMIPNODELIM)	10	
Nível de *probing* (CPX_PARAM_PROBE)	Muito agressivo	
Estratégia de *probing* na enumeração (CPX_PARAM_DIVETYPE)	*Probing dive*	
Limite de soluções inteiras (CPX_PARAM_INTSOLLIM)		1

Tabela 4.5: Parâmetros do *branch-and-cut* do CPLEX utilizados na resolução das formulações na heurística *relax-and-fix*.

no máximo 1 segundo para as instâncias com 4 e 6 times, no máximo 10 segundos para as instâncias com 8 times, no máximo 2 minutos para as instâncias 10, 10A e 10B, e por volta de 3 minutos para a instância 10C. As tabelas 4.6 e 4.7 apresentam, respectivamente, as distâncias das soluções e os tempos de execução para as instâncias estritas e relaxadas com pelo menos 14 times. Infelizmente, assim como nas tentativas anteriores na literatura, a heurística *relax-and-fix* também não conseguiu encontrar soluções factíveis para as instâncias estritas com mais de 14 times. Para as instâncias em que foram encontradas soluções, a heurística demonstrou ser confiável, pois foi capaz de obter soluções em 240 das 250 execuções (25 instâncias e 10 valores de b). A maioria das melhores soluções (14 das 25) foram encontradas com $b = 4$, 6 ou 7. Por fim, também acrescentamos na tabela 4.7 o total acumulado de tempo para encontrar as soluções para $b \in \{4, 6, 7\}$ (coluna "4,6,7") e para $b \in \{1, 2, \ldots, 10\}$ (coluna "1–10").

4.3.4 Melhores resultados

Vamos comparar agora os melhores resultados obtidos pelos nossos métodos (exatos e heurístico) com os melhores resultados reportados na literatura até então. A comparação das formulações mais relevante para as instâncias estritas com até 10 times são os tempos de execução, já que todas foram resolvidas na otimalidade. A tabela 4.2 indica que a formulação fortalecida $\mathcal{F}_{2+}$ é melhor que $\mathcal{F}_1$ nesse critério para estas instâncias menores, e que a escolha do valor de k é importante para atingir bons resultados.

Os principais avanços que obtivemos em relação à literatura foram nas instâncias com pelo menos 14 times, as quais não possuíam soluções ótimas conhecidas naquela época. A tabela 4.8 sumariza os melhores resultados para estas instâncias. Os melhores limitantes inferiores e superiores (soluções) da literatura foram obtidos em 3 horas de computação para as instâncias com 14 e 16 times, e em 5 horas para a instância de 30 times (somente solução, nenhum limitante inferior foi reportado anteriormente para esta instância). Os limitantes inferiores fornecidos pela formulação relaxada $\mathcal{F}^R_{2+}$-BB já são melhores que os melhores limitantes inferiores da literatura em 25 das 29 instâncias. Estes podem ser calculados em menos de 1 minuto para as instâncias com até 16 times, e em aproximadamente 2 horas para a instância com 30 times. No período de 3 horas de computação, os melhores limitantes inferiores obtidos pelo *branch-and-cut* para $\mathcal{F}_{2+}$-BB

			Tamanho da janela móvel de variáveis binárias (parâmetro b)									
Instância	$n-d_1$	$\lfloor \frac{n}{2} \rfloor - d_2$	1	2	3	4	5	6	7	8	9	10
14	7	3		178226	174652	168408	170629	169981	168408	**165573**	166942	170250
14	6	3	165529	165764	**159522**	163818	159622	162377	160907	159601	164450	163573
14	5	3	160913	158103	156442	158054	158293	156456	157942	157404	**155439**	155958
14A	7	3	168422	171938	166357	164136	160830	163471	163860	164784	**160046**	163433
14A	6	3		156183	157219	162614	158619	156437	158237	157034	158133	**154628**
14A	5	3	153955	154251	154218	154891	152588	154441	**149331**	149956	152855	152490
14B	7	3		173698	162952	162634	160242	163629	**157884**	162305	**157884**	163329
14B	6	3	162891	163161	158818	157927	157222	**153611**	158291	155358	158382	158735
14B	5	3	156321	155125	153724	**150268**	150865	151251	150933	150760	150954	153202
14C	7	3	176479	174595	166559	163227	160262	160274	163720	**159518**	164778	161049
14C	6	3	164223	155523	158304	**152158**	154954	154026	155025	155435	154366	155200
14C	5	3	153697	151914	**149662**	150415	149727	150346	151048	150678	150471	151426
16	8	2	168647	168094	166688	**160705**	161771	167482	163733	164187	165098	167015
16	7	3		181556	177557	**169994**	174009	170487	172549	179940	173088	177242
16	7	2	163438	157184	153996	158043	157379	155796	155766	154153	158130	**153978**
16A	8	2	183344	174841	178320	173956	175092	176821	**173950**	176664	176266	174546
16A	7	3	186667	193983	185198	188957	183192	192059	182889	**181119**	188879	188475
16A	7	2	177235	165675	168224	167010	168058	**164620**	168032	170486	167538	168810
16B	8	2	187840	186326	187514	185223	186853	185541	**182673**	184750	183238	185253
16B	7	3	207308	194064	189257	191234	188064	191026	187488	188198	189596	**187007**
16B	7	2	183841	172083	171962	172704	172336	172688	**170194**	172169	172274	173143
16C	8	2	193358	184873	181205	**180221**	182568	184103	182706	187598	185487	184753
16C	7	3	189776		188855	191021	187649	**185528**	189251	189367		194055
16C	7	2	174744	174401	172033	171802	170909	**169184**	170267	171046	172319	170854
30	5	5	**466765**	484447	479056	475572	487552	471724				

Tabela 4.6: Soluções obtidas pela heurística *relax-and-fix* nas instâncias estritas e relaxadas com pelo menos 14 times. Os melhores valores aparecem em negrito e os valores em branco indicam que o método não encontrou uma solução.

			Tamanho da janela móvel de variáveis binárias (parâmetro b)											
Instância	$n-d_1$	$\lfloor \frac{n}{2} \rfloor - d_2$	1	2	3	4	5	6	7	8	9	10	4,6,7	1–10
14	7	3	157	254	339	501	716	947	1099	1238	1528	1815	2547	8594
14	6	3	147	262	302	599	721	953	1106	1181	1424	1336	2657	8031
14	5	3	125	192	278	493	725	810	989	944	1053	1112	2291	6721
14A	7	3	179	383	511	722	775	1132	1376	1647	1703	1743	3231	10171
14A	6	3	159	348	485	578	898	1053	1356	1374	1404	1546	2988	9201
14A	5	3	150	254	397	640	802	931	1099	1227	1316	1319	2670	8134
14B	7	3	170	331	504	497	785	901	1232	1405	1386	1641	2631	8852
14B	6	3	170	303	524	677	925	1172	1290	1453	1552	1579	3139	9645
14B	5	3	150	272	434	680	917	1096	1170	1170	1251	1331	2946	8472
14C	7	3	151	334	608	681	889	1062	1342	1346	1578	1674	3085	9665
14C	6	3	151	271	476	539	736	1068	1244	1371	1433	1434	2851	8723
14C	5	3	137	227	384	588	775	995	1045	1198	1241	1223	2629	7813
16	8	2	701	1398	1596	1711	2022	2287	2529	2968	3079	3077	6528	21369
16	7	3	722	1577	1876	2231	3264	4073	4417	5199	5710	6073	10721	35143
16	7	2	771	1293	1620	2169	2572	2800	2901	3239	3628	3584	7869	24577
16A	8	2	848	1333	1676	1729	2200	2259	2351	2818	3045	2954	6339	21213
16A	7	3	954	1690	1952	2718	3311	3537	4473	4830	5591	6506	10728	35560
16A	7	2	821	1110	1984	2099	2272	2379	3089	3191	3469	3819	7567	24232
16B	8	2	927	1633	1861	1985	2325	2577	2934	2968	3305	3396	7495	23911
16B	7	3	935	1464	1939	2536	2981	4265	4296	4519	5534	5832	11097	34301
16B	7	2	872	1316	1950	2128	2499	2948	2996	3525	3438	4112	8072	25784
16C	8	2	844	1326	1563	1892	1842	2211	2671	2791	3187	3557	6774	21884
16C	7	3	764	1188	1812	2433	2721	3235	3991	4615	1122	4836	9659	26717
16C	7	2	834	1151	1558	1780	1948	2437	2729	2893	3012	3200	6947	21543
30	5	5	16352	13817	11825	10821	8397	14917	5208	4841	5033	6455	30946	97666

Tabela 4.7: Tempos de execução (segundos) da heurística *relax-and-fix* nas instâncias estritas e relaxadas com pelo menos 14 times.

superam todos os limitantes inferiores conhecidos naquela época em margens que variam de 8 até 24 mil milhas. A coluna "Melhor $\mathcal{F}$" sob "Limitante inferior" apresenta os melhores limitantes que encontramos para as diferentes formulações avaliadas nos experimentos. Estes limitantes são melhores que aqueles de $\mathcal{F}_{2+}$-BB em 13 das 29 instâncias.

No lado direito da tabela 4.8, comparamos a qualidade das melhores soluções conhecidas na literatura até então com aquelas obtidas pelas nossas formulações e heurística *relax-and-fix*. Todas as nossas melhores soluções foram obtidas por alguma configuração da heurística *relax-and-fix* (coluna "Melhor RF"). A única instância que não foi melhorada com relação ao resultado da literatura foi 14C-7-3 (por apenas 57 milhas). Para as 24 demais instâncias com soluções conhecidas, nossas melhores soluções possuem distâncias entre 0,36 e 27,6 mil milhas menores. Quando comparado com as melhores soluções obtidas pelas formulações (coluna "Melhor $\mathcal{F}$"), as melhores soluções da heurística *relax-and-fix* possuem distâncias entre 1,68 e 28,6 mil milhas menores. Contudo, exceto pelas instâncias com 14 times, esta comparação é injusta pois encontrar a melhor solução da heurística *relax-and-fix* para todos os valores de $b \in \{1, 2, \ldots, 10\}$ ultrapassa os limites de 3 e 5 horas (veja a tabela 4.7). Portanto, incluímos a coluna "RF 4,6,7" que contém a melhor solução obtida nas execuções da heurística *relax-and-fix* com $b = 4$, 6 e 7. Estas execuções obtêm a maioria dos nossos melhores resultados (14 de 25) e, no geral, permanece dentro dos limites de tempo com apenas duas exceções: instância 16B-7-3, que

			Limitante inferior				Valor da solução				
Instância	$n-d_1$	$\lfloor \frac{n}{2} \rfloor - d_2$	Melhor lit.	$\mathcal{F}^R_{2+}$-BB	$\mathcal{F}_{2+}$-BB	Melhor $\mathcal{F}$	Melhor lit.	Melhor $\mathcal{F}$	Pior RF	RF 4,6,7	Melhor RF
14	7	3	141253	143089	150871	150871	166964	174715	178226	168408	165573
14	6	3	141064	142716	149904	150041	173681	164169	165764	160907	159522
14	5	3	141134	142478	150194	150270	165558	159511	160913	156456	155439
14A	7	3	133279	135149	141308	143517	160407	165968	171938	163471	160046
14A	6	3	133194	134971	142456	143931	164857	159054	162614	156437	154628
14A	5	3	133023	134884	142600	143504	162380	154840	154891	149331	149331
14B	7	3	131373	131684	142614	142614	161129	168659	173698	157884	157884
14B	6	3	130799	131542	143378	143378	168476	158050	163161	153611	153611
14B	5	3	130628	131301	143147	143706	160443	154739	156321	150268	150268
14C	7	3	126843	131163	138601	141268	159461	164512	176479	160274	159518
14C	6	3	126613	130921	140393	141791	166395	153841	164223	152158	152158
14C	5	3	126427	130556	141801	141801	163662	152046	153697	150346	149662
16	8	4	134471	134151	150523	151748					
16	8	2	134347	132563	143840	143840	178775	183167	168647	160705	160705
16	7	3	121933	127593	145987	145987	185966	188486	181556	169994	169994
16	7	2	121670	126946	141440	141440	166114	163193	163438	155766	153978
16A	8	4	148377	147551	166101	166626					
16A	8	2	146992	145595	157972	157972	188432	190233	183344	173950	173950
16A	7	3	137178	142945	160314	160314	199016	205345	193983	182889	181119
16A	7	2	137806	142056	155342	155342	172728	176793	177235	164620	164620
16B	8	4	146646	146805	161882	162251					
16B	8	2	145058	145383	158035	158035	201039	206044	187840	182673	182673
16B	7	3	139833	141838	158244	158244	202395	215692	207308	187488*	187007
16B	7	2	139742	141552	155403	155403	184923	184991	183841	170194	170194
16C	8	4	145012	152698	164235	165431					
16C	8	2	144398	150451	160596	160596	202023	192741	193358	180221	180221
16C	7	3	142467	148932	161838	161838	213157	206505	194055	185528	185528
16C	7	2	142399	148481	158527	158527	181013	182011	174744	169184	169184
30	5	5		336124	361175	367877	483224		487552	471724*	466765

Tabela 4.8: Comparação entre os melhores resultados obtidos pelos nossos métodos exatos e heurístico e os melhores resultados da literatura nas instâncias estritas e relaxadas com pelo menos 14 times.

excede em apenas 5 minutos o limite de 3 horas, e instância 30 (marcadas com um "*" na tabela 4.8). Quando comparadas com as melhores soluções da literatura, as soluções em "RF 4,6,7" possuem menores distâncias em 22 dos 25 casos. As piores soluções obtidas pela heurística *relax-and-fix* (coluna "Pior RF") são melhores que as melhores soluções da literatura em 18 dos 25 casos. Por fim, para a instância relaxada com 30 times, 4 das 6 soluções obtidas pela heurística *relax-and-fix* são melhores que a melhor solução reportada na literatura, apresentando uma redução considerável de 16459 milhas.

4.4 Conclusões

Ao fortalecer uma formulação matemática existente para o TUP, obtivemos outra formulação que além de ser resolvida mais rapidamente que a versão original, também fornece limitantes inferiores e superiores melhores para as instâncias do *benchmark* do TUP. Esta nova formulação desempenha um papel crucial em nossa implementação de uma heurística *relax-and-fix* para este problema, pois cada iteração da heurística envolve a resolução de uma formulação intermediária que não pode consumir muito tempo. Como resultado disso, melhoramos todos os limitantes inferiores para as instâncias do *benchmark*, bem como 24 dos 25 melhores limitantes superiores (soluções), conhecidos na época em que o

estudo foi realizado. Além disso, fomos os primeiros a fornecer limitantes inferiores fortes para as instâncias com mais de 16 times.

Após a conclusão deste estudo, foram publicados novos métodos aproximativos, exatos e heurísticos para o TUP [5, 35, 42, 43, 50, 51, 56]. Acreditamos que o estudo apresentado neste capítulo mostrou que havia margens para melhoria dos resultados para as instâncias do *benchmark* do TUP. Isso motivou outros pesquisadores a desenvolverem tais métodos e, desta forma, estimulamos o avanço da literatura deste problema. Cabe ressaltar também que o método de decomposição proposto em [51] empregou a formulação proposta neste capítulo. Desta forma, a nossa formulação promoveu o surgimento posterior de uma nova abordagem (tornando-a viável) que obteve limitantes inferiores ainda melhores para as instâncias do *benchmark* do TUP.

Mesmo após este estudo (e os mais atuais), o TUP permanece sendo ainda um problema que apresenta um alto grau de dificuldade, possuindo muitas instâncias de médio porte sem soluções factíveis (ou ótimas) conhecidas. Acreditamos que a combinação de métodos exatos e heurísticos é uma direção de pesquisa promissora. As formulações do TUP merecem um estudo poliédrico aprofundado e nossa heurística *relax-and-fix* pode ser modificada de várias formas. Por exemplo, a janela móvel de variáveis binárias pode ter diferentes formas, com foco antecipadamente na atribuição dos árbitros em rodadas mais problemáticas, e/ou usar randomização. Finalmente, pouca atenção tem sido dada para a melhoria do modelo de programação por restrições proposto em [47]. Uma versão mais efetiva desse modelo (com rotinas de busca mais robustas) poderia se tornar parte da heurística *relax-and-fix*.

Capítulo 5

Formulação matemática baseada em sub-rotas

As formulações matemáticas baseadas em partição de conjuntos e em fluxo em rede propostas para o TUP em [42] e [56] podem ser consideradas complementares no sentido que a primeira produz limitantes inferiores fortes para as instâncias com até 18 times e a segunda para as instâncias com mais de 18 times. Neste capítulo, apresentamos o estudo publicado em [35] onde introduzimos uma nova formulação matemática baseada em sub-rotas para o TUP que explora os pontos fortes destas duas formulações. Experimentos computacionais mostram que a nova formulação proposta encontra vários limitantes inferiores melhores que aqueles obtidos pelas duas formulações anteriores para as instâncias com até 18 times, e limitantes inferiores melhores que todos os conhecidos na literatura para as instâncias com mais de 18 times.

5.1 Formulação matemática

O modelo de programação linear inteira descrito neste capítulo generaliza duas das formulações propostas para o TUP na literatura. Em [56], os autores introduzem uma formulação baseada em fluxo em rede (FFR) cujas variáveis representam as viagens realizadas pelos árbitros entre duas rodadas consecutivas durante o torneio. Uma relaxação Lagrangiana desta formulação pode ser resolvida rapidamente e produz limitantes inferiores bons. Ainda em [56], e também em [42], é apresentada uma formulação baseada em partição de conjuntos (FPC) cujas variáveis representam sequências completas de viagens dos árbitros, percorrendo todas as $4n - 2$ rodadas do torneio e satisfazendo as restrições (2.3)–(2.5). Embora a relaxação linear da FPC retorne limitantes inferiores consideravelmente mais fortes que a relaxação Lagrangiana da FFR, o tempo necessário para resolver a primeira aumenta rapidamente à medida que a quantidade de times aumenta, o que torna inviável otimizar a FPC em instâncias com mais de 18 times.

O nosso modelo generaliza as formulações anteriores no seguinte sentido. Enquanto as variáveis da FFR e FPC representam sequências de viagens de tamanho 2 e $4n - 2$ (em termos da quantidade de sedes/rodadas percorridas), respectivamente, o tamanho das sequências de viagens representadas pelas variáveis do nosso modelo é definido por

um parâmetro que pode ter qualquer valor entre 2 e $4n-2$ (incluindo os extremos). Esta flexibilidade nos permite estudar empiricamente a relação de custo-benefício entre o tempo de otimização (vantagem da FFR) e a qualidade do limitante inferior (vantagem da FPC).

Seja $2 \leq w \leq 4n-2$ o parâmetro mencionado acima que determina o tamanho das sequências de viagens. Para um valor fixo de w, criamos as variáveis do modelo dividindo o torneio T da instância em seções indexadas pelos inteiros em $S = \{1, 2, \ldots, \lceil \frac{4n-3}{w-1} \rceil\}$ da seguinte forma. Para $s \in S$, a s-ésima seção de T, denotada por T_s, é composta pelas rodadas consecutivas que vão de $(s-1)(w-1)+1$ até $\min\{s(w-1)+1, 4n-2\}$. Note que todas as seções possuem exatamente w rodadas, exceto pela última, que pode ser menor. A figura 5.1 ilustra um torneio com quatro times e seis rodadas subdivididas em seções para $w = 2$, 3, 4 e 6.

$w = 2$

Rodadas					
1	2	3	4	5	6
	T_2		T_4		
(1,3)	(1,2)	(1,4)	(3,1)	(2,1)	(4,1)
(2,4)	(3,4)	(3,2)	(4,2)	(4,3)	(2,3)
T_1		T_3		T_5	

$w = 3$

Rodadas					
1	2	3	4	5	6
		T_2			
(1,3)	(1,2)	(1,4)	(3,1)	(2,1)	(4,1)
(2,4)	(3,4)	(3,2)	(4,2)	(4,3)	(2,3)
T_1				T_3	

$w = 4$

Rodadas					
1	2	3	4	5	6
			T_2		
(1,3)	(1,2)	(1,4)	(3,1)	(2,1)	(4,1)
(2,4)	(3,4)	(3,2)	(4,2)	(4,3)	(2,3)
T_1					

$w = 6$

Rodadas					
1	2	3	4	5	6
(1,3)	(1,2)	(1,4)	(3,1)	(2,1)	(4,1)
(2,4)	(3,4)	(3,2)	(4,2)	(4,3)	(2,3)
T_1					

Figura 5.1: Seções de um torneio com 4 times e 6 rodadas para $w = 2$, 3, 4 e 6.

Para cada seção $s \in S$, o nosso modelo contém variáveis para representar cada sequência de viagem que percorre todas as rodadas em T_s e satisfaz as restrições (2.4) e (2.5). Como existe somente uma seção quando $w = 4n-2$, também é exigido que as sequências de viagens obedeçam à restrição (2.3) neste caso particular. Quando $2 \leq w < 4n-2$, não é possível impor diretamente (2.3). No entanto, note que seções consecutivas possuem uma rodada em comum (veja a figura 5.1), o que nos permite conectar as suas sequências de viagens para criar sequências maiores. Na próxima seção vamos introduzir a formulação matemática proposta e detalhar as restrições que garantem que as sequências de viagens sejam corretamente combinadas a fim de planejar rotas factíveis de viagens para os n árbitros.

5.1.1 Formulação matemática inicial

Para um valor fixo de w e qualquer $s \in S$, definimos P_s como sendo o conjunto das sequências de viagens em T_s que percorrem todas as suas rodadas e satisfazem as restrições (2.4) e (2.5). (Quando $w = 4n - 2$, vamos ter somente P_1 e, neste caso, as sequências também deverão satisfazer a restrição (2.3)). Para cada $p \in P = \bigcup_{s \in S} P_s$, crie uma variável binária x_p que é igual a um quando p é parte da solução do modelo, e igual a zero caso contrário. Vamos denotar a distância percorrida na sequência de viagens p por d_p. Definimos G_s como sendo o conjunto dos jogos nas rodadas de T_s, e P_{sg} o conjunto de todas as sequências em P_s que contém o jogo $g \in G_s$. Doravante, vamos usar o termo *rota simples* para nos referir a qualquer sequência de viagens em P, e o termo *rota* para nos referir a uma sequência ordenada de rotas simples $r_1, \ldots, r_m$ tal que r_i e r_{i+1} são de seções consecutivas, e o último jogo de r_i é igual ao primeiro jogo de r_{i+1} para qualquer $i = 1, \ldots, m-1$. Dada uma rota Q, denotaremos por $P(Q)$ o conjunto de todas as rotas simples em Q. Uma *rota completa* é uma rota que visita todas as rodadas do torneio, ou seja, contém uma rota simples de cada seção de T. Uma rota é infactível quando ela contém dois ou mais jogos que violam as restrições (2.4) ou (2.5), ou quando ela é uma rota completa e viola a restrição (2.3). Por fim, vamos denotar o conjunto de todas as rotas infactíveis por $\mathbb{U}$. Agora estamos prontos para apresentar nossa formulação matemática.

$$\min \sum_{p \in P} d_p x_p \tag{5.1}$$

Sujeito a:

$$\sum_{p \in P_{sg}} x_p = 1, \quad \forall\, s \in S, g \in G_s, \tag{5.2}$$

$$\sum_{p \in P(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}, \tag{5.3}$$

$$x_p \in \{0, 1\}, \quad \forall\, p \in P. \tag{5.4}$$

A função objetivo (5.1) minimiza a distância total viajada pelos árbitros. A restrição (5.2) garante que todos os jogos em cada seção sejam visitados por uma rota simples, exceto pelos jogos nas rodadas compartilhadas por duas seções consecutivas que são visitados por duas rotas simples (uma terminando e outra iniciando neste jogo). As restrições (5.2) e (5.4) juntas asseguram que uma solução factível seja formada por n rotas completas que satisfazem as restrições (2.1) e (2.2). As restrições (2.3)–(2.5) são impostas por (5.3), que previne que rotas infactíveis façam parte da solução excluindo ao menos uma de suas rotas simples constituintes. Vamos denotar por $\mathcal{T}$ o politopo do TUP, ou seja, a envoltória convexa de (5.2)–(5.4).

Esta formulação matemática com $w = 2$ e junto com a desigualdade (5.5), introduzida na próxima seção, é equivalente à formulação baseada em fluxo em rede apresentada em [56]. Quando $w = 4n - 2$, as desigualdades (5.3) e (5.5)–(5.7) (vistas a seguir) se tornam desnecessárias e esta formulação é equivalente à formulação baseada em partição de conjuntos introduzida em [42, 56] (sem os cortes adicionais apresentados em [56]).

5.1.2 Desigualdades válidas fortes

A relaxação linear de (5.2)–(5.4) não provê limitantes inferiores fortes, principalmente porque (5.3) é uma restrição fraca. Em [56], os autores apresentam uma forma de fortalecer (5.3) aplicando o *lifting* proposto em [23], explicado a seguir. Seja $U = (u_1, u_2, \ldots)$ uma rota infactível, e $H^+(U) = P(U) \cup \{p \in P \mid (u_1, u_2, \ldots, u_i, p)$ é uma rota infactível para algum $i = 1, \ldots, |P(U)| - 1\}$. Então, (5.5) é uma versão mais forte de (5.3).

$$\sum_{p \in H^+(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}. \tag{5.5}$$

A corretude de (5.5) vem do fato que, por construção, quaisquer $|P(U)|$ rotas simples em $H^+(U)$ que satisfazem (5.2) formam uma rota infactível. Alternativamente, as provas de corretude para desigualdades similares para o problema de roteamento de veículos com janelas de tempo vistas em [23] também podem ser aplicadas a (5.5).

A fim de obter desigualdades válidas adicionais para $\mathcal{T}$, vamos explorar algumas características de simetria do TUP. Ao invertermos a ordem das rodadas de um torneio, transformando a rodada r na rodada $4n-1-r$, para todo $1 \leq r \leq 4n-2$, vamos resultar em uma instância modificada do problema que é equivalente à instância original. A diferença fundamental é que os árbitros percorrem rotas com os sentidos também invertidos. As seções do torneio também são invertidas, ou seja, a seção $s' = |S| - s + 1$ da instância modificada contém a rodada $r' = 4n - 1 - r$ se, e somente se, a seção s da instância original contiver a rodada r. Portanto, $P'_{s'}$ (o conjunto das rotas simples na seção s' da instância modificada) contém as rotas simples invertidas que pertencem a P_s na instância original. Como consequência, as variáveis da formulação da instância modificada são equivalentes às variáveis das rotas invertidas correspondentes na formulação da instância original. Aplicando esta equivalência à versão de (5.5) para a instância modificada, vamos obter (5.6), que é válida para $\mathcal{T}$ na instância original.

$$\sum_{p \in H^-(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}, \tag{5.6}$$

onde $H^-(U) = P(U) \cup \{p \in P \mid (p, u_i, u_{i+1}, \ldots, u_{|P(U)|})$ é uma rota infactível para algum $i = 2, \ldots, |P(U)|\}$. As desigualdades (5.5) e (5.6) são linearmente independentes e, portanto, não são redundantes juntas. Além disso, os resultados computacionais apresentados na seção 4.3 indicam que a inclusão de (5.6) fortalece consideravelmente a relaxação linear de (5.2)–(5.5).

A seguir, vamos obter duas famílias adicionais de desigualdades válidas para $\mathcal{T}$ derivadas de cliques em grafos de conflitos. Seja $s \in S$, $s \neq |S|$, $g \in G_s \cap G_{s+1}$, definimos A_{sg} como sendo o grafo cujos vértices correspondem às rotas simples em P_s que terminam no jogo g, e às rotas simples em P_{s+1} que começam no jogo g. Denotaremos por $v^A_{sg}(p)$ o vértice de A_{sg} que corresponde à rota simples p. Dois vértices de A_{sg}, digamos $v^A_{sg}(p_1)$ e $v^A_{sg}(p_2)$, são adjacentes se, e somente se, p_1 e p_2 pertencerem à mesma seção ou, caso contrário, formarem juntos uma rota infactível. Vamos denotar o conjunto de cliques em

A_{sg} por $\mathbb{A}_{sg}$. A desigualdade (5.7) é válida para $\mathcal{T}$.

$$\sum_{p \mid v^A_{sg}(p) \in C} x_p \leq 1, \quad \forall\, s \in S, s \neq |S|, g \in G_s \cap G_{s+1}, C \in \mathbb{A}_{sg}. \tag{5.7}$$

De forma análoga, seja B_s o grafo cujos vértices correspondem às rotas simples em P_s para $s \in S$. Denotaremos por $v^B_s(p)$ o vértice de B_s que corresponde à rota simples p. Dois vértices em B_s, digamos $v^B_s(p_1)$ e $v^B_s(p_2)$, são adjacentes se, e somente se, p_1 e p_2 possuem ao menos um jogo em comum. O conjunto de cliques em B_s será denotado por $\mathbb{B}_s$. A desigualdade (5.8) é válida para $\mathcal{T}$ por causa de (5.2).

$$\sum_{p \mid v^B_s(p) \in C} x_p \leq 1, \quad \forall\, s \in S, C \in \mathbb{B}_s. \tag{5.8}$$

Vamos nos referir às restrições (5.5) e (5.6) como *desigualdades de caminho*, e às restrições (5.7) e (5.8) como *desigualdades de clique*.

5.2 Separação das desigualdades válidas fortes

Como a quantidade de desigualdades de caminho e clique cresce exponencialmente em função de n, é inviável adicioná-las *a priori* no modelo. Ao invés disso, desenvolvemos rotinas de separação que encontram desigualdades violadas destas famílias e utilizamos estas rotinas em um algoritmo de planos de corte. Primeiramente, vamos apresentar alguns resultados auxiliares que aprimoram a separação das desigualdades de caminho e, em seguida, descrevemos as rotinas de separação.

5.2.1 Resultados auxiliares para as desigualdades de caminho

Dizemos que uma rota infactível é *minimal à direita* (*esquerda*) se ela se tornar factível ao remover em sua sequência a rota simples posicionada mais à direita (esquerda). Uma rota infactível é *minimal* quando ela for minimal tanto à direita quanto à esquerda.

Proposição 5.1. *Se U não é uma rota infactível minimal à direita (esquerda), a sua desigualdade (5.5) ((5.6)) associada a U é redundante.*

Demonstração. Seja U uma rota infactível que não é minimal à direita, e Z o conjunto minimal de rotas simples em U que se removidas será obtida uma rota infactível U' minimal à direita. Ao somarmos, para cada $p \in Z$, as igualdades (5.2) com s igual a seção de p e g igual ao primeiro jogo de p, e adicionar ao resultado a desigualdade (5.5) associada a U', vamos resultar na desigualdade (5.5) associada a U. A prova para o caso minimal à esquerda é análoga. □

Note que as desigualdades (5.5) associadas a duas rotas infactíveis minimais à direita que diferem apenas em sua última rota simples, são idênticas. De forma análoga, duas rotas infactíveis minimais à esquerda que diferem apenas em sua primeira rota simples também geram a mesma desigualdade (5.6). Portanto, vamos apresentar a seguir versões

modificadas de (5.5) e (5.6) que previnem nosso algoritmo de separação de produzir desigualdades repetidas. Seja $\mathbb{F}$ o conjunto de rotas factíveis, e $\mathbb{F}' = \{F = (f_1, f_2, \dots) \in \mathbb{F} \mid f_{|P(F)|} \notin P_{|S|}\}$ e $\mathbb{F}'' = \{(f_1, f_2, \dots) \in \mathbb{F} \mid f_1 \notin P_1\}$ os conjuntos de rotas factíveis que excluem as rotas simples da última e primeira seção de T, respectivamente. Definimos $F' = (f'_1, f'_2, \dots) \in \mathbb{F}'$, $F'' = (f''_1, f''_2, \dots) \in \mathbb{F}''$, $K^+(F') = P(F') \cup \{p \in P \mid (f'_1, f'_2, \dots, f'_i, p)$ é uma rota infactível para algum $i = 1, \dots, |P(F')|\}$ e $K^-(F'') = P(F'') \cup \{p \in P \mid (p, f''_i, f''_{i+1}, \dots, f''_{|P(F'')|})$ é uma rota infactível para algum $i = 1, \dots, |P(F'')|\}$. Ao invés de usarmos (5.5) e (5.6), substituiremos estas desigualdades por (5.9) e (5.10), respectivamente.

$$\sum_{p \in K^+(F)} x_p \leq |P(F)|, \quad \forall\, F \in \mathbb{F}', \tag{5.9}$$

$$\sum_{p \in K^-(F)} x_p \leq |P(F)|, \quad \forall\, F \in \mathbb{F}''. \tag{5.10}$$

Note que (5.9) e (5.10) são equivalentes às desigualdades (5.5) associadas a rotas minimais à direita e (5.6) associadas a rotas minimais à esquerda, respectivamente, contudo as primeiras são definidas em função de rotas factíveis, ao passo que as últimas são definidas em relação a rotas infactíveis. Por exemplo, dada uma rota infactível U minimal à direita (esquerda), a desigualdade (5.5) ((5.6)) para U é igual à desigualdade (5.9) ((5.10)) para a rota factível obtida ao remover o último (primeiro) jogo em U. Além disso, a desigualdade (5.9) ((5.10)) para uma dada rota factível F elimina apenas rotas infactíveis minimais à direita (esquerda) (veja a proposição 5.1) e existe uma correspondência um-para-um entre uma desigualdade (5.9) ou (5.10) e uma rota factível F em $\mathbb{F}'$ ou $\mathbb{F}''$ porque $K^+(F)$ e $K^-(F)$ são determinados unicamente a partir de F.

Embora nossa rotina de separação (vista na próxima seção) procure por violações de (5.9) e (5.10), estes cortes podem ser densos, o que reduz potencialmente o desempenho computacional. Portanto, vamos adicionar no modelo as versões esparsas (5.11) e (5.12) equivalentes a (5.9) e (5.10), respectivamente.

$$\sum_{p \in \tilde{K}^+(F')} x_p - x_{f'_1} \geq 0, \quad \forall\, F' \in \mathbb{F}', \tag{5.11}$$

$$\sum_{p \in \tilde{K}^-(F'')} x_p - x_{f''_{|P(F'')|}} \geq 0, \quad \forall\, F'' \in \mathbb{F}'', \tag{5.12}$$

onde $\tilde{K}^+(F') = \{p \in (P \setminus P(F')) \mid (f'_1, f'_2, \dots, f'_i, p)$ é uma rota factível para algum $i = 1, \dots, |P(F')|\}$, e $\tilde{K}^-(F'') = \{p \in (P \setminus P(F'')) \mid (p, f''_i, f''_{i+1}, \dots, f''_{|P(F'')|})$ é uma rota factível para algum $i = 1, \dots, |P(F'')|\}$. Intuitivamente, (5.11) e (5.12) são mais esparsas que (5.9) e (5.10) porque os conjuntos $\tilde{K}^+$ e $\tilde{K}^-$, empregados em (5.11) e (5.12), contêm rotas simples que resultam em factibilidade, o que tende a ser bem menos numeroso que as rotas simples que induzem infactibilidade contidas nos conjuntos K^+ e K^-, usados em (5.9) e (5.10). Dada uma rota $F' \in \mathbb{F}'$, obtemos (5.11) multiplicando (5.9) por -1 e adicionando ao resultado, para todo $i = 1, \dots, |P(F')|$, as igualdades (5.2) com $s = i + 1$ e g igual ao último jogo de f'_i. De forma análoga, podemos combinar a negação de (5.10)

com (5.2) para obter (5.12).

5.2.2 Separação das desigualdades de caminho

O algoritmo 2 descreve a rotina de separação das desigualdades (5.9) para as rotas que violam as restrições (2.4) ou (2.5). Dada uma solução x^*, enumeramos as rotas em $\mathbb{F}'$ procurando por violações de desigualdades (5.9) através da chamada do procedimento SEP-FREQ-REC para cada seção $s \in S$, exceto para a última seção. Este procedimento verifica recursivamente as desigualdades (5.9) para as rotas que iniciam em s, o que pode tomar tempo exponencial. Desta forma, vamos ignorar estrategicamente algumas rotas, como visto a seguir.

Normalmente, a maioria das variáveis assumem valores nulos ou muitos próximos de zero na solução x^* dada como entrada, o que contribui muito pouco para potenciais violações de (5.9). Por isso, vamos desconsiderar rotas simples cujas variáveis possuem valores menores que 0,001. Para tanto, usamos os seguintes conjuntos no procedimento SEP-FREQ-REC: $P^+ = \{p \in P \mid x^*_p \geq 0{,}001\}$, $P^+_s = P_s \cap P^+$ e $P^+_{sg} = P_{sg} \cap P^+_s$. Os passos nas linhas 8–16 do algoritmo 2 enumeram todas as rotas factíveis obtidas pela adição de rotas simples em P^+_{sg} (ou P^+_s quando F é vazio) no fim de F (criando F'). Se F' induzir uma desigualdade (5.9) (linha 17) violada mais que 0,009 (limite mínimo imposto a fim de promover um progresso razoável no valor do limitante inferior), a desigualdade (5.11) correspondente é adicionada na formulação (linha 18). Nas linhas 20 e 21, são enumeradas rotas derivadas de F' de forma recursiva somente quando a próxima seção não é a última ($s+1 < |S|$) e quando $\sum_{p' \in K^+(F') \cap P^+} x^*_{p'} > |P(F')| - 1 + 0{,}009$. Se a desigualdade anterior não for satisfeita, as rotas que estendem F' não podem satisfazer a condição na linha 17. Para constatar isso, considere a rota F'' obtida ao adicionar ℓ rotas simples no fim de F'. A desigualdade na linha 17 para F'' terá o lado direito igual a $|P(F')| + \ell + 0{,}009$, e o lado esquerdo com as variáveis das rotas em $K^+(F') \cap P^+$ e variáveis adicionais cujos valores em x^* totalizam no máximo $\ell + 1$, o que resulta na desigualdade da linha 20 depois de cancelar ℓ em ambos lados. Esta verificação previne a enumeração desnecessária de um grande número de rotas. Os valores dos somatórios calculados nas linhas 17 e 20 permanecem disponíveis nas chamadas recursivas consecutivas para permitir atualizações incrementais, economizando um tempo adicional de computação.

Algoritmo 2 Rotina de separação das desigualdades (5.9) para rotas que violam as restrições (2.4) ou (2.5).

1: **procedimento** SEP-FREQ(solução x^*)
2: **para todo** $s \in S, s \neq |S|$ **faça**
3: SEP-FREQ-REC(x^*, (), s);
4: **fim para**
5: **fim procedimento**
6:
7: **procedimento** SEP-FREQ-REC(solução x^*, rota F, seção s)
8: **se** $F = ()$ **então**
9: Faça $E = P_s^+$;
10: **senão**
11: Faça g o último jogo em F;
12: Faça $E = P_{sg}^+$;
13: **fim se**
14: **para todo** $p \in E$ **faça**
15: Faça F' igual a F com p adicionado no fim;
16: **se** F' é uma rota factível **então**
17: **se** $\sum_{p' \in K^+(F') \cap P^+} x^*_{p'} > |P(F')| + 0{,}009$ **então**
18: Adicione na formulação a desigualdade (5.11) para F';
19: **fim se**
20: **se** $s + 1 < |S|$ e $\sum_{p' \in K^+(F') \cap P^+} x^*_{p'} > |P(F')| - 1 + 0{,}009$ **então**
21: SEP-FREQ-REC(x^*, F', $s + 1$);
22: **fim se**
23: **fim se**
24: **fim para**
25: **fim procedimento**

Vamos efetuar a separação das desigualdades (5.10) para as rotas que violam as restrições (2.4) ou (2.5) com simples modificações de SEP-FREQ e SEP-FREQ-REC, criando suas respectivas contrapartidas SEP-FREQ-INV e SEP-FREQ-INV-REC. O procedimento SEP-FREQ-INV chama SEP-FREQ-INV-REC para cada seção $s \in S$, exceto pela primeira. O procedimento SEP-FREQ-INV-REC também efetua a enumeração das rotas, mas percorrendo as seções em ordem inversa, com as seguintes modificações dos passos do algoritmo 2. O jogo g se torna o primeiro jogo de F na linha 11. A rota p passa a ser inserida no início de F na linha 15. As desigualdades nas linhas 17, 18 e 20 são modificadas para estar de acordo com (5.10) e (5.12). A primeira condição na linha 20 se torna $s - 1 > 1$. Por fim, o terceiro parâmetro da chamada recursiva na linha 21 agora é $s - 1$.

Empiricamente, não é muito vantajoso separar as desigualdades (5.9) e (5.10) que eliminam rotas infactíveis longas que violam as restrições (2.4) ou (2.5), a menos que estas rotas sejam minimais. Seja $\tilde{U} = (\tilde{u}_1, \tilde{u}_2, \dots)$ uma rota infactível minimal que viola apenas (2.4) ou (2.5) e $|P(\tilde{U})| \geq 3$. Se a sua rota interna $(\tilde{u}_2, \cdots, \tilde{u}_{|P(\tilde{U})|-1})$ percorre pelo

menos $q_{\max} - 1$ rodadas, $\tilde{U}$ não pode ser minimal, onde $q_{\max} = \max\{q_1, q_2\}$. Portanto, definimos subconjuntos de $\mathbb{F}'$ e $\mathbb{F}''$ para (5.9) e (5.10), respectivamente, que excluem desigualdades que eliminam somente rotas não minimais que violam (2.4) ou (2.5). Dada uma rota $Q = (q_1, q_2, \dots)$, considere $I(Q)$ igual $(|P(Q)| - 1)(w - 1) + 1$ se $|P(Q)| > 1$, ou 0 caso contrário (quantidade de rodadas percorridas na rota Q desconsiderando a sua primeira ou última rota simples). Definimos $\tilde{\mathbb{F}}' = \{F \in \mathbb{F}' \mid I(F) < q_{\max} - 1\}$ e $\tilde{\mathbb{F}}'' = \{F \in \mathbb{F}'' \mid I(F) < q_{\max} - 1\}$, e criamos as rotinas de separação Sep-Freq-Mnl e Sep-Freq-Mnl-Rec (Sep-Freq-Inv-Mnl e Sep-Freq-Inv-Mnl-Rec) para separar as desigualdades (5.9) ((5.10)) para rotas em $\tilde{\mathbb{F}}'$ ($\tilde{\mathbb{F}}''$). A rotina Sep-Freq-Mnl-Rec é obtida a partir da seguinte modificação de Sep-Freq-Rec. A condição extra $(|P(F')| - 1)(w - 1) + w < q_{\max} - 1$ é adicionada em conjunção às demais condições do "se" na linha 20 e, obviamente, a chamada recursiva na linha 21 se torna Sep-Freq-Mnl-Rec. A rotina Sep-Freq-Inv-Mnl-Rec é obtida de forma análoga a partir de Sep-Freq-Inv-Rec adicionando a condição $(|P(F')| - 1)(w - 1) + w < q_{\max} - 1$ antes de sua chamada recursiva. As rotinas Sep-Freq-Mnl e Sep-Freq-Inv-Mnl são semelhantes a Sep-Freq e Sep-Freq-Inv, contudo elas chamam Sep-Freq-Mnl-Rec e Sep-Freq-Inv-Mnl-Rec, respectivamente.

Agora vamos focar nas violações da restrição (2.3). As rotas factíveis em $\mathbb{F}'$ ($\mathbb{F}''$) que não passam pela sede de ao menos um time e incluem rotas simples de cada uma das seções $1, 2, \dots, |S| - 1$ $(2, 3, \dots, |S|)$ correspondem às desigualdades (5.9) ((5.10)) que eliminam rotas que podem violar (2.3). Estas desigualdades serão separadas pela rotina Sep-Vst-Rec (Sep-Vst-Inv-Rec) que é uma variação de Sep-Freq-Rec (Sep-Freq-Inv-Rec). Essencialmente, a desigualdade violada deve ser adicionada no modelo somente quando $s = |S| - 1$ ($s = 2$) e F' não passa pela sede de ao menos um time. Entretanto, constatamos empiricamente que o impacto no limitante inferior obtido ao separar (5.9) e (5.10) para rotas que violam (2.3) é pouco significativo, sendo que o tempo extra consumido pela rotina de separação é considerável. Por esse motivo, a fim de garantir que as soluções do modelo satisfaçam (2.3), optamos por separar (5.13) ao invés de (5.9) e (5.10).

$$\sum_{p \in K'(F')} x_p \leq |F'|, \quad \forall\ F' = (f'_1, f'_2, \dots) \in \mathbb{F}', \tag{5.13}$$

onde $K'(F') = P(F') \cup \{p \in P \mid (f'_1, f'_2, \dots, f'_{|P(F')|}, p)$ é uma rota infactível$\}$. Embora seja mais fraca que (5.9)–(5.10) (pois $K'(F') \subset K^+(F')$ quando $|F'| \geq 2$), (5.13) pode ser separada pela enumeração de bem menos rotas, o que reduz consideravelmente o custo computacional. O algoritmo 3 descreve a rotina de separação para (5.13). Esta rotina é semelhante a Sep-Vst-Rec, diferindo apenas com relação aos somatórios calculados nas linhas 12 e 16 do algoritmo 3. Apesar de Sep-Vst-Frc-Rec procurar por violações de (5.13), esta rotina adiciona na formulação a desigualdade (5.11) correspondente, que é mais forte, assim como é feito no algoritmo 2.

Algoritmo 3 Rotina de separação das desigualdades (5.13) para rotas que violam a restrição (2.3).

procedimento SEP-VST-FRC-REC(solução x^*, rota F, seção s)
  se $F = ()$ **então**
    Faça $E = P_s^+$;
  senão
    Faça g o último jogo em F;
    Faça $E = P_{sg}^+$;
  fim se
  para todo $p \in E$ **faça**
    Faça F' igual a F com p adicionado no fim;
    se F' é uma rota factível **então**
      se $s = |S| - 1$ e F' não visita a sede de ao menos um time **então**
        se $\sum_{p' \in K'(F') \cap P^+} x^*_{p'} > |F'| + 0{,}009$ **então**
          Adicione na formulação a desigualdade (5.11) para F';
        fim se
      fim se
      se $s + 1 < |S|$ e $\sum_{p' \in P(F') \cap P^+} x^*_{p'} > |F'| - 1 + 0{,}009$ **então**
        SEP-VST-FRC-REC(x^*, F', $s + 1$);
      fim se
    fim se
  fim para
fim procedimento

Muito embora as rotinas de separação descritas até aqui tenham sido apresentadas através de algoritmos recursivos, o que as torna mais fáceis de entender, todas elas foram implementadas como procedimentos não recursivos a fim de acelerar os tempos de execução. A complexidade de tempo destas rotinas é: Sep-Freq e Sep-Freq-Inv são $O(|S|n^{w+(w-1)(|S|-1)})$, Sep-Vst-Rec é $O(n^{w+(w-1)(|S|-1)})$, Sep-Freq-Mnl e Sep-Freq-Inv-Mnl são $O(|S|n^{w+(w-1)z})$ onde $z = \left\lceil \frac{max\{q_1,q_2\}+1}{w-1} \right\rceil\}$ (quantidade máxima de seções de uma rota minimal qualquer), e Sep-Vst-Frc-Rec é $O(|S|n^{2w-1})$.

5.2.3 Separação das desigualdades de clique

Vamos separar as desigualdades (5.7) e (5.8) da seguinte forma. Dada uma solução x^*, começamos construindo os grafos A_{sg} e B_s (veja a seção 5.1.2). Após atribuir peso x^*_p para cada vértice $v^A_{sg}(p)$ e $v^B_s(p)$, nós procuramos por uma clique de peso máximo em ambos grafos. Uma desigualdade violada existe se, e somente se, a clique de peso máximo encontrada possuir peso total maior que 1. Os algoritmos 4 e 5 descrevem as rotinas de separação para (5.7) e (5.8), respectivamente. Usamos o resolvedor Cliquer [32] para encontrar as cliques de peso máximo. Como este é um problema $\mathcal{NP}$-difícil [24], reduzimos o tamanho dos dois grafos considerando apenas os vértices das rotas simples p com

$x_p^* \geq 0{,}01$. Os subgrafos de A_{sg} e B_s obtidos desta forma são denotados por $\tilde{A}_{sg}$ e $\tilde{B}_s$, respectivamente. Além disso, como o resolvedor Cliquer lida apenas com pesos inteiros, o peso dos vértices $v_{sg}^{\tilde{A}}(p)$ e $v_s^{\tilde{B}}(p)$ é convertido para $\lfloor 100x_p^* \rfloor$, e já que encontrar uma clique de peso máximo pode tomar muito tempo, paramos a execução do resolvedor no momento em que uma clique com peso maior ou igual a 101 é encontrada. As desigualdades mais fortes em (5.7) e (5.8) são aquelas associadas a cliques maximais em A_{sg} e B_s, respectivamente. Portanto, após obter uma clique C em um dos subgrafos, nós inspecionamos o grafo original em busca de vértices ausentes em C que são adjacentes a todos os vértices de C. Se tal vértice existir, ele é incluído em C e o procedimento continua considerando a clique C atualizada e os vértices que ainda não foram verificados. Depois de percorrer todos os vértices, a desigualdade violada é adicionada à formulação. (Veja as linhas 8-12 no algoritmo 4, e as linhas 7-11 no algoritmo 5.) Por fim, cabe ressaltar que a complexidade de tempo dos algoritmos 4 e 5 é exponencial visto que eles envolvem a resolução do problema da clique de peso máximo.

Algoritmo 4 Rotina de separação das desigualdades (5.7).

1: **procedimento** SEP-CLIQUE-SEC-ADJ(solução x^*)
2: **para todo** $s \in S, s \neq |S|$ **faça**
3: **para todo** $g \in G_s \cap G_{s+1}$ **faça**
4: Construa o grafo $\tilde{A}_{sg}$ para as rotas $p \in P_{sg} \cup P_{(s+1)g}$ com $x_p^* \geq 0{,}01$;
5: Atribua peso $\lfloor 100x_p^* \rfloor$ para cada vértice $v_{sg}^{\tilde{A}}(p)$ de $\tilde{A}_{sg}$;
6: Execute o resolvedor Cliquer no grafo ponderado $\tilde{A}_{sg}$;
7: **se** uma clique C com peso maior ou igual a 101 foi encontrada **então**
8: **para todo** $p \in P_{sg} \cup P_{(s+1)g}$ **faça**
9: **se** $v_{sg}^{A}(p) \notin C$ e $v_{sg}^{A}(p)$ é adjacente, em A_{sg}, a todos os vértices de C **então**
10: Adicione $v_{sg}^{A}(p)$ em C;
11: **fim se**
12: **fim para**
13: Adicione na formulação a desigualdade (5.7) para C;
14: **fim se**
15: **fim para**
16: **fim para**
17: **fim procedimento**

Algoritmo 5 Rotina de separação das desigualdades (5.8).

procedimento SEP-CLIQUE-SEC-IGL(solução x^*)
  para todo $s \in S$ **faça**
    Construa o grafo $\tilde{B}_s$ para as rotas $p \in P_s$ com $x_p^* \geq 0{,}01$;
    Atribua peso $\lfloor 100 x_p^* \rfloor$ para cada vértice $v_s^{\tilde{B}}(p)$ de $\tilde{B}_s$;
    Execute o resolvedor Cliquer no grafo ponderado $\tilde{B}_s$;
    se uma clique C com peso maior ou igual a 101 foi encontrada **então**
      para todo $p \in P_s$ **faça**
        se $v_s^B(p) \notin C$ e $v_s^B(p)$ é adjacente, em B_s, a todos os vértices de C **então**
          Adicione $v_s^B(p)$ em C;
        fim se
      fim para
      Adicione na formulação a desigualdade (5.8) para C;
    fim se
  fim para
fim procedimento

5.3 Resultados computacionais

Realizamos experimentos computacionais para mostrar a relevância das desigualdades propostas na seção 5.1.2, avaliar o impacto do parâmetro w (tamanho das sequências de viagens dos árbitros) nos limitantes inferiores produzidos pela relaxação da nossa formulação, e comparar o desempenho do nosso algoritmo *branch-and-cut* (descrito na seção 5.3.3) com os outros métodos da literatura.

Nossa implementação foi feita em C++ usando a biblioteca em C (*Callable Library*) do pacote ILOG CPLEX versão 12.6.1, e compilada com o GCC 4.6.3. Todos os experimentos foram conduzidos em uma máquina equipada com um processador Intel Xeon X3430 2,40GHz e 8GB de memória RAM, executando o sistema operacional Linux Ubuntu 12.04.3.

Utilizamos em nossos experimentos as instâncias do *benchmark* do TUP [44] (veja a descrição desse *benchmark* na página 14). As instâncias com menos de 14 times foram desconsideradas pois elas são facilmente resolvidas pelos métodos atuais da literatura. Consideramos os valores de q_1 e q_2 usualmente adotados na literatura e também incluímos $q_1 = q_2 = 5$ para as instâncias com 26, 28, 30 e 32 times, que foram utilizados em [51].

Antes de continuar, existem dois aspectos importantes a serem destacados. O primeiro aspecto é referente às variáveis do modelo. Embora o número de variáveis em nossa formulação cresça exponencialmente em função de w, optamos por enumerar todas *a priori* e adicioná-las no modelo durante o início da execução através do algoritmo 6, ao invés de recorrer a um método de geração de colunas. A tabela 5.1 apresenta o número de variáveis na formulação para todas as instâncias e valores de w entre 2 e 10 (considerados nos experimentos). As instâncias com letras em seus nomes foram omitidas nesta tabela porque possuem os mesmos torneios que as instâncias originais correspondentes e, logo, resultam nas mesmas quantidades de variáveis. Os valores ausentes para um par instância

e w correspondem a modelos com mais de 5 milhões de variáveis, os quais vamos desconsiderar em nossos experimentos. Como, dependendo do caso, a enumeração poderia levar muito tempo, ela foi interrompida sempre que esta quantidade limite foi atingida e por isso nenhum valor foi reportado. O tempo gasto na enumeração das variáveis está incluído nos tempos de execução reportados nas próximas seções e não excede 15 segundos.

Algoritmo 6 Enumeração das variáveis da formulação.

```
1: procedimento ENUM-VARS
2:     para todo s ∈ S faça
3:         ENUM-VARS-REC(s, (), 0);                ▷ Gera todas as viagens em P_s
4:     fim para
5: fim procedimento
6:
7: procedimento ENUM-VARS-REC(seção s, rota simples p, inteiro ℓ);   ▷ Adiciona
   jogos no fim da rota simples p de tamanho ℓ até que ela atinja o tamanho da seção s
8:     se ℓ = w ou (s − 1)(w − 1) + 1 + ℓ > 4n − 2 então
9:         Adiciona a variável x_p na formulação;
10:    senão
11:        para todo jogo g na rodada (s − 1)(w − 1) + 1 + ℓ faça
12:            Faça p′ igual a p com g adicionado no fim;
13:            se p′ é uma rota simples factível então
14:                ENUM-VARS-REC(s, p′, ℓ + 1);
15:            fim se
16:        fim para
17:    fim se
18: fim procedimento
```

Inst.	q_1	q_2	Parâmetro w								
			2	3	4	5	6	7	8	9	10
14	7	3	875	1463	3124	6707	14480	26097	43858	92909	157212
14	6	3	875	1463	3124	6707	14480	30345	56849	140195	272418
14	5	3	875	1463	3124	6707	16354	37921	79866	224807	465183
16	8	4	1397	2961	5654	10844	18305	29823	37045	57076	97664
16	8	2	1397	3679	11624	38436	117394	331902	844815	2378813	7202046
16	7	3	1397	2961	7368	19089	43920	98326	204509	512763	1327904
16	7	2	1397	3679	11624	38436	117394	331902	973859	3031348	10546030
18	9	4	2081	5384	13994	34561	81585	171573	345990	621342	1211598
20	10	5	2962	9069	28332	72276	172373	393620	818194	1492658	2417177
22	11	5	4063	14405	53264	171979	535731	1497634	4036925		
24	12	6	5407	21810	97332	368098	1167827	3219784			
26	13	6	7009	31677	158375	717269	2615617				
26	5	5	7009	31677	158375	717269	3329528				
28	14	7	8909	44638	248893	1318194					
28	5	5	8909	44638	248893	1318194					
30	15	7	11124	61206	391728	2282757					
30	5	5	11124	61206	391728	2282757					
32	16	8	13673	81972	568954	3777946					
32	5	5	13673	81972	568954	3777946					

Tabela 5.1: Quantidade de variáveis no modelo proposto com w variando de 2 a 10.

O segundo aspecto relevante é com relação a forma como nós comparamos os nossos tempos de execução com aqueles apresentados em [42, 43, 51, 56], visto que os experimentos destas referências foram conduzidos em ambientes computacionais diferentes do nosso. Em vez de tentar estabelecer uma relação de velocidade confiável entre dois CPUs distintos (o que é uma tarefa extremamente difícil), para o propósito de avaliar nossos resultados é suficiente saber que a máquina que utilizamos é mais lenta que todas as outras, o que pode ser verificado, por exemplo, no seguinte *web site*: `www.cpubenchmark.net` (acessado em julho de 2015). Portanto, quando dizemos que "encontramos um limitante inferior melhor e X vezes mais rápido que [*referência*]", isso significa que o verdadeiro *speed-up* é ainda maior que X. Se a relaxação exata entre as CPUs fosse utilizada em nossas comparações, as conclusões só poderiam se tornar ainda mais favoráveis para nosso método. Com estas observações em mente, nós continuamos com as análises dos resultados.

5.3.1 O impacto das desigualdades válidas

Vamos começar pela avaliação de diferentes combinações das desigualdades válidas apresentadas na seção 5.1.2 com o intuito de mostrar o impacto nos tempos de execução e na qualidade dos limitantes inferiores. Resolvemos a relaxação linear da nossa formulação seis vezes para cada instância, cada vez utilizando um procedimento que consiste em um subconjunto ordenado distinto das rotinas de separação apresentadas na seção 5.2, escolhidos empiricamente da seguinte forma:

Sep1: Sep-Freq, Sep-Vst-Rec.

Sep2: Sep-Freq-NI, Sep-Vst-NI-Rec.

Sep3: Sep-Freq-NI-Mnl, Sep-Vst-NI-Rec.

Sep4: Sep-Freq-NI-Mnl, Sep-Vst-Frc-Rec.

Sep5: Sep-Freq-NI-Mnl, Sep-Vst-Frc-Rec, Sep-Clique-Sec-Adj.

Sep6: Sep-Freq-NI-Mnl, Sep-Vst-Frc-Rec, Sep-Clique-Sec-Adj, Sep-Clique-Sec-Igl.

A rotina Sep-Freq-NI corresponde à execução de Sep-Freq seguida por Sep-Freq-Inv, Sep-Freq-NI-Mnl corresponde à Sep-Freq-Mnl seguida por Sep-Freq-Inv-Mnl, e Sep-Vst-NI-Rec corresponde à Sep-Vst-Rec seguida por Sep-Vst-Inv-Rec.

Os procedimentos anteriores (combinações das rotinas de separação) foram empregados em um algoritmo de planos de corte da seguinte forma. Começamos com a otimização da relaxação linear da formulação que inclui somente (5.1) e (5.2). Em seguida, dado $i \in [1, 6]$, o procedimento Sep$i$ é aplicado na solução ótima encontrada, executando as suas rotinas de separação na ordem em que elas aparecem na definição do procedimento. Quando uma rotina dentro de Sep$i$ termina sua execução, a próxima rotina é executada somente se a anterior não adicionou nenhuma desigualdade no modelo. Caso contrário, Sep$i$ termina, o modelo é reotimizado (com o método *dual Simplex*), e Sep$i$ é executada novamente. Este processo é repetido até que não sejam encontradas mais

desigualdades violadas. Como Sep-Freq-NI, Sep-Freq-NI-Mnl e Sep-Vst-NI-Rec consistem de duas rotinas cada, elas recebem um tratamento especial: a sua segunda rotina é executada sempre, mesmo se a primeira adicionar desigualdades no modelo.

Para valores de $w \in [2, 4n-2]$ e $i \in [1, 6]$ fixos, vamos denotar por $\mathcal{M}_w^i$ o modelo de programação linear inteira formado por (5.1), (5.2), (5.4) e todas as desigualdades separadas por Sep$i$. A relaxação linear de $\mathcal{M}_w^i$, obtida pela substituição de (5.4) por $0 \leq x_p \leq 1$, é denotada por $\mathcal{M}_w^{Ri}$. Resolvemos cada relaxação linear $\mathcal{M}_w^{Ri}$ com nosso algoritmo de planos de corte e reportamos os limitantes inferiores e os tempos de execução (limitados em 3 horas) na tabela 5.2. Nestes experimentos, usamos os mesmos valores de w adotados nos experimentos com o algoritmo *branch-and-cut* (veja as explicações na seção 5.3.3).

Inst.	q_1	q_2	w	Limitante inferior						Tempo de execução (segundos)					
				$\mathcal{M}_w^{R1}$	$\mathcal{M}_w^{R2}$	$\mathcal{M}_w^{R3}$	$\mathcal{M}_w^{R4}$	$\mathcal{M}_w^{R5}$	$\mathcal{M}_w^{R6}$	$\mathcal{M}_w^{R1}$	$\mathcal{M}_w^{R2}$	$\mathcal{M}_w^{R3}$	$\mathcal{M}_w^{R4}$	$\mathcal{M}_w^{R5}$	$\mathcal{M}_w^{R6}$
14	7	3	4	151371,3	153680,4	153621,9	153621,9	154863,3	**155041,5**	0,28	0,74	0,46	0,46	1,78	3,07
14	6	3	4	150721,9	152670,6	152670,5	152670,5	153927,7	**154187,7**	0,21	0,34	0,25	0,24	0,86	1,48
14	5	3	4	149963,0	151792,7	151792,7	151792,7	152928,3	**153095,0**	0,11	0,20	0,12	0,12	0,26	0,44
14A	7	3	4	145138,1	147900,5	147872,8	147872,8	148706,3	**149081,4**	0,24	0,51	0,31	0,30	1,00	2,62
14A	6	3	4	144470,1	147039,7	147036,9	147036,9	147937,8	**148428,9**	0,14	0,30	0,20	0,20	0,52	1,59
14A	5	3	4	143973,3	146259,4	146259,4	146259,4	147065,8	**147421,0**	0,09	0,18	0,10	0,09	0,21	0,57
14B	7	3	4	145404,2	147263,1	147203,7	147201,9	148609,1	**148776,4**	0,23	0,77	0,46	0,42	1,57	2,49
14B	6	3	4	144787,5	146422,8	146422,8	146422,0	147785,0	**147959,1**	0,17	0,42	0,27	0,25	0,84	1,42
14B	5	3	4	144191,9	145529,6	145529,6	145529,6	146819,5	**147138,0**	0,09	0,20	0,10	0,10	0,27	0,55
14C	7	3	4	143650,8	146213,7	146039,7	146036,3	147415,5	**147686,6**	0,47	0,77	0,47	0,44	1,83	3,55
14C	6	3	4	142792,7	145005,7	145005,5	145005,5	146477,5	**146644,6**	0,15	0,37	0,26	0,26	1,05	2,02
14C	5	3	4	142150,3	144305,7	144305,8	144305,8	145776,0	**145953,5**	0,11	0,14	0,11	0,11	0,31	0,55
16	8	4	10	171712,1	176495,3	176495,7	176391,6	181095,3	**182696,7**	20,88	130,07	108,95	91,91	867,52	2877,17
16	8	2	8	147603,9	150068,3	150065,1	150055,5	**152853,0**	**152853,0**	106,53	1455,42	1184,43	978,76	10800,00	10800,00
16	7	3	4	148826,5	151856,2	151838,8	151838,8	157223,4	**157377,2**	0,86	2,44	1,37	1,35	10,26	13,34
16	7	2	4	141176,4	143872,7	143864,3	143864,3	147669,1	**147853,3**	0,65	1,64	1,22	1,20	7,84	10,14
16A	8	4	10	184979,7	188739,3	188739,9	188649,9	194032,1	**195581,3**	26,15	148,27	127,81	101,74	1389,07	3151,16
16A	8	2	8	157507,5	159312,5	159312,4	159311,4	**164893,2**	**164893,2**	151,92	626,02	515,39	501,40	10800,00	10800,00
16A	7	3	4	164939,1	167559,7	167539,7	167539,7	169767,1	**169970,9**	0,93	2,28	1,29	1,28	8,39	12,56
16A	7	2	4	155891,5	158068,2	158028,1	158028,1	160733,1	**160905,6**	0,60	1,32	0,92	0,91	6,57	9,73
16B	8	4	10	192098,3	197791,3	197790,3	197764,2	202768,2	**203952,1**	33,71	161,88	130,53	118,79	1262,69	2759,63
16B	8	2	8	160446,3	162696,2	162696,2	162696,2	**167241,3**	**167241,3**	139,06	282,94	253,69	253,35	10800,00	10800,00
16B	7	3	4	161769,3	165175,7	165144,9	165144,9	169537,3	**169617,5**	0,87	2,02	1,25	1,25	10,78	12,85
16B	7	2	4	155095,8	157774,9	157755,7	157755,7	162711,1	**162936,6**	0,81	1,82	1,25	1,24	7,70	12,24
16C	8	4	10	184697,1	190615,5	190615,7	190556,8	195863,7	**197220,7**	28,44	96,56	78,94	66,35	595,89	1511,01
16C	8	2	8	161294,0	164240,8	164241,2	164233,7	167196,7	**167341,3**	123,97	400,58	355,47	327,12	7996,15	10800,00
16C	7	3	4	164417,6	166988,9	166930,6	166930,6	169894,2	**169967,0**	0,96	2,57	1,54	1,53	10,13	12,91
16C	7	2	4	157474,4	159745,2	159733,8	159733,8	162813,5	**162903,3**	0,72	1,63	1,28	1,26	9,10	12,26
18	9	4	9	196674,4	201793,5	201795,9	201772,9	205489,9	**205743,8**	359,99	1294,34	1168,20	1086,19	7949,31	10800,00
20	10	5	10	238778,9	245908,8	**245960,6**	245907,2	245907,2	245907,2	3582,13	10800,00	10800,00	9134,04	10800,00	10800,00
22	11	5	7	260340,7	**266460,9**	266423,7	266423,5	266423,5	266423,5	3413,92	10800,00	9790,81	9764,14	10800,00	10800,00
24	12	6	7	292168,5	**297586,7**	297556,7	297556,7	297556,7	297556,7	10800,00	10800,00	10800,00	10800,00	10800,00	10800,00
26	13	6	6	327716,1	333517,8	**333678,9**	**333678,9**	**333678,9**	**333678,9**	8498,39	10800,00	10800,00	10800,00	10800,00	10800,00
26	5	5	4	307130,4	313683,9	313683,7	313683,7	323346,4	**323684,2**	45,25	121,77	76,61	76,51	823,52	1140,39
28	14	7	5	364692,1	374601,6	**374619,7**	**374619,7**	**374619,7**	**374619,7**	6116,04	10800,00	10800,00	10800,00	10800,00	10800,00
28	5	5	4	348811,5	355275,6	355275,5	355275,5	362132,7	**362585,2**	76,80	182,79	133,63	133,56	1140,83	1836,80
30	15	7	5	413149,5	421985,0	**422012,1**	**422012,1**	**422012,1**	**422012,1**	10119,78	10800,00	10800,00	10800,00	10800,00	10800,00
30	5	5	4	397669,4	404757,8	404757,8	404757,8	414403,9	**414865,9**	131,91	304,63	238,15	238,06	2305,95	3232,17
32	16	8	5	453106,0	**462944,9**	**462944,9**	**462944,9**	**462944,9**	**462944,9**	10800,00	10800,00	10800,00	10800,00	10800,00	10800,00
32	5	5	4	429802,6	442418,9	442418,9	442418,9	455453,6	**455885,7**	266,27	675,47	532,11	531,13	5654,81	7369,47

Tabela 5.2: Limitantes inferiores e tempos de execução para as relaxações lineares $\mathcal{M}_w^{Ri}$ com $i \in [1, 6]$. Os melhores limitantes inferiores aparecem em negrito.

Vamos comparar agora as diferentes relaxações lineares com base nos resultados da tabela 5.2. $\mathcal{M}_w^{R1}$ é composto por (5.1), (5.2) e (5.9), ao passo que $\mathcal{M}_w^{R2}$ é igual a $\mathcal{M}_w^{R1}$ mais (5.10). Os limitantes inferiores melhoram significativamente ao incluir (5.10), aumentando em torno de 1300 a 5900 milhas (0,9 a 3,2%) nas instâncias com até 16 times, e em torno de 5100 a 12600 milhas (1,8 a 3%) nas instâncias com mais de 16 times. Por outro lado,

os tempos de execução aumentam até 6,23 vezes em todas as instâncias com 16 times, à exceção de uma (a instância 16 com $q_1 = 8$ e $q_2 = 2$ leva 13,7 vezes mais tempo para ser resolvida), e até 3,6 vezes nas demais instâncias. $\mathcal{M}_w^{R3}$ difere de $\mathcal{M}_w^{R2}$ apenas com relação a (5.9) e (5.10). Em $\mathcal{M}_w^{R3}$, desconsideramos desigualdades em (5.9) e (5.10) que eliminam rotas não minimais que violam (2.4) ou (2.5), como visto na seção 5.2. $\mathcal{M}_w^{R3}$ é resolvido até duas vezes (1,37 vezes na média) mais rápido que $\mathcal{M}_w^{R2}$, ao passo que os limitantes inferiores providos pelo primeiro são no máximo 174 milhas (valor desprezível) menores que aqueles providos pelo último. Mesmo desprezando desigualdades, $\mathcal{M}_w^{R3}$ ainda obtém limitantes inferiores maiores que $\mathcal{M}_w^{R2}$ para algumas instâncias (por exemplo, na instância 16C com $q_1 = 8$ e $q_2 = 2$). Isto ocorre porque as rotinas de separação das desigualdades de caminho não são exatas, uma vez que as variáveis com valor menor que 0,001 são ignoradas. $\mathcal{M}_w^{R4}$ inclui todas as restrições em $\mathcal{M}_w^{R3}$, exceto pelas desigualdades em (5.9) e (5.10) que eliminam rotas que violam (2.3), substituídas pelas desigualdades em (5.11) que induzem a satisfação de (5.13). $\mathcal{M}_w^{R4}$ é resolvido ligeiramente (1,05 vezes na média) mais rápido que $\mathcal{M}_w^{R3}$, enquanto os limitantes inferiores produzidos pelo primeiro são no máximo 105 milhas (valor desprezível) menores que os limitantes inferiores do último. $\mathcal{M}_w^{R5}$ é igual a $\mathcal{M}_w^{R4}$ mais (5.7). Adicionar (5.7) leva a melhorias significativas nos limitantes inferiores, aumentando os valores em torno de 800 a 5500 milhas (0,6 a 3,5%) nas instâncias com no máximo 16 times, e em torno de 3700 a 13000 milhas (1,8 a 3%) nas instâncias com mais de 16 times, com algumas exceções: as instâncias com 20 ou mais times e $q_1 = n$ mantiveram os mesmos limitantes porque não foram adicionadas desigualdades (5.7) violadas dentro do tempo limite de 3 horas. Entretanto, em termos dos tempos de execução comparados com $\mathcal{M}_w^{R4}$, $\mathcal{M}_w^{R5}$ é resolvido até 4,2 vezes mais lentamente nas instâncias com 14 times, entre 11 e 42,6 vezes mais lentamente nas instâncias com 16 times com $q_1 = 8$ e $q_2 = 2$, e até 13,7 vezes mais lentamente nas demais instâncias. $\mathcal{M}_w^{R6}$ inclui todas as desigualdades em $\mathcal{M}_w^{R5}$ mais (5.8). Quase todos os melhores limitantes inferiores apresentados na tabela 5.2 foram providos por $\mathcal{M}_w^{R6}$, exceto quando a otimização deste modelo atingiu o limite de tempo, onde os resultados obtidos foram relativamente tão bons quanto aqueles de $\mathcal{M}_w^{R4}$ ou $\mathcal{M}_w^{R5}$, ou ligeiramente melhores ou piores que os resultados de $\mathcal{M}_w^{R2}$ ou $\mathcal{M}_w^{R3}$, visto que os dois últimos incluem diferentes conjuntos de desigualdades (5.9) e (5.10). Comparado com $\mathcal{M}_w^{R5}$, os limitantes inferiores de $\mathcal{M}_w^{R6}$ são até 1600 milhas (0,9%) maiores nas instância com 16 times com $q_1 = 8$ e $q_2 = 4$, e até 490 milhas (0,3%) nas demais instâncias. À primeira vista, como $\mathcal{M}_w^{R6}$ é resolvido até 3,32 vezes mais lentamente que $\mathcal{M}_w^{R5}$ e obtém pequenas melhorias nos limitantes inferiores, não haveria razões para utilizar as desigualdades (5.8). Contudo, experimentos preliminares com o *branch-and-cut* apresentado na seção 5.3.3 indicam que (5.8) contribui para uma redução significativa no tamanho da árvore de enumeração. Por este motivo, optamos por focar em $\mathcal{M}_w^{R6}$ em nossos experimentos subsequentes.

5.3.2 O impacto do parâmetro w

As tabelas 5.3 e 5.4 apresentam, respectivamente, os limitantes inferiores e os tempos de execução para $\mathcal{M}_w^{R6}$ com w variando entre 2 e 10. Como mencionado antes, resolvemos apenas as formulações com no máximo 5 milhões de variáveis (veja o número de variáveis

para cada instância e valor de w na tabela 5.1). O tempo de execução foi limitado em 3 horas. As tabelas 5.3 e 5.4 também incluem, no lado direito, os limitantes inferiores e os tempos de execução para as relaxações das melhores formulações de programação linear inteira para o TUP publicadas até o momento da escrita deste texto: a relaxação Lagrangiana da formulação baseada em fluxo em rede (FFR) proposta por [56], a relaxação linear da formulação baseada em partição de conjuntos (FPC) introduzida em [42] e [56], e a relaxação linear da formulação baseada em partição de conjuntos com cortes adicionais (FPCCA) estudada em [56]. Embora os limitantes inferiores reportados em [42] e [56] para FPC sejam os mesmos, os tempos de execução correspondentes são diferentes e aparecem, respectivamente, nas colunas FPC[42] e FPC[56] da tabela 5.4.

Inst.	q_1	q_2	$\mathcal{M}_2^{R6}$	$\mathcal{M}_3^{R6}$	$\mathcal{M}_4^{R6}$	$\mathcal{M}_5^{R6}$	$\mathcal{M}_6^{R6}$	$\mathcal{M}_7^{R6}$	$\mathcal{M}_8^{R6}$	$\mathcal{M}_9^{R6}$	$\mathcal{M}_{10}^{R6}$	FFR	FPC	FPCCA
14	7	3	152596,7	153805,9	155041,5	156029,2	156644,5	156728,2	**156799,3**	156718,5	156751,4	150934,5	156439,3	157016,3
14	6	3	152423,5	153401,3	154187,7	154715,3	154851,6	154925,8	154925,2	**154975,5**	154924,1	150909,7	154439,9	155252,6
14	5	3	151916,2	152707,7	153095,0	153173,7	153210,3	153196,0	**153250,3**	153230,9	153216,0	150621,1	152941,3	153486,5
14A	7	3	146776,2	147937,8	149081,4	149982,8	150408,5	**150548,7**	150469,7	150443,3	150471,1	145049,6	149992,7	150707,9
14A	6	3	146685,9	147345,6	148428,9	148655,2	148977,7	**149096,8**	148980,4	148962,8	149054,9	145018,1	148168,7	149283,7
14A	5	3	146362,1	146727,2	147421,0	147528,0	147704,9	**147728,1**	147592,5	147616,4	147622,2	144884,9	147097,5	147999,3
14B	7	3	146648,9	147822,9	148776,4	149706,9	150163,7	150390,9	150296,3	**150533,7**	150488,5	144053,6	149767,0	150699,7
14B	6	3	146531,2	147337,4	147959,1	148733,0	148842,6	148938,8	148880,3	148964,4	**149070,3**	144054,1	148243,9	149240,8
14B	5	3	146136,8	146801,8	147138,0	147369,5	147455,9	147366,1	**147617,9**	147463,4	147598,1	143866,8	146846,2	147784,8
14C	7	3	145208,9	146438,2	147686,6	148550,6	148934,8	149106,3	149076,5	**149179,0**	149138,1	143276,4	148613,2	149489,7
14C	6	3	145150,2	145900,2	146644,6	147075,3	147140,5	147137,9	147186,3	**147250,3**	147236,6	143233,8	146774,6	147644,9
14C	5	3	144847,6	145536,4	145953,5	146105,9	146161,2	146210,1	**146230,1**	146212,4	146145,2	143149,5	145794,4	146597,9
16	8	4	157573,8	163238,6	167294,2	174986,5	178375,3	179509,2	181402,9	181063,4	**182696,7**	152507,6	184187,6	185056,8
16	8	2	144224,4	146585,4	148506,6	150717,2	152104,1	152027,0	**152853,0**	149662,8		142134,8	155045,2	155712,5
16	7	3	153900,5	155923,5	157377,2	158188,3	158460,0	158466,9	**158635,7**	158626,9	158222,0	150532,2	158257,4	158883,4
16	7	2	144176,9	146475,8	147853,3	148369,0	148534,3	148574,7	**148629,7**	147865,3		142145,2	148341,8	148980,7
16A	8	4	170424,0	176358,3	179870,1	187519,0	190972,0	192643,0	194233,8	194015,0	**195581,3**	164945,9	198969,7	200007,6
16A	8	2	157765,9	159929,7	161300,4	164116,1	164511,7	164512,8	**164893,2**	163339,0		155641,5	166575,5	167360,0
16A	7	3	166719,6	168505,7	169970,9	170615,1	170880,4	171103,7	**171251,7**	171205,0	170763,1	162700,0	170575,1	171426,6
16A	7	2	157682,6	159756,9	160905,6	161525,5	161617,4	161715,0	**161731,9**	160634,4		155963,5	161571,2	161975,1
16B	8	4	170001,6	177606,1	182450,8	192774,4	196594,2	198726,5	201609,1	202655,0	**203952,1**	165008,4	207505,4	208496,8
16B	8	2	157967,4	161411,3	163379,5	166207,6	166892,4	166889,7	**167241,3**	165511,6		156402,2	169363,4	170040,3
16B	7	3	165010,3	167814,3	169617,5	170351,2	170837,9	170993,9	**171110,0**	171083,2	170860,4	162073,7	170632,5	171280,6
16B	7	2	157936,9	161267,6	162936,6	163536,6	163812,7	163814,9	**163894,9**	162738,2		156442,1	163539,7	164160,9
16C	8	4	171801,3	176480,1	180684,9	187282,9	191314,0	192331,8	195932,0	195866,4	**197220,7**	167256,9	200682,6	201107,5
16C	8	2	160069,3	161761,6	163307,9	165357,6	166335,5	166488,9	**167341,3**	165813,3		158947,2	168783,6	169270,9
16C	7	3	166754,7	168194,5	169967,0	170841,1	171373,7	171366,7	**171596,4**	171454,5	171225,4	164380,8	171216,0	171827,6
16C	7	2	160006,2	161596,2	162903,3	163393,5	163858,2	**163907,0**	163771,9	163122,1		158906,2	163850,8	164182,8
18	9	4	187132,4	192865,6	196085,7	200213,7	201992,9	203102,9	203813,8	**205743,8**	204027,0	181430,7	212121,6	212793,6
20	10	5	220179,1	229339,6	234897,0	237819,0	243686,7	244967,0	242752,5	244388,4	**245907,2**	213513,3		
22	11	5	248369,7	257481,4	261951,9	263768,8	264970,6	**266423,5**	265699,9			241909,2		
24	12	6	277716,5	286579,0	293662,7	295812,5	295828,3	**297556,7**				270662,0		
26	13	6	317247,7	325892,7	331844,8	331456,4	**333678,9**					310366,9		
26	5	5	314581,9	321305,4	323684,2	**323843,5**	323070,6							
28	14	7	355413,1	366501,5	371985,2	**374619,7**						348059,2		
28	5	5	352797,1	360574,9	**362585,2**	362571,0								
30	15	7	406137,0	417363,9	420846,9	**422012,1**						396222,1		
30	5	5	404198,1	412133,4	**414865,9**	413756,4								
32	16	8	439408,5	460082,6	459192,4	**462944,9**						427436,2		
32	5	5	435769,2	452052,4	**455885,7**	450956,9								

Tabela 5.3: Limitantes inferiores obtidos com $\mathcal{M}_w^{R6}$ para $w \in [2, 10]$ (os melhores valores aparecem em negrito), e com as relaxações das melhores formulações da literatura (FFR, FPC e FPCCA).

As tabelas 5.3 e 5.4 mostram que, embora os tempos de execução de $\mathcal{M}_w^{R6}$ aumentem consideravelmente à medida que w cresce, os limitantes inferiores nem sempre melhoram e, em alguns casos, podem até piorar. Por exemplo, considere a instância 14 com $q_1 = 7$ e $q_2 = 3$. O limitante inferior com $w = 6$ leva 24,54 segundos para ser calculado e possui no máximo 155 milhas a menos que os limitantes inferiores obtidos com $w > 6$, que levam

Inst.	q_1	q_2	$\mathcal{M}_2^{R6}$	$\mathcal{M}_3^{R6}$	$\mathcal{M}_4^{R6}$	$\mathcal{M}_5^{R6}$	$\mathcal{M}_6^{R6}$	$\mathcal{M}_7^{R6}$	$\mathcal{M}_8^{R6}$	$\mathcal{M}_9^{R6}$	$\mathcal{M}_{10}^{R6}$	FFR	FPC[56]	FPC[42]	FPCCA
14	7	3	0,14	0,51	3,07	10,42	24,54	43,15	87,16	258,95	630,87	0,59	526,56	42,10	742,86
14	6	3	0,07	0,29	1,48	3,28	11,65	31,77	90,46	551,29	1741,88	0,54	108,70	40,80	443,61
14	5	3	0,05	0,16	0,44	1,05	3,74	17,80	57,12	587,51	1998,03	0,52	40,50	50,30	216,54
14A	7	3	0,12	0,63	2,62	11,13	27,78	53,05	91,42	273,05	881,75	0,58	107,57	40,70	638,13
14A	6	3	0,08	0,40	1,59	4,27	11,18	47,92	113,57	527,70	2090,22	0,55	77,54	45,50	450,26
14A	5	3	0,06	0,26	0,57	1,97	6,80	28,70	134,03	792,61	4513,95	0,54	42,13	47,80	220,36
14B	7	3	0,14	0,68	2,49	8,73	22,14	58,06	84,32	364,78	879,30	0,60	377,83	43,50	842,78
14B	6	3	0,12	0,45	1,42	4,36	11,67	35,78	126,73	401,60	1939,73	0,54	52,16	49,60	462,99
14B	5	3	0,07	0,29	0,55	1,78	6,82	29,42	160,98	1035,51	5777,11	0,53	31,68	55,70	480,79
14C	7	3	0,13	0,72	3,55	11,52	22,21	63,54	103,36	297,25	879,67	0,56	442,33	44,60	806,03
14C	6	3	0,12	0,30	2,02	3,72	9,11	27,48	72,84	370,68	1534,01	0,56	326,59	47,70	796,43
14C	5	3	0,09	0,18	0,55	1,39	4,49	28,97	111,04	818,00	5163,33	0,52	54,40	49,50	396,86
16	8	4	0,81	14,29	42,07	201,03	343,69	431,65	1032,18	2385,00	2877,17	0,86	457,10	172,00	771,61
16	8	2	0,18	1,73	14,24	272,94	1843,58	4820,36	10800,00	10800,00		0,60	50247,89	7092,00	59186,31
16	7	3	0,34	2,48	13,34	59,54	154,72	402,81	691,82	3445,89	10800,00	0,71	2207,89	10500,00	2959,44
16	7	2	0,15	1,58	10,14	114,91	515,94	1902,23	6051,61	10800,00		0,69	2598,89	10102,00	5113,53
16A	8	4	1,20	15,51	32,32	198,92	386,43	551,95	781,06	1816,88	3151,16	0,81	373,20	172,00	923,43
16A	8	2	0,15	1,58	12,42	352,65	1400,26	4330,80	10800,00	10800,00		0,58	14548,00	5403,00	17380,10
16A	7	3	0,44	2,91	12,56	61,85	172,06	491,18	1381,38	5342,18	10800,00	0,58	3266,34	371,00	4303,05
16A	7	2	0,13	1,21	9,73	93,21	348,10	1415,13	5790,52	10800,00		0,56	3918,92	7476,00	6044,11
16B	8	4	1,01	11,28	39,69	179,43	349,11	467,83	782,94	2449,49	2759,63	0,81	342,27	202,00	771,07
16B	8	2	0,16	1,67	15,07	287,66	1459,32	4453,18	10800,00	10800,00		0,74	42129,13	5162,00	53974,07
16B	7	3	0,38	2,38	12,85	58,67	141,48	389,96	1051,29	4142,73	10800,00	0,73	3236,07	880,00	4053,36
16B	7	2	0,15	1,33	12,24	110,03	515,78	1699,03	7627,61	10800,00		0,72	3077,02	9021,20	5781,35
16C	8	4	0,99	10,14	40,07	142,23	236,83	246,88	663,24	1037,80	1511,01	0,86	201,13	234,00	586,44
16C	8	2	0,19	1,76	14,95	161,89	1003,26	2501,33	10800,00	10800,00		0,74	13634,98	7380,00	22101,70
16C	7	3	0,33	2,76	12,91	61,77	164,26	411,04	1012,05	3788,13	10800,00	0,79	851,66	449,00	1658,40
16C	7	2	0,15	1,45	12,26	68,82	353,68	1518,86	10800,00	10800,00		0,71	3319,74	10578,00	4790,26
18	9	4	1,65	32,74	165,18	686,79	1856,79	2428,94	6430,24	10800,00	10800,00	1,07	17834,91		22980,33
20	10	5	3,61	109,79	955,08	2038,08	4351,67	7870,05	10800,00	10800,00	10800,00	1,53			
22	11	5	4,52	197,52	2233,87	7476,42	10800,00	10800,00	10800,00			1,95			
24	12	6	11,68	387,45	10800,00	10800,00	10800,00	10800,00				2,90			
26	13	6	11,92	621,74	10800,00	10800,00	10800,00					3,75			
26	5	5	3,54	100,85	1140,39	5094,29	10800,00								
28	14	7	16,98	973,06	10800,00	10800,00						4,29			
28	5	5	5,08	134,21	1836,80	9919,03									
30	15	7	23,43	1588,45	10800,00	10800,00						4,16			
30	5	5	6,87	211,68	3232,17	10800,00									
32	16	8	34,30	3920,92	10800,00	10800,00						7,09			
32	5	5	9,60	452,96	7369,47	10800,00									

Tabela 5.4: Tempos de execução (em segundos) para $\mathcal{M}_w^{R6}$ com $w \in [2, 10]$, e para as relaxações das melhores formulações da literatura (FFR, FPC e FPCCA).

entre 43,15 e 630,87 segundos para serem calculados. Além disso, o limitante inferior encontrado para esta instância com $w = 8$ é maior que aqueles encontrados com $w > 8$. Uma possível causa para a deterioração dos limitantes inferiores de $\mathcal{M}_w^{R6}$ à medida que w cresce é o aumento do tamanho do modelo. Como as relaxações lineares de modelos grandes levam muito tempo para serem otimizadas e nosso tempo de execução é limitado, poucas iterações do método de planos de corte são executadas. Com poucos cortes, é esperado que os limitantes inferiores percam qualidade. Este comportamento indica que devemos ter cuidado ao escolher o valor de w que será utilizado em nosso algoritmo *branch-and-cut*.

Vamos comparar agora $\mathcal{M}_w^{R6}$ com FFR, FPC e FPCCA. As variáveis de FFR são equivalentes às variáveis de $\mathcal{M}_2^{R6}$, contudo $\mathcal{M}_2^{R6}$ inclui desigualdades válidas adicionais. Nas 28 instâncias com até 16 times, os limitantes inferiores produzidos por $\mathcal{M}_2^{R6}$ são entre 1100 e 5478 milhas maiores que aqueles produzidos por FFR. Além disso, 25 dos 28 limitantes melhorados tomaram menos tempo para serem calculados com $\mathcal{M}_2^{R6}$ do que com FFR. Nas instâncias com 18 ou mais times, embora $\mathcal{M}_2^{R6}$ seja resolvido até 5,6 vezes mais lentamente que FFR, os limitantes inferiores obtidos por $\mathcal{M}_2^{R6}$ são entre 5701 e

11972 milhas maiores que aqueles encontrados por FFR. As variáveis de FPC e FPCCA representam as rotas completas que satisfazem (2.3)–(2.5) e, logo, são equivalentes às variáveis de $\mathcal{M}^{R6}_{4n-2}$. Note que $4n-2$ é um valor de w muito maior que os valores que consideramos para este parâmetro em nossos experimentos. Todavia, os limitantes inferiores obtidos por $\mathcal{M}^{R6}_{6}$ nas instâncias com 16 times e $q_1 = 7$ e em todas as instâncias com 14 times são melhores que aqueles obtidos por FPC. Além disso, nestas instâncias, $\mathcal{M}^{R6}_{6}$ é resolvido entre 3,9 e 35,8 vezes mais rápido que FPC em [56], e entre 1,5 e 67,9 vezes mais rápido que FPC em [42]. Os limitantes inferiores de $\mathcal{M}^{R6}_{6}$ são no máximo 555 milhas menores que aqueles obtidos por FPCCA nestas mesmas instâncias, ao passo que levaram entre 9,9 e 88,4 vezes menos tempo para serem calculados. Apesar destes bons resultados, $\mathcal{M}^{R6}_{w}$ não vai bem nas instâncias com 16 times e $q_1 = n = 8$. Os melhores limitantes inferiores de $\mathcal{M}^{R6}_{w}$ nestas instâncias são entre 1442 e 3553 milhas menores que aqueles obtidos por FPC, e entre 1929 e 4544 milhas menores que aqueles obtidos por FPCCA, e são calculados tomando entre 1,5 e 18,3 vezes mais tempo que FPC em [42]. Acreditamos que FPC e FPCCA tendem a superar $\mathcal{M}^{R6}_{w}$, à medida que os valores de q_1 e q_2 crescem, por causa do aumento do número de rotas proibidas eliminadas em FPC e FPCCA que ainda podem fazer parte de soluções fracionárias de $\mathcal{M}^{R6}_{w}$, o que leva a limitantes inferiores piores. Na instância 18, embora os limitantes inferiores de FPC e FPCCA obtidos em [56] sejam 6378 milhas (3,1%) e 7050 milhas (3,4%) menores que os melhores limitantes inferiores de $\mathcal{M}^{R6}_{w}$, eles foram obtidos em 4 horas e 57 minutos e 6 horas e 23 minutos, respectivamente.

A vantagem de $\mathcal{M}^{R6}_{w}$ fica mais perceptível à medida que o tamanho do problema aumenta, o que é constatado nas tabelas 5.3 e 5.4. Como FPC e FPCCA possuem um número exponencial de variáveis e a resolução do problema de *pricing* correspondente toma um tempo relativamente considerável, as relaxações destas formulações consomem muito tempo para serem otimizadas nas instâncias com mais de 18 times. Além de atingir bons resultados nas instâncias com 16 times e $q_1 = 7$ e em todas as instâncias com 14 times, $\mathcal{M}^{R6}_{w}$ não só pode ser resolvido no limite de 3 horas para todas as instâncias com mais de 18 times, mas também produz os melhores limitantes inferiores conhecidos até o momento para estas instâncias.

5.3.3 Algoritmo *branch-and-cut*

Visto o bom desempenho de $\mathcal{M}^{R6}_{w}$ nos resultados anteriores, desenvolvemos um algoritmo *branch-and-cut* utilizando o CPLEX para resolver este modelo. Como (5.7)–(5.10) e (5.13) são exponenciais em número, iniciamos a otimização com um modelo contendo somente (5.1), (5.2) e (5.4), e adicionamos (5.7)–(5.10) e (5.13) durante a enumeração na medida em que estas desigualdades se tornarem violadas. Em cada nó da árvore de enumeração, o CPLEX chama através de *callbacks* o procedimento Sep6 para efetuar a separação.

Nós utilizamos os seguintes ajustes de parâmetros no algoritmo *branch-and-cut* do CPLEX. Experimentos preliminares mostraram que as heurísticas primais do CPLEX não obtêm boas soluções para $\mathcal{M}^{6}_{w}$. Portanto, vamos focar em encontrar bons limitantes inferiores e soluções ótimas ajustando o parâmetro "*MIP emphasis*" para "*best bound*" e desabilitando as heurísticas primais. Modificamos também o parâmetro "*MIP probing*

level" para forçar a execução de um *probing* moderado nas variáveis, já que efetuar um *probing* agressivo consome muito tempo e não melhora os resultados. Em particular, notamos que o algoritmo levou muito tempo para escolher uma variável para ramificar (*branching*) nas instâncias cuja otimização da relaxação linear tomou muito tempo (mais de 1000 segundos, veja a coluna $\mathcal{M}_w^{R6}$ na tabela 5.2 para identificar estas instâncias.). Por isso, somente nestas instâncias, ajustamos o algoritmo para escolher como variável para ramificar aquela cujo valor é o mais distante de um inteiro. Isto acelera o processo de ramificação e aumenta o número de nós explorados dentro do tempo limite, o que leva a melhores resultados. Por fim, desabilitamos todas as rotinas de geração de cortes do CPLEX a fim de avaliar apenas o impacto dos nossos cortes.

Conduzimos experimentos preliminares para determinar quais valores de w serão utilizados para cada instância do *benchmark* baseado na relação de custo-benefício entre velocidade e qualidade dos limitantes identificada na seção 5.3.2. Nossos testes indicaram que o *branch-and-cut* obtém melhores resultados ao ajustar $w = 4$ para todas as instâncias com $q_1 < n$ e para as instâncias com 14 times e $q_1 = n$. Os limitantes inferiores providos por $\mathcal{M}_4^{R6}$ para estas instâncias não estão tão longe dos melhores apresentados na tabela 5.3 e são calculados rapidamente, permitindo a enumeração de muito mais nós. Para as demais instâncias com $q_1 = n$, compensa utilizar as relaxações que tomam mais tempo pois elas obtêm limitantes inferiores consideravelmente melhores. Portanto, estas instâncias são resolvidas com os valores de w que atingiram os melhores limitantes inferiores (valores em negrito) na tabela 5.3.

Executamos nosso algoritmo *branch-and-cut* com os limites de tempo de 3 e 24 horas a fim de efetuar comparações justas entre nossos resultados e aqueles publicados na literatura. Reportamos os melhores limitantes inferiores obtidos dentro de cada limite de tempo na tabela 5.5. A coluna "Limitante inferior" contém o melhor limitante inferior encontrado, a coluna "Iterações" a quantidade de iterações executadas pelo método *Simplex* e a coluna "Cortes" o total de desigualdades violadas adicionadas no modelo por Sep6. As instâncias com limitantes inferiores apresentados em negrito e marcados com um "*" foram resolvidas na otimalidade pelo nosso algoritmo. Estes são os únicos limitantes superiores encontrados dentro do limite de tempo.

Como visto na tabela 5.5, todas as instâncias com 14 times, $q_1 = 5$ e $q_2 = 3$, exceto uma, foram resolvidas na otimalidade no limite de 3 horas. A única formulação matemática publicada capaz de obter soluções comprovadamente ótimas para instâncias com mais de 12 times é a FPCCA resolvida pelo *branch-and-price-and-cut* apresentado em [56]. Este método encontrou soluções ótimas para as instâncias 14 e 14A com $q_1 = 5$ e $q_2 = 3$ após 34 horas e 45 minutos e 11 horas e 24 minutos de execução, respectivamente, ao passo que nós resolvemos todas as instâncias com 14 times, $q_1 = 5$ e $q_2 = 3$ em não mais que 3 horas e 10 minutos (14, 14A e 14B levaram 31, 9 e 17 minutos, respectivamente). Os resultados revelam que grande parte da melhoria dos limitantes inferiores é atingida pelo *branch-and-cut* dentro das primeiras 3 horas de computação. Estender o limite de tempo para 24 horas somente produz um acréscimo médio de 1368 milhas nos limitantes inferiores, embora o ganho varie bastante (desvio padrão de 1413 milhas) entre uma instância e outra.

A fim de complementar as informações reportadas na tabela 5.5, apresentamos o per-

Inst.	q_1	q_2	w	Tempo limite									
				3 horas					24 horas				
				Limitante inferior	Tempo	Iterações	Nós	Cortes	Limitante inferior	Tempo	Iterações	Nós	Cortes
14	7	3	4	158578,5	10800,00	6578554	5462	16004	159271,8	86400,00	34265950	25830	26417
14	6	3	4	157469,3	10800,00	8680509	12259	14585	158037,6	86400,00	41153400	52618	23008
14	5	3	4	**154962,0***	1823,44	1951142	10980	7622					
14A	7	3	4	152537,1	10800,00	7231231	7036	15132	153257,5	86400,00	36554874	33291	25296
14A	6	3	4	151611,1	10800,00	10066365	15628	12439	152169,5	86400,00	50868260	68271	20215
14A	5	3	4	**149331,0***	524,79	836360	3300	4342					
14B	7	3	4	152647,1	10800,00	7157927	5776	15660	153191,1	86400,00	34391926	23233	27720
14B	6	3	4	151360,5	10800,00	9863357	14392	12906	151821,9	86400,00	45898826	58131	22350
14B	5	3	4	**149455,0***	1003,70	1668752	8887	5211					
14C	7	3	4	151129,1	10800,00	6314167	4764	16719	151791,0	86400,00	30767570	20898	28766
14C	6	3	4	149820,4	10800,00	9316255	13735	13601	150286,6	86400,00	44207534	54463	22375
14C	5	3	4	148333,2	10800,00	8106602	27195	13572	**148349,0***	11457,49	8330667	29379	13705
16	8	4	10	183386,2	10800,00	1093933	8	9802	185936,7	86400,00	7361031	54	27141
16	8	2	8	152569,6	10800,00	206239	1	5230	153725,0	86400,00	1214144	13	9179
16	7	3	4	159841,1	10800,00	3261959	1649	13424	160664,0	86400,00	28910558	13240	28292
16	7	2	4	149560,8	10800,00	2634818	1573	13536	149988,5	86400,00	23603021	12705	30384
16A	8	4	10	196183,3	10800,00	1145764	12	13048	198330,5	86400,00	7415309	75	27001
16A	8	2	8	164625,8	10800,00	207790	1	4917	165915,3	86400,00	1256325	12	9541
16A	7	3	4	173028,2	10800,00	3065182	1052	12339	174226,3	86400,00	28361114	9054	28039
16A	7	2	4	162675,1	10800,00	2817934	1696	13326	163052,0	86400,00	26985443	14253	27689
16B	8	4	10	205073,4	10800,00	1126908	8	12883	207781,1	86400,00	7296067	61	31270
16B	8	2	8	167241,6	10800,00	190363	1	4021	168223,9	86400,00	1991551	7	7452
16B	7	3	4	172131,7	10800,00	2971901	1235	13041	173178,0	86400,00	29106668	10769	27111
16B	7	2	4	164978,2	10800,00	3988299	2986	13180	165581,1	86400,00	33172881	24074	25088
16C	8	4	10	198274,6	10800,00	1190729	11	11403	202369,4	86400,00	7259649	76	22285
16C	8	2	8	167339,8	10800,00	178530	1	4245	167530,7	86400,00	1124098	2	4983
16C	7	3	4	172377,7	10800,00	2787489	1072	13455	173273,9	86400,00	26730153	9351	29042
16C	7	2	4	164531,3	10800,00	3519161	1998	13513	165125,2	86400,00	27695874	13998	28036
18	9	4	9	205781,9	10800,00	257650	1	9386	206759,4	86400,00	1681072	16	12350
20	10	5	10	245897,4	10800,00	93440	1	9842	250372,6	86400,00	572624	1	17704
22	11	5	7	266415,6	10800,00	189486	1	20364	269735,5	86400,00	1062045	5	33425
24	12	6	7	297898,6	10800,00	107241	1	19403	301441,0	86400,00	671652	1	38858
26	13	6	6	333504,9	10800,00	127822	1	19216	336854,4	86400,00	730203	1	34010
26	5	5	4	324387,1	10800,00	1409902	247	11564	324753,2	86400,00	11158237	2770	22082
28	14	7	5	374630,6	10800,00	220783	1	26780	377356,2	86400,00	1149468	1	36817
28	5	5	4	363072,1	10800,00	916593	91	10571	363541,4	86400,00	7865382	1422	20455
30	15	7	5	422026,0	10800,00	144276	1	20906	424537,6	86400,00	749184	1	34822
30	5	5	4	415296,4	10800,00	614318	43	9642	415747,5	86400,00	5650380	1063	18417
32	16	8	5	462894,6	10800,00	99418	1	14476	468803,5	86400,00	590596	1	32946
32	5	5	4	455836,4	10800,00	376627	1	9909	456685,9	86400,00	3843703	492	17838

Tabela 5.5: Resultados do algoritmo *branch-and-cut*. Os tempos reportados estão em segundos. Os asteriscos indicam valores ótimos.

centual médio de cortes separados para cada família de desigualdades (seguido por $\pm$ e seu respectivo desvio padrão). Com os tempos de execução limitados em 3 horas, as quantidades médias foram: 18,5% $\pm$ 14,3% de (5.7), 12,2% $\pm$ 12,8% de (5.8), 30,6% $\pm$ 11,6% de (5.9), 36,3% $\pm$ 15,5% de (5.10) e 2,4% $\pm$ 8,4% de (5.13). Com os tempos de execução limitados em 24 horas, os valores se mantiveram similares: 21,2% $\pm$ 16,9% de (5.7), 11,9% $\pm$ 9,6% de (5.8), 30,2% $\pm$ 11,1% de (5.9), 35,4% $\pm$ 14,5% de (5.10) e 1,3% $\pm$ 7,1% de (5.13).

Comparamos na tabela 5.6, observando os limites de tempo de execução, os limitantes inferiores encontrados pelo nosso algoritmo *branch-and-cut* (BC) com os melhores limitantes inferiores encontrados na literatura, obtidos pelos seguintes métodos: método de decomposição (MD) [51], *branch-and-price* (BP) [42], *branch-and-bound* (BB) [56], *branch-and-price-and-cut* (BPC) [56], e *branch-and-bound* com limitantes inferiores base-

ados em decomposição (BB-LID) [43]. Em [51], os resultados de MD foram reportados para dois limites de tempo distintos: até 3 horas (o qual chamaremos de MD3) e mais de 3 horas (o qual chamaremos de MD+). A execução dos métodos BB e BP foi limitada em 3 horas, e dos métodos BPC e BB-LID em 48 horas. Diferentemente dos outros métodos, BB-LID encontrou várias soluções ótimas e provou a infactibilidade de algumas instâncias antes de atingir o tempo limite. Portanto, para estes resultados, vamos incluir os tempos de execução correspondentes entre parênteses na última coluna da tabela 5.6. Comparamos os limitantes inferiores encontrados por BC dentro de 3 horas com aqueles obtidos por BB, BP, MD3 e BB-LID dentro de 3 horas, e aqueles encontrados por BC dentro de 24 horas com aqueles obtidos por MD+, BPC e BB-LID em mais de 3 horas. Assim como antes, os limitantes inferiores marcados com "*" são valores ótimos. Um limitante inferior aparece em negrito nesta tabela se nenhum outro melhor foi encontrado dentro do mesmo limite de tempo em que ele foi obtido.

Os resultados na tabela 5.6 sugerem que BB-LID se destaca mais nas instâncias menores, ao passo que BC lida melhor com as instâncias maiores. Para ver isso, vamos dividir nossa análise em dois conjuntos complementares de instâncias. O primeiro (MENORES) é composto pelas instâncias com até 18 times, enquanto o segundo (MAIORES) contém as demais instâncias (com 20 ou mais times).

Para 20 das 29 instâncias do grupo MENORES, os limitantes inferiores de BB-LID são estritamente maiores que aqueles encontrados pelos demais métodos. BB-LID resolve 19 instâncias na otimalidade e produz 4 provas de infactibilidade. Os métodos BPC e BC resolvem apenas 2 e 4 instâncias na otimalidade, respectivamente.

Agora vamos focar nas 11 instâncias do grupo MAIORES, cujos tamanhos se tornam mais próximos do número atual de times na Liga Profissional de Beisebol (30 times). Nem todos os métodos conseguem lidar com instâncias grandes e, portanto, várias delas possuem poucos resultados reportados. Estão disponíveis na literatura para as instâncias do grupo MAIORES apenas 7, 1, 4 e 7 resultados obtidos pelos métodos BB, MD3, MD+ e BB-LID, respectivamente. Estes resultados, junto com aqueles encontrados por BC, aparecem nas últimas 11 linhas da tabela 5.6. Começamos com os resultados obtidos em 3 horas de computação. Sob este limite, os resultados de BC, BB e MD3 são: BC produz limitantes inferiores para todas as 11 instâncias, BB para 7, e MD3 para apenas uma instância. O método BC é nitidamente o vencedor já que ele computa os melhores limitantes inferiores para todas as instâncias do grupo MAIORES. A melhoria média/máxima/mínima nos limitantes inferiores é de 23548,4 / 32379,7 / 11571,4 milhas, com desvio padrão de 6718,9 milhas. A vantagem de BC sobre os outros métodos nas instâncias do grupo MAIORES também é confirmada na análise dos resultados obtidos com mais de 3 horas de computação. Nesta análise, são considerados os métodos MD+ e BB-LID, que podem ser vistos como complementares em relação ao grupo MAIORES no sentido que existem resultados reportados por exatamente um deles para cada instância (7 por BB-LID e 4 por MD+). Novamente, os limitantes inferiores de BC são os melhores nas 11 instâncias. A melhoria média/máxima/mínima nos limitantes inferiores é de 38248,0 / 99108,5 / 2644,5 milhas, com desvio padrão de 32671,3 milhas. Além disso, note que os limitantes inferiores de BC obtidos em 3 horas de execução permanecem os maiores mesmo quando comparados com os limitantes inferiores dos outros métodos exe-

Inst.	q_1	q_2	Tempo limite / Método								
			3 horas				24 horas	> 3 horas	48 horas		
			BC	BB	BP	MD3	BC	MD+	BPC	BB-LID	
14	7	3	158578,5	154175,6	157812,8	156536,0	159271,8	159797,0	158900,2	**164440,0***	(228,6)
14	6	3	157469,3	154036,7	155570,4	156551,0	158037,6	156551,0	157083,4	**158875,0***	(51,6)
14	5	3	154962,0*	153318,8	153759,6	153066,0		153066,0	154962,0*	**154962,0***	(130,2)
14A	7	3	152537,1	147866,4	151243,5	151406,0	153257,5	153199,0	152635,7	**158760,0***	(123,0)
14A	6	3	151611,1	147773,1	149285,4	150998,0	152169,5	150998,0	151043,2	**152981,0***	(30,0)
14A	5	3	149331,0*	147358,3	147966,4	148299,0		148299,0	149331,0*	**149331,0***	(67,2)
14B	7	3	152647,1	147159,6	151165,8	149910,0	153191,1	151059,0	152517,6	**157884,0***	(241,2)
14B	6	3	151360,5	147031,5	149208,6	149267,0	151821,9	149267,0	150941,3	**152740,0***	(103,2)
14B	5	3	149455,0*	146606,1	147638,3	147534,0		147534,0	149311,6	**149455,0***	(63,0)
14C	7	3	151129,1	146104,6	150101,6	151122,0	151791,0	151581,0	150925,9	**154913,0***	(45,6)
14C	6	3	149820,4	145982,2	147820,0	148728,0	150286,6	148728,0	148986,5	**150858,0***	(100,2)
14C	5	3	148333,2	145598,1	146622,1	146764,0	148349,0*	146764,0	147902,9	**148349,0***	(764,4)
16	8	4	183386,2	156206,4	**193457,1**	168847,0	185936,7	185939,0	191458,0	**inf.**	(13977,6)
16	8	2	152569,6	145829,7	**155045,2**	151481,0	153725,0	151481,0	**156088,1**	145531,0	
16	7	3	**159841,1**	153649,4	158586,0	155707,0	160664,0	158480,0	160161,3	**165765,0***	(24296,4)
16	7	2	**149560,8**	145787,0	148341,8	147138,0	149988,5	147138,0	149488,0	**150433,0***	(66118,8)
16A	8	4	196183,3	168882,5	**200648,5**	185119,0	198330,5	185119,0	206141,2	**inf.**	(13549,2)
16A	8	2	164625,8	158645,6	**166624,1**	162788,0	165915,3	162788,0	**168274,4**	160739,0	
16A	7	3	**173028,2**	166459,3	172420,1	170342,0	174226,3	172964,0	172471,4	**178511,0***	(15101,4)
16A	7	2	**162675,1**	158621,8	161571,2	161640,0	163052,0	161640,0	162621,7	**163709,0***	(57922,2)
16B	8	4	205073,4	169684,4	**209346,5**	188195,0	207781,1	208418,0	215520,6	**inf.**	(13764,6)
16B	8	2	167241,6	159525,2	**170092,6**	167768,0	168223,9	167768,0	**170384,4**	165737,0	
16B	7	3	**172131,7**	165753,2	172058,0	170940,0	173178,0	173023,0	172695,9	**180204,0***	(136216,8)
16B	7	2	**164978,2**	159538,6	163649,6	164012,0	165581,1	164012,0	164816,0	**167190,0***	(138118,8)
16C	8	4	198274,6	170370,6	**205643,8**	179213,0	202369,4	188561,0	206368,8	**inf.**	(14216,4)
16C	8	2	167339,8	161296,6	**168783,6**	163543,0	167530,7	166001,0	**169697,7**	164541,0	
16C	7	3	**172377,7**	166562,3	171767,6	170133,0	173273,9	171377,0	172754,6	**176161,0**	
16C	7	2	**164531,3**	161241,1	163850,8	163305,0	165125,2	163305,0	164625,7	**166479,0***	(135509,4)
18	9	4	**205781,9**	184222,0			206759,4		**213805,5**	193632,0	
20	10	5	**245897,4**	216462,6			**250372,6**			220907,0	
22	11	5	**266415,6**	245030,5			**269735,5**			243052,0	
24	12	6	**297898,6**	272970,0			**301441,0**			250590,0	
26	13	6	**333504,9**	312705,5			**336854,4**			289651,0	
26	5	5	**324387,1**				**324753,2**	318690,0			
28	14	7	**374630,6**	350290,9			**377356,2**			322208,0	
28	5	5	**363072,1**				**363541,4**	358593,0			
30	15	7	**422026,0**				**424537,6**			339331,0	
30	5	5	**415296,4**	398032,9		403725,0	**415747,5**	413103,0			
32	16	8	**462894,6**	430514,9			**468803,5**			369695,0	
32	5	5	**455836,4**				**456685,9**	443281,0			

Tabela 5.6: Comparação entre os limitantes inferiores obtidos pelo *branch-and-cut* e os melhores limitantes inferiores da literatura. O tempo (em segundos) que BB-LID levou para provar a otimalidade ou infactibilidade das instâncias aparece entre parênteses na última coluna.

cutados por longos períodos de tempo. Observe também que BB-LID parece sofrer de problemas de escalabilidade, já que seu bom desempenho nas instâncias do grupo MENORES não se repete no grupo MAIORES. Na verdade, todos os limitantes de BB-LID para as instâncias do grupo MAIORES, exceto um, são piores que aqueles obtidos por BB em 3 horas (desconsideramos aqui a instância com 30 times, $q_1 = 15$ e $q_2 = 7$ para a qual nenhum limitante de BB está disponível).

Por fim, avaliamos quando é interessante dar mais tempo de computação para BC em termos de melhorias nos limitantes inferiores. Comparando os resultados nas colunas 4 e 8 (BC limitado em 3 e 24 horas) para as últimas 11 linhas da tabela 5.6, constatamos

que o aumento médio/máximo/mínimo no limitante inferior, de 3 para 24 horas, é de 2542,6 / 5908,9 / 366,1 milhas, com desvio padrão de 1833,9 milhas. Estes valores são grosseiramente uma ordem de magnitude menores que aqueles da comparação de BC com os outros métodos. Isto pode ser um indicativo de que não há um ganho substancial nos limitantes inferiores se permitirmos que BC execute por muito mais tempo.

5.4 Conclusões

Introduzimos neste capítulo um modelo de programação linear inteira parametrizado para o TUP que generaliza os dois melhores modelos existentes na literatura. Nossa parametrização determina o tamanho das sequências de viagens dos árbitros, que varia de 2 a $4n-2$ rodadas, representadas pelas variáveis binárias de decisão do modelo. Esta flexibilidade nos permite explorar a relação de custo-benefício entre o tempo de execução (que diminui quando as sequências de viagens são curtas) e a qualidade do limitante inferior (que aumenta quando as sequências de viagens são longas). Este modelo é posteriormente fortalecido por novas famílias de desigualdades válidas, adicionadas na formulação na medida em que se tornam violadas durante a execução do algoritmo *branch-and-cut* (BC) implementado.

Nossos resultados computacionais comprovam a relevância e o impacto de nossas desigualdades e confirmam a relação de velocidade/qualidade em função do tamanho das viagens (em termos da quantidade de sedes/rodadas percorridas) representadas pelas variáveis do modelo. O BC foi desenvolvido com o objetivo de resolver instâncias de tamanhos mais próximos daqueles encontrados em situações reais. Nossos experimentos mostram que o método escala melhor que outros métodos alternativos existentes, uma vez que ele continua a encontrar limitantes inferiores fortes mesmo para instâncias com 20 ou mais times, melhorando todos os limitantes conhecidos na literatura para estas instâncias. Embora as instâncias menores não tenham sido foco do nosso trabalho, é importante ressaltar que somente um método foi melhor que BC nas instâncias com 14, 16 e 18 times. Com base em sua robustez em produzir limitantes inferiores de alta qualidade para instâncias médias e grandes, acreditamos que BC atualmente pode ser considerado como um dos métodos mais competitivos para o TUP.

Como trabalho futuro, sugerimos o estudo de heurísticas primais acopladas ao BC a fim de obter limitantes superiores bons e ajudar a poda da árvore de enumeração. Além disso, ao invés de incluir todas as variáveis *a priori*, poderia ser utilizada uma rotina de *pricing* para adicioná-las dinamicamente (assim como é feito em um algoritmo *branch-and-price-and-cut*) visando melhorar os tempos execução. Suspeitamos que o problema de *pricing* seja desafiador pois ele precisa levar em consideração os nossos planos de cortes específicos, mas acreditamos que talvez o ganho de tempo ao resolver relaxações lineares menores compense o processamento extra gasto no *pricing*.

Capítulo 6

Considerações finais

Nesta tese, concentramos nossos esforços no estudo do problema dos árbitros viajantes (TUP, do inglês *traveling umpire problem*). O TUP consiste em um problema de otimização cujo objetivo principal é atribuir árbitros às partidas de um torneio *round robin* duplo minimizando a distância total viajada por eles durante toda a competição. Além disso, neste problema são impostas restrições que impedem que qualquer árbitro apite mais de um jogo em uma mesma sede, ou de um mesmo time, durante determinadas quantidades de rodadas consecutivas, e exige que cada árbitro apite um jogo de cada time em sua sede ao menos uma vez. O TUP é uma versão abstrata do problema real que ocorre durante o planejamento do calendário de jogos da Liga Profissional de Beisebol (MLB, do inglês *Major League Baseball*) dos Estados Unidos, sendo que as principais características que tornam o problema da MLB difícil de ser resolvido foram incorporadas no TUP.

Desde a sua proposição em 2007 (em [46]) até os dias atuais, o TUP tem se mostrado um problema de elevado grau de dificuldade. Isso nos motivou a estudá-lo a fim de propor soluções para este problema. Como resultado, elaboramos e divulgamos três diferentes pesquisas sobre o TUP através das seguintes publicações (em ordem cronológica):

1. L. Oliveira, C. C. Souza, e T. Yunes. Improved bounds for the traveling umpire problem: A stronger formulation and a relax-and-fix heuristic. *European Journal of Operational Research*, 236(2):592–600, 2014.

2. L. Oliveira, C. C. Souza, e T. Yunes. On the complexity of the traveling umpire problem. *Theoretical Computer Science*, 562:101–111, 2015.

3. L. Oliveira, C. C. Souza, e T. Yunes. Lower bounds for large traveling umpire instances: New valid inequalities and a branch-and-cut algorithm. *Computers & Operations Research*, 72:147–159, 2016.

Estes trabalhos são descritos detalhadamente nos capítulos 3, 4 e 5 desta tese, e introduzem novos métodos de resolução, modelos matemáticos e uma prova de complexidade para o TUP. Em resumo, os principais resultados obtidos com as pesquisas desenvolvidas nesta tese são:

- Demonstramos que este problema é $\mathcal{NP}$-completo, para determinados parâmetros de entrada, resolvendo esta questão que estava em aberto há sete anos desde a sua proposição (capítulo 3);

- Introduzimos uma nova formulação baseada em fluxo em rede que melhorou todos os limitantes inferiores conhecidos até então, e foi o primeiro método capaz de produzir limitantes inferiores não triviais para instâncias com mais de 16 times (capítulo 4);

- Acreditamos que esta formulação também contribuiu indiretamente para a obtenção de limitantes inferiores ainda melhores pelo método de decomposição proposto em [51], já que ela tornou viável a resolução de subproblemas maiores na otimalidade, o que melhora consideravelmente os limitantes inferiores produzidos por aquele método;

- Desenvolvemos uma heurística *relax-and-fix*, baseada na formulação mencionada nos itens anteriores, que encontrou soluções melhores para 24 das 25 instâncias com 14, 16 e 30 times (capítulo 4);

- Por fim, enquanto os melhores métodos atuais permaneceram focados na resolução de instâncias com no máximo 18 times e não escalam para instâncias maiores, formulamos um segundo modelo matemático baseado em sub-rotas, capaz de lidar com instâncias grandes (com torneios de 20 a 32 times, tamanhos mais realistas com relação ao problema da MLB), que obteve os melhores limitantes inferiores conhecidos para estas instâncias até o momento da escrita desta tese (capítulo 5).

Esperamos que nossos trabalhos estimulem o interesse em novas pesquisas tanto para o TUP quanto para os demais problemas relacionados a esportes e promovam avanços nesta área de conhecimento. Diversos trabalhos futuros foram indicados como sugestões nas conclusões dos capítulos 3, 4 e 5.

Referências

[1] J. Armstrong e R. J. Willis. Scheduling the cricket world cup – a case-study. *Journal of the Operational Research Society*, 44(11):1067–1072, 1993.

[2] T. Bartsch, A. Drexl, e S. Kröger. Scheduling the professional soccer leagues of austria and germany. *Computers & Operations Research*, 33(7):1907–1937, 2006.

[3] M. S. Bazaraa, J. J. Jarvis, e H. F. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, Hoboken, NJ, USA, 4a edição, 2010.

[4] J. C. Bean e J. R. Birge. Reducing travelling costs and player fatigue in the national basketball association. *Interfaces*, 10(3):98–102, 1980.

[5] M. Bender e S. Westphal. A combined approximation for the traveling tournament problem and the traveling umpire problem. *Journal of Quantitative Analysis in Sports*, 12(3):139–149, 2016.

[6] D. Bertsimas e J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, USA, 1997.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, e C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, 3a edição, 2009.

[8] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, USA, 1963.

[9] D. de Werra. Geography, games and graphs. *Discrete Applied Mathematics*, 2(4): 327–337, 1980.

[10] D. de Werra. Scheduling in sports. In P. Hansen, editor, *Annals of Discrete Mathematics (11) Studies on Graphs and Discrete Programming*, volume 59, p. 381–395. North-Holland, 1981.

[11] D. de Werra. Minimizing irregularities in sports schedules using graph theory. *Discrete Applied Mathematics*, 4(3):217–226, 1982.

[12] D. de Werra. On the multiplication of divisions: The use of graphs for sports scheduling. *Networks*, 15(1):125–136, 1985.

[13] J. Dinitz, E. Lamken, e W. D. Wallis. Scheduling a tournament. In C. J. Colbourn e J. Dinitz, editores, *Handbook of Combinatorial Designs*, p. 578–584. CRC Press, 1995.

[14] K. Easton, G. Nemhauser, e M. Trick. The traveling tournament problem description and benchmarks. In T. Walsh, editor, *Principles and Practice of Constraint Programming – CP 2001*, volume 2239 of *Lecture Notes in Computer Science*, p. 580–584. Springer, 2001.

[15] J. R. Evans. A microcomputer-based decision support system for scheduling umpires in the american baseball league. *Interfaces*, 18(6):42–51, 1988.

[16] J. R. Evans, J. E. Hebert, e R. F. Deckro. Play ball – the scheduling of sports officials. *Perspectives in Computing*, 4(1):18–29, 1984.

[17] A. Farmer, J. S. Smith, e M. L. T. Scheduling umpire crews for professional tennis tournaments. *Interfaces*, 37(2):187–196, 2007.

[18] J. A. Ferland e C. Fleurent. Computer aided scheduling for a sport league. *INFOR*, 29(1):14–25, 1991.

[19] C. Fleurent e J. A. Ferland. Allocating games for the NHL using integer programming. *Operations Research*, 41(4):649–654, 1993.

[20] M. J. Fry e J. W. Ohlmann. Introduction to the special issue on analytics in sports, part ii: Sports scheduling applications. *Interfaces*, 42(3):229–231, 2012.

[21] M. R. Garey e D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, USA, 1979.

[22] T. Januario, S. Urrutia, C. C. Ribeiro, e D. de Werra. Edge coloring: A natural model for sports scheduling. *European Journal of Operational Research*, 254(1):1–8, 2016.

[23] B. Kallehauge, N. Boland, e O. B. G. Madsen. Path inequalities for the vehicle routing problem with time windows. *Networks*, 49(4):273–293, 2007.

[24] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller e J. W. Thatcher, editores, *Complexity of Computer Computations*, p. 85–103. Plenum Press, 1972.

[25] G. Kendall, S. Knust, C. C. Ribeiro, e S. Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1):1–19, 2010.

[26] T. P. Kirkman. On a problem in combinations. *The Cambridge and Dublin Mathematical Journal*, 2:191–204, 1847.

[27] S. Knust. Scheduling non-professional table-tennis leagues. *European Journal of Operational Research*, 200(2):358–367, 2010.

[28] E. Lucas. *"Sixième récréation: Les jeux de demoiselles"*, volume 2, p. 161–197. Gauthier-Villars, Paris, França, 1883.

[29] E. Mendelsohn e A. Rosa. One-factorizations of the complete graph – a survey. *Journal of Graph Theory*, 9(1):43–65, 1985.

[30] R. Miyashiro e T. Matsui. Minimizing the carry-over effects value in a round robin tournament. In E. K. Burke e H. Rudová, editores, *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, p. 460–463, 2006.

[31] G. L. Nemhauser e L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[32] S. Niskanen e P. R. J. Östergård. Cliquer user's guide, version 1.0. Relatório Técnico T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, 2003.

[33] L. Oliveira, C. C. Souza, e T. Yunes. Improved bounds for the traveling umpire problem: A stronger formulation and a relax-and-fix heuristic. *European Journal of Operational Research*, 236(2):592–600, 2014.

[34] L. Oliveira, C. C. Souza, e T. Yunes. On the complexity of the traveling umpire problem. *Theoretical Computer Science*, 562:101–111, 2015.

[35] L. Oliveira, C. C. Souza, e T. Yunes. Lower bounds for large traveling umpire instances: New valid inequalities and a branch-and-cut algorithm. *Computers & Operations Research*, 72:147–159, 2016.

[36] R. V. Rasmussen. Scheduling a triple round robin tournament for the best danish soccer league. *European Journal of Operational Research*, 185(2):795–810, 2008.

[37] R. V. Rasmussen e M. A. Trick. Round robin scheduling – a survey. *European Journal of Operational Research*, 188:617–636, 2008.

[38] A. Rosa e W. D. Wallis. Premature sets of 1-factors or how not to schedule round robin tournaments. *Discrete Applied Mathematics*, 4(4):291–297, 1982.

[39] J. A. M. Schreuder. Constructing timetables for sport competitions. *Mathematical Programming Study*, 13:58–67, 1980.

[40] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, NY, USA, 1986.

[41] T. Toffolo e T. Wauters. Traveling umpire problem automated benchmark. `http://gent.cs.kuleuven.be/tup`, acessado em agosto de 2015.

[42] T. A. M. Toffolo, S. V. Malderen, T. Wauters, e G. V. Berghe. Branch-and-price and improved bounds to the traveling umpire problem. In *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling*, p. 420–432, 2014.

[43] T. A. M. Toffolo, T. Wauters, S. V. Malderen, e G. V. Berghe. Branch-and-bound with decomposition-based lower bounds for the traveling umpire problem. *European Journal of Operational Research*, 250(3):737–744, 2016.

[44] M. A. Trick. Traveling umpire problem: Data sets and results. `http://mat.tepper.cmu.edu/TUP`, acessado em abril de 2015.

[45] M. A. Trick. Challenge traveling tournament instances. `http://mat.gsia.cmu.edu/TOURN`, acessado em dezembro de 2016.

[46] M. A. Trick e H. Yildiz. Benders' cuts guided large neighborhood search for the traveling umpire problem. In P. Van Hentenryck e L. Wolsey, editores, *Proceedings of the Fourth Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, volume 4510 of *Lecture Notes in Computer Science*, p. 332–345. Springer-Verlag, 2007.

[47] M. A. Trick e H. Yildiz. Benders' cuts guided large neighborhood search for the traveling umpire problem. *Naval Research Logistics*, 58(8):771–781, 2011.

[48] M. A. Trick e H. Yildiz. Locally optimized crossover for the traveling umpire problem. *European Journal of Operational Research*, 216(2):286–292, 2012.

[49] M. A. Trick, H. Yildiz, e T. Yunes. Scheduling major league baseball umpires and the traveling umpire problem. *Interfaces*, 42(3):232–244, 2012.

[50] J. Vennekens. Solving the travelling umpire problem with answer set programming. In *Proceedings of the 28th Benelux conference on artificial intelligence*, p. 96–103, 2016.

[51] T. Wauters, S. V. Malderen, e G. V. Berghe. Decomposition and local search based methods for the traveling umpire problem. *European Journal of Operational Research*, 238:886–898, 2014.

[52] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, USA, 2a edição, 2001.

[53] L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998.

[54] M. B. Wright. Scheduling english cricket umpires. *Journal of the Operational Research Society*, 42(6):447–452, 1991.

[55] M. Wright. Scheduling fixtures for basketball new zealand. *Computers & Operations Research*, 33(7):1875–1893, 2006.

[56] L. Xue, L. Z., e A. Lim. Two exact algorithms for the traveling umpire problem. *European Journal of Operational Research*, 243(3):932–943, 2015.

[57] D. Yamaguchi, S. Imahori, R. Miyashiro, e T. Matsui. An improved approximation algorithm for the traveling tournament problem. In Y. Dong, D.-Z. Du, e O. Ibarra, editores, *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, volume 5878 of *Lecture Notes in Computer Science*, p. 679–688. Springer-Verlag, 2009.

[58] H. Yildiz. *Methodologies and Applications for Scheduling, Routing & Related Problems*. Tese de doutorado, Tepper School of Business, Carnegie Mellon University, 2008.