O problema dos árbitros viajantes: complexidade, modelagem e algoritmos

Lucas de Oliveira
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP)
Campinas, SP, Brasil
E-mail: lucas.oliveira@ic.unicamp.br

Cid Carvalho de Souza
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP)
Campinas, SP, Brasil
E-mail: cid@ic.unicamp.br

Tallys Yunes
Escola de Administração de Empresas
Universidade de Miami
Coral Gables, FL, EUA
E-mail: tallys@miami.edu

***Resumo*—Estudamos neste doutorado o problema dos árbitros viajantes (TUP, do inglês *traveling umpire problem*), que consiste em um problema de otimização baseado no problema real de alocação de árbitros às partidas da Liga Profissional de Beisebol dos Estados Unidos. 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 e respeitando restrições que impõem que cada árbitro não apite jogos de um mesmo time frequentemente e apite ao menos um jogo na sede de cada time. 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 de excelente qualidade para o TUP, melhorando diversos resultados da literatura.**

1. Introdução

Nos últimos anos houve um crescimento do número de estudos abordando problemas relacionados com esportes, que surgem nas mais diferentes modalidades, tais como basquete, beisebol, críquete, futebol, hóquei, tênis, etc [1]. A maioria destes problemas tratam o planejamento do calendário de jogos da competição ou a alocação dos árbitros que irão apitar as partidas. As revisões bibliográficas [1], [2] e [3] apresentam diversas referências de estudos sobre problemas da área de esportes.

Neste doutorado, nos concentramos em estudar o problema dos árbitros viajantes (TUP, do inglês *traveling umpire problem*), que é uma versão abstrata que incorpora as principais características do problema real de alocação de equipes de árbitros às partidas da Liga Profissional de Beisebol (MLB, do inglês *Major League Baseball*) dos Estados Unidos [4]. 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:

(I) Cada partida deve ser apitada por exatamente um árbitro;
(II) Cada árbitro é atribuído a exatamente uma partida em cada rodada do torneio;
(III) Cada árbitro apita pelo menos uma partida de cada time em sua casa;
(IV) Cada árbitro apita no máximo um jogo em uma mesma sede durante $q_1 = n - d_1$ rodadas consecutivas;
(V) 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 satifaz estas restrições e minimiza a distância total viajada pelos árbitros durante todo o torneio.

Ilustramos na figura 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 I e II são satisfeitas. Todos os árbitros visitam cada sede ao menos uma vez o que satisfaz a restrição III (por exemplo, as oito sedes são visitadas pelo árbitro 1 nas rodadas 1–7 e 9). 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 IV e V (por exemplo, o árbitro 1 visita as sedes 5, 3, 7 e 2 nas quatro primeiras rodadas e apita os jogos dos times 5, 1, 3 e 6 nas duas primeiras rodadas, sem repetir sedes ou times nestas rodadas consecutivas).

Os trabalhos sobre o TUP publicados na literatura deixam evidente a sua grande dificuldade de resolução (veja

	Rodadas						
Árbitro	1	2	3	4	5	6	7
1	(5,1)	(3,6)	(7,1)	(2,3)	(1,4)	(8,2)	(6,7)
2	(4,8)	(1,2)	(6,4)	(8,1)	(3,5)	(1,6)	(4,2)
3	(2,6)	(8,7)	(5,2)	(4,7)	(6,8)	(7,5)	(3,1)
4	(7,3)	(5,4)	(8,3)	(6,5)	(2,7)	(4,3)	(8,5)

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

Figura 1. Solução de um instância do TUP com um torneio de 8 times e $d_1 = d_2 = 0$.

seção 2). Até mesmo encontrar boas soluções factíveis para torneios com 14 ou 16 times (consideravelmente menores que a competição da MLB que possui 30 times) revela-se uma tarefa difícil e demanda métodos mais robustos. Portanto, isso nos motivou a estudá-lo e tê-lo como tema deste doutorado.

Antes do início das nossas pesquisas, havia na literatura apenas três trabalhos sobre o TUP: [4], [5] (extensão de [6]) e [7]. Com base neles, constatamos que: *(1)* era difícil obter boas soluções para o TUP, visto que todos os métodos desenvolvidos até então retornavam em alguns casos soluções muito piores que as melhores conhecidas ou não encontravam soluções para instâncias comprovadamente factíveis, *(2)* a formulação matemática de programação linear inteira proposta para o problema produzia resultados razoavelmente satisfatórios apenas para instâncias com no máximo 16 times e, *(3)* apesar das fortes suspeitas deste problema ser $\mathcal{NP}$-completo, não se conhecia nenhum resultado teórico sólido corroborando esse sentido.

Neste doutorado, desenvolvemos três diferentes linhas de pesquisa com o objetivo de promover avanços para solucionar os problemas descritos no parágrafo anterior. Nossos esforços culminaram com a publicação de três artigos: [8], [9] e [10]. Primeiramente, propusemos e avaliamos em [8] uma formulação matemática de programação linear inteira baseada em fluxo em rede mais forte e compacta para o TUP. Com ela foi possível obter limitantes inferiores melhores para as instâncias com até 16 times e reportar os primeiros limitantes inferiores fortes para instâncias com mais de 16 times. Através desta formulação, também desenvolvemos uma heurística *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 [9] que este problema é, de fato, $\mathcal{NP}$-completo para determinados tipos de instâncias. Por fim, apresentamos em [10] 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 limitantes inferiores melhores para todas as instâncias grandes com 20 ou mais times.

O restante do texto está estruturado da seguinte forma. Na próxima seção é apresentada uma revisão dos trabalhos sobre o TUP publicados na literatura. Na seção 3 é demonstrado que o TUP é um problema $\mathcal{NP}$-completo (conteúdo publicado em [9]). A seção 4 introduz uma formulação matemática de programação linear inteira baseada em fluxo em rede e uma heurística *relax-and-fix* (conteúdo publicado em [8]). A seção 5 apresenta uma formulação matemática de programação linear inteira baseada em sub-rotas (conteúdo publicado em [10]). Por fim, na seção 6 discutimos as conclusões dos trabalhos realizados.

2. Revisão da literatura

A seguir apresentamos uma breve revisão da literatura do TUP, incluindo as nossas publicações durante o doutorado.

O TUP possui um conjunto de instâncias *benchmark* [11] de domínio público com torneios de 4 até 32 times. O nome da cada instância possui o número de times em seu torneio, seguido facultativamente por uma letra, cuja presença indica que a instância tem o mesmo torneio da instância original (sem letra) mas uma matriz de distâncias entre as sedes diferente. Todos os trabalhos na literatura utilizam este conjunto de instâncias para avaliar os métodos propostos.

Os autores em [5] (extensão de [6] que introduziu o TUP) propuseram um modelo de programação linear inteira e um modelo de programação por restrições para o TUP. Métodos exatos foram executados para estes modelos, resolvendo na otimalidade apenas instâncias com no máximo 10 times e provando a infactibilidade de uma instância com 12 times. Também foi desenvolvida uma heurística gulosa guiada por cortes de Bender que encontrou soluções melhores que os métodos exatos (dentro dos limites de tempo) para várias instâncias com 14, 16 e 30 times.

Em [4] é apresentada uma heurística *simulated anneaing* aplicada tanto ao TUP quanto ao problema real da MLB que não superou nenhum dos resultados obtidos para o TUP pela heurística gulosa guiada por cortes de Bender. Em [7] é proposto um algoritmo genético com um sofisticado operador de cruzamento que obteve soluções melhores para diversas instâncias com 14, 16 e 30 times.

O trabalho apresentado por nós em [8] introduziu uma formulação matemática de programação linear inteira baseada em fluxo em rede mais forte e compacta que aquela vista em [5]. Ao resolvê-la, obtivemos melhores limitantes inferiores para todas as instâncias *benchmark*, e fomos os primeiros a prover limitantes inferiores fortes para as instâncias com mais de 16 times. Além disso, utilizamos esta formulação para desenvolver uma heurística *relax-and-fix* que encontrou soluções melhores que todas, exceto uma, as melhores conhecidas até então.

Em [12] foram desenvolvidas uma busca em profundidade iterativa e uma busca local iterativa. A busca em profundidade melhorou diversas soluções nas instâncias com 14 e 16 times, ao passo que a busca local encontrou melhores soluções para as instâncias com 26 ou mais times. Os autores também propuseram um esquema de decomposição que

obteve limitantes inferiores melhores para todas as instâncias consideradas nos experimentos.

Um modelo de programação linear inteira baseado em partição de conjuntos foi proposto em [13], e resolvido com um algoritmo *branch-and-price*. Vários limitantes inferiores e algumas soluções foram melhorados para as instâncias com 14 e 16 times. Através de um método adaptado, os autores também conseguiram provar a infactibilidade de quatro instâncias com 16 times.

Dois modelos de programação linear inteira foram formulados em [14], sendo que um foi resolvido por um algoritmo *branch-and-bound* e o outro por um algoritmo *branch-and-price-and-cut*. O *branch-and-bound* conseguiu melhorar vários dos limitantes inferiores nas instâncias com mais de 18 times, já o *branch-and-price-and-cut* melhorou vários dos limitantes inferiores das instâncias com até 18 times e foi o primeiro a resolver (duas) instâncias com 14 times na otimalidade (gastando em torno de 11 horas em uma e 34 horas na outra).

Em [10] propusemos um modelo de programação linear inteira baseado em sub-rotas que generaliza aqueles vistos em [13] e [14]. Um algoritmo *branch-and-cut* foi desenvolvido para resolver este modelo. Este método obteve limitantes inferiores competitivos paras as instâncias com até 18 times e melhores para todas as instâncias com 20 ou mais times. Além disso, foram resolvidas na otimalidade as duas instâncias com 14 times solucionadas em [14], contudo gastando no máximo 30 minutos, e mais outras duas instâncias com 14 times.

Em [15] foi introduzido um algoritmo *branch-and-bound* combinado com uma rotina de geração de limitantes inferiores que aplica a técnica de decomposição apresentada em [12]. Este método resolveu na otimalidade todas as instâncias com 14 times gastando poucos minutos e 11 instâncias com 16 times gastando poucas horas, sendo o primeiro a conseguir resolver na otimalidade instâncias com 16 times. Além disso, alguns limitantes inferiores e superiores também foram melhorados para as instâncias com 16 times. 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 [10].

A primeira prova de complexidade para o TUP foi apresentada por nós em [9]. 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 $d_2 = \lfloor \frac{n}{2} \rfloor - 1$.

Uma aproximação combinada foi desenvolvida para o TUP em [16]. Nesta aproximação, os autores descrevem uma forma de construir 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 [17] um algoritmo *answer set programming* que mostrou-se competitivo quando comparado com os métodos apresentados em [5], mas não superou nenhum dos melhores resultados da literatura.

3. Complexidade do TUP

Demonstramos em [9] que o TUP é $\mathcal{NP}$-completo 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. Nesta seção vamos descrever os detalhes desta redução e apresentar apenas os enunciados dos lemas, teoremas e corolários envolvidos nesta prova. As demonstrações dos mesmos podem ser vistas em [9], bem como exemplos maiores e explicações mais detalhadas.

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 nesta seção adotamos índices (para os times, rodadas, etc) que iniciam em zero. Nas demais seções 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. Então, 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 (1)–(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 $b+a-1$.

$$U_{a,b} = U_{a,b}[0, a-2], \quad (1)$$

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

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.

$$Q_{a,b}[s] = \{(q(0), b+a-1), (q(1), q(a-2)), (q(2), q(a-3)), \vdots (q(a/2-1), q(a-a/2))\}, \forall\ 0 \le s \le a-2, \quad (3)$$

$$q(c) = b + ((s+c) \bmod (a-1)). \quad (4)$$

A figura 2 exemplifica o torneio $U_{a,b}$ com $a = 8$ e $b = 0$. A definição algébrica de $U_{a,b}$ resulta em um método equivalente ao método do círculo ou polígono [18], conhecido também como método de Kirkman [19]. Em [9] demonstramos que $U_{a,b}$ sempre resulta corretamente em um torneio *round robin* simples.

$U_{8,0}$

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)

Figura 2. Torneio $U_{8,0}$.

Além de U, definimos nas equações (5)–(7) um outro tipo de torneio, denotado por $P_{a,b}$, com $a > 0$ rodadas e $2a$ times com índices consecutivos iniciando em $b \ge 0$. 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.

$$P_{a,b} = P_{a,b}[0, a-1], \quad (5)$$

$$P_{a,b}[s_1, s_2] = X_{a,b}[s_1], X_{a,b}[s_1+1], ..., X_{a,b}[s_2], \quad \forall\ 0 \le s_1 \le s_2 \le a-1, \quad (6)$$

$$X_{a,b}[s] = \{(b+0, b+a+(s \bmod a)), (b+1, b+a+((s+1) \bmod a)), \vdots (b+a-1, b+a+((s+a-1) \bmod a))\}, \forall\ 0 \le s \le a-1. \quad (7)$$

Note que em um torneio $P_{a,b}$ cada time disputa exatamente um jogo em cada rodada. A figura 3 apresenta o torneio $P_{a,b}$ com $a = 8$ e $b = 0$. A notação $P_{a,b}[s_1, s_2]$ definida em (6) 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, 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 $g = h$.

$P_{8,0}$

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)

Figura 3. Torneio $P_{8,0}$.

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. O torneio T é definido por (8)–(11). A figura 4 apresenta todos os jogos de T para $k = 4$, indicando qual termo em (8)–(11) gerou cada parte do torneio.

$$T_1 = \overline{P}_{2k,0}[0, k-1] \oslash (\overline{U}_{k,0} \odot \overline{U}_{k,k} \odot \overline{U}_{k,2k} \odot \overline{U}_{k,3k}), \quad (8)$$

$$T_2 = \overline{P}_{2k,0}[k, 2k-1] \oslash (U_{k,0} \odot U_{k,k} \odot U_{k,2k} \odot U_{k,3k}), \quad (9)$$

$$T_3 = (\overline{P}_{k,0} \odot \overline{P}_{k,2k}) \oslash (P_{k,0} \odot P_{k,2k}), \quad (10)$$

$$T = T_1 \oplus P_{2k,0}[0, k-1] \oplus T_2 \oplus P_{2k,0}[k, 2k-1] \oplus T_3. \quad (11)$$

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

Demonstração. Veja em [9]. □

3.2. Redução em tempo polinomial

Para concluirmos que o TUP é $\mathcal{NP}$-completo, mostraremos 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 reduziremos este problema ao TUP em tempo polinomial.

Lema 3.1. *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. Veja em [9]. □

Vamos ver agora como converter, em tempo polinomial, uma instância do problema de decisão descrito no enunciado do lema 3.1 em uma instância do TUP com $d_1 \le n/2$ e $d_2 = \lfloor n/2 \rfloor - 1$.

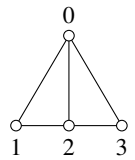
Rodadas																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\overline{P}_{8,0}[0,3] \oslash (\overline{U}_{4,0} \odot \overline{U}_{4,4} \odot \overline{U}_{4,8} \odot \overline{U}_{4,12})$							$P_{8,0}[0,3]$				$\overline{P}_{8,0}[4,7] \oslash (U_{4,0} \odot U_{4,4} \odot U_{4,8} \odot U_{4,12})$						
(8,0)	(3,0)	(9,0)	(3,1)	(10,0)	(3,2)	(11,0)	(0,8)	(0,9)	(0,10)	(0,11)	(12,0)	(0,3)	(13,0)	(1,3)	(14,0)	(2,3)	(15,0)
(9,1)	(2,1)	(10,1)	(0,2)	(11,1)	(1,0)	(12,1)	(1,9)	(1,10)	(1,11)	(1,12)	(13,1)	(1,2)	(14,1)	(2,0)	(15,1)	(0,1)	(8,1)
(10,2)	(7,4)	(11,2)	(7,5)	(12,2)	(7,6)	(13,2)	(2,10)	(2,11)	(2,12)	(2,13)	(14,2)	(4,7)	(15,2)	(5,7)	(8,2)	(6,7)	(9,2)
(11,3)	(6,5)	(12,3)	(4,6)	(13,3)	(5,4)	(14,3)	(3,11)	(3,12)	(3,13)	(3,14)	(15,3)	(5,6)	(8,3)	(6,4)	(9,3)	(4,5)	(10,3)
(12,4)	(11,8)	(13,4)	(11,9)	(14,4)	(11,10)	(15,4)	(4,12)	(4,13)	(4,14)	(4,15)	(8,4)	(8,11)	(9,4)	(9,11)	(10,4)	(10,11)	(11,4)
(13,5)	(10,9)	(14,5)	(8,10)	(15,5)	(9,8)	(8,5)	(5,13)	(5,14)	(5,15)	(5,8)	(9,5)	(9,10)	(10,5)	(10,8)	(11,5)	(8,9)	(12,5)
(14,6)	(15,12)	(15,6)	(15,13)	(8,6)	(15,14)	(9,6)	(6,14)	(6,15)	(6,8)	(6,9)	(10,6)	(12,15)	(11,6)	(13,15)	(12,6)	(14,15)	(13,6)
(15,7)	(14,13)	(8,7)	(12,14)	(9,7)	(13,12)	(10,7)	(7,15)	(7,8)	(7,9)	(7,10)	(11,7)	(13,14)	(12,7)	(14,12)	(13,7)	(12,13)	(14,7)

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

Figura 4. Torneio T com $k = 4$.

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 (8)–(11), 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 5 ilustra um grafo G com 4 vértices, sendo ao menos um deles universal, (instância da variação do problema do ciclo hamiltoniano vista no lema 3.1) 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 16 times neste exemplo é aquele apresentado na figura 4. As sedes $0, \ldots, 3$ 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.



Sede	0	1	2	3	4	$\cdots$	15
0	1	0	0	0	1	$\cdots$	1
1	0	1	0	1	1	$\cdots$	1
2	0	0	1	0	1	$\cdots$	1
3	0	1	0	1	1	$\cdots$	1
4	1	1	1	1	1	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
15	1	1	1	1	1	$\cdots$	1

Figura 5. Exemplo de um grafo G com 4 vértices (à esquerda) e a matriz de distâncias entre as sedes da instância $I(G)$ do TUP (à direita).

Por fim, mostramos que a redução está correta provando o teorema 3.2 e, por consequência, chegamos ao resultado do corolário 3.1.

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.*

Demonstração. Veja em [9]. □

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. Veja em [9]. □

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. 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.

4. Formulação matemática baseada em fluxo em rede

Nesta seção vamos estudar a primeira formulação matemática de programação linear inteira baseada em fluxo

em rede proposta para o TUP na literatura e, com base nela, introduzimos 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*. O conteúdo apresentado nesta seção foi publicado em [8].

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.

A primeira formulação matemática de programação linear inteira para o TUP, baseada em fluxo em rede, foi introduzida em [6], e também utilizada em [4], [5] e [7]. 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 $S = \{1, \ldots, 4n-2\}$;
- 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} = $\begin{cases} 1, & \text{se o árbitro } u \text{ está na sede } i \text{ durante} \\ & \text{a rodada } s, \text{ e viaja para a sede } j \\ & \text{na rodada } s+1, \\ 0, & \text{caso contrário.} \end{cases}$

Agora estamos prontos para apresentar a formulação.

$$\text{Minimizar} \sum_{i \in T} \sum_{j \in T} \sum_{u \in U} \sum_{s \in S: s < |S|} d_{ij} z_{ijsu} \tag{12}$$

Sujeito a:

$$\sum_{u \in U} x_{isu} = 1, \quad \begin{array}{l} \forall\, i \in T, s \in S: \\ \text{OPP}[s,i] > 0, \end{array} \tag{13}$$

$$\sum_{i \in T: \text{OPP}[s,i] > 0} x_{isu} = 1, \quad \forall\, s \in S, u \in U, \tag{14}$$

$$\sum_{s \in S: \text{OPP}[s,i] > 0} x_{isu} \geq 1, \quad \forall\, i \in T, u \in U, \tag{15}$$

$$\sum_{c \in CV_s: \text{OPP}[c,i] > 0} x_{icu} \leq 1, \quad \begin{array}{l} \forall\, i \in T, u \in U, s \in S: \\ s \leq |S| - (n - d_1 - 1), \end{array} \tag{16}$$

$$\sum_{c \in CT_s} \left(x_{icu} + \sum_{\substack{j \in T: \\ \text{OPP}[c,j] = i}} x_{jcu} \right) \leq 1, \quad \begin{array}{l} \forall\, i \in T, u \in U, \\ s \in S: s \leq q, \end{array} \tag{17}$$

$$x_{isu} + x_{j(s+1)u} - z_{ijsu} \leq 1, \quad \begin{array}{l} \forall\, i, j \in T, u \in U, \\ s \in S: s < |S|, \end{array} \tag{18}$$

$$x_{isu} \in \{0,1\}, \quad \begin{array}{l} \forall\, i \in T, u \in U, \\ s \in S, \end{array} \tag{19}$$

$$z_{ijsu} \in \{0,1\}, \quad \begin{array}{l} \forall\, i, j \in T, u \in U, \\ s \in S: s < |S|, \end{array} \tag{20}$$

onde $q = |S| - (\lfloor \frac{n}{2} \rfloor - d_2 - 1)$.

A função objetivo (12) minimiza a distância total viajada pelos árbitros. As restrições (13) e (14) determinam que cada jogo deve ser atribuído a um árbitro e que cada árbitro deve apitar um jogo, respectivamente. As restrições III, IV e V são modeladas por (15), (16) e (17), respectivamente. A consistência entre as atribuições dos árbitros aos jogos e suas viagens (variáveis x e z) é imposta por (18). Por fim, as restrições (19) e (20) definem o domínio das variáveis.

Ainda em [5], a formulação acima foi fortalecida com a inclusão das seguintes restrições:

$$x_{isu} = 0, \quad \begin{array}{l} \forall\, i \in T, u \in U, s \in S: \\ \text{OPP}[s,i] < 0, \end{array} \tag{21}$$

$$z_{ijsu} - x_{isu} \leq 0, \quad \begin{array}{l} \forall\, i, j \in T, u \in U, \\ s \in S: s < |S|, \end{array} \tag{22}$$

$$z_{ijsu} - x_{j(s+1)u} \leq 0, \quad \begin{array}{l} \forall\, i, j \in T, u \in U, \\ s \in S: s < |S|, \end{array} \tag{23}$$

$$\sum_{j \in T} z_{jisu} - \sum_{j \in T} z_{ij(s+1)u} = 0, \quad \begin{array}{l} \forall\, i \in T, u \in U, \\ s \in S: s < |S| - 1, \end{array} \tag{24}$$

$$\sum_{i \in T} \sum_{j \in T} z_{ijsu} = 1, \quad \begin{array}{l} \forall\, u \in U, s \in S: \\ s < |S|. \end{array} \tag{25}$$

A restrição (21) proíbe a atribuição de árbitros às sedes onde não são realizados jogos em uma dada rodada. As restrições (22) e (23) permitem que um árbitro viaje de uma sede i para uma 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 (24) 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 (25) impõe que todo árbitro deve se mover de um jogo para outro entre rodadas consecutivas.

A formulação (12)–(25) 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 [20] 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 (26) na formulação, onde K é um conjunto de n tuplas que atribui cada árbitro à sede de um jogo diferente na rodada k:

$$x_{iku} = 1, \quad \forall\, (i,u) \in K. \tag{26}$$

Vamos nos referir à rodada k como *rodada de quebra de simetria*, e à formulação (12)–(26) como $\mathcal{F}_1$. A relaxação

linear de $\mathcal{F}_1$, onde as restrições de integralidade (19)–(20) são substituídas por $0 \leq x_{isu} \leq 1$ e $0 \leq z_{ijsu} \leq 1$, será denotada por $\mathcal{F}_1^R$.

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

$$x_{i1u} = \sum_{j \in T} z_{ij1u}, \ \forall\, i \in T, u \in U, \tag{27}$$

$$x_{isu} = \sum_{j \in T} z_{ji(s-1)u}, \ \forall\, i \in T, u \in U, s \in S : s > 1. \tag{28}$$

A restrição (27) 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, (28) 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 (27) e (28) 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 (27) e (28) para eliminar as variáveis x e removendo as restrições (14), (18), (22), (23) e (25), e por $\mathcal{F}_2^R$ a relaxação linear de $\mathcal{F}_2$.

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

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. Veja em [8]. □

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. Veja em [8]. □

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

$$z_{iisu} = 0, \quad \begin{array}{l} \forall\, i \in T, u \in U, s \in S : s < |S|, \\ \text{se } d_1 < n-1, \end{array} \tag{29}$$

$$z_{ijsu} = 0, \quad \begin{array}{l} \forall\, i \neq j \in T, u \in U, s \in S : s < |S|, \\ \text{se } d_2 < \lfloor \frac{n}{2} \rfloor - 1 \text{ e } (\ \text{OPP}[s,i] = j \\ \text{ou } \text{OPP}[s+1,j] = i \ \text{ ou} \\ \text{OPP}[s,i] = \text{OPP}[s+1,j]\ ). \end{array} \tag{30}$$

Note que (29) e (30) são válidas porque elas proíbem a atribuição de valores positivos às variáveis z que violariam as restrições IV e V (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 (29) e (30).

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 [21, 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. 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 (15) 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 (15) completamente durante muitas iterações as variáveis fixadas podem tornar impossível a satisfação de (15) mais tarde. Portanto, vamos introduzir (15) gradativamente da seguinte forma. Desconsideramos completamente esta restrição durante as iterações $t \leq n$. Nas iterações $n+1, n+2, \ldots, 2n-1$ impomos (15) somente para os árbitros cujos índices são menores ou iguais a 1, 2, …, $n-1$, respectivamente, e nas iterações $t > 2n-1$ impomos (15) para todos os árbitros.

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 [5]. 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* (RF) propostas. Consideramos em nossos experimentos as instâncias do *benchmark* do TUP [11] (uma descrição desse *benchmark* se encontra na página 2). 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 nesta seção, a literatura do TUP possuía apenas os seguintes trabalhos: [4], [5], [6], [7] e [20]. Os melhores limitantes inferiores e superiores reportados no site do *benchmark* do TUP [11] nesta época eram compostos por resultados destas referências e também por resultados preliminares do trabalho apresentado em [12]. Desta forma, as análises realizadas nesta seção comparam os resultados obtidos pelos métodos propostos com os melhores resultados disponíveis na literatura naquela época.

Em [8] conduzimos um estudo experimental aprofundado avaliando os limitantes inferiores, limitantes superiores e tempos de execução para as formulações $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_{2+}$ e suas relaxações lineares considerando diferentes rodadas de quebra de simetria (parâmetro k). Também avaliamos os resultados da heurística *relax-and-fix* para diferentes tamanhos da janela de variáveis binárias (parâmetro b). Nesta seção, por questões de espaço, vamos apresentar apenas a comparação dos nossos melhores resultados com os melhores resultados da literatura. A análise mais detalhada e os ajustes utilizados para cada um dos métodos avaliados podem ser visto em [8].

Embora a nova formulação tenha conseguido reduzir consideravelmente o tempo gasto para resolver na otimalidadas as instâncias com 10 times e o tempo gasto para provar a infactilibidade da instância com 12 times, os principais avanços que obtivemos em relação à literatura foram nas instâncias com pelo menos 14 times. A tabela 1 sumariza os melhores resultados para estas instâncias. O sufixo "-BB" nos nomes das formulações indica que utilizamos $k = 2n-1$ (rodada do meio).

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 $\mathcal{F}_{2+}^R$-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 para $\mathcal{F}_{2+}$-BB superam todos os limitantes inferiores conhecidos 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 1, comparamos a qualidade das melhores soluções conhecidas na literatura até então com aquelas obtidas pelas 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

			Limitante inferior				Valor da solução				
Instância	q_1	q_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 1. COMPARAÇÃO ENTRE OS MELHORES RESULTADOS OBTIDOS PELOS NOSSOS MÉTODOS EXATOS E HEURÍSTICO E OS MELHORES RESULTADOS DA LITERATURA NAS INSTÂNCIAS COM PELO MENOS 14 TIMES.

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 os tempos de execução em [8]). 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 excede em apenas 5 minutos o limite de 3 horas, e instância 30 (marcadas com um "*" na tabela 1). 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 com 30 times, a heurística *relax-and-fix* também encontrou soluções que superam 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.

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.

5. Formulação matemática baseada em sub-rotas

Nesta seção, apresentamos o estudo publicado em [10] onde introduzimos uma nova formulação matemática baseada em sub-rotas para o TUP que explora os pontos fortes das formulações matemáticas baseadas em partição de conjuntos e em fluxo em rede propostas para o TUP em [13] e [14], e inclui novas desigualdades válidas fortes.

5.1. Formulação matemática

O modelo de programação linear inteira descrito nesta seção generaliza duas das formulações propostas para o TUP na literatura. Em [14], 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 [14], e também em [13], 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 III, IV e V. 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 $S = \{1, 2, \ldots, \left\lceil \frac{4n-3}{w-1} \right\rceil\}$ 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 6 ilustra um torneio com quatro times e seis rodadas subdivididas em seções para $w = 2$, 3, 4 e 6.

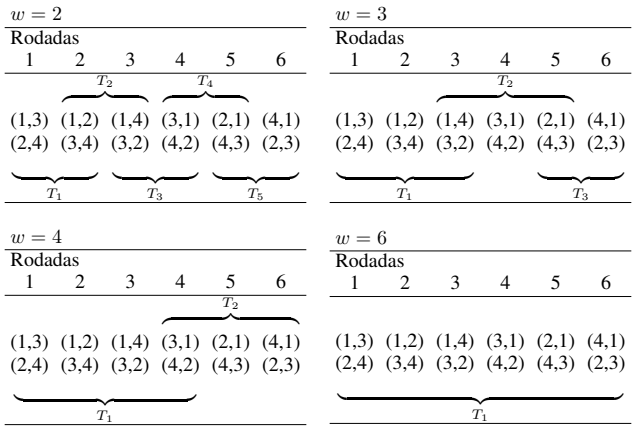


Figura 6. Seções de um torneio com 4 times e 6 rodadas para $w = 2$, 3, 4 e 6.

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 IV e V. Como existe somente uma seção quando $w = 4n-2$, também é exigido que as sequências de viagens obedeçam à restrição III neste caso particular. Quando $2 \leq w < 4n-2$, não é possível impor diretamente III. No entanto, note que seções consecutivas possuem uma rodada em comum (veja a figura 6), o que nos permite conectar as suas sequências de viagens para criar sequências maiores. A seguir, 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.

5.2. 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 IV e V. (Quando $w = 4n-2$, vamos ter somente P_1 e, neste caso, as sequências também deverão satisfazer a restrição III). 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 IV ou V, ou quando ela é uma rota completa e viola a restrição III. Por fim, vamos denotar o conjunto de todas as rotas infactíveis por $\mathbb{U}$. Agora estamos prontos para apresentar nossa formulação matemática.

$$\text{Minimizar} \sum_{p \in P} d_p x_p \tag{31}$$

Sujeito a:

$$\sum_{p \in P_{sg}} x_p = 1, \quad \forall\, s \in S, g \in G_s, \tag{32}$$

$$\sum_{p \in P(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}, \tag{33}$$

$$x_p \in \{0, 1\}, \quad \forall\, p \in P. \tag{34}$$

A função objetivo (31) minimiza a distância total viajada pelos árbitros. A restrição (32) 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 (32) e (34) juntas asseguram que uma solução factível seja formada por n rotas completas que satisfazem as restrições I e II. As

restrições III, IV e V são impostas por (33), 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 (32)–(34).

Esta formulação matemática com $w = 2$ e junto com a desigualdade (35), introduzida na próxima seção, é equivalente à formulação baseada em fluxo em rede apresentada em [14]. Quando $w = 4n - 2$, as desigualdades (33) e (35)–(37) (vistas a seguir) se tornam desnecessárias e esta formulação é equivalente à formulação baseada em partição de conjuntos introduzida em [13], [14] (sem os cortes adicionais apresentados em [14]).

5.3. Desigualdades válidas fortes

A relaxação linear de (32)–(34) não provê limitantes inferiores fortes, principalmente porque (33) é uma restrição fraca. Em [14], os autores apresentam uma forma de fortalecer (33) aplicando o *lifting* proposto em [22], 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, (35) é uma versão mais forte de (33).

$$\sum_{p \in H^+(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}. \tag{35}$$

A corretude de (35) vem do fato que, por construção, quaisquer $|P(U)|$ rotas simples em $H^+(U)$ que satisfazem (32) 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 [22] também podem ser aplicadas a (35).

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 (35) para a instância modificada, vamos obter (36), que é válida para $\mathcal{T}$ na instância original.

$$\sum_{p \in H^-(U)} x_p \leq |P(U)| - 1, \quad \forall\, U \in \mathbb{U}, \tag{36}$$

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 (35) e (36) são linearmente independentes e, portanto, não são redundantes juntas. Além disso, os resultados computacionais apresentados na seção 4.3 indicam que a inclusão de (36) fortalece consideravelmente a relaxação linear de (32)–(35).

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 (37) é válida para $\mathcal{T}$.

$$\sum_{p \mid v^A_{sg}(p) \in C} x_p \leq 1, \quad \begin{array}{l} \forall\, s \in S, s \neq |S|, \\ g \in G_s \cap G_{s+1}, C \in \mathbb{A}_{sg}. \end{array} \tag{37}$$

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 (38) é válida para $\mathcal{T}$ por causa de (32).

$$\sum_{p \mid v^B_s(p) \in C} x_p \leq 1, \quad \forall\, s \in S, C \in \mathbb{B}_s. \tag{38}$$

Vamos nos referir às restrições (35) e (36) como *desigualdades de caminho*, e às restrições (37) e (38) como *desigualdades de clique*.

5.4. Resultados computacionais

Desenvolvemos um algoritmo *branch-and-cut* para resolver a nova formulação proposta nesta seção para o TUP. O algoritmo enumera as variáveis *a priori* e as adiciona no modelo durante o início da execução. (O tempo gasto na enumeração em cada execução não excedeu 15 segundos em nossos experimentos.) Como a quantidade de desigualdades de caminho e clique cresce exponencialmente em função de n, apresentamos em [10] rotinas de sepação para estas famílias de desigualdades. O algoritmo *branch-and-cut* utiliza estas rotinas da seguinte forma. Iniciamos a otimização com um modelo contendo somente (31), (32) e (34), e executamos as rotinas de separação em cada nó da árvore de enumeração para adicionar desigualdades (35)–(37) na medida em que estas se tornarem violadas. A implementação do método 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. Mais detalhes sobre o algoritmo *branch-and-cut* desenvolvido são encontrados em [10].

Efetuamos em [10] experimentos computacionais para mostrar a relevância das desigualdades fortes propostas,

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* com os outros métodos da literatura. Por questões de espaço, vamos apresentar a seguir apenas a comparação dos resultados obtidos pelo *branch-and-cut* com os melhores resultados da literatura. O estudo experimental completo, contendo as escolhas de valores para w, as combinações das rotinas de separação das desigualdades e as demais análises é apresentado em [10].

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 [11] (veja a descrição desse *benchmark* na página 2). As instâncias com menos de 14 times foram desconsideradas pois elas são facilmente resolvidas pelos métodos atuais da literatura.

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. Comparamos na tabela 2, 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) [12], *branch-and-price* (BP) [13], *branch-and-bound* (BB) [14], *branch-and-price-and-cut* (BPC) [14], e *branch-and-bound* com limitantes inferiores baseados em decomposição (BB-LID) [15]. Em [12], 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 2. 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 2 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 2. 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 executados 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).

5.5. Conclusões

Introduzimos nesta seção um modelo de programação linear inteira parametrizado para o TUP que generaliza os dois melhores modelos existentes na literatura. Este modelo foi 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. O BC foi desenvolvido com o objetivo de resolver instâncias de tamanhos mais próximos daqueles encontrados em situações reais. Nossos resultados computacionais comprovam que o método escala melhor

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 2. COMPARAÇÃO ENTRE OS LIMITANTES INFERIORES OBTIDOS PELO BRANCH-AND-CUT E OS MELHORES LIMITANTES INFERIORES DA LITERATURA.

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 inferiores 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 visando avaliar ser haverá melhoria nos tempos execução.

6. Considerações finais

Neste doutorado, 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 [6]) 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*, vol. 236, no. 2, pp. 592–600, 2014.
2) L. Oliveira, C. C. Souza, e T. Yunes, "On the complexity of the traveling umpire problem," *Theoretical Computer Science*, vol. 562, pp. 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*, vol. 72, pp. 147–159, 2016.

Estes trabalhos são brevemente descritos nas seções 3, 4 e 5, e apresentados detalhadamente nas referências citada acima. Novos métodos de resolução, modelos matemáticos e uma prova de complexidade para o TUP foram introduzidos nestes trabalhos. Em resumo, os principais resultados obtidos com as pesquisas desenvolvidas 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 (seção 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 (seção 4);
- Acreditamos que esta formulação também contribuiu indiretamente para a obtenção de limitantes inferiores ainda melhores pelo método de decomposição proposto em [12], 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 (seção 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 deste texto (seção 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 das seções 3, 4 e 5.

Agradecimentos

Os autores agradecem ao CNPq (processo 142278/2013-0) e à CAPES (processo 01-P-01965-2012) pelo apoio financeiro.

Referências

[1] G. Kendall, S. Knust, C. C. Ribeiro, and S. Urrutia, "Scheduling in sports: An annotated bibliography," *Computers & Operations Research*, vol. 37, no. 1, pp. 1–19, 2010.

[2] M. J. Fry and J. W. Ohlmann, "Introduction to the special issue on analytics in sports, part ii: Sports scheduling applications," *Interfaces*, vol. 42, no. 3, pp. 229–231, 2012.

[3] R. V. Rasmussen and M. A. Trick, "Round robin scheduling – a survey," *European Journal of Operational Research*, vol. 188, pp. 617–636, 2008.

[4] M. A. Trick, H. Yildiz, and T. Yunes, "Scheduling major league baseball umpires and the traveling umpire problem," *Interfaces*, vol. 42, no. 3, pp. 232–244, 2012.

[5] M. A. Trick and H. Yildiz, "Benders' cuts guided large neighborhood search for the traveling umpire problem," *Naval Research Logistics*, vol. 58, no. 8, pp. 771–781, 2011.

[6] ——, "Benders' cuts guided large neighborhood search for the traveling umpire problem," in *Proceedings of the Fourth Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, ser. Lecture Notes in Computer Science, P. Van Hentenryck and L. Wolsey, Eds., vol. 4510. Springer-Verlag, 2007, pp. 332–345.

[7] ——, "Locally optimized crossover for the traveling umpire problem," *European Journal of Operational Research*, vol. 216, no. 2, pp. 286–292, 2012.

[8] L. Oliveira, C. C. Souza, and T. Yunes, "Improved bounds for the traveling umpire problem: A stronger formulation and a relax-and-fix heuristic," *European Journal of Operational Research*, vol. 236, no. 2, pp. 592–600, 2014.

[9] ——, "On the complexity of the traveling umpire problem," *Theoretical Computer Science*, vol. 562, pp. 101–111, 2015.

[10] ——, "Lower bounds for large traveling umpire instances: New valid inequalities and a branch-and-cut algorithm," *Computers & Operations Research*, vol. 72, pp. 147–159, 2016.

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

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

[13] T. A. M. Toffolo, S. V. Malderen, T. Wauters, and 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*, 2014, pp. 420–432.

[14] L. Xue, L. Z., and A. Lim, "Two exact algorithms for the traveling umpire problem," *European Journal of Operational Research*, vol. 243, no. 3, pp. 932–943, 2015.

[15] T. A. M. Toffolo, T. Wauters, S. V. Malderen, and G. V. Berghe, "Branch-and-bound with decomposition-based lower bounds for the traveling umpire problem," *European Journal of Operational Research*, vol. 250, no. 3, pp. 737–744, 2016.

[16] M. Bender and S. Westphal, "A combined approximation for the traveling tournament problem and the traveling umpire problem," *Journal of Quantitative Analysis in Sports*, vol. 12, no. 3, pp. 139–149, 2016.

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

[18] J. Dinitz, E. Lamken, and W. D. Wallis, "Scheduling a tournament," in *Handbook of Combinatorial Designs*, C. J. Colbourn and J. Dinitz, Eds. CRC Press, 1995, pp. 578–584.

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

[20] H. Yildiz, "Methodologies and applications for scheduling, routing & related problems," Ph.D. dissertation, Tepper School of Business, Carnegie Mellon University, 2008.

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

[22] B. Kallehauge, N. Boland, and O. B. G. Madsen, "Path inequalities for the vehicle routing problem with time windows," *Networks*, vol. 49, no. 4, pp. 273–293, 2007.