DEPARTAMENTO DE COMPUTAÇÃO

D E C O M

Programação de jogos de competições esportivas: Uma abordagem heurística – Parte II

Marcio Tadayuki Mine

Marcone Jamilson Freitas Souza
Orientador
Gustavo Peixoto Silva
Co-orientador

Relatório Final de Bolsa de Pesquisa FAPEMIG - PROBIC
Fevereiro / 2006

U F O P

UNIVERSIDADE FEDERAL DE OURO PRETO

Resumo

Este trabalho tem seu enfoque no Problema de Programação de Jogos de Competições Esportivas realizadas em dois turnos completos. Esse problema consiste na elaboração de uma escala de jogos entre os times participantes de uma competição esportiva que ocorre durante certo período, em vários locais e obedecem a um determinado conjunto de restrições. O objetivo principal do problema é minimizar a distância total percorrida pelos times durante a competição e, dessa forma, reduzir os custos com transporte. Deve-se, também, equilibrar as distâncias viajadas pelos times, cujo propósito é que nenhum time seja prejudicado injustamente pelo alto desgaste físico dos jogadores conseqüente da longa distância viajada. O Problema de Programação de Jogos de Competições Esportivas é de grande importância, pois as competições podem gerar lucro para os clubes, dado que grandes empresas patrocinam esses eventos como também redes de televisão dão cobertura completa dos mesmos. O número elevado de combinações para construir as tabelas, associado com a necessidade de atender às restrições impostas, torna-se complexa a elaboração de tabelas. Na literatura este problema é conhecido como *Sports Timetabling* ou *Travelling Tournment Problem* e pertence à classe de problemas NP-difíceis, o que dificulta, ou torna-se inviável, a resolução por métodos de programação matemática, ditos exatos, para casos reais ou instâncias muito grandes. Desta forma, o problema é normalmente abordado por técnicas heurísticas. Este trabalho apresenta técnicas heurísticas para a resolução do Problema de Programação de Jogos de Competições Esportivas do Campeonato Brasileiro de Futebol, bem como uma metodologia eficiente de geração de soluções iniciais, baseada em *backtracking*, que contempla a duas restrições desse problema. Para validar os métodos, são utilizados as instâncias disponíveis na página do orientador deste projeto de pesquisa, a saber, http://www.decom.ufop.br/prof/marcone/projects/ttp/bssp.html, relativas ao Campeonato Brasileiro de Futebol dos anos de 2004, 2005 e 2006. Resultados computacionais mostram que a heurística ILS-MRD, formada pela combinação das técnicas *Iterated Local Search* e Método Randômico de Descida é robusta e eficiente para a resolução do problema de programação de jogos do Campeonato Brasileiro de Futebol, além de superar significativamente as soluções produzidas pela Confederação Brasileira de Futebol, bem como superar uma solução existente na literatura para o Campeonato Brasileiro de Futebol de 2004.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 O Problema de Programação de Jogos

Para muitas competições esportivas (tais como futebol, futsal, voleibol, basquetebol), onde os jogos são disputados entre dois times e realizados em vários locais ao longo de um determinado período de tempo, há a necessidade de se fazer um bom escalonamento desses jogos. Esse problema, conhecido na literatura como Problema de Programação de Jogos de Competições Esportivas (*Travelling Tournament Problem*), consiste em construir uma tabela de jogos entre os times participantes de uma competição esportiva, realizados em vários locais ao longo de um determinado período de tempo e obedecendo a um conjunto de restrições.

Problemas dessa natureza contêm em geral muitas restrições que devem ser satisfeitas e diferentes objetivos a serem atingidos como, por exemplo, a minimização dos deslocamentos dos times durante o campeonato, a realização de apenas uma partida por rodada para cada time, a realização de determinados jogos em estádios e em datas pré-estabelecidas, o número máximo de partidas consecutivas realizadas na sede do time e fora dela, entre outras.

A geração de escalonamentos satisfatórios, respeitando essas restrições e objetivos, é um problema muito difícil de ser resolvido. A dificuldade de solução desse problema é devida ao grande número de possibilidades a serem analisadas. De acordo com Concílio e Zuben (2002), para uma competição envolvendo n times confrontando-se entre si em turnos completos, o número de combinações possíveis é dado pela fórmula (1):

$$(n-1)!(n-3)!(n-5)!...(n-(n-1))!\times 2^{(n-1)\times\frac{n}{2}} \quad (1)$$

Para exemplificar a magnitude do espaço de soluções, para uma competição com 20 participantes há 2,9062 x 10^{130} combinações possíveis.

Estudos têm sido realizados para tentar resolver esses problemas com uma variedade de abordagens: programação inteira, programação linear, programação por restrições, técnicas heurísticas e metaheurísticas.

Trata-se, portanto, de um problema de grande importância e interesse prático, uma vez que o bom escalonamento de uma tabela influencia não apenas os resultados dos jogos, mas também todo rendimento financeiro da competição. Conforme descrito anteriormente, conseguir um resultado satisfatório é uma tarefa muito difícil, mesmo quando realizada por um especialista ou pela utilização de ferramentas convencionais.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral desenvolver um método eficiente para resolver o problema de programação de jogos de competições esportivas realizadas em dois turnos completos, espelhados ou não. Entretanto, o foco principal é contribuir com a divulgação das técnicas de otimização aplicadas à resolução de problemas reais, como o Campeonato Brasileiro de Futebol (realizado em dois turnos espelhados), de forma que os organizadores desse evento (no caso, a Confederação Brasileira de Futebol) e de outros eventos esportivos reconheçam sua importância e possam aplicá-las em suas competições.

1.3 Estrutura do trabalho

O presente trabalho está dividido em sete capítulos, incluindo esta introdução.

No Capítulo 2 é apresentada uma revisão bibliográfica sobre as diversas metaheurísticas utilizadas na resolução do problema de programação de jogos bem como a forma com que diversos autores tratam esse problema. Apresenta-se também um resumo sobre as metodologias heurísticas *Simulated Annealing*, Busca Tabu, GRASP (*Greedy Randomized Adaptive Search Procedure*), VNS (*Variable Neighborhood Search*), VND (*Variable Neighborhood Descent*), o ILS (*Iterated Local Search*) e .o Método de Reconexão por Caminhos (*Path Relinking*).

O Capítulo 3 apresenta os dois problemas típicos: espelhado e não espelhado, e suas respectivas particularidades.

No Capítulo 4 é apresentado as metodologias adotadas para resolver o problema, a saber: como representar uma solução, quais os tipos de movimentos realizados nos dois problemas, as estruturas de vizinhança, como gerar uma solução inicial, como avaliar uma solução, os métodos *Simulated Annealing*, VND, VNS/VND, Busca Tabu, ILS o método de Reconexão por Caminhos aplicados aos problemas e o funcionamento básico de cada algoritmo.

No capítulo 5 são apresentados os resultados obtidos.

No capítulo 6 é informada a situação atual desse projeto de pesquisa.

No capítulo 7 são apresentadas as conclusões e os trabalhos futuros.

2 Revisão Bibliográfica

2.1 Introdução

O Problema de Programação de Jogos de Competições Esportivas (PPJ), conhecida na literatura como *sports timetabling ou Traveling Tournament Problem* (TTP) é um problema pertencente à classe de problemas NP-difíceis. Dessa forma, a obtenção da melhor solução existente por meio de métodos de programação matemática, dito exatos, está limitada, via de regra, a problemas de pequenas dimensões. Para dimensões mais elevadas, a abordagem mais comum é através de técnicas heurísticas. Essas técnicas apresentam como grande vantagem a flexibilidade na manipulação do problema, permitindo incluir, com maior facilidade, qualquer espécie de restrição do problema. A principal desvantagem dessas técnicas é a impossibilidade de garantir a otimalidade da solução final gerada. Entretanto, com o aparecimento das chamadas metaheurísticas, ou heurísticas inteligentemente flexíveis, tais como Busca Tabu (Glover, 1986; Glover, 1989, Glover, 1990), *Simulated Annealing* (Kirkpatrick *et al*., 1983), GRASP (Feo e Resende, 1995), Método de Pesquisa em Vizinhança Variável (Mladenovic e Hansen, 1997) e Algoritmos Genéticos (Goldberg, 1989), que possuem mecanismos para escapar de ótimos locais, muito progresso se tem conseguido com o objetivo de melhorar a qualidade da solução final gerada.

Em vista desse fato, este problema é normalmente abordado por técnicas heurísticas. Concílio (2000) e Concílio e Zuben (2002) abordam o problema da montagem da tabela de jogos do Campeonato Paulista de Futebol de 1997, que envolveu 16 times, utilizando programação genética. As restrições abordadas nesse trabalho são: (a) cada participante deve enfrentar todos os seus adversários uma única vez durante o torneio; (b) todos os participantes devem jogar em todas as rodadas; (c) a diferença do número de jogos de cada participante em seu domínio e fora dele é no máximo um; (d) o número máximo de jogos consecutivos no domínio do participante (ou fora) não é superior a um número *r*. Na função de aptidão de uma escala leva-se em consideração se todos os participantes estão percorrendo uma distância parecida ao longo das rodadas, de forma a não privilegiar este ou aquele participante em termos de custo e tempo de deslocamento. Para avaliar esta parcela da função de aptidão toma-se o inverso da diferença entre as distâncias máxima e mínima percorridas pelos participantes do torneio, considerando todas as rodadas.

Nemhauser e Trick (1998) tratam o problema da alocação de jogos de uma competição de basquetebol, realizada em turno único, envolvendo 9 times de diferentes localidades dos Estados Unidos. O procedimento proposto contém 3 fases e utiliza uma combinação de uma técnica de programação inteira e uma técnica enumerativa. A primeira fase consiste em gerar um conjunto de padrões de cardinalidade igual ao número *n* de times. Cada padrão é uma seqüência de *n*-1 letras, sendo que cada letra pode assumir um dentre dois valores, a saber, casa (C) ou fora (F). Na segunda fase os jogos são designados aos padrões, sem se estabelecer quais são os times realmente envolvidos em cada jogo. Finalmente, na última fase, os times são designados aos padrões, produzindo uma escala. As duas primeiras fases são resolvidas por técnicas de programação matemática, enquanto a última é resolvida por uma técnica de enumeração implícita.

Henz (2001) melhorou o procedimento anterior aplicando a técnica de programação por restrições a todas as três etapas. Contudo, essa metodologia se restringia a problemas de pequenas dimensões.

Trick (2001) apresenta um método de duas fases para resolver o problema. Na primeira fase são alocados jogos sem levar em consideração o mando de campo de um jogo. Na segunda fase, o mando de campo é considerado, sendo respeitado o limite estabelecido de jogos em casa e fora de casa. As duas fases são resolvidas por uma combinação das técnicas de programação por restrições e programação inteira. O algoritmo proposto resolve eficientemente problemas envolvendo até 20 times. Entretanto, o modelo tratado não considera o caso de competições realizadas em dois turnos, mecanismo de disputa freqüentemente utilizado em várias modalidades de competições esportivas, entre elas as latinas.

Biajoli *et al.* (2003) utilizaram *Simulated Annealing* com duas estruturas de vizinhança para resolver o problema de programação de jogos da primeira divisão do Campeonato Brasileiro de Futebol de 2002. Essa competição envolveu 26 times e foi realizada em turno único em 29 rodadas. Comparando as soluções obtidas com a solução gerada manualmente pela entidade organizadora do evento, verificou-se uma melhora de até 11,4% em relação ao deslocamento dos times.

Biajoli (2003) estendeu o algoritmo *Simulated Annealing*, anteriormente desenvolvido para tratar competições em turno único, adaptando as duas estruturas de vizinhança para resolver o problema relativo à primeira divisão do Campeonato Brasileiro de Futebol de 2003, competição realizada em dois turnos completos e envolvendo 24 clubes. O algoritmo desenvolvido utiliza um procedimento de reaquecimento para escapar de mínimos locais em baixas temperaturas. Os resultados obtidos foram 9,6% melhores do que aqueles gerados manualmente pela Confederação Brasileira de Futebol (CBF), o que possibilitaria economias da ordem de R$ 1.760.000,00 nos gastos com deslocamento dos times.

Várias outras modalidades de competições esportivas também foram objeto de investigação. Por exemplo, problemas relacionados a competições de beisebol foram abordados em Russel e Leung (1994) e Cain (1977). Schreuder (1992) analisa aspectos combinatoriais na construção da tabela de jogos do Campeonato Alemão de Futebol. Bean e Birge (1980) e Campbell e Chen (1976) são exemplos de outras abordagens para tratar problemas de competições de basquetebol. Outros autores, tais como Schaefer (1999) e de Werra (1988), exploram o tema sem considerar uma aplicação específica.

Para comparar os diversos métodos propostos na literatura, foi introduzido por Easton *et al.* (2001) e Dorigo *et al.* (2001) um conjunto de problemas-teste referente a um problema padrão de programação de jogos de competições esportivas, denominado *Traveling Tournament Problem* (TTP). Esses problemas foram baseados na liga americana de beisebol, sendo alguns deles reais e outros, variações desses. As restrições são as seguintes: (a) cada time não pode jogar mais do que 3 jogos consecutivos dentro ou fora de casa; (b) as competições são realizadas em dois turnos completos; (c) O jogo de um time i na casa do time j não pode ser seguido por um jogo do time j na casa do time i. O objetivo é minimizar a distância total percorrida por todos os times ao longo da competição.

Recentemente, foi incluído um novo conjunto de problemas-teste, introduzido por Ribeiro e Urrutia (2004a), considerando o caso espelhado, na qual os competidores jogam duas vezes, uma vez em cada turno, sendo o segundo turno espelho do primeiro. Observa-se, entretanto, que tal conjunto de problemas não retrata restrições comuns a competições esportivas latinas, como as brasileiras. Nessas competições, por exemplo, não se permite que times de um mesmo estado joguem na última rodada de cada turno.

Crauwels e Van Oudheusden (2003) desenvolveram um procedimento baseado em Colônia de Formigas para tratar o TTP não espelhado. Nesse procedimento, cada time é representado por uma formiga, a qual deixa uma certa quantidade de "feromônio" sempre que completa uma rota, isto é, sempre que termina todos os seus jogos. A quantidade de

feromônio deixada em um arco da rota é inversamente proporcional à distância percorrida. Arcos com maior quantidade de feromônio são estimulados a serem freqüentados por outras formigas. Após uma solução viável ser obtida, é realizado um procedimento de busca local que utiliza duas estruturas de vizinhança, aplicadas na forma do Método de Descida em Vizinhança Variável (Mladenovic e Hansen, 1997). Apesar de o método desenvolvido ter alcançado a melhor solução em um dos problemas-teste e aproximado de outro, para os demais problemas os resultados obtidos não foram satisfatórios, ficando muito distantes da melhor solução encontrada na literatura, mesmo com a utilização da fase de intensificação proposta.

A seguir, descrevem-se, com mais detalhes, os trabalhos encontrados na literatura que foram importantes para o início do desenvolvimento da pesquisa.

Costa (1995) desenvolveu um método híbrido combinando as técnicas Algoritmos Genéticos e Busca Tabu para resolver um problema de escalonamento de jogos da Liga Americana de Hóquei. Noções de Inteligência Artificial são introduzidas para direcionar o processo de evolução natural. A fase de mutação do Algoritmo Genético é trocada por uma busca no espaço de solução comandado pelo método Busca Tabu. Ao invés de uma mutação aleatória mudar os componentes, cada indivíduo é submetido a um processo de otimização separado antes de interagir novamente com outros membros da população. Segundo o autor, a junção desses dois algoritmos foi feita pelo fato do Busca Tabu possuir muitos parâmetros para serem ajustados, enquanto os métodos baseados em Algoritmos Genéticos possuem poucos parâmetros para serem ajustados e lidam com um conjunto de soluções ao mesmo tempo, e não com uma única como faz a Busca Tabu. A Figura 2.1 esquematiza as fases do Algoritmo Evolutivo Busca Tabu (EBT).

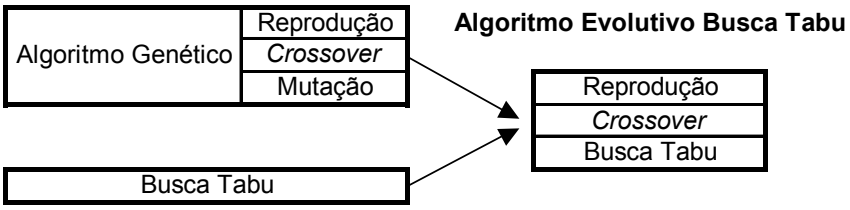

Figura 2.1 – Busca Tabu combinado com Algoritmo Genético

O EBT é constituído de três fases, conforme mostra a Figura 2.1, repetidas sucessivamente por todo o processo.

A geração da população inicial de escalas é feita de forma determinística.

Após a construção da população inicial, três fases são aplicadas, a saber: a fase de reprodução, a fase de *crossover* e a aplicação do Busca Tabu.

A fase de reprodução é m mecanismo artificial baseado na seleção natural onde os indivíduos mais fracos morrem e os mais fortes proliferam. Durante a fase de reprodução um pedaço do material genético que compõem o DNA de uma população é copiado de acordo com o valor da sua função objetivo. O número esperado desse material genético numa nova população é proporcional à sua aptidão. Assim, futuras gerações serão cada vez mais aptas.

A fase de *crossover* constitui na fase que ocorre a troca de informações que produz diversidade e inovação na população. Pedaços de diferentes códigos genéticos se juntam aleatoriamente para criar novos descendentes. Cada descendente criado terá características derivadas de ambos os pais. Existem diferentes maneiras de se fazer *crossover*. Uma

possibilidade baseia-se em utilizar duas posições p_1 e p_2 dos pais. Dois indivíduos novos serão criados trocando as informações contidas entra as posições p_1 e p_2.

A terceira e última fase consiste na aplicação do algoritmo Busca Tabu, que procura repetidamente por um vizinho melhor possível da solução corrente. Isso faz com que áreas interessantes do espaço de busca sejam analisadas rapidamente e com grande probabilidade de se encontrar bons resultados.

A avaliação de uma solução *s* é feita através da seguinte função objetivo *f*, baseada em penalização, apresentada pela fórmula (2).

$$f(s) = \sum_{i \in C_b} w_i \times f_i(s) \tag{2}$$

onde C_b é o conjunto de restrições a serem cumpridas e que foram relaxadas, w_i é a penalidade aplicada a cada vez que a restrição do tipo *i* é desrespeitada e f_i é a função que calcula o grau de violação da *i*-ésima restrição relaxada na escala *s*.

As restrições consideradas nesse trabalho são as seguintes:

a) cada time aponta 56 datas para poderem jogar em casa, sendo que apenas 41 dessas datas serão de fato escolhidas;

b) a distância total percorrida pelos times ao longo da competição deve ser minimizada;

c) um time não pode jogar mais do que dois dias e nem mais que três jogos em 5 dias consecutivos;

d) um time não pode ficar fora de seu domínio mais do que 14 dias seguidos;

e) um time não pode jogar mais de um jogo por dia;

f) um time deve jogar regularmente enquanto não está em sua sede. Descanso de mais de três dias consecutivos devem ser evitados quando se está jogando fora de casa;

g) Em dias certos, como Natal, Reveillon, não podem acontecer rodadas do campeonato;

h) Alguns estádios não podem sediar jogos em determinados dias por motivos de shows, festas, circo nos mesmos;

i) Um time não pode jogar mais que 7 jogos fora de casa;

j) Dois jogos não podem ser escalonados em dois dias consecutivos se um dos times envolvidos viajar mais que 900 milhas.

Dessas restrições, as restrições (b), (c), (e), (f) e (h) foram relaxadas e as funções *f* que avaliam essas restrições são as seguintes:

- $f_1(s)$ indica quantas vezes um time joga mais de um jogo por dia;
- $f_2(s)$ indica quantas vezes a indisponibilidade do estádio não foi respeitada;
- $f_3(s)$ indica a distância total (em milhares de milhas) percorrida pelos times durante a temporada;
- $f_4(s)$ indica quantas vezes um time joga mais de dois jogos em três dias seguidos ou mais de três jogos em cinco dias seguidos;

- $f_5(s)$ indica quantas paradas de mais de três dias consecutivos ocorrem enquanto um time está viajando.

Na definição da função objetivo, o autor considerou os seguintes pesos: $w_1 = 26$, $w_2 = 13$, $w_3 = 1$, $w_4 = 10$ e $w_5 = 2$.

Anagnostopoulos *et al.* (2003) utilizaram a metaheurística *Simulated Annealing* (Kirkpatrick *et al.*, 1983) com estratégias de oscilação e reaquecimento para resolver o problema de alocação de jogos da Liga Americana de Beisebol.

A representação utilizada é a descrita no Capítulo 4.2 e uma solução *s* é avaliada através da complexa função objetivo *C*(*s*), apresentada pela fórmula (3).

$$C(s) = \begin{cases} \sum_{i \in T} custo(i) & \text{se a solução } s \text{ for viável} \\ \sqrt{\sum_{i \in T} custo(i)^2 + \left[w \times f\left(nbv(s)\right)\right]^2} & \text{se a solução } s \text{ for inviável} \end{cases} \quad (3)$$

onde a função *custo* é a função apresentada na Seção 4.4, *T* é o conjunto de todos os times, *w* é o parâmetro denominado peso, que representa a penalidade aplicada a cada inviabilidade na solução, f é a função $f = 1 + \sqrt{v} \ln v / 2$, onde $v = nbv(s)$, estabelecida para que o valor da função objetivo de soluções com muitas inviabilidades não cresça tão lentamente. A função *nbv* é $nbv = inv_1 + inv_2$, onde inv_1 e inv_2 é o número de vezes que ocorre a inviabilidade do tipo:

inv_1: Um time não pode jogar mais de três vezes consecutivas dentro ou fora de casa.

inv_2: Um jogo T_i *x* T_j, realizado na sede do time *i* não pode ser precedido de seu jogo inverso (T_j *x* T_i, realizado na casa do time T_j).

Para gerar soluções vizinhas, utilizam-se cinco estruturas de vizinhança, em que cada uma implementa um dos seguintes movimentos:

Swap Homes: movimento que troca o mando de campo de um jogo.

Swap Rounds: movimento que troca todos os jogos de duas rodadas quaisquer.

Swap Teams: movimento que troca todos os jogos de dois times T_i e T_j de todas as rodadas, exceto aquelas em que eles se confrontam.

Partial Swap Rounds: movimento que troca os jogos nas rodadas R_i e R_j de um time T_i.

Partial Swap Teams: movimento que troca os jogos dos times T_i e T_j em uma rodada R_i.

Conforme foi citado anteriormente, a metaheurística utilizada é a *Simulated Annealing* (Kirkpatrick *et al.*, 1983) que implementa um mecanismo de reaquecimento de temperatura, cujo propósito é escapar de ótimos locais a baixas temperaturas, já que, nessas temperaturas, a probabilidade de aceitação de soluções piores são remotas. Outra estratégia implementada é a estratégia de oscilação, que consiste em variar o peso *w*, alternando, dessa forma, a análise do espaço de soluções viáveis e inviáveis.

Biajoli *et al.* (2004) propõem a metaheurística *Simulated Annealing* para resolução da programação de jogos para a Primeira Divisão do Campeonato Brasileiro de Futebol de 2004, uma especialização do Problema de Programação de Jogos (PPJ).

Para o problema abordado, realizado em dois turnos, sendo o segundo "espelho" do primeiro, as seguintes restrições são levadas em consideração:

1. Cada time joga somente uma vez por rodada;
2. Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos;
3. Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos sendo um em casa e o outro na casa do adversário;
4. As duas últimas rodadas de cada turno terão a configuração inversa das duas primeiras rodadas de cada turno com relação ao mando de campo;
5. Dois times do mesmo estado não poderão se confrontar na última rodada do campeonato;
6. A diferença entre os jogos realizados por um time em cada turno em casa e fora de casa de um time não pode ser maior que 1;
7. Um time não pode jogar mais que duas vezes consecutivas dentro ou fora de casa.

Na aplicação do *Simulated Annealing*, o algoritmo inicia com uma escala gerada de forma aleatória que é obtida usando uma simples procura utilizando *backtracking*. As próximas etapas seguem o esquema do algoritmo tradicional, isto é, dado uma temperatura *t*, é selecionado aleatoriamente um movimento de vizinhança e calcula-se a variação da função objetivo principal. Três movimentos diferentes foram definidos nesse trabalho para definir tipos diferentes de vizinhança, a saber:

1. Troca do mando de campo de ambos os jogos do primeiro e segundo turno entre dois times;
2. Troca de dois jogos diferentes entre duas rodadas do mesmo turno;
3. Troca de todos os jogos de um time pelos jogos de outro time.

Após a aplicação do *Simulated Annealing,* uma escala é devolvida com solução. A esta solução aplica-se um procedimento de refinamento da mesma trocando os times de diferentes cidades entre si. A única restrição que deve ser analisada nesse processo é a restrição de jogos entre times do mesmo estado na última rodada. Esse refinamento melhorou a solução final em mais de 2% em poucos minutos.

Em relação à escala feita manualmente, a escala automática provou a eficiência do algoritmo reduzindo em 12,8% o total dos custos envolvido na competição e balanceou a distância viajada por todos os times melhorando esse balanceamento em 38,4%.

Ribeiro e Urrutia (2004) desenvolveram um método híbrido utilizando as metodologias GRASP (Feo e Resende, 1995) e o *Iterated Local Search* (Lourenço *et al.*, 2002) para resolver um problema de alocação de jogos de competições esportivas.

Nesse trabalho, uma solução é representada por fatorações 1-orientadas ordenadas (*ordered oriented 1-factorizations*) de um grafo completo não direcionado. Cada nó deste grafo representa um time. Um arco do nó *i* para o nó *j* no *k*-ésimo *1-fator* das fatorações 1-orientadas ordenadas representa um jogo entre os times *i* e *j* sediado na cidade do time *j* na rodada *k*.

Para o problema abordado foram consideradas as quatro estruturas de vizinhança descritas a seguir:

Home-away swap (HAS): estrutura de vizinhança que inverte o mando de campo de um jogo.

Team swap (TS): estrutura de vizinhança que troca todos os jogos de dois times em todas as rodadas.

Partial swap rounds (PSR): seja t_1, t_2, t_3 e t_4 quarto times e r_1 e r_2, duas rodadas em que os jogos $t_1 \times t_3$ e $t_2 \times t_4$ estão alocados na rodada r_1 e os jogos $t_1 \times t_4$ e $t_2 \times t_3$ estão alocados na rodada r_2. O movimento nessa estrutura de vizinhança consiste em trocar as rodadas em que esses jogos ocorrem. Ou seja, os jogos $t_1 \times t_4$ e $t_2 \times t_3$ serão realizados na rodada r_1 e os jogos $t_1 \times t_3$ e $t_2 \times t_4$ serão realizados na rodada r_2.

Game rotation (GR): nesta estrutura, um vizinho é gerado a partir de um movimento que força a alocação de um jogo arbitrário em uma rodada qualquer. Em seguida, a solução é atualizada de uma forma determinística (através do movimento nomeado como *ejection chain move*).

Na fase de construção GRASP é utilizado uma heurística constituída de 3 fases:

Na primeira fase, uma escala é construída para o problema de turno único, com n times abstratos. Para isso, utiliza-se o método do polígono conforme descrito a seguir.

Inicialmente, os $n-1$ times abstratos são colocados consecutivamente em sentido horário nos nós numerados de um polígono regular com $n-1$ nós. Os times 1,2,...,n-1 são colocados nos nós 1,2,...,n-1 respectivamente, enquanto que o time n não é inserido no polígono. A cada rodada $k = 1,2,...,n-1$ o time localizado no nó $l = 2,...,n/2$ joga contra o time localizado no nó $n + 1 - l$. Para toda rodada, o time localizado no nó 1 joga contra o time n. Depois de cada rodada, cada time 1,2,...,n-1 é movido em sentido horário para o próximo nó do polígono. A Figura 2.2 ilustra a aplicação deste método para $n = 6$.

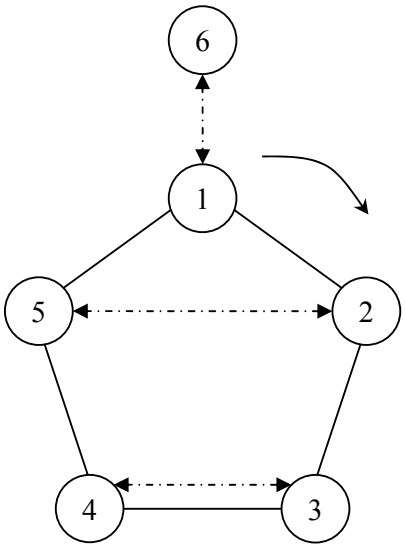

Figura 2.2 – Método do polígono para a primeira rodada com n = 6.

Em seguida, a escala é duplicada. Essa escala é utilizada para gerar uma matriz quadrada n x n de oponentes consecutivos. Cada célula (i,j) da matriz contém o número de vezes em que os times i e j são oponentes consecutivos dos outros times. A Figura 2.3 mostra uma matriz de oponentes consecutivos para uma escala produzida pelo método do polígono para $n = 16$ times abstratos. Esta figura mostra que o método do polígono cria diversos padrões na escala. Como exemplo, nota-se que os times 8 e 10 são oponentes consecutivos de outros times 26 vezes ao longo do torneio.

$$\begin{pmatrix}
0 & 4 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 4 & 3 & 4 & 4 & 4 & 4 & 4 \\
4 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 \\
4 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \\
4 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 26 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 1 & 26 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 1 & 0 & 2 & 25 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 2 & 0 & 2 & 25 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 \\
4 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 \\
4 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0
\end{pmatrix}$$

Figura 2.3 – Matriz de oponentes consecutivos para $n = 16$.

Na segunda fase, os times reais, cuja distância entre suas sedes são pequenas, são preferencialmente associados aos times abstratos que possuem maior número de oponentes consecutivos, conforme a heurística descrita a seguir.

Primeiro, os pares de times abstratos são ordenados em ordem decrescente de acordo com o número de oponentes consecutivos que possuem. Depois, os times reais são ordenados em ordem crescente de distância em relação aos seus oponentes e são associados aos times abstratos nesta ordem. Seja t_1 o próximo time real a ser associado a um time abstrato e seja *t2* o time, cuja cidade é mais próxima à cidade de t_1:

Se t_2 já foi associado a um time abstrato, então encontre o primeiro par (a,b) de times abstratos, tal que a esteja designado ao time t_2 e b não esteja ainda designado a nenhum time real. Nesse caso, associa-se o time abstrato b ao time real t_1.

Se t_2 ainda não foi associado a um time abstrato, então encontre o primeiro par (a,b) de times abstratos, tal que nenhum deles esteja associado a um time real. Nessa situação, associa-se tanto a quanto b ao time real t_1.

Na terceira fase, é estabelecido o local de realização de cada jogo. Para esta fase é proposto um método de dois passos.

No primeiro passo deste método constrói-se uma escala viável, enquanto que o segundo passo tenta melhorar a escala aplicando-se um método de busca local para reduzir a distância percorrida. Na primeira rodada, os mandos de campo são associados aleatoriamente. Nas rodadas seguintes até a penúltima rodada $(n\text{-}2)$ do primeiro turno é utilizada uma estratégia para determinar o mando de campo de um jogo entre os times t_1 e t_2. Seja n_i o número de jogos que o time i jogou consecutivamente, tanto dentro ou fora de casa, nas rodadas anteriores.

Caso $n_{t2} > n_{t1}$:

Se o time t_2 jogou seu último jogo em casa, então a realização deste jogo será na sede de t_1.

Caso contrário, a realização do jogo será na sede do time t_2.

Caso $n_{t2} < n_{t1}$:

Se o time t_1 jogou seu último jogo em casa, então a realização deste jogo será na sede de t_2.

Caso contrário, a realização do jogo será na sede do time t_1.

Caso $n_{t2} = n_{t1}$:

Se o time t_2 jogou seu último jogo em casa e o time t_1 jogou fora de casa, então a realização deste jogo será na sede de t_2.

Se o time t_1 jogou seu último jogo em casa e o time t_2 jogou fora de casa, então a realização deste jogo será na sede de t_1.

Caso contrário, o local da realização do jogo será determinada aleatoriamente.

O mando de campo da última rodada do primeiro turno é estabelecido de forma aleatória.

A primeira rodada da segunda fase do problema de alocação de jogos realizados em dois turnos espelhados é exatamente a primeira rodada da primeira fase com mando de campo invertido. De forma semelhante, a última rodada do primeiro turno e a primeira rodada do segundo turno são consideradas consecutivas. Os estádios são alocados aleatoriamente na última rodada. Se a alocação de um estádio tornar a escala inviável, então o mando de campo para o jogo correspondente é invertido. Se a escala ainda permanece inviável, todos os passos anteriores são repetidos e uma nova alocação de estádios é realizada para evitar o conflito.

Finalmente, a escala obtida é submetida através de simples processo de busca local e é retornada como a solução da fase de construção do GRASP.

Na fase de busca local do GRASP é utilizado a metaheurística ILS, que começa de uma solução ótima local viável. A partir dessa solução é gerado um vizinho aleatório pertencente à estrutura de vizinhança GR descrita anteriormente. Este vizinho é submetido a um processo de busca local baseado na metaheurística VND, que utiliza as três estruturas de vizinhança HAS, TS e PRS desenvolvidas.

2.2 Metaheurísticas

Metaheurísticas são procedimentos de busca local, com capacidade de escapar de ótimos locais, destinados a resolver aproximadamente um problema de otimização.

As metaheurísticas diferenciam-se entre si basicamente pelas seguintes características:

a) critério de escolha de uma solução inicial;

b) definição da vizinhança $\boldsymbol{N}(s)$ de uma solução s;

c) critério de uma solução vizinha dentro de $\boldsymbol{N}(s)$;

d) critério de término;

Esses procedimentos são baseados na noção de vizinhança. Para definirmos o que é uma vizinhança, seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções s', denominado *vizinhança* de s. Cada solução $s' \in N(s)$ é chamada de *vizinho* de s e é obtido de s a partir de uma operação chamada de *movimento*.

Em linhas gerais, esses métodos começam de uma solução inicial s_0, navegam pelo espaço de busca, através do movimento, passando de uma solução para outra que seja sua vizinha.

A seguir, são apresentadas as metaheurísticas referenciadas ao longo deste trabalho.

2.2.1 *Simulated Annealing*

O *Simulated Annealing* (SA) é uma metaheurística, proposto originalmente por Kirkpatrick *et al*., em 1983, que simula um sistema de resfriamento (como, por exemplo, de chapas metálicas) aceitando movimentos de piora, segundo uma função probabilística.

Este método inicia-se a partir de uma solução inicial aleatória s_0 e a uma determinada temperatura T_0, suficientemente alta, a qual é diminiuida gradativamente até que seja alcançada uma temperatura extremamente baixa, dita temperatura de congelamento.

A cada temperatura, é efetuado um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura, denotado por *SAmax*) de um procedimento que consiste em gerar uma solução vizinha *s'* a solução corrente *s* e, em seguida, é avaliado o valor da variação da função objetivo, denominado $\Delta = f(s') - f(s)$.

Considerando um problema de minimização, caso o movimento seja de melhora, isto é, $\Delta < 0$, então o movimento é aceito e a solução vizinha *s'* torna-se a solução corrente.

Caso $\Delta \geq 0$, ou seja, o movimento é de piora, a solução vizinha é aceita com uma probabilidade $e^{-\Delta/T}$, onde *T* é a temperatura atual do sistema.

Esta probabilidade regula a aceitação de movimentos de piora, sendo que em altas temperaturas há maiores chances de se aceitar uma solução pior, e conseqüentemente, aumenta-se a chance de escapar de ótimos locais. À medida que a temperatura é diminuída, o método tende a aceitar movimentos de melhora com maior freqüência, convergindo, assim, a um ótimo local.

Atingido o equilíbrio térmico, a temperatura é diminuída, segundo uma razão de resfriamento α, $\{\alpha \in \Re \mid 0 < \alpha < 1\}$.

O método pára quando a temperatura de congelamento é atingida, ou seja, quando $T > \varepsilon$, com ε suficientemente próximo de zero.

A Figura 2.4 apresenta o pseudocódigo do *Simulated Annealing* básico, cujo objetivo é minimizar uma função objetivo *f*.

1) Gere uma solução aleatória s_0
2) Faça $s^* \leftarrow s_0$, onde s^* é a melhor solução encontrada até então.
3) Defina a temperatura inicial do sistema, T_0
4) enquanto $(T > \varepsilon; \varepsilon \to 0)$, faça
   a) enquanto $(iterT > SA\max)$, faça
      i) $iterT \leftarrow iterT + 1$
      ii) Gere aleatoriamente um vizinho $s' \in N(s)$
      iii) Calcule a variação da função objetivo, $\Delta = f(s') - f(s)$
      iv) Se $\Delta < 0$ (movimento de melhora), então $s \leftarrow s'$.
      v) Caso contrário ($\Delta \geq 0$), gere aleatoriamente um valor $x \in [0,1]$ e aceite o movimento somente se a expressão $x < e^{-\Delta/T}$ for satisfeita.
      vi) Se $f(s') < f(s^*)$, então atualize s^*
   b) Reduza a temperatura por um fator α, ou seja, $T \leftarrow \alpha \times T$
   c) $iterT \leftarrow 0$
5) Retorne a melhor solução encontrada, s^*.

Figura 2.4 – Pseudocódigo do algoritmo *Simulated Annealing*.

2.2.2 Busca Tabu

A metaheurística Busca Tabu (BT) é um procedimento adaptativo que guia uma heurística de descida a continuar a explorar um espaço de soluções, sejam viáveis ou inviáveis, de modo a evitar a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado.

Para guiar a busca, é utilizada uma estrutura de memória, denominada de Lista Tabu, que armazena um conjunto de movimentos proibidos, chamados de movimentos tabu. Esses movimentos são os últimos movimentos efetuados nas últimas iterações do método.

Basicamente, o procedimento BT desloca-se de uma solução corrente *s* para seu vizinho *s'*, o qual é o melhor vizinho pertencente a um determinado conjunto de toda a vizinhança *N(s)*. O movimento reverso ao que gerou o melhor vizinho é, então, adicionado na lista tabu.

A lista tabu clássica contém os movimentos reversos aos últimos $|T|$ movimentos realizados (onde $|T|$ é um parâmetro do método) e funciona como uma fila de tamanho fixo, isto é, quando um novo movimento é adicionado à lista, o mais antigo sai. Assim, na exploração do subconjunto *V* da vizinhança *N*(*s*) da solução corrente *s*, ficam excluídos da busca os vizinhos que são obtidos de *s* por movimentos que constam na lista tabu.

Os vizinhos serão aceitos se os movimentos responsáveis pela sua geração não estejam presentes na lista tabu. No entanto, se o movimento for tabu e satisfizer a expressão $f(s') \leq A(f(s))$, então ele ainda assim poderá ser aceito. A função *A(.)* é denominada função de aspiração, cujo propósito é evitar que os movimentos pertencentes a lista tabu não prejudiquem, de forma significativa, a exploração do espaço de soluções. Desta forma, a função de aspiração é um mecanismo capaz de remover o *status* tabu de um movimento garantindo, assim, o progresso do método. Como um exemplo de uma função de aspiração, temos $A(s) = f(s^*) - 1$. Assim, o método aceitará um movimento tabu somente se ele conduzir a um vizinho melhor que s^*.

Duas regras são normalmente utilizadas de forma a interromper o procedimento. Pela primeira, pára-se quando é atingido um certo número máximo de iterações sem melhora no valor da melhor solução. Pela segunda, quando o valor da melhor solução chega a um limite inferior f_{min} conhecido (ou próximo dele). Esse segundo critério evita a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é julgada suficientemente boa.

Os parâmetros principais de controle do método de Busca Tabu são a cardinalidade $|T|$ da lista tabu, a função de aspiração A, a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração e *BTmax*, o número máximo de iterações sem melhora no valor da melhor solução.

A Figura 2.5 apresenta o pseudocódigo do algoritmo Busca Tabu para um problema de minimização.

1) Gere uma solução inicial s_0 e faça $s^* \leftarrow s_0$, onde s^* é a melhor solução encontrada até então.
2) $iter \leftarrow 0$, onde *iter* é o contador do número de iterações efetuadas pelo método.
3) $melhorIter \leftarrow 0$, onde *melhorIter* é a iteração mais recente que gerou s^*.
4) $TList \leftarrow \emptyset$, onde *TList* é a lista tabu
5) Inicialize a Função de Aspiração *A(.)*
6) enquanto $\left(\left(f(s) > f_{\min}\right) \wedge \left(iter - melhorIter < BT \max\right)\right)$ faça
   a) $iter \leftarrow iter + 1$
   b) Seja $s' \leftarrow s \oplus m$ o melhor vizinho da vizinhança *N(s)*, tal que o movimento $m \notin TList$ ou atenda a função de aspiração *A(s)*.
   c) Atualize a lista tabu: *TList* ← *TList* - {*movimento mais antigo*} + {*movimento que gerou s'*}
   d) Atualize a função de aspiração *A(.)*
   e) Mova a solução corrente para a solução vizinha: $s \leftarrow s'$
   f) Se $f(s') < f(s^*)$, então faça $s^* \leftarrow s'$ e $melhorIter \leftarrow iter$
7) Retorne a melhor solução s^*

Figura 2.5 – Pseudocódigo do algoritmo Busca Tabu.

2.2.3 GRASP

GRASP (*Greedy Randomized Adaptive Search Procedure* - Procedimento de Busca Adaptativa Gulosa e Randomizada) é um método iterativo proposto por Feo e Resende (1995), constituído basicamente de duas fases: uma fase de construção e uma fase de busca local, cujo objetivo é convergir à solução encontrada na fase de construção para um ótimo local. A Figura 2.6 apresenta o pseudocódigo básico do método GRASP.

1) $f(s^*) \leftarrow \infty$
2) faça *GRASPmax* iterações de:
   a) *fase de construção*: construa uma solução *s* utilizando uma heurística parcialmente gulosa
   b) *fase de busca local*: submeta *s* a um procedimento de busca local
   c) faça $s^* \leftarrow s$, caso a solução *s* melhore *s**.
3) retorne a melhor solução obtida, *s**

Figura 2.6 – Pseudocódigo do algoritmo GRASP

A primeira fase do GRASP é a fase de construção, no qual uma solução viável é construída elemento a elemento. Cada elemento da solução é avaliado e, em seguida, é adicionado ordenadamente de acordo com um critério guloso, em uma lista, denominada de Lista de Candidatos (LC). Através de um fator $\alpha \in [0,1]$ é criado a Lista Restrita de Candidatos (LRC), cujos elementos são os melhores da LC e cujo tamanho é $|LRC| = \alpha \times |LC|$, onde $|LC|$ é a cardinalidade da lista de candidatos. Definido a LRC, seleciona-se, aleatoriamente, um candidato da LRC e, em seguida, atualiza-se ambas as listas LC e LRC. O método pára quando $LC = \emptyset$.

De acordo com Feo e Resende (1995), o parâmetro α, que determina o tamanho da LRC, influencia significativamente na qualidade e diversidade da solução gerada pela fase de construção. Valores de α muito baixos (próximos de zero), ou seja, que determinam um tamanho muito limitado para a LC, geram soluções próximas à solução puramente gulosa e implicam em uma baixa diversidade das soluções geradas. Já uma escolha de α próxima da seleção puramente aleatória leva a uma grande diversidade de soluções construídas mas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O pseudocógido da fase de construção é apresentado na Figura 2.7;

1) Classifique os elementos da solução utilizando um critério guloso e adicione-os a uma lista de candidatos LC
2) Crie uma LRC, composto pelos melhores elementos da LC
3) Selecione aleatoriamente um candidato pertencente a LRC para ser incluído na solução
4) Atualize as listas LC e LRC, considerando a remoção do elemento removido.
5) Repita este procedimento até que a solução seja concluída.

Figura 2.7 – Fase de construção GRASP

A segunda fase do GRASP consiste em refinar a solução gerada pela fase de construção, aplicando um método de busca local. A velocidade de convergência para um ótimo local irá depender da qualidade da solução construída. Quanto melhor for a qualidade da solução gerada pela heurística de construção, maior será a velocidade de convergência desta solução para um ótimo local.

2.2.4 Método de Descida em Vizinhança Variável

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto originalmente em Mladenovic e Hansen (1997), é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada. Apresenta-se, na Figura 2.8, o pseudocódigo desse algoritmo para um problema de minimização.

1) Seja r o número de estruturas de vizinhança distintas
2) $s \leftarrow s_0$, onde s_0 é a solução inicial e s é a solução corrente
3) $k \leftarrow 1$, onde k é a estrutura de vizinhança corrente
4) enquanto $(k \leq r)$ faça
   a) encontre o melhor vizinho $s' \in N^k(s)$, onde k é a k-ésima estrutura de vizinhança
   b) se $f(s') < f(s)$, então $s \leftarrow s'$ e $k \leftarrow 1$
   c) caso contrário, $k \leftarrow k+1$
5) retorne a solução s

Figura 2.8 – Pseudocódigo do algoritmo VND

2.2.5 Método de Pesquisa em Vizinhança Variável

O Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS), proposto por Mladenovic e Hansen (1997) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Contrariamente à outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança. O pseudocódigo do algoritmo é apresentado pela Figura 2.9.

1) Defina um conjunto de estruturas de vizinhança $N^k; k = 1,2,...,k\max$
2) Defina uma solução inicial s_0 e, então, faça $s \leftarrow s_0$
3) enquanto (critério de parada não for satisfeito) faça
   a) $k \leftarrow 1$
   b) enquanto $(k \leq k\max)$ faça
      i) gere, aleatoriamente, um vizinho *s'* a solução *s*, pertencente a k-ésima estrutura de vizinhança ($s' \in N^k(s)$)
      ii) encontre o ótimo local do vizinho *s'*, denotado por *s''*.
      iii) se $f(s'') < f(s)$, então $s \leftarrow s''$ e $k \leftarrow 1$
      iv) caso contrário, $k \leftarrow k+1$
4) retorne a solução *s*.

Figura 2.9 – Pseudocódigo do algoritmo VNS

Esse algoritmo inicia-se de uma solução inicial qualquer e a cada iteração seleciona-se aleatoriamente um vizinho *s'* dentro da vizinhança $N^k(s)$ da solução *s* corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, *s''*, for melhor que a solução s corrente, a busca continua de *s''* recomeçando da primeira estrutura de vizinhança $N^l(s)$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N^{k+1}(s)$. Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de CPU, o número máximo de iterações ou número máximo de iterações consecutivas entre dois melhoramentos.

2.2.6 *Iterated Local Search* (ILS)

O ILS, proposto por (Lourenço *et al*., 2002), é um método de busca local que procura focar a busca não no espaço completo de soluções, mas num pequeno subespaço definido por soluções que são ótimas locais de determinado mecanismo de otimização. O sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha da busca local, das perturbações e do critério de aceitação.

Esse método funciona da seguinte maneira: seja *C* a função custo do problema a ser minimizada. Seja *S* o conjunto de todas as soluções e seja *s* uma solução candidata.

Deseja-se explorar *S**, um conjunto menor de *S* somente com soluções *s**, ótimas localmente, usando um caminho que passa de uma solução *s** para uma próxima solução sem a restrição de usar somente o vizinho mais próximo.

Determinando a solução corrente *s**, primeiramente aplica-se uma mudança ou perturbação que faz as mudanças para um estado intermediário *s'* pertencente a *S*. Após isso, o procedimento de busca local é aplicado em *s'* e encontra-se uma solução *s*'* em *S**. Se *s*'* passa no teste de aceitação, ela torna o próximo elemento da caminhada em *S**, senão, retorna-se a *s**. Esse procedimento pode encontrar regiões promissoras no espaço de soluções. Para isso, a intensidade das perturbações não pode ser nem tão pequenas e nem tão grandes. Se elas forem muito pequenas, *s'* freqüentemente pertence à região de atração de *s** e com isso novas soluções de *S** serão exploradas. Se as perturbações forem muito grandes, *s'* irá pertencer a uma região aleatória de atração, isto é, não existe nenhuma polarização na amostragem, então é preciso reiniciar o algoritmo.

Geralmente a caminhada do ILS não é reversível, isto é, se a caminhada for de s_1 para

s_2, não se consegue vir de s_2 para s_1.

Como perturbações determinísticas podem conduzir a curtos ciclos, pode-se randomizar a perturbação ou ser adaptativa para evitar esse tipo de ciclagem.

O potencial do ILS está nas amostragens do conjunto de soluções ótimas locais. A eficiência dessa amostragem depende tanto dos tipos de perturbações quanto do critério de aceitação.

A Figura 2.10 mostra o pseudocódigo algoritmo ILS básico.

1) Defina uma solução inicial s_0
2) $s^* \leftarrow BuscaLocal(s_0)$
3) enquanto (critério de parada não for satisfeito) faça
   a) $s' \leftarrow Perturbação(histórico, s^*)$
   b) $s^{*\prime} \leftarrow BuscaLocal(s')$
   c) $s^* \leftarrow CritérioAceitação(histórico, s^*, s^{*\prime})$
4) retorne a solução s^*.

Figura 2.10 – Pseudocódigo do algoritmo ILS

2.2.7 *Método de Reconexão por Caminhos (Path Relinking)*

A técnica Reconexão por Caminhos (*Path Relinking*) é uma estratégia de intensificação proposta por Glover (1996) que exploram trajetórias que conectam soluções elite. O termo soluções elite se refere às soluções de boa qualidade geradas anteriormente à aplicação da técnica ou durante sua aplicação.

O processo de busca por melhores soluções consiste em, a partir de uma ou mais soluções elites, gerar e explorar o espaço de soluções e assim chegar a outras soluções elites. Para isso, é necessário aplicar movimentos na solução corrente que possuem características da solução elite, e assim, incorporar características de boa qualidade à solução corrente.

Segundo Rosseti (2003), a técnica de Reconexão por Caminhos pode ser aplicada segundo duas estratégias:

- Como uma estratégia de pós-otimização entre todos os pares de soluções elite;
- Como uma estratégia de intensificação a cada ótimo local encontrado após a fase de busca local.

A técnica de Reconexão por Caminhos aplicada como uma estratégia de intensificação a cada ótimo local é aplicada a pares (s_1, s_2) de soluções, onde s_1 é a solução corrente encontrada após um processo de busca local e s_2 é uma solução elite escolhida de forma aleatória, pertencente a um conjunto limitado de soluções elites. Essas soluções elites foram encontradas a partir da exploração do espaço de soluções. Como o conjunto de soluções elite encontra-se inicialmente vazio, todas as soluções encontradas na fase de busca local são apenas inseridas no conjunto. Quando o conjunto de soluções elite estiver cheio, e a cada busca local, a solução obtida torna-se candidata a fazer parte desse conjunto. Para uma solução entrar nesse conjunto é necessário que ela seja melhor que a pior solução do conjunto. Além disso, é necessário que a solução candidata possua um percentual mínimo de diferença para cada solução pertencente ao conjunto de soluções elite.

A Figura 2.11 ilustra o pseudocódigo da técnica Reconexão por Caminhos unidirecional.

5) $g' \leftarrow s$
6) Atribua a g'' a melhor solução entre s e g
7) Calcule o conjunto de movimentos possíveis $\Delta(s, g)$
8) enquanto $(|\Delta(s, g)| \neq 0)$ faça
   a) Atribua a g''' a melhor solução obtida aplicando o melhor movimento de $\Delta(s, g)$ a g'
   b) Exclua esse movimento de $\Delta(s, g)$
   c) $g' \leftarrow g'''$
   d) se $f(g') < f(g'')$ faça
      i) $g'' \leftarrow g'$
9) retorne a solução g''.

Figura 2.11 – Pseudocódigo do algoritmo Reconexão por Caminhos

O algoritmo inicia-se calculando a diferença simétrica $\Delta(s_1, s_2)$ cujo valor representa o número de movimentos que deve ser aplicado a s_1, dita solução corrente, para que se chegue à solução elite s_2. A cada iteração, o melhor movimento de $\Delta(s_1, s_2)$ ainda não aplicado à solução corrente é efetuado até que se alcance a solução elite. Após a aplicação do movimento, esse movimento é excluído do conjunto e, finalmente, verifica-se se a solução corrente melhora a melhor solução encontrada até então. Caso isso ocorra, a solução corrente torna-se candidata a se tornar uma solução elite. O algoritmo pára quando ocorre a convergência da solução corrente para a solução elite.

De acordo com Rosseti (2003), diversas alternativas têm sido consideradas e combinadas em implementações recentes:

- Não aplicar a Reconexão por Caminhos a cada iteração, mas sim periodicamente, para não onerar o tempo final do algoritmo;
- Explorar duas trajetórias potencialmente diferentes, usando s_1 como solução inicial e depois s_2 no mesmo papel;
- Explorar apenas uma trajetória, usando s_1 ou s_2 como solução inicial e
- Não percorrer a trajetória completa de s_1 até s_2, mas sim apenas parte dela (Reconexão por Caminhos Truncada).

Em Ribeiro *et al.* (2002), observa-se que a exploração de duas trajetórias potencialmente diferentes para cada par (s_1,s_2) de soluções, consome aproximadamente, o dobro de tempo para exploração de apenas uma delas, com um ganho muito pequeno na qualidade da solução. Observa-se também que se a solução inicial do método Reconexão por Caminhos for a melhor dentre s_1 e s_2, melhores soluções são encontradas quando apenas uma das trajetórias é investigada. Como justificativa tem-se que a exploração da vizinhança da solução inicial é mais intensa e que soluções de melhora são encontradas próximas à solução inicial, fazendo com que o algoritmo possa executar por um tempo menor e parar antes de se alcançar a solução guia.

3 O Problema de Programação de Jogos Abordado

3.1 Introdução

Realizar escalonamento de jogos para competições esportivas é uma tarefa que exige bastante tempo e estudo dos profissionais que lidam com este tipo de problema. Isso se deve pelo fato de que cada problema apresenta restrições a serem atendidas e objetivos diferentes a serem cumpridos.

O problema de programação de jogos foi pioneiramente estabelecido por Easton, Nemhauser e Trick, a partir do problema de programação de jogos da liga americana de beisebol. Para este problema, foi criado um conjunto de problemas-teste disponível em http://mat.gsia.cmu.edu/TOURN/, considerando as seguintes restrições:

- ✓ Não deve haver mais que três jogos consecutivos dentro ou fora de casa para um qualquer time.
- ✓ Um jogo *Ti x Tj*, sediado na cidade do time *Ti* e o jogo *Tj x Ti*, sediado na cidade do time *Tj*, não devem ser realizados em rodadas consecutivas.

Posteriormente, a este conjunto foi adicionado um outro conjunto de problemas-teste envolvendo regras típicas de competições latino-americanas (Ribeiro e Urrutia, 2004a).

Os problemas abordados no presente trabalho tratam da programação de jogos de competições esportivas realizadas em dois turnos. A Seção 3.2 trata do caso não espelhado, isto é, quando a ordem de realização dos jogos realizados no segundo turno independe da ordem dos jogos do primeiro turno. Já a Seção 3.3 trata do caso espelhado, isto é, os jogos do segundo turno são os mesmos do primeiro turno com os mandos de campo invertidos.

3.2 Descrição do problema não espelhado

O problema de programação de jogos de competições esportivas realizadas em dois turnos não espelhados (PPJNE) é um problema típico de alocação de jogos da Liga Americana de Beisebol (*Major League Baseball*, MLB). Neste problema, os dois confrontos envolvendo cada par de times podem ser realizados em duas rodadas quaisquer, desde que não sejam consecutivas. Desta forma, esses confrontos podem ser realizados inclusive em um mesmo turno. A outra restrição é que cada time não pode participar de mais que 3 (três) jogos consecutivos em casa ou fora de sua sede.

Seja *Ti x Tj* a representação de um jogo do time *Ti* contra o time *Tj*, sediado na cidade do time *Ti*. Então, uma escala de jogos pode ser representada conforme a tabela a seguir. Considere a Tabela 3.1 do PPJNE com 6 times, no qual os números representam os times envolvidos.

Tabela 3.1 – Representação de uma tabela do PPJNE.

Rodada \ Jogo	1	2	3
1	1 x 3	4 x 2	5 x 6
2	2 x 6	4 x 1	5 x 3
3	4 x 5	3 x 6	2 x 1
4	6 x 1	3 x 4	2 x 5
5	6 x 2	1 x 4	3 x 5
6	1 x 2	6 x 3	5 x 4
7	4 x 3	5 x 2	1 x 6
8	5 x 1	4 x 6	2 x 3
9	2 x 4	6 x 5	3 x 1
10	6 x 4	3 x 2	1 x 5

Nesta tabela observa-se que todos os jogos são efetuados em um único turno de dez rodadas. Observa-se, também, que o par de jogos envolvendo dois times podem ser feitos em duas rodadas quaisquer. Ou seja, o par de jogos pode estar em duas rodadas próximas, como, por exemplo, os jogos *5x1* e *1x5*, nas rodadas 8 e 10 respectivamente, ou pode estar em duas rodadas distantes, como, por exemplo, os jogos *1x3* e *3x1*, nas rodadas 1 e 9 respectivamente.

3.3 Descrição do problema espelhado

O problema de programação de jogos de competições esportivas realizadas em dois turnos espelhados (PPJE) é um problema de alocação de jogos típico de competições latino-americanas como, por exemplo, o Campeonato Brasileiro de Futebol. Neste problema há uma divisão entre o primeiro e o segundo turno, sendo que os jogos do segundo turno são realizados na mesma seqüência dos jogos do primeiro turno. Na literatura, diz-se que os turnos são "espelhados". Ou seja, os jogos do segundo turno são os mesmos do primeiro turno, com o mando de campo invertido. Para exemplificar, considere uma tabela para o PPJE composto por 6 times. A Tabela 3.2 ilustra uma escala do PPJE.

Tabela 3.2 – Representação de uma tabela do PPJE.

Rodada \ Jogo	1	2	3
1	1 x 6	2 x 5	4 x 3
2	5 x 1	3 x 2	4 x 6
3	1 x 4	2 x 6	3 x 5
4	1 x 3	2 x 4	6 x 5
5	2 x 1	3 x 6	5 x 4
6	6 x 1	5 x 2	3 x 4
7	1 x 5	2 x 3	6 x 4
8	4 x 1	6 x 2	5 x 3
9	3 x 1	4 x 2	5 x 6
10	1 x 2	6 x 3	5 x 4

Nesta tabela, os jogos são distribuídos em jogos do primeiro e segundo turno. As cinco primeiras rodadas são pertencentes ao primeiro turno e as cinco seguintes, são referentes ao segundo turno. Exemplificando o problema espelhado, percebe-se que os confrontos *1x6*, *2x5* e 4x3, da primeira rodada do segundo turno (denotado pela rodada 6) são os mesmos da primeira rodada do primeiro turno, porém com o mando de campo invertido: *6x1*, *5x2* e *3x4*. O mesmo ocorre com as rodadas 2 e 7, 3 e 8, 4 e 9 e 5 e 10.

No conjunto de problemas-teste disponível em http://mat.gsia.cmu.edu/TOURN/, há uma única restrição considerada, a saber, o número máximo de jogos consecutivos de cada time, jogando em casa ou fora de casa, é limitado a 3.

Já no conjunto de problemas-teste disponível na página do orientador da pesquisa, a saber, http://www.decom.ufop.br/prof/marcone/projects/ttp/bssp.html, relativos ao Campeonato Brasileiro de Futebol dos anos de 2004, 2005 e 2006, consideram-se as seguintes restrições:

1. Cada time joga somente uma vez por rodada;
2. Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos;
3. Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos, sendo um em casa e o outro na casa do adversário;
4. As duas últimas rodadas de cada turno terão a configuração inversa das duas primeiras rodadas de cada turno com relação ao mando de campo;
5. Não poderá haver jogos entre times do mesmo estado na última rodada;
6. A diferença entre os jogos feitos em cada turno em casa e fora de casa de um time não pode ser maior que uma unidade;
7. Um time não pode jogar mais que duas vezes consecutivas dentro ou fora de casa.

4 Metodologia

Neste capítulo apresenta-se a metodologia proposta para resolver o PPJE. As Seções 4.1 e 4.2 descrevem como uma solução do PPJ é representada. Na Seção 4.3 são apresentados os movimentos que constituem as estruturas de vizinhança utilizadas para resolução do problema. Na Seção 4.4 mostra-se como uma solução é avaliada. Três metodologias para geração da solução inicial são apresentadas na Seção 4.5. As metaheurísticas utilizadas para resolver o PPJ são apresentadas nas Seções 4.6 (*Simulated Annealing*), 4.7 (VND), 4.8 (VNS/VND), 4.9 (Busca Tabu), 4.10 (ILS e MRD) e 4.11 (*Path Relinking*).

4.1 Representação de Biajoli *et al*. (2004)

Na abordagem de Biajoli *et al*. (2004), uma tabela, ou solução $s = (s_{ij})$ do problema, é representada por uma matriz com o número de linhas de acordo com o número de rodadas da competição e com o número de colunas de acordo com o número de confrontos que se tem por rodada.

Para explicar a tabela, suponha um campeonato realizado em dois turnos com a participação de 6 (n) times que se confrontam em 10 ($nr = 2n - 2$) rodadas. Nesse campeonato todos os times jogam em todas as rodadas. Então uma solução s para este problema é uma matriz com 10 linhas e 3 colunas ($nr \times n/2$), sendo as 5 ($nr/2$) primeiras linhas referentes aos jogos do primeiro turno e as 5 ($1- nr/2$) linhas restantes referentes aos jogos do segundo turno. Nesta matriz cada célula s_{ij} representa um jogo do tipo $T_i \times T_j$. Um jogo do tipo T_i x T_j (onde T_i tem mando de campo) ou T_j x T_i (onde T_j tem mando de campo) é representado unicamente por "(sinal) $T_i \times T_j$", sendo o mando de campo definido pelo sinal. Quando o sinal é positivo, o mando de campo é do time i e, quando é negativo, o mando de campo é do time j. Ou seja, o jogo $T_i \times T_j$ é representado por $(+T_i \times T_j)$ e o jogo $T_j \times T_i$, por $(-T_i \times T_j)$. Nessa representação, os times são representados pelas letras do alfabeto. A Tabela 4.1 ilustra um fragmento de uma escala.

Tabela 4.1 - Representação de uma escala do PPJE

Rodada \ Jogo	1	2	3
1	+ A x F	+ B x E	- D x C
2	- E x A	+ C x B	+ D x F
3	+ A x D	+ B x F	+ C x E
4	+ A x C	- B x D	- F x E
5	- B x A	+ C x F	- E x D
6	- A x F	- B x E	+ D x C
7	+ E x A	- C x B	- D x F
8	- A x D	- B x F	- C x E
9	- A x C	+ B x D	+ F x E
10	+ B x A	- C x F	+ E x D

Nesta tabela, o jogo – *E x A*, localizado no primeiro jogo da segunda rodada, mostra que o time *E* joga contra o time *A* na casa do time *A*. Já no primeiro jogo da nona rodada, o jogo + *E x A* mostra que o time *E* joga em sua sede contra o time *A*.

4.2 Representação de Anagnostopoulos *et al*. (2003)

A principal representação de uma solução adotada neste projeto de pesquisa é a utilizada por Anagnostopoulos *et al*. (2003).

Seja n o número de times envolvidos e $nr = 2n - 2$ o número de rodadas do campeonato. A solução nesta representação é uma matriz de n linhas por nr colunas. Cada linha referencia a um time T_i, cada coluna representa uma rodada R_k e cada célula representa o oponente do time T_i na rodada R_k. O sinal associado ao oponente denota o mando de campo, ou seja, caso o sinal seja positivo, significa que o time T_i irá jogar contra um oponente em sua própria sede. Caso o sinal seja negativo, significa que o time T_i irá jogar contra seu oponente na sede do time que está confrontando. A Tabela 4.2 exemplifica a solução da Tabela 4.1 na representação de Anagnostopoulos *et al*. (2003), onde os times 1, 2, 3, 4, 5 e 6 representam, respectivamente, os times A, B, C, D, E e F..

Tabela 4.2 – Solução na representação de Anagnostopoulos *et al*. (2003)

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	+ 4	+ 3	- 2	- 6	+ 5	- 4	- 3	+ 2
2	+ 5	- 3	+ 6	+ 4	+ 1	- 5	+ 3	- 6	- 4	- 1
3	- 4	+ 2	+ 5	- 1	+ 6	+ 4	- 2	- 5	+ 1	- 6
4	+ 3	+ 6	- 1	- 2	- 5	- 3	- 6	+ 1	+ 2	+ 5
5	- 2	+ 1	- 3	- 6	+ 4	+ 2	- 1	+ 3	+ 6	- 4
6	- 1	- 4	- 2	+ 5	- 3	+ 1	+ 4	+ 2	- 5	+ 3

Na Tabela 4.2, observa-se que a notação +5, encontrada na primeira rodada do time 2, indica que o time 2 joga em sua sede contra o time 5, enquanto que na rodada 4 do time 3, encontra-se a notação -1, cujo significado é que o time 1 joga em sua sede contra o time 4.

Nesta representação, pode-se verificar a trajetória percorrida por um time durante o campeonato. Como exemplo, a trajetória do time 6, na solução da Tabela 4.2, será:

O time 6 inicia o campeonato dentro de sua sede e, no primeiro turno, desloca-se para a sede do time 1. Em seguida, desloca-se para a sede do time 4 (na segunda rodada) sem retornar à sua própria sede. Na rodada 3, ele segue da sede do time 4 para a sede do time 2. Na quarta rodada, ele retorna a sua sede (pois o jogo será *6x5*, na sede do time 6) e desloca-se para a sede do time 3, na quinta rodada e assim sucessivamente até a 10ª rodada, onde encerrará o campeonato em sua sede. Isso não ocorre com o time 2, pois seu último confronto será na sede do time 5. Neste caso, é necessário que o time 2 retorne a sua sede após a décima rodada, para que assim possa encerrar o campeonato. As trajetórias dos times 2 e 6 são descritas a seguir:

Trajetória do time 6: $6 \xrightarrow{1} 1 \xrightarrow{2} 4 \xrightarrow{3} 2 \xrightarrow{4} 6 \xrightarrow{5} 3 \xrightarrow{6} 6 \xrightarrow{7} 5 \xrightarrow{8} 6$

Trajetória do time 2: $2 \xrightarrow{2} 3 \xrightarrow{3} 2 \xrightarrow{6} 6 \xrightarrow{7} 4 \xrightarrow{8} 1 \xrightarrow{9} 2 \xrightarrow{10} 5 \mapsto 2$

onde a notação $T_i \xrightarrow{r} T_j$ significa que o time T_i desloca-se para a sede do time T_j, nada rodada r; e a notação $T_i \mapsto T_j$ significa que o time T_j retorna da sede do time T_i à sua sede, no final do campeonato.

4.3 Estruturas de vizinhança

Aplicam-se, neste trabalho, seis estruturas de vizinhança, dentre elas, cinco adotadas por Anagnostopoulos *et al*. (2003) e uma proposta por Biajoli *et al*. (2004). A primeira estrutura de vizinhança N^1 gera soluções vizinhas através do movimento *swap rounds*. A segunda estrutura de vizinhança N^2 é composta pelo movimento *swap teams*, a terceira estrutura de vizinhança N^3 é constituída pelo movimento *swap homes*. As estruturas de vizinhança N^4 e N^5 são constituídas respectivamente pelos movimentos *partial swap rounds* e *partial swap teams*. As três primeiras estruturas de vizinhança têm complexidade $O(n^2)$ enquanto que as outras duas têm complexidade $O(n^3)$. E a última estrutura de vizinhança N^6 é definida pelo movimento *replace teams*.

Considere a solução *s* apresentada na Tabela 4.3.

Tabela 4.3 – Solução *s*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	+ 4	+ 3	- 2	- 4	- 3	+ 2	+ 5	- 6
2	+ 5	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 3	- 5
3	- 4	+ 2	+ 5	- 1	+ 6	- 5	+ 1	- 6	- 2	+ 4
4	+ 3	+ 6	- 1	- 2	- 5	+ 1	+ 2	+ 5	- 6	- 3
5	- 2	+ 1	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 1	+ 2
6	- 1	- 4	- 2	+ 5	- 3	+ 2	- 5	+ 3	+ 4	+ 1

O movimento *swap rounds* consiste em trocar todos os jogos de uma rodada *Ri* e *Rj*. Efetuando-se este movimento na solução *s*, para as rodadas 3 e 7, tem-se como resultado a solução a seguir:

Tabela 4.4 – Movimento *swap rounds*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	- 3	+ 3	- 2	- 4	+ 4	+ 2	+ 5	- 6
2	+ 5	- 3	- 4	+ 4	+ 1	- 6	+ 6	- 1	+ 3	- 5
3	- 4	+ 2	+ 1	- 1	+ 6	- 5	+ 5	- 6	- 2	+ 4
4	+ 3	+ 6	+ 2	- 2	- 5	+ 1	- 1	+ 5	- 6	- 3
5	- 2	+ 1	+ 6	- 6	+ 4	+ 3	- 3	- 4	- 1	+ 2
6	- 1	- 4	- 5	+ 5	- 3	+ 2	- 2	+ 3	+ 4	+ 1

O movimento *swap teams* consiste em trocar todos os jogos dos times *Ti* e *Tj*, de todas as rodadas, com exceção daquelas em que eles se confrontam. Este movimento é realizado em duas partes: A primeira consiste em trocar os jogos dos times *Ti* e *Tj*, conforme exemplificada na Tabela 4.5, considerando a solução *s*, apresentado na tabela 4.3 e os times 2 e 5.

Tabela 4.5 – Movimento *swap teams* – parte 1

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	+ 4	+ 3	- 2	- 4	- 3	+ 2	+ 5	- 6
2	+ 5	+ 1	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 1	- 5
3	- 4	+ 2	+ 5	- 1	+ 6	- 5	+ 1	- 6	- 2	+ 4
4	+ 3	+ 6	- 1	- 2	- 5	+ 1	+ 2	+ 5	- 6	- 3
5	- 2	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 3	+ 2
6	- 1	- 4	- 2	+ 5	- 3	+ 2	- 5	+ 3	+ 4	+ 1

Como a simples troca dos jogos gera inconsistência na escala, a segunda parte do movimento efetua, de forma determinística, a atualização da solução:

Tabela 4.6 – Movimento *swap teams* – parte 2

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 2	+ 4	+ 3	- 5	- 4	- 3	+ 5	+ 2	- 6
2	+ 5	+ 1	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 1	- 5
3	- 4	+ 5	+ 2	- 1	+ 6	- 2	+ 1	- 6	- 5	+ 4
4	+ 3	+ 6	- 1	- 5	- 2	+ 1	+ 5	+ 2	- 6	- 3
5	- 2	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 3	+ 2
6	- 1	- 4	- 5	+ 2	- 3	+ 5	- 2	+ 3	+ 4	+ 1

A terceira estrutura de vizinhança gera soluções vizinhas a partir do movimento *swap homes*, o qual aplicado à solução *s* da tabela 4.3 e aos times 1 e 4, gera a solução a seguir:

Tabela 4.7 – Movimento *swap homes*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	- 4	+ 3	- 2	+ 4	- 3	+ 2	+ 5	- 6
2	+ 5	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 3	- 5
3	- 4	+ 2	+ 5	- 1	+ 6	- 5	+ 1	- 6	- 2	+ 4
4	+ 3	+ 6	+ 1	- 2	- 5	- 1	+ 2	+ 5	- 6	- 3
5	- 2	+ 1	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 1	+ 2
6	- 1	- 4	- 2	+ 5	- 3	+ 2	- 5	+ 3	+ 4	+ 1

O movimento *partial swap rounds* consiste em trocar um jogo das rodadas *Ri* e *Rj* de um time *Tk*. A Tabela 4.9 mostra o resultado da aplicação do movimento *partial swap rounds* para as rodadas 2 e 9 e para o time 2, na solução *s'* apresentada na Tabela 4.8. Observa-se que é necessário, após a troca dos jogos das rodadas 2 e 9 do time 2, fazer a correção da escala para que a representação fique consistente. Isto pode ser feito trocando-se os jogos dos times 1, 4 e 6 nas rodadas 2 e 9.

Tabela 4.8 – Solução *s*'

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 2	+ 2	+ 3	- 5	- 4	- 3	+ 5	+ 4	- 6
2	+ 5	+ 1	- 1	- 5	+ 4	+ 3	+ 6	- 4	- 6	- 3
3	- 4	+ 5	+ 4	- 1	+ 6	- 2	+ 1	- 6	- 5	+ 2
4	+ 3	+ 6	- 3	- 6	- 2	+ 1	+ 5	+ 2	- 1	- 5
5	- 2	- 3	+ 6	+ 2	+ 1	- 6	- 4	- 1	+ 3	+ 4
6	- 1	- 4	- 5	+ 4	- 3	+ 5	- 2	+ 3	+ 2	+ 1

Tabela 4.9 – Movimento *partial swap rounds*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	+ 4	+ 2	+ 3	- 5	- 4	- 3	+ 5	- 2	- 6
2	+ 5	- 6	- 1	- 5	+ 4	+ 3	+ 6	- 4	+ 1	- 3
3	- 4	+ 5	+ 4	- 1	+ 6	- 2	+ 1	- 6	- 5	+ 2
4	+ 3	- 1	- 3	- 6	- 2	+ 1	+ 5	+ 2	+ 6	- 5
5	- 2	- 3	+ 6	+ 2	+ 1	- 6	- 4	- 1	+ 3	+ 4
6	- 1	+ 2	- 5	+ 4	- 3	+ 5	- 2	+ 3	- 4	+ 1

O movimento *partial swap teams* consiste em trocar os jogos dos times *Ti* e *Tj* em uma rodada *Rk*. A Tabela 4.10 mostra o resultado da aplicação do movimento *partial swap teams* para os times 3 e 5 na rodada 8. Após a troca dos jogos dos times 3 e 5 na 8ª rodada, a escala é, então, atualizada de forma determinística para corrigir a inconsistência gerada na representação.

Tabela 4.10 – Movimento *partial swap teams*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	+ 4	+ 5	- 2	- 4	- 3	+ 2	+ 3	- 6
2	+ 3	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 5	- 5
3	- 2	+ 2	+ 5	- 6	+ 6	- 5	+ 1	- 4	- 1	+ 4
4	+ 5	+ 6	- 1	- 2	- 5	+ 1	+ 2	+ 3	- 6	- 3
5	- 4	+ 1	- 3	- 1	+ 4	+ 3	+ 6	- 6	- 2	+ 2
6	- 1	- 4	- 2	+ 3	- 3	+ 2	- 5	+ 5	+ 4	+ 1

Os movimentos *partial swap rounds* e *partial swap teams*, pertencentes às estruturas de vizinhança N^4 e N^5, podem transformar uma escala com rodadas espelhadas em uma escala com rodadas não-espelhadas e vice-versa. Assim, esses dois movimentos são importantes para a resolução do problema de programação de jogos com turnos não espelhados, pois ampliam consideravemente o espaço de soluções, já que o método de geração da solução inicial descrito na seção 4.5 gera somente escalas espelhadas ao final do processo.

O movimento replace teams consiste em substituir um determinado time Ti por outro time T_j e vice-versa. A Tabela 4.11 mostra o resultado da aplicação desse movimento à solução *s* da Tabela 4.3, considerando os dois confrontos envolvendo os times 1 e 2.

Tabela 4.11 – Movimento *replace teams*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
2	+ 6	- 5	+ 4	+ 3	- 1	- 4	- 3	+ 1	+ 5	- 6
1	+ 5	- 3	+ 6	+ 4	+ 2	- 6	- 4	- 2	+ 3	- 5
3	- 4	+ 1	+ 5	- 2	+ 6	- 5	+ 2	- 6	- 1	+ 4
4	+ 3	+ 6	- 2	- 1	- 5	+ 2	+ 1	+ 5	- 6	- 3
5	- 1	+ 2	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 2	+ 1
6	- 2	- 4	- 1	+ 5	- 3	+ 1	- 5	+ 3	+ 4	+ 2

4.4 Função de avaliação

Uma solução *s* é avaliada pela função *f* apresentada pela fórmula (4) considerando o problema de alocação de jogos da Liga Americana de Beisebol.

$$f(s) = \sum_{i \in T} custo(i) + \sum_{j \in C} w_j \times inv_j \tag{4}$$

em que
- $f(s)$: função de avaliação
- T: conjunto dos times participantes da competição;
- C: conjunto de restrições descritas na Seção 3.1;
- $custo(i)$: custo de um time $i \in T$, conforme definido mais adiante;
- w_j: penalidade por desrespeitar a restrição $j \in C$;
- inv_j: número de vezes que a restrição $j \in C$ está sendo desrespeitada.

O custo de um time $i \in T$ é definido da seguinte forma. Seja a solução *s* apresentada na Tabela 4.3 e d_{ij} a distância entre as sedes dos times *i* e *j* e seja $custo(i)$ o custo de deslocamento efetuado pelo time *i* em toda a competição, considerando que no seu início cada time sai da sua sede e à ela retorna ao final da competição. Assim, por exemplo, o custo de deslocamento do time 1 na solução da Tabela 4.3, é calculado pela fórmula (5).

$$custo(1) = d_{15} + d_{51} + d_{12} + d_{24} + d_{43} + d_{31} + d_{16} + d_{61} \tag{5}$$

Para avaliar uma solução do PPJE para o Campeonato Brasileiro de Futebol, utiliza-se a função *f* definida pela fórmula (4), tal que o conjunto *C* é composto das restrições descritas na Seção 3.3. A essa fórmula é acrescida ainda mais um componente, $dif(s)$, que representa a diferença entre o custo máximo e o custo mínimo dos times, isto é $dif(s) = \max\{custo(i),\ i \in T\} - \min\{custo(i),\ i \in T\}$

4.5 Geração de uma solução inicial

4.5.1 Método de três fases para geração de uma solução inicial

Para a geração da solução inicial, utiliza-se o método de três fases proposto por Ribeiro e Urrutia (2004a), conforme descrito a seguir.

Na primeira fase, utiliza-se o Método do Polígono. Considera-se, inicialmente, um polígono regular com *n*-1 nós numerados em sentido horário, onde *n* é o número de times. A cada um dos *n*-1 nós é associado um time "abstrato" (no caso, um literal). Para toda rodada, o time localizado no *k*-ésimo nó, $k = 2,3,...,n/2$, do polígono joga contra o time situado no nó $n+1-k$. Considera-se, também, um time abstrato, no caso o *n*-ésimo, fora desse polígono. Esse *n*-ésimo time abstrato joga contra o time associado ao primeiro nó do polígono. Após a determinação dos jogos de cada rodada, cada time localizado no nó 1,2,...,*n*-1 é movido em sentido horário para o próximo nó do polígono.

A Figura 4.1 mostra a formação dos jogos de cada rodada através do método do polígono para um campeonato envolvendo 6 times.

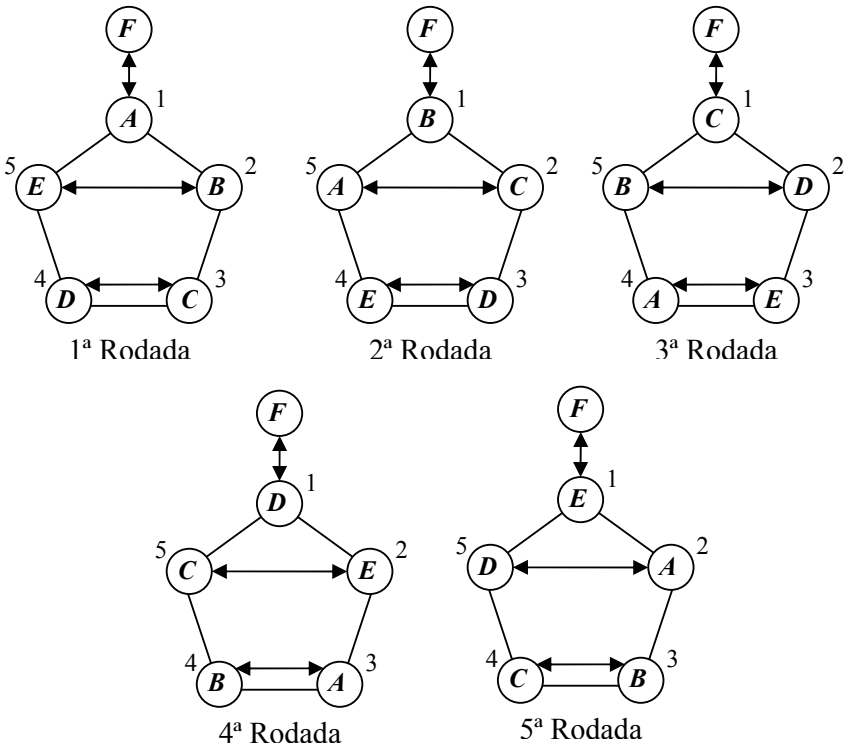

Figura 4.1 – Método do Polígono

Nessa figura, as letras situadas internamente aos nós do polígono representam os times abstratos e os números externos aos nós do polígono representam seus respectivos índices. O círculo situado fora (no caso, acima) do polígono refere-se ao *n*-ésimo time abstrato (anteriormente não associado a nenhum nó do polígono) e as setas representam os confrontos a serem realizados.

Em seguida, a escala gerada pelo Método do Polígono é duplicada. A Tabela 4.12 mostra uma escala gerada pelo Método do Polígono.

Tabela 4.12 – Solução inicial obtida pelo Método do Polígono

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	F	D	B	E	C	F	D	B	E	C
2	E	C	A	D	F	E	C	A	D	F
3	D	B	E	F	A	D	B	E	F	A
4	C	A	F	B	E	C	A	F	B	E
5	B	F	C	A	D	B	F	C	A	D
6	A	E	D	C	B	A	E	D	C	B

Essa escala é utilizada para gerar uma matriz quadrada $n \times n$ de oponentes consecutivos. Um time T_k é dito oponente consecutivo dos times T_i e T_j se houver os confrontos $T_i \times T_k$ e $T_j \times T_k$ em rodadas consecutivas, não importando a ordem de realização dos jogos nem o mando de campo. Cada célula (i,j) dessa matriz contém o número de vezes em que os times T_i e T_j são oponentes consecutivos dos outros times. Assim, observa-se que

os times A e C são oponentes consecutivos 6 vezes de outros times, conforme destacado na Tabela 4.13.

Tabela 4.13 – Solução inicial obtida pelo Método do Polígono

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	F	D	B	E	C	F	D	B	E	C
2	E	C	A	D	F	E	C	A	D	F
3	D	B	E	F	A	D	B	E	F	A
4	C	A	F	B	E	C	A	F	B	E
5	B	F	C	A	D	B	F	C	A	D
6	A	E	D	C	B	A	E	D	C	B

A Figura 4.2 mostra a matriz de oponentes consecutivos para a escala da Tabela 4.13.

	A	B	C	D	E	F
A	0	1	6	5	2	4
B	1	0	2	5	6	4
C	6	2	0	2	3	3
D	5	5	2	0	2	4
E	2	6	3	2	0	3
F	4	4	3	4	3	0

Figura 4.2 – Matriz de oponentes consecutivos para $n = 6$

Na segunda fase, os times reais, cuja distância entre suas sedes são pequenas, são preferencialmente associados aos times abstratos que possuem maior número de oponentes consecutivos. Essa idéia de alocação baseia-se no fato de que se os confrontos de um dado time T_k são realizados em rodadas consecutivas contra outros dois times T_i e T_j, então seria conveniente que as sedes desses dois times fossem próximas, de forma a minimizar o deslocamento do time T_k. Assim, quanto maior for a freqüência de confrontos consecutivos dos times T_i e T_j contra o time T_k, menor será a distância total a ser percorrida pelo time T_k. Descreve-se, a seguir, esta heurística em detalhes.

Primeiro, os pares de times abstratos são ordenados em ordem decrescente de acordo com o número de oponentes consecutivos que possuem. Depois, os times reais são ordenados em ordem crescente de distância em relação aos seus oponentes e são associados aos times abstratos nesta ordem. Dessa forma, os times abstratos que possuem maior número de oponentes consecutivos serão preferencialmente associados aos times reais, cujas distâncias são menores.

A Figura 4.3 mostra a matriz de distâncias para $n = 6$ times e a Tabela 4.14 mostra os pares ordenados de times reais com suas respectivas distâncias e pares ordenados de times abstratos com o número de vezes em que estes são oponentes consecutivos de outros times. Observa-se nestas figuras que os números 1,2,3,4,5,6 representam os times reais e as letras A,B,C,D,E,F os times abstratos.

	1	2	3	4	5	6
1	0	745	665	929	605	521
2	745	0	80	337	1090	315
3	665	80	0	380	1020	257
4	929	337	380	0	1380	408
5	605	1090	1020	1380	0	1010
6	521	315	257	408	1010	0

Figura 4.3 – Matriz de distâncias para $n = 6$

Tabela 4.14 – Tabelas de pares ordenados de times abstratos e reais

Times Reais		
1º	(2,3)	80
2º	(3,6)	257
3º	(2,6)	315
4º	(2,4)	337
5º	(3,4)	380
6º	(4,6)	408
7º	(1,6)	521
8º	(1,5)	605
9º	(1,3)	665
10º	(1,2)	745
11º	(1,4)	929
12º	(5,6)	1010
13º	(3,5)	1020
14º	(2,5)	1090
15º	(4,5)	1380

Times Abstratos					
1º	(A,C)	6	16º	(E,C)	3
2º	(C,A)	6	17º	(C,F)	3
3º	(B,E)	6	18º	(F,C)	3
4º	(E,B)	6	19º	(E,F)	3
5º	(A,D)	5	20º	(F,E)	3
6º	(D,A)	5	21º	(A,E)	2
7º	(B,D)	5	22º	(E,A)	2
8º	(D,B)	5	23º	(B,C)	2
9º	(A,F)	4	24º	(C,B)	2
10º	(F,A)	4	25º	(C,D)	2
11º	(B,F)	4	26º	(D,C)	2
12º	(F,B)	4	27º	(D,E)	2
13º	(D,F)	4	28º	(E,D)	2
14º	(F,D)	4	29º	(A,B)	1
15º	(C,E)	3	30º	(B,A)	1

A associação dos times reais aos seus respectivos times abstratos é feito através da heurística cujo pseudocódigo é mostrado na Figura 4.4.

.

1) Repita os seguintes passos até que todos os times reais sejam associados aos times abstratos:
   a) Seja T_1 o próximo time real a ser associado a um time abstrato e seja T_2 o time, cuja cidade é mais próxima à cidade de T_1.
   b) Se *jáFoiAssociadoAUmTimeAbstrato*(T_2) então
      i) Encontre o primeiro par (a,b) de times abstratos, tal que a esteja associado ao time T_2 e b ainda não esteja associado a nenhum time real.
      ii) *associarTimeAbstratoAoTimeReal*(b,T_1)
   c) Senão
      i) Encontre o primeiro par (a,b) de times abstratos, tal que nenhum deles esteja associado a um time real
      ii) *associarTimeAbstratoAoTimeReal*(a,T_1)

Figura 4.4 – Algoritmo para associação dos times reais aos times abstratos

A seguir, descreve-se a aplicação do algoritmo da Figura 4.4 para o problema com 6 times, considerando os dados mostrados nos passos anteriores.

De acordo com a tabela de pares ordenados para os times reais (mostrado na Tabela 4.14), o primeiro time a ser associado a um time abstrato poderia ser o time 2 ou o time 3. É convencionado que o time do par (T_1, T_2) a ser associado primeiramente é o time T_1. Assim, no exemplo considerado, o time real 2 é o primeiro a ser associado a um time abstrato e, de acordo com a matriz de distâncias da Tabela 4.15, o time 3 é o time mais próximo do time 2. Logo, $T_1 = 2$ e $T_2 = 3$.

O próximo passo é encontrar o primeiro par (a,b) de times abstratos tal que nem a nem b estejam associados a um time real. Nesse caso, obtém-se o par abstrato (*A,C*) e, dessa forma, associa-se o time 2 ao time abstrato A.

Continuando a aplicação do algoritmo, obtém-se as seguintes associações:

Tabela 4.15 – Tabela de Associações

T_1	T_2	par (a,b)	Associação
2	3	(A,C)	A=2
3	2	(A,C)	C=3
6	3	(C,E)	E=6
4	2	(A,D)	D=4
1	6	(E,B)	B=1
5	1	(B,F)	F=5

Logo, a escala gerada associando-se os times abstratos aos times reais é mostrado na Tabela 4.16.

Tabela 4.16 – Solução inicial obtida pelo Método do Polígono

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	6	3	2	4	5	6	3	2	4	5
2	5	4	1	6	3	5	4	1	6	3
3	4	1	6	5	2	4	1	6	5	2
4	3	2	5	1	6	3	2	5	1	6
5	2	6	4	3	1	2	6	4	3	1
6	1	5	3	2	4	1	5	3	2	4

Na terceira fase, é estabelecido o local de realização de cada jogo, isto é, a definição do mando de campo. O objetivo dessa fase é gerar uma escala viável. Para tanto, procura-se alternar o máximo possível o mando de campo. Na primeira rodada, os mandos de campo são associados aleatoriamente. Nas rodadas seguintes até a penúltima rodada (n-2) do primeiro turno é utilizada uma estratégia para determinar o mando de campo de um jogo entre os times T_1 e T_2. Seja N_i o número de jogos que o time T_i jogou consecutivamente, tanto dentro ou fora de casa, nas rodadas anteriores. O pseudocódigo do algoritmo de determinação do mando de campo é descrito na Figura 4.5.

1) Determine, de forma aleatória, o mando de campo dos jogos da primeira rodada.
2) Para cada rodada k = 2,3,...,(n-2) do primeiro turno faça
   a) Se $N_2 > N_1$
      i) Se T_2 jogou seu último jogo em casa, então a realização do jogo será na casa do time T_1.
      ii) Se T_2 jogou seu último jogo fora de casa, então a realização do jogo será na casa do time T_2.
   b) Se $N_2 < N_1$
      i) Se T_1 jogou seu último jogo em casa, então a realização do jogo será na casa do time T_2.
      ii) Se T_1 jogou seu último jogo fora de casa, então a realização do jogo será na casa do time T_1.
   c) Se $N_2 = N_1$
      i) Se T_1 jogou seu último jogo em casa e o time T_2 jogou seu último jogo fora de casa, então a realização do jogo será na casa de T_2.
      ii) Se T_1 jogou seu último jogo fora de casa e o time T_2 jogou seu último jogo em casa, então a realização do jogo será na casa de T_1.
      iii) Caso contrário, o mando de campo é determinado aleatoriamente.
3) Determine, de forma aleatória, o mando de campo da última rodada n-1 do primeiro turno.
4) Determinar os jogos do segundo turno espelhados ao primeiro turno.
5) Se a alocação de um estádio no passo (3) tornar a escala inviável, então o mando de campo para o jogo correspondente é invertido.
6) Caso a escala ainda permanecer inviável, reinicie o algoritmo.
7) Retorne a escala gerada como a solução inicial

Figura 4.5 – Algoritmo para determinação do mando de campo dos jogos

Inicialmente, determina-se, aleatoriamente, o mando de campo da primeira rodada. A Tabela 4.17 mostra a aplicação da terceira fase desse método para as rodadas 2, 3 e 4. Nesta tabela, s_1 e s_2 correspondem ao local que os times T_1 e T_2 realizaram seus jogos na rodada

anterior (+ significa que o jogo foi realizado em casa e – significa que o jogo foi realizado fora de casa) e "Local do Jogo" corresponde ao local de realização do jogo $T_1 \times T_2$ na rodada corrente. A notação * significa que o mando de campo foi determinado de forma aleatória.

Tabela 4.17 – Solução inicial obtida pelo Método do Polígono

Rodada	T1	T2	N1	N2	s1	s2	Local do Jogo
2	1	3	0	0	+	+	1*
	2	4	0	0	+	-	4
	5	6	0	0	-	-	5*
3	1	2	1	0	+	-	2
	3	6	0	0	-	+	3
	4	5	0	1	+	-	5
4	1	4	0	0	-	-	1*
	2	6	0	2	+	-	6
	3	5	0	1	+	+	3

Esta tabela mostra, por exemplo, que na rodada 2, o time 1 jogará contra o time 3. Ela mostra, também, que tanto o time 1 quanto o time 3 não jogaram consecutivamente em casa ou fora (pois $N_1 = N_2 = 0$). Além disso, mostra-se que na rodada anterior tanto o time 1 quanto o time 2 jogaram em casa (pois $s_1 = s_2 = +$). Desta forma, o mando de campo é determinado de forma aleatória, já que é indiferente a alocação com relação à prioridade de mando de campo de cada time. No caso apontado, o local do jogo foi escolhido aleatoriamente como sendo o do time 1 (1*).

Uma outra situação está ilustrada na quarta linha desta tabela. Mostra-se que na rodada 3 o time 1 jogará contra o time 2 fora de casa, isto é, o local do jogo deve ser a casa do time 2. Isto se justifica devido ao fato de o time 1 já ter um jogo consecutivo ($N_1 = 1$) em casa ($s_1 = +$), enquanto o time 2 não tem nenhum jogo consecutivo ($N_2 = 0$), além de ter jogado fora de casa na rodada anterior ($s_2 = -$).

Na última rodada do primeiro turno, o mando de campo é determinado de forma aleatória e o mando de campo dos jogos do segundo turno é o inverso do mando de campo dos jogos do primeiro turno (ou seja, o segundo turno é "espelho" do primeiro turno).

Após a aplicação da terceira fase ao exemplo em consideração, obtém-se a seguinte escala, mostrado pela Tabela 4.18, como a solução inicial.

Tabela 4.18 – Solução inicial obtida pelo Método do Polígono

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 6	+ 3	- 2	+ 4	- 5	- 6	- 3	+ 2	- 4	+ 5
2	+ 5	- 4	+ 1	- 6	+ 3	- 5	+ 4	- 1	+ 6	- 3
3	+ 4	- 1	+ 6	+ 5	- 2	- 4	+ 1	- 6	- 5	+ 2
4	- 3	+ 2	- 5	- 1	+ 6	+ 3	- 2	+ 5	+ 1	- 6
5	- 2	+ 6	+ 4	- 3	+ 1	+ 2	- 6	- 4	+ 3	- 1
6	- 1	- 5	- 3	+ 2	- 4	+ 1	+ 5	+ 3	- 2	+ 4

O Problema de Programação de Jogos de Competições Esportivas abordado neste trabalho utiliza as instâncias do Campeonato Brasileiro de 2004, 2005 e 2006, realizado, respectivamente, com 24, 22 e 20 times em dois turnos espelhados. Assim, cada fase de geração da solução inicial é realizada primeiramente apenas para o primeiro turno e depois de concluída essa fase gera-se o segundo turno espelhado do primeiro.

Nesse problema, devem ser levadas em consideração as seguintes restrições:

1. Um time T_i não pode jogar mais de uma vez em uma mesma rodada;
2. Um time T_i não pode jogar mais que duas vezes consecutivas dentro ou fora de casa;
3. Não pode haver jogos envolvendo times de mesmo estado da federação na última rodada de cada turno;
4. Jogos da primeira e segunda rodadas devem ter mandos de campo invertido para cada time;
5. Jogos da penúltima e última rodadas devem também ter mandos de campo invertido para cada time;
6. Para cada time o mando de campo da última rodada deve invertido em relação ao mando de campo da primeira rodada.

O método de três fases utilizado para gerar uma solução inicial contempla duas das seis restrições, a saber:

A primeira fase desse método utiliza o Método do Polígono para determinar os jogos de cada rodada, o qual impede que um time jogue mais de uma vez por rodada, o que faz com que a primeira restrição seja satisfeita.

A terceira fase do método de geração da solução inicial é a fase de determinação do mando de campo. Essa fase tem como objetivo promover um certo equilíbrio entre o número de jogos realizados dentro de casa quanto fora de casa por determinado time T_i ao longo de todo campeonato, mais especificamente, o método não deixa mais que duas vezes consecutivas. Desta forma, a segunda restrição é satisfeita.

Entretanto, as restrições 3 e 4 descritas na Seção 3.3, não conseguiram ser solucionadas pelo método de três fases e nem as metaheurísticas conseguiram contemplá-las. Isso se deve ao fato de o método proposto restringir muito o espaço de soluções ao longo de suas fases, dificultando a aceitação dos movimentos realizados.

Para conseguir solucionar esse problema, foi proposto pelos autores do presente projeto um outro método de geração de solução inicial. Esse método baseado em *backtracking* possui duas fases, sendo estas detalhadas na seção 4.5.2.

4.5.2 Método de duas fases para geração de uma solução inicial

Uma solução inicial é gerada por um procedimento de duas fases, ambas baseadas em *backtracking*. Na primeira, é feita a alocação das duas primeiras e duas últimas rodadas do primeiro turno. Na segunda fase são geradas as rodadas intermediárias desse turno. A seguir, gera-se o segundo turno mantendo-se a mesma seqüência de jogos do primeiro turno, porém com o mando de campo invertido.

4.5.2.1 Primeira fase

Nessa fase, utiliza-se um algoritmo baseado em *backtracking* para gerar quatro rodadas, cujo objetivo é satisfazer às restrições (c) e (d) descritas no Capítulo 2. Essas rodadas formarão as duas primeiras e as duas últimas rodadas do primeiro turno.

Inicialmente, gera-se uma matriz denominada "matriz base", cuja composição é, inicialmente, de uma rodada com seus respectivos mandos de campo definidos aleatoriamente. Em seguida, gera-se uma matriz de domínios, constituída dos oponentes

possíveis para um determinado time, segundo as configurações de mando de campo e os oponentes definidos na matriz base. O método baseado em *backtracking* é iniciado escolhendo-se aleatoriamente um oponente do primeiro time. Definido esse oponente, atualiza-se a matriz de domínios para que um time que já foi alocado não jogue mais contra um outro time na mesma rodada. Nas próximas iterações, o método consiste em escolher o oponente do próximo time e atualizar a matriz de domínios. Caso o domínio do próximo time seja vazio, deve-se fazer um processo de *backtracking*, que consiste em desalocar o jogo do time anterior, atualizar a matriz de domínios desconsiderando o oponente desalocado e escolher um outro oponente para esse time. Caso não haja oponentes, continua-se o processo de *backtracking* até que se encontre um outro oponente para o time corrente. Ao completar uma rodada, adiciona-se essa rodada à base e o processo de *backtracking* é forçado até que se atinja o primeiro time. Assim, uma nova rodada será iniciada a partir do próximo oponente do primeiro time. O método é encerrado após a formação das quatro rodadas. Se o domínio do primeiro time não possuir mais nenhum oponente e as quatro rodadas ainda não tiverem sido completadas, reinicia-se esta fase.

Ao final da primeira fase, duas das quatro rodadas irão formar a primeira e a segunda rodada, sendo que a segunda rodada terá a configuração de mando de campo inversa à da primeira rodada. As duas outras rodadas irão formar a penúltima e última rodada do primeiro turno, sendo que a configuração de mando de campo dessas duas últimas será inversa à configuração de mando de campo das duas primeiras.

Para exemplificar esse procedimento, considere uma competição envolvendo 8 times. A matriz base e a matriz de domínios para este caso é mostrada, respectivamente, nas Tabelas 4.19 e 4.20.

Tabela 4.19 – Matriz base inicial

Times	MC	Rodadas			
1	+	4			
2	-	5			
3	+	6			
4	-	1			
5	+	2			
6	-	3			
7	+	8			
8	-	7			

Tabela 4.20 – Matriz de domínios

Times	Domínios		
1	2	6	8
2	1	3	7
3	2	4	8
4	3	5	7
5	4	6	8
6	1	5	7
7	2	4	6
8	1	3	5

Na Tabela 4.19, a coluna *Times* representam os times, enquanto a coluna *MC* representa o mando de campo segundo a representação de Anagnostopoulos *et al*. (2003). Cada célula da coluna *Rodadas* representa um oponente do respectivo time. Na Tabela 4.20, a coluna *Domínios* representa os oponentes possíveis para o respectivo time da coluna *Times*.

O método baseado em *backtracking* é iniciado pela escolha aleatória de um oponente para o time 1, no caso, o time 2. Essa iteração é ilustrada na Tabela 4.21.

Tabela 4.21 – 1ª Iteração

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5				1	3	7
3	+	6				2	4	8
4	-	1				3	5	7
5	+	2				4	6	8
6	-	3				1	5	7
7	+	8				2	4	6
8	-	7				1	3	5

A Tabela 4.22 apresenta a matriz de domínios atualizada.

Tabela 4.22 – Matriz de domínios atualizada para a 1ª iteração

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5				1		
3	+	6				4	8	
4	-	1				3	5	7
5	+	2				4	6	8
6	-	3				5	7	
7	+	8				4	6	
8	-	7				3	5	

Nas próximas iterações, o método consiste em escolher o oponente do próximo time e atualizar a matriz de domínios. As cinco iterações seguintes são apresentadas nas Tabelas 4.23, 4.24, 4.25, 4.26 e 4.27.

Tabela 4.23 – 2ª Iteração: Alocação do oponente do time 2

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6				4	8	
4	-	1				3	5	7
5	+	2				4	6	8
6	-	3				5	7	
7	+	8				4	6	
8	-	7				3	5	

Tabela 4.24 – 3ª Iteração: Alocação do oponente do time 3

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1				3		
5	+	2				6	8	
6	-	3				5	7	
7	+	8				6		
8	-	7				5		

Tabela 4.25 – 4ª Iteração: Alocação do oponente do time 4

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2				6	8	
6	-	3				5	7	
7	+	8				6		
8	-	7				5		

Tabela 4.26 – 5ª Iteração: Alocação do oponente do time 5

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2	6			8		
6	-	3				5		
7	+	8						
8	-	7						

Tabela 4.27 – 6ª Iteração: Alocação do oponente do time 6

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2	6			8		
6	-	3	5					
7	+	8						
8	-	7						

Observando-se a Tabela 4.27, pode-se verificar que na próxima iteração (7ª) não existe nenhum oponente para jogar contra o time 7, ou seja, o domínio do time 7 é vazio. Nesse caso, deve-se fazer o processo de *backtracking*. As duas iterações seguintes são mostradas nas Tabelas 4.28 e 4.29.

Tabela 4.28 – 7ª Iteração: *Backtracking* 1

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2	6			8		
6	-	3						
7	+	8						
8	-	7						

Tabela 4.29 – 8ª Iteração: *Backtracking* 2

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2	8					
6	-	3				7		
7	+	8				6		
8	-	7				5		

Observa-se na Tabela 4.29 que o time 5 ainda pode jogar contra o time 8. Assim, define-se o time 8 para jogar contra o time 5 e continua-se alocando os oponentes para os times restantes. A Tabela 4.30 mostra o resultado da 11ª iteração:

Tabela 4.30 – 11ª Iteração: Alocação do oponente do time 8

Times	MC	Rodadas				Domínios		
1	+	4	2			6	8	
2	-	5	1					
3	+	6	4			8		
4	-	1	3					
5	+	2	8					
6	-	3	7					
7	+	8	6					
8	-	7	5					

Como na 11ª iteração foi possível construir uma rodada, o método finaliza a formulação da rodada corrente e inicia a próxima etapa. Essa etapa consiste em elaborar a 3ª rodada escolhendo, aleatoriamente, um time pertencente ao domínio corrente do time 1 e atualizando a matriz de domínios. A Tabela 4.31 mostra a primeira iteração para a construção da terceira rodada.

Tabela 4.31 – 1ª Iteração: Alocação do oponente do time 1 para a terceira rodada

Times	MC	Rodadas				Domínios		
1	+	4	2	6		8		
2	-	5	1			3	7	
3	+	6	4			2	4	8
4	-	1	3			3	5	7
5	+	2	8			4	8	
6	-	3	7			5	7	
7	+	8	6			2	4	
8	-	7	5			3	5	

Assim, continua-se o processo sucessivamente até que sejam obtidas as quatro rodadas. Caso as quatro rodadas não possam ser obtidas, reinicia-se o método gerando-se uma nova rodada inicial aleatória, com seus respectivos mandos de campos também definidos aleatoriamente. A Tabela 4.32 mostra as rodadas obtidas pela primeira fase do método de geração da solução inicial.

Tabela 4.32 – Rodadas geradas pela primeira fase do método

Times	MC	Rodadas			
1	+	4	2	6	8
2	-	5	1	7	3
3	+	6	4	8	2
4	-	1	3	5	7
5	+	2	8	4	6
6	-	3	7	1	5
7	+	8	6	2	4
8	-	7	5	3	1

Essas rodadas irão compor as duas primeiras e duas últimas rodadas do primeiro turno, sendo que a primeira e a última rodada terão a mesma configuração de mando de campo estabelecido em *MC* e a segunda e a penúltima rodada terão a configuração inversa de mando de campo de *MC*. A Tabela 4.33 mostra a escala obtida pela primeira fase do método de geração da solução inicial.

Tabela 4.33 – Configuração da escala ao final da primeira fase

Time \ Rodada	1	2	3	4	5	6	7
1	+ 4	- 2				- 6	+ 8
2	- 5	+ 1				+ 7	- 3
3	+ 6	- 4				- 8	+ 2
4	- 1	+ 3				+ 5	- 7
5	+ 2	- 8				- 4	+ 6
6	- 3	+ 7				+ 1	- 5
7	+ 8	- 6				- 2	+ 4
8	- 7	+ 5				+ 3	- 1

4.5.2.2 Segunda fase

Na segunda fase do método de geração da solução inicial, geram-se as rodadas intermediárias do primeiro turno ainda não definidas, através de um processo de *backtracking* análogo ao procedimento utilizado na Seção 3.4.1, diferindo-se nos aspectos apontados a seguir. A matriz base não possui nenhuma configuração de mando de campo pré-determinada. Conseqüentemente, a matriz de domínios é composta de todos os oponentes possíveis de um determinado time, excluindo-se aqueles alocados na primeira fase do método, e sem considerar um mando de campo específico. Para atribuir o mando de campo aos jogos das rodadas geradas nessa fase, utiliza-se um procedimento que procura alternar o mando de campo da seqüência de jogos de cada time, tal como em Ribeiro e Urrutia (2004).

A Tabela 4.34 mostra a escala resultante da segunda fase do método de geração da solução inicial.

Tabela 4.34 – Configuração da escala ao final da segunda fase

Time \ Rodada	1	2	3	4	5	6	7
1	+ 4	- 2	+ 3	- 5	- 7	- 6	+ 8
2	- 5	+ 1	+ 4	+ 6	- 8	+ 7	- 3
3	+ 6	- 4	- 1	+ 7	+ 5	- 8	+ 2
4	- 1	+ 3	- 2	+ 8	- 6	+ 5	- 7
5	+ 2	- 8	+ 7	+ 1	- 3	- 4	+ 6
6	- 3	+ 7	- 8	- 2	+ 4	+ 1	- 5
7	+ 8	- 6	- 5	- 3	+ 1	- 2	+ 4
8	- 7	+ 5	+ 6	- 4	+ 2	+ 3	- 1

A seguir, essa escala é espelhada para formar o segundo turno do campeonato. A Tabela 4.35 mostra a solução inicial gerada pelo método de geração da solução inicial.

Tabela 4.35 – Solução inicial

Time \ Rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	+ 4	- 2	+ 3	- 5	- 7	- 6	+ 8	- 4	+ 2	- 3	+ 5	+ 7	+ 6	- 8
2	- 5	+ 1	+ 4	+ 6	- 8	+ 7	- 3	+ 5	- 1	- 4	- 6	+ 8	- 7	+ 3
3	+ 6	- 4	- 1	+ 7	+ 5	- 8	+ 2	- 6	+ 4	+ 1	- 7	- 5	+ 8	- 2
4	- 1	+ 3	- 2	+ 8	- 6	+ 5	- 7	+ 1	- 3	+ 2	- 8	+ 6	- 5	+ 7
5	+ 2	- 8	+ 7	+ 1	- 3	- 4	+ 6	- 2	+ 8	- 7	- 1	+ 3	+ 4	- 6
6	- 3	+ 7	- 8	- 2	+ 4	+ 1	- 5	+ 3	- 7	+ 8	+ 2	- 4	- 1	+ 5
7	+ 8	- 6	- 5	- 3	+ 1	- 2	+ 4	- 8	+ 6	+ 5	+ 3	- 1	+ 2	- 4
8	- 7	+ 5	+ 6	- 4	+ 2	+ 3	- 1	+ 7	- 5	- 6	+ 4	- 2	- 3	+ 1

Com esse procedimento é possível gerar uma solução inicial satisfazendo as restrições 3 e 4 descritas no Seção 3.3. Caso não houvesse esse tratamento especial, essas restrições seriam difíceis de serem contempladas pelo método de refinamento, uma vez que os movimentos utilizados não são fortes o suficiente para eliminar, a partir de qualquer solução inicial, as inviabilidades decorrentes do não atendimento a essas restrições. Assim, para alcançar bons resultados no trabalho proposto, foi implementado os dois métodos de geração de solução inicial descritos nas Seções 4.5.1 e 4.5.2, tendo o segundo alcançado melhores resultados do que o primeiro. Constatou-se durante a pesquisa que o primeiro método proposto, juntamente com as estruturas de vizinhança utilizadas, não possuía uma configuração adequada para que um método de refinamento pudesse contemplar as inviabilidades 3 e 4 descritas na Seção 3.3 e, assim, obter uma solução viável.

4.5.3 Adaptação do *Simulated Annealing* para geração da solução inicial

Outra forma pesquisada para gerar uma solução inicial é utilizar uma adaptação da representação de Biajoli *et al*. (2004), no qual uma solução é representada por uma matriz de *nr* linhas por *nj* colunas, composto por $nr = 2 \times n - 2$ rodadas e $nj = n/2$ jogos em cada rodada, onde *n* representa o número de times.

A adaptação se refere à representação de um jogo. Neste trabalho, um jogo segue a notação *Ti x Tj*, cujo significado é que o time *Ti* joga em sua sede contra o time *Tj*.

A Tabela 4.36 exemplifica uma tabela na representação de Biajoli *et al*. (2004) adaptada para a geração de uma solução inicial.

Tabela 4.36 – Representação de uma tabela de jogos

Rodada \ Jogo	1	2	3
1	1 x 3	4 x 2	5 x 6
2	2 x 6	4 x 1	5 x 3
3	4 x 5	3 x 6	2 x 1
4	6 x 1	3 x 4	2 x 5
5	6 x 2	1 x 4	3 x 5
6	1 x 2	6 x 3	5 x 4
7	4 x 3	5 x 2	1 x 6
8	5 x 1	4 x 6	2 x 3
9	2 x 4	6 x 5	3 x 1
10	6 x 4	3 x 2	1 x 5

Diferentemente da solução de Biajoli *et al*. (2004), esta representação não utiliza um sinal para representar o mando de campo. Neste caso, observa-se que na rodada 1, encontra-se o jogo *1x3*, que significa que o time 1 joga dentro de casa contra o time 3. O jogo inverso, no caso, está expresso na rodada 9 (*3x1*).

A solução inicial, então, é gerada da seguinte forma: Aleatoriamente, distribuem-se todos os jogos, em rodadas quaisquer. Isto, evidentemente, pode fazer com que um time jogue contra mais de um oponente em uma mesma rodada. Caso isto ocorra, é necessário remover este tipo de inviabilidade (denotada por inv^{sg}).

Para isso, propõe-se a aplicação de uma adaptação da metaheurística *Simulated Annealing*, indicada pela Figura 2.4, até que esta inviabilidade seja removida. As estruturas de vizinhança N^1, N^2 e N^3, descritas na Seção 4.3, são utilizadas para a geração de soluções vizinhas e, adicionalmente é considerado mais uma estrutura de vizinhança N^7, composto pelo movimento *swap games*, que efetua a troca de dois jogos de duas rodadas distintas. A Tabela 4.37 exemplifica o movimento *swap games* para os jogos *4x5*, da rodada 3 e *6x3*, da rodada 6, aplicado à solução da Tabela 4.36. Note que esse movimento implica em duas inviabilidades do tipo inv^{sg}, uma na 3ª rodada e uma na 6ª rodada.

Tabela 4.37 – Movimento *swap games*

Rodada \ Jogo	1	2	3
1	1 x 3	4 x 2	5 x 6
2	2 x 6	4 x 1	5 x 3
3	6 x 3	3 x 6	2 x 1
4	6 x 1	3 x 4	2 x 5
5	6 x 2	1 x 4	3 x 5
6	1 x 2	4 x 5	5 x 4
7	4 x 3	5 x 2	1 x 6
8	5 x 1	4 x 6	2 x 3
9	2 x 4	6 x 5	3 x 1
10	6 x 4	3 x 2	1 x 5

A função de avaliação é a descrita na Seção 4.4 acrescida de mais uma parcela $\gamma \times inv^{sg}$, onde γ representa a penalidade pela ocorrência de cada inviabilidade do tipo inv^{sg}.

O pseudocódigo do algoritmo *Simulated Annealing* adaptado à geração de uma solução inicial é descrito na Figura 4.6.

1) Gere uma solução aleatória s_0, na representação descrita nessa seção (4.5.3)
2) Seja s_0^*, uma solução na representação de Anagnostopoulos *et al*. (2003), conforme descrita na Seção 4.2
3) Faça $s^* \leftarrow s_0$, onde s^* é a melhor solução encontrada até então.
4) Defina a temperatura inicial do sistema, T_0
5) enquanto $\left(T > \varepsilon; \varepsilon \to 0 \wedge inv^{sg} \neq 0\right)$, faça
   a) enquanto $\left(iterT > SA\max\right)$, faça
      i) $iterT \leftarrow iterT + 1$
      ii) Gere aleatoriamente um vizinho $s' \in V(s)$, onde $V = N^1 \cup N^2 \cup N^3 \cup N^7$
      iii) Calcule a variação da função objetivo, $\Delta = f(s') - f(s)$
      iv) Se $\Delta < 0$ (movimento de melhora), então $s \leftarrow s'$.
      v) Caso contrário ($\Delta \geq 0$), gere aleatoriamente um valor $x \in [0,1]$ e aceite o movimento somente se a expressão $x < e^{-\Delta/T}$ for satisfeita.
      vi) Se $f(s') < f(s^*)$, então atualize s^*
   b) Reduza a temperatura por um fator α, ou seja, $T \leftarrow \alpha \times T$
   c) $iterT \leftarrow 0$
6) $s_0 \leftarrow s^*$
7) $s_0^* \leftarrow transform(s_0)$, onde *transform* é uma função que converte uma solução na representação descrita nessa Seção (4.5.3) para uma solução equivalente na solução de Anagnostopoulos *et al*. (2003), conforme descrita na Seção 4.2.
8) Retorne, como solução inicial, a solução na representação de Anagnostopoulos *et al*. (2003), s_0^*

Figura 4.6 – Algoritmo *Simulated Annealing* adaptado à geração de uma solução inicial

Ao término da aplicação do *Simulated Annealing*, a solução obtida é, então, convertida para a representação de Anagnostopoulos *et al*. (2003), a qual é submetida a um processo de refinamento, no caso, através de uma das seguintes metaheurísticas: *Simulated Annealing* ou VND, por exemplo.

Este algoritmo, porém, se mostrou ineficiente para gerar uma solução, pois utiliza a metaheurística *Simulated Annealing*, demandando um tempo proibitivo para gerar uma escala viável. Para instâncias maiores, este método não é recomendado e, portanto, optou-se por utilizar os métodos descritos nas Seções 4.5.1 e 4.5.2.

4.6 SA aplicado ao PPJ

Para a resolulção do PPJ, considera-se uma solução inicial s_0 obtida a partir das técnicas apresentadas na Seção 4.5 e, logo após, aplica-se a metaheurística *Simulated Annealing* básica, descrita na Seção 2.2.1 com a seguinte modificação:

Um vizinho da solução s é gerado aleatoriamente a partir da vizinhança $N^k(s)$, onde k é o tipo da estrutura de vizinhança definida na Seção 4.3, também escolhida aleatoriamente.

O pseudocódigo do algoritmo modificado é apresentado na Figura 4.7.

1) Gere uma solução aleatória s_0
2) Faça $s^* \leftarrow s_0$, onde s^* é a melhor solução encontrada até então.
3) Defina a temperatura inicial do sistema, T_0
4) enquanto $(T > \varepsilon; \varepsilon \to 0)$, faça
   a) enquanto $(iterT > SA\max)$, faça
      i) $iterT \leftarrow iterT + 1$
      ii) Gere aleatoriamente um vizinho $s' \in N^k(s)$, onde k é o tipo da estrutura de vizinhança escolhido aleatoriamente.
      iii) Calcule a variação da função objetivo, $\Delta = f(s') - f(s)$
      iv) Se $\Delta < 0$ (movimento de melhora), então $s \leftarrow s'$.
      v) Caso contrário ($\Delta \geq 0$), gere aleatoriamente um valor $x \in [0,1]$ e aceite o movimento somente se a expressão $x < e^{-\Delta/T}$ for satisfeita.
      vi) Se $f(s') < f(s^*)$, então atualize s^*
   b) Reduza a temperatura por um fator α, ou seja, $T \leftarrow \alpha \times T$
   c) $iterT \leftarrow 0$
5) Retorne a melhor solução encontrada, s^*.

Figura 4.7 – Algoritmo *Simulated Annealing* aplicado ao PPJ

4.7 VND aplicado ao PPJ

Foi implementado também o método VND básico, apresentado na seção 2.2.4. Inicialmente, uma solução s_0 é gerada aleatoriamente através do Método de Três Fases descrito na seção 4.5.1. Inicialmente, considera-se a primeira estrutura de vizinhança N^1 e, a cada iteração do método, é selecionado o melhor vizinho *s*' pertencente à *k*-ésima estrutura de vizinhança N^k a saber, N^1, N^2, N^3, N^4 e N^5, apresentadas na seção 4.3. Se a solução vizinha *s*' for melhor que a solução corrente, então a busca continua a partir de *s*' e da primeira estrutura de vizinhança N^1. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança N^{k+1}, onde N^k é a estrutura de vizinhança corrente. O método pára quando todas as estruturas de vizinhança são analisadas.

O pseudocódigo do algoritmo VND é o mesmo apresentado na Figura 2.8, substituindo a linha (1) por "1) Considere as estruturas de vizinhança N^1, N^2, N^3, N^4, N^5 e N^6 descritas na Seção 4.3 e $r = 5$".

4.8 VNS/VND aplicado ao PPJ

Para resolver o PPJ, além do *Simulated Annealing*, é aplicado também um método híbrido composto pelas metaheurísticas VNS e VND. Neste caso, considera-se o VNS apresentado na Seção 2.2.5, sendo que a busca local é feita pelo VND, apresentado na seção 2.2.4. Para ambos os métodos, são consideradas as cinco estruturas de vizinhança N^1, N^2, N^3, N^4, N^5 e N^6, descritas na Seção 4.3.

Inicialmente, considera-se uma solução inicial arbitrária s_0 e a cada iteração seleciona-se aleatoriamente um vizinho *s'* pertencente à vizinhança da solução *s* corrente. Faz-se então uma busca local deste vizinho através do método VND. Se a solução ótima local, *s''*, for melhor que a solução *s* corrente, a busca continua de *s''* recomeçando da primeira estrutura de vizinhança N^1. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança N^{k+1}, onde N^k é a estrutura de vizinhança corrente. Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de processamento T_{max} e o número máximo de iterações VNS_{Max}.

O pseudocódigo do método VNS/VND é apresentado na Figura 4.8.

1) Considere as seguintes estruturas de vizinhança, N^1, N^2, N^3, N^4 e N^5 descritas na Seção 4.3 e $k\max = 6$.
2) Considere *T* como o tempo de processamento corrente do algoritmo e *iter* o número de iterações efetuadas pelo método.
3) $T \leftarrow 0$
4) Defina uma solução inicial s_0 e, então, faça $s \leftarrow s_0$
5) enquanto $((iter < VNS\max) \wedge (T < T\max))$ faça
   a) $k \leftarrow 1$ e $iter \leftarrow iter + 1$
   b) enquanto $(k \leq k\max)$ faça
      i) gere, aleatoriamente, um vizinho *s'* a solução *s*, pertencente a k-ésima estrutura de vizinhança ($s' \in N^k(s)$)
      ii) $s'' \leftarrow VND(s')$
      iii) se $f(s'') < f(s)$, então $s \leftarrow s''$ e $k \leftarrow 1$
      iv) caso contrário, $k \leftarrow k + 1$
   c) atualize o tempo de processamento, *T*.
6) retorne a solução *s*.

Figura 4.8 – Algoritmo VNS/VND aplicado ao PPJ

4.9 Busca Tabu aplicado ao PPJ

A metaheurística Busca Tabu utilizado para resolver o PPJ é o método básico descrito na Seção 2.2.2, sendo que a vizinhança *N* é composta pelas seis estruturas, N^1, N^2, N^3, N^4, N^5 e N^6 descritas na Seção 4.3.

O pseudocódigo do algoritmo Busca Tabu implementado é apresentado na Figura 4.9.

1) Gere uma solução inicial s_0 e faça $s^* \leftarrow s_0$, onde s^* é a melhor solução encontrada até então.
2) $iter \leftarrow 0$, onde *iter* é o contador do número de iterações efetuadas pelo método.
3) $melhorIter \leftarrow 0$, onde *melhorIter* é a iteração mais recente que gerou s^*.
4) $TList \leftarrow \emptyset$, onde *TList* é a lista tabu
5) enquanto $\left(\left(f(s) > f_{\min}\right) \wedge \left(iter - melhorIter < BT \max\right)\right)$ faça
   a) $iter \leftarrow iter + 1$
   b) Seja $s' \leftarrow s \oplus m$ o melhor vizinho da vizinhança $N(s)$, $N = \bigcup_{i=1}^{6} N^i$, tal que o movimento $m \notin T$ ou tal que $f(s') < f(s^*)$ (função de aspiração).
   c) Atualize a lista tabu: $T^k \leftarrow T^k$ - {*movimento mais antigo*} + {*movimento que gerou s'*}, onde *k* é a vizinhança do melhor vizinho.
   d) Mova a solução corrente para a solução vizinha: $s \leftarrow s'$
   e) Se $f(s') < f(s^*)$, então faça $s^* \leftarrow s'$ e
6) Retorne a melhor solução s^*

Figura 4.9 – Algoritmo Busca Tabu aplicado ao PPJ

Observa-se que na linha 5-b) da Figura 4.9, determina-se o melhor vizinho na vizinhança $N = \bigcup_{i=1}^{6} N^i$. Caso esse melhor vizinho esteja na vizinhança $N^k(s)$, então armazena-se o movimento reverso na lista tabu T^k, com $k = 1,2,\ldots,6$. Portanto, consideram-se seis listas tabu, uma para cada tipo de movimento.

4.10 ILS aplicado ao PPJ

Para refinar uma solução gerada pelo método descrito na Seção 4.5.2, foi proposto um método híbrido, ILS-MRD, composto pela metaheurística Busca Local Iterada (*Iterated Local Search*, ILS) (Lourenço *et al*., 2002) e pelo Método Randômico de Descida (MRD), sendo este utilizado como um mecanismo de busca local para o ILS. O pseudocódigo do procedimento ILS adaptado ao PPJE é apresentado na Figura 4.10.

procedimento ILS-MRD
  $s_0 \leftarrow SolucaoInicialAleatoria$
  $s \leftarrow MRD(s_0, IterMRD)$
  $kp \leftarrow kp_0$
  $iter \leftarrow 0$
  enquanto $\left(kp < kp_{\max}\right)$
    enquanto $\left(iter - melhorIter < iter_{\max}\right)$
      $iter \leftarrow iter + 1$

```
        s' ← perturbação(s, kp)
        s'' ← MRD(s', IterMRD)
        se (f(s'') < f(s)) faça
            s ← s''
            melhorIter ← iter
            kp ← kp_0
        fim-se
    fim-enquanto
    kp ← kp + delta
fim-enquanto
retorne s
```

Figura 4.10 – Algoritmo ILS-MRD

Esse algoritmo inicia-se gerando uma solução inicial aleatória, s_0, através do método descrito na Seção 4.5.1 ou do método desenvolvido na Seção 4.5.2. Em seguida, aplica-se um mecanismo de busca local, o Método Randômico de Descida (MRD). Nesse método escolhe-se aleatoriamente um tipo de movimento (dentre os apresentados na Seção 4.3) e, a partir dele, é gerado um vizinho aleatório. Se esse vizinho for melhor que a solução corrente, ele passa a ser a nova solução corrente. Caso contrário, outro vizinho é gerado. O método MRD pára quando um número máximo de iterações sem melhora, no caso, *IterMRD*, for atingido. A cada iteração do método ILS-MRD, gera-se uma perturbação na solução corrente. Essa perturbação consiste em realizar k movimentos em uma estrutura de vizinhança escolhida aleatoriamente, sendo k um número arbitrário compreendido entre um e kp. A essa nova solução vizinha s' , aplica-se o MRD, gerando uma solução vizinha ótima local, s''. Se a solução s'' for melhor que a solução s corrente, então s'' é aceita como a nova solução corrente e reinicia-se o valor de kp. Essa repetição é executada até que $iter_{\max}$ iterações sem melhora sejam realizadas. Quando isso ocorrer, incrementa-se o grau de perturbação, kp, em um fator *delta*, até que kp atinja seu valor máximo ($kp_{\max}$).

4.11 Reconexão por Caminhos aplicado ao PPJE

Para a resolução do PPJE, utiliza-se o método de Reconexão por Caminhos (*Path Relinking*) descrito na Seção 2.2.7 como uma estratégia de pós-otimização.

O pseudocódigo do algoritmo Reconexão por Caminhos é apresentado na Figura 4.11.

1) Seja a solução elite g a melhor solução encontrada pelo método heurístico usado
2) Seja s_0 uma solução inicial obtida por um dos métodos descritos nas Seções 4.5.1 e 4.5.2.
3) $s \leftarrow s_0$
4) enquanto $(|\Delta(s,g)| \neq 0)$ faça
   a) Atribua a g' a melhor solução obtida aplicando o melhor movimento de $\Delta(s,g)$ a s
   b) Exclua esse movimento de $\Delta(s,g)$
   c) $s \leftarrow g'$
   d) se $f(g') < f(g)$ faça
      i) $g \leftarrow g'$
5) $s \leftarrow g$
6) retorne a solução s.

Figura 4.11 – Pseudocódigo do algoritmo Reconexão por Caminhos

Inicialmente gera-se uma solução inicial aleatória a partir de um dos métodos descritos nas Seções 4.5.1 e 4.5.2 e define-se a solução elite g como sendo a melhor solução encontrada por um método heurístico de refinamento.

Em seguida, aplica-se à solução corrente o melhor movimento pertencente ao conjunto $\Delta(s,g)$. Os movimentos desse conjunto consistem em fixar um componente "atômico", ou seja, um jogo, da solução elite g na solução corrente s.

Para exemplificar um movimento do *Path Relinking*, considere as Tabelas 4.38 e 4.39 que mostram, respectivamente, uma solução corrente e uma solução elite para um campeonato envolvendo 6 times.

Tabela 4.38 – Solução corrente do *Path Relinking*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	- 2	+ 6	+ 4	- 5	+ 3	+ 2	- 6	- 4	+ 5	- 3
2	+ 1	+ 5	- 6	- 3	+ 4	- 1	- 5	+ 6	+ 3	- 4
3	+ 6	- 4	- 5	+ 2	- 1	- 6	+ 4	+ 5	- 2	+ 1
4	+ 5	+ 3	- 1	- 6	- 2	- 5	- 3	+ 1	+ 6	+ 2
5	- 4	- 2	+ 3	+ 1	+ 6	+ 4	+ 2	- 3	- 1	- 6
6	- 3	- 1	+ 2	+ 4	- 5	+ 3	+ 1	- 2	- 4	+ 5

Tabela 4.39 – Solução elite do *Path Relinking*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 3	+ 4	- 6	+ 2	- 5	- 3	- 4	+ 6	- 2	+ 5
2	- 4	+ 5	+ 3	- 1	+ 6	+ 4	- 5	- 3	+ 1	- 6
3	- 1	+ 6	- 2	- 5	- 4	+ 1	- 6	+ 2	+ 5	+ 4
4	+ 2	- 1	+ 5	- 6	+ 3	- 2	+ 1	- 5	+ 6	- 3
5	+ 6	- 2	- 4	+ 3	+ 1	- 6	+ 2	+ 4	- 3	- 1
6	- 5	- 3	+ 1	+ 4	- 2	+ 5	+ 3	- 1	- 4	+ 2

Nesse exemplo, considera-se que já foram realizadas duas iterações do *Path Relinking*, na qual foram fixados dois jogos (*2*x*5* na rodada 2 e *6*x*4* na rodada 4). As células hachuradas nas Tabelas 4.38 e 4.39 representam esses jogos.

Suponha-se agora que o próximo movimento *m* a ser realizado é fixar o confronto *4*x*2* na rodada 1. A Tabela 4.40 apresenta a solução corrente após a aplicação desse movimento na solução apresentada na Tabela 4.38.

Tabela 4.40 – Solução corrente após a aplicação do movimento *m*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 5	+ 6	+ 4	- 5	+ 3	- 5	- 6	- 4	+ 5	- 3
2	- 4	+ 5	- 6	- 3	+ 4	+ 4	- 5	+ 6	+ 3	- 4
3	+ 6	- 4	- 5	+ 2	- 1	- 6	+ 4	+ 5	- 2	+ 1
4	+ 2	+ 3	- 1	- 6	- 2	- 2	- 3	+ 1	+ 6	+ 2
5	- 1	- 2	+ 3	+ 1	+ 6	+ 1	+ 2	- 3	- 1	- 6
6	- 3	- 1	+ 2	+ 4	- 5	+ 3	+ 1	- 2	- 4	+ 5

Nessa Tabela, observa-se que a aplicação do movimento *m* gerou inconsistências nas rodadas 4 e 5 (e, consequentemente, nas rodadas 9 e 10, uma vez que o problema é espelhado). Dessa forma, é necessário efetuar o restabelecimento da consistência dessa tabela, o qual é realizado com o auxílio de uma árvore guia.

Antes de se iniciar a construção da árvore guia, converte-se a solução corrente para a representação de Biajoli *et al.* (2004) adaptada para a geração de uma solução inicial, descrita na Seção 4.3.3. A Tabela 4.41 mostra a solução da Tabela 4.40 convertida. Como o problema é espelhado, a atualização irá refletir automaticamente para o segundo turno e, dessa forma, utiliza-se somente o primeiro turno da solução para efetuar a atualização.

Tabela 4.41 – Solução convertida para a representação adaptada de Biajoli *et al.* (2004)

R \ J	1	2	3
1	4 x 2	1 x 5	3 x 6
2	1 x 6	2 x 5	4 x 3
3	6 x 2	5 x 3	1 x 4
4	3 x 2	6 x 4	5 x 1
5	2 x 4	1 x 3	5 x 6

Após a conversão da solução, inicia-se a construção da árvore, que é um tipo abstrato de dados auxiliar utilizada para guiar a atualização, uma vez que é necessário um mecanismo de *backtracking* para que o restabelecimento da consistência da solução seja sempre possível. Cada nó dessa árvore é composto de 5 campos, a saber, a rodada afetada (R), o jogo a ser inserido na rodada afetada (JI), os jogos que devem ser inseridos (JAI), os jogos que devem ser removidos (JAR) e uma variável de controle (ST) que será verdadeiro (T) se existir um caminho viável de atualização e falso (F) caso contrário. A Figura 4.12 mostra a raiz da árvore após o preenchimento de seus campos, considerando o movimento *m* realizado.

R	1	
JI	4 x 2	
JAI	2 x 1	4 x 5
JAR	4 x 2	1 x 5
ST		

Figura 4.12 – Primeira iteração da atualização da solução do *Path Relinking*

De acordo com a raiz da árvore (Figura 4.12), na rodada 1 foi inserido o jogo *4*x*2*. Com essa inserção, é necessário efetuar o restabelecimento da consistência dessa rodada, uma vez que haviam os confrontos *2*x*1* e *4*x*5*. Como esses jogos foram retirados da rodada, será necessário inseri-los em outra rodada da solução. Para restabelecer a viabilidade da rodada 1, cria-se um novo confronto entre os times restantes dos jogos desfeitos, isto é, cria-se um confronto entre o time 1 e o time 5 e também o fixa na rodada. Como esse novo confronto já existe na solução, será necessário retirá-lo da rodada na qual estava inserido. A Tabela 4.42 mostra a solução do problema após o restabelecimento da consistência da primeira rodada.

Tabela 4.42 – Solução após o restabelecimento da consistência da rodada 1

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	- 5	+ 6	+ 4	- 5	+ 3	+ 5	- 6	- 4	+ 5	- 3
2	- 4	+ 5	- 6	- 3	+ 4	+ 4	- 5	+ 6	+ 3	- 4
3	+ 6	- 4	- 5	+ 2	- 1	- 6	+ 4	+ 5	- 2	+ 1
4	+ 2	+ 3	- 1	- 6	- 2	- 2	- 3	+ 1	+ 6	+ 2
5	- 4	- 2	+ 3	+ 1	+ 6	+ 4	+ 2	- 3	- 1	- 6
6	- 3	- 1	+ 2	+ 4	- 5	+ 3	+ 1	- 2	- 4	+ 5

Após a conclusão do remanejamento da rodada 1, o passo seguinte é localizar o primeiro jogo que se encontra no campo JAR do nó raiz. De acordo com esse campo, o próximo jogo a ser removido é o jogo *4*x*2*. Logo, deve-se localizar a rodada onde esse jogo se encontra e construir o nó seguinte com os dados referentes às modificações ocorridas na rodada, de maneira similar às efetuadas na raiz. A Tabela 4.43 mostra a solução após a inserção do jogo 2 x 1 na rodada 5.

Tabela 4.43 – Solução após a inserção do jogo *2*x*1* na rodada 5

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	- 5	+ 6	+ 4	- 5	- 2	+ 5	- 6	- 4	+ 5	+ 2
2	- 4	+ 5	- 6	- 3	+ 1	+ 4	- 5	+ 6	+ 3	- 1
3	+ 6	- 4	- 5	+ 2	+ 4	- 6	+ 4	+ 5	- 2	- 4
4	+ 2	+ 3	- 1	- 6	- 3	- 2	- 3	+ 1	+ 6	+ 3
5	+ 1	- 2	+ 3	+ 1	+ 6	- 1	+ 2	- 3	- 1	- 6
6	- 3	- 1	+ 2	+ 4	- 5	+ 3	+ 1	- 2	- 4	+ 5

A Figura 4.13 mostra a árvore guia, após a criação do nó referente à localização e remoção do confronto *4*x*2*.

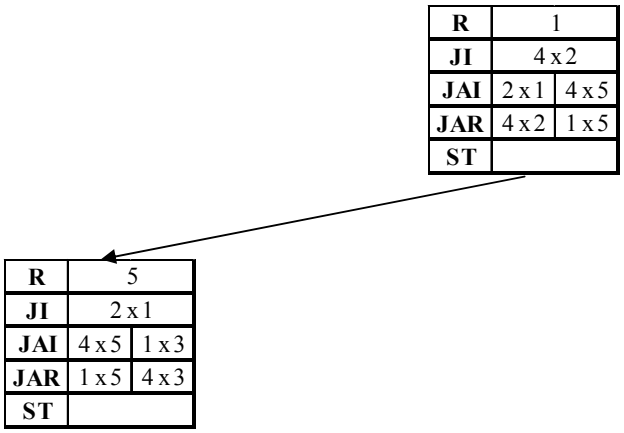

Figura 4.13 – Nó criado após a localização do confronto *4*x*2* e a inserção do confronto *2*x*1*

De acordo com a Figura 4.13, com a retirada do confronto *4*x*2* e a inserção do jogo *2*x*1* na rodada 5, gerou-se uma inconsistência nessa rodada. Para corrigir essa inconsistência, removeu-se o jogo *1*x*3* dessa rodada e definiu-se o jogo resultante *4*x*3*, também inserido nessa rodada.

Dessa forma, o algoritmo de atualização prossegue criando os próximos nós da árvore. Isso é feito através da localização dos jogos que estão duplicados na solução e inserindo em seu lugar os jogos que foram retirados das rodadas inconsistentes. Esse procedimento é realizado até que campos JAI e JAR contenham os mesmos jogos sem levar em conta o mando de campo. Quando isso ocorre, a solução está, finalmente, atualizada.

Porém, quando a inserção de um jogo implica na exclusão de um jogo já fixado, essa inserção não pode ser efetuada e, dessa forma, há a necessidade de fazer o *backtracking*. Com isso, outro ramo da árvore é explorado (ou seja, tenta-se fazer a inserção do segundo jogo situado no campo JAI do último nó viável pesquisado). A Figura 4.14 ilustra a ocorrência de *backtracking*, já que um dos oponentes do jogo que se deseja fixar já está fixado em um outro confronto na mesma rodada.

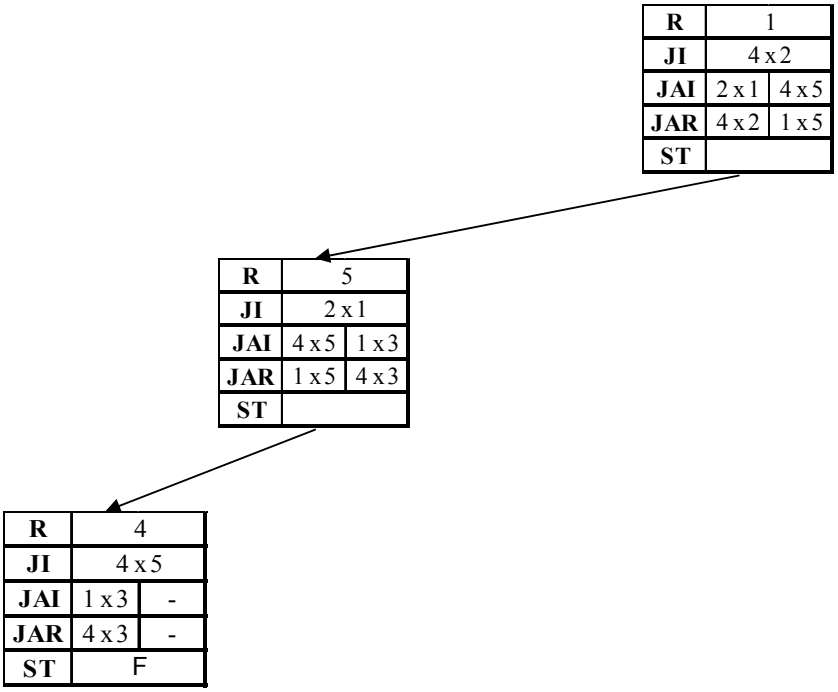

Figura 4.14 – Nó criado após a tentativa de inserção do jogo *4*x*5* na rodada 4.

De acordo com a Tabela 4.43 e a Figura 4.14, o próximo jogo a ser inserido é o *4*x*5*, na rodada 4. Porém, a inserção desse jogo implica na alteração do jogo *6*x*4*, o qual já foi fixado anteriormente, o que impossibilita essa inserção. Define-se, então, a variável de controle ST como F e realiza-se o *backtracking* para se explorar o outro ramo da árvore, referente à inserção do outro jogo localizado em JAI na mesma rodada. Como é possível a inserção do jogo *1*x*3* nessa rodada, efetua-se a inserção e a construção do próximo nó (como mostrado na Figura 4.15).

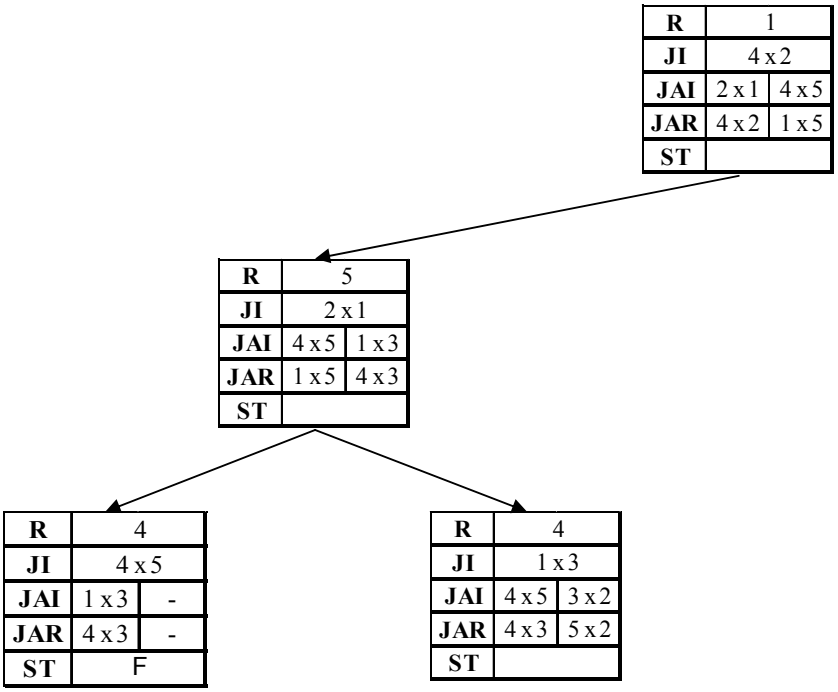

Figura 4.15 – *Bactracking* decorrente da inviabilidade de inserção do jogo *4*x*5* na rodada 4

Assim, prossegue-se a atualização da solução a partir desse novo ramo da árvore. Esses passos são realizados sucessivamente até que se consiga atualizar toda a tabela, retirando dessa todas as inconsistências de jogos duplicados. A Figura 4.16 mostra a árvore guia completa após o término da atualização da solução.

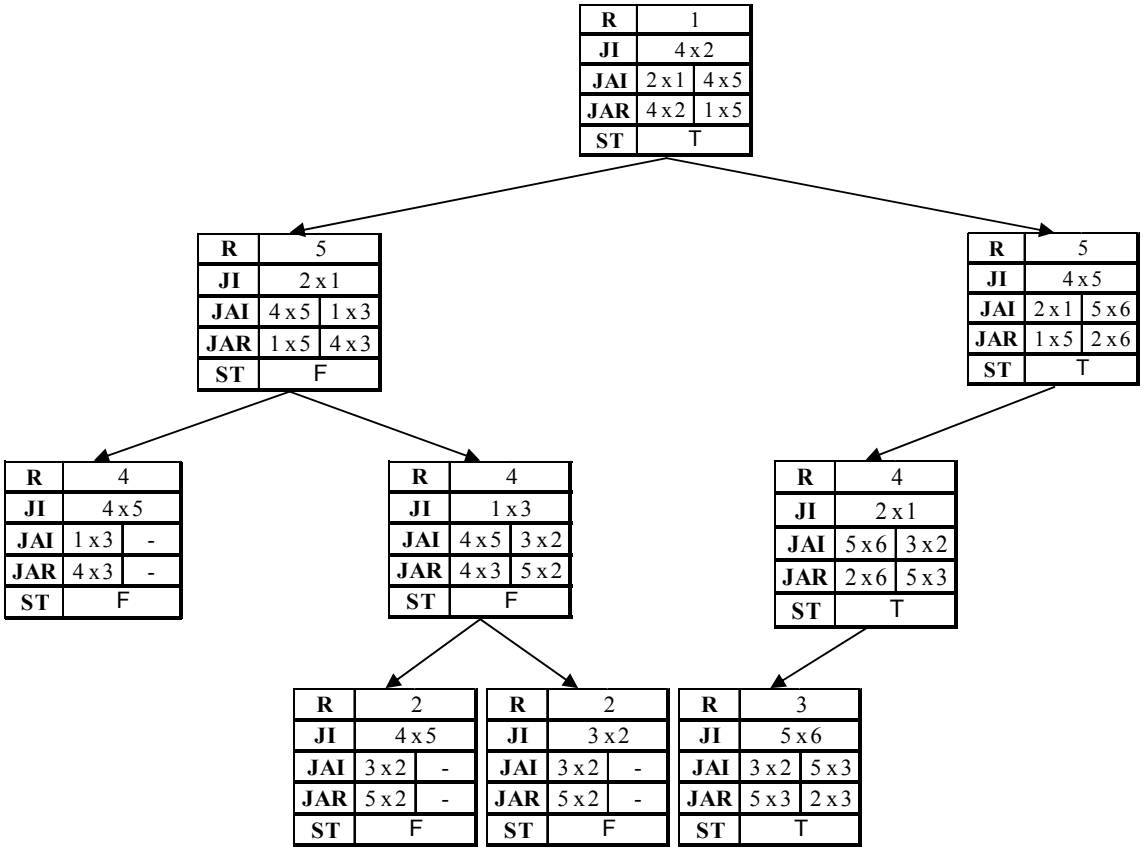

Figura 4.16 – Árvore guia completa após o término da atualização da solução

A Tabela 4.44 apresenta a solução da Tabela 4.38 após a aplicação do movimento *m*, sendo que as regiões hachuradas representam os jogos fixos.

Tabela 4.44 – Solução corrente atualizada após a aplicação do movimento *m*

Time \ Rodada	1	2	3	4	5	6	7	8	9	10
1	+ 5	+ 6	+ 4	- 2	+ 3	- 5	- 6	- 4	+ 2	- 3
2	- 4	+ 5	+ 3	+ 1	+ 6	+ 4	- 5	- 3	- 1	- 6
3	+ 6	- 4	- 2	- 5	- 1	- 6	+ 4	+ 2	+ 5	+ 1
4	+ 2	+ 3	- 1	- 6	+ 5	- 2	- 3	+ 1	+ 6	- 5
5	- 1	- 2	+ 6	+ 3	- 4	+ 1	+ 2	- 6	- 3	+ 4
6	- 3	- 1	- 5	+ 4	- 2	+ 3	+ 1	+ 5	- 4	+ 2

5 Resultados

Apresentam-se, nesse capítulo, os resultados obtidos aplicando-se o método ILS-MRD (*Iterated Local Search* e Método Randômico de Descida) para resolver o PPJE. O programa foi desenvolvido na linguagem C/C++ utilizando o ambiente Borland C++ Builder, versão 5.0. Foi utilizado um computador ATLHON XP 2,0 GHz com 256 MB de memória RAM, rodando no sistema operacional Windows XP.

Para validar os métodos, foram utilizadas três instâncias encontradas na literatura, disponíveis em http://www.decom.ufop.br/prof/marcone/projects/ttp/bssp.html. Essas instâncias referem-se aos dados dos times participantes da primeira divisão do campeonato brasileiro de futebol de 2004 (bssp2004), 2005 (bssp2005) e 2006 (bssp2006-A) e da segunda divisão do campeonato de 2006 (bssp2006-B). Observa-se que esta competição assumiu os moldes atuais a partir de 2004. Em 2003 a regra para a restrição 4, apontada na Seção 3.3, era diferente. Antes de 2003, o campeonato era realizado em turno único e os times não necessariamente jogavam em todas as rodadas. Em 2004 a competição reuniu 24 times de 9 estados da federação. Em 2005, quatro desses times desceram para a segunda divisão do campeonato e dois novos times ascenderam à primeira divisão, resultando em 22 times participantes do campeonato de 2005, representando 10 estados da federação. Em 2006, quatro desses times desceram para a segunda divisão do campeonato e dois novos times ascenderam à primeira divisão, resultando em 20 times participantes da primeira divisão do campeonato de 2006, representando 9 estados da federação. Já a segunda divisão do campeonato de 2006 teve 20 participantes representantes de 12 estados da federação. Para cada instância foram realizadas 100 execuções, cada qual partindo de uma semente diferente de números aleatórios. Os parâmetros adotados no método ILS-MRD foram: $kp_0 = 1$, $kp_{\max} = 5$, $iter_{\max} = 350$, $delta = 2$, $IterMRD = 1000$.

Na Tabela 5.1, *Melhor Valor* é o melhor valor encontrado na literatura e *Melhor*, *Média* e *Desvio* são, respectivamente, o melhor valor encontrado pelo método considerado, a média em 100 execuções e o desvio percentual obtido através da fórmula (6) e *TM* é o tempo médio, em segundos, de processamento do método em 100 execuções.

$$Desvio = 100 \times \frac{Média - MelhorValor}{MelhorValor} \tag{6}$$

Tabela 5.1 – Resultados computacionais obtidos pelo ILS-MRD

Instâncias	Melhor Valor	ILS-MRD			
		Melhor	Média	Desvio	TM
bssp2004	842789[(1)]	806134	825089	-2,1	1772
bssp2005	909119[(2)]	743621	760350	-16,4	1303
bssp2006-A	708964[(2)]	600514	631650	-10,9	834
bssp2006-B	1060130[(2)]	991222	1006146	-5,1	2068

[(1)] Biajoli *et al*. (2004); [(2)] CBF

Com base nos resultados obtidos na Tabela 5.1, observa-se claramente que a heurística ILS-MRD é um método eficiente e robusto para a resolução do PPJE, pois as soluções obtidas foram, na média, 2,1% melhores que a melhor solução encontrada na literatura para o campeonato de 2004, 16,4% melhores que a solução apresentada pela CBF para o campeonato de 2005 e 10,9% e 5,1% melhores que a solução elaborada pela CBF para, , respectivamente,

a primeira e segunda divisão e do campeonato de 2006. Através desse método foi possível obter soluções de alta qualidade, sendo que a melhor solução obtida para o campeonato de 2004 (primeira divisão), 2005 (primeira divisão), 2006 (primeira divisão) e 2006 (segunda divisão), apresentou melhora de, respectivamente, 4,4%, 18,2%, 15,4% e 6,5% em relação à melhor solução existente na literatura. Em termos de tempo computacional, o método também se mostrou eficiente, pois a instância bssp2004 foi resolvida em cerca de 30 minutos, enquanto o algoritmo de Biajoli *et al*. (2004) encontrou sua melhor solução em 45 minutos em um computador com processador AMD Athlon, 1,5 GHz e 256 MB de RAM.

A Tabela 5.2 mostra uma comparação entre a tabela elaborada manualmente pela Confederação Brasileira de Futebol (CBF), a tabela obtida pela heurística *Simulated Annealing* de Biajoli *et al*. (2004) e a tabela obtida pelo método proposto neste trabalho (ILS-MRD). Nessa tabela, *DIST* representa a distância total viajada por todos os times durante o campeonato, em quilômetros, *DIF* é a diferença de distância entre os times que percorreram o maior e o menor percurso e *FO* é o valor da função de avaliação *f* descrita na Seção 4.4.

Tabela 5.2 – Comparação entre as metodologias de resolução do PPJE

Instâncias	CBF			Biajoli *et al*. (2004)			ILS-MRD		
	DIST	DIF	FO	DIST	DIF	FO	DIST	DIF	FO
bssp2004	905316	86610	991926	789480	53309	842789	754935	51199	806134
bssp2005	838464	70655	909119	-	-	-	696800	46821	743621
bssp2006-A	658195	50769	708964	-	-	-	562886	37628	600514
bssp2006-B	998675	61454	1060129	-	-	-	967374	23848	991222

Analisando a Tabela 5.2, verifica-se que o método ILS-MRD encontra soluções melhores que as produzidas manualmente pela CBF e melhor que a solução obtida por Biajoli *et al*. (2004) para o campeonato de 2004. Observa-se que o método ILS-MRD obteve a sua melhor solução com redução na distância total percorrida por todos os times durante o campeonato de 2004 de 4,4% em relação à melhor solução encontrada na literatura (Biajoli *et al*., 2004) e 16,6% em relação à solução obtida pela CBF. Para o campeonato de 2005, o método reduziu esta distância em 16,9% em relação à tabela elaborada pela CBF. E para a primeira e segunda divisão do campeonato de 2006, o percentual de melhora foi de 14,5% e 3,14% respectivamente. Observa-se, também, que houve uma redução na diferença entre o time que mais viajou e menos viajou no campeonato de 2004 em 4,0% e 40,9%, respectivamente, em relação à solução de Biajoli *et al*. (2004) e em relação à solução da CBF. No campeonato de 2005, o método proposto reduziu esta diferença em 33,7% em relação à solução da CBF. Na primeira divisão do campeonato de 2006, o ILS-MRD reduziu obteve melhoras de 25,9% e na segunda divisão desse campeonato, a redução foi de 61,2%. Em outras palavras, considerando o custo de transporte aéreo a R$0,70/Km e uma delegação de 20 pessoas por cada time, a economia que o método ILS-MRD proporcionaria em relação à solução adotada pela CBF é de mais de R$ 2 milhões para o campeonato de 2004, quase R$ 2 milhões para o campeonato de 2005 e cerca de R$ 1,5 milhões e R$ 0,5 milhão para, respectivamente, a primeira e a segunda divisão do campeonato de 2006.

6 Situação atual do projeto

Todas as etapas do projeto foram concluídas. Apresenta-se, a seguir, o histórico de atividades desenvolvidas neste projeto de pesquisa.

Inicialmente, foi feita uma revisão de literatura sobre o assunto, apresentando-se as principais metodologias encontradas para resolver o problema.

Em seguida, coletou-se os problemas-teste disponíveis em http://mat.gsia.cmu.edu/TOURN relativos ao problema de programação de jogos da liga americana de beisebol e, adicionalmente, coletou-se um novo conjunto de problemas-teste, disponível em http://www.decom.ufop.br/prof/marcone/projects/ttp/bssp.html, relativos ao problema real de alocação de jogos da Primeira Divisão do Campeonato Brasileiro de Futebol dos anos de 2004, 2005 e 2006 e da Segunda Divisão do Campeonato de 2006.

Fez-se também um estudo sobre a melhor representação a ser adotada e, no caso, foi escolhida a representação de Anagnostopoulos *et al.* (2003), pois, em relação à representação de Biajoli *et al.* (2004), exige menor uso de memória e a velocidade de avaliação de uma solução é relativamente maior. Além disso, a representação de Anagnostopoulos *et al.* (2003) contempla uma restrição que não é verificada pela representação de Biajoli *et al.* (2004), a saber, de que cada time deve jogar somente uma vez em cada rodada.

Foram implementados três métodos de geração de solução inicial, o primeiro proposto por Ribeiro e Urrutia (2004), o segundo desenvolvido pelo autores (descrito em Silva *et al.*, 2005) e o terceiro, também desenvolvido pelos autores, surgiu de uma adaptação do *Simulated Annealing* e da representação de Biajoli *et al.* (2004) para se gerar uma solução inicial. Os dois primeiros métodos são capazes de gerar soluções iniciais com rapidez, porém o primeiro não é capaz de contemplar restrições importantes do problema espelhado, sendo assim descartado para resolver problemas dessa natureza. Em seu lugar foi utilizado o segundo método, o qual, além de contemplar tais restrições, gera soluções iniciais de boa qualidade. O terceiro método foi testado mas não usado devido ao alto esforço computacional para se gerar soluções iniciais e, ainda, de baixa qualidade. Esse esforço pode ser justificado pela utilização da metaheurística *Simulated Annealing* combinado com uma representação não muito favorável, a saber, a Biajoli *et al*., (2004).

Foi feita também uma adaptação das três estruturas de vizinhança, *swap rounds*, *swap teams* e *swap homes*, adotadas por Anagnostopoulos *et al*. (2003) para a resolução do problema de programação de jogos realizados em turnos espelhados. Os dois movimentos restantes, *partial swap rounds* e *partial swap teams*, fazem efeito somente no problema não espelhado, sendo que no problema espelhado eles se comportam, respectivamente, como o *swap rounds* e *swap teams*. Foi desenvolvida uma nova estrutura de vizinhança, a saber, *replace teams*, que consiste em substituir um determinado time por outro e vice-versa.

Foram implementadas, ainda, as heurísticas *Simulated Annealing*, VND, VNS/VND, Busca Tabu, *Path Relinking* e uma técnica híbrida composta pelos métodos ILS e MRD. Optou-se por apresentar formalmente apenas os testes relativos ao método ILS-MRD, o qual obteve o melhor desempenho e, ainda, pôde superar os melhores resultados existentes da literatura referentes ao problema do campeonato brasileiro. As demais heurísticas também foram testadas, porém nenhum resultado melhor do que aqueles divulgados na literatura foram encontrados e, portanto, estes foram omitidos.

Foi feito um estudo sobre a aplicação da heurística GRASP para o PPJ e PPJE. A única forma de aplicar essa heurística era utilizando o método de geração da solução inicial proposto por Ribeiro e Urrutia (2004). Esse método, porém, não é capaz de gerar uma solução

favorável à contemplação de todas as restrições do Campeonato Brasileiro de Futebol. Como havia um grande potencial em conseguir excelentes resultados resolvendo-se o PPJE e os resultados obtidos no projeto anterior para o PPJ se encontrava ainda muito distantes dos valores da literatura, optou-se por focalizar no projeto as instâncias reais do Campeonato Brasileiro de Futebol.

Com os métodos de geração de solução inicial e a metaheurística ILS-MRD, conseguiu-se obter resultados melhores que os existentes na literatura. Com isso, obteve-se duas importantes publicações: um artigo nos anais do Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM2005) e um resumo estendido nos anais do Encontro Regional de Matemática Aplicada e Computacional (ERMAC 2005).

Como maior contribuição deste projeto, foi desenvolvido uma metodologia de geração de solução inicial específica para o problema do campeonato brasileiro de futebol. Esta metodologia encontra-se descrita em http://www.decom.ufop.br/prof/marcone/projects/ttp/rt-decom-10-2005.pdf.

7 Conclusões e trabalhos futuros

Este trabalho aborda o problema de programação de jogos de competições esportivas realizados em dois turnos completos, espelhados ou não. O principal foco deste trabalho foi a resolução do problema de alocação de jogos da primeira divisão do Campeonato Brasileiro de Futebol dos anos 2004, 2005 e 2006 e da segunda divisão.do campeonato de 2006

Para a geração de uma solução inicial foi desenvolvido, inicialmente, uma adaptação da metaheurística *Simulated Annealing* que utiliza a representação de Biajoli *et al*. (2004). Esse método, porém, demandava muito tempo para gerar uma solução que pudesse ser convertida para a representação adotada neste trabalho, a saber, a de Anagnostopoulos *et al*. (2003). Dessa forma, essa abordagem foi descartada. Como conseqüência desse fato, foi pesquisado e implementado um novo método de geração de soluções iniciais, denominado Método das Três Fases, proposto por Ribeiro e Urrutia (2004). Essa abordagem permite gerar soluções rápidas e de qualidade satisfatória, considerando as restrições da Liga Americana de Beisebol, contudo ela não é capaz de gerar uma solução adequada de acordo com restrições do Campeonato Brasileiro de Futebol. Em outras palavras, as heurísticas de refinamento, juntamente com as estruturas de vizinhança desenvolvidas, não são capazes de contemplar a todas as restrições do Campeonato Brasileiro de Futebol, partindo-se de uma solução inicial gerada pelo método de Ribeiro e Urrutia (2004). Para contornar esse problema, foi desenvolvido o Método de Duas Fases, baseado em *backtracking*, cujo propósito é gerar uma solução inicial aleatória satisfazendo a duas das restrições do problema de programação de jogos do Campeonato Brasileiro de Futebol de 2004, 2005 e 2006, as quais não eram contempladas pelo método anterior.

Na fase de refinamento da solução foi utilizada uma adaptação da metaheurística *Iterated Local Search* (ILS), denominada ILS-MRD, que explora um subespaço de soluções ótimas locais através do Método Randômico de Descida (MRD). Esse método navega no espaço de soluções utilizando as estruturas de vizinhança descritas na Seção 4.3. Nessa heurística foi implementada uma técnica que consiste em aumentar o grau de perturbação de forma sistemática sempre que não havia melhora na solução corrente em um determinado número de iterações, retornando-se ao grau mínimo de perturbação quando ocorria alguma melhora. Essa técnica teve, na prática, influência significativa para melhorar a qualidade da solução final.

Com base nos resultados obtidos, observa-se que o ILS-MRD é um método robusto e eficiente, pois, em média, as soluções obtidas pelo método foram 2,1% melhor que a melhor solução encontrada na literatura para o campeonato de 2004, 16,4% melhor que a solução elaborada pela CBF para o campeonato de 2005 de 2005 e 10,9% e 5,1% melhores, respectivamente, para a primeira e segunda divisão e do campeonato de 2006 em relação às soluções elaboradas pela CBF. A melhor solução encontrada pelo método reduziu a distância total percorrida por todos os times durante o campeonato de 2004 em 4,4% em relação à melhor solução encontrada na literatura (Biajoli *et al*., 2004) e 16,6% em relação à solução obtida manualmente pela CBF. Para o campeonato de 2005 e para a primeira e a segunda divisão do campeonato de 2006, o método reduziu esta distância em, respectivamente, 16,9%, 10,9% e 5,1%, em relação à tabela elaborada pela CBF. Houve, também, uma redução de 4,0% em relação à solução de Biajoli *et al*. (2004) e 40,9% em relação à solução da CBF na diferença entre o time que mais viajou e o que menos viajou no campeonato. No campeonato de 2005, o método proposto reduziu esta diferença em 33,7% em relação à solução da CBF. E no campeonato de 2006, o ILS-MRD reduziu esta diferença em 25,9% e 61,2% para a

primeira e segunda divisão respectivamente. Portanto, pode-se concluir que o método ILS-MRD é um método adequado para resolver o PPJE.

Para trabalhos futuros, pode-se utilizar o método *Path Relinking* como uma estratégia de intensificação a cada ótimo local encontrado após a fase de busca local do ILS-MRD. Este seria executado, por exemplo, toda vez que um determinado número de iterações for atingido ou toda vez que uma solução melhor que a corrente é encontrada.

Pode-se utilizar também novas formas de representação de uma solução, como, por exemplo, a representação em grafos e, dessa forma, aumentar a velocidade dos métodos utilizando-se de algoritmos eficientes especificamente desenvolvidos para essa representação.

Referências Bibliográficas

ANAGNOSTOPOULOS, A.; MICHEL, L.; VAN HENTEENRYCK; VERGADOS, Y. (2003) A Simulated Annealing Approach to the Traveling Tournament Problem. Proceedings of CP'AI'OR'03, Montreal, Canada.

BEAN, J. C.; BIRGE, J. R. (1980) Reducting Traveling Costs and Player Fatigue in National Basketball Association. *Interfaces*, v. 10, p. 98-102.

BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F. (2003) Escala de jogos de torneios esportivos: uma abordagem via Simulated Annealing. *Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional*, SBPO, Natal, v. 1, p. 1295-1306.

BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F.; LUCENA, A.; CABRAL, L. A. F.; PONTES, R. C. V (2004) Scheduling the Brazilian Soccer Championship: A Simulated Annealing Approach. *Proceedings of Practice and Theory of Automated Timetabling V*, PATAT, Pittsburgh, USA, p. 433-436.

CONCÍLIO, R.; ZUBEN, F. J. (2002) Uma Abordagem Evolutiva para Geração Automática de Turnos Completos em Torneios. *Revista Controle & Automação*, v. 13, n. 2, p. 105-122.

COSTA, D. (1995) An Evolutionary Tabu Search Algorithm and the NHL Scheduling Problem. *INFORS*, v. 33, n. 3, p. 161-178.

CRAUWELS, H.; VAN OUDHEUSDEN, D. (2003) Ant Colony Optimization and Local Improvement. *Proceedings of CP'AI'OR'03*, Montreal, Canada.

de WERRA, D. (1988) Some Models of Graphs for Scheduling Sports Competitions, *Discrete Applied Mathematics*, v. 21, p. 47-65.

DORIGO, M.; NEMHAUSER, G.L.; TRICK, M.A. (2001) The Traveling Tournament Problem: Description and Benchmarks. *Lecture Notes in Computer Science*, v. 2239, p. 580-585.

EASTON, K.; NEMHAUSER, G. L.; TRICK, M. A. (2003) Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach. *Lecture Notes in Computer Science*, v. 2740, p. 100-109.

FEO, T. A.; RESENDE, M. G. C. (1995) Greedy randomized adaptive search procedures, *Journal of Global Optimization*, v. 6, p. 109-133.

GLOVER, F. (1996) Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges. *In* R. Barr, R. Helgason and J. Kenington (eds), *Interfaces in Computer Sciences and Operations Research*, p. 1-75, Kluwer Academic Publishers, Norwell, MA.

GLOVER, F. e LAGUNA, M. (1997) *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.

HENZ, M. (2001) Scheduling a Major College Basketball Conference: Revisited, *Operations Research*, v. 49, p. 1-12.

KIRKPATRICK, S.; GELLAT, D. C.; VECCHI, M. P. (1983) Optimization by Simulated Annealing. *Science*, v. 220, p. 671-680.

LOURENÇO, H. R., MARTIN, O., STÜTZLE, T. (2002) Iterated Local Search. *In* F. Glover and G. Kochenberger (eds), *Handbook of Metaheuristics*, p. 321-353, Kluwer Academic Publishers, Norwell, MA.

MLADENOVIC, N.; HANSEN, P. (1997) A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100.

NEMHAUSER, G. L.; TRICK, M. A. (1998) Scheduling a Major College Basketball Conference. *Operations Research*, v. 46, n. 1, p. 1-8.

RIBEIRO, C. C.; UCHOA, E.; WERNECK, R. F. (2002). A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*,

14:228_246.
RIBEIRO, C.; URRUTIA, S. (2004a). Heuristics for the Mirrored Traveling Tounament Problem. *Proceedings of Practice and Theory of Automated Timetabling V*, PATAT, Pittsburgh, USA, p. 323-342.
RIBEIRO, C.; URRUTIA, S. (2004b). OR on the ball: Applications in sports scheduling management. *OR/MS Today*, v. 31, p. 50-54.
RUSSELL, R. A.; LEUNG, J. M. (1994) Devising a cost effective schedule for a baseball league. *Operations Research*, v. 42, p. 614-625.
SCHAEFER, A. (1999). Scheduling Sport Tournaments using Constraint Logic Programming. *Constraints*, v. 4, p. 43-65.
SCHREUDER, J. A. M. (1992) Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues. *Discrete Applied Mathematics*, v. 35, p. 301-312.
ROSSETI, I. C. M. (2003) Estratégias seqüenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos. Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.
SILVA, M. S. A.; MINE, M. T.; SOUZA, M. J. F. (2005) Um método de geração de solução inicial, baseado em *backtracking*, para o problema de programação de jogos do Campeonato Brasileiro de Futebol. Relatório Técnico DECOM 10/2005, Departamento de Computação, Universidade Federal de Ouro Preto, 17 p. Disponível em http://www.decom.ufop.br/prof/marcone/projects/ttp/Publications/rt-decom-10-2005.pdf.
TRICK, M. A. (2001) A Schedule-Then-Break Approach to Sports Timetabling. *Lecture Notes in Computer Science*, v. 2079, p. 242-252.

Anexo A – Artigo publicado no ERMAC 2005

V ERMAC-R3 5º Encontro Regional de Matemática Aplicada e Computacional

19-21 de outubro de 2005
Universidade Potiguar – Natal/RN

Busca Local Iterada aplicada à resolução do Problema de Programação de Jogos de competições esportivas realizadas em dois turnos espelhados

Marcio Tadayuki Mine, Matheus de Souza Alves Silva, Marcone Jamilson Freitas Souza, Gustavo Peixoto Silva

Depto de Computação, ICEB, UFOP,
35400-000, Ouro Preto, MG
tadayuki@iceb.ufop.br, matheus@iceb.ufop.br, marcone@iceb.ufop.br, gustavo@iceb.ufop.br

Resumo: *Este trabalho aborda o problema de programação de jogos da primeira divisão do campeonato brasileiro de futebol. Este problema consiste em organizar uma tabela de jogos para os times participantes do campeonato. A competição é realizada em dois turnos, sendo os jogos do segundo turno realizados na mesma seqüência de confrontos do primeiro, porém com o mando de campo invertido. O objetivo principal é minimizar o deslocamento total efetuado pelos times envolvidos, respeitando-se um determinado conjunto de restrições, por exemplo, um time não pode jogar mais que duas vezes consecutivas em sua sede ou fora dela. Neste trabalho, o problema é abordado pela metaheurística "Iterated Local Search", usando-se quatro tipos diferentes de movimentos para explorar o espaço de soluções viáveis do problema. A metodologia desenvolvida é testada com as instâncias reais de 2004 e 2005 relativas ao Campeonato Brasileiro de Futebol da primeira divisão e comparada com as tabelas geradas pela entidade organizadora do evento, a Confederação Brasileira de Futebol (CBF), bem como com um resultado apresentado na literatura para o campeonato de 2004. Mostra-se que a metodologia proposta é capaz de melhorar significativamente as tabelas aplicadas pela CBF, além de superar o resultado da literatura.*

Palavras-chave: Problema de Programação de Jogos; Campeonato Brasileiro de Futebol; Otimização Combinatória; Metaheurísticas; *Iterated Local Search*.

1. Introdução

As competições esportivas despertam a atenção de um grande número de pessoas e, por isso, têm sido de grande interesse para a mídia, já que representam fontes geradoras de lucro. Por outro lado, a organização desses eventos envolve, em muitos casos, como por exemplo, o Campeonato Brasileiro de Futebol, somas fabulosas. Esses valores representam os gastos com os deslocamentos dos times participantes da competição, sendo de interesse, portanto, sua redução. Para a redução desses custos é de suma importância um bom escalonamento dos jogos. Além do aspecto financeiro, ressalta-se a importância de equilibrar, entre os times, a distância total percorrida por eles ao longo da competição, de maneira que o desempenho dos times não seja afetado de forma desigual pelo desgaste dos jogadores com os deslocamentos.

Na literatura, o problema de organizar uma tabela de jogos de uma competição esportiva realizada em vários locais ao longo de um determinado período é conhecido como Problema de Programação de Jogos (PPJ) ou *Traveling Tournament Problem* (TTP) (veja Dorigo *et al*. (2001)). O principal objetivo do problema é minimizar o deslocamento total efetuado pelos times envolvidos, respeitando-se um determinado conjunto de restrições. No caso mais comum, as rodadas são completas, isto é, todos os times jogam na rodada, e cada time joga contra todos os demais duas vezes, sendo uma na sua sede e outra na sede do adversário. O problema pode ser espelhado ou não espelhado. Será espelhado quando os jogos do segundo turno forem realizados na mesma seqüência dos jogos do primeiro turno, porém com mando de campo invertido. No problema não espelhado impede-se apenas que os dois confrontos entre dois times não sejam consecutivos. Para ambos os casos, há um conjunto de problemas-teste na literatura, introduzidos inicialmente por Dorigo

et al. (2001) e Easton *et al*. (2003), disponíveis em http://www.mat.gsia.cmu.edu/TOURN/, considerando a restrição adicional de que cada time não pode jogar mais do que três jogos consecutivos dentro ou fora de casa.

A alocação ótima dos jogos é uma tarefa complexa, pois se enquadra na classe de problemas NP-difíceis (veja Easton *et al*. (2003)) sendo, portanto, um problema altamente combinatório. De acordo com Concílio e Zuben (2002), para *n* times confrontando-se entre si em turnos completos, o número de tabelas (soluções) possíveis é dado pela fórmula (1).

$$(n-1)!(n-3)!(n-5)!...(n-(n-1))!\times 2^{(n-1)\times\frac{n}{2}} \qquad (1)$$

Para exemplificar a magnitude do espaço de soluções do problema, para uma competição com 20 times participantes há 2,9062 x 10^{130} tabelas possíveis. Dessa forma, a resolução desse problema exclusivamente por métodos exatos, está limitada, via de regra, a problemas de pequenas dimensões. Para dimensões mais elevadas, a abordagem mais comum é através de técnicas aproximadas ou heurísticas.

Costa (1995) desenvolveu um método híbrido combinando as técnicas Algoritmos Genéticos e Busca Tabu para resolver um problema de escalonamento de jogos da Liga Americana de Hóquei. Noções de Inteligência Artificial são introduzidas para direcionar o processo de evolução natural. A fase de mutação do Algoritmo Genético é trocada por uma busca no espaço de solução comandada pelo método Busca Tabu. Ao invés de uma mutação aleatória mudar os componentes, cada indivíduo é submetido a um processo de otimização separado antes de interagir novamente com outros membros da população.

Concílio e Zuben (2002) abordam o problema da montagem da tabela de jogos do Campeonato Paulista de Futebol de 1997, que envolveu 16 times, utilizando programação evolutiva. Essa competição foi realizada em turno único, com rodadas cheias.

Crauwels e Van Oudheusden (2003) desenvolveram um procedimento baseado em Colônia de Formigas para tratar o TTP não espelhado. Nesse procedimento, cada time é representado por uma formiga, a qual deixa uma certa quantidade de "feromônio" sempre que completa uma rota, isto é, sempre que termina todos os seus jogos. A quantidade de feromônio deixada em um arco da rota é inversamente proporcional à distância percorrida. Arcos com maior quantidade de feromônio são estimulados a serem freqüentados por outras formigas. Após uma solução viável ser obtida, é realizado um procedimento de busca local que utiliza duas estruturas de vizinhança, aplicadas na forma do Método VND (Mladenovic e Hansen, 1997).

Anagnostopoulos *et al*. (2003) implementaram a metaheurística *Simulated Annealing* (Kirkpatrick *et al*., 1983) com um mecanismo de reaquecimento da temperatura, cujo propósito é escapar de ótimos locais a baixas temperaturas. Implementaram, também, uma estratégia de oscilação, que consiste em variar o peso dado ao não atendimento às restrições do problema, alternando, dessa forma, a análise do espaço de soluções viáveis e inviáveis. Com essa metodologia, os autores obtiveram quase todos os melhores resultados da literatura para as instâncias do TTP não espelhado.

Biajoli *et al*. (2003) abordam com *Simulated Annealing* a primeira divisão do Campeonato Brasileiro de Futebol de 2002. Essa competição envolveu 26 times e foi realizada em turno único em 29 rodadas. O método proposto melhorou a solução adotada pela entidade organizadora do evento em 11,4%, tendo como referência o deslocamento total dos times.

Biajoli *et al*. (2004) trataram a primeira divisão do Campeonato Brasileiro de Futebol de 2004 também com o método *Simulated Annealing*. Após a execução do método é aplicado um procedimento de refinamento na solução resultante. Trata-se de uma busca local que consiste em substituir cada time *i* por um time *j* e vice-versa para todos os jogos envolvendo os times *i* e *j*. A única restrição analisada nesse último processo é a restrição de jogos entre times do mesmo estado da federação na última rodada. Em relação à escala feita manualmente pela CBF, o método provou sua eficiência reduzindo em 12,8% a distância total viajada pelos times e melhorou em 38,4% o balanceamento da distância viajada entre eles.

Ribeiro e Urrutia (2004a) desenvolveram um método híbrido utilizando as metodologias GRASP (Feo e Resende, 1995) e o método ILS (Lourenço *et al*., 2002) para resolver os problemas-teste da literatura relativos ao TTP espelhado e não espelhado, bem como uma instância bastante adaptada do Campeonato Brasileiro de Futebol de 2003. Para explorar o espaço de soluções do problema foram considerados quatro movimentos: (a) *Home-away swap* (HAS), que consiste em inverter o mando de campo de um jogo; (b) *Team swap* (TS), que troca todos os jogos de dois times em todas as rodadas; (c) *Partial swap rounds* (PSR), que considera quatro times e permuta os dois jogos entre eles de uma rodada r_1 por dois outros jogos entre eles de uma outra rodada r_2; (d) *Game rotation* (GR), que consiste em forçar a alocação de um jogo a uma rodada, seguida de uma atualização determinística da solução através do movimento nomeado como *ejection chain move*. Na fase de construção GRASP é utilizado uma

heurística constituída de três fases: Na primeira fase, uma escala é construída para o problema de turno único, com n times abstratos. Na segunda fase, os times reais, cuja distância entre suas sedes são pequenas, são preferencialmente associados aos times abstratos que possuem maior número de oponentes consecutivos. Na terceira fase, é estabelecido o local de realização de cada jogo. A fase de construção termina com um simples processo de busca local. Na fase de refinamento do GRASP é utilizada a metaheurística ILS, tendo como método de busca local o método VND (Mladenovic e Hansen, 1997), o qual utiliza os movimentos HAS, TS e PRS. As soluções ótimas locais são perturbadas com o movimento GR. Com esse método os autores superaram todas as melhores soluções da literatura para o TTP espelhado.

Várias outras metodologias são apresentadas na literatura para resolver problemas de programação de jogos. Problemas relacionados a competições de beisebol são abordados em Russel e Leung (1994); enquanto Nemhauser e Trick (1998), Henz (2001) e Bean e Birge (1980) tratam competições de basquetebol. Schreuder (1992) analisa aspectos combinatoriais na construção da tabela de jogos do Campeonato Alemão de Futebol. Schaefer (1999) e de Werra (1988) exploram o tema sem considerar uma aplicação específica.

Pelos resultados ilustrados na literatura afim, conclui-se que o escalonamento automatizado dos jogos em competições esportivas é, de grande importância, pois usualmente consegue obter reduções significativas nos custos envolvidos. Observa-se que as instâncias da literatura que retratam o problema espelhado, disponíveis em http://www.mat.gsia.cmu.edu/TOURN/, não consideram restrições comuns a competições esportivas latinas, como as brasileiras. Nesse sentido, este trabalho apresenta uma metodologia para resolver instâncias reais do problema de programação de jogos do Campeonato Brasileiro de Futebol. O método, denominado ILS-MRD, parte de uma solução inicial gerada por *backtracking*. A seguir, esta solução é refinada pela metaheurística *Iterated Local Search* (Lourenço *et al*., 2002), a qual considera, para a exploração do espaço de soluções, quatro tipos diferentes de movimentos.

Este trabalho está organizado como segue. Na seção 2 descreve-se o problema abordado. A seção 3 apresenta a metodologia abordada. A seção 4 apresenta os resultados encontrados e a seção 5 conclui o trabalho apontando para trabalhos futuros.

2. Descrição do Problema Abordado

O problema de programação de jogos de competições esportivas realizadas em dois turnos espelhados (PPJE) é o problema típico de alocação de jogos organizado pela Confederação Brasileira de Futebol (CBF). O problema é constituído da seguinte maneira: cada um dos n times participantes da competição joga duas vezes contra os demais, uma em sua sede e outra na sede do oponente, não necessariamente nessa ordem. No PPJE, os jogos são realizados em dois turnos completos (isto é, todos os times jogam em todas as rodadas), sendo que os jogos do segundo turno são os mesmos do primeiro turno, inclusive na mesma ordem, porém com mando de campo invertido. Assume-se que a cada time está associado um estádio, localizado em uma certa cidade, considerada sua sede, e que as distâncias entre as sedes dos times sejam conhecidas. Considera-se que cada time inicia o campeonato na sua sede, deslocando-se dela sua sede se fizer a primeira partida fora de casa e a ela retornando ao final do campeonato, caso faça a última partida fora de casa. Quando um time jogar duas partidas consecutivas fora de casa, assume-se que ele sairá da cidade do primeiro oponente diretamente para a cidade do segundo, sem retornar à sua sede. A Tabela 1 ilustra uma escala de jogos de uma competição envolvendo seis times.

Tabela 1: Exemplo de uma escala de jogos do PPJE

Rodadas									
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 x 5	3 x 2	4 x 6	5 x 6	5 x 3	5 x 1	2 x 3	6 x 4	6 x 5	3 x 5
4 x 2	1 x 6	2 x 5	2 x 1	6 x 2	2 x 4	6 x 1	5 x 2	1 x 2	2 x 6
6 x 3	5 x 4	3 x 1	4 x 3	1 x 4	3 x 6	4 x 5	1 x 3	3 x 4	4 x 1
1º Turno					**2º Turno**				

Na Tabela 1, as cinco primeiras rodadas correspondem aos jogos do primeiro turno e as cinco últimas correspondem aos jogos do segundo turno. Em cada célula há um confronto da forma $T_i \times T_j$, que significa que o time T_i joga na sua sede contra o time T_j. Por exemplo, na rodada 1 há a notação "1×5", que expressa que o time 1 joga em sua sede contra o time 5. Observa-se, também, que a primeira rodada do segundo turno (rodada 6) contém os jogos inversos aos jogos da primeira rodada do primeiro turno (rodada 1), ou seja, a primeira rodada do segundo turno é o "espelho" da primeira rodada do segundo turno. O mesmo ocorre para as demais rodadas do primeiro e segundo turno da tabela, caracterizando-se, assim, uma escala do PPJE.

São as seguintes as restrições consideradas no PPJE, válidas para os campeonatos da primeira divisão do Campeonato Brasileiro de Futebol em 2004 e 2005:

a) Cada time joga somente uma vez por rodada;
b) Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos;
c) Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos, sendo um em casa e o outro na casa do adversário;
d) As duas últimas rodadas de cada turno terão a configuração inversa das duas primeiras rodadas de cada turno com relação ao mando de campo;

e) Não poderá haver jogos entre times do mesmo estado na última rodada;
f) A diferença entre os jogos feitos em cada turno em casa e fora de casa de um time não pode ser maior que uma unidade;
g) Um time não pode jogar mais que duas vezes consecutivas dentro ou fora de casa.

O objetivo principal do PPJE é minimizar a distância total viajada por todos os times durante o campeonato. O objetivo secundário é minimizar a diferença entre a distância do time que mais viajou e o que menos viajou no campeonato, de forma a balancear a distância viajada por cada time.

Observa-se que no modelo de Ribeiro e Urrutia (2004a, 2004b), os autores trabalham apenas com o objetivo principal. A restrição (g) é também diferente, pois é permitido que cada time jogue até três vezes consecutivas dentro ou fora de casa. Além disso, os autores tratam o campeonato de 2003, cuja restrição (d) é diferente. Esses autores, entretanto, não mencionam as restrições consideradas. De qualquer forma, tratam um problema bastante diferente do real.

3. Metodologia

3.1. Representação de uma solução

Adota-se a representação de uma solução de Anagnostopoulos *et al*. (2003). Seja n o número de times envolvidos e $nr = 2n - 2$ o número de rodadas do campeonato. A solução nesta representação é uma matriz de n linhas por nr colunas. Cada linha referencia a um time T_i, cada coluna representa uma rodada R_k e cada célula (i, R_k) representa o oponente do time T_i na rodada R_k. Um sinal associado ao oponente denota o mando de campo. O sinal positivo indica que o time T_i irá jogar contra um oponente em sua própria sede. Um sinal negativo indica, por sua vez, que o time T_i jogará contra seu oponente na sede do time que está confrontando. A Tabela 2 exemplifica uma solução, considerando 6 times e 10 rodadas.

Tabela 2: Exemplo de solução na representação de Anagnostopoulos *et al*. (2003)

T \ R	1	2	3	4	5	6	7	8	9	10
1	+ 6	- 5	+ 4	+ 3	- 2	- 4	- 3	+ 2	+ 5	- 6
2	+ 5	- 3	+ 6	+ 4	+ 1	- 6	- 4	- 1	+ 3	- 5
3	- 4	+ 2	+ 5	- 1	+ 6	- 5	+ 1	- 6	- 2	+ 4
4	+ 3	+ 6	- 1	- 2	- 5	+ 1	+ 2	+ 5	- 6	- 3
5	- 2	+ 1	- 3	- 6	+ 4	+ 3	+ 6	- 4	- 1	+ 2
6	- 1	- 4	- 2	+ 5	- 3	+ 2	- 5	+ 3	+ 4	+ 1

Na Tabela 2, observa-se, por exemplo, que a notação +5, encontrada na primeira rodada do time 2, indica que o time 2 joga em sua sede contra o time 5, enquanto que na rodada 4 do time 3, encontra-se a notação -1, cujo significado é que o time 3 joga fora de sua sede (no caso, na casa do oponente) contra o time 1.

Nesta representação, a trajetória percorrida por um time durante o campeonato é determinada facilmente. Como exemplo, será mostrada a trajetória do time 6 na solução da Tabela 2, o qual inicia o campeonato dentro de sua sede e, na primeira rodada, desloca-se para a sede do time 1. Em seguida, desloca-se para a sede do time 4 (na rodada 2) sem retornar à sua própria sede. Na rodada 3, ele segue da sede do time 4 para a sede do time 2. Na quarta rodada, ele retorna à sua sede (pois o jogo será 6×5, na sede do time 6) e desloca-se para a sede do time 3, na quinta rodada e assim sucessivamente até a 10ª rodada, onde encerrará o campeonato em sua sede. Isso não ocorre com o time 2, pois seu último confronto será na sede do time 5. Neste caso, é necessário que o time 2 retorne à sua sede após a décima rodada, para que assim possa encerrar o campeonato. As trajetórias dos times 2 e 6 são descritas a seguir. A notação $T_i \xrightarrow{r} T_j$ indica que o time T_i desloca-se para a sede do time T_j na rodada r, e $T_i \mapsto T_j$ significa que o time T_j retorna da sede do time T_i à sua sede, no final do campeonato:

Trajetória do time 6:

$$6 \xrightarrow{1} 1 \xrightarrow{2} 4 \xrightarrow{3} 2 \xrightarrow{4} 6 \xrightarrow{5} 3 \xrightarrow{6} 6 \xrightarrow{7} 5 \xrightarrow{8} 6$$

Trajetória do time 2:

$$2 \xrightarrow{2} 3 \xrightarrow{3} 2 \xrightarrow{6} 6 \xrightarrow{7} 4 \xrightarrow{8} 1 \xrightarrow{9} 2 \xrightarrow{10} 5 \mapsto 2$$

3.2. Estruturas de Vizinhança

Para explorar o espaço de soluções do problema aplicam-se, neste trabalho, os movimentos *swap rounds*, *swap teams* e *swap homes*, utilizados por Anagnostopoulos *et al*. (2003), e mais um movimento, denominado *replace teams*, utilizado por Biajoli *et al*. (2004). O primeiro movimento consiste em trocar os jogos entre duas rodadas. O segundo, em trocar os jogos de todas as rodadas de dois times entre si, com exceção dos jogos em que eles se confrontam. O terceiro, em alterar a configuração de mando de campo de um determinado jogo e o quarto em renomear um determinado time por outro e vice-versa.

3.3. Função de avaliação

Uma solução s é avaliada pela função f apresentada pela fórmula (2). As duas primeiras componentes dessa função representam os objetivos propriamente ditos e a terceira penaliza o não atendimento às restrições do problema.

$$f(s) = \sum_{i \in T} custo(i) + dif(s) + \sum_{j \in C} w_j \times inv_j \qquad (2)$$

em que $f(s)$ é a função de avaliação; T é o conjunto dos times participantes da competição; C é o conjunto de restrições descritas na seção 2; $custo(i)$

é o custo de um time $i \in T$, conforme definido mais adiante; $dif(s)$ é a diferença entre o custo máximo e o custo mínimo dos times; w_j é a penalidade por desrespeitar a restrição $j \in C$; inv_j é o número de vezes que a restrição $j \in C$ está sendo desrespeitada.

O custo de um time $i \in T$ é definido da seguinte forma. Seja a solução s apresentada na Tabela 2 e d_{ij} a distância entre as sedes dos times i e j e seja $custo(i)$ o custo de deslocamento efetuado pelo time i em toda a competição, considerando que no seu início cada time sai da sua sede e à ela retorna ao final da competição. Assim, por exemplo, o custo de deslocamento do time 1 na solução da Tabela 2, é calculado pela fórmula (3).

$$custo(1) = d_{15} + d_{51} + d_{12} + d_{24} + d_{43} + d_{31} + d_{16} + d_{61} \quad (3)$$

3.4. Geração de uma solução inicial

Uma solução inicial é gerada por um procedimento de duas fases, ambas baseadas em *backtracking*. Na primeira, é feita a alocação das duas primeiras e duas últimas rodadas do primeiro turno. Na segunda fase são geradas as rodadas intermediárias desse turno. A seguir, gera-se o segundo turno mantendo-se a mesma seqüência de jogos do primeiro turno, porém com o mando de campo invertido. Com esse procedimento é possível gerar uma solução inicial satisfazendo as restrições (c) e (d) da seção 2. Caso não houvesse esse tratamento especial, essas restrições seriam difíceis de serem contempladas pelo método de refinamento, uma vez que os movimentos utilizados não são fortes o suficiente para eliminar, a partir de qualquer solução inicial, as inviabilidades decorrentes do não atendimento a essas restrições. Detalhes desse procedimento podem ser encontrados em Silva *et al*. (2005).

3.5. *Iterated Local Search* aplicada ao PPJE

Para refinar uma solução gerada pelo método descrito na seção 3.4, foi proposto um método híbrido, ILS-MRD, composto pela metaheurística Busca Local Iterada (*Iterated Local Search*, ILS) (Lourenço *et al*., 2002) e pelo Método Randômico de Descida (MRD), sendo este usado como um mecanismo de busca local para o ILS. O pseudocódigo do procedimento ILS adaptado ao PPJE é apresentado na Figura 1.

```
procedimento ILS-MRD
    s0 ← SolucaoInicialAleatoria
    s ← MRD(s0, IterMRD)
    kp ← kp0
    iter ← 0
    enquanto (kp < kpmax)
        enquanto (iter ← melhorIter < itermax)
            iter ← iter + 1
            s' ← pertubação(s, kp)
            s" ← MRD(s', IterMRD)
            se ( f(s") < f(s) ) faça
                s ← s"
                melhorIter ← iter
                kp ← kp0
            fim-se
        fim-enquanto
        kp ← kp + delta
    fim-enquanto
    retorne s
```

Figura 1: Algoritmo ILS-MRD

Esse algoritmo inicia-se gerando uma solução inicial aleatória, s_0, através do método descrito na seção 3.4. Em seguida, aplica-se um mecanismo de busca local, o Método Randômico de Descida (MRD). Nesse método escolhe-se aleatoriamente um tipo de movimento (dentre os quatro apresentados na seção 3.2) e, a partir dele, é gerado um vizinho aleatório. Se esse vizinho for melhor que a solução corrente, ele passa a ser a nova solução corrente. Caso contrário, outro vizinho é gerado. O método MRD pára quando um número máximo de iterações sem melhora, no caso, *IterMRD*, for atingido. A cada iteração do método ILS-MRD, gera-se uma perturbação na solução corrente. Essa perturbação consiste em realizar k movimentos em uma estrutura de vizinhança escolhida aleatoriamente, sendo k um número arbitrário compreendido entre um e kp. A essa nova solução vizinha s' , aplica-se o MRD, gerando uma solução vizinha ótima local, s". Se a solução s" for melhor que a solução s corrente, então s" é aceita como a nova solução corrente e reinicia-se o valor de kp. Essa repetição é executada até que $iter_{max}$ iterações sem melhora sejam realizadas. Quando isso ocorrer, incrementa-se o grau de perturbação, kp, em um fator *delta*, até que kp atinja seu valor máximo (kp_{max}).

4. Resultados Computacionais

Apresentam-se, nessa seção, os resultados obtidos aplicando-se o método ILS-MRD (*Iterated Local Search* e Método Randômico de Descida) para resolver o PPJE. O programa foi desenvolvido na linguagem C utilizando o ambiente Borland C++ Builder, versão 5.0. Foi utilizado um computador ATLHON XP 2,0 GHz com 256 MB de memória RAM, rodando no sistema operacional Windows XP.

Para validar os métodos, foram utilizadas duas instâncias encontradas na literatura, disponíveis em http://www.decom.ufop.br/prof/marcone/projects/ttp/bssp.html, que se referem à primeira divisão do Campeonato Brasileiro de Futebol de 2004 e 2005. Observa-se que esta competição assumiu os moldes atuais a partir de 2004. Em 2003 a regra para a restrição (d), apontada na seção 2, era diferente. Antes de 2003, o campeonato era realizado em turno único e os times não necessariamente jogavam em todas as rodadas. Em 2004 a competição reuniu 24 times de 9 estados da federação. Em 2005, quatro desses times desceram para a segunda divisão do campeonato e dois novos times ascenderam à primeira divisão, resultando em 22 times participantes do campeonato de 2005, representando 10 estados da federação. Para cada instância foram realizadas 100 execuções, cada qual partindo de uma semente diferente de números aleatórios. Os parâmetros adotados no método ILS-MRD foram: $kp_0 = 1$, $kp_{max} = 5$, $iter_{max} = 350$, $delta = 2$, $IterMRD = 1000$.

Na Tabela 8, *Melhor Valor* é o melhor valor encontrado na literatura até então e *Melhor*, *Média* e *Desvio* são, respectivamente, o melhor valor encontrado pelo método ILS-MRD, a média em 100 execuções e o desvio percentual, obtido por meio da fórmula (4), e *TM* é o tempo médio, em segundos, de processamento do método em 100 execuções:

$$Desvio = 100 \times \frac{Média - MelhorValor}{MelhorValor} \qquad (4)$$

Tabela 8: Resultados computacionais obtidos pelo ILS-MRD

Instâncias	*Melhor Valor*	ILS-MRD			
		Melhor	*Média*	*Desvio*	*TM*
bssp2004	842789[(1)]	806134	825089	-2,1	1772
bssp2005	909119[(2)]	743621	760350	-16,4	1303

[(1)] Biajoli *et al*. (2004); [(2)] CBF

A Tabela 9 compara, em relação à distância total percorrida (*DIST*), à diferença de distância entre o time que mais viajou e o que menos viajou (*DIF*) e ao valor da função *f* de avaliação (*FO*) descrita na seção 3.3, as tabelas produzidas pela Confederação Brasileira de Futebol (CBF), o método *Simulated Annealing* de Biajoli *et al*. (2004) e o método ILS-MRD.

Tabela 9: Comparação entre as metodologias de resolução do PPJE

		bssp2004	**bssp2005**
CBF	DIST	905316	838464
	DIF	86610	70655
	FO	991926	909119
Biajoli et al	DIST	789480	-
	DIF	53309	-
	FO	842789	-
ILS-MRD	DIST	754935	696800
	DIF	51199	46821
	FO	806134	743621

Pelos resultados mostrados nas tabelas 8 e 9, observa-se que o ILS-MRD é um método robusto e eficiente, pois, em média, as soluções obtidas pelo método foram 2,1% melhor que a melhor solução encontrada na literatura para o campeonato de 2004 e 16,4% melhor que a solução elaborada pela CBF para o campeonato de 2005. A melhor solução encontrada pelo método reduziu a distância total percorrida por todos os times durante o campeonato de 2004 em 4,4% em relação à melhor solução encontrada na literatura (Biajoli *et al*., 2004) e 16,6% em relação à solução obtida manualmente pela CBF. Para o campeonato de 2005, o método reduziu essa distância em 16,9% em relação à tabela elaborada pela CBF. Houve, também, uma redução de 4,0% em relação à solução de Biajoli *et al*. (2004) e 40,9% em relação à solução da CBF na diferença entre o time que mais viajou e o que menos viajou no campeonato. No campeonato de 2005, o método proposto reduziu essa diferença em 33,7% em relação à solução da CBF. Considerando o custo de transporte aéreo a R$0,70/Km e uma delegação de 20 pessoas por cada time, a economia que o método ILS-MRD proporcionaria em relação à solução adotada pela CBF é de mais de R$2 milhões para o campeonato de 2004 e quase R$2 milhões para o campeonato de 2005. Em termos de tempo computacional, o método também se mostrou competitivo, pois a instância bssp2004 foi resolvida em cerca de 30 minutos, enquanto o algoritmo de Biajoli *et al*. (2004) encontrou sua melhor solução em 45 minutos em um computador com processador AMD Athlon, 1,5 GHz e 256 MB de RAM.

5. Conclusões e Trabalhos Futuros

Este trabalho aborda o problema de programação de jogos da primeira divisão do Campeonato Brasileiro de Futebol de 2004 e 2005. Uma solução inicial é gerada por um método de duas fases, baseado em *backtracking*, cujo propósito é obter uma solução aleatória satisfazendo a duas das restrições do problema. Na fase de refinamento da solução foi utilizada uma adaptação da metaheurística *Iterated Local Search,*

denominada ILS-MRD, que explora um subespaço de soluções ótimas locais através do Método Randômico de Descida. Esse método navega no espaço de soluções utilizando quatro tipos de movimento, a saber, *swap teams*, *swap rounds*, *swap homes* e *replace teams*. O método ILS-MRD faz uso de um mecanismo que consiste em aumentar o grau de perturbação de forma sistemática sempre que não há melhora na solução corrente em um determinado número de iterações, retornando-se ao grau mínimo de perturbação quando ocorre alguma melhora. Esse mecanismo teve, na prática, influência significativa para melhorar a qualidade da solução final.

Os resultados obtidos mostram que o ILS-MRD é um método robusto e eficiente, pois mesmo as soluções médias foram melhores que as existentes na literatura. Mostrou-se, ainda, que a metodologia proposta é capaz de reduzir significativamente os gastos com transporte.

O método ILS é um método flexível, pois permite facilmente a substituição de seus componentes, como, por exemplo, os mecanismos de perturbação e busca local. Segundo Lourenço *et al*. (2002), o sucesso do ILS está centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha da busca local, das perturbações e do critério de aceitação. Logo, de acordo com esses autores, é ainda possível aperfeiçoar o método ILS-MRD, substituindo-se a heurística de busca local MRD por outras técnicas ou incorporando ao método outros procedimentos de busca. Uma possibilidade é o uso do procedimento Reconexão por Caminhos (*Path Relinking*), que consistiria em armazenar um conjunto de soluções elite ao longo da busca e, a partir de uma dada solução caminhar-se-ia no espaço de soluções considerando uma das soluções elite como solução guia. Outra possibilidade de melhoria é a incorporação de outros tipos de movimento na exploração do espaço de soluções, tais como *game rotation* e *partial swap rounds*, como em Ribeiro e Urrutia (2004).

O artigo pode conter agradecimentos, se necessários.

Agradecimentos

Os autores agradecem à FAPEMIG, processo CEX-429/04, pelo apoio ao desenvolvimento do trabalho.

Referências

[1] ANAGNOSTOPOULOS, A.; MICHEL, L.; VAN HENTEENRYCK; VERGADOS, Y. (2003) A Simulated Annealing Approach to the Traveling Tournament Problem. Proceedings of CP'AI'OR'03, Montreal, Canada.

[2] BEAN, J. C.; BIRGE, J. R. (1980) Reducting Traveling Costs and Player Fatigue in National Basketball Association. *Interfaces*, v. 10, p. 98-102.

[3] BIAJOLI, F. L. (2003) Resolução do Problema de Programação de Jogos do Campeonato Brasileiro de Futebol. Relatório Técnico DECOM/04, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em http:www.decom.ufop.br/prof/marcone. Acesso em 22/04/2005.

[4] BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F. (2003) Escala de jogos de torneios esportivos: uma abordagem via Simulated Annealing. *Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional*, SBPO, Natal, v. 1, p. 1295-1306.

[5] BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F.; LUCENA, A.; CABRAL, L. A. F.; PONTES, R. C. V (2004) Scheduling the Brazilian Soccer Championship: A Simulated Annealing Approach. *Proceedings of Practice and Theory of Automated Timetabling V*, PATAT, Pittsburgh, USA, p. 433-436.

[6] CAIN Jr, W.O. (1977) A Computer Assisted Heuristic Approach Used to Schedule the Major League Baseball Clubs. *In* Ladany, S.P. and Machol, R.E. (eds), *Optimal Strategies in Sports*, North-Holland, Amsterdan, p. 32-41.

[7] CAMPBELL, R. T.; CHEN, D. S. (1976) A Minimum Distances Basketball Scheduling Problem. *In* Machol, R.E., Ladany, S.P. and Morrison, D.G. (eds), *Management Science in Sports*, North-Holland, Amsterdan, p. 15-25.

[8] CONCÍLIO, R. (2000) *Contribuições à solução de problemas de escalonamento pela aplicação conjunta de computação evolutiva e otimização com restrições*. Dissertação de mestrado, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, São Paulo.

[9] CONCÍLIO, R.; ZUBEN, F. J. (2002) Uma Abordagem Evolutiva para Geração Automática de Turnos Completos em Torneios. *Revista Controle & Automação*, v. 13, n. 2, p. 105-122.

[10] COSTA, D. (1995) An Evolutionary Tabu

Search Algorithm and the NHL Scheduling Problem. *INFORS*, v. 33, n. 3, p. 161-178.

[11] CRAUWELS, H.; VAN OUDHEUSDEN, D. (2003) Ant Colony Optimization and Local Improvement. *Proceedings of CP'AI'OR'03*, Montreal, Canada.

[12] de WERRA, D. (1988) Some Models of Graphs for Scheduling Sports Competitions, *Discrete Applied Mathematics*, v. 21, p. 47-65.

[13] DORIGO, M.; NEMHAUSER, G.L.; TRICK, M.A. (2001) The Traveling Tournament Problem: Description and Benchmarks. *Lecture Notes in Computer Science*, v. 2239, p. 580-585.

[14] EASTON, K.; NEMHAUSER, G. L.; TRICK, M. A. (2003) Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach. *Lecture Notes in Computer Science*, v. 2740, p. 100-109.

[15] FEO, T. A.; RESENDE, M. G. C. (1995) Greedy randomized adaptive search procedures, *Journal of Global Optimization*, v. 6, p. 109-133.

[16] GLOVER, F. (1986) Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, v. 13, p. 533-549.

[17] GLOVER, F. (1989) Tabu Seach: Part I. *ORSA Journal on Computing*, v. 1, p.190-206.

[18] GLOVER, F. (1990) Tabu search: Part II. *ORSA Journal on Computing*, v. 2, p. 04-32.

[19] GOLDBERG, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Berkeley.

[20] HENZ, M. (2001) Scheduling a Major College Basketball Conference: Revisited, *Operations Research*, v. 49, p. 1-12.

[21] KIRKPATRICK, S.; GELLAT, D. C.; VECCHI, M. P. (1983) Optimization by Simulated Annealing. *Science*, v. 220, p. 671-680.

[22] LOURENCO, H. R., MARTIN, O., STÜTZLE, T. (2002) Iterated Local Search. *In* F. Glover and G. Kochenberger (eds), *Handbook of Metaheuristics*, p. 321 - 353, Kluwer Academic Publishers, Norwell, MA.

[23] MLADENOVIC, N.; HANSEN, P. (1997) A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100.

[24] NEMHAUSER, G. L.; TRICK, M. A. (1998) Scheduling a Major College Basketball Conference. *Operations Research*, v. 46, n. 1, p. 1-8.

[25] RIBEIRO, C.; URRUTIA, S. (2004a). Heuristics for the Mirrored Traveling Tounament Problem. *Proceedings of Practice and Theory of Automated Timetabling V*, PATAT, Pittsburgh, USA, p. 323-342.

[26] RIBEIRO, C.; URRUTIA, S. (2004b). OR on the ball: Applications in sports scheduling management. *OR/MS Today*, v. 31, p. 50-54.

[27] RUSSELL, R. A.; LEUNG, J. M. (1994) Devising a cost effective schedule for a baseball league. *Operations Research*, v. 42, p. 614-625.

[28] SCHAEFER, A. (1999). Scheduling Sport Tournaments using Constraint Logic Programming. *Constraints*, v. 4, p. 43-65.

[29] SCHREUDER, J. A. M. (1992) Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues. *Discrete Applied Mathematics*, v. 35, p. 301-312.

[30] SILVA, M. S. A.; MINE, M. T.; SOUZA, M. J. F. (2005) Um método de geração de solução inicial, baseado em *backtracking*, para o problema de programação de jogos do campeonato brasileiro de futebol. Relatório Técnico DECOM 10/2005, Departamento de Computação, Universidade Federal de Ouro Preto, 10 p. Disponível em http://www.decom.ufop.br/prof/marcone/projects/ttp/rt-decom-10-2005.pdf.

[31] TRICK, M. A. (2001) A Schedule-Then-Break Approach to Sports Timetabling. *Lecture Notes in Computer Science*, v. 2079, p. 242-252.

Anexo B – Relatório da Melhor Solução do BSSP2006-A

PROGRAMAÇÃO DE JOGOS DO CAMPEONATO BRASILEIRO DE FUTEBOL 2006 – SÉRIE A

Times envolvidos no campeonato

Times	Notações	Times	Notações	Times	Notações	Times	Notações
Atlético-PR	APR	Flamengo	FLA	Internacional	INT	Santa Cruz	STC
Botafogo	BOT	Fluminense	FLU	Juventude	JUV	Santos	SAN
Corinthians	COR	Fortaleza	FOR	Palmeiras	PAL	São Caetano	SCA
Cruzeiro	CRU	Goiás	GOI	Paraná	PAR	São Paulo	SPA
Figueirense	FIG	Grêmio	GRE	Ponte Preta	PON	Vasco	VAS

Nessa tabela, *Times* representam os times participantes da Série A do Campeonato Brasileiro de Futebol do ano de 2006 e *Notações* representam as notações utilizadas para identificar o respectivo time na tabela de programação de jogos.

Programação dos jogos da Série A do Campeonato Brasileiro de Futebol do ano de 2006

Rodada	Jogos										
1	JUV x APR	PAL x BOT	VAS x COR	CRU x FLU	FLA x FIG	STC x FOR	GOI x PON	INT x GRE	SAN x PAR	SCA x SPA	Jogos do Primeiro Turno
2	APR x SCA	BOT x VAS	COR x SAN	FOR x CRU	FIG x INT	PAR x FLA	FLU x JUV	GRE x GOI	PON x PAL	SPA x STC	
3	APR x PON	BOT x CRU	SCA x COR	JUV x FIG	FLA x SAN	FLU x GRE	SPA x FOR	INT x GOI	PAL x PAR	VAS x STC	
4	GOI x APR	STC x BOT	GRE x COR	CRU x VAS	FIG x SPA	FLA x PAL	PAR x FLU	PON x FOR	SAN x INT	JUV x SCA	
5	APR x FLU	FIG x BOT	COR x PAR	SAN x CRU	STC x FLA	FOR x VAS	GOI x JUV	GRE x SCA	PAL x INT	PON x SPA	
6	PAR x APR	BOT x PON	COR x STC	SPA x CRU	FLU x FIG	FOR x FLA	SCA x GOI	PAL x GRE	INT x JUV	VAS x SAN	
7	APR x FOR	BOT x PAR	SPA x COR	CRU x FIG	FLA x INT	FLU x SAN	STC x GOI	PON x GRE	JUV x PAL	SCA x VAS	
8	APR x BOT	COR x PON	CRU x JUV	FIG x SCA	GRE x FLA	STC x FLU	PAR x FOR	GOI x PAL	VAS x INT	SAN x SPA	
9	INT x APR	SCA x BOT	COR x FLA	GOI x CRU	PON x FIG	FLU x SPA	FOR x JUV	GRE x VAS	PAL x SAN	PAR x STC	
10	FIG x APR	BOT x COR	INT x CRU	FLA x SCA	FLU x PAL	FOR x GRE	VAS x GOI	JUV x PON	SPA x PAR	SAN x STC	
11	APR x SPA	BOT x JUV	COR x INT	CRU x SCA	FIG x PAL	FLA x VAS	PON x FLU	SAN x FOR	GOI x PAR	STC x GRE	
12	VAS x APR	SPA x BOT	FLU x COR	PAL x CRU	GOI x FIG	PON x FLA	SCA x FOR	GRE x SAN	PAR x INT	STC x JUV	
13	COR x APR	BOT x GOI	CRU x STC	PAR x FIG	JUV x FLA	FOR x FLU	GRE x SPA	INT x SCA	PAL x VAS	SAN x PON	
14	APR x GRE	INT x BOT	FOR x COR	FLA x CRU	FIG x STC	FLU x GOI	JUV x SAN	SCA x PAL	PON x PAR	VAS x SPA	
15	FLA x APR	GRE x BOT	CRU x COR	FIG x FOR	FLU x SCA	GOI x SAN	PON x INT	SPA x JUV	STC x PAL	PAR x VAS	
16	SAN x APR	BOT x FLU	COR x FIG	CRU x PON	GOI x FLA	INT x FOR	GRE x PAR	VAS x JUV	PAL x SPA	STC x SCA	
17	APR x CRU	FOR x BOT	PAL x COR	FIG x GRE	FLA x FLU	SPA x GOI	INT x STC	JUV x PAR	VAS x PON	SCA x SAN	
18	APR x STC	BOT x FLA	COR x GOI	PAR x CRU	FIG x SAN	FLU x VAS	FOR x PAL	GRE x JUV	SPA x INT	PON x SCA	
19	PAL x APR	SAN x BOT	JUV x COR	CRU x GRE	VAS x FIG	FLA x SPA	INT x FLU	GOI x FOR	SCA x PAR	STC x PON	
20	APR x JUV	BOT x PAL	COR x VAS	FLU x CRU	FIG x FLA	FOR x STC	PON x GOI	GRE x INT	PAR x SAN	SPA x SCA	Jogos do SegundoTurno
21	SCA x APR	VAS x BOT	SAN x COR	CRU x FOR	INT x FIG	FLA x PAR	JUV x FLU	GOI x GRE	PAL x PON	STC x SPA	
22	PON x APR	CRU x BOT	COR x SCA	FIG x JUV	SAN x FLA	GRE x FLU	FOR x SPA	GOI x INT	PAR x PAL	STC x VAS	
23	APR x GOI	BOT x STC	COR x GRE	VAS x CRU	SPA x FIG	PAL x FLA	FLU x PAR	FOR x PON	INT x SAN	SCA x JUV	
24	FLU x APR	BOT x FIG	PAR x COR	CRU x SAN	FLA x STC	VAS x FOR	JUV x GOI	SCA x GRE	INT x PAL	SPA x PON	
25	APR x PAR	PON x BOT	STC x COR	CRU x SPA	FIG x FLU	FLA x FOR	GOI x SCA	GRE x PAL	JUV x INT	SAN x VAS	
26	FOR x APR	PAR x BOT	COR x SPA	FIG x CRU	INT x FLA	SAN x FLU	GOI x STC	GRE x PON	PAL x JUV	VAS x SCA	
27	BOT x APR	PON x COR	JUV x CRU	SCA x FIG	FLA x GRE	FLU x STC	FOR x PAR	PAL x GOI	INT x VAS	SPA x SAN	
28	APR x INT	BOT x SCA	FLA x COR	CRU x GOI	FIG x PON	SPA x FLU	JUV x FOR	VAS x GRE	SAN x PAL	STC x PAR	
29	APR x FIG	COR x BOT	CRU x INT	SCA x FLA	PAL x FLU	GRE x FOR	GOI x VAS	PON x JUV	PAR x SPA	STC x SAN	
30	SPA x APR	JUV x BOT	INT x COR	SCA x CRU	PAL x FIG	VAS x FLA	FLU x PON	FOR x SAN	PAR x GOI	GRE x STC	
31	APR x VAS	BOT x SPA	COR x FLU	CRU x PAL	FIG x GOI	FLA x PON	FOR x SCA	SAN x GRE	INT x PAR	JUV x STC	
32	APR x COR	GOI x BOT	STC x CRU	FIG x PAR	FLA x JUV	FLU x FOR	SPA x GRE	SCA x INT	VAS x PAL	PON x SAN	
33	GRE x APR	BOT x INT	COR x FOR	CRU x FLA	STC x FIG	GOI x FLU	SAN x JUV	PAL x SCA	PAR x PON	SPA x VAS	
34	APR x FLA	BOT x GRE	COR x CRU	FOR x FIG	SCA x FLU	SAN x GOI	INT x PON	JUV x SPA	PAL x STC	VAS x PAR	
35	APR x SAN	FLU x BOT	FIG x COR	PON x CRU	FLA x GOI	FOR x INT	PAR x GRE	JUV x VAS	SPA x PAL	SCA x STC	
36	CRU x APR	BOT x FOR	COR x PAL	GRE x FIG	FLU x FLA	GOI x SPA	STC x INT	PAR x JUV	PON x VAS	SAN x SCA	
37	STC x APR	FLA x BOT	GOI x COR	CRU x PAR	SAN x FIG	VAS x FLU	PAL x FOR	JUV x GRE	INT x SPA	SCA x PON	
38	APR x PAL	BOT x SAN	COR x JUV	GRE x CRU	FIG x VAS	SPA x FLA	FLU x INT	FOR x GOI	PAR x SCA	PON x STC	

Nessa tabela, cada célula representa um jogo no formato T_i x T_j, que representa que o time T_i irá jogar, em casa, contra o time T_j na respectiva rodada. Por exemplo, na 1ª rodada, há o jogo `JUV x APR` que significa que o Juventude irá jogar, em casa, contra o Atlético-PR na primeira rodada.
Nessa tabela, as rodadas 1 a 19 são relativas aos jogos do primeiro turno e as rodadas 20 a 38 são referentes aos jogos do segundo turno.

Objetivos

O objetivo principal dessa programação de jogos é minimizar a distância total viajada por todos os times durante o campeonato. O objetivo secundário é minimizar a diferença entre a distância do time que mais viajou e o que menos viajou no campeonato, balanceando desta forma, a distância viajada por cada time.

Principais restrições contempladas

Na programação de jogos apresentada anteriormente, foram satisfeitas todas essas restrições:

h) Cada time joga somente uma vez por rodada;
i) Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos;
j) Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos, sendo um em casa e o outro na casa do adversário;
k) As duas últimas rodadas de cada turno terão a configuração inversa das duas primeiras rodadas de cada turno com relação ao mando de campo;
l) Não poderá haver jogos entre times do mesmo estado na última rodada;
m) A diferença entre os jogos feitos em cada turno em casa e fora de casa de um time não pode ser maior que uma unidade;
n) Um time não pode jogar mais que duas vezes consecutivas dentro ou fora de casa.

Considerações

Nessa programação, considera-se que cada time inicia o campeonato na sua sede, deslocando-se dela sua sede se fizer a primeira partida fora de casa e a ela retornando ao final do campeonato, caso faça a última partida fora de casa. Quando um time jogar duas partidas consecutivas fora de casa, assume-se que ele sairá da cidade do primeiro oponente diretamente para a cidade do segundo, sem retornar à sua sede.

Distâncias a serem percorridas pelos times durante o campeonato

Os dados a seguir mostram a estimativa das distâncias a serem percorridas pelos times durante o campeonato:

Times	Distâncias	Times	Distâncias	Times	Distâncias	Times	Distâncias
Atlético-PR	24.665	Flamengo	22.576	Internacional	29.769	Santa Cruz	54.359
Botafogo	23.842	Fluminense	22.772	Juventude	34.182	Santos	19.455
Corinthians	23.443	Fortaleza	57.083	Palmeiras	21.878	São Caetano	21.236
Cruzeiro	27.845	Goiás	31.370	Paraná	23.696	São Paulo	19.988
Figueirense	25.099	Grêmio	29.140	Ponte Preta	23.401	Vasco	27.087

De acordo com esses dados, pode-se verificar que os times percorrerão um total de 562.886 Km durante o Campeonato Brasileiro de 2006 (Série A). Cada time percorrerá, em média, 28.144,3 Km e a diferença entre o time que mais viajará e o que menos viajará será de 37.628 Km.
Pode-se verificar também que o Fortaleza será o time que mais viajará (57.083 Km) e o time que menos viajará será o Santos (19.455 Km).

Relação de distâncias entre as 'sedes' dos times

A definição das distâncias entre as sedes dos times foi estabelecida da seguinte forma:
Primeiramente determina-se a 'sede' de um time associando-se o aeroporto cuja distância seja relativamente próxima à sede real desse time. Em seguida, calcula-se a distância geodésica entre as 'sedes' dos times e dessa forma, obtém-se a seguinte relação de distâncias:

	APR	BOT	COR	CRU	FIG	FLA	FLU	FOR	GOI	GRE	INT	JUV	PAL	PAR	PON	STC	SAN	SCA	SPA	VAS
APR	0	779	421	1017	271	779	779	3038	1119	543	543	543	421	0	336	2766	421	421	421	779
BOT	779	0	365	372	840	0	0	2440	1026	1251	1251	1251	365	779	475	2048	365	365	365	0
COR	421	365	0	606	545	365	365	2669	939	932	932	932	0	421	113	2354	0	0	0	365
CRU	1017	372	606	0	1148	372	372	2082	802	1540	1540	1540	606	1017	680	1747	606	606	606	372
FIG	271	840	545	1148	0	840	840	3216	1373	412	412	412	545	271	509	2883	545	545	545	840
FLA	779	0	365	372	840	0	0	2440	1026	1251	1251	1251	365	779	475	2048	365	365	365	0
FLU	779	0	365	372	840	0	0	2440	1026	1251	1251	1251	365	779	475	2048	365	365	365	0
FOR	3038	2440	2669	2082	3216	2440	2440	0	2106	3583	3583	3583	2669	3038	2716	795	2669	2669	2669	2440
GOI	1119	1026	939	802	1373	1026	1026	2106	0	1634	1634	1634	939	1119	901	2077	939	939	939	1026
GRE	543	1251	932	1540	412	1251	1251	3583	1634	0	0	0	932	543	869	3285	932	932	932	1251
INT	543	1251	932	1540	412	1251	1251	3583	1634	0	0	0	932	543	869	3285	932	932	932	1251
JUV	543	1251	932	1540	412	1251	1251	3583	1634	0	0	0	932	543	869	3285	932	932	932	1251
PAL	421	365	0	606	545	365	365	2669	939	932	932	932	0	421	113	2354	0	0	0	365
PAR	0	779	421	1017	271	779	779	3038	1119	543	543	543	421	0	336	2766	421	421	421	779
PON	336	475	113	680	509	475	475	2716	901	869	869	869	113	336	0	2428	113	113	113	475
STC	2766	2048	2354	1747	2883	2048	2048	795	2077	3285	3285	3285	2354	2766	2428	0	2354	2354	2354	2048
SAN	421	365	0	606	545	365	365	2669	939	932	932	932	0	421	113	2354	0	0	0	365
SCA	421	365	0	606	545	365	365	2669	939	932	932	932	0	421	113	2354	0	0	0	365
SPA	421	365	0	606	545	365	365	2669	939	932	932	932	0	421	113	2354	0	0	0	365
VAS	779	0	365	372	840	0	0	2440	1026	1251	1251	1251	365	779	475	2048	365	365	365	0

Aeroportos e suas respectivas coordenadas geográficas

A tabela a seguir mostra os aeroportos (através do código IATA – *International Air Transport Association*, e suas coordenadas geográficas) que foram associados como 'sedes' de cada time.

Time	IATA	Latitude	Longitude
Atlético-PR	CWB	25º 52' 31" S	49º 10' 32" W
Botafogo	GIG	22º 36' 48" S	43º 15' 02" W
Corinthians	GRU	23º 08' 26" S	46º 28' 23" W
Cruzeiro	PLU	19º 07' 51" S	43º 57' 02" W
Figueirense	FLN	27º 13' 40" S	48º 33' 08" W
Flamengo	GIG	22º 36' 48" S	43º 15' 02" W
Fluminense	GIG	22º 36' 48" S	43º 15' 02" W
Fortaleza	FOR	03º 34' 46" S	38º 31' 57" W
Goiás	GYN	16º 40' 43" S	49º 15' 14" W
Grêmio	CXJ	29º 42' 11" S	51º 11' 21" W
Internacional	CXJ	29º 42' 11" S	51º 11' 21" W
Juventude	CXJ	29º 42' 11" S	51º 11' 21" W
Palmeiras	GRU	23º 08' 26" S	46º 28' 23" W
Paraná	CWB	25º 52' 31" S	49º 10' 32" W
Ponte Preta	CPQ	23º 25' 00" S	47º 08' 04" W
Santa Cruz	REC	08º 35' 07" S	34º 55' 22" W
Santos	GRU	23º 08' 26" S	46º 28' 23" W
São Caetano	GRU	23º 08' 26" S	46º 28' 23" W
São Paulo	GRU	23º 08' 26" S	46º 28' 23" W
Vasco	GIG	22º 36' 48" S	43º 15' 02" W

Anexo C – Relatório da Melhor Solução do BSSP2006-B

PROGRAMAÇÃO DE JOGOS DO CAMPEONATO BRASILEIRO DE FUTEBOL 2006 – SÉRIE B

Times envolvidos no campeonato

Times	Notações	Times	Notações	Times	Notações	Times	Notações
América	AME	Coritiba	CBA	Marília	MAR	Remo	REM
Atlético-MG	ATL	CRB	CRB	Náutico	NAU	Santo André	STA
Avaí	AVA	Gama	GAM	Paulista	PAU	São Raimundo	SRA
Brasiliense	BRA	Guarani	GUA	Paysandu	PAY	Sport	SPO
Ceará	CEA	Ituano	ITU	Portuguesa	POR	Vila Nova	VIL

Nessa tabela, *Times* representam os times participantes da Série B do Campeonato Brasileiro de Futebol do ano de 2006 e *Notações* representam as notações utilizadas para identificar o respectivo time na tabela de programação de jogos.

Programação dos jogos da Série B do Campeonato Brasileiro de Futebol do ano de 2006

Rodada	Jogos										
1	AME x PAU	ATL x BRA	GAM x AVA	CEA x CRB	CBA x GUA	ITU x STA	MAR x POR	NAU x SPO	REM x PAY	SRA x VIL	Jogos do Primeiro Turno
2	GUA x AME	CRB x ATL	AVA x ITU	BRA x CEA	PAU x CBA	PAY x GAM	SPO x MAR	VIL x NAU	POR x REM	STA x SRA	
3	AME x NAU	ATL x SRA	STA x AVA	BRA x VIL	GUA x CEA	MAR x CBA	PAY x CRB	REM x GAM	ITU x POR	SPO x PAU	
4	GAM x AME	ATL x GUA	PAU x AVA	NAU x BRA	CEA x SPO	CBA x REM	CRB x STA	SRA x ITU	VIL x MAR	POR x PAY	
5	AVA x AME	CEA x ATL	GUA x BRA	CBA x PAY	CRB x MAR	NAU x GAM	PAU x ITU	SRA x POR	REM x SPO	STA x VIL	
6	AME x ATL	AVA x CEA	BRA x REM	SPO x CBA	ITU x CRB	GAM x SRA	VIL x GUA	MAR x STA	PAY x NAU	POR x PAU	
7	AME x PAY	ATL x MAR	CRB x AVA	BRA x SRA	REM x CEA	CBA x VIL	SPO x GAM	GUA x POR	ITU x NAU	STA x PAU	
8	SRA x AME	GAM x ATL	VIL x AVA	POR x BRA	CEA x ITU	CRB x CBA	STA x GUA	NAU x MAR	PAU x REM	PAY x SPO	
9	AME x STA	ATL x PAU	AVA x BRA	GAM x CEA	NAU x CBA	SRA x CRB	PAY x GUA	SPO x ITU	MAR x REM	POR x VIL	
10	MAR x AME	SPO x ATL	AVA x POR	BRA x PAY	VIL x CEA	CBA x SRA	PAU x CRB	ITU x GAM	GUA x NAU	REM x STA	
11	ITU x AME	STA x ATL	REM x AVA	CBA x BRA	CEA x MAR	CRB x GAM	GUA x SPO	POR x NAU	PAU x SRA	VIL x PAY	
12	AME x REM	ATL x ITU	PAY x AVA	BRA x SPO	NAU x CEA	GAM x CBA	CRB x VIL	SRA x GUA	MAR x PAU	STA x POR	
13	AME x VIL	ATL x CBA	AVA x SRA	BRA x GAM	POR x CEA	SPO x CRB	GUA x PAU	ITU x MAR	NAU x REM	PAY x STA	
14	CEA x AME	REM x ATL	AVA x NAU	CRB x BRA	CBA x POR	VIL x GAM	ITU x GUA	MAR x SRA	PAU x PAY	STA x SPO	
15	AME x CBA	ATL x AVA	BRA x STA	PAY x CEA	POR x CRB	GAM x PAU	GUA x MAR	REM x ITU	SRA x NAU	SPO x VIL	
16	AME x POR	NAU x ATL	SPO x AVA	MAR x BRA	CEA x CBA	GUA x CRB	GAM x STA	ITU x PAY	VIL x PAU	SRA x REM	
17	CRB x AME	ATL x PAY	AVA x GUA	PAU x BRA	CEA x SRA	CBA x ITU	MAR x GAM	STA x NAU	POR x SPO	REM x VIL	
18	BRA x AME	POR x ATL	AVA x CBA	STA x CEA	CRB x REM	GUA x GAM	VIL x ITU	PAY x MAR	PAU x NAU	SPO x SRA	
19	AME x SPO	ATL x VIL	MAR x AVA	ITU x BRA	CEA x PAU	CBA x STA	NAU x CRB	GAM x POR	REM x GUA	SRA x PAY	
20	PAU x AME	BRA x ATL	AVA x GAM	CRB x CEA	GUA x CBA	STA x ITU	POR x MAR	SPO x NAU	PAY x REM	VIL x SRA	Jogos do SegundoTurno
21	AME x GUA	ATL x CRB	ITU x AVA	CEA x BRA	CBA x PAU	GAM x PAY	MAR x SPO	NAU x VIL	REM x POR	SRA x STA	
22	NAU x AME	SRA x ATL	AVA x STA	VIL x BRA	CEA x GUA	CBA x MAR	CRB x PAY	GAM x REM	POR x ITU	PAU x SPO	
23	AME x GAM	GUA x ATL	AVA x PAU	BRA x NAU	SPO x CEA	REM x CBA	STA x CRB	ITU x SRA	MAR x VIL	PAY x POR	
24	AME x AVA	ATL x CEA	BRA x GUA	PAY x CBA	MAR x CRB	GAM x NAU	ITU x PAU	POR x SRA	SPO x REM	VIL x STA	
25	ATL x AME	CEA x AVA	REM x BRA	CBA x SPO	CRB x ITU	SRA x GAM	GUA x VIL	STA x MAR	NAU x PAY	PAU x POR	
26	PAY x AME	MAR x ATL	AVA x CRB	SRA x BRA	CEA x REM	VIL x CBA	GAM x SPO	POR x GUA	NAU x ITU	PAU x STA	
27	AME x SRA	ATL x GAM	AVA x VIL	BRA x POR	ITU x CEA	CBA x CRB	GUA x STA	MAR x NAU	REM x PAU	SPO x PAY	
28	STA x AME	PAU x ATL	BRA x AVA	CEA x GAM	CBA x NAU	CRB x SRA	GUA x PAY	ITU x SPO	REM x MAR	VIL x POR	
29	AME x MAR	ATL x SPO	POR x AVA	PAY x BRA	CEA x VIL	SRA x CBA	CRB x PAU	GAM x ITU	NAU x GUA	STA x REM	
30	AME x ITU	ATL x STA	AVA x REM	BRA x CBA	MAR x CEA	GAM x CRB	SPO x GUA	NAU x POR	SRA x PAU	PAY x VIL	
31	REM x AME	ITU x ATL	AVA x PAY	SPO x BRA	CEA x NAU	CBA x GAM	VIL x CRB	GUA x SRA	PAU x MAR	POR x STA	
32	VIL x AME	CBA x ATL	SRA x AVA	GAM x BRA	CEA x POR	CRB x SPO	PAU x GUA	MAR x ITU	REM x NAU	STA x PAY	
33	AME x CEA	ATL x REM	NAU x AVA	BRA x CRB	POR x CBA	GAM x VIL	GUA x ITU	SRA x MAR	PAY x PAU	SPO x STA	
34	CBA x AME	AVA x ATL	STA x BRA	CEA x PAY	CRB x POR	PAU x GAM	MAR x GUA	ITU x REM	NAU x SRA	VIL x SPO	
35	POR x AME	ATL x NAU	AVA x SPO	BRA x MAR	CBA x CEA	CRB x GUA	STA x GAM	PAY x ITU	PAU x VIL	REM x SRA	
36	AME x CRB	PAY x ATL	GUA x AVA	BRA x PAU	SRA x CEA	ITU x CBA	GAM x MAR	NAU x STA	SPO x POR	VIL x REM	
37	AME x BRA	ATL x POR	CBA x AVA	CEA x STA	REM x CRB	GAM x GUA	ITU x VIL	MAR x PAY	NAU x PAU	SRA x SPO	
38	SPO x AME	VIL x ATL	AVA x MAR	BRA x ITU	PAU x CEA	STA x CBA	CRB x NAU	POR x GAM	GUA x REM	PAY x SRA	

Nessa tabela, cada célula representa um jogo no formato T_i x T_j, que representa que o time T_i irá jogar, em casa, contra o time T_j na respectiva rodada. Por exemplo, na 1ª rodada, há o jogo `AME x PAU` que significa que o América irá jogar, em casa, contra o Paulista na primeira rodada.
Nessa tabela, as rodadas 1 a 19 são relativas aos jogos do primeiro turno e as rodadas 20 a 38 são referentes aos jogos do segundo turno.

Objetivos

O objetivo principal dessa programação de jogos é minimizar a distância total viajada por todos os times durante o campeonato. O objetivo secundário é minimizar a diferença entre a distância do time que mais viajou e o que menos viajou no campeonato, balanceando desta forma, a distância viajada por cada time.

Principais restrições contempladas

Na programação de jogos apresentada anteriormente, foram satisfeitas todas essas restrições:

o) Cada time joga somente uma vez por rodada;
p) Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos;
q) Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos, sendo um em casa e o outro na casa do adversário;
r) As duas últimas rodadas de cada turno terão a configuração inversa das duas primeiras rodadas de cada turno com relação ao mando de campo;
s) Não poderá haver jogos entre times do mesmo estado na última rodada;
t) A diferença entre os jogos feitos em cada turno em casa e fora de casa de um time não pode ser maior que uma unidade;
u) Um time não pode jogar mais que duas vezes consecutivas dentro ou fora de casa.

Considerações

Nessa programação, considera-se que cada time inicia o campeonato na sua sede, deslocando-se dela sua sede se fizer a primeira partida fora de casa e a ela retornando ao final do campeonato, caso faça a última partida fora de casa. Quando um time jogar duas partidas consecutivas fora de casa, assume-se que ele sairá da cidade do primeiro oponente diretamente para a cidade do segundo, sem retornar à sua sede.

Distâncias a serem percorridas pelos times durante o campeonato

Os dados a seguir mostram a estimativa das distâncias a serem percorridas pelos times durante o campeonato:

Times	Distâncias	Times	Distâncias	Times	Distâncias	Times	Distâncias
América	58.020	Coritiba	39.801	Marília	42.493	Remo	62.633
Atlético-MG	43.981	CRB	47.691	Náutico	47.781	Santo André	40.449
Avaí	44.946	Gama	40.568	Paulista	45.354	São Raimundo	63.649
Brasiliense	42.098	Guarani	41.707	Paysandu	58.452	Sport	51.810
Ceará	63.216	Ituano	40.071	Portuguesa	47.696	Vila Nova	44.958

De acordo com esses dados, pode-se verificar que os times percorrerão um total de 967.374 Km durante o Campeonato Brasileiro de 2006 (Série B). Cada time percorrerá, em média, 48368,7 Km e a diferença entre o time que mais viajará e o que menos viajará será de 23.848 Km.
Pode-se verificar também que o São Raimundo será o time que mais viajará (63.649 Km) e o time que menos viajará será o Coritiba (39.801 Km).

Relação de distâncias entre as 'sedes' dos times

A definição das distâncias entre as sedes dos times foi estabelecida da seguinte forma:
Primeiramente determina-se a 'sede' de um time associando-se o aeroporto cuja distância seja relativamente próxima à sede real desse time. Em seguida, calcula-se a distância geodésica entre as 'sedes' dos times e dessa forma, obtém-se a seguinte relação de distâncias:

	AME	ATL	AVA	BRA	CEA	CBA	CRB	GAM	GUA	ITU	MAR	NAU	PAU	PAY	POR	REM	STA	SRA	SPO	VIL
AME	0	1994	3142	1924	446	2998	496	1924	2664	2601	2601	392	2601	1690	2601	1690	2601	3114	392	2190
ATL	1994	0	1148	687	2082	1017	1575	687	680	606	606	1747	606	2322	606	2322	606	2871	1747	802
AVA	3142	1148	0	1497	3216	271	2712	1497	509	545	545	2883	545	3236	545	3236	545	3312	2883	1373
BRA	1924	687	1497	0	1857	1265	1639	0	995	1002	1002	1807	1002	1746	1002	1746	1002	2185	1807	270
CEA	446	2082	3216	1857	0	3038	833	1857	2716	2669	2669	795	2669	1267	2669	1267	2669	2734	795	2106
CBA	2998	1017	271	1265	3038	0	2592	1265	336	421	421	2766	421	2989	421	2989	421	3038	2766	1119
CRB	496	1575	2712	1639	833	2592	0	1639	2255	2181	2181	174	2181	1893	2181	1893	2181	3182	174	1909
GAM	1924	687	1497	0	1857	1265	1639	0	995	1002	1002	1807	1002	1746	1002	1746	1002	2185	1807	270
GUA	2664	680	509	995	2716	336	2255	995	0	113	113	2428	113	2741	113	2741	113	2941	2428	901
ITU	2601	606	545	1002	2669	421	2181	1002	113	0	0	2354	0	2748	0	2748	0	3007	2354	939
MAR	2601	606	545	1002	2669	421	2181	1002	113	0	0	2354	0	2748	0	2748	0	3007	2354	939
NAU	392	1747	2883	1807	795	2766	174	1807	2428	2354	2354	0	2354	1942	2354	1942	2354	3279	0	2077
PAU	2601	606	545	1002	2669	421	2181	1002	113	0	0	2354	0	2748	0	2748	0	3007	2354	939
PAY	1690	2322	3236	1746	1267	2989	1893	1746	2741	2748	2748	1942	2748	0	2748	0	2748	1512	1942	1870
POR	2601	606	545	1002	2669	421	2181	1002	113	0	0	2354	0	2748	0	2748	0	3007	2354	939
REM	1690	2322	3236	1746	1267	2989	1893	1746	2741	2748	2748	1942	2748	0	2748	0	2748	1512	1942	1870
STA	2601	606	545	1002	2669	421	2181	1002	113	0	0	2354	0	2748	0	2748	0	3007	2354	939
SRA	3114	2871	3312	2185	2734	3038	3182	2185	2941	3007	3007	3279	3007	1512	3007	1512	3007	0	3279	2106
SPO	392	1747	2883	1807	795	2766	174	1807	2428	2354	2354	0	2354	1942	2354	1942	2354	3279	0	2077
VIL	2190	802	1373	270	2106	1119	1909	270	901	939	939	2077	939	1870	939	1870	939	2106	2077	0

Aeroportos e suas respectivas coordenadas geográficas

A tabela a seguir mostra os aeroportos (através do código IATA – *International Air Transport Association*, e suas coordenadas geográficas) que foram associados como 'sedes' de cada time.

Time	IATA	Latitude	Longitude
América	NAT	05º 30' 54" S	35º 14' 57" W
Atlético-MG	PLU	19º 07' 51" S	43º 57' 02" W
Avaí	FLN	27º 13' 40" S	48º 33' 08" W
Brasiliense	BSB	15º 46' 51" S	47º 54' 46" W
Ceará	FOR	03º 34' 46" S	38º 31' 57" W
Coritiba	CWB	25º 52' 31" S	49º 10' 32" W
CRB	MCZ	09º 31' 00" S	35º 47' 00" W
Gama	BSB	15º 46' 51" S	47º 54' 46" W
Guarani	CPQ	23º 25' 00" S	47º 08' 04" W
Ituano	GRU	23º 08' 26" S	46º 28' 23" W
Marília	GRU	23º 08' 26" S	46º 28' 23" W
Náutico	REC	08º 35' 07" S	34º 55' 22" W
Paulista	GRU	23º 08' 26" S	46º 28' 23" W
Paysandu	BEL	01º 05' 23" S	48º 28' 44" W
Portuguesa	GRU	23º 08' 26" S	46º 28' 23" W
Remo	BEL	01º 05' 23" S	48º 28' 44" W
Santo André	GRU	23º 08' 26" S	46º 28' 23" W
São Raimundo	MAO	03º 28' 02" S	60º 03' 02" W
Sport	REC	08º 35' 07" S	34º 55' 22" W
Vila Nova	GYN	16º 40' 43" S	49º 15' 14" W