

**Universidade Federal de Ouro Preto
Departamento de Ciências Exatas e Aplicadas
Curso Sistemas de Informação**



**Abordagens Heurísticas para o
Problema de Agendamento de
Jogos de Competições Esportivas**

Gustavo Benício Gonçalves

**TRABALHO DE
CONCLUSÃO DE CURSO**

ORIENTAÇÃO:

PROF. MSc. George Henrique Godim da Fonseca

**Outubro, 2014
João Monlevade/MG**

Gustavo Benício Gonçalves

**Abordagens Heurísticas para o Problema de
Agendamento de Jogos de Competições
Esportivas**

Orientador: Prof. MSc. George Henrique Godim da Fonseca

Monografia apresentada ao Curso de
Sistemas de Informação do Departamento de
Ciências Exatas e Aplicadas, como requisito
parcial para aprovação na Disciplina
Trabalho de Conclusão de Curso II.

Universidade Federal de Ouro Preto
João Monlevade
Outubro de 2014



Curso Sistemas de Informação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Abordagens Heurísticas para o Problema no Agendamento de Jogos de Competições Esportivas

Gustavo Benício Gonçalves

Monografia apresentada ao Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CEA499 – Trabalho de Conclusão de Curso II, do curso de Bacharelado em Sistemas de Informação, e aprovada pela Banca Examinadora abaixo assinada:

Prof. Me. George Henrique Godim da Fonseca
Mestre em Ciência da Computação - UFOP
Orientador

Departamento de Computação e Sistemas - UFOP

Profª. Me. Janniele Aparecida Soares
Mestre em Ciência da Computação - UFOP
Examinador

Departamento de Computação e Sistemas - UFOP

Prof. Me. Arthur de Assis Silva
Mestre em Ciência da Computação - UFOP
Examinador

Departamento de Computação e Sistemas - UFOP

Prof. Matheus Guedes Vilas Boas
Bacharel em Sistemas de Informação - UFOP
Examinador
Departamento de Computação e Sistemas - UFOP

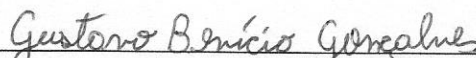
João Monlevade, 04 de outubro de 2014

Curso Sistemas de Informação

TERMO DE RESPONSABILIDADE

O texto do trabalho de conclusão de curso intitulado Abordagens Heurísticas para o Problema de Agendamento de Jogos de Competições Esportivas é de minha inteira responsabilidade. Declaro que não há utilização indevida de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos referidos autores.

João Monlevade, 04 de Outubro de 2014



Gustavo Benício Gonçalves

Resumo

O presente trabalho aborda o Problema de Agendamento de jogos de Competições Esportivas, mais conhecido como *Traveling Tournament Problem* (TTP), que tem como objetivo minimizar as distâncias percorridas pelos times no decorrer de uma competição. Encontrar calendário ideal para uma competição esportiva é economicamente muito importante, uma vez que essas competições geram gastos e receitas para os seus envolvidos. Além da necessidade de obter lucro, os times querem ser campeões, e para isso há a necessidade de que seus jogadores fiquem menos cansados com as viagens. O TTP é um problema de otimização que pertence à classe dos problemas NP-difíceis, fazendo com que sua resolução através de métodos de modelagem matemática seja difícil ou até inviável. Este trabalho apresenta o uso de técnicas heurísticas aplicadas ao modelo específico da *Major League Baseball* (MLB), a principal liga de beisebol americana. O extenso tamanho dos Estados Unidos da América e a distribuição geográfica dos times torna a elaboração do calendário de jogos uma tarefa fundamental e difícil. O objetivo principal deste problema é reduzir a distância total percorrida pelos times, desde a saída de sua sede no início do campeonato até o seu retorno após seu último jogo. Esse tipo de problema é conhecido como *Double Round Robin*, onde todos os times jogarão entre si por duas vezes na competição, tendo seus mandos de campo invertidos. Para resolver este problema, foram implementadas três técnicas diferentes de heurísticas. Foi proposto inicialmente a técnica *Late Acceptance Hill Climbing* (LAHC), uma heurística recentemente proposta que vem se mostrando promissora em problemas de agendamento. Foi utilizado também o algoritmo *Iterated Local Search* (ILS), uma meta-heurística muito simples e de fácil entendimento, mas que tem mostrado resultados muito bons para diferentes classes de problemas de otimização. Outra meta-heurística abordada é a *Simulated Annealing* (SA), muito recorrente na solução destes tipos de problemas. Para encerrar, foram realizados experimentos computacionais das meta-heurísticas e apresentados o melhor resultado, a média e o desvio padrão de cada um deles. Na comparação entre os algoritmos o LAHC foi o que obteve melhores resultados.

Agradecimentos

Primeiramente quero agradecer a Deus, por guiar meus passos e me dar forças durante toda minha vida. Ao meu professor e orientador MSc. George Henrique Godim Fonseca, pela confiança investida a mim e pela orientação que cujos conselhos foram muito importantes para a realização deste trabalho. A todos meus amigos que me proporcionaram momentos de alegria e descontração, mas também estiveram ali quando mais precisava. À minha família, por todo apoio e incentivo. Em especial a minha mãe Maria de Fátima e meu padrinho Carlos Wagner, pois sem eles eu não teria chegado até aqui e a quem dedico este trabalho. E a todos que de alguma forma contribuíram neste trabalho.

Sumário

| | | |
|----------|--|-----------|
| | Sumário | 7 |
| | Lista de ilustrações | 9 |
| | Lista de tabelas | 10 |
| 1 | INTRODUÇÃO | 12 |
| 1.1 | Objetivos | 12 |
| 1.2 | Revisão Bibliográfica | 12 |
| 1.3 | Organização do trabalho | 14 |
| 2 | O PROBLEMA DO AGENDAMENTO DE COMPETIÇÕES ES- PORTIVAS | 16 |
| 2.1 | Descrição do problema | 16 |
| 2.2 | Restrições | 16 |
| 2.3 | Metodologia de Representação | 17 |
| 2.4 | Função Objetivo | 18 |
| 3 | TÉCNICAS HEURÍSTICAS ABORDADAS | 20 |
| 3.1 | Métodos construtivos | 20 |
| 3.1.1 | Método do Polígono | 20 |
| 3.1.2 | Backtracking | 23 |
| 3.2 | Estruturas de Vizinhaça | 24 |
| 3.2.1 | Swap Teams | 25 |
| 3.2.2 | Swap Rounds | 26 |
| 3.2.3 | Swap Homes | 26 |
| 3.2.4 | Swap Homes All | 27 |
| 3.2.5 | Partial Swap Rounds | 27 |
| 3.2.6 | Partial Swap Teams | 29 |
| 3.3 | Metaheurísticas | 29 |
| 3.3.1 | Iterated Local Search | 29 |
| 3.3.2 | Late Acceptance Hill Climbing | 31 |
| 3.3.3 | Simulated Annealing | 32 |
| 4 | EXPERIMENTOS COMPUTACIONAIS | 34 |
| 4.1 | Ambiente de Testes | 34 |
| 4.2 | Caracterização da Instâncias | 34 |

| | | |
|-----|---------------------------------------|----|
| 4.3 | Configuração dos parâmetros | 34 |
| 4.4 | Resultados Computacionais | 36 |
| 5 | CONSIDERAÇÕES FINAIS | 40 |
| | Referências | 41 |

Lista de ilustrações

| | |
|--|----|
| Figura 1 – Representação do método com $n = 6$, na primeira fase. | 21 |
|--|----|

Lista de tabelas

| | |
|---|----|
| Tabela 2.1 – Estrutura de representação de Anagnostopoulos et al. (2006) | 17 |
| Tabela 3.1 – Representação resultante da associação de jogos. | 21 |
| Tabela 3.2 – Matriz de oponentes consecutivos da Tabela 3.1 | 22 |
| Tabela 3.3 – Matriz de distâncias entre os times | 22 |
| Tabela 3.4 – Times Reais | 23 |
| Tabela 3.5 – Times Abstratos | 23 |
| Tabela 3.6 – Resultado final do Método do Polígono | 23 |
| Tabela 3.7 – Exemplo de solução Anagnostopoulos et al. (2006) | 24 |
| Tabela 3.8 – Movimento $Swap\ Teams(T_2, T_5)$ | 25 |
| Tabela 3.9 – Movimento $Swap\ Teams(T_2, T_5)$ | 25 |
| Tabela 3.10–Resultado do movimento $Swap\ Teams(T_2, T_5)$ | 25 |
| Tabela 3.11–Colunas escolhidas para o movimento $Swap\ Rounds(R_3, R_{10})$ | 26 |
| Tabela 3.12–Resultado do movimento $Swap\ Rounds(R_3, R_{10})$ | 26 |
| Tabela 3.13–Movimento $Swap\ Homes(T_1, T_4)$ | 27 |
| Tabela 3.14–Resultado do movimento $Swap\ Homes(T_1, T_4)$ | 27 |
| Tabela 3.15–Movimento $Swap\ Homes\ All$ | 27 |
| Tabela 3.16–Representação dos elementos para $Partial\ Swap\ Rounds\ (R_2, R_9, T_2)$. . | 28 |
| Tabela 3.17–Movimento $Partial\ Swap\ Rounds\ (R_2, R_9, T_2)$ | 28 |
| Tabela 3.18–Resultado Final do Movimento $Partial\ Swap\ Rounds\ (R_2, R_9, T_2)$. . . | 28 |
| Tabela 3.19–Movimento $Partial\ Swap\ Teams\ (T_2, T_4, R_9)$ | 29 |
| Tabela 3.20–Resultado Final do Movimento $Partial\ Swap\ Teams\ (T_2, T_4, R_9)$ | 29 |
| Tabela 4.1 – Parâmetros para o SA | 35 |
| Tabela 4.2 – Parâmetros para o LAHC | 35 |
| Tabela 4.3 – Parâmetros para o ILS | 36 |
| Tabela 4.4 – Resultados obtidos pelo algoritmo <i>Simulated Annealing</i> | 36 |
| Tabela 4.5 – Resultados obtidos pelo algoritmo <i>Late Acceptance Hill Climbing</i> . . . | 37 |
| Tabela 4.6 – Resultados obtidos pelo algoritmo <i>Iterated Local Search</i> | 37 |
| Tabela 4.7 – Comparativo entre as Melhores Soluções | 37 |
| Tabela 4.8 – Comparativo da Média e Desvio Padrão dos resultados | 38 |
| Tabela 4.9 – Comparativo entre melhores soluções conhecidas $\times LAHC$ | 38 |

Lista de Algoritmos

| | | |
|-----|---|----|
| 3.1 | ILS | 30 |
| 3.2 | Late Acceptance Hill Climbing | 31 |
| 3.3 | Simulated Annealing | 33 |

1 Introdução

Neste trabalho é utilizado o modelo de instâncias específico da *Major League Baseball*(MLB), a principal liga de beisebol americano. Será utilizado o modelo de calendário *Double Round Robin* para o torneio, a fim de encontrar o agendamento com a menor distância total viajada.

Devido ao extenso tamanho dos Estados Unidos da América e a distribuição geográfica dos times, encontrar o calendário ideal para a competição é uma tarefa fundamental, uma vez que essas competições geram gastos e receitas para os seus envolvidos. Além da necessidade de obter lucro, os times querem ser campeões, e para isso há a necessidade de que seus jogadores fiquem menos cansados com as viagens.

1.1 Objetivos

Este trabalho tem como objetivo geral abordar diferentes meta-heurísticas aplicadas ao problema de agendamento em competições esportivas para resolver o problema da *Major League Baseball*. Bem como, descrever métodos de solução inicial e vizinhanças utilizados. Como objetivo específico, deseja-se comparar os resultados das meta-heurísticas, onde será utilizada uma configuração padrão de estrutura de vizinhanças e tempo de execução para obter maior precisão nas comparações. Easton, Nemhauser e Trick (KIM, 2012, apud ,p. 5) afirmam que o objetivo deste TTP é encontrar um calendário do torneio *double-robin* para minimizar a distância total percorrida pelas equipes, satisfazendo, ao mesmo tempo, as restrições específicas deste torneio.

1.2 Revisão Bibliográfica

Silva G.; Mine (2005), propuseram em seu trabalho (Um Método de Geração de uma Solução Inicial baseado em Backtracking para o Problema de Programação de Jogos do Campeonato Brasileiro de Futebol) uma metodologia para gerar uma solução inicial aleatória, a qual é baseada em *backtracking* para o TTP satisfazendo as restrições do problema. Eles utilizaram como instância, a primeira divisão do Campeonato Brasileiro de Futebol, que é realizado em dois turnos completos e espelhados.

Anagnostopoulos et al. (2006) utilizaram em seu trabalho (A simulated annealing approach to the traveling tournament problem), o uso da meta-heurística Simulated Annealing, proposto por Kirkpatrick e Vecchi. (1983), utilizando-se de estratégias de oscilação e reaquecimento para resolver o problema da alocação de jogos na MLB *Major League Baseball*. Neste trabalho foram utilizadas cinco estruturas de vizinhança: *Swap Homes*, *Swap Rounds*, *Swap Teams*, *Partial Swap Homes* e *Partial Swap Teams*. O

trabalho obteve ótimos resultados se aproximando aos melhores da competição internacional Benchmark (2014), para o problema da Major League Baseball (MLB).

Hentenryck e Vergados (2006), em seu trabalho (Traveling tournament scheduling: A systematic evaluation of simulated annealing), apresentam a meta-heurística Simulated Annealing para a resolução do Traveling Tournament Problem. Seus resultados mostram que, com as melhorias adequadas, o algoritmo SA pode ser tornar poderoso para a solução de problemas desta natureza. O SA conseguiu igualar, e até melhorar, a maioria das melhores soluções conhecidas na competição internacional Benchmark (2014), para o problema da Major League Baseball (MLB).

Fonseca et al. (2014), em seu trabalho (Goal solver: a hybrid local search based solver for high school timetabling), apresentam uma abordagem de busca local para o High School Timetabling Problem. Empregaram como abordagem a Kingston High School Timetabling Engine (KHE), uma plataforma para gerir de forma eficiente as instâncias e soluções. O KHE possui como principal característica o recálculo de custo incremental, que acelera o cálculo das variações de custo ao modificar soluções usando os métodos de busca local.

Para a implementação, eles desenvolveram várias estruturas de vizinhança que foram utilizadas em uma meta-heurística híbrida baseada em Simulated Annealing e Iterated Local Search. A abordagem utilizada foi capaz de encontrar soluções viáveis para quase todos os casos para o problema e ganhou o terceiro International Timetabling Competition.

Mine, Souza e Silva (2006), em seu trabalho (Programação de jogos de competições esportivas: Uma abordagem heurística – Parte II), propuseram o uso de um método híbrido, ILS-MRD, composto da meta-heurística ILS, proposto por Lourenco, Martin e Stutzle (2001), e o *Método de Descida Randômico* para solucionar o problema. Propuseram também, a montagem da solução inicial pelo método do polígono, onde a ideia principal é gerar uma boa solução inicial de maneira determinística. Para a execução dos testes, foram utilizadas instâncias da primeira divisão do Campeonato Brasileiro dos anos 2004 e 2005. Os resultados obtidos neste trabalho superaram significativamente as soluções produzidas pela Confederação Brasileira de Futebol.

Kim (2012), em seu trabalho (Iterated Local Search for the Traveling Tournament Problem), propôs o uso da meta-heurística Iterated Local Search (ILS). Inicialmente ele desenvolveu um ILS básico para analisar a sua aplicabilidade para o TTP. Para a execução dos experimentos computacionais foram utilizadas as instâncias da MLB *Major League Baseball*, que estão disponíveis na competição internacional do problema, em Benchmark (2014). O ILS igualou as instâncias menores, com 4, 6 e 8 times, em apenas alguns segundos. Para as instâncias de 10, 12 e 14 times, obtiveram resultados melhores do que a maioria das outras abordagens da mesma competição.

Urrutia, Ribeiro e Melo (2007), em seu trabalho (A new lower bound to the traveling tournament problem), propuseram um novo método para determinar um *lower bound* melhor do que as já conhecidas para o Traveling Tournament Problem. Os resultados

ilustram que eles conseguiram introduzir um novo *lower Bound* para o TTP, melhorando os resultados para muitos casos da referência da competição internacional para o problema Benchmark (2014).

Uthus, Riddle e Guesgen (2009), em seu trabalho (DFS* and the traveling tournament problem), apresentaram um método de solução exata para resolver o Traveling Tournament Problem para a competição internacional para o problema Benchmark (2014), e conseguiram obter excelentes resultados para as instâncias da MLB de tamanhos de 10, 12 e 14 times.

Langford (2010), em seu trabalho (An improved neighbourhood for the traveling tournament problem), propõe uma melhor estrutura de busca em vizinhança para o TTP, que foi testado utilizando o algoritmo Simulated Annealing . A vizinhança engloba esquemas que geram soluções viáveis e inviáveis e pode ser gerado de forma eficiente.

Soluções encontradas usando esta vizinhança, estão entre as melhores conhecidas, na competição internacional Benchmark (2014), para o problema da Major League Baseball (MLB). O resultado para a instância de 10 equipes obteve o melhor de todos resultados, posteriormente foi comprovada como a ideal.

Burke e Bykov (2008), em seu trabalho (The Late Acceptance Hill-Climbing Heuristic), apresentam uma nova metodologia busca meta-heurística, o *Late Acceptance Hill Climbing* (LAHC). Essa meta-heurística foi criada com três objetivos: ser um procedimento de busca local que não exige um modelo de resfriamento artificial; usar de forma eficiente as informações coletadas durante iterações anteriores da busca; e empregar um sistema simples de aceitação. Apresentaram também, que a abordagem LAHC é bastante simples, fácil de implementar e ainda é um processo de pesquisa eficaz.

Diversos trabalhos já aplicaram com sucesso a meta-heurística LAHC, como por exemplo, Fonseca, Santos e M. (2013), aplicou o algoritmo e algumas variações para resolver o High School Timetabling Problem. Ele apresenta também que a combinação do Simulated Annealing e LAHC é uma alternativa muito promissora. Seus resultados apontam que o LAHC é um método bastante confiável e pode competir amplamente com outros algoritmos de busca local. (TIERNEY, 2013) aplicou o algoritmo ao Liner Shipping Fleet Repositioning Problem (LSFRP) e Goerler, Schulte e Voß (2013) solucionou o Traveling Purchaser Problem (TPP), que é uma variação do problema do caixeiro viajante, empregando o algoritmo LAHC. Até o presente momento nenhum trabalho abordou a aplicação da meta-heurística LAHC ao Problema de Agendamento de Jogos em Competições Esportivas.

1.3 Organização do trabalho

O presente trabalho está organizado em 5 capítulos incluindo esta introdução. O Capítulo 2 apresenta uma fundamentação teórica onde é apresentado detalhadamente

o Problema do Agendamento de Competições Esportivas incluindo a sua descrição em detalhes, sua estrutura de representação, as restrições envolvidas e a função objetivo. No Capítulo 3 são apresentadas as metodologias abordadas para resolver o problema, são eles, os algoritmos utilizados para a geração de uma solução inicial, *Método do Polígono* e *Backtracking*, as vizinhanças adotadas para a resolução do problema principal deste trabalho, e como é o funcionamento de cada uma das meta-heurísticas. No Capítulo 4 são apresentadas as instâncias dos testes que foram utilizados, os resultados obtidos com a implementação dos algoritmos que foram descritos no Capítulo 3 e a comparação entre os resultados. No Capítulo 5 são feitas as discussões sobre os resultados, considerações finais e conclusões gerais.

2 O Problema do Agendamento de Competições Esportivas

Na literatura, o problema do agendamento de competições esportivas é conhecido como *Travelling Tournament Problem* (TTP), e tem como objetivo construir uma tabela dos confrontos entre as equipes de uma competição. Muitas competições envolvem equipes que estão dispersas geograficamente, e os seus jogos precisam ter um local, um mandante e um visitante. Desta forma, o visitante necessita se deslocar de sua casa até o campo do seu oponente para o confronto, onde este deslocamento gera gastos.

Estudos de problemas desta natureza vêm se tornando fundamentais, devido à grande importância que as finanças têm para os envolvidos. Outro ponto muito importante é, como o desgaste dos jogadores com as viagens influenciam no desempenho das equipes durante o campeonato, podendo ao seu final, ser um fator importante e até decisivo de qual equipe será a campeã. A tarefa de montar uma agenda de jogos é considerado um tipo de problema dos mais desafiadores.

2.1 Descrição do problema

Existem atualmente algumas diferentes formas de abordagens para o TTP, com restrições e objetivos específicos para cada tipo de torneio. O presente trabalho tem como objetivo procurar, desenvolver e apresentar heurísticas eficientes para resolver os problemas envolvidos no agendamento dos jogos de competições esportivas. O principal foco será reduzir a distância percorrida pelas equipes envolvidas na *Major League Baseball* (MLB), dos Estados Unidos.

2.2 Restrições

O problema do agendamento para a *Major League Baseball*, que foi originalmente introduzido por Easton, Nemhauser e Trick (KIM, 2012, apud ,p. 5), tem como principal objetivo, minimizar a distância percorrida pelas equipes envolvidas na competição, assumindo que cada equipe inicia a competição em sua cidade natal e ao final retornará a ela, obedecendo as seguintes restrições:

1. Não poderá haver mais do que três jogos em casa ou três jogos fora consecutivos para qualquer equipe.

2. A competição é *Double Round Robin*, ou seja, rodadas duplas, onde um determinado time A jogará com um time B em sua casa, posteriormente jogarão novamente, só que com os mandos de campo invertidos.
3. Para N equipes serão necessárias $2N - 2$ células.
4. Não há jogos consecutivos envolvendo as mesmas equipes, por exemplo, se o time A jogou com time B em uma rodada X , eles não poderão se confrontar na rodada $X + 1$.
5. Todas as equipes deverão jogar em todas as rodadas e participarão de apenas um jogo em cada rodada.

2.3 Metodologia de Representação

A representação das distâncias entre as sedes de n equipes utiliza uma matriz d simétrica de tamanho $n \times n$, em que d_{ij} representa a distância entre duas equipes i e j .

Para representar uma solução do problema, este trabalho adotou o modelo de representação utilizado por Anagnostopoulos et al. (2006), e que foi utilizado também como um dos padrões do validador de soluções em Benchmark (2014), o site de competições de TTP, onde envolvem diversas abordagens dentre elas a utilizada neste trabalho.

A solução é representada por uma matriz de dimensões $n \times nr$, onde n é a quantidade de times e nr a quantidade de rodadas. Cada linha representa os adversários de um time t_i , cada coluna representa uma rodada r_j . E o sinal frente a cada um de seus adversários é o seu mando de campo, o valor negativo quer dizer que o time t_i jogará fora, na sede de seu adversário, caso o valor seja positivo o time t_i jogará em casa contra seu adversário. Pode-se observar que se um determinado time joga em casa, seu oponente da mesma rodada está jogando fora contra ele. Por Exemplo, na tabela 2.1 pode-se observar na primeira rodada que o Time 1, possui o valor $+6$ e o Time 6 da mesma rodada o valor -1 .

Tabela 2.1 – Estrutura de representação de Anagnostopoulos et al. (2006)

| 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 | - 5 | + 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 |

Pode-se observar nesta representação toda a trajetória percorrida por um time durante toda a competição. Por exemplo, o time 3 que inicia a competição em sua sede e logo na

primeira rodada se desloca para a sede do time 4. Em seguida, ele retorna à sua sede para na segunda rodada jogar contra o time 5. Depois, na terceira rodada, ele permanece em sua sede para o confronto contra a equipe 2.

Na quarta rodada, ele desloca-se para a sede do time 1 e assim sucessivamente até a décima e última rodada, onde encerrará o campeonato em sua sede. Isso não ocorre com o time 4, pois seu último confronto será na sede do time 3, onde após a décima rodada ele deverá retornar a sua sede. A seguir será apresentado a trajetória entre sedes, percorridas pelos times 3 e 4:

Trajetoória do time 3: $3 \rightarrow 4 \Rightarrow 3 \Rightarrow 3 \Rightarrow 1 \Rightarrow 3 \Rightarrow 2 \Rightarrow 3 \Rightarrow 6 \Rightarrow 5 \Rightarrow 3 \rightarrow 3$

Trajetoória do time 4: $4 \rightarrow 4 \Rightarrow 4 \Rightarrow 1 \Rightarrow 5 \Rightarrow 2 \Rightarrow 4 \Rightarrow 4 \Rightarrow 4 \Rightarrow 6 \Rightarrow 3 \rightarrow 4$

A notação $T_i \rightarrow T_j$ representa a viagem do time antes do primeiro turno do campeonato, quando ele estará indo para a sede de seu primeiro adversário ou após o campeonato se encerrar, quando estará retornando a sua sede; e a notação $T_i \Rightarrow T_j$ representa a viagem entre as sedes durante o campeonato. Valor em T_i representa a sede que o time está e T_j para qual ele vai. Quando os valores de T_i e T_j forem iguais quer dizer que o time não se deslocou.

2.4 Função Objetivo

O principal objetivo da competição *MLB* é minimizar a distância percorrida pelas equipes envolvidas obedecendo todas restrições impostas. Então, partindo da ideia que todas restrições estão sendo respeitadas e assumindo que cada equipe inicia a competição em sua cidade natal e ao final seu final retornará a ela. A função objetivo utilizada neste trabalho é expressa na seguinte forma:

$$F(S) = \sum_{i=1}^t custo(i)$$

F(S): Função de Avaliação

t: Total de times envolvidos na competição

custo(i): Custo de deslocamento do time $i \in T$ durante toda a competição.

O custo de um time $i \in T$ é definido pela soma das distâncias percorridas em todas as viagens realizadas por i , como apresentado anteriormente em 2.3, onde $T_i \rightarrow T_j$ a d_{ij} , que é a distância entre as sedes dos times i e j . Sendo assim, o exemplo do custo do time 3 na representação da tabela 2.1 ficaria assim:

$$\text{custo}(3) = d_{34} + d_{43} + d_{33} + d_{31} + d_{13} + d_{32} + d_{23} + d_{36} + d_{65} + d_{53} + d_{33}$$

3 Técnicas Heurísticas Abordadas

3.1 Métodos construtivos

Para auxiliar as meta-heurísticas e obter melhores resultados, é bastante interessante obter um ponto de partida, que é uma solução inicial factível para o problema. Então, os métodos de construção para este trabalho obedecem a todas as restrições, pois, se não fossem tratadas neste momento, dificilmente os algoritmos de refinamento conseguiriam reverter tais quebras de restrições, se utilizando apenas dos movimentos desenvolvidos.

3.1.1 Método do Polígono

O método do polígono é um método construtivo abordado por (MINE; SOUZA; SILVA, 2006), onde a ideia principal é gerar uma boa solução inicial, em que serão geradas todas as rodadas, de todos os turnos de forma que não quebre nenhuma restrição do problema, para que se possa ter uma melhor performance na fase de refinamento. Este método tem sua realização divididas em três fases.

FASE 1: O Polígono

Criação de jogos de times abstratos onde é utilizado um método, que dá o nome ao método construtivo, *Método do Polígono*. Onde a partir de n = números de times do campeonato são gerados n nós em um polígono regular. Então se retira um destes nós sobrando $n-1$ nós, que passam a ser os vértices deste polígono. O nó que foi retirado fica sendo o pivô, que é associado a um dos nós do polígono e assim todos demais são emparelhados de acordo com o número restante de lados do polígono. Desta forma, rotacionando o polígono e emparelhando os elementos até que o nó pivô tenha se associado a todos o outros nós do polígono. Como representa a figura 1 a seguir:

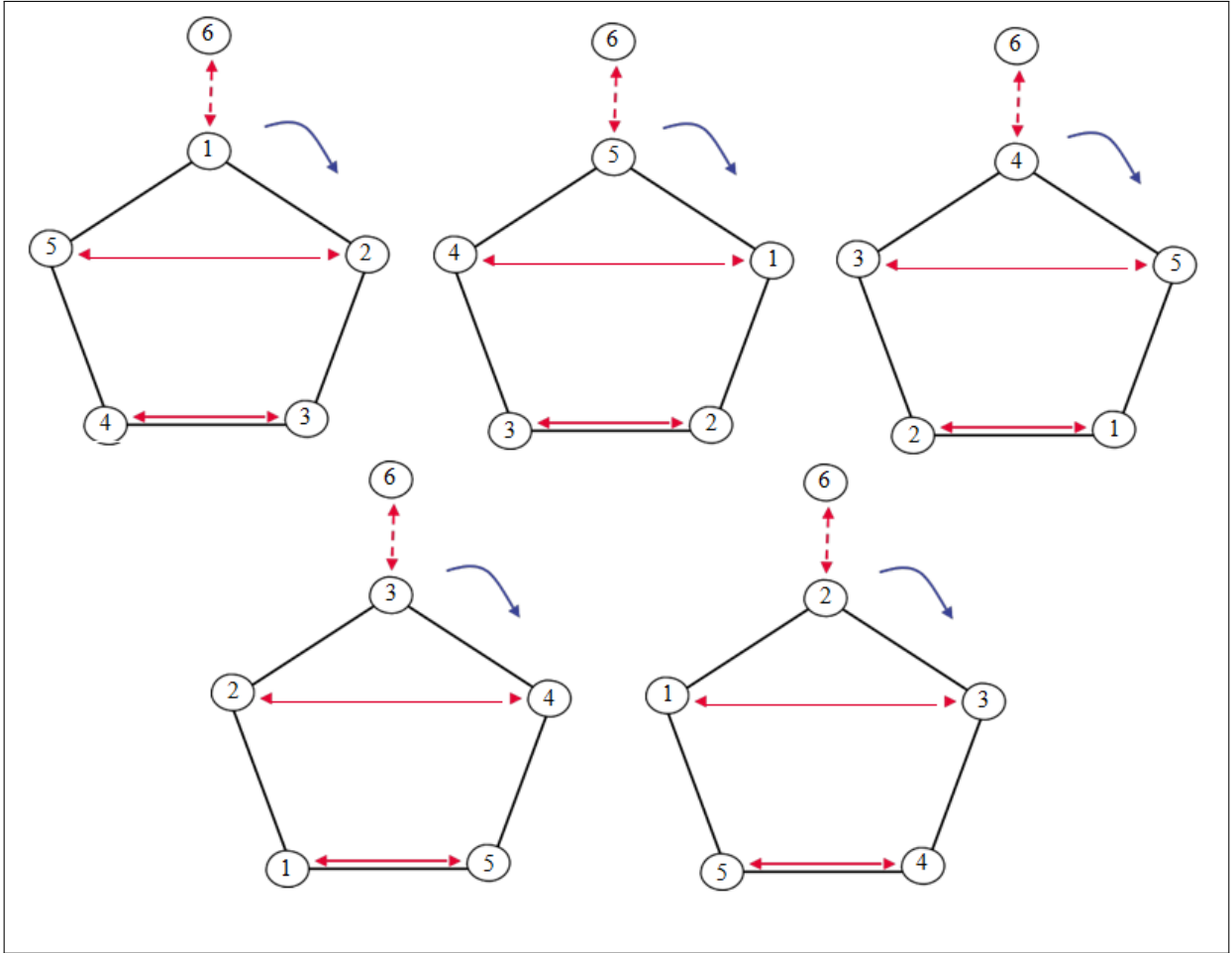


Figura 1 – Representação do método com $n = 6$, na primeira fase.

A associação dos jogos entre os times a partir do método representado na figura acima 1 irá gerar a tabela resultante 3.1 , representada a seguir:

Tabela 3.1 – Representação resultante da associação de jogos.

| Rodada/Time | 1 | 2 | 3 | 4 | 5 | 6 |
|---------------|---|---|---|---|---|---|
| 1 e 6 | 6 | 5 | 4 | 3 | 2 | 1 |
| 2 e 7 | 4 | 3 | 2 | 1 | 6 | 5 |
| 3 e 8 | 2 | 1 | 5 | 6 | 3 | 4 |
| 4 e 9 | 5 | 4 | 6 | 2 | 1 | 3 |
| 5 e 10 | 3 | 6 | 1 | 5 | 4 | 2 |

A partir dos resultados obtidos na Tabela 3.1 é montada uma tabela de times consecutivos $C_{i,j}$ de tamanho $|T| \times |T|$, onde $|t|$ é o total de times envolvidos. O cruzamento entre dois determinados times ti e tj em C refere-se a quantidade de oponentes consecutivos que esses dois times possuem entre si. Por exemplo: Pode-se observar os jogos das rodadas da primeira e segunda fases dos times 1 e 3 na Tabela 3.1 comparar os oponentes consecutivos entre si.

Na Rodada 1 o Time 3 jogará com o Time 4 que por sua vez será o adversário do Time 1 na rodada 2, isso é, eles têm o Time 1 como oponente consecutivo. Desta forma

incrementará na Tabela 3.2, nas posições $C_{1,3}$ e $C_{3,1}$ mais 1 oponente consecutivo. Depois continua-se a comparação encontrando em outras rodadas mais dois oponentes consecutivos, os adversários 2 e 5. Vale lembrar que existem as fases de turno e retorno, totalizando 6 componentes consecutivos para os times 1 e 3. O resultado final das comparações pode ser observada na Tabela 3.2 a seguir:

Tabela 3.2 – Matriz de oponentes consecutivos da Tabela 3.1

| Times | 1 | 2 | 3 | 4 | 5 | 6 |
|-------|---|---|---|---|---|---|
| 1 | 0 | 1 | 6 | 5 | 2 | 4 |
| 2 | 1 | 0 | 2 | 5 | 6 | 4 |
| 3 | 6 | 2 | 0 | 2 | 3 | 3 |
| 4 | 5 | 5 | 2 | 0 | 2 | 4 |
| 5 | 2 | 6 | 3 | 2 | 0 | 3 |
| 6 | 4 | 4 | 3 | 4 | 3 | 0 |

FASE 2: Associação Times Reais \times Fictícios

Considerando a matriz de distâncias entre os times representado em 3.3.

Tabela 3.3 – Matriz de distâncias entre os times

| Times | 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 |

Nesta Fase os Times Reais são associados aos Times Abstratos, onde são geradas duas estruturas. Uma estrutura é formada de pares de times (x, y) e a respectiva distância entre suas sedes, e relacionado as distâncias é ordenada em ordem crescente, representado na Tabela 3.4.

Na outra estrutura, de times abstratos, é ordenado de forma decrescente levando em consideração o número de oponentes consecutivos entre um par de times (x, y) , representado na Tabela 3.5. Após as duas estruturas prontas, associa-se os pares de times reais com menor distância entre si com os pares de times da tabela de abstratos com maior número de oponentes consecutivos. Após esta associação é gerada a solução final de confrontos com os times reais.

Tabela 3.4 – Times Reais

| (x, y) distância | | |
|--------------------|-------|------|
| 1° | (2,3) | 80 |
| 2° | (3,6) | 257 |
| 3° | (2,6) | 315 |
| 4° | (2,4) | 337 |
| 5° | (3,4) | 380 |
| 6° | (4,6) | 408 |
| 7° | (1,6) | 521 |
| 8° | (1,5) | 605 |
| 9° | (1,3) | 665 |
| 10° | (1,2) | 745 |
| 11° | (1,4) | 929 |
| 12° | (5,6) | 1010 |
| 13° | (3,5) | 1020 |
| 14° | (2,5) | 1090 |
| 15° | (4,5) | 1380 |

Tabela 3.5 – Times Abstratos

| (x, y) consecutivos | | | | | |
|-----------------------|-------|---|-----|-------|---|
| 1° | (1,3) | 6 | 16° | (5,3) | 3 |
| 2° | (3,1) | 6 | 17° | (3,6) | 3 |
| 3° | (2,5) | 6 | 18° | (6,3) | 3 |
| 4° | (5,2) | 6 | 19° | (5,6) | 3 |
| 5° | (1,4) | 5 | 20° | (6,5) | 3 |
| 6° | (4,1) | 5 | 21° | (1,5) | 2 |
| 7° | (2,4) | 5 | 22° | (5,1) | 2 |
| 8° | (4,2) | 5 | 23° | (2,3) | 2 |
| 9° | (1,6) | 4 | 24° | (3,2) | 2 |
| 10° | (6,1) | 4 | 25° | (3,4) | 2 |
| 11° | (2,6) | 4 | 26° | (4,3) | 2 |
| 12° | (6,2) | 4 | 27° | (4,5) | 2 |
| 13° | (4,6) | 4 | 28° | (5,4) | 2 |
| 14° | (6,4) | 4 | 29° | (1,2) | 1 |
| 15° | (3,5) | 3 | 30° | (2,1) | 1 |

Tabelas de associação Times Reais \times Times Abstratos

FASE 3: Definição dos Locais do jogos

Nesta Fase estabelece o local de realização dos jogos, ou seja, aonde deverá ocorrer cada confronto obedecendo os critérios de restrições do problema. A Tabela 3.6 mostra o exemplo do resultado final da fase de construção que terminou nesta fase.

Tabela 3.6 – Resultado final do 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 |

3.1.2 Backtracking

Este é um método construtivo onde uma solução é gerada passo a passo, elemento por elemento, de forma a gerar uma solução factível para servir de pontapé inicial para

os algoritmos de refinamento. Proposto por Silva G.; Mine (2005) o *Backtracking* é um procedimento onde divide-se em duas fases de construção.

Na primeira fase, os times são alocados de maneira aleatória obedecendo as regras 4 e 5, descritas no Capítulo 2. São geradas as rodadas através de um processo de *backtracking*. Segundo Silva G.; Mine (2005), este processo é iniciado escolhendo aleatoriamente o oponente para o primeiro time. Obedecendo a regra da competição em que um time não pode jogar duas vezes na mesma rodada, estes dois times que já foram alocados deixam de ser oponentes possíveis para outros times do campeonato nesta rodada. Desta forma, é realizado o preenchimento dos próximos confrontos das rodadas.

O processo de *backtracking* ocorrerá quando tentar alocar um jogo para um time, mas não for possível por não existir nenhum oponente possível para ele. Assim terá que desalocar o jogo do time anterior e desconsiderar seu oponente desalocado para esse time. Caso persista a falta de oponentes o processo de *backtracking* continua. Este processo pode persistir até que se atinja a primeira rodada. Este método de alocação dos times terminará quando terminarem de alocar todas as rodadas do campeonato.

Na segunda fase, a configuração dos mandos de campo também é gerada de forma aleatória, obedecendo a restrição 1, também descrita no capítulo 2. A geração da solução inicial foi realizada apenas para o primeiro turno e duplicada para o segundo turno, apenas invertendo os mandos de campo dos jogos, desta forma estará obedecendo as regras 2 e 3, obtendo desta forma uma solução factível.

3.2 Estruturas de Vizinhaça

São considerados neste trabalho seis tipos diferentes de movimentos de vizinhaça, sendo que 3 deles *swap rounds*, *swap teams* e *swap homes* foram adotadas por Anagnostopoulos et al. (2006), além dos movimentos de *Partial swap rounds*, *Partial swap teams* e *swap homes All*. Esses movimentos de vizinhaça serão utilizados para os algoritmos de refinamento que irão refinar a solução inicial.

Como base para os movimentos de vizinhaça que serão abordados a seguir, considere a matriz de solução representada na Tabela 3.7.

Tabela 3.7 – Exemplo de solução Anagnostopoulos et al. (2006)

| 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 |

3.2.1 Swap Teams

Este movimento constitui na troca de todos os jogos entre duas determinadas equipes T_i e T_j , com exceção da rodada em que eles se confrontam. Considere os movimentos de troca entre os times 2 e 5, mostrados na Tabela 3.8 a seguir:

Tabela 3.8 – Movimento *Swap Teams*(T_2, T_5)

| 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 |

Considerando as duas linhas escolhidas, são realizadas as trocas de seus jogos como apresentada na Tabela 3.9 a seguir:

Tabela 3.9 – Movimento *Swap Teams*(T_2, T_5)

| 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 | - 3 | + 6 | + 4 | + 1 | - 6 | - 4 | - 1 | + 3 | + 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 | + 1 | - 3 | - 6 | + 4 | + 3 | + 6 | - 4 | - 1 | - 2 |
| 6 | - 1 | - 4 | - 5 | + 2 | - 3 | + 5 | - 2 | + 3 | + 4 | + 1 |

Após a troca das linhas referente aos jogos dos dois times, é necessário realizar a normalização da tabela, assim gera o resultado do movimento *Swap Teams*, representado na Tabela 3.10 a seguir:

Tabela 3.10 – Resultado do movimento *Swap Teams*(T_2, T_5)

| 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 |

3.2.2 Swap Rounds

O movimento de *Swap Rounds* é um movimento muito simples, que consiste na troca de todos os jogos de uma determinada rodada R_i com todos jogos de uma rodada R_j . O presente trabalho considera a troca apenas dos times envolvidos nos jogos mantendo assim o mando de campo. Considere os movimentos de troca entre as rodadas 3 e 10 mostrados na Tabela 3.11 a seguir:

Tabela 3.11 – Colunas escolhidas para o movimento *Swap Rounds*(R_3, R_{10})

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

Após serem trocadas as rodadas entre os times, o resultado está apresentado na Tabela 3.12, a seguir:

Tabela 3.12 – Resultado do movimento *Swap Rounds*(R_3, R_{10})

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

3.2.3 Swap Homes

O movimento de *Swap Homes* consiste em trocar o mando de campo dos confrontos entre dois determinados times T_i e T_j . Neste caso considere que os times escolhidos foram 1 e 4, destacados na Tabela 3.13 a seguir:

Tabela 3.13 – Movimento *Swap Homes*(T_1, T_4)

| 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 |

O resultado do movimento está apresentado na Tabela 3.14, a seguir:

Tabela 3.14 – Resultado do movimento *Swap Homes*(T_1, T_4)

| 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 |

3.2.4 Swap Homes All

Este movimento consiste em inverter os mandos de campo de todos os times do campeonato. Considerando a tabela de jogos 3.7, o resultado do movimento será a Tabela 3.15, exibida a seguir:

Tabela 3.15 – Movimento *Swap Homes All*

| 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 |

3.2.5 Partial Swap Rounds

O movimento *Partial Swap Rounds* tem como ideia principal a troca entre rodadas assim como o *Swap Rounds*, porém esta troca é parcial. Consiste assim em trocar apenas

um determinado jogo de um time T_k das rodadas R_i e R_j . Na Tabela 3.16 apresenta a troca entre rodadas 2 e 9 para o time 2.

Tabela 3.16 – Representação dos elementos para *Partial Swap Rounds* (R_2, R_9, T_2)

| 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 |

O movimento gera inconsistência na tabela de jogos, como pode-se ver na Tabela 3.17 o time 2 na segunda rodada agora joga com o time 6 na casa do adversário, mas o time 6 não aparece jogando com o time 2, mas sim jogando com o time 4. O mesmo acontece nona rodada.

Tabela 3.17 – Movimento *Partial Swap Rounds* (R_2, R_9, T_2)

| 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 | - 6 | - 1 | - 5 | + 4 | + 3 | + 6 | - 4 | + 1 | - 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 |

Esta normalização é realizada de forma determinística, reproduzindo o resultado final apresentado na Tabela 3.18 a seguir:

Tabela 3.18 – Resultado Final do Movimento *Partial Swap Rounds* (R_2, R_9, T_2)

| 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 |

3.2.6 Partial Swap Teams

O movimento *Partial Swap Teams* tem a mesma ideia do anterior, onde parte de realizar parcialmente o movimento de *Swap Teams*. Este movimento consiste em realizar a troca entre jogos de dois determinados times i e j em uma determinada rodada R_k . Na Tabela 3.19 apresenta a troca dos adversários entre os times 2 e 4 na a rodada 9.

Tabela 3.19 – Movimento *Partial Swap Teams* (T_2, T_4, R_9)

| 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 |

Com a movimentação apenas destes dois elementos a tabela a ser gerada será inconsistente, necessitando assim que sejam realizados movimentos determinísticos para corrigir esta inconsistência. O resultado final destes movimentos podem ser observados na Tabela 3.20, a seguir:

Tabela 3.20 – Resultado Final do Movimento *Partial Swap Teams* (T_2, T_4, R_9)

| 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 |

3.3 Metaheurísticas

3.3.1 Iterated Local Search

O método *Iterated Local Search*(ILS) foi proposto por Lourenço *et al.* (LOURENCO; MARTIN; STUTZLE, 2001) e consiste em um processo de busca iterativo, onde a ideia é partir de uma solução inicial S_0 qualquer factível para o problema e realizar melhorias de acordo com conhecimentos adquiridos em análise de escolhas anteriores. Neste tópico será apresentado com mais detalhes o funcionamento do algoritmo.

Segundo Kim (2012) os principais componentes do ILS são:

- Solucao Inicial;
- Busca Local;
- Perturbação;
- Critério de Aceitação.

Kim (2012) explica que a busca local é a parte mais importante do algoritmo, pois é ela que conduz o algoritmo na procura de ótimos locais, enquanto os outros componentes como perturbação e o critério de aceitação são responsáveis apenas para fugir e evitar que fique preso em ótimos locais, e desta forma equilibrar a busca entre intensificação e diversificação.

A intensificação de busca se dará pela busca local juntamente com pequenas perturbações, possibilitando encontrar melhores soluções em uma pequena área de busca. A diversificação de busca dará pelas grandes perturbações que possibilitarão, assim, caminhar pelo espaço de soluções. Desta forma é necessário saber a forma correta de utilizar a perturbação, pois, ela deve ser fraca para que não se percam as características do ótimo local e forte para conseguir escapar de um ótimo local e assim explorar outras regiões no espaço de busca.

O Algoritmo 3.1 representado a seguir possui uma solução inicial s_0 gerada na linha 2, para que, na linha 3, seja realizada uma busca local para formar um bom ponto de partida que irá iniciar o algoritmo. Após esse início do algoritmo, ele entra no laço para a busca. É realizada uma perturbação em S , onde, s' serve somente para receber o resultado vindo de *Perturbação* e servir como parâmetro para a *BuscaLocal*. A solução s'' será gerada para ser a melhor solução a ser retornada da busca local. O método de *CritérioAceitação* decidirá qual a melhor solução e que servirá como parâmetro para a próxima perturbação.

Para auxiliar na *Perturbação* e *CritérioAceitação* tem-se um elemento chamado *histórico* que é um elemento importante, pois nele que são armazenadas as informações adquiridas até o momento, como por exemplo, uma lista de soluções geradas anteriormente a fim de evitar soluções repetidas e também saber a hora de intensificar ou diversificar a busca.

Algoritmo 3.1 ILS

```

1: função ILS
2:  $S_0 \leftarrow \text{GeraSoluçãoInicial}()$ 
3:  $S \leftarrow \text{BuscaLocal}(S_0)$ 
4:   enquanto (os critérios de parada não estiverem satisfeitos) faça
5:      $S' \leftarrow \text{Perturbação}(\text{historico}, S)$ 
6:      $S'' \leftarrow \text{BuscaLocal}(S')$ 
7:      $S \leftarrow \text{CritérioAceitação}(S, S'', \text{historico})$ 
8:   fim enquanto
9: fim função

```

3.3.2 Late Acceptance Hill Climbing

Esta técnica *Late Acceptance Hill Climbing* (LAHC), que também foi utilizada para o Problema da Programação de Horários Escolares Fonseca, Santos e M. (2013), foi recentemente proposto por Burke e Bykov (2008). Se trata de um processo adaptado a partir do método clássico de *Hill Climbing* Russell e Norvig (2002). A diferença está em não aceitar uma solução candidata imediatamente, mas sim, compará-la a uma determinada solução i , vinda de uma lista de funções objetivo de soluções geradas anteriormente a fim de aceitá-la ou não.

Segundo Burke e Bykov (2008), esse método foi criado com três objetivos:

- Ser uma busca local que não exige um procedimento de resfriamento artificial;
- Usar de forma eficiente as informações coletadas em iterações anteriores;
- Aplicar um mecanismo simples de aceite de soluções candidatas.

No Algoritmo 3.2 representado a seguir, cria uma lista f'_k , $\forall k \in \{0, \dots, l-1\}$, onde l é o tamanho da lista, para armazenar o valor de função de avaliação (*fitness*) das soluções de iterações anteriores que foram aceitas. Inicialmente os campos da lista são preenchidos com o valor *fitness* da solução inicial.

A cada iteração uma solução s' é gerada, que será candidata a ser aceita caso seu *fitness* seja menor ao *fitness* da posição $i \bmod l$, em f'_k , e ela será a solução de ponto de partida para gerar uma nova solução s' . Caso seu *fitness* seja menor do que o *fitness* da melhor solução gerada até o momento, a melhor solução é atualizada.

Algoritmo 3.2 Late Acceptance Hill Climbing

```

1: função LAHC( $s, l$ )
2:    $f'_k \leftarrow f(s) \forall k \in \{0, \dots, l-1\}$ ;
3:    $s^* \leftarrow s$ ;
4:    $i \leftarrow 0$ ;
5:   enquanto ( $elapsedTime < timeout$ ) faça
6:     Gere um Vizinho Qualquer  $s' \in N(s)$ ;
7:      $v = i \bmod tamanhoLista$ ;
8:     se ( $f(s') < f'_v$ ) então
9:        $s \leftarrow s'$ 
10:    se ( $f(s) < f(s^*)$ ) então
11:       $s^* \leftarrow s$ ;
12:    fim se
13:  fim se
14:   $f'_v \leftarrow f(s)$ ;
15:   $i \leftarrow i + 1$ ;
16: fim enquanto
17:  Retorne  $s$ ;
18: fim função

```

3.3.3 Simulated Annealing

O método *Simulated Annealing* (SA) foi proposto originalmente por Kirkpatrick e Vecchi. (1983). Este método, parte da analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos. Começa a partir de uma solução inicial aleatória s_0 e uma temperatura inicial alta T_0 , onde essa temperatura é reduzida gradativamente.

Neste trabalho, por se tratar de um problema de minimização, o movimento de melhora é dado quando $\Delta < 0$. Desta forma a solução s' gerada em movimento aleatório passa ser a solução corrente s para a próxima geração. Caso $\Delta \geq 0$, seu movimento é de piora, assim haverá o critério de aceitação pela probabilidade $e^{-\Delta/T}$, onde T é a temperatura corrente, esta probabilidade controla a aceitação dos movimentos de piora.

Quanto maior a temperatura, maior é a agitação das moléculas. Permitindo assim, que durante a fase inicial do algoritmo tenha uma maior probabilidade de que uma solução vizinha s' de piora seja aceita. Desta forma evita que o algoritmo fique preso em ótimos locais, possibilitando a exploração de outras regiões do espaço de soluções. A cada temperatura T é realizado um número fixo de iterações, denotado por SAm_{ax} , que representa a quantidade necessária de iterações para que o sistema atinja equilíbrio térmico na temperatura corrente. Para o resfriamento da temperatura T , é definido uma taxa fixa de resfriamento, denominado por α , Fonseca (2013). O Algoritmo 3.3, a seguir mostra o método *Simulated Annealing* básico para o problema.

Algoritmo 3.3 Simulated Annealing

```

1: função SIMULATED ANNEALING ( $f(\cdot)$ ,  $N(\cdot)$ ,  $\alpha$ ,  $SAmax$ ,  $T_0$ ,  $s$ )
2:    $s^* \leftarrow s$ 
3:    $IterT \leftarrow 0$ 
4:    $T \leftarrow T_0$ 
5:   enquanto ( $T > 0$ ) faça
6:     enquanto ( $IterT < SAmax$ ) faça
7:        $IterT \leftarrow IterT + 1$ ;
8:       Gere um Vizinho Qualquer  $s' \in N(s)$ ;
9:        $\Delta = f(s') - f(s)$ ;
10:      se ( $\Delta < 0$ ) então
11:         $s \leftarrow s'$ 
12:        se ( $f(s') < f(s^*)$ ) então
13:           $s^* \leftarrow s'$ ;
14:        fim se
15:      senão
16:        Tome  $x \in [0, 1]$ ;
17:        se ( $x < e^{-\Delta/T}$ ) então
18:           $s \leftarrow s'$ ;
19:        fim se
20:      fim se
21:    fim enquanto
22:     $T \leftarrow \alpha \times T$ ;
23:     $IterT \leftarrow 0$ ;
24:  fim enquanto
25:   $s \leftarrow s^*$ ;
26:  Retorne  $s$ ;
27: fim função

```

4 Experimentos Computacionais

Neste capítulo serão apresentados os resultados computacionais para o TTP, obtidos através da aplicação das meta-heurísticas *Simulated Annealing*, *Iterated Local Search* e *Late Acceptance Hill Climbing*, que foram abordadas no capítulo anterior. Para gerar uma solução inicial dos experimentos, foi utilizado o método *Backtracking*, para formar o ponto de partida factível. Serão apresentados os melhores resultados, a média e desvio padrão para 10 execuções realizadas para cada configuração.

4.1 Ambiente de Testes

O experimentos foram realizados em um notebook Intel(R) Core i7-2860QM 2.5 GHz, com 8 GBytes de Memória RAM DDR3, sobre o sistema operacional Microsoft Windows 7 Home Premium de 64Bits. Os algoritmos foram desenvolvidos na linguagem Java na versão 1.7, usando o ambiente de desenvolvimento Eclipse Indigo.

4.2 Caracterização da Instâncias

Para realizar os experimentos foram usadas instâncias da família NL, que foram disponibilizadas em Benchmark (2014) por Trick, onde possui as instâncias para este e outros campeonatos. Segundo Kim (2012), Trick quem originalmente propôs o *Traveling Tournament Problem* e quem pessoalmente administra o site com as instâncias e mantém uma competição com os melhores resultados obtidos para cada uma destas instâncias.

Um dos primeiros exemplos de instâncias publicados no site de Trick Benchmark (2014) são da família dos NL-x que utiliza a base de dados reais da Liga Nacional de Beisebol, onde x é o número de equipes. Para os experimentos computacionais desta família, foram utilizadas seis tamanhos sendo: *NL-6*, *NL-8*, *NL-10*, *NL-12*, *NL-14* e *NL-16*.

4.3 Configuração dos parâmetros

Para a realização dos experimentos de forma a obter uma melhor comparação entre as heurísticas, foi definido o *timeout* de 900 segundos (15 minutos). Todos os resultados foram validados pelo validador de soluções de Luca Di Gaspero e Andrea Schaerf que pode ser acessado em <http://tabu.diegm.uniud.it/ttp/>.

A execução das vizinhanças foi realizado de maneira aleatória, de forma que foi distribuída a mesma probabilidade de acontecimento para cada uma. Após testes realizados pode-se observar a eficiência que os algoritmos possuíam de acordo com os parâmetros passados para suas execuções. Então, de forma empírica foram estabelecidos os parâmetros

específicos para cada meta-heurística, como pode ser observado nas tabelas 4.1 4.2 4.3 a seguir.

A Tabela 4.1 apresenta os parâmetros utilizados para o Algoritmo *Simulated Annealing*. Sendo que a primeira coluna apresenta o tamanho das instâncias, a segunda coluna apresenta SA_{max} que é o número máximo de iterações para cada temperatura e a terceira coluna apresenta a sua taxa de resfriamento, representado por α .

Foi considerado na execução do *Simulated Annealing* a temperatura inicial de 1 e a temperatura final de 0.001. Foi utilizado $NL10$ como base para encontrar o melhor valor de SA_{max} para o algoritmo, desta forma para definir SA_{max} para os demais tamanhos foi dada pela fórmula $SA_{max} = NLx * (\frac{SA_{max}}{10})$. Para este caso foi escolhido o parâmetro de $SA_{max} = 25.000$.

Tabela 4.1 – Parâmetros para o SA

| Simulated Annealing | | |
|---------------------|-------------------|----------|
| Instância | SA _{max} | α |
| NL6 | 15000 | 0.96 |
| NL8 | 20000 | 0.96 |
| NL10 | 25000 | 0.96 |
| NL12 | 30000 | 0.96 |
| NL14 | 35000 | 0.96 |
| NL16 | 40000 | 0.96 |

A Tabela 4.2 apresenta os parâmetros utilizados para o Algoritmo *Late Acceptance Hill Climbing*. Sendo que a primeira coluna apresenta o tamanho das instâncias, a segunda coluna apresenta o tamanho da lista de elementos a serem mantidos para comparação durante a execução e a terceira coluna apresenta o número de iterações máximo sem melhora.

Para a escolha do tamanho da lista, partiu através da fórmula $totalLista = totalLista * NLx$, onde $totalLista$ recebe ela mesma que está vindo de parâmetro vezes o número de times envolvidos na competição. Para este caso foi escolhido o parâmetro de $totalLista = 10.000$.

Tabela 4.2 – Parâmetros para o LAHC

| Late Acceptance Hill Climbing | | |
|-------------------------------|------------------|---------------------|
| Instância | Tamanho da Lista | Número de Iterações |
| NL6 | 120000 | 1000000 |
| NL8 | 160000 | 1000000 |
| NL10 | 200000 | 1000000 |
| NL12 | 240000 | 1000000 |
| NL14 | 280000 | 1000000 |
| NL16 | 320000 | 1000000 |

A Tabela 4.3 apresenta os parâmetros utilizados para o Algoritmo *Iterated Local Search*. Sendo que a primeira coluna apresenta o tamanho das instâncias e a segunda coluna apresenta o número de iterações máximo sem melhora.

Partindo de testes realizados no tamanho *NL10* para encontrar o melhor valor de *ILSmax* para o algoritmo, foi definida a fórmula $ILSmax = NLx * (\frac{ILSmax}{10})$ para distribuir os valores de *ILSmax* para os demais tamanhos. Para este caso foi escolhido o parâmetro de $ILSmax = 1000$

Tabela 4.3 – Parâmetros para o ILS

| Iterated Local Search | |
|-----------------------|---------|
| Instância | ILS Max |
| NL6 | 600 |
| NL8 | 800 |
| NL10 | 1000 |
| NL12 | 1200 |
| NL14 | 1400 |
| NL16 | 1600 |

4.4 Resultados Computacionais

Para todos os resultados apresentados foram considerados dez execuções para cada uma das instâncias de cada meta-heurística. Então, desta forma, as Tabelas 4.4, 4.5 e 4.6 apresentam os resultados por cada uma das meta-heurísticas, onde $f(s^*)$ apresenta a melhor solução obtida para a instância, $f(\bar{s})$ apresenta a média dos resultados e σ o desvio padrão.

Tabela 4.4 – Resultados obtidos pelo algoritmo *Simulated Annealing*

| Simulated Annealing | | | |
|---------------------|----------|--------------|----------|
| Instância | $f(s^*)$ | $f(\bar{s})$ | σ |
| NL6 | 23916 | 23916 | 0,0 |
| NL8 | 40663 | 41047 | 287,5 |
| NL10 | 65231 | 66576 | 670,5 |
| NL12 | 125713 | 128332 | 1280,2 |
| NL14 | 231580 | 234240 | 1431,9 |
| NL16 | 331137 | 333930 | 1867,7 |

Tabela 4.5 – Resultados obtidos pelo algoritmo *Late Acceptance Hill Climbing*

| Late Acceptance Hill Climbing | | | |
|-------------------------------|----------|--------------|----------|
| Instância | $f(s^*)$ | $f(\bar{s})$ | σ |
| NL6 | 23916 | 23994 | 78,1 |
| NL8 | 39721 | 41198 | 1990,3 |
| NL10 | 62397 | 63754 | 2014,7 |
| NL12 | 119649 | 123721 | 3217,3 |
| NL14 | 203933 | 210058 | 4605,1 |
| NL16 | 296759 | 301538 | 2213,5 |

Tabela 4.6 – Resultados obtidos pelo algoritmo *Iterated Local Search*

| Iterated Local Search | | | |
|-----------------------|----------|--------------|----------|
| Instância | $f(s^*)$ | $f(\bar{s})$ | σ |
| NL6 | 23916 | 23916 | 0,0 |
| NL8 | 39776 | 40203 | 469,4 |
| NL10 | 62613 | 64411 | 1165,8 |
| NL12 | 121094 | 125202 | 1858,3 |
| NL14 | 217247 | 225378 | 3701,3 |
| NL16 | 311747 | 320434 | 5464,6 |

Com o intuito de realizar comparações entre os resultados obtidos pelas meta-heurísticas as tabelas 4.7 e 4.8, a seguir, apresentam comparações dos melhores resultados, da média e do desvio padrão, respectivamente, das soluções apresentadas anteriormente. Os melhores resultados estão representados em negrito.

Tabela 4.7 – Comparativo entre as Melhores Soluções

| $f(s^*)$ | | | |
|-----------|--------------|---------------|--------------|
| Instância | SA | LAHC | ILS |
| NL6 | 23916 | 23916 | 23916 |
| NL8 | 40663 | 39721 | 39776 |
| NL10 | 65231 | 62397 | 62613 |
| NL12 | 125713 | 119649 | 121094 |
| NL14 | 231580 | 203933 | 217247 |
| NL16 | 331137 | 296759 | 311747 |

Tabela 4.8 – Comparativo da Média e Desvio Padrão dos resultados

| Instância | SA | | LAHC | | ILS | |
|-----------|--------------|---------------|---------------|----------|--------------|------------|
| | $f(\bar{s})$ | σ | $f(\bar{s})$ | σ | $f(\bar{s})$ | σ |
| NL6 | 23916 | 0,0 | 23994 | 78,1 | 23916 | 0,0 |
| NL8 | 41047 | 287,5 | 41198 | 1990,3 | 40203 | 469,4 |
| NL10 | 66576 | 670,5 | 63754 | 2014,7 | 64411 | 1165,8 |
| NL12 | 128332 | 1280,2 | 123721 | 3217,3 | 125202 | 1858 |
| NL14 | 234240 | 1431,9 | 210058 | 4605,1 | 225378 | 3701,3 |
| NL16 | 333930 | 1867,7 | 301538 | 2213,5 | 320434 | 5464,6 |

De acordo com os resultados apresentados neste capítulo pelas Tabelas 4.4, 4.5 e 4.6 pode-se observar que todas as meta-heurísticas conseguiram obter bons resultados e que atendem às restrições do problema. Ao comparar todos os resultados obtidos neste trabalho, pode-se observar que o algoritmo *Late Acceptance Hill Climbing* obteve os melhores resultados entre as demais meta-heurísticas em todas as instâncias utilizadas.

A Tabela 4.9 mostra uma comparação entre os melhores resultados do *LAHC*, apresentados neste trabalho, com os melhores resultados conhecidos para as instâncias abordadas, disponíveis no site Benchmark (2014) que serviu como referência para as instâncias.

Tabela 4.9 – Comparativo entre melhores soluções conhecidas \times *LAHC*

| Instância | $f(s^*)$ | | |
|-----------|------------------|--------------|---------|
| | Melhor conhecido | LAHC | Gap (%) |
| NL6 | 23916 | 23916 | 0,0 |
| NL8 | 39721 | 39721 | 0,0 |
| NL10 | 59436 | 62397 | 4,9 |
| NL12 | 110729 | 119649 | 8,0 |
| NL14 | 188728 | 203933 | 8,0 |
| NL16 | 261687 | 296759 | 13,4 |

Apesar dos esforços do Algoritmo *LAHC*, com o espaço de tempo 15 minutos, não foi possível bater os melhores resultados do torneio da referência, a *Challenge Traveling Tournament* (BENCHMARK, 2014). Porém, os resultados obtidos foram animadores: os gaps obtidos entre o Algoritmo *LAHC* e a melhor solução conhecida foram baixos, variando de 0% a 13% no maior problema abordado. Além do mais, para a obtenção dos melhores resultados conhecidos não foi considerado tempo limite de execução e não era vetado o uso de supercomputadores, o que faz com que métodos baseados em enumeração de soluções sejam capazes de atingir excelentes resultados: alguns desses resultados foram obtidos dessa forma, tais como os trabalhos de Uthus, Riddle e Guesgen (2009) e de Urrutia, Ribeiro e Melo (2007). No contexto de abordagens heurísticas para o problema se destacaram os trabalhos de Langford (2010) e Hentenryck e Vergados (2006). Outro fator importante é que, os trabalhos detentores dos melhores resultados para esse problema são

fruto de vários anos de pesquisa; assim, estudos futuros sobre novas vizinhanças podem tornar o Algoritmo *LAHC* ainda mais competitivo para esse problema.

5 Considerações Finais

Neste trabalho foram propostas três abordagens de meta-heurísticas, sendo elas a *Iterated Local Search*, *Late Acceptance Hill-Climbing* e *Simulated Annealing* para solucionar o problema da alocação de jogos em um sistema de torneio que utiliza-se *Double Round Robin*, onde as equipes se enfrentam entre si duas vezes com o mando de campo para cada uma delas.

O foco principal deste trabalho foi comparar os resultados entre as meta-heurísticas. Para essas comparações, foram utilizados os times da *Major League Baseball*, dos Estados Unidos, diante das instâncias de 6, 8, 10, 12, 14 e 16 equipes. Para obter a maior eficiência na comparação dos algoritmos desenvolvidos, foi utilizado um padrão no tempo de execução. Como proposta para o mecanismo de perturbação, foram utilizados 6 tipos de movimentos diferentes de vizinhança que eram gerados de maneira aleatória.

Primeiramente, foi desenvolvido o algoritmo da meta-heurística *SA* que para escapar de ótimos locais, ele utiliza da ideia de recozimento e assim ele pode se deslocar a outras áreas de soluções. O algoritmo *LAHC* se assemelha como um método da descida, porém de maneira mais eficiente. Ele mantém uma quantidade determinada de resultados anteriores para comparar um deles com a solução corrente. Desta forma, evita que fique preso em um ótimo local permitindo a busca em outras áreas de soluções, permitindo algumas vezes, movimentos de piora. O algoritmo *ILS* a cada iteração realiza um determinado número de perturbações na solução candidata e a partir disso uma busca local dela, para que assim possa explorar ótimos locais em outras regiões no espaço de busca.

Considerando as configurações do tempo de execução e das vizinhanças envolvidas, os algoritmos conseguiram gerar resultados factíveis, ou seja, atenderam as restrições do problema. O Algoritmo *SA* obteve resultados apenas razoáveis, tendo o pior desempenho entre as 3 abordagens presentes neste trabalho. Já o Algoritmo *ILS* obteve resultados mais satisfatórios em relação ao citado anteriormente. O *LAHC* foi o algoritmo que produziu resultados mais interessantes para o problema. Quando comparados às melhores soluções conhecidas, os resultados do *LAHC* foram animadores: o algoritmo foi capaz de encontrar a melhor solução conhecida para dois dos problemas abordados e de se aproximar das soluções dos demais mesmo sobre um limite de tempo e utilizando computadores de configuração de hardware convencional.

Como trabalhos futuros sugere-se: (i) incorporar novos movimentos de vizinhança aos algoritmos implementados; (ii) estudar a hibridização de meta-heurísticas para a solução do problema como feito com sucesso em Fonseca et al. (2014); e (iii) propor heurísticas de programação matemática ao problema, como em Santos et al. (2012).

Referências

- ANAGNOSTOPOULOS, A. et al. A simulated annealing approach to the traveling tournament problem. 2006.
- BENCHMARK. *Travel Tournament Problem: Description and benchmarks*. 2014. Disponível em: <<http://mat.gsia.cmu.edu/TOURN/>>.
- BURKE, E. K.; BYKOV, Y. The late acceptance hill-climbing heuristic. *Department of Computing Science and Mathematics, University of Stirling*, 2008.
- FONSECA, G. H. G. *Métodos de Busca Heurística para Problemas de Programação de Horários Modelados em XHSTT*. Dissertação (Mestrado) — Universidade Federal de Ouro Preto, 2013.
- FONSECA, G. H. G.; SANTOS, H. G.; M., T. T. A. Late acceptance hill-climbing applied to the high school timetabling problem. *IX Multidisciplinary International Scheduling Conference: Theory and Applications*, 2013.
- FONSECA, G. H. G. da et al. Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, Springer US, p. 1–21, 2014.
- GOERLER, A.; SCHULTE, F.; VOß, S. An application of late acceptance hill-climbing to the traveling purchaser problem. 2013.
- HENTENRYCK, P. V.; VERGADOS, Y. Traveling tournament scheduling: A systematic evaluation of simulated annealing. In: BECK, J.; SMITH, B. (Ed.). *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3990). p. 228–243. ISBN 978-3-540-34306-6. Disponível em: <http://dx.doi.org/10.1007/11757375_19>.
- KIM, B. M. *Iterated Local Search for the Traveling Tournament Problem*. Tese (Doutorado) — Technische Universität Wien, 2012.
- KIRKPATRICK, D. G. S.; VECCHI, M. Optimization by simulated annealing. *Science*, n. 220, p. 671, 1983.
- LANGFORD, G. An improved neighbourhood for the traveling tournament problem. *arXiv preprint arXiv:1007.0501*, 2010.
- LOURENCO, H. R.; MARTIN, O. C.; STUTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Social Science Research Network*. Kluwer Academic Publishers, 2001. v. 57, n. 513, p. 49. Disponível em: <<http://arxiv.org/abs/math/0102188>>.
- MINE, M. T.; SOUZA, M. J. F.; SILVA, G. P. Programação de jogos de competições esportivas: Uma abordagem heurística – Parte II. 2006.
- RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002. Disponível em: <<http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0137903952>>.

SANTOS, H. G. et al. Towards generic mip solvers for rostering and timetabling. In: *Proceedings of the Fourth International Workshop on Model-based Metaheuristics*. [S.l.: s.n.], 2012. p. 12.

SILVA G.; MINE, M. S. M. Um método de geração de uma solução inicial baseado em backtracking para o problema de programação de jogos do campeonato brasileiro de futebol. 2005.

TIERNEY, K. Late acceptance hill-climbing applied for the liner shipping fleet repositioning problem. 2013.

URRUTIA, S.; RIBEIRO, C. C.; MELO, R. A. A new lower bound to the traveling tournament problem. *Proceedings of the IEEE symposium on computational intelligence in scheduling*, p. 15–18, 2007.

UTHUS, D. C.; RIDDLE, P. J.; GUESGEN, H. W. Dfs* and the traveling tournament problem. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. [S.l.]: Springer, 2009. p. 279–293.